Implementing Support for Extensible Power Modeling in gem5

Alex Smith, Matthew D. Sinclair
University of Wisconsin-Madison
adsmith@cs.wisc.edu sinclair@cs.wisc.edu

I. Introduction

Power consumption has increasingly become a first-class design constraint [1] to satisfy requirements for scientific workloads and other widely used workloads, such as machine learning. To meet performance and power requirements, system designers often use architectural simulators, such as gem5, to model component and system-level behavior. However, performance and power modeling tools are often isolated and do not make it accessible to integrate with one another for rapid performance and power system co-design. Although studies have previously explored power modeling with gem5 and validation on real hardware [2], there are several flaws with this approach. First, power models are sometimes not open source, making it difficult to apply them to different simulated systems. The current interface for implementing power models in gem5 also relies on hard-coded strings provided by the user to model dynamic and static power. This makes defining power models for components cumbersome and restrictive, as gem5's `MathExpr` string formula parser has support for limited mathematical operations. Third, previous works only implement one form of power model for one component. This unnecessarily limits users from combining other power models, which may model certain system components with higher accuracy. Instead, we posit that decoupling how power models are integrated with simulators from the design of power models themselves will enable better power modeling in simulators. Accordingly, we extend our prior work on designing and implementing an extensible, generalizable power modeling interface [3] by integrating support for McPAT [4] into it and validating it emits correct power values.

II. Implementation and Methodology

Our implementation introduces a new SimObject class which accepts a Python function instead of a string equation to model power consumption. This enables users to define a power model with more flexible, customized behavior than the current `MathExpr` API. For example, `MathExpr` does not support nesting functions and only supports operations that can be expressed in a string. This interface enables users to integrate any power model which best suits their needs, including pre-existing power models or power models they have created.

Methodology: To demonstrate this flexibility, we implemented McPAT into this interface for the in-order TimingSimple and Minor CPUs and the out-of-order (O3) CPU, each with private L1 data and instruction caches and a shared L2 cache. Next, to validate our implementation, we selected a handful of accessible benchmarks including Hello World, IAX, SAX, IAXPY, SAXPY, and DAXPY. For each benchmark, we validated their new gem5 power results against those from McPAT for all 3 CPUs. Although these benchmarks have natural regions of interest (ROIs), we model the power for the entire program due to issues with m5ops improperly resetting statistics.

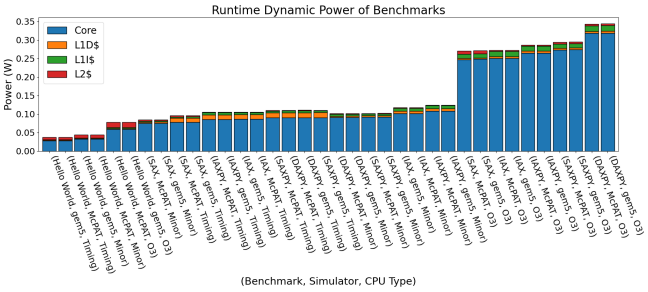

Fig. 1: Power modeling results. Each tick is in the form of (Benchmark, Simulator, CPU Type)

Results: Figure 1 shows our implemented gem5 McPAT support has nearly identical power consumption to standalone McPAT. Although most results follow expected patterns (e.g., SAX consumes less energy than SAXPY), several do not. For example, DAXPY does not always consume more energy than SAXPY, despite the increased precision in DAXPY, because McPAT does not properly model the difference between single and double precision operations. Similarly, some instructions are categorized as vector instructions, which McPAT does not support – causing instances such as IAXPY to report less power than IAX under Timing. Nevertheless, since our integrated support mirrors that of McPAT, these issues are not specific to our integration.

III. Conclusion and Future Work

To overcome the limitations of gem5's current power modeling API, we implemented a new power modeling interface in gem5. To demonstrate the potential of our approach, we implemented and are releasing this support, which we validated against the standalone tool. Thus, this support makes co-designing early-stage designs for both performance and power more practical for gem5's users, and makes power models accessible to computer architecture researchers. Moving forward, we are extending this support to model power [5], [6] for gem5's GPU models [7]–[10].

ACKNOWLEDGMENTS

This work is supported in by the Semiconductor Research Corporation grant 3151.001, National Science Foundation grant ENS-1925485, and by the DOE's Office of Science, Office of Advanced Scientific Computing Research through EXPRESS: 2023 Exploratory Research for Extreme Scale Science.

REFERENCES

[1] T. Mudge, "Power: A First-class Architectural Design Constraint," *Computer*, vol. 34, no. 4, pp. 52–58, 2001.

[2] B. K. Reddy, M. J. Walker, D. Balsamo, S. Diestelhorst, B. M. Al-Hashimi, and G. V. Merrett, "Empirical CPU Power Modelling and Estimation in the gem5 Simulator," in *27th International Symposium on Power and Timing Modeling, Optimization and Simulation*, ser. PATMOS, 2017, pp. 1–8.

[3] A. Smith, B. Bruce, J. Lowe-Power, and M. D. Sinclair, "Designing Generalizable Power Models For Open-Source Architecture Simulators," in *3rd Open-Source Computer Architecture Research Workshop*, ser. OSCAR, 2024.

[4] S. Li, J. H. Ahn, R. D. Strong, J. B. Brockman, D. M. Tullsen, and N. P. Jouppi, "McPAT: An Integrated Power, Area, and Timing Modeling Framework for Multicore and Manycore Architectures," in *Proceedings of the 42nd annual IEEE/ACM International Symposium on Microarchitecture*, ser. MICRO, 2009, pp. 469–480.

[5] V. Kandiah, S. Peverelle, M. Khairy, A. Manjunath, J. Pan, T. G. Rogers, T. M. Aamodt, and N. Hardavellas, "AccelWattch: A Power Modeling Framework for Modern GPUs," in *Proceedings of the 54th IEEE/ACM International Symposium on Microarchitecture*, ser. MICRO, October 2021.

[6] P. Delestrac, J. Miquel, D. Bhattacharjee, D. Moolchandani, F. Catthoor, L. Torres, and D. Novo, "Analyzing GPU Energy Consumption in Data Movement and Storage," in *IEEE 35th International Conference on Application-specific Systems, Architectures and Processors*, ser. ASAP. IEEE, 2024, pp. 143–151.

[7] A. Gutierrez, B. M. Beckmann, A. Dutu, J. Gross, M. LeBeane, J. Kalamatianos, O. Kayiran, M. Poremba, B. Potter, S. Puthoor, M. D. Sinclair, M. Wyse, J. Yin, X. Zhang, A. Jain, and T. Rogers, "Lost in Abstraction: Pitfalls of Analyzing GPUs at the Intermediate Language Level," in *2018 IEEE International Symposium on High Performance Computer Architecture*, ser. HPCA, Feb 2018, pp. 608–619.

[8] V. Ramadas, D. Kouchekinia, N. Osuji, and M. D. Sinclair, "Closing the Gap: Improving the Accuracy of gem5's GPU Models," in *5th gem5 Users' Workshop*, June 2023.

[9] V. Ramadas, M. Poremba, B. M. Beckmann, and M. D. Sinclair, "Simulation Support for Fast and Accurate Large-Scale GPGPU and Accelerator Workloads," in *3rd Open-Source Computer Architecture Research Workshop*, ser. OSCAR, 2024.

[10] K. Roarty and M. D. Sinclair, "Modeling Modern GPU Applications in gem5," in *3rd gem5 Users' Workshop*, June 2020.