GPGPU Footprint Models to Estimate Per-Core Power

Rathijit Sen rathijit@cs.wisc.edu
David A. Wood david@cs.wisc.edu
Department of Computer Sciences
University of Wisconsin-Madison

Abstract—We explore the problem of how to easily estimate the per-core power distribution of GPGPUs from the total power of all cores. We show that the dynamic energy consumption of a core for a given kernel, represented by its *work footprint*, is approximately proportional to the total time taken by all work units executing on that core, and the static power, represented by its *core footprint*, is proportional to the time that the core has assigned work. Footprints can be easily tracked using two hardware counters per GPU core. We also show how per-core power estimates can be used to compute power-performance pareto frontiers that identify opportunities for saving power and energy in cases of non-uniform work distribution by exploiting per-core DVFS support for GPGPUs.

Index Terms—GPGPU, power, footprint, DVFS, pareto frontier

1 Introduction

General-Purpose Graphics Computing Units (GPGPUs) are increasingly gaining popularity as energy-efficient accelerators for computational tasks traditionally performed on CPUs. Further improvements in energy efficiency may be possible through power management features such as dynamic voltage and frequency scaling (DVFS) for individual cores (NVIDIA: Streaming Multiprocessors, AMD: Compute Units). An essential step in predicting whether such support will be useful for a given kernel is to estimate power for individual cores for the kernel execution. Hardware support for estimating per-core power is not available on modern GPUs (and most CPUs). In this paper we propose a low-cost solution that uses timing footprints to apportion total power to individual cores.

GPU microarchitectural power models [9] estimate dynamic energy consumption by tracking switching activity. However, rather than tracking individual bit-flips on each core, it suffices to track one architectural-level metric for relative estimates. We show that *the dynamic energy consumed by a core for a given kernel is approximately proportional to the total time taken by all work units executing on that core*. This is the core's *work footprint* and is a proxy for the amount of work done by it. The core's *core footprint* is the total time it is powered on and indicates its static power consumption. Tracking work and core footprints requires only two counters per core, one per footprint.

Our simple analytic footprint model rapidly estimates core power distributions that are qualitatively similar to those from GPUWattch [9], a much more detailed simulation model, with relative errors being mostly within 10% for the NVIDIA GTX480 (Section 3). In conjunction with a device model (Section 4) and a per-core DVFS model (Section 5) we demonstrate one potential use of footprint models in predicting pareto-optimal system configurations for saving power and energy.

2 Work and Core Footprints

For now we will assume that all cores operate at the same frequency and voltage. Section 5 will relax this restriction.

The work footprint of core c, $WF(c)$, represents the amount of work done by the core. By "work" we mean bit-flips in hardware structures (e.g., pipelines, functional units, register files, etc.) caused by kernel execution. Let $t(b,c)$ denote the execution time of threadblock b on core c. $t(b,c) = 0$ if threadblock b is not assigned to core c. We hypothesize that for the given run

$$WF(c) \propto \sum_b t(b,c) \tag{1}$$

That is, the work footprint is proportional to the sum of threadblock execution times on that core with the constant of proportionality being specific to that run. The work footprint across all cores, WF, is the sum of core work footprints. Assuming the same constant of proportionality across all cores for a given kernel and device, we get

$$WF = \sum_c WF(c) \propto \sum_c \sum_b t(b,c) \tag{2}$$

Let $DE(c)$ denote the average dynamic energy consumed by core c and DE denote the average total dynamic core energy. We hypothesize that, to the first order, the average dynamic core energy for the given run is distributed in proportion to the timing approximation of work footprints. That is,

$$\frac{DE(c)}{DE} = \frac{DE(c)}{\sum_c DE(c)} = \frac{WF(c)}{WF} = \frac{\sum_b t(b,c)}{\sum_c \sum_b t(b,c)} \tag{3}$$

The above is an approximation because the rightmost ratio uses execution time as a proxy for switching activity. It works well unless stalls or divergence (branch and memory) increase execution times disproportionate to switching activities. Since we are taking ratios, it also works if cores observe these events in similar proportions relative to their total work. We expect the approximation to be good for a large number of current kernels.

Let kernel execution time be T. We compute the average total dynamic core power, DP, as

$$DP = \frac{DE}{T} \tag{4}$$

The core footprint, $CF(c)$ for core c, represents the time taken by the core to finish all work assigned to it. The largest core footprint determines T, that is, $T = \max_c\{CF(c)\}$.

We assume that the cores are power-gated when they finish their assigned work. So, the average static core power (averaged over total kernel time), $SP(c)$, of core c is proportional to the ratio of its core footprint to the total kernel time.

$$SP(c) \propto \frac{CF(c)}{T} \tag{5}$$

Manuscript submitted: 01-Mar-2015. Manuscript revised: 08-Jun-2015. Manuscript accepted: 06-Jul-2015.

TABLE 1
Per-core average dynamic power (and energy) distribution for NQU estimated by GPUWattch, Work footprint model (Eqn 3), Proxy1, and Proxy2.

Core Num	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
GPUWattch	5.83%	16.74%	5.83%	6.16%	5.83%	5.83%	6.16%	6.17%	6.16%	6.16%	6.16%	5.83%	5.83%	5.49%	5.83%
Work Footprint	5.74%	17.14%	5.81%	5.90%	5.77%	5.79%	6.03%	6.05%	6.04%	6.10%	6.12%	5.94%	5.91%	5.74%	5.91%
Proxy1	4.72%	31.96%	4.75%	4.88%	4.74%	4.75%	4.90%	4.93%	4.92%	4.91%	4.95%	4.90%	4.92%	4.88%	4.88%
Proxy2	6.67%	6.67%	6.67%	6.67%	6.67%	6.67%	6.67%	6.67%	6.67%	6.67%	6.67%	6.67%	6.67%	6.67%	6.67%

The average total core static power, SP, is the sum of static powers of all cores.

$$SP = \sum_{c} SP(c) \quad (6)$$

We can measure footprints with low implementation cost – $CF(c)$ with a counter that tracks how long core c is enabled and $WF(c)$ with a counter that accumulates the time each execution context on core c takes to finish all assigned work. We use $CF(c)$ and $WF(c)$ values to estimate per-core power distributions as unitless ratios (or, percentages, e.g., Table 1).

In this work we obtain $CF(c)$ and $WF(c)$ values from GPGPU-Sim [1] simulations. Our models estimate relative per-core power. We use the baseline total dynamic core power number from GPUWattch to estimate absolute per-core dynamic power from footprint model estimates when needed, e.g., for the full-system pareto frontier study in Section 5. Although Figures 2 and 4 show absolute power estimates, computed as stated above, they are for visualization only—relative errors and relative distributions are unaffected by absolute values.

3 Core Dynamic Power Distribution Accuracy

We will now investigate the accuracy of the work footprint model for estimating per-core dynamic power distributions. We use GPGPU-Sim for performance simulations, using the NVIDIA GTX480 as the modeled architecture, and GPUWattch for power simulations (with sampling frequency adjusted depending on kernel runtime), to obtain reference numbers. The GTX480 has 15 cores. The core clock frequency (default: 700 MHz) is half of the shader clock frequency.

We study model predictions for 6306 executions involving 49 kernels in the following workloads: ISPASS [1] (AES, LIB, LPS, MUM, NN, NQU, RAY, STO, WP), Rodinia [4] (backprop, gaussian, heartwall, kmeans, hotspot, nw, lud, leukocyte, lavaMD, srad_v1), Parboil [11] v0.2 (cutcp, mri-q, sgemm, stencil, tpacf), SHOC [5] v1.1.5 (BFS, FFT, MD, Reduction).

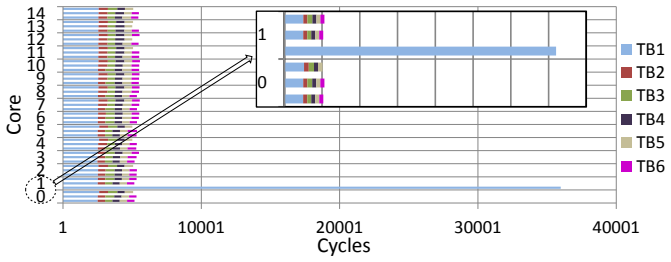


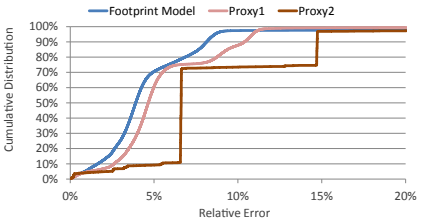
Fig. 1. Cycle trace for NQU with all cores running at 700 MHz. Inset zooms in on Cores 0 and 1. The kernel uses 3 execution contexts per core, with at most 6 threadblocks (TBs) assigned to each context.

Figure 1 shows a cycle trace of threadblock (TB) executions for the NQU (N=8 QUeens) kernel as simulated by GPGPU-Sim. This kernel has 256 TBs with at most 3 TBs concurrently dispatched to any core resulting in a maximum of 3×15 = 45 concurrent contexts. Figure 1 depicts 45 horizontal stacked bars, one for each context. We number the TBs dispatched to each context sequentially from TB1 through TB6. Core 1 has the largest footprint among all cores. NQU has significant load imbalance for this default input size.

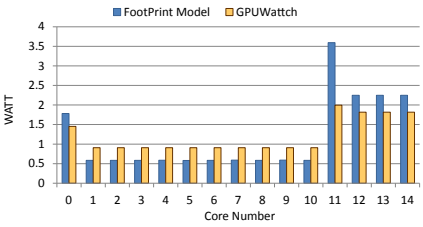
Table 1 shows good agreement between the per-core average dynamic power distributions estimated by GPUWattch and by our work footprint model (Equation 3). The percentages represent relative core power averaged over the kernel execution time. As expected, core 1 uses more power than other cores.

For comparison, we also consider two alternative models: Proxy1 (dynamic power in proportion to core footprints) and Proxy2 (equal dynamic power to all cores used). Proxy2 is inspired by earlier models [8] that did not differentiate between cores by power consumption. The work footprint model aims to improve upon Proxy1 for cases where the work distribution across cores differs from the distribution of core completion times and both aim to improve upon Proxy2 for variations across cores. Table 1 shows that the work footprint model has the closest fit to the GPUWattch estimates for this kernel.

Figure 2a shows cumulative distributions of relative error (=abs((predicted/simulated)-1)) for core dynamic powers of 6171 kernel executions (~98%) that used more than one core. We observe that the work footprint model exceeds 10% error for only 2.6% of predictions whereas Proxy1 and Proxy2 exceed it for 12.2% and 26.6% of predictions respectively. With an average error of 4.9%, the work footprint model works better than the other schemes (average errors: 5.6%, 9.1%) for little additional tracking overhead.



(a) Prediction Error

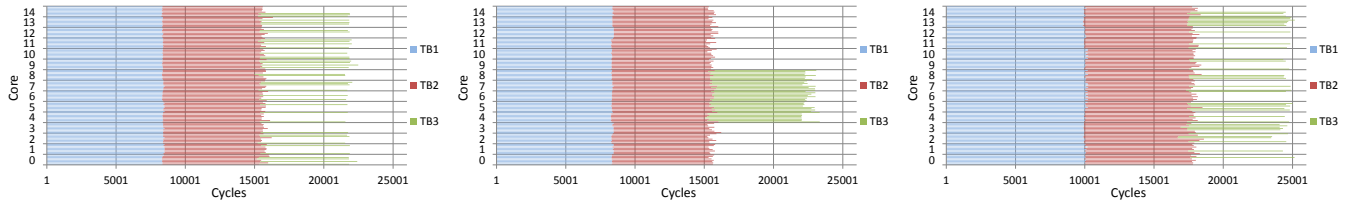


(b) BFS Iteration 6

Fig. 2. (a) Cumulative distribution of per-core dynamic power estimation error. Worst case errors – Work footprint model: 80.1% (BFS, Iter. 6), Proxy1: 90.9% (NQU), Proxy2: 313.6% (Gaussian Fan2, Iter. 4). (b) Kernel instance with largest relative error in prediction (Core 11).

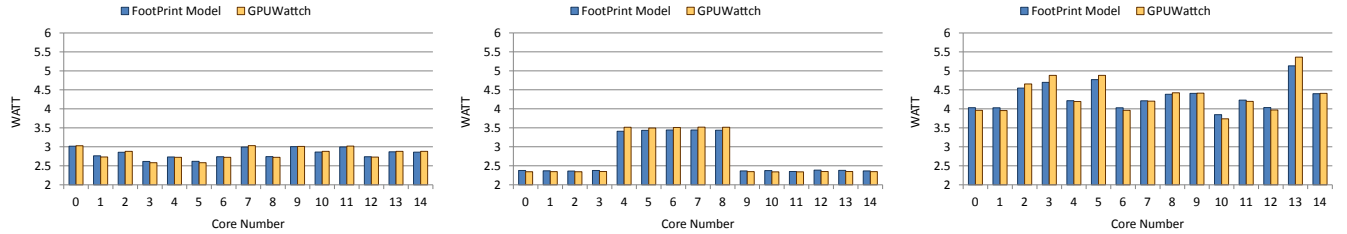
Figure 2b shows the largest error for the work footprint model, occurring at Core 11 because of extra cycles due to data access stalls and divergence instead of proportionally more work.

Per-core footprints for a given kernel can change with GPU configuration. Figure 3 shows cycle traces for the NN Fourth-Layer kernel for the baseline (3a), with *Tail Aggregation* (3b), and with core frequency scaling (3c). Tail Aggregation uses a new scheduling policy that aggregates TBs in the last "batch" (TB3) to a few cores. This saves 19.6% total core static power with 3.7% performance loss. By exacerbating inequality in per-core completion times (core footprints), it also creates new opportunities for saving energy with per-core DVFS. Frequency scaling (increase from 700 MHz to 1200 MHz) speeds up by



(a) Baseline: All @700 MHz (b) Tail aggregation: All @700 MHz (c) Frequency scaled: All @1200 MHz

Fig. 3. Cycle traces for NN FourthLayer kernel. Work and Core Footprints change with (b) scheduler (Tail Aggregation) and (c) frequency changes.



(a) Baseline: All @700 MHz (b) Tail aggregation: All @700 MHz (c) Frequency scaled: All @1200 MHz

Fig. 4. Per-core dynamic power for NN FourthLayer kernel. For these results, the footprint model uses the total dynamic core power numbers from the baseline configuration, speedup and work footprints from the target configurations. (b) Tail Aggregation can increase savings opportunities with per-core DVFS – non-critical cores (0-3, 9-14) consume ~57% baseline dynamic core energy, but can be run slower without affecting kernel time.

53.2% (more, faster cycles). Total dynamic power (fixed voltage) changes almost similarly with performance. Figure 4 shows that the work footprint model, given target speedups and $WF(c)$ and the baseline total dynamic core power, estimates per-core dynamic powers well.

4 GPU Power Model

Other components besides cores also consume power and need to be accounted for to estimate DVFS benefits. Here we describe a simple model to estimate device power for runs with different core frequencies (fixed voltage) if we know the kernel execution times of those runs and the power and execution time for the baseline configuration (all cores running at 700 MHz). We estimate device power as the sum of the following:

1) *Core dynamic*: estimated using Equation 4.
2) *Core static*: estimated using Equations 5 and 6. We assume a static power of 1W per core when it is used.
3) {*L2cache, Network-on-chip, DRAM, Memory controller*} *dynamic*: estimated by scaling the corresponding baseline power numbers with T^{-1}, similar to Equation 4.
4) *Constant*: 11W (dynamic) + 27W (additional non-core static, e.g., for voltage regulators, peripherals, etc.).

On tests with core frequencies (all cores) from 100-1400 MHz, in steps of 100 MHz (except baseline 700 MHz), the model works well with an average relative error of $<<$ 1% and a worst-case error of $<$ 8% compared to simulated numbers. Section 5 further refines the core power model.

5 Per-core Frequency and Voltage Scaling

So far we have assumed that all cores run at the same frequency (and voltage). We now use the insight that for fixed voltage and a fixed amount of work, to the first order, the core dynamic energy is not affected by the clock rate of that core. So, if $DE(c)$ is known for some core frequency $f(c)$, it can be used as an approximation for executions with a different value of $f(c)$ provided that the work assigned to core c is unchanged.

The experiments in this paper use initial $DE(c)$ estimates with all cores at the same frequency. For initial $DE(c)$ estimates with different core frequencies, we need to modify Equation 1 to include a correction factor, $\gamma(c)$, to cancel the effects of changed timing with respect to other cores. This leads to

$$WF(c) \propto \gamma(c) * \sum_{b} t(b,c) \tag{7}$$

$\gamma(c) = f(c)$ for perfectly scalable kernels, $\gamma(c) \rightarrow 1$ for kernels whose executions are primarily limited by interactions with non-core system resources, e.g, memory. $\gamma(c)$ may be estimated using regression models for performance scaling.

Since core voltage is generally changed in conjunction with core frequency, we add a cost model to $WF(c)$ and $SP(c)$ calculations to handle its effect on energy and power. Let $V(c)$ denote the voltage of core c, $cost_{de}(V)$ and $cost_{sp}(V)$ denote scaling relationships of dynamic energy and static power with voltage V. Equations 7 and 5 now become

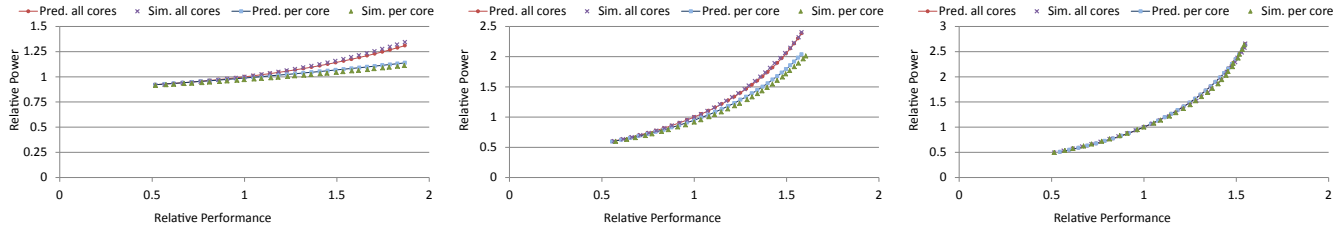
$$WF(c) \propto cost_{de}(V(c)) * \gamma(c) * \sum_{b} t(b,c) \tag{8}$$

$$SP(c) \propto cost_{sp}(V(c)) * \frac{CF(c)}{T} \tag{9}$$

We use our simple analytic models to explore the potential benefits for per-core DVFS support in GPUs for kernels that create unequal footprints on cores. We consider a deployment scenario where each core has a fixed, but independent frequency (and voltage) setting while executing a kernel and assume voltage-frequency pairs of (0.55V, 100MHz), (1.0V, 700MHz) [9], $\Delta V \propto \Delta f$, $cost_{de}(c) \propto V(c)^2$, $cost_{sp}(c) \propto V(c)$ [3], 10MHz frequency granularity, and that the GPU reports footprints but not directly per-core power.

We execute one profiling run with all cores set at 700 MHz to gather baseline time, power, and footprints. In order to correct for performance scaling bottlenecks, we execute two more performance profiling runs, with all cores at 100 MHz (expected performance: (1/7)x) and at 1400 MHz (expected performance: 2x), then fit a quadratic curve between the three points to translate between expected and realistic performances.

Figure 5 shows predicted pareto frontiers for three example kernels, with ("Pred. per core") and without ("Pred. all cores") per-core DVFS settings. The expected performance range is 0.5x-2x compared to baseline but results in a translated realistic range with the quadratic model. It also shows simulated ("Sim.") results for the predicted configurations and these are



(a) NQU (b) Heartwall Iteration 2 (c) Backprop layerforward kernel

Fig. 5. Predicted pareto frontiers with (1,1) representing the baseline execution. "per core" assumes per-core DVFS support, "all cores" assumes all cores have the same voltage and frequency. Cores are power-gated when they finish their assigned work. %Savings = $\left(1-\frac{\text{"per core" power}}{\text{"all cores" power}}\right)$.

close to the predicted values. Higher divergence between "all cores" and "per core" frontiers indicates larger power-savings potentials and is caused by both per-core power and completion time differences. This limit study shows that footprint variations can be exploited to save power – at max. performances, NQU saves 17.3% (predicted: 13.1%), heartwall saves 16.3% (predicted: 14.6%) relative power. Conversely, little variation enables negligible relative power savings, e.g., for backprop.

Our simple performance scaling model may cause performance errors (eg., ~10% for BFS Iteration 6) when kernel performance scaling differs from individual core performance scaling, e.g., with non-uniform stalls or divergence, or with changes in scheduling and shared resource contention. More sophisticated performance scaling models could address these issues. If the performance model is accurate, then errors in the footprint model may lead to suboptimal frequency settings (hence, less power savings) for cores not on the critical path.

A limitation of our approach is that profiling overheads will reduce DVFS benefits unless amortized over multiple executions with the same input. Nevertheless, our simple prediction scheme can be used to rapidly identify maximum DVFS benefits by exploiting inter-core footprint variation.

6 Related Work

Analytical modeling of GPU power consumption is not new [7], [8], [10], [12]. However, previous work tracks switching activity, multiple architectural events (e.g., executions of thirty instruction types [12]), or does not differentiate between cores. In contrast, our high-level model uses just two counters (one per footprint) per core to estimate relative per-core power.

Researchers have analyzed PTX instructions to estimate kernel execution time and power consumption, either fully [6], [7], [10] or in conjunction with profiling information [12]. Such techniques may also be extended to estimate threadblock execution times and kernel performance scaling, thereby reducing (or eliminating) the profiling runs described in Section 5.

Online thread criticality prediction for CPUs [2] uses cache miss counts to identify critical threads. It is relevant to our power-savings approach and will be useful in exploiting timing variation due to events such as cache misses that are hard to statically analyze, but may need to be extended to include instruction counts (completed and remaining) in order to account for work variation across GPU cores.

GPUWattch [9] studied DVFS benefits by lowering voltage during memory stalls to save power without (ideally) affecting performance. It does not benefit compute-bound kernels or exploit inter-core work variation. In contrast, our work focuses on inter-core variation but does not save power on stalls. Since these two approaches have complementary strengths, it would be beneficial to combine them for maximum benefits.

While prior limit studies [8] analyzed DVFS benefits on GPUs using oracles, they did not model inter-core variation or per-core DVFS. Those limits relate to "all cores" frontiers extended to include periodic DVFS changes. Our work can further extend those studies by selecting optimal per-core footprints.

7 Conclusion

We propose a simple technique using timing footprints to deduce the per-core dynamic power distribution for GPGPUs. Experimental validation using state-of-the-art simulators show good correlation between predicted and simulated distributions. We also demonstrate how per-core power estimates can be used to infer optimal power-performance relationships.

Acknowledgments

We thank Srilatha Manne, Indrani Paul, and Wei Huang for discussions about per-core DVFS support in GPUs and Mark Hill, Jason Power, anonymous reviewers, and the Associate Editor for helpful review comments. This work was supported in part with NSF grants CCF-1218323 and CNS-1302260. The views expressed herein are not necessarily those of the NSF. Wood has significant financial interests in AMD and Google.

References

[1] A. Bakhoda, G. L. Yuan, W. W. L. Fung, H. Wong, and T. M. Aamodt, "Analyzing cuda workloads using a detailed GPU simulator," in *ISPASS*, 2009, pp. 163–174.

[2] A. Bhattacharjee and M. Martonosi, "Thread Criticality Predictors for Dynamic Performance, Power, and Resource Management in Chip Multiprocessors," in *ISCA*, 2009, pp. 290–301.

[3] J. A. Butts and G. S. Sohi, "A Static Power Model for Architects," in *MICRO*, 2000, pp. 191–201.

[4] S. Che, M. Boyer, J. Meng, D. Tarjan, J. W. Sheaffer, S.-H. Lee, and K. Skadron, "Rodinia: A benchmark suite for heterogeneous computing," in *IISWC*, 2009, pp. 44–54.

[5] 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 *GPGPU*, 2010, pp. 63–74.

[6] S. Hong and H. Kim, "An Analytical Model for a GPU Architecture with Memory-level and Thread-level Parallelism Awareness," in *ISCA*, 2009, pp. 152–163.

[7] S. Hong and H. Kim, "An Integrated GPU Power and Performance Model," in *ISCA*, 2010, pp. 280–289.

[8] J. Lee, V. Sathisha, M. Schulte, K. Compton, and N. S. Kim, "Improving Throughput of Power-Constrained GPUs Using Dynamic Voltage/Frequency and Core Scaling," in *PACT*, 2011, pp. 111–120.

[9] 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 *ISCA*, 2013, pp. 487–498.

[10] C. Luo and R. Suda, "A Performance and Energy Consumption Analytical Model for GPU," in *DASC*, 2011, pp. 658–665.

[11] J. A. Stratton, C. Rodrigrues, I.-J. Sung, N. Obeid, L. Chang, G. Liu, and W.-M. W. Hwu, "Parboil: A Revised Benchmark Suite for Scientific and Commercial Throughput Computing," UIUC, Urbana, Tech. Rep. IMPACT-12-01, 2012.

[12] Q. Xie, T. Huang, Z. Zou, L. Xia, Y. Zhu, and J. Jiang, "An accurate power model for GPU processors," in *ICCCT*, 2012, pp. 1141–1146.