2021 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS) | 978-1-7281-8643-6/21/$31.00 ©2021 IEEE | DOI: 10.1109/ISPASS51385.2021.00042

Architecture-Level Energy Estimation for Heterogeneous Computing Systems

Francis Wang Yannan Nellie Wu Matthew Woicik Joel S. Emer Vivienne Sze
Massachusetts Institute of Technology
{frwang, nelliewu, mwoicik, jsemer, sze}@mit.edu

***Abstract*—Due to the data and computation intensive nature of many popular data processing applications, *e.g.*, deep neural networks (DNNs), a variety of accelerators have been proposed to improve performance and energy efficiency. As a result, computing systems have become increasingly heterogeneous, with application-specific processing offloaded from the CPU to specialized accelerators. To understand the energy efficiency of such systems, it is desirable to characterize holistically the energy consumption of the CPU, the accelerator, and the data transfers in between. We present a modularized architecture-level energy estimation framework that captures the energy breakdown across the various CPU and accelerator components with a unified energy estimation back-end that allows easy integration of accelerator modeling frameworks for emerging designs. Using DNN workloads as examples, we show that CPU-end preprocessing and data transfers to and from the accelerator can account for up to 45-50% of total energy when assessing the system as a whole. Related open-source code is available at https://accelergy.mit.edu.**

***Index Terms*—Architecture-Level Estimation, SoC Energy Estimation, Deep Neural Network Accelerators**

I. Introduction

Many popular workloads, such as deep neural networks (DNNs) and graph algorithms, are computation heavy and have well defined memory access patterns with abundant data reuse. This makes them attractive candidates for hardware acceleration [1]–[5]. When processing such workloads, system-on-chip (SoC) platforms often offload the computation and memory intensive portions (*e.g.*, the convolutional and fully connected layers of a DNN) to specialized accelerators, which can achieve higher performance and energy efficiency than the CPU. It is important for designers to have a holistic understanding of the entire heterogeneous system by considering the energy expended by the CPU, the domain-specific accelerator, and the data transfer in between. There are existing works that provide early stage architecture-level energy estimations for such heterogeneous systems [6]–[8]. However, they either assume a simplified CPU-end setup for a limited class of workloads (*e.g.*, DNNs coded in TensorFlow [8]), and/or limited accelerator modeling capabilities (*e.g.*, hard-coded mapping search for a fixed accelerator architecture or limited hardware energy characterizations [6]–[8]). To address this problem, we propose a modularized evaluation framework that supports the modeling of heterogeneous systems for arbitrary classes of workloads, including new and upcoming application domains. The framework enables interactions between three existing evaluation frameworks, Gem5 [9], Timeloop [10], and Accelergy [11], each providing comprehensive modeling of its target platform.

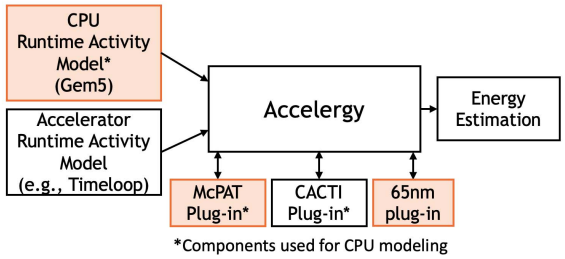

Fig. 1: High-level infrastructure with runtime activity modeling frameworks and a unified energy estimation back-end Accelergy [11]. Extensions to existing frameworks are shaded. Detailed Gem5-Accelergy setup shown in Figure 3. Detailed Timeloop-Accelergy setup similar to [12].

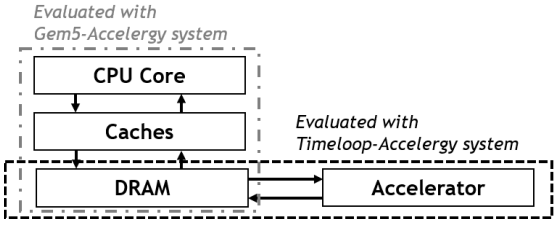

Fig. 2: High-level illustration of how a simple CPU-accelerator system can be evaluated with the setup in Figure 1.

II. Estimation Framework

Figure 1 shows the high level organization of the estimation framework. Gem5 [9] is used to model the CPU, Timeloop [10], [13] is used as an example of an accelerator modelling framework, and Accelergy [11] serves as a unified energy estimation back-end. Accelergy is able to support energy estimation plug-ins from various sources, each responsible for different components of the system. Figure 2 shows how a simple CPU-accelerator system can be evaluated with the proposed setup.

The Gem5-Accelergy setup is shown in Figure 3. We use the Gem5 simulator in full system mode to model the micro-architectural activity of CPU workloads. This allows us to characterize arbitrary CPU programs and takes into account the overhead of the operating system and system calls.

978-1-7281-8643-6/21/$31.00 ©2021 IEEE
DOI 10.1109/ISPASS51385.2021.00042

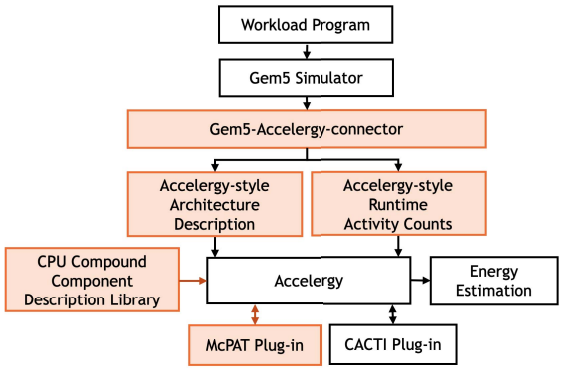

Fig. 3: Gem5-Accelergy setup for CPU energy estimation. Extensions to existing frameworks are shaded.

Gem5 generates the runtime activity counts for the various components (*e.g.*, caches, the reorder buffer, and various ALU units). It also outputs an architectural description of the organization of the high-level components and their associated hardware attributes (*e.g.*, buffer sizes, bus widths). To support interactions between Gem5 and Accelergy, we implemented the gem5-accelergy connector which converts gem5 outputs to Accelergy-style architectural descriptions and activity counts. To describe the high-level components in the CPU architectures, we created a library of YAML-based component descriptions. These can be easily updated if the user would like to add or modify components. For CPU component energy estimations, we created an energy estimation plug-in based on McPAT [14] to determine the energy cost of microarchitectural activities (*e.g.*, an instruction fetch). Timeloop allows us to model the energy consumption of tensor accelerators (*e.g.*, GEMM and DNN accelerators) by performing an extensive mapping space search. More details on the Timeloop and Timeloop-Accelergy interactions can be found in [10] and [12]. Note that although we use Timeloop as an example accelerator modeling framework, different frameworks can easily be plugged in to replace Timeloop to model other accelerator designs for different application domains.

III. Experimental Results

We validated the Gem5-Accelergy system on [15]. As shown in Figure 4, both the total energy consumption and the component energy breakdowns closely match with the ground truth. Next, we performed a case study to evaluate the energy consumption of DNN inference workloads (AlexNet [16] and ResNet-18 [17]) on a heterogeneous system that consists of an out-of-order CPU core and the DNN accelerator Eyeriss [1]. Specifically, data preprocessing (*i.e.*, normalizing the input image) and pooling layers are performed with PyTorch on the CPU while convolutional and fully-connected layers are performed on Eyeriss. Data is transferred between the CPU and the accelerator by reading and writing from the DRAM. Figure 5 shows the energy breakdown by layer type in the case where all of the computations are done on the CPU

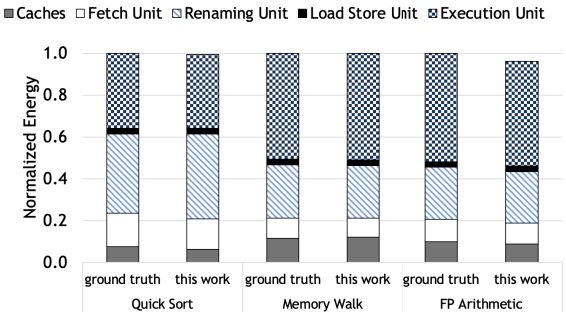

Fig. 4: Gem5-Accelergy energy estimation validation on various workloads.

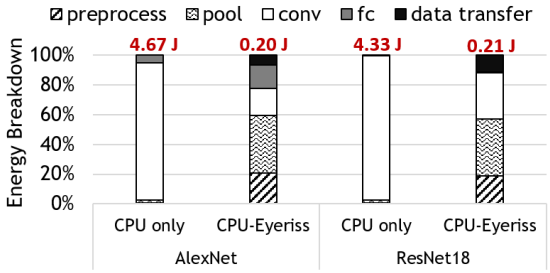

Fig. 5: Energy breakdown for CPU-only and CPU-Accelerator workloads for AlexNet and ResNet-18. We use Eyeriss [1] as the accelerator design. Total energy consumption is labeled at the top of each bar.

and when computations are offloaded to the accelerator. As the accelerator is much more energy efficient than the CPU, components of energy consumption that once were dwarfed by the convolutional layers such as data preprocessing and pooling now become significant. The energy cost of data transfers through the DRAM is only present when offloading to the accelerator. This also consumes a significant portion of total energy. In both cases, processing of the convolutional and fully connected layers only consumes about a third of total energy, down from about 97%.

IV. Conclusion

This work presents a generally applicable architecture-level energy estimation framework to understand the component energy breakdowns in heterogeneous computing systems. The unified energy estimation back-end and modularity of the framework allows easy integration of various accelerator modeling frameworks. This framework is well suited to modeling use cases where some parts of the workload are offloaded to an accelerator while other parts remain on the CPU. With our DNN case study, we have shown that it is insufficient to look at just the energy consumption of the accelerator. Even if most of the computation is offloaded to the accelerator, CPU processing and data transfers contribute significantly to total system energy and should be taken into account in a holistic evaluation of the SoC.

REFERENCES

[1] Y.-H. Chen, T. Krishna, J. Emer, and V. Sze, "Eyeriss: An Energy-Efficient Reconfigurable Accelerator for Deep Convolutional Neural Networks," in *2016 IEEE Int. Solid-State Circuits Conf. (ISSCC)*, 2016.

[2] Y.-H. Chen, J. Emer, and V. Sze, "Eyeriss: A spatial architecture for energy-efficient dataflow for convolutional neural networks," in *2016 Int. Symp. Comp. Arch. (ISCA)*, 2016.

[3] T.-J. Yang, Y.-H. Chen, J. Emer, and V. Sze, "A method to estimate the energy consumption of deep neural networks," in *2017 Asilomar Conf. on Signals, Systems, and Computers*, 2017.

[4] Y.-H. Chen, T.-J. Yang, J. Emer, and V. Sze, "Eyeriss v2: A Flexible Accelerator for Emerging Deep Neural Networks on Mobile Devices," *IEEE J. on Emerging and Selected Topics in Circuits and Systems*, 2019.

[5] K. Hegde, H. Asghari-Moghaddam, M. Pellauer, N. Crago, A. Jaleel, E. Solomonik, J. S. Emer, and C. Fletcher, "ExTensor: An accelerator for sparse tensor algebra," in *2019 IEEE/ACM Int. Symp. Microarch. (MICRO)*, 2019.

[6] Y. S. Shao, S. L. Xi, V. Srinivasan, G. Wei, and D. Brooks, "Co-designing accelerators and soc interfaces using gem5-aladdin," in *2016 IEEE/ACM Int. Symp. Microarch. (MICRO)*, 2016.

[7] S. L. Xi, Y. Yao, K. Bhardwaj, P. Whatmough, G.-Y. Wei, and D. Brooks, "Smaug: End-to-end full-stack simulation infrastructure for deep learning workloads," *ACM Trans. Archit. Code Optim.*, vol. 17, no. 4, Nov. 2020.

[8] N. Bohm Agostini, S. Dong, E. Karimi, M. Torrents Lapuerta, J. Cano, J. L. Abellán, and D. Kaeli, "Design space exploration of accelerators and end-to-end dnn evaluation with tflite-soc," in *2020 IEEE Int. Symp. on Comp. Arch. and High Performance Comp. (SBAC-PAD)*, 2020, pp. 10–19.

[9] N. Binkert, B. Beckmann, G. Black, S. K. Reinhardt, A. Saidi, A. Basu, J. Hestness, D. R. Hower, T. Krishna, S. Sardashti, R. Sen, K. Sewell, M. Shoaib, N. Vaish, M. D. Hill, and D. A. Wood, "The gem5 simulator," *SIGARCH Comput. Archit. News*, vol. 39, no. 2, p. 1–7, Aug. 2011.

[10] A. Parashar, P. Raina, Y. S. Shao, Y.-H. Chen, V. A. Ying, A. Mukkara, R. Venkatesan, B. Khailany, S. W. Keckler, and J. Emer, "Timeloop: A Systematic Approach to DNN Accelerator Evaluation," in *2019 Int. Symp. on Perf. Analysis of Systems and Software (ISPASS)*, 2019.

[11] Y. N. Wu, J. S. Emer, and V. Sze, "Accelergy: An Architecture-Level Energy Estimation Methodology for Accelerator Designs," in *2019 Int. Conf. on Computer-Aided Design (ICCAD)*, 2019.

[12] Y. N. Wu, V. Sze, and J. S. Emer, "An Architecture-Level Energy and Area Estimator for Processing-in-Memory Accelerator Designs," in *2020 Int. Symp. on Perf. Analysis of Systems and Software (ISPASS)*, 2020.

[13] Y. N. Wu, P.-A. Tsai, A. Parashar, V. Sze, and J. S. Emer, "Sparseloop: An Analytical, Energy-Focused Design Space Exploration Methodology for Sparse Tensor Accelerators," in *2021 Int. Symp. on Perf. Analysis of Systems and Software (ISPASS)*, 2021.

[14] 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 *2009 IEEE/ACM Int. Symp. Microarch. (MICRO)*, 2009.

[15] F. A. Endo, D. Couroussé, and H.-P. Charles, "Micro-architectural simulation of embedded core heterogeneity with gem5 and mcpat," in *2015 Workshop on Rapid Sim. and Perf. Eval. (RAPIDO)*, 2015.

[16] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *25th Int. Conf. on Neural Information Processing Systems (NIPS)*, 2012.

[17] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conf. on Comp. Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778.