Low-Power Scientific Computing

Ganesh Dasika Ankit Sethia Trevor Mudge Scott Mahlke

Advanced Computer Architecture Laboratory
University of Michigan
Ann Arbor, MI
{gdasika, asethia, tnm, mahlke}@umich.edu

Introduction: Scientists and mathematicians are increasingly realizing the computational benefits of using modern, multi-core architectures. In response to this, manufacturers of traditional desktop graphics-processing units (GPUs) have evolved their architectures to create desktop and server GPGPUs (General Purpose Graphics Processing Units). These GPGPUs are quickly becoming the platform of choice for many high-performance, highly parallel applications. GPGPUs are also commodity hardware products commonly available in many desktop and laptop computers, making them rather inexpensive. The tools to program them are easily available as well; Nvidia's Compute Unified Device Architecture (CUDA) package, for example, provides a small set of extensions to the C programming language, allowing for straightforward implementation of parallel algorithms on GPGPUs. Individual cores in Intel's up-and-coming Larrabee processor implement the ubiquitous x86 ISA, allowing users to use a host of already-existing development tools to port their applications to it. Server products like the Nvidia Tesla S1070 with even more compute power are also available.

Several applications, from a wide variety of domains, including medical imaging, electronic design automation, physics simulations, and stock pricing models, observe remarkable speed-ups on GPUs – at times, over 300X. Based on these dramatic performance increases, GPGPUs seem like an ideal computing substrate for high-performance, scientific computing. However, there are two major problems with GPGPUs – power consumption and an unbalanced ratio of compute ability to memory bandwidth.

The disadvantages of GPGPUs: As shown in Figure 1, the GTX280 consumes over 200W of power resulting in low power-efficiency at peak performance. Other solutions like the Tesla S1070 can consume over 1kW. Other general-purpose solutions from IBM, Intel and ARM, while consuming significantly less power, have similar, or worse, performance-per-Watt efficiency.

Though power is not necessarily a significant drawback for video game graphics acceleration, requiring powerful cooling systems is a significant impediment for more portable platforms. Some of the applications that are sped-up with GPUs use them to accelerate underlying algorithms that are often deployed in systems where portability or power consumptions is a critical issue. For instance, polynomial multiplication is used in very advanced cryptographic systems,

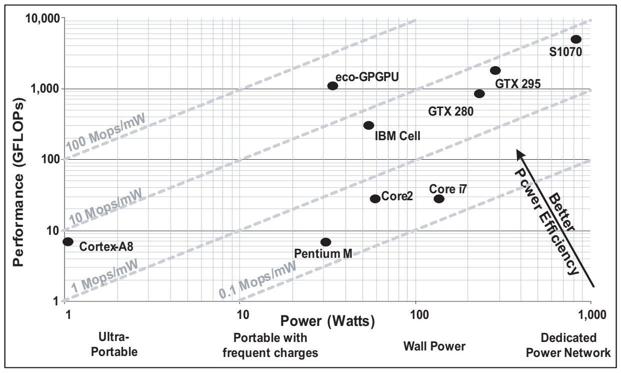

Figure 1: Peak performance and power characteristics of several high-performance commercial processors and GPGPUs are provided: ARM Cortex-A8, Intel Pentium M, Intel Core 2, Intel i7, IBM Cell, Nvidia GTX 280, Nvidia GTX 295, and Nvidia Tesla S1070. eco-GPGPU is the architecture template proposed in this work.

real-time FFT solving is required for complex GPS receivers, and low-density parity-check error correcting codes are used in WiMAX and WiFi. Monte Carlo Recycling algorithms for computational finance, are often deployed in dense urban areas like Manhattan where, while portability is not an issue, power and cooling certainly is an important cost concern.

Further, There is a large disparity between GPGPUs' compute abilities and the amount of memory bandwidth they support. For instance, the latest generation of Nvidia GPGPUs, the GTX285, has a peak performance of 1,063 GFLOPs but can transfer only up to 159 GB/s of data – an average of 0.15 bytes of data per FP operation. For graphics applications, such a ratio is sustainable, but several of the applications that are using GPUs in this manner have a much higher bandwidth requirement. Portable medical imaging applications, for instance, require on the order of 1 byte/instruction.

eco-GPGPU: To create an architecture best suited for scientific computation, four different benchmarks were initially analyzed: `acfdtd2d`, which implements the electrodynamics modeling method finite-difference time-domain; `sde`, a stochastic differential equation solver; `volatility`, which measures the volatility of a common stock market index; and `mbir`, a model-based iterative reconstruction technique for CT scanners using ultra-low doses of x-ray radiation. These

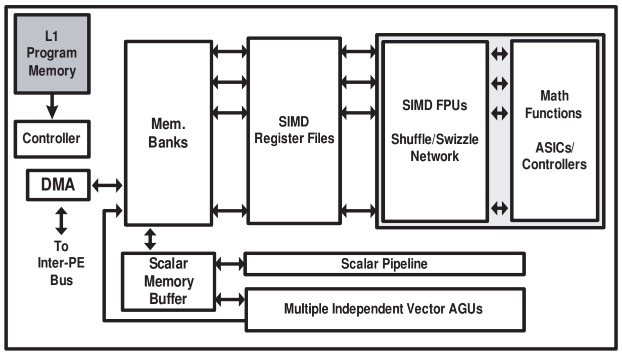

Figure 2: eco-GPGPU processor PE architecture template.

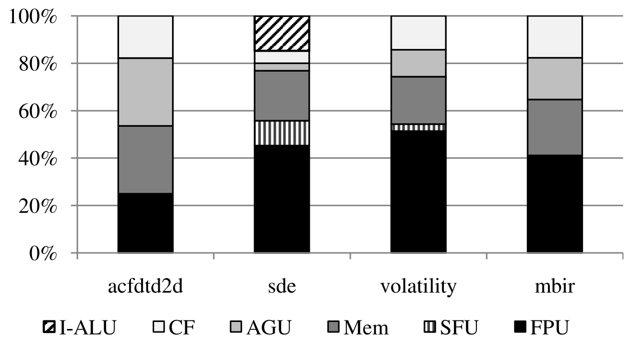

Figure 3: Instruction type breakdown showing the % of instructions used for integer ALU (I-ALU) control-flow (CF), address generation (AGU), loads and stores (Mem), special math library functions (SFU), and general floating-point computation (FPU). Note the relatively large % of memory instructions in all of the benchmarks.

benchmarks were chosen as they are representative of programs that use GPGPUs to improve performance.

Figure 3 characterizes the type and frequency of instructions in each benchmark, showing the percentage of integer arithmetic, control-flow, address-generation, load/store, special-function floating-point library functions (e.g. logarithms, trigonometry) and FP arithmetic, respectively. As can be seen from this graph, the computation in these benchmarks is predominantly FP arithmetic, though there are some integer operations as well. In most cases, the control-flow instructions are those to check the terminating condition of the loop. Benchmarks with a high % of control-flow instructions have if-else conditions within the loop kernel, which can be incorporated into arithmetic instructions using techniques such as predication. One very important aspect of these applications that is not shown in this figure is that they are comprised primarily of fully-parallelizable, do-all loops. They are, therefore, quite easily modified to run on wide, SIMD-parallel architectures.

Based on our analysis of these applications, we designed the eco-GPGPU processor to effectively perform low-power, scientific computing by utilizing the following architectural features:

1. A wide SIMD machine to effectively exploit the parallelism available in most scientific applications.

2. A floating-point multiply-accumulate execution unit to efficiently support commonly occurring computation subgraphs while minimizing accesses to the large vector register file.

3. An intricate shuffle and swizzle network for efficient inter-lane operand-passing.

4. Special function units for infrequently-occurring, yet necessary, math library functions such as sine, cosine and divide.

Overall, eco-GPGPU achieves 1,024 GFLOPs while consuming 36W of power on a 65nm process.

However, provided the same off-chip memory bandwidth and an off-chip memory latency of several hundred cycles as that of the Nvidia GPGPUs, eco-GPGPU will, too, offer only a fraction of its peak performance, leading to power wasted in stall cycles. Therefore, while eco-GPGPU's power consumption is still less than that of GPGPUs, computation ability is still potentially being wasted.

There are a few different alternatives to mitigate problems with off-chip memory latency. Large caches offer a dense, lower-power alternative to register contexts, either to store data required in future iterations of the program kernel or in order to cache infrequently accessed register thread contexts. Even though modern GPUs have very large caches, these are often in the form of graphics-specific texture caches, and not easily used for other applications. Further, our analysis indicates that many scientific computing benchmarks access data in a streaming nature – loaded values are located in contiguous memory locations. Computed results are also stored in contiguous locations and are rarely ever re-used. This allows for creating a memory system that can easily predict what data is required and at what time and can operate independently of address-generation and explicit load/store instructions issued by the processor core.

Insufficient off-chip bandwidth can be addressed in two ways. 3D-stacked DRAM, especially in newer processes that support several layers of DRAM, allow for placing gigabytes of data directly above the processor, with a through-silicon bandwidth in the terabytes/second regime. Latency is mitigated using this technique as well, as transfer times between the 3D-stacked DRAM and the processor layer is usually only tens of clock cycles. Data compression, based on the nature of the application data, is another viable alternative. Our experiments have shown that scientific benchmarks that process sparse matrices, especially medical image reconstruction, can have their data compressed to 10% of their original size, effectively increasing the off-chip bandwidth by 10X.

Conclusion: Based on our analysis and initial architecture, eco-GPGPU is a viable alternative to modern-day GPGPUs as a platform for scientific computing. Further, the same level of performance as current GPGPUs can be delivered at significantly reduced power. With an ever-increasing emphasis on reducing power density in computers and also increasing their portability, studying and building processors like eco-GPGPU should be an important aspect of future computer architecture research.