A Power-Aware Heterogeneous Architecture Scaling Model for Energy-Harvesting Computers

Harsh Desai
harshd@andrew.cmu.edu
Brandon Lucia
blucia@andrew.cmu.edu
Carnegie Mellon University
https://abstract.ece.cmu.edu

Abstract—Energy-harvesting devices are the key to enabling future ubiquitous sensing applications, because they are long lived and require little maintenance. On-device processing of sensed data, such as images, avoids the high energy cost of communicating data to the edge or cloud. This work observes that the on-device computing performance of an energy-harvesting system depends not only on execution time, but also on energy collection time. With high input power, a faster, higher-power processor quickly completes processing because energy collection time is low. At low input power, a slower, more energy efficient processor minimizes end-to-end latency by more judiciously using the slowly collected energy. This paper describes the PHASE model, which captures this charge latency effect. Using the model, we develop PHASE architectures, which include heterogeneous processing components of different efficiency and performance. A PHASE architecture uses the combination of heterogeneous components that minimizes end-to-end latency, including recharge time. Our results show that the PHASE model helps understand end-to-end latency in an energy harvesting device, yielding PHASE architectures that complete up to $9\times$ more work on a fixed energy budget than typical energy-harvesting architecture.

Index Terms—Energy-harvesting, Heterogenous Architectures, Accelerators, Power-aware Computing.

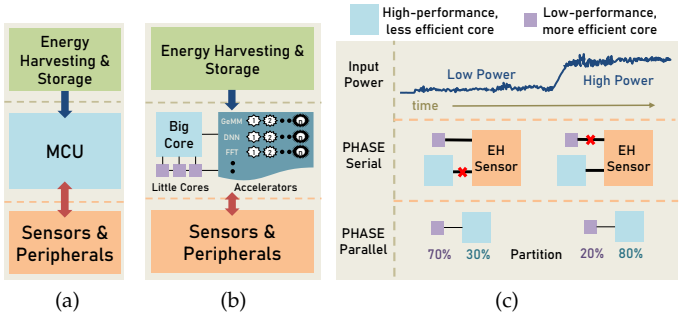
1 Introduction

Future deeply-embedded applications such as civil infrastructure and wildlife monitoring will depend on sensor deployments that scale to billions or trillions of devices and operate sustainably with little maintenance over long time periods. Energy-harvesting devices meet the requirements of these applications by integrating small, cheap energy collection and storage elements, with communication, computing, and sensing components. Avoiding the need for battery replacements, energy harvesting devices allow for long-term deployments with little maintenance. However, the performance of such systems depends on energy availability in the device's environment, warranting special attention from computer architects that is typically not given to today's energy-harvesting computer systems.

The architecture of a typical energy-harvesting device is shown in Figure 1a: a low-power microcontroller (MCU) connected to peripherals and a power system. Energy-harvesting components collect and store energy from sources, such as radio waves (RF), solar, or thermal gradients. The system collects and stores sensor data, processing it using its MCU, which may include basic linear algebra acceleration [1], and may in the future include more sophisticated vector units [2] or accelerators [3], [4].

Since communication typically dominates *power* consumption, recent work [2], [5] presents increasingly complex on-device computation (e.g., ML inference) to avoid transmitting uninteresting data. As communication is less frequent, computing energy becomes comparable to communication energy, requiring efficient on-device computing.

Computation on an energy-harvesting device is driven by collected energy. A device may collect energy up front, before executing and run without interruption. Alternatively, a device may collect small amounts of energy at a time and operate intermittently [6], relying on checkpoints [6]–[9] or tasks [10]–[12] to ensure correctness and progress across power interruptions. Energy collection time is fundamentally limited by the input power available in the environment. If power is high, the device charges quickly, spending less time charging and more time computing. If power is low, the device charges slowly, spending more time collecting energy and less time computing.



(a) (b) (c)

Fig. 1: (a) shows a typical energy-harvesting architecture, with a simple processing core. (b) shows our PHASE architecture with heterogeneous single-workload accelerators and cores. (c) shows how PHASE-serial and PHASE-parallel architectures select fastest architectural configuration depending on input power.

We observe that end-to-end latency for a fixed workload depends not only on compute time, but also recharge time. Consequently, the selection of which architecture mimimizes end-to-end latency depends not only its raw performance, but also on the efficiency with which it uses the energy collected over time. Moreover, the best choice of architecture varies dynamically as input power varies: higher power needs higher performance, lower power needs higher efficiency. Energy-harvesting architectures should use heterogeneity to vary performance and energy efficiency to optimize end-to-end latency.

We present PHASE, a class of heterogeneous architectures

and a framework for modeling their end-to-end performance on harvested energy. PHASE helps understand performance across power environments, enabling architects and system designers to optimize for a workload's end-to-end latency. PHASE additionally helps understand the optimal dynamic partitioning of an application across parallel computing resources in an energy-harvesting system. We validate PHASE against real-world accelerator architectures, showing that PHASE optimizes end-to-end workload latency with power-dependent core switching, increasing the total work done by up to $9\times$. PHASE also provides optimal parallel partitioning, improving end-to-end latency by up to $10\times$ compared to static partitioning.

2 Performance Model for Energy-Harvesting Systems

PHASE analytically models the end-to-end latency of an arbitrary computational workload on an energy-harvesting computer system. Unlike a system with continuously available power, the end-to-end latency of an energy-harvesting system (t_{e2e}) includes *recharge latency* (t_{recharge}), which depends on execution time (t_{exec}), input power (P_{input}), and power consumption (P_{exec}):

$$\begin{aligned} t_{\text{e2e}} &= t_{\text{exec}} + t_{\text{recharge}} \\ &= t_{\text{exec}} + \frac{E_{\text{exec}}}{P_{\text{input}}} \\ &= t_{\text{exec}} + \frac{t_{\text{exec}} \times P_{\text{exec}}}{P_{\text{input}}} \end{aligned} \tag{1}$$

Equation (1) shows that *input power* dictates performance by governing recharge latency.

This basic equation models a situation where workload execution and recharging are not simultaneous. Most systems recharge energy as they execute [13], [14]:

$$t_{\text{e2e}} = max(t_{\text{exec}}, \frac{t_{\text{exec}} \times P_{\text{exec}}}{P_{\text{input}}}) \tag{2}$$

The end-to-end latency depends on execution time and recharging time. Input power determines which one of these parameters dominates t_{e2e}.

The Need for Single-workload Heterogeneous Architectures Our PHASE model shows that energy-constrained systems should be heterogeneous to improve performance. As input power changes, a system should adapt its architecture accordingly. At high input power, execution time matters most and the system should optimize with a high performance architecture, even if at a cost in energy. At low input power, recharge time matters most; the system should prioritize efficiency, even if at a cost in execution time.

3 Heterogeneous PHASE Architectures for Energy-Constrained Systems

Using our model, we present PHASE architectures, which are systems that have *single-workload heterogeneity*, incorporating many architectural components (e.g., processors, accelerators, ASICs) capable of performing the *same* computation with different performance and efficiency. A PHASE architecture monitors input power and chooses the architecture that minimizes latency at a power level. A system could, for example, include heterogeneous CPUs [15], could vary core settings (e.g., DVFS), and could use ASICs [16] or emerging core microarchitectures [2], [17].

We envision two types of PHASE architectures. The first type is *PHASE-serial*, in which a single component runs a workload at a time, depending on input power. Components share last-level cache, enabling efficient switching between component architectures without data movement (assuming the same input format).

The second type is *PHASE-parallel*, which partitions a workload to run in parallel on heterogeneous components. Partitioning a workload and mapping it onto parallel components is a key research challenge of PHASE-parallel operation. Figure 1c is a sketch of a PHASE architecture.

A challenge faced by both types of PHASE architectures is deciding when and how often to switch between the architectural components (like energy-aware scheduling [7]). In image-processing workloads (e.g., a smart camera [5], [18]), for example, the system could monitor input power and assess whether to switch at each new image. Translating between different components' input formats is another key research challenge.

The PHASE model described in Section 2 is directly applicable to a PHASE-serial system: to decide whether to switch, the system evaluates the end-to-end latency of each architectural component for the current input power, choosing the fastest option. We next extend the PHASE model to PHASE-parallel architectures, enabling power-aware distribution of parallel work across architectural components with differing performance and efficiency.

3.1 Modeling PHASE-Parallel Architectures

In a PHASE-parallel architecture, the increased energy recharging cost of parallelism must not outweigh the parallel speedup. PHASE models the effect of parallelization on end-to-end latency, as the input power varies.

Figure 1b shows an example of a PHASE architecture, where we have one *big* core, several *small* cores, and a set of application-specific accelerators. For a system with n architectural components, in which the i^{th} component runs $\alpha(i)$ fraction of the parallelizable workload, (2) becomes:

$$\begin{aligned} t_{\text{e2e}} &= max(t_{\text{serial\_exec}} + t_{\text{parallel\_exec}}, t_{\text{recharge\_total}}) \\ &= max\{t_{\text{serial\_exec}} + max\left[\alpha(i) * t_{\text{parallel\_exec}}(i)\right]_{i=0}^{n}, \\ &\quad \frac{E_{\text{serial\_exec}} + \sum_{i=0}^{n} \alpha(i) * E_{\text{parallel\_exec}}(i)}{P_{\text{input}}}\} \end{aligned} \tag{3}$$

Here, $t_{\text{serial\_exec}}$ and $E_{\text{serial\_exec}}$ are the latency and energy costs of the serial part of the workload. $t_{\text{parallel\_exec}}(i)$ and $E_{\text{parallel\_exec}}(i)$ represent the latency and energy cost of the i^{th}-architectural component running the entire parallel part. The choice of component to run the serial part of the workload is a (simple) *PHASE-serial* choice. Equation (3) shows that end-to-end latency in a PHASE-parallel architecture depends on choosing the optimal workload partitioning, α, which in turn depends on input power.

4 Results & Discussions

We evaluated PHASE in simulation to show that power-dependent core switching in PHASE-serial optimizes latency and increases the total work done in a fixed amount of time. We also show that dynamically partitioning parallel work according to input power in a PHASE-parallel architecture yields speedup over a power-oblivious, fixed partitioning.

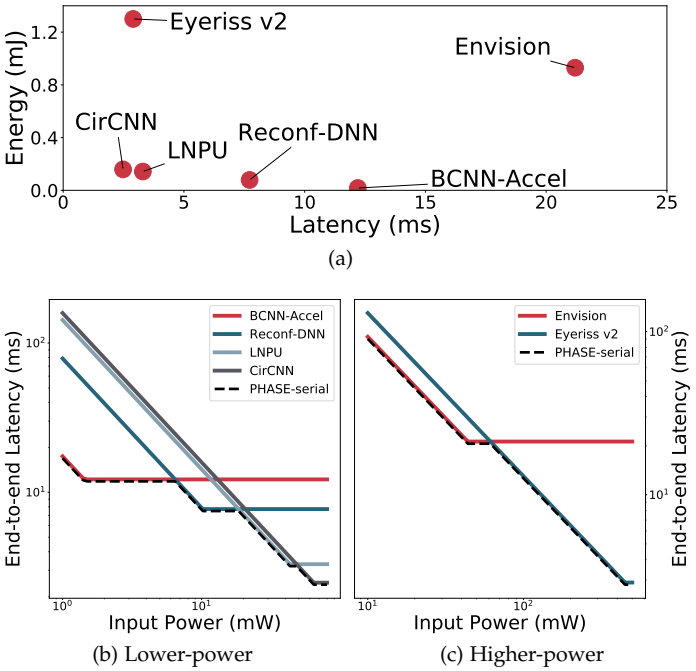


Fig. 2: (a) Energy vs Latency for different DNN accelerators. (b) and (c) End-to-end inference latency vs input power for different accelerators.

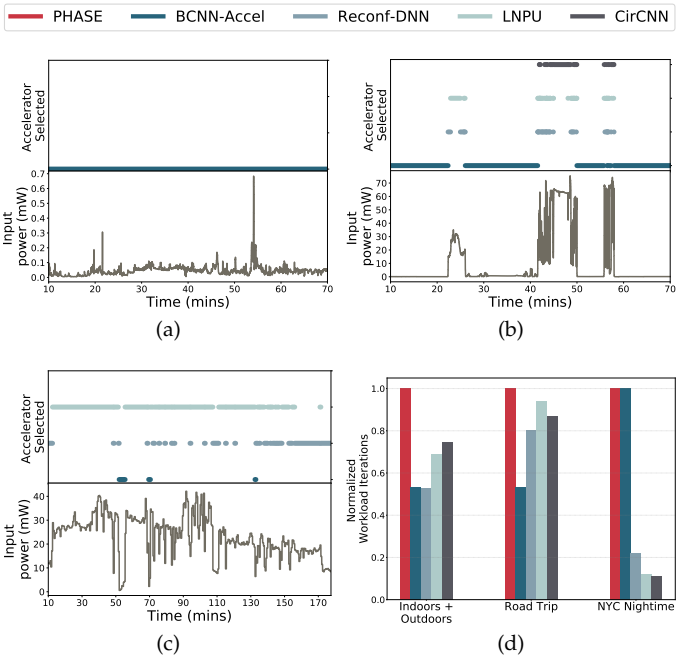


Fig. 3: Timeline of our simulation on three different input traces: (a)Indoors+Outdoors, (b) Road trip and (c) NYC nighttime. (d) shows the total iterations of AlexNet inference completed by the different systems, normalized to the PHASE-serial architecture.

4.1 PHASE-serial Architecture Improves Latency

We show that PHASE-serial outperforms a fixed architecture on AlexNet inference [19] under variable input power. Figure 2a shows the energy and latency for a single AlexNet inference on several different architectures, taken from their publications [3], [4], [20]–[23]. Using the analytical model in Section 2, we evaluate two PHASE-serial architectures: one switching between four lower-power DNN accelerators [20]–[23], and one with the two higher-power accelerators [3], [4]. PHASE-serial selects the accelerator with the lowest end-to-end latency for a given power level. We then compare our PHASE-serial architecture with systems based on each of the accelerators individually.

Figures 2b and 2c plot end-to-end inference latency for these PHASE-serial architectures and for each individual accelerator as power varies. The PHASE-serial architectures switch accelerators depending on the input power level, netting a $10\times$ improvement in end-to-end latency at low power levels for the low-power accelerators 2b, and a similar benefit for the high-power accelerators 2c. The data are an upper bound on performance, ignoring work migration overheads and granularity, but show the promise of PHASE-serial heterogeneity.

4.2 PHASE Increases Efficiency

PHASE increases the total work completed in a fixed amount of time and energy compared to a baseline oblivious to input power. To validate this, we ran a discrete-event simulation counting total AlexNet inference completions using the low-power PHASE-serial architecture shown in Figure 2b, using our empirical AlexNet energy and latency values in simulation. To simulate input power, we model a solar panel with $10cm^2$ area and 20% efficiency and real irradiance traces from the EnHANTs dataset [24]. We simulated three traces: Indoors+Outdoors, Roadtrip, and New York at Night. Across all three traces, we compared our PHASE-serial architecture with systems based on each accelerator individually, counting the total AlexNet iterations completed in the entire trace duration.

Figure 3(a–c) show that PHASE-serial minimizes end-to-end latency as power varies. Figure 3(d) compares PHASE-serial with each accelerator individually, showing AlexNet inferences completed for each irradiance trace. PHASE-serial completes up to $9\times$ more work with low power (e.g., CircCNN, NYC Night) and $2\times$ more work with high power (e.g., BCNN).

4.3 PHASE Optimally Partitions Parallel Work

Equation (3) shows that end-to-end latency for a parallel workload depends on both the parallel partitioning (α) and input power. Accordingly, as input power varies, the value of α that achieves the minimum end-to-end latency also varies. PHASE-parallel architectures dynamically vary α with input power to optimize end-to-end latency, which we refer to as optimal parallel partitioning. Figure 4 models PHASE-parallel's partitioning. The plots assume two heterogeneous cores, A and B. The data compare PHASE's dynamic, input-dependent partitioning to static partitioning. Each architecture's A and B have a different relative power consumption and latency to complete the modeled workload. The plots show the optimal parallel partition on the left y-axis, solid curve, and the resultant end-to-end latency on the right y-axis, dashed curve. The data show that the latency of different partitions varies with power *by an order of magnitude*. Figure 4c shows speedup, using the PHASE model to optimally partition versus a fixed partition for several different power and latency ratios. The data show that PHASE's optimal partitioning translates into a $1.6 - 10\times$ speedup compared to fixed partitioning.

5 Conclusion

In this paper, we presented PHASE, an architecture and performance model for heterogeneous energy-harvesting systems.

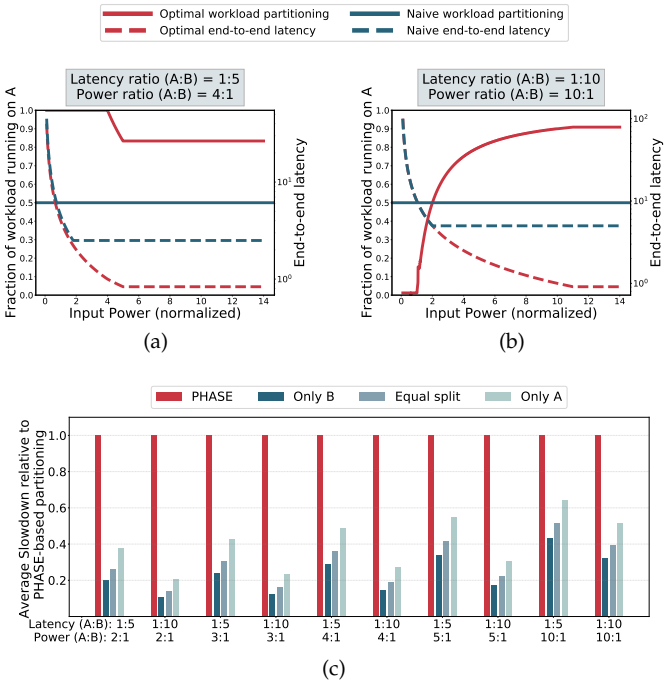


Fig. 4: (a) and (b) compare the end-to-end workload latencies of two PHASE-parallel architectures with a fixed naive $50:50$ partitioning, for different input power levels. (c) shows the average slowdown of different naive partitionings compared to our PHASE-based partitioning for different PHASE-parallel architectures.

The PHASE model shows that input power dictates performance in energy-harvesting systems because recharging takes time. Based on the PHASE model, our architectures, PHASE-serial and PHASE-parallel leverage heterogeneity to choose a configuration of architectural components that minimize latency for a given input power level. Our evaluation shows that a PHASE-serial multi-accelerator architecture can perform up to $9\times$ the work done by a single-accelerator system. Further, the optimal partitioning of work in PHASE-parallel systems provides up to $10\times$ speedup over naive, fixed partitionings. We envision that the PHASE model will guide researchers in defining future, high-performance energy-harvesting systems. Future work should explore efficient adaptation in PHASE architectures, including support to translate between input formats and to schedule architectural reconfiguration.

Acknowledgments

We thank the anonymous reviewers and members of the Abstract research group at CMU for their feedback on this work. We thank Nathan Beckmann for his input to our performance model. This work was funded by National Science Foundation Award #1815882.

References

[1] Texas Instruments, "Low Energy Accelerator." http://www.ti.com/lit/ds/symlink/msp430fr5994.pdf, 2020.

[2] G. Gobieski *et al.*, "Manic: An energy-efficient, parallel architecture for ultra-low-power embedded systems," in *Proceedings of International Symposium on Microarchitecture*, 2019.

[3] Y. Chen, T. Yang, J. Emer, and V. Sze, "Eyeriss v2: A flexible accelerator for emerging deep neural networks on mobile devices," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 9, pp. 292–308, June 2019.

[4] B. Moons, R. Uytterhoeven, W. Dehaene, and M. Verhelst, "14.5 envision: A 0.26-to-10tops/w subword-parallel dynamic-voltage-accuracy-frequency-scalable convolutional neural network processor in 28nm fdsoi," *2017 IEEE International Solid-State Circuits Conference (ISSCC)*, pp. 246–247, 2017.

[5] M. Nardello *et al.*, "Camaroptera: A batteryless long-range remote visual sensing system," in *Proceedings of the Workshop on Energy Harvesting & Energy-Neutral Sensing Systems*, (New York, NY, USA), p. 814, Association for Computing Machinery, 2019.

[6] B. Lucia and B. Ransford, "A simpler, safer programming and execution model for intermittent systems," in *Proceedings of the Conference on Programming Language Design and Implementation*, PLDI '15, (New York, NY, USA), ACM, 2015.

[7] K. Maeng and B. Lucia, "Supporting peripherals in intermittent systems with just-in-time checkpoints," in *Proceedings of the Conference on Programming Language Design and Implementation*, PLDI 2019, (New York, NY, USA), pp. 1101–1116, ACM, 2019.

[8] M. Hicks, "Clank: Architectural support for intermittent computation," in *2017 ACM/IEEE 44th Annual International Symposium on Computer Architecture (ISCA)*, pp. 228–240, June 2017.

[9] D. Balsamo *et al.*, "Hibernus++: A self-calibrating and adaptive system for transiently-powered embedded devices," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 35, no. 12, pp. 1968–1980, 2016.

[10] E. Ruppel and B. Lucia, "Transactional concurrency control for intermittent, energy-harvesting computing systems," in *Proceedings of the 40th ACM SIGPLAN Conference on Programming Language Design and Implementation*, PLDI 2019, (New York, NY, USA), pp. 1085–1100, ACM, 2019.

[11] K. S. Yıldırım *et al.*, "Ink: Reactive kernel for tiny batteryless sensors," in *Proceedings of the 16th ACM Conference on Embedded Networked Sensor Systems*, pp. 41–53, ACM, 2018.

[12] J. Hester, K. Storer, and J. Sorber, "Timely execution on intermittently powered batteryless sensors," in *Proceedings of the 15th ACM Conference on Embedded Network Sensor Systems*, SenSys '17, (New York, NY, USA), pp. 17:1–17:13, ACM, 2017.

[13] N. Jackson *et al.*, "Capacity over capacitance for reliable energy harvesting sensors," in *Proceedings of the 18th International Conference on Information Processing in Sensor Networks*, IPSN '19, (New York, NY, USA), pp. 193–204, ACM, 2019.

[14] A. Colin, E. Ruppel, and B. Lucia, "A reconfigurable energy storage architecture for energy-harvesting devices," in *Proceedings of the Twenty-Third International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS '18, (New York, NY, USA), pp. 767–781, ACM, 2018.

[15] Arm, "big.LITTLE Technology." https://greensoft.cs.txstate.edu/index.php/2019/12/12/big-little-technology-the-future-of-mobile/, 2019.

[16] Y. S. Shao, B. Reagen, G.-Y. Wei, and D. Brooks, "Aladdin: A pre-rtl, power-performance accelerator simulator enabling large design space exploration of customized architectures," in *Proceeding of the 41st Annual International Symposium on Computer Architecuture*, ISCA 14, p. 97108, IEEE Press, 2014.

[17] T. Nowatzki *et al.*, "Stream-dataflow acceleration," in *Proceedings of the 44th Annual International Symposium on Computer Architecture*, ISCA 17, (New York, NY, USA), p. 416429, Association for Computing Machinery, 2017.

[18] G. Gobieski, B. Lucia, and N. Beckmann, "Intelligence beyond the edge: Inference on intermittent embedded systems," in *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS '19, (New York, NY, USA), pp. 199–213, ACM, 2019.

[19] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proceedings of the Conference on Neural Information Processing Systems*, NIPS12, pp. 84–90, Curran Associates Inc., 2012.

[20] C. Ding *et al.*, "Circnn: Accelerating and compressing deep neural networks using block-circulantweight matrices," *CoRR*, vol. abs/1708.08917, 2017.

[21] J. Lee *et al.*, "7.7 lnpu: A 25.3tflops/w sparse deep-neural-network learning processor with fine-grained mixed precision of fp8-fp16," in *2019 IEEE International Solid- State Circuits Conference - (ISSCC)*, pp. 142–144, Feb 2019.

[22] S. Yin, P. Ouyang, J. Yang, T. Lu, X. Li, L. Liu, and S. Wei, "An ultra-high energy-efficient reconfigurable processor for deep neural networks with binary/ternary weights in 28nm cmos," in *2018 IEEE Symposium on VLSI Circuits*, pp. 37–38, June 2018.

[23] S. Yin, P. Ouyang, S. Zheng, D. Song, X. Li, L. Liu, and S. Wei, "A 141 uw, 2.46 pj/neuron binarized convolutional neural network based self-learning speech recognition processor in 28nm cmos," in *2018 IEEE Symposium on VLSI Circuits*, pp. 139–140, June 2018.
[24] M. Gorlatova, A. Wallwater, and G. Zussman, "Networking low-power energy harvesting devices: Measurements and algorithms," in *Proc. IEEE INFOCOM'11*, Apr. 2011.