Power Aware Computing on GPUs

Kiran Kasichayanula, Dan Terpstra, Piotr Luszczek, Stan Tomov, Shirley Moore
Innovative Computing Laboratory
University of Tennessee Knoxville
kkasicha, terpstra, luszczek, tomov, shirley @eecs.utk.edu

Gregory D. Peterson
EECS, University of Tennessee Knoxville
gdp@eecs.utk.edu

***Abstract*—Energy and power density concerns in modern processors have led to significant computer architecture research efforts in power-aware and temperature-aware computing. With power dissipation becoming an increasingly vexing problem, power analysis of Graphical Processing Unit (GPU) and its components has become crucial for hardware and software system design. Here, we describe our technique for a coordinated measurement approach that combines real total power measurement and per-component power estimation. To identify power consumption accurately, we introduce the Activity-based Model for GPUs (AMG), from which we identify activity factors and power for microarchitectures on GPUs that will help in analyzing power tradeoffs of one component versus another using microbenchmarks. The key challenge addressed in this work is real-time power consumption, which can be accurately estimated using NVIDIA's Management Library (NVML). We validated our model using Kill-A-Watt power meter and the results are accurate within 10%. This work also analyses energy consumption of MAGMA (Matrix Algebra on GPU and Multicore Architectures) BLAS2, BLAS3 kernels, and Hessenberg kernels.**

***Keywords*-power-aware, temperature-aware, GPUs, AMG, MAGMA power analysis, NVML, NVIDIA C2075**

I. Introduction

With power consumption and heat dissipation issues pushing multi-core CPUs to the limit the importance of a Graphical Processing Unit (GPU) cannot be emphasized enough.

GPU accelerated computing systems have drawn the attention of researchers because they have tremendous computational power and high memory bandwidth, and are inherently well suited for massively data parallel computation. In the November 2011 ranking, 39 of the Top 500 computer systems utilized GPUs, up from 17 systems listed in the June 2011 ranking [1].

Coming along with this exciting computational capability, the power consumption of supercomputers has become a serious issue. For example, the average power consumption of the TOP 10 supercomputing centers was 1.32 MW in 2008, and climbed to 3.2 MW in 2010, translating to multi-million-dollar electric bills. Designers must employ aggressive techniques to keep the ballooning energy cost under control. The consequences of growing energy consumption are more complex cooling solutions and noisier fans. Cooling modern video cards is becoming much more difficult, especially when users are asking for quiet cooling solutions.

The power consumption increases with performance, and engineers are now paying more attention to power consumption for new GPU designs. The cost to maintain such huge machines is expensive too, e.g. 12MW at $0.10/kW-h is $1200 an hour or about $10.5 million per year.

Though prior work has been done on power measurement of GPUs, [2]–[4], the real-time measurement of individual GPU components using a software approach, is new. In this work, we develop our model to measure real-time power usage of micro-architectures running representative computational kernels through the use of NVML (Nvidia Management Library) [5].

We refer to estimating at this granularity as per-component power estimation. Per-component power estimation is useful for selectively enabling and disabling micro-architectural resources. As power-management becomes increasingly important, coarse-granularity power estimation is likely to become inadequate to manage and continuously reallocate power budgets for individual micro-architectural components. To address the challenges of estimating per-component power in hardware, we propose a new analytical model, called the Activity-based Model for GPUs (AMG), to estimate activity factors and power for micro-architectural components on GPUs. This model does not rely on real-time current monitoring or simulating hundreds of utilization statistics similar to [6].

We maintain that only a few input statistics are sufficient to estimate per-component dynamic power of a GPU because the myriad per-component events are related to a small set of global parameters, such as load rate or the execution time of that component. We use this key observation to drive the development of AMG. Using minimal input data, AMG′s linear-regression-based methodology can estimate activity for tens or hundreds of micro-components. We monitor only the representative metrics and use the monitored metrics to extrapolate the metrics of interest. After we get all the desired metrics about the component events, we apply a per-event energy model derived from a circuit model, to those components to calculate the power consumption of each component. We also show power vs temperature of several kernels.

CFP1225P-ART/12 $26.00 © 2012 IEEE
DOI 10.1109/SAAHPC.2012.26

A. Primary Contributions of this work

- The AMG model for real-time measurement of power and energy consumption on GPUs;
- Per-component analysis of power consumption of different GPU components like floating point units, shared memory, and global memory;
- Power and energy consumption analysis of BLAS2, BLAS3 kernels and MAGMA Hessenberg kernel.

II. Related Work

Early work focussed on measurement of power dissipation using external devices such as clamp probes [2]. The use of markers is clearly explained and they have used previous marker positions to estimate the next marker position using the matching method by [2]. The performance and power relationship they have derived is

$$w = 72 + 1.02 * 10^{-10} \rho f \tag{1}$$

$$\rho = \begin{cases} 1 & \text{if } \theta \equiv \text{even} \\ 1 + 0.71(16 - (\theta mod 16)/\theta) & \text{if } \theta \text{ is odd} \end{cases} \tag{2}$$

The number of threads per block be θ, the flops performance be f (flops), and the average wattage be w (W).

The root mean square error of their approximation is 0.29 W [2].

Using hardware devices might be problematic especially since we need a separate power supply and devices like Kill-A-Watt lack a method to log the data automatically [7]. [3] used a LEAP-Server to monitor power of subcomponents of a system such as GPU with micro-scale capability. Isci et al. developed a hardware based counter model for power estimation of sub-components of a CPU [8]. A combination of P4 hardware performance events were used to estimate the power. The counter based run time for power monitoring is based on access rate heuristics, which can be used as weights to analyze power from run-time power. Breakdown of components was based on physical attributes rather than conceptual grouping. Power of each sub-component was derived using the formulae below:

$$Power(C_i) = AccessRate(C_i) \times ArchitecturalScaling(C_i) \times MaxPower(C_i) + NonGatedClockPower(C_i) \tag{3}$$

$$TotalPower = \sum_{i=1}^{22} Power(C_i) + IdlePower \tag{4}$$

Hong et al. estimated the number of cores needed for optimal power and performance using GPGPUs [4]. The theory behind this is that when a memory bound application is executed, performance does not increase proportionally with the number of cores. Their conclusions show us that by not using all the cores we can save energy up to 22.09%. They have also estimated the power consumption of sub components using micro-benchmarks in such a way that the floating point benchmark has a high number of floating point operations.

Power consumption can be divided into two parts: dynamic power and static power:

$$Power = Dynamic_{power} + Static_{power}. \tag{5}$$

Power consumption of sub-component of GPU can be modelled:

$$Power(C_i) = AccessRate(C_i) \times ArchitecturalScaling(C_i) \times MaxPower(C_i) + NonGatedClockPower(C_i) \tag{6}$$

Chen et al. showed us that in the previous generation GPUs there was no support for sensors to measure power, but with the evolution of the new Fermi architecture this has changed [9]. The older Fermi GPUs, like the C2050, have partially supported power measurement using power states P0-P15 where P0 is the power state when the GPU is running under full load and P15 is the idle state power consumption, while newer generations such as the C2075 have sensors which measure power in watts. Chen et al. developed a GPU power consumption model based on a linear regression tree [10], and random forest methods [11]. A regression is a statistical analysis assessing the relationship between two variables. Random forest uses various models to obtain performance, which is better than any individual method, and in this case consists of many decision trees, and returns the class that is the mode of classes output by individual trees. The most influential variables and several performance-sensitive architecture metrics were identified using the random forest model. Verification of their model was done using leave-one-out cross validation with an average percentage error of 7.77%.

III. Power Measurement on Nvidia Fermi C2075

A. Power Measurement using NVML

Nvidia Management Library (NVML), a high level utility called nvidia-smi not only provides a way to measure power but also various other features like the ability to set ECC (Error Correction Code) to zero if it is not needed, or to monitor memory usage, among other things. For a full list of features available via the nvidia-smi utility please refer to the NVML manual [5]. NVML can be used to measure power when running the kernel but since nvidia-smi is a high level utility the rate of sampling power usage is very low and unless the kernel is running for a very long time we would not notice the change in power. NVML offers a lot of useful utilities for not only GPUs such as C2075 but also the Nvidia Tesla C2050 GPU where one would see power in power states rather than milliwatts. The nvmlDeviceGetPowerUsage function in the NVML library retrieves

the power usage reading for the device, in milliwatts. This is the power draw for the entire board, including GPU, memory, etc. The reading is accurate to within a range of +/- 5 watts error with milliwatt precision. It is only available if power management mode is supported.

We can also query for power management support using nvmlDeviceGetPowerManagementMode. For a C2050 GPU we would observe power states P0-P15 using the NVML function call nvmlDeviceGetPowerState where P0 is the power state when the GPU is running under full load and P15 is the power state when the GPU is completely idle for a long time. We can also retrieve temperature using the NVML high level utility or using the Nvidia Management Library's function call nvmlDeviceGetTemperature.

B. Measuring Power Consumption using external device

We used Kill A Watt to validate our power model [7]. The Nvidia C2075 GPU is connected via PCI-Express to the main processor, but the power delivered through PCI-Express to the C2075 is not sufficient since PCI-Express can deliver only small amounts of power. We connected the C2075 to an external N110EF-00 power supply so we can attach a power meter and validate our results. Figure 1 shows us the power management connections and gives a clear idea of how we validate our model.

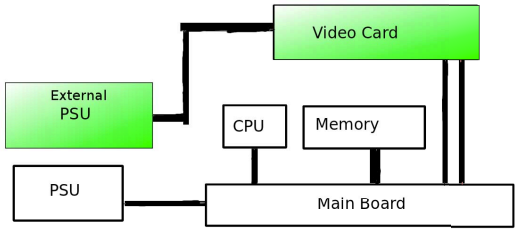

Figure 1. Validation using external power supply

IV. Activity-based Model for GPUs (AMG)

A key challenge to effective runtime power management is to know the real-time power consumption. Although the power estimation for processors, memories, disks, and fans has been introduced, the power estimation technique of GPUs is relatively less addressed. However, runtime power estimation for individual micro-architectural components on GPUs, such as caches and ALUs, would be useful for fine-grain management of package temperature and power requirements. We refer to estimating at this level as per-component power estimation. Per-component power estimation is useful for selectively enabling and disabling of micro-architectural resources.

To address the challenges of estimating per-component power in hardware, we propose a new analytical model, called Activity-based Model for GPUs (AMG), to estimate activity factors and power for micro-architectural components on GPUs. This model does not rely on real-time current monitoring or simulating hundreds of utilization statistics. We expect only a few input statistics are sufficient to estimate per-component dynamic power of a GPU because the myriad per-component events are related to a small set of global parameters, such as execution time and load rate. We use this key observation to drive the development of AMG. In spite of limited input data, AMG′s linear-regression-based methodology can estimate activity for tens or hundreds of micro components. We first analyze the correlation of a variety of performance metrics. Then we monitor only the least correlated metrics and use monitored metrics to extrapolate the concerned metrics. After we get all the concerned metrics about the component events, we apply a per-event energy model to those components to calculate the power of each component. We believe that AMG makes a further step towards understanding and reducing the power of GPU systems through the usage of architecture level performance counters.

A. Frequency of measurement of NVML Sensor

Frequency of measurement is an important part of our analysis because if we measure power readings at a higher frequency than proposed, we observe the power reading repeating in a regular fashion. For example, if one calls NVML at 200 Hz frequency, one would observe 21 power measurement values repeating three times for our benchmark. To validate this, we have conducted a series of experiments where we measured Taylor series benchmark power at 200 Hz and at 625 Hz and each value repeats 3 and 6 times, respectively. So the maximum power measurement frequency is 62.5 Hz (Figure 2). These experiments show that over sampling the sensor will provide us with no additional information.

V. PTX analysis

PTX stands for Parallel Thread eXecution, which is a pseudo-assembly code for GPUs [12]. PTX provides us with insight about how our code gets mapped into the CUDA architecture. It provides a machine independent ISA for C/C++.

We look at the PTX code to analyze the number of registers used, number of branches, global memory access,

and floating point operations which is the key for our micro-benchmarks. This analysis is of particular interest to us because, if we wrote a micro-benchmark to test floating point operations, we would like to minimize data transfer and stress the floating point operations using registers.

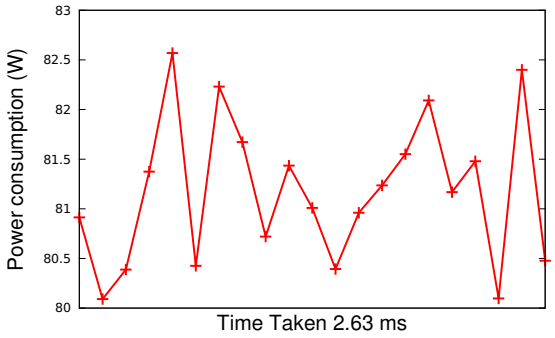

Figure 2. Frequency of measurment 62.5 Hz

A. Idle Power

Idle power is the power consumed by the GPU when the GPU is turned on but no kernel is running. We measured the short idle power of a C2075 when the GPU is just turned on and does not do any work at 80W, which is also known as startup power for any kernel to be launched. When the GPU is in a long idle state, i.e., when the GPU is doing nothing for a long period of time, we measured the power consumed at 35W. The TDP (Thermal Design Power), as reported by NVIDIA, for the C2075 is 225W [13].

B. Run-time Power Consumption

We measure the run-time power of a kernel with the NVML library by running the kernel on a thread and NVML on another thread using Pthreads. We have chosen Pthreads because we would like to reduce overhead, and the only communication we would like to have with the main thread is a flag variable and variable to store power readings that are set to be volatile. The thread that is running NVML stops when the flag is reset, which is when the GPU kernel stops executing. For our run-time power consumption measurements of different micro-architectures, such as floating point, shared memory, and global memory, we have designed micro-benchmarks such as memory copy with coalesced memory and with noncoalesced memory. For the floating point benchmark derived from a Taylor series, we run 1 million operations with measure power of 14 blocks and each block running 1024 threads. We used enough threads to cover the arithmetic latency of the SMs (Streaming Multiprocessors), which means that on a Compute Capability 2.0 GPU, we need about 10 warps (groups of 32 threads) per SM. So that means, for example, on a Fermi C2075 GPU with 14 SMs, we would need at least 4480 threads, divided into at least 14 blocks.

Table I
POWER CONSUMPTION OF VARIOUS COMPONENTS

$P_{u,i}$ Values for different components	Value (W)
Floating Point	2.2
Shared Memory	1
Global Memory	3.0

Table II
BASE POWER FOR VARIOUS COMPONENTS

$B_{u,i}$ Values for different components	Value(W)
Floating Point	6
Shared Memory	3.85
Global Memory	10

The way to manage the number of active SMs is changing the number of active blocks [4]. We use 14 blocks to run each benchmark since the C2075 has 14 SMs that run simultaneously. If more than 14 blocks are assigned, the next blocks waits for one of the blocks to finish working and then starts working. Energy consumption varies with the number of SMs because of the low activity factors, as idle SMs do not consume as much energy as active SMs.

We construct our model as

$$\text{Total power consumption} = \text{Idle Power} + \text{Runtime Power} \tag{7}$$

$$\text{Runtime Power} = \sum_{i=1}^{e} (N_{SM} \times P_{u,i} \times U_{u,i}) + B_{u,i} \times U_{u,i} \tag{8}$$

N_{SM} Number of components
$P_{u,i}$ Power consumption of active component
e number of architectural component types
$B_{u,i}$ Base power of component
$U_{u,i}$ Utilization

C. Floating Point Operations

The intent of this benchmark is to create kernels that use the floating point components, but with little or no other parts of SPs used. The benchmark scales from 1 to 14 SMs, with each of their floating point ALUs heavily used. The power contribution of floating point components is to fit a line parameterized by the number of SMs that are busy. There are 14 SMs in the GPU, so average power consumption decreases after 14 blocks because idle SMs donot consume as much power as active SMs.

We designed our floating point benchmark based on a Taylor series in such a way that each thread computes the Taylor series of an element. We iterate each calculation 16000 times to make the kernel run long enough so that we can get stable power readings. During this process of measuring floating point instructions, we only use registers for storage. The memory usage reported by cudaMemGetInfo is found to be 80 MB mainly because that is the memory that is set apart by the compiler for the GPU usage. We expect the memory usage to be much lower than that since most of the

variables are reused in an iterative way by each thread. We use cublasSasum to add the thread's results together so that the compiler will not be able to optimize.

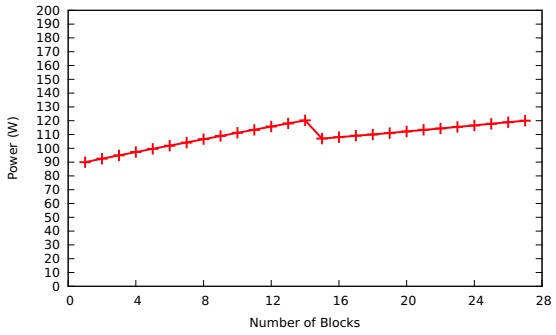

Figure 3. Average Power consumption of Floating point operations

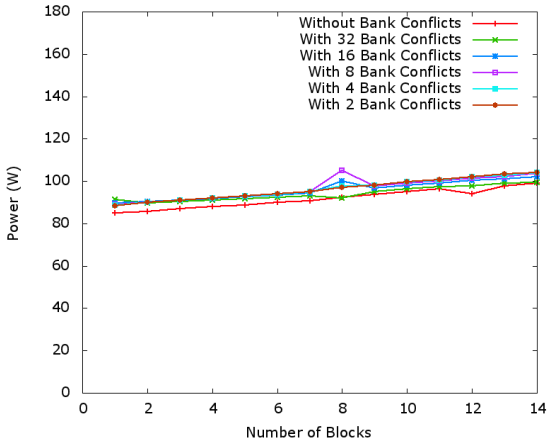

Figure 4. Power consumed by shared memory without bank conflicts

D. Shared Memory

We use __shared__ to allocate shared memory as explained in the CUDA manual [14]. We wrote micro-benchmarks for shared memory with and without bank conflicts. We use the cudaFuncSetCacheConfig function to increase the shared memory size from 16K to 48K. This allows us to estimate average power consumed by the kernel when we use the shared memory completely. The default is 48 KB for shared cache and 16 KB for L1. Shared memory is divided into 32 banks and each bank holds a 32-bit value (integer or float), so we write micro-benchmarks to exhibit the energy difference between shared memory with bank conflicts and without bank conflicts.

Table III
SHARED MEMORY

Case1	No bank conflicts
Case2	two bank conflicts
Case3	four bank conflicts
Case4	eight bank conflicts
Case5	sixteen bank conflicts

Shared memory without bank conflicts is designed to have regular access patterns. A set of micro-benchmarks is designed to analyze shared memory.

The advantages of using shared memory over global memory are many:

1) Cooperation between threads.
2) Much faster than global memory.
3) If one thread loads data it can be used all the threads.
4) The amount of shared memory is configurable via the cudaFuncSetCacheConfig function.

E. Global Memory

Global memory space is the largest memory available on a GPU. For example, on NVIDIA C2075 there are 6 GB of GDDR5, which is global memory implemented with Dynamic Random Access Memory (DRAM). The latency of global memory is on the order of hundreds of cycles, and the bandwidth is also very limited. By looking at the PTX code we can actually identify the global memory access.

1) Coalesced Memory: Since access to global memory is via 32, 64, or 128 byte accesss, we design our benchmark in such a way each thread can access memory in a regular pattern of 128 bytes. Coalesced memory access is very important for instruction throughput. The local and global variables use global memory. If we declare an array of large size without using shared memory, it resides in global memory and access of strides of 128 is actually better than an irregular pattern. Figure 5 shows power consumption of coalesced memory.

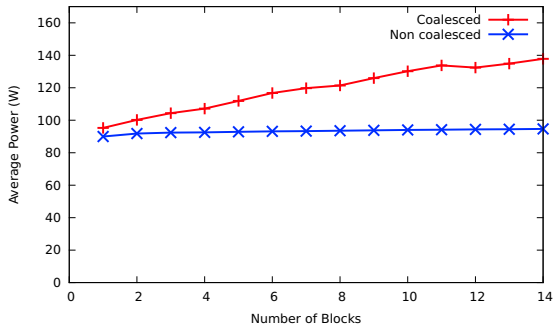

Figure 5. Power consumed by global memory memory access

2) Noncoalesced Memory: Memory access to global memory which are not regular patterns to global memory are called noncoalesced access. Our results show that the noncoalesced memory consumes at least twice the energy consumed for 16k writes and 16k reads compared to coalesced memory access. Figure 5 shows power consumption of noncoalesced memory access. The noncoalesced memory

access is at least 4 times slower than coalesced memory access which results in huge energy consumption.

VI. Power Prediction

To predict power using our model we use naive matrix matrix multiply from the CUDA SDK, i.e. one which does not use shared memory and in the later section we show average power prediction for MAGMA kernels. For matrix multiply which does not use shared memory the number of global memory reads is N^2 and writes for the kernel is N since we consider two matrices of N * N. The average power consumed by this kernel of size 14K is 130 W. We choose matrix of size 14K because C2075 GPU has 14 SM and each SM has 1024 threads so for a matrix of size 14K all the threads in each SM are working. The average power consumed by the kernel that uses shared memory is 120 W since the number of reads and writes to global memory decrease by a large factor.

Predicting power using AMG is an important step since previous generation GPUs such as Nvidia C2050 do not fully support NVML. To predict power we need the execution time of each component

We split the matrix matrix multiply so that we can tease out computation and memory. We write a CUDA kernel which performs the same number of floating point operations as the matrix multiply and measure the time taken and average power consumed by the kernel and we follow the same rule for memory operations.

Table IV
Matrix Matrix Multiply example

Kernel	Run time Power(W)	Processing Time (sec)
Matrix Matrix Multiply	50	111
Floating point	37	45
Memory	45	47

Time taken by floating point component = 45 seconds
Number of SMs used M = 14
Power consumed/SM by active component = 2.2 W
Time taken by global memory = 47 seconds
Number of SMs used M = 14
Power consumed/SM by active component = 3 W

$$\text{Run time power} = (((14 \times 2.2 \times 0.405) + 6 \times 0.405) + \quad (9)$$

$$((14 \times 3 \times 0.423) + 10 \times 0.423)) \quad (10)$$

Runtime Power = 36.9 W Idle Power = 80 W
Total Power for Matrix Multiply = 116.9 W

$$\%Error = \frac{|ActualValue - PredictedValue|}{ActualValue} * 100 \quad (11)$$

$$\%Error = \frac{13.91}{130} * 100 \quad (12)$$

$$\%Error = 10.7\%$$

Table V
Matrix Matrix Multiply evaluated using AMG

Parameter	FPU	Global
M	14	14
$P_{u,i}$ W	2.2	3
$B_{u,i}$ W	6	10
$U_{u,i}$ W	0.405	0.423

Using matrix matrix multiply we have shown that our model predicts power consumption if we know the execution rates. The execution time of each individual components does not add to the total execution time, so that is the primary reason for the error. If we could obtain more precise execution rates of each individual component we might be able to obtain higher accuracy.

A. Power and Temperature relationship

The power consumption is a very critical parameter of contemporary integrated circuits. It is obvious that a circuit should consume as little power as possible and ought to work with maximum speed and efficiency. However, power parameters are dependent on temperature, which can change with power dissipated in the circuit. In a GPU there are several SMs working simultaneously which further increases power. With rising temperature, power consumption becomes higher, too. The principal reason for that behavior is the increased amount of leakage current with higher temperatures, and the negative temperature coefficient of the transistors [15].

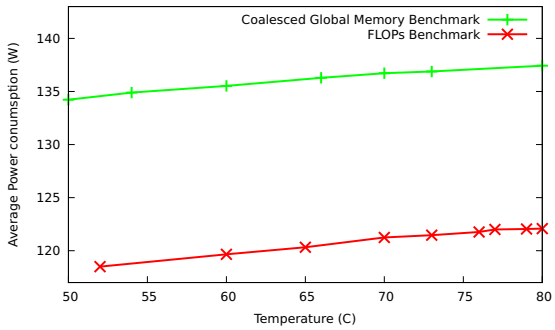

Figure 6. Increase in power consumption with temperature

Figure 6 shows the increase in power consumption when Taylor series and memory copy benchmarks are executed at various startup temperatures. To figure out the temperature influence on power, a kernel is executed that applies a workload to the GPU in order to raise the temperature of the GPU to a certain value. The power consumption of FLOPs benchmark increases by 4 W when startup temperature increase from 50 C to 80 C. Because the NVML power measurements are only accurate to within 5 W, we don't

consider temperature hereafter. The temperature influence on power consumption is only 3-4% for the current generation GPU. Thermal slowdown occurs at 90 C and thermal shutdown at 100 C.

VII. Power Analysis of MAGMA Kernels

The energy consumption of linear algebra kernels is of vital importance, as these kernels are widely used, so we measured BLAS2 and BLAS3 MAGMA kernels. Results show that the MAGMA implementations of these algorithms achieve astounding energy efficiency.

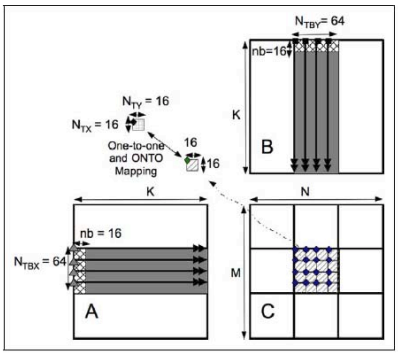

Figure 7. GPU GEMMs on Fermi architecture [17]

The real-time power consumption for GEMM and GEMV kernels is measured, and power consumed by the MAGMA DGEMM is 180 W and for DGEMV power consumption is 135 W.

VIII. BLAS 2 and BLAS 3 Kernels

The MAGMA kernels utilize CPU and GPU for the computations. The measuring frequency is 125 kHz which is twice the NVML update frequency. The impact on CPU computations while spawning Pthreads to measure power using NVML is small as frequency is not very high. The DGEMM performance for a matrix of size 9K is 296.11 GFLOPS, which is 58% of the theoretical peak, and the performance of SGEMM for a size 9K matrix is 632 GFLOPS.

The performance of SGEMV and DGEMV is considerably less compared to SGEMM and DGEMM because the BLAS 2 kernels do not utilize the GPU as efficiently as BLAS 3 kernels. SGEMV and DGEMV deliver 60 GFLOPS and 30.47 GFLOPS for a 9K matrix.

The GFLOPS per Watt for a matrix of size 10112 is 1.49; this illustrates that the GPU not only has better performance compared to CPU, but also saves on energy. Figure 8 shows the power consumed by double precision MAGMA GEMM and MAGMA GEMV. The same amount of power is consumed by DGEMM and SGEMM kernels because an SP can issue two single precision instructions or one double

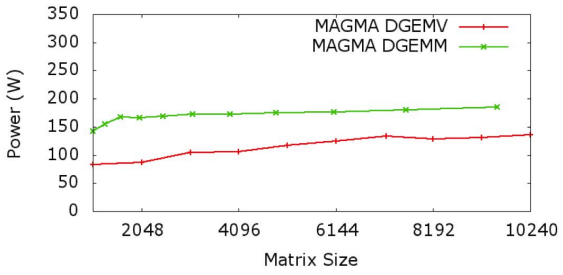

Figure 8. Average Power consumption of double precision MAGMA BLAS3 and BLAS2 Kernels

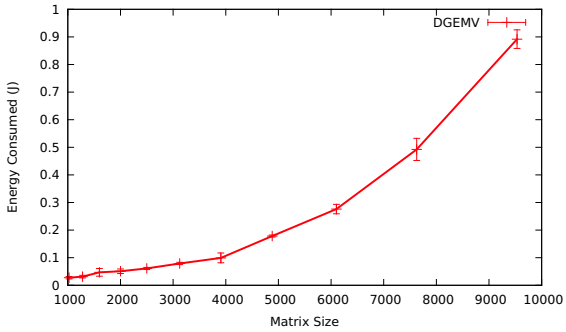

Figure 9. Energy consumption of double precision MAGMA BLAS2 Kernels

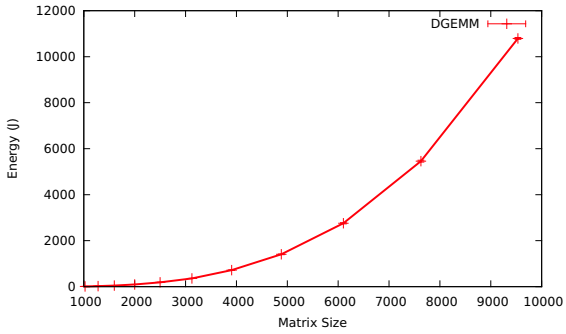

Figure 10. Energy consumption of double precision MAGMA BLAS3 Kernels

precision every two clocks, but energy varies since single precision MAGMA GEMMs and GEMVs are twice as fast as double precision.

A. Measuring time taken in MAGMA Kernels

The function get_current_time() calls gettimeofday(), so the resolution is a microsecond. Before calling the gettimeofday() there is a call to cudaThreadSynchronize() to make sure previous GPU tasks have completed. Thus one can measure the time of a particular GPU kernel by surrounding it by calls to get_current_time(). If between two get_current_time() calls there are functions transferring data, the time measure will include the time for the transfer. We

measure the time for DGEMM on the GPU, i.e., we assume the data and the result will be in the GPU memory.

B. Predictions for MAGMA kernels for matrix of size 10K based on AMG

From our AMG model

$$\text{Total power} = \text{Idle Power} + \text{Runtime Power} \quad (13)$$

$$\text{Runtime Power} = \sum_{i=1}^{e} (N_{SM} \times P_{u,i} \times U_{u,i}) + B_{u,i} \times U_{u,i} \quad (14)$$

C. MAGMA Hessenberg

The Hessenberg reduction algorithm is of the form $Q^T AQ = H$. In contrast to the LU factorization, the other algorithm that we have studied, namely the Hessenberg Reduction, cannot be entirely expressed in terms of GEMMs. Proper task splitting on hybrid architectures using the Hessenberg principle, has been to known to give enormous performance benefits [18]. According to [18] the operation count to reduce an N by N matrix is $(10/3)n^3$ and this makes Hessenberg reduction a suitable candidate for acceleration. But 20% of the total flops of the algorithm, which take 70% of time, are in level 2 BLAS. This makes the algorithm memory bound and we observe it for an energy consumption that is close to the power consumption for matrix-vector operations. Figure 11 shows us the power consumption for MAGMA DGEHRD which is signal averaged over 24 data samples.

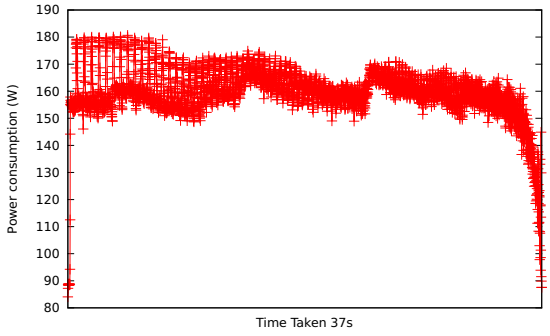

Figure 11. MAGMA Dgehrd power consumption for a 10k matrix

According to the MAGMA DGETRF algorithm

$$\forall panel = \begin{cases} 64 \text{ small DGEMV} \\ 6 \text{ large DGEMM} \end{cases}$$

.

Table VI shows the number of DGEMM and DGEMV calls in MAGMA Hessenberg. There is a large difference between the number of DGEMM and DGEMV calls measured and observed because the frequency of the NVML sensor is on the order of 62.5 Hz whereas the clock frequency of the GPU is of the order of 1.15 GHz.

Table VI
DGEMM AND DGEMV CALLS IN MAGMA HESSENBERG FOR A MATRIX OF SIZE 10 K

Source	Number of DGEMM calls	Number of DGEMV calls
Algorithm	936	9984
NVML	367	1457

Table VII
AVERAGE POWER CONSUMPTION FOR A MATRIX SIZE 10 K

MAGMA Kernel Name	Average power consumed (W) of matrix size 8K
DGETRF	165
DGEHRD	150

1) Power prediction for MAGMA DGEMM: MAGMA Double precision General Matrix Matrix multiply uses the GPU completely. Power is predicted using our AMG model. Even though MAGMA kernels do a very good job of hiding data latencies with computations utilization rates for shared memory and global memory are 100%.
Power consumed by floating point component = $14 * 2.2 \times 0.58 + 6 \times 0.58$ = 17.864 W
Power consumed by shared memory = $14 \times 1 \times 1 + 3 \times 1$ = 17 W
Power consumed by global memory = $14 \times 3 \times 1 + 10 \times 1$ = 52 W
runtime power = 86.864 W Idle power= 80W
Total power Predicted = 166.864 W
Total power measured = 180 W
error = 7.3%
MAGMA DGEMM achieves 58% of the theoretical peak that is the reason utilization for floating point is .58.

2) Power prediction for MAGMA DGEMV: MAGMA Double precision General Matrix Vector does not utilize the GPU fully as the matrix-vector operations get stalled by memory since the memory reads and writes to global memory are not as fast as floating point operations on shared memory or registers.
Power consumed by floating point component $14 \times 2.2 \times 0.4 + 6 \times 0.4$ = 14.72
Power consumed by shared memory $1 \times 14 \times 0.2 + 3 \times 0.2$ = 3.4 W
Power consumed by Global memory $3 \times 14 \times 0.8 + 10 \times 0.8$ = 41.6W
Total runtime power = 63.4 W
Idle power = 80 W Total power Predicted = 143.4 W
Total power measured = 135 W
error = 6.2%

The total power predicted is close to the measured power consumption. One of the difficulties was finding the execution time of each component. Floating point components are busy only half of the time as memory needs to be fetched. Shared memory is used for $1/5^{th}$ of the time and global

memory is used the rest of the time, that is the reason performance of the DGEMV is only 30 GFLOPS; it is a memory bound kernel.

Table VIII
POWER CONSUMPTION OF MAGMA BLAS2 AND BLAS3 KERNELS

MAGMA Kernel	Average power consumed (W) of matrix size 8K
DGEMM	180
SGEMM	180
DGEMV	135

IX. CONCLUSION

To address the challenges of estimating per-component power in hardware, we proposed a new analytical model, called Activity-based Model for GPUs (AMG), to estimate activity factors and power for micro-architectural components on GPUs. The power model using AMG predicts the power consumption and the execution time with a maximum error of 10% for the evaluated GPGPU kernels. Live measurements using Nvidia Management Library (NVML) are of particular interest to users, so we have measured power on Nvidia C2075 GPU using NVML. We have also analyzed power consumption of various MAGMA kernels, level 2 and level 3 BLAS kernels. We have also analyzed power consumption of arithmetic intensive MAGMA BLAS2 and BLAS3 kernels.

There are several key contributions of this work. The measurement technique itself is portable, and can offer a viable alternative to many of the power simulations currently guiding research evaluations. The component breakdowns offer sufficient detail to be useful on their own, and their properties as a power signature for the power aware phase analysis seem to be even more promising. In conclusion, this work offers both a measurement technique, as well as a characterization of a GPU's various components. We feel it offers a promising alternative to purely estimation-based power research.

X. FUTURE WORK

Our model can also be used by compilers or programmers to optimize program configurations as we have demonstrated in the work. In our future work, we will build a multi CPU_GPU model, that will give us a complete picture of power consumption for a system like Keeneland [19], which has 120 nodes with 240 CPUs and 750 GPUs.

Power consumption for GPUs that do not support power management mode is reported in Power states (P0-P15), where P0 is the power state under maximum load, and P15 is idle power consumption. We would like to deduce power numbers in Watts for those states so that users can understand the meaning of power states. With ARM-based CPU/GPU hybrid systems being deployed to reduce energy consumption, issues for modelling such hybrid systems will be of special interest to us. The Barcelona Supercomputing Center is developing a new ARM based machine to achieve 4 to 10 times the energy efficiency of today's supercomputers [20].

ACKNOWLEDGMENT

This work has been partially supported by NSF grant CNS0910899 and by DOE SciDAC grant DE-SC0006733. Furthermore, we thank NVIDIA for providing us with GPUs and insight into the NVIDIA Management Library, both of which were invaluable resources in this research.

REFERENCES

[1] J. D. Hans Meuer, Erich Strohmaier, "TOP500 Supercomputer Site," http://www.top500.org, 2012.

[2] R. Suda and D. Q. Ren, "Accurate Measurements and Precise Modeling of Power Dissipation of CUDA Kernels toward Power Optimized High Performance CPU-GPU Computing," in *Proceedings of the 2009 International Conference on Parallel and Distributed Computing, Applications and Technologies*, ser. PDCAT '09. Washington, DC, USA: IEEE Computer Society, 2009, pp. 432–438. [Online]. Available: http://dx.doi.org/10.1109/PDCAT.2009.65

[3] M. Rofouei, T. Stathopoulos, S. Ryffel, W. Kaiser, and M. Sarrafzadeh, "Energy-Aware High Performance Computing with Graphic Processing Units," in *Proceedings of the 2008 Conference on Power Aware Computing and Systems*, ser. HotPower'08. Berkeley, CA, USA: USENIX Association, 2008, pp. 11–11. [Online]. Available: http://dl.acm.org/citation.cfm?id=1855610.1855621

[4] S. Hong and H. Kim, "An integrated GPU Power and Performance Model," *SIGARCH Comput. Archit. News*, vol. 38, no. 3, pp. 280–289, Jun. 2010. [Online]. Available: http://doi.acm.org/10.1145/1816038.1815998

[5] NVIDIA Corporation, "NVML API Reference," http://developer.download.nvidia.com/compute/DevZone/NVML/doxygen/index.html, 2012.

[6] Texas Instruments, "OMAP-L138 Power Consumption Summary," http://processors.wiki.ti.com/index.php/OMAP-L138_Power_Consumption_Summary\#Activity-Based_Models, 2012.

[7] Kill A Watt, "KILL A WATT P3," http://www.p3international.com/products/special/P4400/P4400-CE.html.

[8] C. Isci and M. Martonosi, "Runtime Power Monitoring in High-End Processors: Methodology and Empirical Data," in *Proceedings of the 36th Annual IEEE/ACM International Symposium on Microarchitecture*, ser. MICRO 36. Washington, DC, USA: IEEE Computer Society, 2003, pp. 93–. [Online]. Available: http://dl.acm.org/citation.cfm?id=956417.956567

[9] J. Chen, B. Li, Y. Zhang, L. Peng, and J. Peir, "Statistical GPU Power Analysis using Tree-based Methods," in *Green Computing Conference and Workshops (IGCC), 2011 International*, July 2011, pp. 1 –6.

[10] Breiman, Leo and Friedman, Jerome H. and Olshen, Richard A. and Stone, Charles J., *Classification and Regression Trees*. Chapman & Hall, New York, NY, 1984.

[11] L. Breiman, "Random Forests," *Mach. Learn.*, vol. 45, pp. 5–32, October 2001. [Online]. Available: http://dl.acm.org/citation.cfm?id=570181.570182

[12] NVIDIA Corporation, "PTX manual," http://developer.download.nvidia.com/compute/DevZone/docs/html/C/doc/ptx_isa_3.0.pdf, 2012.

[13] R. Doe, "This is a test entry of type @ONLINE," Jun. 2009. [Online]. Available: http://www.test.org/doe/

[14] NVIDIA, *CUDA C programming guide*. nvidia corporation, 2012.

[15] F. Fallah and M. Pedram, "Standby and Active Leakage Current Control and Minimization in CMOS VLSI Circuits." *IEICE Transactions*, vol. 88-C, no. 4, pp. 509–519, 2005. [Online]. Available: http://dblp.uni-trier.de/db/journals/ieicet/ieicet88c.html\#FallahP05

[16] Fatahalian, K. and Sugerman, J. and Hanrahan, P., "Understanding the Efficiency of GPU Algorithms for Matrix-Matrix Multiplication," in *Proceedings of the ACM SIGGRAPH/EUROGRAPHICS Conference on Graphics Hardware*, ser. HWWS '04. New York, NY, USA: ACM, 2004, pp. 133–137. [Online]. Available: http://doi.acm.org/10.1145/1058129.1058148

[17] Nath, Rajib and Tomov, Stanimire and Dongarra, Jack, "An Improved Magma Gemm For Fermi Graphics Processing Units," *Int. J. High Perform. Comput. Appl.*, vol. 24, no. 4, pp. 511–515, Nov. 2010. [Online]. Available: http://dx.doi.org/10.1177/1094342010385729

[18] Tomov, Stanimire and Nath, Rajib and Dongarra, Jack, "Accelerating the Reduction to Upper Hessenberg, Tridiagonal, and Bidiagonal forms through Hybrid GPU-based Computing," *Parallel Comput.*, vol. 36, pp. 645–654, December 2010. [Online]. Available: http://dx.doi.org/10.1016/j.parco.2010.06.001

[19] NICS, "Keeneland Supercomputer," http://keeneland.gatech.edu/, 2012.

[20] NVIDIA, "ARM Based Supercomputer," http://www.computer.org/csdl/mags/co/2012/03/mco2012030018.html, 2012.