Right-Provisioned IoT Edge Computing: An Overview

Tosiron Adegbija
tosiron@email.arizona.edu
Department of Electrical & Computer Engineering
University of Arizona, Tucson, AZ, USA

Roman Lysecky
rlysecky@ece.arizona.edu
Department of Electrical & Computer Engineering
University of Arizona, Tucson, AZ, USA

Vinu Vijay Kumar
vinuv@google.com
Google
Mountain View, CA

ABSTRACT

Edge computing on the Internet of Things (IoT) is an increasingly popular paradigm in which computation is moved closer to the data source (i.e., edge devices). Edge computing mitigates the overheads of cloud-based computing arising from increased response time, communication bandwidth, data security and privacy, energy consumption, etc. However, given the potentially stringent resource constraints and functional requirements of emerging IoT devices, edge computing must neither be over- or under-provisioned for its stated purpose. In this paper, we present an overview of the problem of *right-provisioned IoT edge computing*, wherein IoT devices are equipped with resources that are 'just enough,' even when 'just enough' may not be clearly defined at design time. We highlight a few research directions and key challenges that must be addressed to enable right-provisioned IoT edge computing.

CCS CONCEPTS

• **Computer systems organization → Embedded systems**; **Processors and memory architectures**;

KEYWORDS

Internet of Things; edge computing; low-power embedded systems; right-provisioned; adaptable computing; high-level synthesis; memory hierarchy; computation migration; IoT security.

ACM Reference format:
Tosiron Adegbija, Roman Lysecky, and Vinu Vijay Kumar. 2019. Right-Provisioned IoT Edge Computing: An Overview. In *Proceedings of Great Lakes Symposium on VLSI 2019, Tysons Corner, VA, USA, May 9–11, 2019 (GLSVLSI '19)*, 6 pages.
https://doi.org/10.1145/3299874.3319338

1 INTRODUCTION

Unlike on the Internet, where most of the data is generated by humans, on the Internet of Things (IoT), most of the data will be generated by 'things' or edge devices. Considering that the IoT continues to grow rapidly, with billions of devices that span a wide variety of applications and domains, the challenge of efficiently processing the vast amounts of data produced by IoT devices will become even more pressing [1]. *Edge computing* [2] has emerged as a paradigm in which computation is moved closer to the edge devices of the IoT network, in order to reduce the overheads concomitant to the bandwidth and latency challenges of transmitting data to the cloud for computations. In this paper, we distinguish edge computing as computation on the edge devices [3], as opposed to fog computing or mobile edge computing (MEC), where computation is performed at base stations or routers [2, 4].

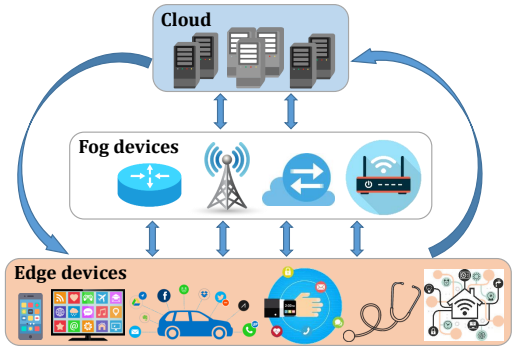

Figure 1: Overview of the Internet of Things (IoT) devices and where computations can be performed, ranging from edge devices to cloud devices.

As illustrated in Figure 1, computations can be performed on the fog devices, or even closer to the data source on the edge devices. However, moving computations closer to the edge devices is especially challenging because edge devices usually have stringent resource constraints (e.g., physical size, energy) and functional requirements (e.g., response time, application execution) that necessitate that the computational resources are *right-provisioned* for the individual device, system, or application.

Right-provisioned IoT edge computing implies that the edge device is neither under-provisioned nor over-provisioned for the requisite functions. When under-provisioned, the device is unable to satisfy its functional requirements; for instance, response time is high, or worse, a hard deadline may be violated [5]. On the other hand, when over-provisioned, the device wastes resources and incurs unnecessary overheads (e.g., cost, size, energy consumption). As such, the provisioning of computational resources must be 'just enough' for all its functional requirements, including computations, communications, security, etc.

To achieve right-provisioned IoT edge computing, several challenges must be addressed. First, considering that right-provisioning could be very nuanced for different systems and applications—a right-provisioned system for one application may be severely under- or over-provisioned for another—right-provisioning must first be

GLSVLSI '19, May 9–11, 2019, Tysons Corner, VA, USA

ACM ISBN 978-1-4503-6252-8/19/05.
https://doi.org/10.1145/3299874.3319338

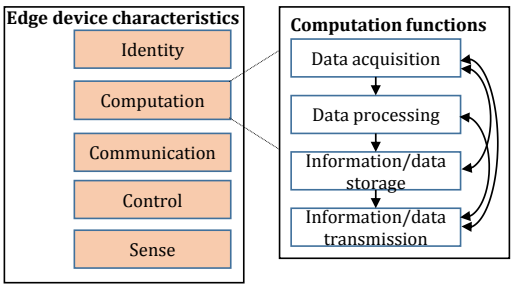

Figure 2: High-level characteristics and computational requirements of an edge device.

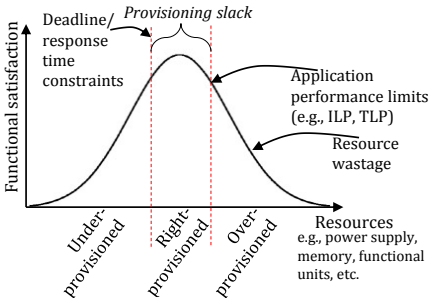

Figure 3: Defining right-provisioning.

clearly defined for the target system. This could raise further challenges, depending on how much a priori information is available about the system. Given a known system, where the system's behavior is completely known at design time, right-provisioning can be relatively easy defined for the system. However, most IoT devices involve some variability, where applications, system inputs, objective functions, or user requirements may change drastically during runtime. In such systems, the provisioning must be adaptable to dynamically changing runtime requirements, without introducing substantial overheads from the adaptability.

This paper provides an overview of the problem of right-provisioned IoT edge computing. We explore what it means for an IoT device to be right-provisioned with respect to its computational capabilities and explore some of the nuances of right-provisioning. We also highlight some recent work and a few research challenges that need to be addressed to enable right-provisioned computing in emerging IoT edge devices.

2 DEFINING RIGHT-PROVISIONED IOT EDGE COMPUTING

Figure 2 presents a high-level summary of the general characteristics of edge devices and their key computational functions. These characteristics and functions may also apply to fog devices, but for brevity, we simply refer to edge devices in the rest of this paper. The major characteristics, which have been discussed in detail in prior work, include *unique identity* (e.g., IPv6 addressing) [6], *communication* (e.g., Bluetooth, WiFi) [7], *control* (e.g., actuation) [8], *sense* (via various sensors) [9], and *computation* (e.g., microcontrollers, microprocessors) [10]. This paper focuses on right-provisioning for the computation functions, which can further be broken down into *data acquisition*, *data processing*, *information or data storage*, and *information or data transmission*. Details of these functions have been presented in prior work (e.g., [3, 10]).

The first step to designing right-provisioned edge devices is to understand what it means for the device to be right-provisioned for its target application, given the required computation functions. At this point, we draw a distinction between *optimization* and right-provisioning. Optimization implies that the system is designed to achieve the *best possible* results given the system constraints, e.g., minimize the energy consumption without degrading the performance beyond a given threshold. Right-provisioning, on the other hand, means that the best results need not be achieved as long as the device's functional requirements are fully satisfied.

Understanding the device requirements is very challenging due to the wide variety of devices and applications that exist on the IoT. In general, right-provisioning may also involve allowing for some *provisioning slack*. That is, the system does not necessarily have to be *perfectly* provisioned, as long as its functional requirements are met. Figure 3 illustrates this idea. The system resources (e.g., power supply, memory, functional units) are under-provisioned when they are unable to satisfy the applications' functional requirements, such as a deadline or response time constraint.

For instance, consider a task that must complete in time ts. Assuming a soft deadline requirement [11], which allows the task to be completed within a given slack ls beyond the deadline, without any system failure, the system is considered right-provisioned as long as its computational resources allow the task to be completed within $(t + l)$s. The system is considered over-provisioned when additional resources, beyond what is required to achieve $(t + l)$s, do not result in any functional benefits to the system. For instance, a system may be over-provisioned with out-of-order execution resources if the application has limited instruction level parallelism (ILP) and the application's instructions must execute in order [12]. In some cases, the slack produced by the over-provisioned system may be leveraged for other applications, whereas in other cases, the over-provisioning is simply tantamount to resource wastage.

Therefore, an edge device is right-provisioned if the available resources R are such that $r \leq R \leq (r + s)$, where r is the resource requirement to achieve the system's minimum functional requirements, and s is the provisioning slack and may be bounded by the system's design constraints. Both r and s can be generalized to any functional requirements, such as energy, response time, etc. It is clear that right-provisioning is difficult to define, given the variety of applications, systems, and requirements. The next two subsections attempt to further define some of the nuances of right-provisioned edge computing.

2.1 Granularity of Right-Provisioning

One of the first questions that must be answered when designing a right-provisioned edge device is the execution granularity for which the device is provisioned. We define four alternatives for execution granularity, namely, *task-specific*, *application-specific*, *domain-specific*, and *general purpose*. Task-specific (or phase-based) right-provisioning refers to a case where an edge device is right-provisioned for a specific task, kernel, or application phase. For

example, inference tasks, such as audio sensing, which are commonly executed by edge devices (e.g., wearables) represent a good candidate for task-specific right-provisioning [13].

Application-specific right-provisioning refers to satisfying the requirements of a specific application. For example, prior work (e.g. [3]) has proposed edge devices specifically for ECG applications in the Internet of Medical Things (IoMT). Domain-specific right-provisioning refers to satisfying the requirements of a domain of applications, which would likely have a similar set of characteristics or functional requirements. For example, Yamaga et al. [14] propose an approximate-ReRAM that uses deep neural network (DNN) for image recognition applications. While domain-specific right-provisioning may not satisfy the functional requirements of individual applications, unlike application-specific right-provisioning, the domain-specific approach allows for better application flexibility and may be better overall for a system that executes a variety of similar applications. Finally, general-purpose right-provisioning involves equipping the edge device with resources to satisfy a wide variety of tasks and applications, without necessary being right-provisioned for a single task, application, or domain [13]. Depending on the target edge device, the various granularities of right-provisioning can be achieved in various ways as we discuss in the following subsection.

2.2 Static vs. Dynamic Right-Provisioning

Edge devices can be provisioned either *statically* or *dynamically*. A statically right-provisioned edge device is designed with the appropriate computing resources, which remain static throughout the device's lifetime. There is a wide variety of systems that enable statically right-provisioned edge devices, ranging from application-specific integrated circuits (ASICs) [15] to CPUs that are specifically tuned for the executing applications or a group of applications [16]. Most current edge devices are statically provisioned. For instance, the FitBit features a low-power ARM Cortex M3 microcontroller, and the NEST thermostat features the higher-performance ARM Cortex A8 processor for its machine learning functions. In both cases, the computing resources remain static throughout the devices' lifetime, regardless of what task is being executed. In order for an edge device to be statically right-provisioned, the system must be fully known at design time. That is, the applications must be known and must not be subject to substantial runtime changes (e.g., changes in execution requirements due to new inputs). As such, static right-provisioning is easiest for devices that only execute a single application or a small group of applications whose characteristics are known a priori.

A dynamically right-provisioned edge device, on the other hand, allows the system resources to be dynamically adapted to the runtime functional requirements of executing tasks. Dynamic provisioning is especially important since most applications exhibit variability in runtime behavior due to the applications' characteristics, new inputs, user requirements, etc. For instance, dynamic voltage and frequency scaling (DVFS) is a popular technique that allows the clock frequency (or voltage) to be dynamically adapted to different runtime requirements, in order to reduce energy consumption or meet task deadlines [17].

Dynamic provisioning has the advantage that, unlike static provisioning, various runtime requirements can be satisfied by the edge device. For example, for an edge device that executes multiple applications with different levels of thread-level parallelism, the device can be provisioned with a multicore CPU such that cores can be powered off, depending on the number of parallel threads present in an application. Furthermore, parts of various components (e.g., cache resources, functional units, etc.) can be shut down, depending on the applications' execution requirements. Dynamic right-provisioning can also be achieved using heterogeneous resources, wherein tasks can be dynamically scheduled to the resource that most closely satisfies each task's requirement [16]. While there is a large body of work on dynamic provisioning for a wide variety of general-purpose applications, there still remains much research to be done on novel low-overhead techniques to enable adaptability for IoT edge devices, given the stringent design constraints of these devices.

3 OPEN RESEARCH CHALLENGES

In this section, we highlight a few important open research challenges for right-provisioned IoT edge computing. We note that these challenges are not exhaustive; they represent a variety of research directions in which there is much ongoing research, but still contain several gaps that need further investigations and improvement to fully enable right-provisioned IoT edge computing.

3.1 High-Level Synthesis for Right-Provisioning

One of the driving forces for right-provisioned IoT edge computing will be application-specific systems that adhere to specific timing, resource, and functional requirements. While several system-level hardware design methods exist, current techniques typically trade off productivity for timing precision. Hardware description language (HDLs), such as Verilog and VHDL, require designers to specify system behaviors for each clock cycle (i.e., cycle-accurate description). While these methods allow precise timing specification, design productivity is low.

To increase designer productivity, high-level synthesis (HLS) approaches use high-level languages (e.g., C, C++) to specify system functionality and automatically generate hardware implementations [18]. However, high-level languages rely on the implied order of operations, as opposed to explicitly defined and precise timing. Thus, timing must be inferred from the structure or manually added through tool-specific annotations, thereby leading to computations that may behave differently than the equivalent software [19].

Given these drawbacks of current HLS techniques, new approaches are required to enable the rapid design and optimization of hardware components for IoT edge devices that satisfy their functional requirements and adhere to their timing and resource constraints. We refer to these required new HLS approaches as *timing-driven high-level synthesis (TD-HLS)*. Figure 4 provides an overview of a TD-HLS methodology for designing timing-driven low-power right-provisioned edge devices. The initial goal is to select clock domain and frequencies for each of the system's individual components, such that the combined latencies are less than the overall latency constraint, while satisfying power constraints. During system specification, designers can specify end-to-end latency constraints for a system-level composition of components.

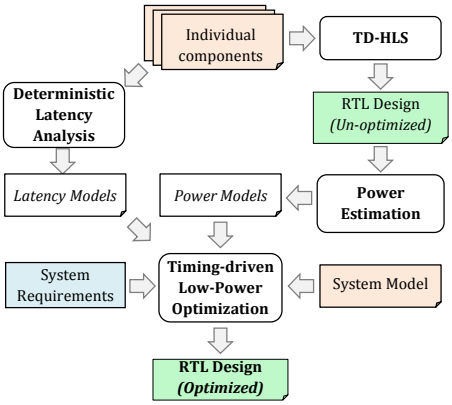

Figure 4: Overview of the Timing-Driven High Level Synthesis (TD-HLS) approach.

Given the synthesized RTL design, commercial tools for FPGAs or ASICs can be used to create power models for each individual component that relate power consumption to the execution frequency. The latency and power model can then be utilized to determine a right-provisioned frequency mapping that meets the system's latency and/or energy constraints.

There are several open problems that must be addressed towards achieving the required TD-HLS techniques. For instance, emerging HLS approaches must provide formal design, synthesis, and optimization frameworks that support synthesizable precisely-timed system models that enable the explicit and integrated specification of precise timing behaviors for various kinds of computation.

Furthermore, these new HLS approaches must be capable of automatically extracting component-level and system-level synthesis constraints required to seamlessly integrate with existing performance-driven HLS tools/approaches. Finally, the new HLS approaches must support design methods, event-driven communication, and self-clock gating for optimizing energy consumption, given system requirements and constraints.

3.2 Computation Migration

One of the most important decisions that must be made in edge computing is *how much* computing *should* and *can* be performed at the edge. While the conventional wisdom is that migrating computations from edge devices to the fog or cloud can improve the IoT system's overall performance and energy consumption [20], a right-provisioned edge device may further alleviate the performance and energy overheads of transferring data between the edge device and other layers (fog or cloud).

From the perspective of the edge device, there are two extremes of computation migration with various tradeoffs. On one extreme, a *zero computation migration* or *local execution* scheme could be used, where all the computations are performed on the edge device. This scheme lends itself to the lowest response time, since it eliminates the latency overheads of transmitting data to the fog or cloud. However, performing all the computations on the edge device may result in prohibitive overhead with respect to energy and form factor. On the other extreme, a *full computation migration* or *full offloading* [21] scheme could be used, where the edge device collects data and transmits all the data to the cloud where all the computations are performed. A happy medium between both extremes is the *partial computation migration* or *partial offloading* [22] scheme where some computations are performed on the edge device, while the rest is transmitted to the fog (or cloud).

There is a growing body of work that studies different computation migration schemes for different applications [4, 23]. For example, Xu et al [24] studied the response time effects of partitioning different components of a deep neural network (DNN) between an edge device and a mobile gateway (fog). However, there is still much room for research into understanding whether or not there is an optimal choice for computation migration, given the system constraints and functional requirements. Understanding the best computation migration scheme to achieve right-provisioning for the target system is a challenging communication-computation co-optimization problem, since the best scheme depends on both the computation resources on the edge device and the communication medium. For instance, while computing on fog devices may be faster or more energy efficient than computing on the edge device, the transmission time to move data between the edge device and the fog may negate the speed or energy benefits of computing on fog devices.

Furthermore, a formulation of the best scheme also depends on how well-known the system is at design time. If the applications, computation resources, and communication characteristics are well-known at design time, the best computation migration scheme may be accurately defined a priori. However, in most realistic systems, some of the required information may be unknown at design time or the systems may change during runtime due to application interference, environmental uncertainty, mobility, or even dynamic functional requirements and constraints. As such, *dynamic computation migration* schemes may be required to adaptively determine where computations must be performed in order to satisfy the functional requirements and system constraints.

3.3 Threat Detection, Mitigation, and Resilience

IoT edge devices are especially vulnerable to security threats due to several intrinsic characteristics, such as mobility, connectivity, scale, resource constraints, complexity, etc. Edge devices are susceptible to several kinds of threats including side-channel attacks (e.g., differential power analysis (DPA) [25]) and hardware-based vulnerabilities (e.g., IC piracy, IC counterfeiting, reverse engineering [26]). As alluded to throughout this paper, IoT edge devices also typically have stringent functional requirements and limited computational resources. As such, resources for achieving security and privacy are usually limited. Furthermore, achieving security must not detract from the computational resources required for satisfying the devices' functional requirements. Thus, security mechanisms for edge computing must also be right-provisioned, such that they do not waste resources, but are also sufficient for the security requirements of individual devices or applications. It is important that emerging security mechanisms are designed to be both proactive (i.e., able to prevent intrusions) and reactive (i.e., able to detect and mitigate the effect of intrusions). In addition, the security functions must be performed while ensuring continuity of operations.

Several efforts have been made to develop novel techniques for detecting and mitigating threats in IoT devices. For example, Kumar

et al. [27] present the design of an energy-efficient secure positive feedback adiabatic logic (EE-SPEAL) to enable the design of DPA-resistant IoT circuits, while saving energy compared to traditional CMOS circuits. Carreon et al. [28] present a probabilistic approach for estimating the presence of malware for individual operations and sequences of operations in an embedded system, including thresholds to minimize false positives based on training data.

There is still much room for research on efficient security mechanisms that can non-invasively detect threats, observe anomalous execution behavior, isolate the system components/data that are affected by the detected threat, and allow the device to operate in a mode that isolates threats (e.g., by disabling affected components). These detection and mitigation methods must be appropriately provisioned for the specific IoT edge device based on the available computing resources and constraints. In addition, similar to computation migration, security functions may need to be distributed across different levels of the IoT system. For instance, given an edge device with limited resources, the runtime threat detection can be customized to use a simple model for defining the normal system execution behaviors, which would allow energy-efficient threat detection but with a lower accuracy. In contrast, a gateway (fog) device with higher computational capabilities and energy resources can be customized to use a more robust model and machine learning algorithms for threat detection, using aggregated data across multiple edge devices.

3.4 Right-Provisioned Energy Supply

Energy efficiency is probably one of the most researched topics with respect to edge devices and remains a daunting challenge as a result of several characteristics, including, limited battery resources, small form factor, scale, mobility, real-time requirements, etc. Several emerging technologies offer much promise for right-provisioned energy supply in edge devices. One such technology is *energy harvesting* [17], which converts power from ambient sources, such as mechanical motion, electromagnetic radiation, etc., into electrical power. For instance, in a wearable device (e.g., smartwatch), the mechanical energy produced by the motion of a human body can be converted into electric energy [29]. As such, energy harvesting capability serves as a renewable energy source, thereby mitigating the need for reliance on batteries for energy. Important directions for right-provisioned energy supply for edge devices include novel approaches for self-powered devices, low-overhead energy storage mechanisms, and techniques for coping with the unpredictability of energy harvesting mechanisms. A framework for maximizing energy efficiency under ultra-low power harvesting scenarios is presented in [30]. Furthermore, new advances must take into account the variability of power and longevity requirements for the different devices and applications, and the fact that these requirements may vary during runtime [31].

Another technology trend that warrants further investigation with respect to edge devices is dark silicon [32]. Dark silicon is an undesirable feature of thermal and current limited modern designs. It can also be an unavoidable byproduct of a platform-based design approach to amortizing design costs over a family of products. A good strategy to take advantage of the inevitability of dark silicon is to divide the design into highly power-efficient always-powered infrastructure regions, and energy efficient but not necessarily power optimal dark regions. The dark silicon regions are enabled on demand non-concurrently based on use cases. The dark silicon regions would be designed to a 'Joule/compute-operation' metric and include right-provisioned complex processing and techniques such as lossy compression, etc. that are power inefficient, but minimize overall energy due to reduced demand on the always powered infrastructure. A model-free reinforcement learning (RL) approach for optimizing power efficiency and lowering energy delay product is presented in [33].

3.5 Adaptable Memories

As can be inferred from Figure 2, one of the most important components of an IoT device is the memory subsystem. At different stages of an edge device's lifecycle, different kinds of data (e.g., raw data, intermediate data, processed information) must be stored or cached for quick access. As such, the memory subsystem has significant implications for several system functions and characteristics, including performance, energy consumption, form factor, etc.

Memory design and optimization techniques for a wide variety of systems and objectives (e.g., energy reduction) is a well-studied research area. However, IoT edge devices have some intrinsic characteristics that can be explored and exploited for further improving the memory subsystem design. For example, some IoT applications are error-tolerant, meaning that a right-provisioned memory subsystem for edge devices executing such applications need not be error-free [3]. However, the applications' error tolerance must be carefully analyzed in order to design memory subsystems that satisfy the requisite functional requirements. Alternatively, hybrid memory systems—featuring both error-free and error-prone memory segments—can be utilized for edge devices that execute applications with different runtime requirements [34].

An emerging class of memories that offer much promise for designing right-provisioned edge devices are non-volatile memories (NVMs) [35]. NVMs (e.g., STT-RAM, flash, PCM, etc.) offer several advantages, including non-volatility (beneficial for designing non-volatile processors that can restore the system state when the device loses power), high density (beneficial for size), and low-static power (beneficial for low energy consumption) [29, 36]. There are several on-going research efforts towards exploiting emerging NVM technologies for IoT applications. For instance, Ueki et al. [37] developed a ReRAM-based memory architecture for IoT applications, demonstrating its low-power benefits. Ghoneim et al. [38] review how NVMs' ultra-dense and ultra-low-power characteristics can be leveraged in designing physically flexible electronic systems for IoT applications.

An important area of research that still warrants further studies involves understanding the variable runtime execution characteristics of IoT applications. Given these variable characteristics, new approaches are also required to design adaptable memories and memory management techniques that allow the memory resources to be dynamically adapted to the applications' requirements. An example of a promising NVM technology for right-provisioned IoT edge computing is the *racetrack memory*, which also offers the aforementioned advantages of NVMs. Racetrack memory, like other NVMs, further offers the advantage of flexibility, since it has several device level design parameters [39]. These parameters must be explored for right-provisioning for a variety of IoT devices. Similarly,

there has also been much prior research into relaxing the retention time of NVMs (e.g., STT-RAM) to trade off the non-volatility for higher energy efficiency. However, given that applications' retention time requirements are highly variable [40], much work still remains to be done to study the applications' requirements and designing adaptable NVM architectures that satisfy those requirements.

4 CONCLUSIONS

This paper presented an overview of the challenge of right-provisioned IoT edge computing. We defined right-provisioned IoT edge computing as a paradigm in which IoT devices are equipped with 'just enough' computing capability to satisfy their functional requirements and resource constraints. We highlighted a few research directions that must be addressed, with respect to high-level synthesis, energy supply, memory, and computation migration. We hope that this paper will motivate researchers to address some of these challenges in order to further spur the growth of the IoT.

REFERENCES

[1] H. Sundmaeker, P. Guillemin, P. Friess, and S. Woelfflé, *Vision and challenges for realising the Internet of Things.* EUR-OP, 2010, vol. 20, no. 10.

[2] B. Varghese, N. Wang, S. Barbhuiya, P. Kilpatrick, and D. S. Nikolopoulos, "Challenges and opportunities in edge computing," in *2016 IEEE International Conference on Smart Cloud (SmartCloud).* IEEE, 2016, pp. 20–26.

[3] F. Samie, L. Bauer, and J. Henkel, "Iot technologies for embedded computing: A survey," in *Proceedings of the Eleventh IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis.* ACM, 2016, p. 8.

[4] P. Mach and Z. Becvar, "Mobile edge computing: A survey on architecture and computation offloading," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 3, pp. 1628–1656, 2017.

[5] J.-E. Kim, T. Abdelzaher, L. Sha, A. Bar-Noy, R. Hobbs, and W. Dron, "On maximizing quality of information for the internet of things: A real-time scheduling perspective," in *2016 IEEE 22nd International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA).* IEEE, 2016, pp. 202–211.

[6] T. Savolainen, J. Soininen, and B. Silverajan, "Ipv6 addressing strategies for iot," *IEEE Sensors Journal*, vol. 13, no. 10, pp. 3511–3519, 2013.

[7] S. Al-Sarawi, M. Anbar, K. Alieyan, and M. Alzubaidi, "Internet of things (iot) communication protocols," in *2017 8th International Conference on Information Technology (ICIT).* IEEE, 2017, pp. 685–690.

[8] C. Shi, J. Liu, H. Liu, and Y. Chen, "Smart user authentication through actuation of daily activities leveraging wifi-enabled iot," in *Proceedings of the 18th ACM International Symposium on Mobile Ad Hoc Networking and Computing.* ACM, 2017, p. 5.

[9] S.-T. Khang, J. W. Yu, and W.-S. Lee, "Compact folded dipole rectenna with rf-based energy harvesting for iot smart sensors," *Electronics Letters*, vol. 51, no. 12, pp. 926–928, 2015.

[10] T. Adegbija, A. Rogacs, C. Patel, and A. Gordon-Ross, "Microprocessor optimizations for the internet of things: A survey," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2017.

[11] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 4, pp. 2322–2358, 2017.

[12] T. Adegbija, A. Rogacs, C. Patel, and A. Gordon-Ross, "Enabling right-provisioned microprocessor architectures for the internet of things," in *International Mechanical Engineering Congress and Exposition.* ASME, 2015.

[13] N. D. Lane, S. Bhattacharya, P. Georgiev, C. Forlivesi, and F. Kawsar, "An early resource characterization of deep learning on wearables, smartphones and internet-of-things devices," in *Proceedings of the 2015 international workshop on internet of things towards applications.* ACM, 2015, pp. 7–12.

[14] Y. Yamaga, Y. Deguchi, S. Fukuyama, and K. Takeuchi, "5x reliability enhanced 40nm taox approximate-reram with domain-specific computing for real-time image recognition of iot edge devices," in *2018 IEEE Symposium on VLSI Technology.* IEEE, 2018, pp. 109–110.

[15] S. Lerner and B. Taskin, "Workload-aware asic flow for lifetime improvement of multi-core iot processors," in *2017 18th International Symposium on Quality Electronic Design (ISQED).* IEEE, 2017, pp. 379–384.

[16] Z. Wang, Y. Liu, Y. Sun, Y. Li, D. Zhang, and H. Yang, "An energy-efficient heterogeneous dual-core processor for internet of things," in *2015 IEEE international symposium on circuits and systems (ISCAS).* IEEE, 2015, pp. 2301–2304.

[17] H. Jayakumar, K. Lee, W. S. Lee, A. Raha, Y. Kim, and V. Raghunathan, "Powering the internet of things," in *Proceedings of the 2014 international symposium on Low power electronics and design.* ACM, 2014, pp. 375–380.

[18] G. Martin and G. Smith, "High-level synthesis: Past, present, and future," *IEEE Design & Test of Computers*, vol. 26, no. 4, pp. 18–25, 2009.

[19] A. Agarwal, M. C. Ng *et al.*, "A comparative evaluation of high-level hardware synthesis using reed–solomon decoder," *IEEE Embedded Systems Letters*, vol. 2, no. 3, pp. 72–76, 2010.

[20] A. M. Rahmani, P. Liljeberg, J.-S. Preden, and A. Jantsch, *Fog computing in the internet of things: Intelligence at the edge.* Springer, 2017.

[21] K. Kumar and Y.-H. Lu, "Cloud computing for mobile users: Can offloading computation save energy?" *Computer*, no. 4, pp. 51–56, 2010.

[22] Y. Wang, M. Sheng, X. Wang, L. Wang, and J. Li, "Mobile-edge computing: Partial computation offloading using dynamic voltage scaling," *IEEE Transactions on Communications*, vol. 64, no. 10, pp. 4268–4282, 2016.

[23] S. Dey, A. Mukherjee, A. Pal, and P. Balamuralidhar, "Partitioning of cnn models for execution on fog devices," in *Proceedings of the 1st ACM International Workshop on Smart Cities and Fog Computing.* ACM, 2018, pp. 19–24.

[24] M. Xu, F. Qian, and S. Pushp, "Enabling cooperative inference of deep learning on wearables and smartphones," *arXiv preprint arXiv:1712.03073*, 2017.

[25] I. Verbauwhede, "Vlsi design methods for low power embedded encryption," in *2016 International Great Lakes Symposium on VLSI (GLSVLSI).* IEEE, 2016, pp. 7–7.

[26] T. Xu, J. B. Wendt, and M. Potkonjak, "Security of iot systems: Design challenges and opportunities," in *Proceedings of the 2014 IEEE/ACM International Conference on Computer-Aided Design.* IEEE Press, 2014, pp. 417–423.

[27] S. D. Kumar, H. Thapliyal, and A. Mohammad, "Ee-spfal: A novel energy-efficient secure positive feedback adiabatic logic for dpa resistant rfid and smart card," *IEEE Transactions on Emerging Topics in Computing*, 2016.

[28] N. A. Carreon, S. Lu, and R. Lysecky, "Hardware-based probabilistic threat detection and estimation for embedded systems," in *2018 IEEE 36th International Conference on Computer Design (ICCD).* IEEE, 2018, pp. 522–529.

[29] H. Jayakumar, A. Raha, Y. Kim, S. Sutar, W. S. Lee, and V. Raghunathan, "Energy-efficient system design for iot devices," in *2016 21st Asia and South Pacific Design Automation Conference (ASP-DAC).* IEEE, 2016, pp. 298–301.

[30] C. Pan, M. Xie, and J. Hu, "Enzyme: An energy-efficient transient computing paradigm for ultralow self-powered iot edge devices," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, no. 11, pp. 2440–2450, Nov 2018.

[31] J. Lee, Y. Zhang, Q. Dong, W. Lim, M. Saligane, Y. Kim, S. Jeong, J. Lim, M. Yasuda, S. Miyoshi, M. Kawaminami, D. Blaauw, and D. Sylvester, "19.2 a 6.4pj/cycle self-tuning cortex-m0 iot processor based on leakage-ratio measurement for energy-optimal operation across wide-range pvt variation," in *2019 IEEE International Solid- State Circuits Conference - (ISSCC)*, Feb 2019, pp. 314–315.

[32] M. B. Taylor, "Is dark silicon useful? harnessing the four horsemen of the coming dark silicon apocalypse," in *DAC Design Automation Conference 2012.* IEEE, 2012, pp. 1131–1136.

[33] S. Yue, D. Zhu, Y. Wang, and M. Pedram, "Reinforcement learning based dynamic power management with a hybrid power supply," in *2012 IEEE 30th International Conference on Computer Design (ICCD)*, Sep. 2012, pp. 81–86.

[34] D. Bortolotti, H. Mamaghanian, A. Bartolini, M. Ashouei, J. Stuijt, D. Atienza, P. Vandergheynst, and L. Benini, "Approximate compressed sensing: ultra-low power biosignal processing via aggressive voltage scaling on a hybrid memory multi-core processor," in *Proceedings of the 2014 international symposium on Low power electronics and design.* ACM, 2014, pp. 45–50.

[35] X. Li, K. Ma, S. George, J. Sampson, and V. Narayanan, "Enabling internet-of-things with opportunities brought by emerging devices, circuits and architectures," in *IFIP/IEEE International Conference on Very Large Scale Integration-System on a Chip.* Springer, 2016, pp. 1–23.

[36] K. Ma, X. Li, J. Li, Y. Liu, Y. Xie, J. Sampson, M. T. Kandemir, and V. Narayanan, "Incidental computing on iot nonvolatile processors," in *Proceedings of the 50th Annual IEEE/ACM International Symposium on Microarchitecture.* ACM, 2017, pp. 204–218.

[37] M. Ueki, K. Takeuchi, T. Yamamoto, A. Tanabe, N. Ikarashi, M. Saitoh, T. Nagumo, H. Sunamura, M. Narihiro, K. Uejima *et al.*, "Low-power embedded reram technology for iot applications," in *2015 Symposium on VLSI Technology (VLSI Technology).* IEEE, 2015, pp. T108–T109.

[38] M. Ghoneim and M. Hussain, "Review on physically flexible nonvolatile memory for internet of everything electronics," *Electronics*, vol. 4, no. 3, pp. 424–479, 2015.

[39] H. Zhang, C. Zhang, Q. Hu, C. Yang, and J. Shu, "Performance analysis on structure of racetrack memory," in *2018 23rd Asia and South Pacific Design Automation Conference (ASP-DAC).* IEEE, 2018, pp. 367–374.

[40] K. Kuan and T. Adegbija, "Lars: Logically adaptable retention time sttram cache for embedded systems," in *Design, Automation & Test in Europe Conference & Exhibition, 2018. DATE'18.* IEEE Computer Society, 2018.