My research has been profoundly affected by two stints outside of academia: (1) working in the national interest at MIT Lincoln Laboratory and (2) starting Tilera (an early multicore company, founded before the term multicore was in popular use) with my then PhD advisor, Anant Agarwal, and my fellow students. While each experience was unique, I encountered the same common frustrations and the last 15 years of my research has been inspired by the challenges I faced in these jobs.

Lessons Learned at MIT Lincoln Laboratory and Tilera Corporation

Specifically, every system I have deployed in the "real-world" has required that my team and I find ways to meet multiple, conflicting goals. And ensure those goals are met despite unpredictable operating environments. For example, at Lincoln Lab we prototyped radar signal processing systems that demanded the performance of a supercomputer, with size and power constraints defined by what was effectively an unventilated utility closet on a US Navy cruiser. At Tilera, I had to help a customer concurrently transcode 20 HD video streams in a 30 Watt power envelope while minimizing their material cost.

Deploying computer systems that meet goals across multiple, conflicting metrics requires overcoming two core challenges. The first is *complexity*: hardware and software expose diverse, configurable parameters whose complicated interactions have non-linear effects on mission-critical metrics like latency, energy, accuracy, and security. The second is *dynamics*: computing systems must reliably adapt to unpredictable changes in operating environment, input workload, and even user needs.

Addressing Complexity and Dynamics with Self-aware Computing

To address these challenges, I have defined and developed the principle of *self-aware computing systems* [23, 24, 59]. The software and hardware systems I build are *aware* of their quantifiable goals—defined in terms of metrics like throughput, latency, power, energy, and accuracy. They continuously monitor critical metrics and adapt their internal behavior to ensure their goals are maintained in complex, dynamic environments. My self-aware systems combine machine learning—to address complexity [5–8, 16, 38, 56, 57, 60, 61, 81, 83]—and control theory—to handle dynamics [9, 11–13, 20, 26, 28, 37, 39–41, 48–50, 64, 76, 77, 80]. *One of my work's most significant benefits is combining learning with control so that deployed computing systems can learn at runtime, while still providing many of the formal guarantees of traditional control systems, making control theoretic benefits available to a wide range of computing systems* [21, 55, 59, 63, 84]. We recently designed a programming framework to make this functionality accessible to users who are not learning and control experts [2, 59]. Given the work's potential to fundamentally change users' interaction with computing systems, *Scientific American* named my self-aware computing model one of ten **World-Changing Ideas**.[1] Additionally, this work was honored with a Presidential Early Career Award for Scientists and Engineers (**PECASE**) in 2019.[2]

As a new graduate student I was told to pick one subfield and only publish there. As a (perhaps unhealthily) stubborn person, I ignored that well-intentioned advice. I felt the best way to show the value of self-aware computing was to apply it to as many different problems as possible. So my publications cover many subfields: I have used self-awareness to improve computing at the circuit-level [68], the architecture-level [5, 63, 65], the OS-level [11, 16, 26, 39, 55], the application-level [7, 12, 13, 28, 50, 76, 77, 80], and by coordinating across multiple layers of the system stack [9, 13, 20, 21, 50, 59, 84]. We have applied these techniques to manage energy on embedded micro-controllers [41], in collaboration with Argonne National Laboratory to manage power for the Theta supercomputer [52], and even for quantum computers [60].[3]

The Next Step: Controlling AI

The CS community and the world at large has been transformed by advances in AI. And of course, like all computing systems, deployed AI inference systems also must meet quantifiable goals including latency and energy, but also inference accuracy. I believe my prior research on **approximate computing and loop perforation** [11, 20, 21, 27, 28, 54, 62, 67] makes me well suited to study similar problems of dynamically managing AI inference to meet latency, energy, and accuracy goals in dynamic environments.[4] While this research is still in its early days, we have begun addressing two fundamental questions:

- First *how can we design and train neural networks that support dynamic (inference-time) tradeoffs between resource usage and accuracy?* The answer so far has been to structure the networks to maximize internal data reuse and use a novel orthoganlized stochastic gradient descent to balance accuracy across multiple network configurations [41, 74].
- Second, *how do we dynamically control these networks to create better computing systems?* Here the answer has been to combine the novel network structure from the first problem with novel control theoretic techniques designed for these adaptable neural networks [41, 75, 80].

The initial results are exciting. Working with scientists at Argonne National Laboratory, we have developed techniques that use control theory to guarantee the accuracy of ML inference in the context of material science simulations [80]. We have also used these techniques on embedded systems that harvest energy from the environment to maximize inference accuracy given a dynamically changing energy budget [41]. I am excited to find more uses for controllable AI going forward.

[1] Editors et al. "World-Changing Ideas: 10 new technologies that will make a difference". In: Scientific American 305.6 (Dec. 2011)

[2] The award citation was for "exceptionally innovative research on programming systems of power-constrained, complex exascale platforms, which will greatly increase user productivity, performance of High Performance Computing applications, and the rate of new scientific insights."

[3] Because so many computing systems are confronted with complexity, dynamics, or both, self-awareness naturally leads to many collaborations, and I am proud to have accumulated 300 co-authors in my pursuit of this work (as of January 2023 according to DBLP).

[4] Our work on Loop Perforation received an **FSE Test of Time Honorable Mention** in 2021.

The Rest of This Statement

The remainder of this statement describes my research on self-aware computing systems, including: (1) learning system models, (2) controlling system dynamics, and (3) combining learning and control to enable new capabilities. After describing these goals, I briefly review some earlier research projects and then elaborate my future plans.

Self-aware Computing Systems

Self-aware computing systems (1) monitor high-level quantifiable goals (like energy, latency, and accuracy), (2) have dynamically configurable parameters that affect those goals (e.g., clockspeed or alternative algorithms for accomplishing some task), and (3) automatically adapt those configurations so that their goals are met despite the challenges of complexity and dynamics [23, 24, 58]. To make this work widely accessible, I have developed programming frameworks that support the design and implementation of self-aware systems without requiring developers to be experts in learning or control [2, 59]

Learning Models of Complex Computer System Behavior

Until the mid-2000s software developers relied on computer architects to turn increasing transistor counts into increasing performance with no software changes. Due do the end of Dennard scaling and the physical limitations of power and energy dissipation, computer architects can no longer provide this "free ride." While additional hardware resources are available, they come at the cost of increased software complexity for determining how best to use those resources. Software systems are also increasingly configurable and require users to directly manage performance related parameters [45]. The interaction of these software and hardware parameters becomes incredibly complex and difficult to model.

My research has explored many ways to integrate machine learning into systems (a field now called ML for Systems). I mostly focus on resource management and scheduling, especially for power and energy [5–8, 16, 38, 56, 57, 60, 61, 81, 83]. However, there in one particular result that I would like to highlight: **Improving learning accuracy does not necessarily improve the system outcome** [6]. This was a somewhat counter intuitive result: we were using learned models to estimate a job's latency and energy for different resource assignments and then scheduling those resources to meet a job's target latency with minimal energy. We assumed that improving the learning accuracy would improve the energy efficiency, but even fairly large accuracy improvements did not reduce energy. After exploration, we determined the reason: this is a constrained optimization problem (meet a target latency with minimal energy) and, of course, only the resource configurations on the optimal frontier of power and performance are useful to solve this problem [18, 44]. Empirically, however, most resource configurations are not on the optimal frontier (typically we found about 90% are *not* optimal for any given task). Unfortunately, when the learning model is optimized for accuracy, it gets the biggest win by improving accuracy for these non-optimal configurations (since they are the majority), which does not improve the systems result (which only concerns optimal configurations, which are a small minority). Once we accounted for this *structure*, we designed a learning approach that was less accurate overall, but better for predicting the optimal configurations. Our less accurate learner gets 26% closer to optimal energy than the most accurate learner we studied. I believe this is an important observation for the emerging field of ML for systems: **the learning approaches should account for the structure of the systems problem they are deployed to solve rather than strictly optimize for accuracy.**

Controlling Computing Systems Through Dynamic Fluctuations

Control theory is a powerful branch of engineering for ensuring that systems meet goals in dynamic environments. Control theory represents a general set of techniques, but its implementations are almost always system-specific [47, 51]. While there is a rich history of applying control solutions to meet latency and throughput goals in computer systems, these solutions require laborious profiling and tuning and must be reimplemented or redesigned wholesale when ported to a new computer system. Essentially, the great bulk of control theory applied to computing requires engineers to be experts in both control and computing. Unlike this prior work, **my research has preserved the generality of control through into implementation, allowing developers with no control background to build systems with formal guarantees that their goals will be met.**

Specifically, I have generalized control solutions in two ways:

- **Automatically synthesizing controllers to manage computer system goals.** These include techniques for building computationally efficient single-input, single-output controllers [12], to techniques for creating hierarchies of controllers that can meet multiple goals with multiple configuration parameters [13, 20], to techniques that automatically synthesize model predictive control solutions [50]. Together, this work makes a range of control solutions available to non-experts, automating much of the most tedious and error-prone process of control design. These synthesis techniques, however, still require users to set some control-specific parameters.
- **Embedding control systems into existing computer systems.** As an alternative, I have explored techniques that hide control parameters from users by embedding them into systems below the user interface. As examples, I have embedded controllers into larger systems to manage approximate computing applications [28], to create portable, energy efficient embedded software [26, 37, 39], guarantee accuracy for machine learning inference in scientific simulations [80], meet energy budgets for embedded systems [41], manage bandwidth in network protocols for video analytics [9], and to automatically configure large scale software systems like HDFS, HBASE, and Hadoop MapReduce [77]. I have even built controllers

that control other control systems [64, 76] In all these examples we have replaced existing static configuration parameters or heuristics with control-based solutions that dynamically adjust parameters based on runtime conditions. **We repeatedly find that the control-based approach keeps the system up and meeting its goals in scenarios where even patches posted by expert developers fail.**

Combining Machine Learning and Control Theory

As noted above, learning techniques are well-suited to build models of complex, configurable computing systems and control techniques are ideal for configuring those systems to meet goals despite dynamic fluctuations. It makes intuitive sense to combine both, and that has been a core piece of my work on self-aware computing.

Initially, I felt stymied trying to bridge the gap between learned, non-linear models of discrete computing systems and the continuous, linear models used by many common control systems. I initially explored this space through several specific problems: (1) learning models of system resource efficiency and combining those with control of application alternatives to maximize application quality for a given energy budget [21], (2) learning application resource requirements and then combining that with control to meet quality-of-service guarantees for minimal cost in infrastructure-as-a-service platforms [84], (3) and learning application resource needs to control latency with minimal energy for GPUs [63].

The key lesson learned from this work was the interface needed to combine the learned model with the control system. In short, we have the learners produce piece-wise linear models representing tradeoffs (for example the tradeoff between energy and latency for a given application and system) and pass these to the controller, which uses this linear model to make efficient resource allocation decisions in response to dynamics. One additional insight was that results improved if the learner could also communicate its uncertainty to the control system. **The result is a general methodology where an abstract control system is customized at runtime by a learner.** We find that it enables: 1) fast reaction to dynamic changes, 2) error tolerance based on runtime feedback, and 3) robust design that guarantees that operat- ing requirements will be respected when possible or the ability to report when those requirements cannot be met [55].

Prior Research Projects

Multicore Architectures: Raw and Tilera As a graduate student at MIT, I had the fortunate opportunity to turn academic research (the Raw processor [22, 70–73]) into a commercial product (the Tilera TILE family of processors [29, 78]). In addition, I have several other papers on multicore and GPU support for signal processing [17, 30–35, 46, 65, 79].

Approximate Computing and Loop Perforation Approximation increases application flexibility; e.g., when resources are scarce, rather than stop computing, an approximate application produces a slightly less accurate result [15,19,27,43,54,62,67]. Controlling such approximate computations was often a motivation for the self-aware computing techniques listed above [11, 20, 21, 28, 75]. Of these, loop perforation is probably the most well-know, perhaps because it is the easiest to apply.

Distributed System Power Management Power management has been a major focus for me, often as a motivator for self-aware systems [10, 25, 52, 81, 83] and sometimes as a goal to itself [3, 4, 53, 69, 82].

Summary and Future Work

My research studies self-awareness as a fundamental property of computing systems. Self-aware systems automatically adapt behavior to meet user-specified goals in complex, dynamic computing systems. While I am primarily a systems builder, I base my work on sound mathematical models which permit some reasoning and assurance about when they will meet user goals, and—perhaps more importantly—providing understanding of when those goals are unreachable.

In the future I plan to extend self-aware computing in several ways:

- **Controlling AI Inference.** This is the direction I am most excited about. The goal and preliminary results are listed on the front page of this statement (as I thought some readers might not make it this far—if you did, I appreciate it!).
- **Security and Privacy of Self-aware Systems.** In my ideal self-aware world, a computer system would consume exactly the energy required for its current task. Unfortunately, this would also amplify side channels. I would love to make security and privacy another quantifiable goal managed alongside energy and latency. We have started preliminary work in this direction looking at the energy cost of encrypted storage [36] and how adaptive sensor systems leak information [42].
- **Self-awareness and Quantum Computing.** I have begun looking into quantum computing with my colleague, Fred Chong. So far, this has mostly consisted of adapting scheduling algorithms for parallel computing (e.g., [3, 4]) to quantum systems [1, 14, 61, 66]. However, one of my favorite results so far is the use of Bayesian optimization to improve hybrid classical-quantum systems, sometimes by as much as 50× the state-of-the-art [60].
- **Creating Modular Self-aware Systems.** Self-aware systems adapt based on feedback. In some of my early work, I realized that deploying two independent self-aware systems that act on the same feedback signal can cause destructive interference. One way to fix that is to assemble a large self-aware system from many small ones and propagate the decisions through each [13,20,21]. The drawback of this approach (and similar approaches by others) is that it requires design time knowledge and does not support dynamic composition of self-aware systems. Thus, a truly modular approach is an important part of future work—I want to independent developers to run their self-aware systems at the same time and know they will work.

References

[1] J. M. Baker, A. Litteken, C. Duckering, H. Hoffmann, H. Bernien, and F. Chong. Exploiting long-distance interactions and tolerating atom loss in neutral atom quantum architectures. In *International Symposium on Computer Architecture,* **ISCA**, 2021.

[2] S. Barati, F. A. Bartha, S. Biswas, R. Cartwright, A. Duracz, D. S. Fussell, H. Hoffmann, C. Imes, J. E. Miller, N. Mishra, Arvind, D. Nguyen, K. V. Palem, Y. Pei, K. Pingali, R. Sai, A. Wright, Y. Yang, and S. Zhang. Proteus: Language and runtime support for self-adaptive software development. **IEEE Software**, 36(2):73–82, 2019.

[3] G. Demirci, H. Hoffmann, and D. H. K. Kim. Approximation algorithms for scheduling with resource and precedence constraints. In *35th International Symposium on Theoretical Aspects of Computer Science,* ***STACS***, 2018.

[4] G. Demirci, I. Marincic, and H. Hoffmann. A divide and conquer algorithm for dag scheduling under power constraints. In *The International Conference for High Performance Computing, Networking, Storage, and Analysis,* **Supercomputing**, 2018.

[5] Z. Deng, L. Zhang, N. Mishra, H. Hoffmann, and F. Chong. Memory cocktail therapy: A general learning-based framework to optimize dynamic tradeoffs in nvms. In *50th Annual IEEE/ACM International Symposium on Microarchitecture,* **MICRO**, 2017.

[6] Y. Ding, N. Mishra, and H. Hoffmann. Generative and multi-phase learning for computer systems optimization. In *International Symposium on Computer Architecture,* **ISCA**, 2019.

[7] Y. Ding, A. Pervaiz, M. Carbin, and H. Hoffmann. Generalizable and interpretable learning for configuration extrapolation. In *29th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering,* **FSE**, 2021.

[8] Y. Ding, A. Rao, H. Song, R. Willett, and H. Hoffmann. NURD: negative-unlabeled learning for online datacenter straggler prediction. In *The Proceedings of the Conference on Machine Learning and Systems* **MLSys**, 2022.

[9] K. Du, A. Pervaiz, X. Yuan, A. Chowdhery, Q. Zhang, H. Hoffmann, and J. Jiang. Server-driven video streaming for deep learning inference. In H. Schulzrinne and V. Misra, editors, *Proceedings of the ACM Special Interest Group on Data Communication on the applications* **SIGCOMM**, 2020.

[10] D. A. Ellsworth, T. Patki, S. Perarnau, S. Seo, A. Amer, J. A. Zounmevo, R. Gupta, K. Yoshii, H. Hoffmann, A. D. Malony, M. Schulz, and P. H. Beckman. Systemwide Power Management with Argo. In *International Parallel and Distributed Processing Symposium Workshops* **IPDPS Workshops**, 2016.

[11] A. Farrell and H. Hoffmann. Meantime: Achieving both minimal energy and timeliness with approximate computing. In *USENIX Annual Technical Conference,* **USENIX ATC**, 2016.

[12] A. Filieri, H. Hoffmann, and M. Maggio. Automated Design of Self-adaptive Software with Control-theoretical Formal Guarantees. In *36th International Conference on Software Engineering,* **ICSE**, 2014.

[13] A. Filieri, H. Hoffmann, and M. Maggio. Automated multi-objective control for self-adaptive software design. In *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering,* **ESEC/FSE**, 2015.

[14] P. Gokhale, Y. Ding, T. Propson, C. Winkler, N. Leung, Y. Shi, D. I. Schuster, H. Hoffmann, and F. T. Chong. Partial compilation of variational algorithms for noisy intermediate-scale quantum machines. In *52nd Annual IEEE/ACM International Symposium on Microarchitecture,* **MICRO**, 2019.

[15] C. Hankendi, H. Hoffmann, and A. Coskun. Adapt&cap: Coordinating system and application-level adaptation for power constrained systems. **IEEE Design & Test**, to appear.

[16] M. Hao, L. Toksoz, N. Li, E. E. Halim, H. Hoffmann, and H. S. Gunawi. LinnOS: Predictability on unpredictable flash storage with a light neural network. In *14th USENIX Symposium on Operating Systems Design and Implementation,* **OSDI**, 2020.

[17] T. T. Hoang, A. Shambayati, H. Hoffmann, and A. A. Chien. Does arithmetic logic dominate data movement? a systematic comparison of energy-efficiency for FFT accelerators. In *26th IEEE International Conference on Application-specific Systems, Architectures and Processors,* **ASAP**, 2015.

[18] H. Hoffmann. Racing and pacing to idle: an evaluation of heuristics for energy-aware resource allocation. In *Proceedings of the Workshop on Power-Aware Computing and Systems,* **HotPower**, 2013.

[19] H. Hoffmann. A case for runtime coordination of accuracy-aware applications and power-aware systems (invited). In *First SIGPLAN Workshop on Probabilistic and Approximate Computing at PLDI*, 2014.

[20] H. Hoffmann. Coadapt: Predictable behavior for accuracy-aware applications running on power-aware systems. In *26th Euromicro Conference on Real-Time Systems,* **ECRTS**, 2014.

[21] H. Hoffmann. JouleGuard: Energy Guarantees for Approximate Applications. In *Proceedings of the 25th Symposium on Operating Systems Principles,* **SOSP**, 2015.

[22] H. Hoffmann, S. Devadas, and A. Agarwal. A pattern for efficient parallel computation on multicore processors with scalar operand networks. In *Proceedings of the 2010 Workshop on Parallel Programming Patterns* **ParaPLoP**, 2010.

[23] H. Hoffmann, J. Holt, G. Kurian, E. Lau, M. Maggio, J. E. Miller, S. M. Neuman, M. E. Sinangil, Y. Sinangil, A. Agarwal, A. P. Chandrakasan, and S. Devadas. Self-aware Computing in the Angstrom Processor. In *The 49th Annual Design Automation Conference 2012,* **DAC**, 2012.

[24] H. Hoffmann, A. Jantsch, and N. D. Dutt. Embodied self-aware computing systems. **Proc. IEEE**, 108(7):1027–1046, 2020.

[25] H. Hoffmann and M. Maggio. PCP: A generalized approach to optimizing performance under power constraints through resource management. In *11th International Conference on Autonomic Computing,* **ICAC**, 2014.

[26] H. Hoffmann, M. Maggio, M. D. Santambrogio, A. Leva, and A. Agarwal. A generalized software framework for accurate and efficient management of performance goals. In *Proceedings of the International Conference on Embedded Software,* **EMSOFT**, 2013.

[27] H. Hoffmann, S. Misailovic, S. Sidiroglou, A. Agarwal, and M. Rinard. Using Code Perforation to Improve Performance, Reduce Energy Consumption, and Respond to Failures . Technical Report MIT-CSAIL-TR-2009-042, MIT, September 2009.

[28] H. Hoffmann, S. Sidiroglou, M. Carbin, S. Misailovic, A. Agarwal, and M. C. Rinard. Dynamic Knobs for Responsive Power-aware Computing. In *Proceedings of the 16th International Conference on Architectural Support for Programming Languages and Operating Systems,* **ASPLOS**, 2011.

[29] H. Hoffmann, D. Wentzlaff, and A. Agarwal. Remote store programming. In *High Performance Embedded Architectures and Compilers, 5th International Conference,* **HiPEAC**, 2010.

[30] W. Hussain, R. Airoldi, H. Hoffmann, T. Ahonen, and J. Nurmi. Design of an accelerator-rich architecture by integrating multiple heterogeneous coarse grain reconfigurable arrays over a network-on-chip. In *IEEE 25th International Conference on Application-Specific Systems, Architectures and Processors,* **ASAP**, 2014.

[31] W. Hussain, R. Airoldi, H. Hoffmann, T. Ahonen, and J. Nurmi. HARP2: an x-scale reconfigurable accelerator-rich platform for massively-parallel signal processing algorithms. **Signal Processing Systems**, 85(3), 2016.

[32] W. Hussain, H. Hoffmann, T. Ahonen, and J. Nurmi. Constraint-driven frequency scaling in a coarse grain reconfigurable array. In *2014 International Symposium on System-on-Chip,* **ISSoC** *2014*, 2014.

[33] W. Hussain, H. Hoffmann, T. Ahonen, and J. Nurmi. Power mitigation by performance equalization in a heterogeneous reconfigurable multicore architecture. **Journal of Signal Processing Systems**, pages 1–11, 2016.

[34] W. Hussain, H. Hoffmann, T. Ahonen, and J. Nurmi. *Design Transformation from a Single-Core to a Multi-Core Architecture Targeting Massively Parallel Signal Processing Algorithms*, pages 7–28. Springer International Publishing, 2017.

[35] W. Hussain, H. Hoffmann, T. Ahonen, and J. Nurmi. Power mitigation by performance equalization in a heterogeneous reconfigurable multicore architecture. **Signal Processing Systems**, 87(3), 2017.

[36] B. D. III, A. J. Feldman, H. Gunawi, and H. Hoffmann. StrongBox: Confidentiality, Integrity, and Performance using Stream Ciphers for Full Drive Encryption. In *Proceedings of the Twenty-third International Conference on Architectural Support for Programming Languages and Operating Systems,* **ASPLOS**, 2018.

[37] C. Imes and H. Hoffmann. Bard: A unified framework for managing soft timing and power constraints. In *Submission*, 2015.

[38] C. Imes, S. A. Hofmeyr, and H. Hoffmann. Energy-efficient application resource scheduling using machine learning classifiers. In *Proceedings of the International Conference on Parallel Processing,* **ICPP**, 2018.

[39] C. Imes, D. H. K. Kim, M. Maggio, and H. Hoffmann. POET: A Portable Approach to Minimizing Energy Under Soft Real-time Constraints. In *21st IEEE Real-Time and Embedded Technology and Applications Symposium,* **RTAS**, 2015.

[40] C. Imes, H. Zhang, K. Zhao, and H. Hoffmann. Copper: Soft real-time application performance using hardware power capping. In *16th International Conference on Autonomic Computing,* **ICAC**, 2019.

[41] T. Kannan and H. Hoffmann. Budget RNNs: Multi-Capacity Neural Networks to Improve In-Sensor Inference under Energy Budgets. In *27th IEEE Real-Time and Embedded Technology and Applications Symposium,* **RTAS**, 2021.

[42] T. Kannan and H. Hoffmann. Protecting adaptive sampling from information leakage on low-power sensors. In *27th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Lausanne,* **ASPLOS**, 2022.

[43] O. Khan, H. Hoffmann, M. Lis, F. Hijaz, A. Agarwal, and S. Devadas. ARCc: A case for an architecturally redundant cache-coherence architecture for large multicores. In *Computer Design, International Conference on* **ICCD**, 2011.

[44] D. H. K. Kim, C. Imes, and H. Hoffmann. Racing and pacing to idle: Theoretical and empirical analysis of energy optimization heuristics. In *2015 IEEE 3rd International Conference on Cyber-Physical Systems, Networks, and Applications,* **CPSNA**, 2015.

[45] C. Li, S. Wang, H. Hoffmann, and S. Lu. Statically inferring performance properties of software configurations. In *Fifteenth EuroSys Conference,* **Eurosys**, 2020.

[46] G. Li, X. Chen, G. Sun, H. Hoffmann, Y. Liu, Y. Wang, and H. Yang. A stt-ram-based low-power hybrid register file for gpgpus. In *Proceedings of the 52nd Annual Design Automation Conference* **DAC**, 2015.

[47] M. Maggio, H. Hoffmann, A. V. Papadopoulos, J. Panerati, M. D. Santambrogio, A. Agarwal, and A. Leva. Comparison of decision-making strategies for self-optimization in autonomic computing systems. **ACM Trans. Auton. Adapt. Syst.**, 7(4), Dec. 2012.

[48] M. Maggio, H. Hoffmann, M. D. Santambrogio, A. Agarwal, and A. Leva. Controlling software applications via resource allocation within the heartbeats framework. In *Proceedings of the 49th IEEE Conference on Decision and Control,* **CDC**, 2010.

[49] M. Maggio, H. Hoffmann, M. D. Santambrogio, A. Agarwal, and A. Leva. Power optimization in embedded systems via feedback control of resource allocation. **IEEE Trans. Contr. Sys. Techn.**, 21(1), 2013.

[50] M. Maggio, A. V. Papadopoulos, A. Filieri, and H. Hoffmann. Automated control of multiple software goals using multiple actuators. In *Symposium on the Foundations of Software Engineering* **FSE**, 2017.

[51] M. Maggio, A. V. Papadopoulos, A. Filieri, and H. Hoffmann. Self-adaptive video encoder: Comparison of multiple adaptation strategies made simple. In *12th IEEE/ACM International Symposium on Software Engineering for Adaptive and Self-Managing Systems,* **SEAMS**, 2017.

[52] I. Marincic and H. H. Venkat Vishwanath. Providing Fairness in Heterogeneous Multicores with a Predictive, Adaptive Scheduler. In *34th IEEE International Parallel and Distributed Processing Symposium,* **IPDPS**, 2020.

[53] I. Marincic, V. Vishwanath, and H. Hoffmann. Polimer: An energy monitoring and power limiting interface for HPC applications. In *Proceedings of the 5th International Workshop on Energy Efficient Supercomputing,* **E2SC@SC**, 2017.

[54] S. Misailovic, S. Sidiroglou, H. Hoffmann, and M. C. Rinard. Quality of service profiling. In *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering,* **ICSE**, 2010.

[55] N. Mishra, C. Imes, J. D. Lafferty, and H. Hoffmann. CALOREE: Learning Control for Predictable Latency and Low Energy. In *Proceedings of the Twenty-third International Conference on Architectural Support for Programming Languages and Operating Systems,* **ASPLOS**, 2018.

[56] N. Mishra, J. D. Lafferty, and H. Hoffmann. ESP: A machine learning approach to predicting application interference. In *14th International Conference on Autonomic Computing,* **ICAC**, 2017.

[57] N. Mishra, H. Zhang, J. D. Lafferty, and H. Hoffmann. A Probabilistic Graphical Model-based Approach for Minimizing Energy Under Performance Constraints. In *Proceedings of the Twentieth International Conference on Architectural Support for Programming Languages and Operating Systems,* **ASPLOS**, 2015.

[58] M. Möstl, J. Schlatow, R. Ernst, H. Hoffmann, A. Merchant, and A. Shraer. Self-aware systems for the internet-of-things. In *Proceedings of the Eleventh IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis,* **CODES**, 2016.

[59] A. Pervaiz, Y. H. Yang, A. Duracz, F. Bartha, R. Sai, C. Imes, R. Cartwright, K. Palem, S. Lu, and H. Hoffmann. GOAL: Supporting general and dynamic adaptation in computing systems. In *Proceedings of the ACM SIGPLAN International Symposium on New Ideas, New Paradigms, and Reflections on Programming and Software* **Onward!**, 2022.

[60] G. S. Ravi, P. Gokhale, Y. Ding, W. M. Kirby, K. N. Smith, J. M. Baker, P. J. Love, H. Hoffmann, K. R. Brown, and F. T. Chong. CAFQA: clifford ansatz for quantum accuracy. In *Proceedings of the 28th ACM International Conference on Architectural Support for Programming Languages and Operating Systems* **ASPLOS**, 2023.

[61] G. S. Ravi, K. N. Smith, J. M. Baker, T. Kannan, N. Earnest, A. Javadi-Abhari, H. Hoffmann, and F. T. Chong. Navigating the dynamic noise landscape of variational quantum algorithms with QISMET. In *Proceedings of the 28th ACM International Conference on Architectural Support for Programming Languages and Operating Systems* **ASPLOS**, 2023.

[62] M. C. Rinard, H. Hoffmann, S. Misailovic, and S. Sidiroglou. Patterns and statistical analysis for understanding reduced resource computing. In *Proceedings of the 25th Annual ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages, and Applications,* **OOPSLA**, 2010.

[63] M. H. Santriaji and H. Hoffmann. GRAPE: Minimizing Energy for Interactive GPU Applications. In *49th Annual IEEE/ACM International Symposium on Microarchitecture* **MICRO**, 2016.

[64] M. H. Santriaji and H. Hoffmann. Formalin: Architectural support for power & performance aware GPU. In *IEEE Conference on Control Technology and Applications,* **CCTA**, 2018.

[65] M. H. Santriaji and H. Hoffmann. MERLOT: Architectural Support for Energy-Efficient Real-time Processing in GPUs. In *24th IEEE Real-Time and Embedded Technology and Applications Symposium,* **RTAS**, 2018.

[66] Y. Shi, N. Leung, P. Gokhale, Z. Rossi, D. I. Schuster, H. Hoffmann, and F. T. Chong. Optimized compilation of aggregated instructions for realistic quantum computers. In *Proceedings of the Twenty-fourth International Conference on Architectural Support for Programming Languages and Operating Systems,* **ASPLOS**, 2019.

[67] S. Sidiroglou-Douskos, S. Misailovic, H. Hoffmann, and M. C. Rinard. Managing performance vs. accuracy trade-offs with loop perforation. In *SIGSOFT/FSE'11 19th ACM SIGSOFT Symposium on the Foundations of Software Engineering* **(FSE-19)** *and ESEC'11: 13rd European Software Engineering Conference (***ESEC-13***), Szeged, Hungary, September 5-9, 2011*, 2011.

[68] Y. Sinangil, S. M. Neuman, M. E. Sinangil, N. Ickes, G. Bezerra, E. Lau, J. E. Miller, H. C. Hoffmann, S. Devadas, and A. P. Chandraksan. A self-aware processor soc using energy monitors integrated into power converters for self-adaptation. In **VLSI** *Circuits Digest of Technical Papers, 2014 Symposium on*. IEEE, 2014.

[69] T. Srivastava, H. Zhang, and H. Hoffmann. Penelope: Peer-to-peer power management. In *the 51st International Conference on Parallel Processing* **ICPP**, 2022.

[70] V. Strumpen, H. Hoffmann, and A. Agarwal. Stream algorithms and architecture. **J. Instruction-Level Parallelism**, 6, 2004.

[71] M. B. Taylor, J. Kim, J. Miller, D. Wentzlaff, F. Ghodrat, B. Greenwald, H. Hoffmann, P. Johnson, W. Lee, A. Saraf, N. Shnidman, V. Strumpen, S. Amarasinghe, and A. Agarwal. A 16-issue Multiple-Program-Counter Microprocessor with Point-to-Point Scalar Operand Network. In *Proceedings of the IEEE International Solid-State Circuits Conference* **ISSCC**, February 2003.

[72] M. B. Taylor, J. S. Kim, J. E. Miller, D. Wentzlaff, F. Ghodrat, B. Greenwald, H. Hoffmann, P. Johnson, J. W. Lee, W. Lee, A. Ma, A. Saraf, M. Seneski, N. Shnidman, V. Strumpen, M. Frank, S. P. Amarasinghe, and A. Agarwal. The raw microprocessor: A computational fabric for software circuits and general-purpose programs. **IEEE Micro**, 22(2), 2002.

[73] M. B. Taylor, W. Lee, J. E. Miller, D. Wentzlaff, I. Bratt, B. Greenwald, H. Hoffmann, P. Johnson, J. S. Kim, J. Psota, A. Saraf, N. Shnidman, V. Strumpen, M. Frank, S. P. Amarasinghe, and A. Agarwal. Evaluation of the raw microprocessor: An exposed-wire-delay architecture for ILP and streams. In *31st International Symposium on Computer Architecture* **ISCA**, 2004.

[74] C. Wan, H. Hoffmann, S. Lu, and M. Maire. Orthogonalized SGD and nested architectures for anytime neural networks. In *Proceedings of the 37th International Conference on Machine Learning,* **ICML**, 2020.

[75] C. Wan, M. H. Santriaji, E. Rogers, H. Hoffmann, M. Maire, and S. Lu. ALERT: accurate learning for energy and timeliness. In A. Gavrilovska and E. Zadok, editors, *USENIX Annual Technical Conference,* **USENIX ATC**, 2020.

[76] S. Wang, H. Hoffmann, and S. Lu. Agilectrl: A self-adaptive framework for configuration tuning. In *The 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering* **ESEC/FSE**, 2022.

[77] S. Wang, C. Li, W. Sentosa, H. Hoffmann, and S. Lu. Understanding and Auto-Adjusting Performance-Sensitive Configurations. In *Proceedings of the Twenty-third International Conference on Architectural Support for Programming Languages and Operating Systems,* **ASPLOS**, 2018.

[78] D. Wentzlaff, P. Griffin, H. Hoffmann, L. Bao, B. Edwards, C. Ramey, M. Mattina, C. Miao, J. F. B. III, and A. Agarwal. On-chip interconnection architecture of the tile processor. **IEEE Micro**, 27(5), 2007.

[79] A. W. B. Yudha, R. Pulungan, H. Hoffmann, and Y. Solihin. A simple cache coherence scheme for integrated CPU-GPU systems. In *57th ACM/IEEE Design Automation Conference,* **DAC**, 2020.

[80] Y. Zamora, L. Ward, G. Sivaraman, I. Foster, and H. Hoffmann. Proxima: Accelerating the Integration of Machine Learning in Atomistic Simulations. In *International Conference on Supercomputing* **ICS**, 2021.

[81] H. Zhang and H. Hoffmann. Maximizing Performance Under a Power Cap: A Comparison of Hardware, Software, and Hybrid Techniques. In *Proceedings of the Twenty-first International Conference on Architectural Support for Programming Languages and Operating Systems,* **ASPLOS**, 2016.

[82] H. Zhang and H. Hoffmann. Performance & energy tradeoffs for dependent distributed applications under system-wide power caps. In *Proceedings of the International Conference on Parallel Processing,* **ICPP**, 2018.

[83] H. Zhang and H. Hoffmann. PoDD: Power-capping dependent distributed applications. In *The International Conference for High Performance Computing, Networking, Storage, and Analysis,* **Supercomputing**, 2019.

[84] Y. Zhou, H. Hoffmann, and D. Wentzlaff. CASH: Supporting IaaS Customers with a Sub-core Configurable Architecture. In *Proceedings of the Twenty-first International Symposium on Computer Architecture,* **ISCA**, 2016.