This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCAD.2017.2717782, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems

Microprocessor Optimizations for the Internet of Things: A Survey

Tosiron Adegbija, *Member, IEEE,* Anita Rogacs, Chandrakant Patel, *Fellow, IEEE,* and Ann Gordon-Ross, *Member, IEEE,*

***Abstract*—The Internet of Things (IoT) refers to a pervasive presence of interconnected and uniquely identifiable physical devices. These devices' goal is to gather data and drive actions in order to improve productivity, and ultimately reduce or eliminate reliance on human intervention for data acquisition, interpretation and use. The proliferation of these connected low-power devices will result in a data explosion that will significantly increase data transmission costs with respect to energy consumption and latency. *Edge computing* reduces these costs by performing computations at the edge nodes, prior to data transmission, to interpret and/or utilize the data. While much research has focused on the IoT's connected nature and communication challenges, the challenges of IoT embedded computing with respect to device microprocessors has received much less attention. This article explores IoT applications' execution characteristics from a microarchitectural perspective and the microarchitectural characteristics that will enable efficient and effective edge computing. To tractably represent a wide variety of next-generation IoT applications, we present a broad IoT application classification methodology based on application functions, to enable quicker workload characterizations for IoT microprocessors. We then survey and discuss potential microarchitectural optimizations and computing paradigms that will enable the design of right-provisioned microprocessors that are efficient, configurable, extensible, and scalable. Our work provides a foundation for the analysis and design of a diverse set of microprocessor architectures for next-generation IoT devices.**

***Index Terms*—Internet of Things, edge computing, low-power embedded systems, microprocessor optimizations, IoT survey, adaptable microprocessors, heterogeneous architectures, energy harvesting, approximate computing.**

I. Introduction and Motivation

THE Internet of Things (IoT) is an emerging technology that refers to a pervasive presence of interconnected and uniquely identifiable physical devices, comprising an expansive variety of devices, protocols, domains, and applications. The IoT will involve devices that gather data and drive actions in order to improve productivity, and ultimately reduce or eliminate reliance on human intervention for data acquisition, interpretation, and use [9]. The IoT has been described as one of the disruptive technologies that will transform life, business, and the global economy [66]. Based on analysis of key potential IoT use-cases (e.g., healthcare, smart cities, smart home, transportation, manufacturing, etc.), it has been estimated that by 2020, the IoT will constitute a trillion dollar economic impact and include more than 50 billion low-power devices that will generate petabytes of data [29], [91], [106].

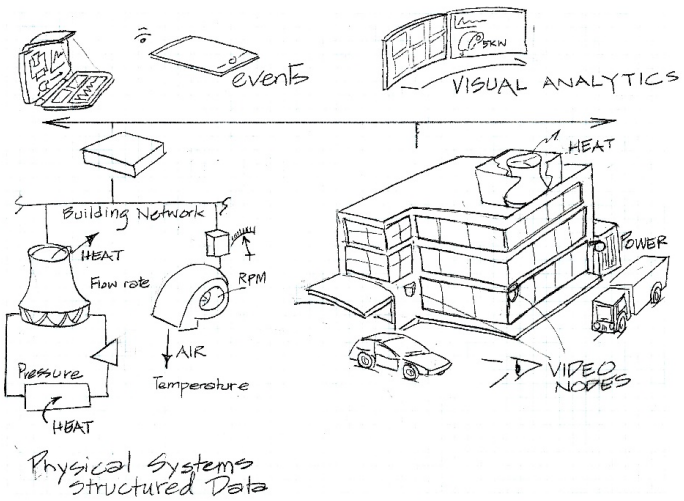

Fig. 1: Illustration of the high-level components of the Internet of Things.

Due to the IoT's expected growth and potential impact, much research has focused on the IoTs communication and software layer [11], [34], [61], [68], however, the challenges of IoT computing, especially with respect to device microprocessors, has received much less attention. Computing on IoT devices introduces new substantial challenges, since IoT devices' microprocessors must satisfy increasingly growing computational and memory demands, maintain connectivity, and adhere to stringent design and operational constraints, such as low cost, low energy budgets, and in some cases, real-time constraints. These challenges necessitate new research focus on microarchitectural optimizations that will enable designers to develop right-provisioned architectures that are efficient, configurable, extensible, and scalable for next-generation IoT devices.

Figure 1 depicts an IoT use-case that illustrates the high-level components of the traditional IoT model. The IoT typically comprises of several low-power/low-performance edge nodes, such as sensor nodes, that gather data and transmit the data to high-performance head nodes, such as servers,

T. Adegbija is with the Department of Electrical and Computer Engineering, University of Arizona, USA, e-mail: tosiron@email.arizona.edu.

A. Rogacs and C. Patel are with Hewlett-Packard (HP) Labs, USA, e-mail: rogacs@hp.com, chandrakant.patel@hp.com.

A. Gordon-Ross is with the University of Florida, USA and the Center for High Performance Reconfigurable Computing (CHRE) at UF. e-mail: ann@ece.ufl.edu.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCAD.2017.2717782, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems

that perform computations for visualization and analytics. In a data center, for example, data aggregation from edge nodes facilitates power and cooling management [14], [73].

However, the growth of the IoT and the resulting exponential increase in acquired/transmitted data poses significant bandwidth and latency challenges. These challenges are exacerbated by the intrinsic resource constraints of most embedded edge nodes (e.g., size, battery capacity, real-time deadlines, cost, etc.). These resource constraints must be taken into account in the design process, and may make it more difficult to achieve design objectives (e.g., minimizing energy, size, etc.). Additionally, increasing consumer demands for high-performance IoT applications will necessitate acquisition and transmission of complex data. For example, a potentially impactful IoT use-case is medical diagnostics [67]. With the advent of technological advances such as cheap portable magnetic resonance imaging (MRI) devices and portable ultrasound machines, several gigabytes (GBs) of high resolution images will be transmitted to medical personnel for remote data processing and medical diagnosis. In some cases, this system must scale to a network of several portable medical devices that transfer data to medical personnel. Transmitting this data will result in bandwidth bottlenecks and pose additional challenges for real-time scenarios (e.g., medical emergencies) where the latency must adhere to stringent deadline constraints.

The IoT can also incur significant, and potentially unsustainable, energy overheads. Previous work [13], [57] established that energy consumed while transmitting data is significantly more than the energy consumed while performing computations on the data. For example, the energy required by Rockwell Automations sensor nodes to transmit one bit of data is 1500-2000X more than the energy required to execute a single instruction (depending on the transmission range and specific computations) [76].

To address these challenges, *fog computing* [16] has been proposed as a virtualized platform that provides compute, storage, and networking services between edge nodes and cloud computing data centers. Rather than performing computations in the cloud, fog computing reduces the bandwidth bottleneck and latency by moving computation closer to the edge nodes. Our study focuses on further reducing the bandwidth, latency, and energy consumption through *edge computing*, where the edge nodes are directly equipped with sufficient computation capacity in order to minimize data transmission [4].

Edge computing performs computations that process, interpret, and use data at the edge nodes. Performing these computations on the edge nodes minimizes data transmission, thereby improving latency, bandwidth, and energy consumption. For example, in the medical diagnostics use-case described above, rather than sending several GBs of MRI data to the medical personnel for diagnoses, the portable MRI machine (the edge node) is equipped with sufficient computational capabilities and algorithms to extract information and interpret the data. Only processed data (e.g., information about an anomaly in the patient) is transmitted to the medical personnel, thus speeding up the diagnoses and reducing the MRI machine's energy consumption. Alternatively, the data could be quantifiably reduced using intelligent algorithms and computations, such that only important information is transmitted to the medical personnel.

Gaura et al. [30] examined the benefits of edge mining, in which data mining takes place on the edge devices. The authors showed that edge mining has the potential to reduce the amount of transmitted data, thus reducing energy consumption and storage requirements. However, the edge nodes computing capabilities must be sufficient/right-provisioned to perform and sustain the required computations, while adhering to the nodes design constraints (e.g., form factor, energy consumption, etc.) [83].

This paper explores microarchitectural optimizations and emerging computing paradigms that will enable edge computing on the IoT. To ensure that microprocessor architectures designed and/or selected for the IoT have sufficient computing capabilities, a holistic approach, involving both application and microarchitecture characteristics, must be taken to determine microarchitectural design tradeoffs. However, due to the wide variety of IoT applications and the diverse set of available architectures, determining the appropriate architectures is very challenging. The study presented herein seeks to address these challenges and motivate future research in this direction.

In this paper, we perform an expansive study and characterization of the emerging IoT application space and propose an application classification to broadly represent IoT applications with respect to their execution characteristics. To enable the design of right-provisioned microprocessors, we propose the use of computational kernels that provide a tractable starting point for representing key computations that occur in the IoT application space. Using computational kernels, rather than full applications, follows the computational dwarfs methodology [8] and allows IoT computational patterns to be accurately represented at a high level of abstraction.

Furthermore, we propose a high-level design methodology for identifying right-provisioned architectures for edge computing use-cases, based on the executing applications and the applications execution characteristics (e.g., compute intensity, memory intensity, etc.). Finally, in order to motivate future research, we survey a few potential microprocessor optimizations and computing paradigms that will enable the design of right-provisioned IoT microprocessor architectures.

II. High-level IoT Characteristics and their Demands on Microprocessor Architectures

The IoT's characteristics necessitate new designs and optimizations for microprocessors that will be employed in IoT devices. We briefly describe seven key characteristics—based on previous research [75]—that, together, distinguish the IoT from other connected systems: *intelligence, heterogeneity, complexity, scale, real-time constraints, spatial constraints,* and *inter-node support*. We also describe the demands that these characteristics place on microprocessor architectures

- **Intelligence:** Since the goal of the IoT is to reduce reliance on human intervention for data acquisition and use [9], raw data must be autonomously collected and processed to create actionable information. IoT microprocessors must be able to dynamically adapt to varying

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCAD.2017.2717782, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems

runtime execution scenarios and adaptable data characteristics [32].

- **Heterogeneity:** One of the key characteristics of the IoT is that it involves a high degree of heterogeneity, featuring different kinds of devices, applications, and contexts [75]. Thus, IoT microprocessors must be specialized to the different execution characteristics of IoT applications. IoT microprocessor heterogeneity may be *chip-level*—a single chip with heterogeneous cores—or *network-level*, where different devices feature different kinds of cores. Despite this heterogeneity, the devices must be able to seamlessly communicate with each other and share resources for efficient data interpretation and use.
- **Complexity:** The organization and management of the IoT will be very complex. Apart from the large numbers of heterogeneous architectures, the architectures must be able to execute a wide variety of applications, many of which may be memory- and compute-intensive. Interactions between the different IoT devices will dynamically vary. Some devices will be added to the IoT network, while others will be removed; these changes may impact individual devices' execution behaviors.
- **Scale:** The IoT will comprise more than 50 billion devices by 2020, and the numbers are expected to grow continuously [91]. In addition to the increase in the number of devices, the interactions among them will also increase. To support this scale, IoT microprocessors must be efficient—cost, energy, and area efficient—and constitute minimal overhead to the IoT device. In addition, the microprocessors must be able to portably execute different kinds of applications.
- **Real-time constraints:** Some of the most important IoT use-cases—for example, patient monitoring, medical diagnostics, aircraft monitoring—involve real-time constraints, where execution must adhere to stringent deadlines. IoT microprocessors must be able to dynamically determine and adhere to deadlines, based on various inputs, such as user inputs, application characteristics, quality of service.
- **Spatial constraints:** Several IoT use-cases are location-based. An IoT device's location may change throughout the device's lifetime. In addition, the device may be exposed to variable, and potentially non-ideal, environmental conditions. For example, tracking devices may be exposed to extreme heat, extreme cold, and/or rain at different times or in different locations. Thus, IoT microprocessors must feature fault tolerance and adaptability that allows them to adhere to variable operation conditions.
- **Inter-node support:** The IoT will comprise of several devices/nodes that can share execution resources among each other. Due to the wide variety of IoT applications that may execute on a device, and the stringent resource constraints, it may be impractical to equip every device with all the execution resources it will require throughout its lifetime. Thus, to maintain efficient execution, IoT devices must be able to share execution resources with each other, when necessary.

III. IoT Application Classification

The IoT offers computing potential for many application domains, including transportation and logistics, healthcare, smart environment, personal and social domains [11], etc. One of the key goals of the IoT, from an edge computing perspective, is to equip edge devices with sufficient resources to perform computations that would otherwise have been transferred to a high-performance device. In order to rightly provision these devices, we must first understand potential applications that will be executed on the devices.

Previous works have proposed classifications for various IoT components. Gubbi et al. [34] presented a taxonomy for a high level definition of IoT components with respect to hardware, middleware, and presentation/data visualization. Tilak et al. [93] presented a taxonomy to classify wireless sensor networks according to different communication functions, data delivery models, and network dynamics. Tory et al. [94] presented a high level visualization taxonomy that classified algorithms based on the characteristics of the data models.

However, there is currently very little research that characterizes these applications with respect to their execution characteristics. One of the biggest challenges the IoT presents is the huge number and diversity of use-cases and potential applications that will be executed on IoT devices. This challenge is exacerbated by the fact that only a small fraction of these applications are currently available in society. Thus, a significant amount of foresight is required in designing microprocessor architectures to support the IoT's emergence and growth.

Much prior work has characterized IoT applications according to different use-cases and domains. For example, Atzori et al. [11] and Sundmaeker et al. [91] categorized IoT applications into three domains: industry, environment, and society. Asin et al. [10] categorized IoT applications into 54 domains under twelve categories. In this work, our goal is a tractable and extensible classification that enables us to identify the IoT applications' key execution characteristics.

As an initial step towards understanding IoT applications' execution characteristics, we performed an expansive study of IoT use-cases and the application functions present in these use-cases. Since it is impractical to consider every IoT application within these use-cases/application domains, based on our study, we propose an application classification methodology that provides a high level, broad, and tractable representation of a variety IoT applications using the application functions. Our IoT application classification consists of six key application functions:

- *sensing*
- *communications*
- *image processing*
- *compression (lossy/lossless)*
- *security*
- *fault tolerance*.

We note that this classification is not exhaustive; however, it represents a wide variety of current and potential IoT applications. The classification also provides an extensible framework that allows emerging applications/application domains to be

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCAD.2017.2717782, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems

analyzed. In this section, we describe the application functions and motivate these functions using a medical diagnostics use-case, where applicable, or other specific examples of current and/or emerging IoT applications.

A. Sensing

Sensing involves data acquisition (e.g., temperature, pressure, motion, etc.) about objects or phenomena, and will remain one of the most common functions in IoT applications. In these applications, activities, information, and data of interest are gathered for further processing and decision making. We use sensing in our IoT application classification to represent applications where data acquired using sensors must be converted to a more useable form. Our motivating example for sensing applications is *sensor fusion* [70], where sensed data from multiple sensors are fused to create data that is considered qualitatively or quantitatively more accurate and robust than the original data.

Sensor fusion algorithms can involve various levels of complexity and compute/memory intensity. For example, sensor fusion could involve aggregating data from various sources using simple mathematical computations, such as addition, minimum, maximum, mean, etc. Alternatively, sensor fusion could involve more computationally complex/expensive applications, such as fusing vector data (e.g., video streams from multiple sources), which requires a substantial increase in intermediate processing.

In a medical diagnostics use-case, for example, sensing is vital in a body area network [19], where non-invasive sensors can be used to automatically monitor a patients physiological activities, including blood pressure, heart rate, motion, etc. Several sensing devices, such as portable electrocardiography (ECG), electroencephalography (EEG), and electromyography (EMG) machines, motion and blood pressure sensors could be equipped with additional computational resources and algorithms that enable the devices to not only gather data, but also analyze the data in order to reduce the amount of transmitted data, with minimal energy or area overheads.

B. Communications

Communications is one of the most common IoT application functions due to the IoTs intrinsic connected structure, where data transfers traverse several connected nodes. There are many communication technologies (e.g., Bluetooth, Wi-Fi, etc.), and communication protocols (e.g., transfer control protocol (TCP), the emerging 6lowpan (IPv6 over low power wireless personal area network), etc.). In this work, we highlight *software defined radio (SDR)* [59], which is a communication system in which physical layer functions (e.g., filters, modems, etc.) that are typically implemented in hardware are implemented in software.

SDR is an emerging and rapidly developing communication system that is driving the innovation of communications technology, and promises to impact all areas of communication. SDR is growing in popularity, and attractive for the IoT, because of its inherent flexibility, which allows for flexible incorporation and enhancements of multiple radio functions, bands, and modes, without requiring hardware updates. SDR typically involves an antenna, an analog-to-digital converter (ADC) connected to an antenna (for receiving) and a digital to analog converter (DAC) connected to the antenna (for transmitting). Digital signal processing (DSP) operations (e.g., Fast Fourier Transform (FFT)) are then used to convert the input signals to any form required by the application.

Even though SDR applications are typically compute intensive, with small data and instruction memory footprints, recent work [20] shows that the overheads of SDR can be kept small in the IoT domain by focusing on optimizing the key kernels (e.g., Synchronization and Finite Impulse Response (FIR)) that dominate SDR computations and power consumption. In general, SDR algorithms can be efficiently executed using general purpose microprocessors or more specialized processors, such as digital signal processors (DSPs) or field-programmable gate arrays (FPGAs). Alternatively, heterogeneous architectures [40] can also combine different kinds of microprocessors to satisfy different operations' execution requirements while minimizing overheads, such as energy consumption. Other examples of communication applications include packet switching and TCP/IP.

C. Image Processing

In the IoT context, image processing represents applications that involve any form of signal processing where the input is an image or video stream from which characteristics/parameters must be extracted/identified. Additionally, this classification also involves applications in which an image/video input must be converted to a more usable form. Several emerging IoT applications, such as automatic number license plate recognition, traffic sign recognition, face recognition, etc., involve various forms of image processing. For example, face recognition involves operations, such as face detection, landmark recognition, feature extraction, and feature classification, all of which involve image processing.

Image processing is important for several impactful IoT use-cases, and necessitates microarchitectures that can efficiently perform image processing operations. For example, in medical diagnostics, image processing can be used to increase the reliability and reproducibility of disease diagnostics. Image processing can provide medical personnel with quantitative data from historical images, which can be used to supplement qualitative data currently used by specialists. In addition, portable medical devices, e.g., portable ultrasounds, can be equipped with image processing applications to provide speedy analysis for remote assessment of patients [69].

The National Institute of Health (NIH) supports the Medical Image Processing, Analysis, and Visualization (MIPAV) application [67], which enables medical researchers to easily share research data and enhance their ability to diagnose, monitor, and treat medical disorders. However, since image processing applications are typically data-rich, and both memory and compute intensive, novel optimization techniques are required to enable the efficient execution of these applications in the context of IoT edge computing. Furthermore, some image processing applications require large input, intermediate, or

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCAD.2017.2717782, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems

output data to be stored (e.g., medical imaging), thus requiring a large amount of storage.

D. Compression

With the increase in data and bandwidth-limited systems, compression can reduce communication requirements to ensure that data is quickly retrieved, transmitted, and/or analyzed. Several emerging IoT use-cases will involve large volumes of data, which will necessitate efficient compression techniques to accommodate the rapid growth of the data and reduce transmission latency and bandwidth costs [100]. Additionally, since most IoT devices are resource-constrained, compression also reduces storage requirements when data must be stored on the edge node. For example, data gathered using sensors in a body area network can be quantifiably and intelligently reduced in order to minimize transmission and storage requirements for medical diagnosis devices.

Compression involves encoding information using fewer bits than the original representation. The data can be encoded at the data source before storage or transmission, known as source encoding, or during transmission, known as channel coding [6]. In our studies, however, we focus on source encoding, as this type of encoding will be more relevant in the context of edge computing.

Compression techniques can be broadly classified as *lossy* or *lossless* compression. Lossy compression (e.g., JPEG) typically exploits the perceptibility of the data in question, and removes unnecessary data, such that the lost data is imperceptible to the user. Alternatively, lossless compression removes statistically redundant data in order to concisely represent data. Lossless compression typically achieves a lower compression ratio and is usually more compute and memory intensive than lossy compression. However, lossy compression may be unsuitable in some scenarios where high data fidelity is required to maintain the quality of service (QoS) (e.g., in medical imaging).

E. Security

Since IoT devices are often deployed in open or potentially unsafe environments, where the devices are susceptible to malicious attacks, security applications are necessary to maintain the integrity of both the devices and the data. Furthermore, sensitive scenarios (e.g., medical diagnostics) may require security applications to prevent unauthorized access to sensitive data and functions. Implantable medical devices, such as pacemakers, implantable cardiac defibrillators, neurostimulators are especially susceptible to potentially fatal security and privacy issues, such as replay attacks [37], [48]. Since medical device security is still in its infancy, there still exists a wide knowledge gap with respect to the microprocessor characteristics that will support security algorithms execution requirements without sacrificing the devices functional requirements.

We highlight data encryption [89], which is a common technique for ensuring data confidentiality, wherein an encryption algorithm is used to generate encrypted data that can only be read/used if decrypted. Data encryption applications (e.g., secure hash algorithm) are typically compute intensive and memory intensive, since encryption speed is also dependent on the memory access latency for data retrieval and storage.

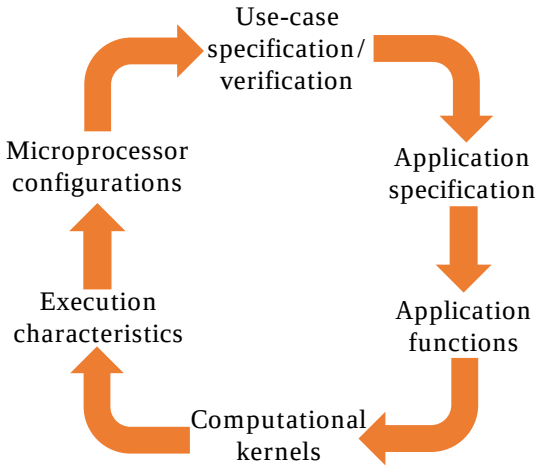

Fig. 2: Illustration of a high-level IoT microprocessor design life-cycle

F. Fault Tolerance

Fault tolerance [39] refers to a system's ability to operate properly when some of its components fail. Fault tolerant applications are especially vital since IoT devices may be deployed in harsh and unattended environments, where QoS must be maintained in potentially adverse conditions, such as cryogenic to extremely high temperatures, shock, vibration, etc. In some emerging IoT devices, such as implantable medical devices, fault tolerance could be the single most critical requirement, since faults can be potentially fatal. Thus, fault tolerance must be incorporated into such devices without accruing significant overheads.

Fault tolerance can be achieved in different ways. Hardware-based techniques usually rely on redundancy—RAID (redundant array of independent disks) [74] is a common example—wherein redundant disks or devices are used to provide fault tolerance in the event of a failure. This kind of redundancy can be achieved in IoT devices using a dedicated IoT device, or integrated into a larger, less constrained device, in order to minimize the attendant overheads of redundancy. Alternatively, redundancy can be incorporated directly into the IoT devices, at the expense of area and power overheads. To reduce the overheads from hardware-based fault tolerance, software-based fault tolerance can also be employed. Software-based fault tolerance [39], [80], [95] involves applications and algorithms that perform operations, such as memory scrubbing, cyclic-redundancy checks (CRC), error detection and correction, etc.

IV. Determining IoT Microprocessor Configurations

One of the major challenges for IoT microprocessor design is determining the best microprocessor configurations that satisfy the IoT device's execution requirements. In this section,

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCAD.2017.2717782, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems

TABLE I: Application functions and sample representative kernels.

Application function	Kernel
Sensing	dense matrix transpose
Communications	Fast Fourier Transform (FFT)
Image processing	Dense matrix multiplication
Lossy compression	jpeg
Lossless compression	lz4
Security	Secure Hash Algorithm (sha)
Fault tolerance	Cyclic redundancy check (crc)

we describe a sample high-level process through which an IoT microprocessor can be designed and optimized. Figure 2 illustrates a high-level IoT microprocessor design life-cycle, consisting of six steps. First, the use-case needs to be specified. This step describes the overall functionality and behavior of the IoT device, which will dictate the microprocessor requirements. Based on the use-case, the required applications to achieve the desired functionality are then specified. For example, a medical diagnostics use-case involving a portable ultrasound device [50] may require applications for image capture, anomaly detection, anomaly recognition, data encryption, and data transmission. Thereafter, the specific functions within each application are determined, and these functions are broken down into their respective *computational kernels*.

Computational kernels are basic execution blocks that represent applications' functions; the kernels disconnect the executions from specific implementations, programming languages, and algorithms. Using computational kernels can make the design process more manageable, since kernels are faster to simulate and can represent a variety of applications/application functions. In addition, kernels expose computational nuances and reveal execution characteristics that may not be visible when considering the full application. Kernels also provide a fine-grained view of applications, such that additional design processes (e.g., hardware/software partitioning [90]) can be sped up. Using computational kernels to represent application functions is supported by the concept of *computational dwarfs* [8]. Computational dwarfs represent patterns of computation at high levels of abstraction to encompass several computational methods in modern computing.

Table I illustrates sample kernels that can be used to represent the different application classes (Section III). After the computational kernels are determined, the kernels' execution characteristics are then determined. One of the most common way for determining these characteristics is through simulations. During design space exploration [85], the kernels' execution characteristics—memory intensity, compute intensity, instructions per cycle, memory references, etc.—using different microprocessor configurations are then analyzed to determine which configurations best satisfy the application resource requirements. These configurations can then be refined, if necessary, after verifying that the functional requirements of the use-case are met.

V. State of the Art in IoT Microarchitecture Configurations

We performed an extensive survey and study of the state-of-the-art in commercial-off-the-shelf (COTS) embedded systems microprocessor architectures from several designers and manufacturers ranging from low-end microcontrollers to high-end/high-performance low-power embedded systems microprocessors. Our studies included publicly available information on these microprocessors' configurations, and conversations with researchers and engineers directly involved with microprocessor design and development in several manufacturing companies.

Based on our studies, we categorized the microprocessors in terms of several microprocessor characteristics, including number of cores, on-chip memory (e.g., cache), off-chip memory support, power consumption, number of pipeline stages, etc. Using this information, we developed a set of four high-level microarchitecture configurations for IoT edge computing support. These configurations represent the range of available state-of-the-art COTS microprocessors, and provide a reference point from which future IoT microprocessors and optimizations can be developed. While microprocessors could include central processing units (CPUs), graphics processing units (GPUs), DSPs, etc., in this survey, we focus on CPUs, since they are typically the backbone for most edge computing applications.

Table II depicts the microarchitecture configurations, comprising of four configurations: *config1, config2, config3,* and *conf4*, representing different kinds of microprocessors. We highlight specific state-of-the-art microcontroller/microprocessor examples to motivate the configurations, however, we note that these configurations are only representative and not necessarily descriptive.

Config1 represents low-power and low-performance microcontroller units, such as the ARM Cortex-M4 [101] found in several IoT-targeted MCUs from several developers, including Freescale Semiconductors, Atmel, and STMicroelectronics. Conf1 contains a single core with 48 MHz clock frequency, three pipeline stages, in-order execution, and support for 1 MB of flash memory.

Config2 represents recently-developed IoT-targeted CPUs, such as the Intel Quark Technology [78], and contains a single core with 400 MHz clock frequency, five pipeline stages, in-order execution, 16 KB level one (L1) instruction and data caches, and support for 2 GB RAM.

Config3 represents mid-range CPUs, such as the ARM Cortex-A7 [98] found in several general purpose embedded systems, and contains four cores with 1 GHz clock frequency, 8 pipeline stages, in-order execution, 32 KB L1 instruction and data caches, 1 MB level two (L2) cache, and support for 2 GB RAM.

Finally, *config4* represents high-end/high-performance embedded systems CPUs, such as the ARM Cortex-A15 [98], and contains four cores with 1.9 GHz clock frequency, 8 pipeline stages, 32 KB L1 instruction and data caches, 2 MB L2 cache, support for 4 GB RAM, and out-of-order execution. Out-of-order execution allows instructions to execute as soon as the instruction becomes available, unlike in-order execution where instructions must execute in program order.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCAD.2017.2717782, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems

TABLE II: State of the art microprocessor configurations.

	Config1	Config2	Config3	Config4
Sample CPU	ARM Cortex M4	Intel Quark	ARM Cortex A7	ARM Cortex A15
Frequency	48 MHz	400 MHz	1 GHz	1.9 GHz
Number of cores	1	1	4	4
Pipeline stages	3	5	8	15
Cache	None	None	32KB I/D L1, 1MB L2	32KB I/D L1, 2MB L2
Memory	512KB flash	2GB RAM	2GB RAM support	1TB RAM support
Execution	In-order	In-order	In-order	Out-of-order

VI. Microprocessor Optimizations, Computing Paradigms, and Future Directions

Using the configurations described in Section V and the kernels listed in Table I as benchmarks, we performed detailed architectural simulations on GEM5 [15] to analyze the characteristics of the configurations. The details of our analysis can be found in our preliminary work [4]. Based on our analysis and extensive surveys, we have identified five key characteristics that IoT microprocessors must have in order to support the IoT's growth:

- **Efficiency**: Due to the typical stringent resource constraints of IoT devices, IoT microprocessors must be optimized for energy, cost, performance, and area efficiency.
- **Configurability**: One of the major observations from our studies is that different IoT applications have vastly different runtime resource requirements. With the growth of the IoT and the current trend of IoT applications, we envision that this variability in runtime resource requirements will increase even further with next-generation IoT applications. Therefore, these variable runtime resource requirements necessitate adaptable/configurable microarchitectures with configurations that can be autonomously specialized to different applications in order to achieve optimal execution, especially in terms of energy efficiency.
- **Security**: Security must be one of the primary design goals of IoT microprocessors, especially since IoT devices will be inherently more susceptible to attacks.
- **Future-proof**: Designing IoT microprocessors will require a lot of foresight. With the rapid emergence of new IoT applications, and the applications' increasing memory and compute requirements, IoT microprocessors must be able to execute future applications without being over-provisioned for current applications.
- **Extensibility**: Future-proofing IoT microprocessors can be achieved by extending the microprocessors with additional functionalities (e.g., specialized instructions, security monitors, new on-chip peripherals). Thus, IoT microprocessors must be designed with ease of integration, customization, and extension in mind. Such extensibility will allow new levels of performance and energy efficiency to be achieved.

In this section, we survey a few potential microprocessor optimizations and computing paradigms, from a context of edge computing, that will enable the aforementioned IoT characteristics, and support the IoT's growth. We first discuss optimizations for achieving adaptability in IoT microprocessors; we then survey and discuss research directions in other computing paradigms that will enable microprocessor optimizations for the IoT, including *non-volatile processors*, *approximate computing*, *in-memory processing*, and *secure microarchitectures*. Our goal in this section is not to provide an exhaustive survey of potential microprocessor optimizations; our goal is to motivate researchers with future directions for developing novel, configurable, and extensible low-overhead IoT microprocessors. Table III summarizes the different computing paradigms, benefits, and references for further details.

A. Configurable/Adaptable Architectures

In order to achieve the right balance between performance, power, and area in IoT applications, IoT microprocessors must be adaptable to the application requirements. This adaptability can be achieved through the ability to change the microprocessor's configuration at runtime or by heterogeneous processors that offer different processing resources for executing the applications. Several microprocessor components can be configured, including the issue queue [28], reorder buffer [53], register files [1], and pipelines [27]. However, in this subsection, we focus on configurable caches [104] due to their potential impact on the microprocessor's end-to-end energy, performance, and area.

The memory will arguably remain the most important microprocessor component for performance and energy consumption. Since emerging IoT applications will increase in memory and compute intensity, IoT microprocessors must be equipped with more advanced memory hierarchies to take advantage of the spatial and temporal locality of the IoT applications. Due to the memory hierarchy's large impact on system performance and energy consumption, much emphasis must be placed on efficient caching techniques for IoT microprocessors.

Previous work has shown that specializing the cache configurations to different application or phase memory requirements can reduce the memory hierarchy's energy consumption by up to 62% [33]. In addition, our studies revealed that the cache is one of the more easily over-provisioned resources in an IoT microprocessor, resulting in high energy consumption with no performance benefits. The energy consumption can be quantifiably reduced, without any performance degradation, by dynamically changing the cache configurations (e.g., reducing the cache size). Thus, a prominent optimization for IoT microprocessors is dynamically configurable cache architectures that allow the caches parameter values to be specified/changed during runtime.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCAD.2017.2717782, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems

Three major challenges must be addressed in order to enable dynamically configurable caches for IoT microprocessors: *augmenting caches for configurability*, *cache tuning algorithms/heuristics*, and *cache tuners*. In order to maximize the benefits of configurable caches, the required hardware optimizations to enable configurability must accrue minimal overhead. For example, a potential technique for enabling cache configurability uses bit-width configuration registers that allow a caches banks to be shutdown to configure the cache size, or concatenated to configure the cache associativity [104].

For optimal execution, the best configurations that achieve optimization goals and satisfy design constraints for different applications must be dynamically determined. Cache tuning determines the optimal cache configurations that match an application's runtime behavior. The cache tuning algorithm/heuristic can result in time and energy overheads, since the processor must stall during the tuning process. Much previous work (e.g., [2], [33], [36], [71], [104]) have proposed different algorithms/heuristics for cache tuning to minimize the potential of overhead and maximize the cache tuning benefits. These works offer a valuable foundation for IoT microprocessors. The intrinsic characteristics of the IoT necessitate further studies and development of innovative cache tuning techniques for IoT microprocessors that are low-overhead, robust, and versatile for the variety of applications that will execute on these microprocessors.

To orchestrate the cache tuning process, hardware and/or software cache tuners employ cache tuning algorithms/heuristics to determine the best cache configurations to meet design constraints. However, the tuner could impose significant power, area, and/or performance overheads while exploring the configuration design space [3]. Thus, to maximize the benefits of configurable caches in IoT microprocessors, novel cache tuners must be designed such that they constitute minimal overhead and effectively implement the cache tuning algorithms.

B. Distributed Heterogeneous Architectures

Heterogeneous architectures [56] allow a coarse-grained specialization of system resources to application requirements by equipping a microprocessor with different kinds of cores or different core configurations. The different cores execute the same instruction set, but have different capabilities and performance levels. Thus, at runtime, the system software evaluates the resource requirements of applications or application phases and determines the core that best satisfies the optimization goals for the executing applications.

One of the major advantages of heterogeneous architectures for IoT microprocessors, from a design perspective, is that existing cores (e.g., CPUs, DSPs, GPUs, etc.) can be reused in the implementation of heterogeneous microprocessors; this allows previous design and verification efforts to be amortized. However, unlike configurable architectures, heterogeneous architectures offer a much smaller design space, which necessitates greater design time effort in determining the best core configurations that will satisfy the application requirements. In addition, in a system with a large number of applications,

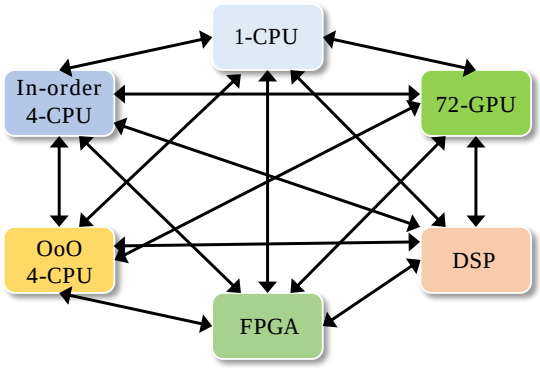

Fig. 3: Distributed heterogeneous architectures.

heterogeneous cores may have a lower optimization potential than configurable cores, since there are fewer configurations to choose from in heterogeneous cores.

Much previous research efforts have targeted heterogeneous cores in general purpose computers, embedded systems, etc., but their applicability to IoT microprocessors have yet to be explicitly determined [68], [88]. Two major challenges that must be addressed in designing heterogeneous microprocessors for the IoT are the number and choice of cores, and scheduling of applications to the appropriate cores. In order to maximize the optimization potential, designers must expend a considerable amount of effort to determine the best cores or configurations to incorporate into the microprocessor. To provide an effective platform that satisfies the execution requirements of a wide variety of application characteristics, the selected cores must cater to a wide range of computational complexities and performance requirements. This effort would require a priori knowledge and analysis of the applications/application domains that will execute on the microprocessor. In addition, potential core configurations and their characteristics must be extensively analyzed.

Given the application execution requirements, the appropriate core on which to execute/schedule the application must also be determined either statically or dynamically [7]. Static scheduling suffices when the applications are known a priori. However, when the applications are unknown, dynamic scheduling evaluates application characteristics at runtime and schedules the applications to the appropriate cores. Much research is needed to develop low-overhead, computationally simple, and accurate scheduling techniques, for IoT microprocessors, that will achieve optimization goals and satisfy the microprocessors resource constraints.

An alternative to heterogeneous cores on a single device is a network of distributed heterogeneous architectures. Figure 3 illustrates the distributed heterogeneous network. Different devices are equipped with different computational resources that may be required by the devices at different times. For example, Figure 3 depicts six nodes containing six different kinds of microprocessors: a single core in-order microprocessor (1-CPU), a 72-core general purpose GPU (72-GPU), a DSP, an FPGA, a quad-core out-of-order processor (OoO 4-CPU), and a quad-core in-order processor (In-order 4-CPU). Assuming an

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCAD.2017.2717782, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems

8-threaded application *A* with deadline constraints arrives on 1-CPU—1-CPU may be under-provisioned for *A*'s execution (i.e., the execution time on 1-CPU will exceed the deadline)—an alternative node in the network can be used to execute *A*. Apart from determining which node contains sufficient resources to execute *A*, the costs—energy and time—to transfer *A* from 1-CPU to the right-provisioned node, in addition to the time to retrieve the results, must not negate the savings from executing *A* on a right-provisioned core

To enable distributed heterogeneous architectures, several research challenges must be addressed: benefits of waiting for a right-provisioned node if the node is busy; tradeoffs of time—execution and transmission times—and energy in the presence or absence of deadline constraints; runtime, low-overhead determination of right-provisioned nodes; developing portable applications and standardizing the communication protocols between different kinds of nodes.

C. Energy Harvesting and Non-Volatile Processors

One of the most important, and most persistent, challenges for IoT devices is energy consumption. Energy harvesting is a promising technique for replacing or supplementing energy sources (e.g., batteries), especially in ultra-low power applications. Energy can be harvested from several sources, including solar, thermal gradients, radio frequency radiation, etc. [18], [31], [35], [60], [77]. These energy sources, however, are typically not reliable; external factors—distance from a power source, physical obstacles, electromagnetic signals—can disrupt the energy supply [97]. Due to this unreliability, traditional processors may be impractical for systems equipped with energy harvesting—when the energy supply is disrupted, volatile processors will lose their operating state.

Non-volatile processors [62] use non-volatile storage components—non-volatile memories—to store the processor state when the power supply is disrupted. When the power is restored, the processor's state is restored from the the non-volatile memory to continue execution. Thus, non-volatile processors allow continuous computation despite power disruptions. For example, one of the earlier non-volatile processors [99] allowed system states to be backed up within 7μs and restored within 3μs. Additionally, since embedded systems typically spend a significant amount of time idling, resulting in high leakage power, non-volatile processors can reduce the idle power by allowing the system to be shut down while idle. The state can then be instantaneously restored on wake-up [63], [102].

There are several potential non-volatile architectures that can be used at different abstraction levels to achieve non-volatile processors for energy harvesting systems. For example, various FeRAMs, STT-RAMs, PCRAMs, ReRAMs have been explored for use in energy harvesting systems [22], [62], [72], [92], [108]. Several factors must be considered when selecting an energy harvesting non-volatile processor system. The input power characteristics—for example, the power behavior when interrupted—affect the choice of non-volatile architectures [65]. The application characteristics also affect the choice of non-volatile architectures. Application characteristics, such as deadlines, real-time, and quality of service (QoS) requirements, must also be taken into consideration. For example, solar powered systems can typically be used to meet real-time QoS requirements more effectively than RF or thermal source [65]. Much research is required to quantify the tradeoffs of different energy sources with respect to the non-volatile architectures.

D. Approximate Computing

Approximate computing has recently gained a lot of traction as a viable alternative to exact computing. Exact computing targets exact numerical or Boolean equivalence, while approximate computing allows a non-exact, inaccurate result that maintains the desired output quality [21]. Approximate computing allows new optimization options for processors that execute *resilient applications*—applications that can produce outputs of sufficient quality despite some imprecise computations, e.g., signal processing, multimedia, graphics, etc. Allowing bounded approximation in processors can provide significant performance and energy gains, while achieving an acceptable amount of accuracy.

Approximate computing can enable energy-efficient edge computing in IoT devices, such as wearable electronics [54], [81]. One of the first steps to incorporating approximate computing into IoT devices is identifying the devices' applications, and the applications' resilience to computing error. In [21], Chippa et al. presented an *automatic resilience characterization* framework to evaluate how amenable an application is to approximate computing. This framework uses approximation models that evaluate different approximate computing techniques for different partitions of an application. This framework relies on significant amounts of a priori knowledge about the executing applications, such as, the computational patterns and input data. Since applications typically have different execution phases, and the phase behaviors could change at runtime, key to efficient approximation is a runtime framework that automatically detects resilient application phases and adjusts the computing exactness to match the currently executing phase's resilience [42].

Several optimizations to enable approximate computing have been developed at different abstraction levels. Using various digital signal processing filters and an electrocardiogram (ECG) application, Venkataraman et al. [96] presented a system-level design flow to study a system's exactness and used elimination heuristics to explore the design space under inexactness, area, and energy constraints. Several other techniques have been proposed [38] for achieving inexactness at the circuit level through different approximate components, such as approximate adders [26], [64], approximate multipliers [55], [58], and approximate logic synthesis [86], [87]. Similarly, approximate computing can also be exploited at the memory level, for example, by storing data approximately [82] or through systems that can tolerate memory errors while maintaining the desired quality of service [17].

E. In-Memory Processing

In-memory processing—or *processing in memory*—has been studied in relation to big data and distributed com-

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCAD.2017.2717782, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems

TABLE III: Summary of computing paradigms and potential benefits

Computing Paradigm	Benefits	References
Configurable architectures	Efficiency, future-proof, configurability	[1]–[3], [27], [28], [33], [36], [53], [71], [104]
Distributed heterogeneous architectures	Extensibility, efficiency, future-proof	[56], [68], [88]
Energy harvesting and non-volatile processors	Efficiency	[18], [31], [35], [60], [62], [77], [97], [22], [63], [65], [72], [92], [99], [102], [108]
Approximate computing	Efficiency	[21], [26], [38], [42], [54], [64], [81], [96], [17], [55], [58], [82], [86], [87]
In-memory processing	Efficiency, extensibility	[5], [24], [41], [44]–[46], [103], [105]
Secure microarchitectures	Security	[12], [23], [25], [47], [49], [51], [52], [79], [84], [107]

puting systems [41], [103]. In-memory processing addresses the well-known processor-memory performance gap through internal memory accesses; it avoids delays caused by off-chip communication. Caches have been widely used to bridge the processor-memory performance gap; in-memory processing further reduces this gap by allowing computations to be performed *on* the memory chip without the need for processor-to-memory communication. With the growth of the IoT, massive amounts of data generated, and resource constraints of IoT devices, in-memory processing offers an attractive optimization for IoT devices.

One of the major attractions of in-memory processing for IoT devices, in the context of edge computing, is the need for real-time in-situ data processing on large data volumes. The data processing must be energy-efficient and involve low hardware overhead. To achieve such capabilities, researchers have explored inherently robust brain-inspired models of computation that involve highly efficient inference applications [24]. An example of such a computing model is the *sparse distributed memory (SDM)*, which can be trained to remember sparse data vectors and retrieve them when presented with noisy or incomplete versions of the vectors [43]. However, SDM architectures are challenging due to the often conflicting design goals of achieving both high throughput and energy efficiency.

To enable sparse distributed memory, *compute memory* has been proposed as a viable implementation architecture [46]. Compute memory [44], [45] is an in-memory processing architecture that implements both memory and processing in a single architecture in order to completely eliminate the processor-memory interface. The compute memory architecture implements inference algorithms in the periphery of the memory array, and does not modify the core bit-cell array, thus maintaining the storage density. The compute memory is able to implement operations, such as the sum of absolute differences, signed multiplication, etc. The SDM implementation, using compute memory, has been shown to achieve both high throughput and energy efficiency for data-rich applications, such as pattern recognition [46].

However, most current compute memory implementations are application-specific. Ahn et al. [5] proposed a processor in memory application that uses specialized instructions, called PIM-enabled instructions, to invoke in-memory computations. The goal of the proposed work was to allow processing in memory operations to be compatible with existing systems and applications, without the need to specifically design the processor in memory for specific applications. Much work exists to extend compute memory architectures to multi-application use-cases, with minimal overheads.

Another architecture, similar to the compute memory, with high potential for IoT devices is the *compute sensor* [105], which offers in-sensor processing. The compute sensor takes advantage of machine learning algorithms' inherent adaptability to noise, and embeds information processing functionality for these algorithms into the sensor substrates. The compute sensor eliminates the sensor-processor interface, wherein sensed data is typically transmitted to a processor for data visualization, as is the case in traditional sensors. The compute sensor significantly reduces energy consumption and latency of feature extraction and classification functions, without sacrificing accuracy [105].

F. Secure Microarchitectures

Security applications are some of the most important applications that will be executed on IoT microprocessors. The IoT's characteristics—pervasiveness, interdependence, connectedness, mobility—makes IoT devices inherently vulnerable to increasing number of attacks. IoT devices are susceptible to physical, side channel, cryptoanalysis, software, and network attacks. Security can no longer be an afterthought in microprocessor design; it is imperative that microprocessors are designed to be inherently secure. However, security in IoT devices is especially challenging due to the typically stringent resource constraints of these devices.

Most IoT devices will have no hardware support for virtualization or enhanced security features, such as trusted execution [52]. The processing capabilities of IoT devices' microprocessors may be exceeded by the resource requirements of security processes and algorithms. As a result, security designers often need to trade off other vital optimization goals, such as energy, performance, or cost [51]. Apart from the resource constraints of IoT device microprocessors, these devices also generate massive amounts of data, some of which may contain private or sensitive information [79].

Much previous research has proposed hardware security techniques for embedded systems [12], [47], [49], [84]. While most of these techniques can also be employed in IoT devices, one critical requirement for IoT devices is the need to have runtime configurable hardware security policies that can adapt to varying security requirements [23]. This requirement is motivated by the resource constraints of IoT devices and the fact that the sensitivity of various computations vary depending on the executing applications. Thus, secure microarchitectures for the IoT must have the capability to be adjusted to satisfy

specific applications' or tasks' needs, while incurring minimal overheads.

There are currently very few techniques that have been proposed for configurable hardware security in microprocessor architectures, especially for IoT devices [23]. The main advantage of security at the level of the microarchitecture is that microarchitectures can typically be easily augmented for runtime configurability. In addition, configurable microarchitectures can be used to achieve multiple optimization goals. Thus, ensuring hardware security, using configurable microarchitectures, need not be at the expense of other optimization goals, such as energy consumption. For example, configurable caches can be used as a moving target defense [107] against side-channel attacks in caches, while also maintaining the other optimization benefits of configurability (e.g., energy and performance) [25].

VII. Conclusions

The Internet of Things (IoT) is expected to transform life, business, and the global economy. The IoT's scale and rapid proliferation will generate massive amounts of data that will result in communication bandwidth bottlenecks, and latency and energy overheads. Edge computing significantly reduces these overheads by equipping IoT devices with right-provisioned microprocessors and algorithms that can perform computations on the edge nodes to interpret, visualize, and use data.

This paper presented an overview of microprocessor characteristics that will support the growth of the IoT, from an edge computing perspective, and optimizations that will enable those characteristics. The survey presented herein should provide researchers with a foundation for designing IoT microprocessors that are efficient, configurable, secure, future-proof, and extensible. We have also discussed some of the challenges with achieving the discussed optimizations, and presented some potential solutions for addressing the challenges. Since edge computing on the IoT is a growing area of research, this study provides a foundation for further research into application requirements and microprocessor optimizations that will support edge computing in next-generation IoT devices.

References

[1] J. Abella and A. González. On reducing register pressure and energy in multiple-banked register files. In *Computer Design, 2003. Proceedings. 21st International Conference on*, pages 14–20. IEEE, 2003.

[2] T. Adegbija, A. Gordon-Ross, and A. Munir. Phase distance mapping: a phase-based cache tuning methodology for embedded systems. *Design Automation for Embedded Systems*, 18(3-4):251–278, 2014.

[3] T. Adegbija, A. Gordon-Ross, and M. Rawlins. Analysis of cache tuner architectural layouts for multicore embedded systems. In *Performance Computing and Communications Conference (IPCCC), 2014 IEEE International*, pages 1–8. IEEE, 2014.

[4] 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.

[5] J. Ahn, S. Yoo, O. Mutlu, and K. Choi. Pim-enabled instructions: A low-overhead, locality-aware processing-in-memory architecture. In *Computer Architecture (ISCA), 2015 ACM/IEEE 42nd Annual International Symposium on*, pages 336–348. IEEE, 2015.

[6] J. B. Anderson and S. Mohan. *Source and channel coding: an algorithmic approach*, volume 150. Springer Science & Business Media, 2012.

[7] H. Arabnejad and J. G. Barbosa. List scheduling algorithm for heterogeneous systems by an optimistic cost table. *IEEE Transactions on Parallel and Distributed Systems*, 25(3):682–694, 2014.

[8] K. Asanovic, R. Bodik, B. C. Catanzaro, J. J. Gebis, P. Husbands, K. Keutzer, D. A. Patterson, W. L. Plishker, J. Shalf, S. W. Williams, et al. The landscape of parallel computing research: A view from berkeley. Technical report, Technical Report UCB/EECS-2006-183, EECS Department, University of California, Berkeley, 2006.

[9] K. Ashton. That 'internet of things' thing. *RFiD Journal*, 22(7):97–114, 2009.

[10] A. Asin and D. Gascon. 50 sensor applications for a smarter world. *Libelium Comunicaciones Distribuidas, Tech. Rep*, 2012.

[11] L. Atzori, A. Iera, and G. Morabito. The internet of things: A survey. *Computer networks*, 54(15):2787–2805, 2010.

[12] S. Babar, A. Stango, N. Prasad, J. Sen, and R. Prasad. Proposed embedded security framework for internet of things (iot). In *Wireless Communication, Vehicular Technology, Information Theory and Aerospace & Electronic Systems Technology (Wireless VITAE), 2011 2nd International Conference on*, pages 1–5. IEEE, 2011.

[13] J. Baliga, R. W. Ayre, K. Hinton, and R. S. Tucker. Green cloud computing: Balancing energy in processing, storage, and transport. *Proceedings of the IEEE*, 99(1):149–167, 2011.

[14] C. E. Bash, C. D. Patel, and R. K. Sharma. Dynamic thermal management of air cooled data centers. In *Thermal and Thermomechanical Phenomena in Electronics Systems, 2006. ITHERM'06. The Tenth Intersociety Conference on*, pages 8–pp. IEEE, 2006.

[15] N. Binkert, B. Beckmann, G. Black, S. K. Reinhardt, A. Saidi, A. Basu, J. Hestness, D. R. Hower, T. Krishna, S. Sardashti, et al. The gem5 simulator. *Computer Architecture News*, 40(2):1, 2012.

[16] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli. Fog computing and its role in the internet of things. In *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*, pages 13–16. ACM, 2012.

[17] 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*, pages 45–50. ACM, 2014.

[18] A. P. Chandrakasan, D. C. Daly, J. Kwong, and Y. K. Ramadass. Next generation micro-power systems. In *2008 IEEE Symposium on VLSI Circuits*, pages 2–5. IEEE, 2008.

[19] M. Chen, S. Gonzalez, A. Vasilakos, H. Cao, and V. C. Leung. Body area networks: A survey. *Mobile networks and applications*, 16(2):171–193, 2011.

[20] Y. Chen, S. Lu, H.-S. Kim, D. Blaauw, R. G. Dreslinski, and T. Mudge. A low power software-defined-radio baseband processor for the internet of things. In *High Performance Computer Architecture (HPCA), 2016 IEEE International Symposium on*, pages 40–51. IEEE, 2016.

[21] V. K. Chippa, S. Venkataramani, S. T. Chakradhar, K. Roy, and A. Raghunathan. Approximate computing: An integrated hardware approach. In *2013 Asilomar Conference on Signals, Systems and Computers*, pages 111–117. IEEE, 2013.

[22] J.-M. Choi, C.-M. Jung, and K.-S. Min. Pcram flip-flop circuits with sequential sleep-in control scheme and selective write latch. *JSTS: Journal of Semiconductor Technology and Science*, 13(1):58–64, 2013.

[23] J. Crenne, R. Vaslin, G. Gogniat, J.-P. Diguet, R. Tessier, and D. Unnikrishnan. Configurable memory security in embedded systems. *ACM Transactions on Embedded Computing Systems (TECS)*, 12(3):71, 2013.

[24] J. M. Cruz-Albrecht, M. W. Yung, and N. Srinivasa. Energy-efficient neuron, synapse and stdp integrated circuits. *IEEE transactions on biomedical circuits and systems*, 6(3):246, 2012.

[25] C. Dai and T. Adegbija. Exploiting configurability as a defense against cache side channel attacks. In *2017 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*. IEEE, 2017.

[26] K. Du, P. Varman, and K. Mohanram. High performance reliable variable latency carry select addition. In *2012 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 1257–1262. IEEE, 2012.

[27] A. Efthymiou and J. D. Garside. Adaptive pipeline structures for speculation control. In *Asynchronous Circuits and Systems, 2003. Proceedings. Ninth International Symposium on*, pages 46–55. IEEE, 2003.

[28] D. Folegnani and A. González. Energy-effective issue logic. In *ACM SIGARCH Computer Architecture News*, volume 29, pages 230–239. ACM, 2001.

[29] Gartner. http://www.gartner.com/newsroom/id/2684616, 2016. Accessed: April 2016.

[30] E. I. Gaura, J. Brusey, M. Allen, R. Wilkins, D. Goldsmith, and R. Rednic. Edge mining the internet of things. *Sensors Journal, IEEE*, 13(10):3816–3825, 2013.

[31] S. Gollakota, M. S. Reynolds, J. R. Smith, and D. J. Wetherall. The emergence of rf-powered computing. *Computer*, 47(1):32–39, 2014.

[32] V. S. Gopinath, J. Sprinkle, and R. Lysecky. Modeling of data adaptable reconfigurable embedded systems. In *Engineering of Computer Based Systems (ECBS), 2011 18th IEEE International Conference and Workshops on*, pages 276–283. IEEE, 2011.

[33] A. Gordon-Ross, F. Vahid, and N. D. Dutt. Fast configurable-cache tuning with a unified second-level cache. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 17(1):80–91, 2009.

[34] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami. Internet of things (iot): A vision, architectural elements, and future directions. *Future Generation Computer Systems*, 29(7):1645–1660, 2013.

[35] K. Gudan, S. Chemishkian, J. J. Hull, M. S. Reynolds, and S. Thomas. Feasibility of wireless sensors using ambient 2.4 ghz rf energy. In *Sensors, 2012 IEEE*, pages 1–4. IEEE, 2012.

[36] H. Hajimiri and P. Mishra. Intra-task dynamic cache reconfiguration. In *VLSI Design (VLSID), 2012 25th International Conference on*, pages 430–435. IEEE, 2012.

[37] D. Halperin, T. S. Heydt-Benjamin, K. Fu, T. Kohno, and W. H. Maisel. Security and privacy for implantable medical devices. *IEEE pervasive computing*, 7(1):30–39, 2008.

[38] J. Han and M. Orshansky. Approximate computing: An emerging paradigm for energy-efficient design. In *2013 18th IEEE European Test Symposium (ETS)*, pages 1–6. IEEE, 2013.

[39] V. Izosimov, P. Pop, P. Eles, and Z. Peng. Design optimization of time-and cost-constrained fault-tolerant distributed embedded systems. In *Proceedings of the conference on Design, Automation and Test in Europe-Volume 2*, pages 864–869. IEEE Computer Society, 2005.

[40] B. Jeff. Big. little system architecture from arm: saving power through heterogeneous multiprocessing and task context migration. In *Proceedings of the 49th Annual Design Automation Conference*, pages 1143–1146. ACM, 2012.

[41] T. Jiang, Q. Zhang, R. Hou, L. Chai, S. A. Mckee, Z. Jia, and N. Sun. Understanding the behavior of in-memory computing workloads. In *Workload Characterization (IISWC), 2014 IEEE International Symposium on*, pages 22–30. IEEE, 2014.

[42] A. B. Kahng and S. Kang. Accuracy-configurable adder for approximate arithmetic designs. In *Proceedings of the 49th Annual Design Automation Conference*, pages 820–825. ACM, 2012.

[43] P. Kanerva. *Sparse distributed memory*. MIT press, 1988.

[44] M. Kang, S. K. Gonugondla, M.-S. Keel, and N. R. Shanbhag. An energy-efficient memory-based high-throughput vlsi architecture for convolutional networks. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1037–1041. IEEE, 2015.

[45] M. Kang, M.-S. Keel, N. R. Shanbhag, S. Eilert, and K. Curewitz. An energy-efficient vlsi architecture for pattern recognition via deep embedding of computation in sram. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 8326–8330. IEEE, 2014.

[46] M. Kang and N. R. Shanbhag. In-memory computing architectures for sparse distributed memory. 2016.

[47] A. Kanuparthi, R. Karri, and S. Addepalli. Hardware and embedded security in the context of internet of things. In *Proceedings of the 2013 ACM workshop on Security, privacy & dependability for cyber vehicles*, pages 61–64. ACM, 2013.

[48] N. Karimian, P. A. Wortman, and F. Tehranipoor. Evolving authentication design considerations for the internet of biometric things (iobt). In *Hardware/Software Codesign and System Synthesis (CODES+ ISSS), 2016 International Conference on*, pages 1–10. IEEE, 2016.

[49] M. M. Kermani, M. Zhang, A. Raghunathan, and N. K. Jha. Emerging frontiers in embedded security. In *2013 26th International Conference on VLSI Design and 2013 12th International Conference on Embedded Systems*, pages 203–208. IEEE, 2013.

[50] G.-D. Kim, C. Yoon, S.-B. Kye, Y. Lee, J. Kang, Y. Yoo, and T.-K. Song. A single fpga-based portable ultrasound imaging system for point-of-care applications. *IEEE transactions on ultrasonics, ferroelectrics, and frequency control*, 59(7):1386–1394, 2012.

[51] P. Kocher, R. Lee, G. McGraw, A. Raghunathan, and S. Moderator-Ravi. Security as a new dimension in embedded system design. In *Proceedings of the 41st annual Design Automation Conference*, pages 753–760. ACM, 2004.

[52] P. Koeberl, S. Schulz, A.-R. Sadeghi, and V. Varadharajan. Trustlite: A security architecture for tiny embedded devices. In *Proceedings of the Ninth European Conference on Computer Systems*, page 10. ACM, 2014.

[53] Y. Kora, K. Yamaguchi, and H. Ando. Mlp-aware dynamic instruction window resizing for adaptively exploiting both ilp and mlp. In *Proceedings of the 46th Annual IEEE/ACM International Symposium on Microarchitecture*, pages 37–48. ACM, 2013.

[54] L. Kugler. Is good enough computing good enough? *Communications of the ACM*, 58(5):12–14, 2015.

[55] P. Kulkarni, P. Gupta, and M. Ercegovac. Trading accuracy for power with an underdesigned multiplier architecture. In *2011 24th Internatioal Conference on VLSI Design*, pages 346–351. IEEE, 2011.

[56] R. Kumar, K. Farkas, N. P. Jouppi, P. Ranganathan, and D. M. Tullsen. Processor power reduction via single-isa heterogeneous multi-core architectures. *Computer Architecture Letters*, 2(1):2–2, 2003.

[57] T. T.-O. Kwok and Y.-K. Kwok. Computation and energy efficient image processing in wireless sensor networks based on reconfigurable computing. In *Parallel Processing Workshops, 2006. ICPP 2006 Workshops. 2006 International Conference on*, pages 8–pp. IEEE, 2006.

[58] K. Y. Kyaw, W. L. Goh, and K. S. Yeo. Low-power high-speed multiplier for error-tolerant application. In *2010 IEEE International Conference of Electron Devices and Solid-State Circuits (EDSSC)*, 2010.

[59] H. Lee, Y. Lin, Y. Harel, M. Woh, S. Mahlke, T. Mudge, and K. Flautner. Software defined radio–a high performance embedded challenge. In *International Conference on High-Performance Embedded Architectures and Compilers*, pages 6–26. Springer, 2005.

[60] C. Li, W. Zhang, C.-B. Cho, and T. Li. Solarcore: Solar energy driven multi-core architecture power management. In *2011 IEEE 17th International Symposium on High Performance Computer Architecture*, pages 205–216. IEEE, 2011.

[61] X. Li, R. Lu, X. Liang, X. Shen, J. Chen, and X. Lin. Smart community: an internet of things application. *IEEE Communications Magazine*, 49(11):68–75, 2011.

[62] Y. Liu, Z. Li, H. Li, Y. Wang, X. Li, K. Ma, S. Li, M.-F. Chang, S. John, Y. Xie, et al. Ambient energy harvesting nonvolatile processors: from circuit to system. In *Proceedings of the 52nd Annual Design Automation Conference*, page 150. ACM, 2015.

[63] Y. Liu, Z. Wang, A. Lee, F. Su, C.-P. Lo, Y. Zhe, et al. A 65nm reram-enabled nonvolatile processor with 6x reduction in restore time and 4x higher clock frequency using adaptive data retention and self-write- termination nonvolatile logics. In *IEEE International Solid-state Circuits Conference (ISSCC)*, pages 84–86. IEEE, 2016.

[64] S.-L. Lu. Speeding up processing with approximation circuits. *Computer*, 37(3):67–73, 2004.

[65] K. Ma, Y. Zheng, S. Li, K. Swaminathan, X. Li, Y. Liu, J. Sampson, Y. Xie, and V. Narayanan. Architecture exploration for ambient energy harvesting nonvolatile processors. In *2015 IEEE 21st International Symposium on High Performance Computer Architecture (HPCA)*, pages 526–537. IEEE, 2015.

[66] McKinsey. Disruptive technologies: advances that will transform life, business, and the global economy. http://www.mckinsey.com, 2016. Accessed: January 2016.

[67] Medical. Medical image processing, analysis, and visualization. http://mipav.cit.nih.gov/, 2015. Accessed: January 2016.

[68] D. Miorandi, S. Sicari, F. De Pellegrini, and I. Chlamtac. Internet of things: Vision, applications and research challenges. *Ad Hoc Networks*, 10(7):1497–1516, 2012.

[69] A. Mort, L. Eadie, L. Regan, A. Macaden, D. Heaney, M.-M. Bouamrane, G. Rushworth, and P. Wilson. Combining transcranial ultrasound with intelligent communication methods to enhance the remote assessment and management of stroke patients: Framework for a technology demonstrator. *Health informatics journal*, 22(3):691–701, 2016.

[70] E. F. Nakamura, A. A. Loureiro, and A. C. Frery. Information fusion for wireless sensor networks: Methods, models, and classifications. *ACM Computing Surveys (CSUR)*, 39(3):9, 2007.

[71] O. Navarro, T. Leiding, and M. Hübner. Configurable cache tuning with a victim cache. In *Reconfigurable Communication-centric Systems-on-Chip (ReCoSoC), 2015 10th International Symposium on*, pages 1–6. IEEE, 2015.

[72] S. Onkaraiah, M. Reyboz, F. Clermidy, J.-M. Portal, M. Bocquet, C. Muller, C. Anghel, A. Amara, et al. Bipolar reram based nonvolatile flip-flops for low-power architectures. In *New Circuits and Systems Conference (NEWCAS), 2012 IEEE 10th International*, pages 417–420. IEEE, 2012.

[73] C. D. Patel, C. E. Bash, R. Sharma, M. Beitelmal, and R. Friedrich. Smart cooling of data centers. In *ASME 2003 International Electronic Packaging Technical Conference and Exhibition*, pages 129–137. American Society of Mechanical Engineers, 2003.

[74] D. A. Patterson, G. Gibson, and R. H. Katz. *A case for redundant arrays of inexpensive disks (RAID)*, volume 17. ACM, 1988.

[75] C. Perera, A. Zaslavsky, P. Christen, and D. Georgakopoulos. Context aware computing for the internet of things: A survey. *IEEE Communications Surveys & Tutorials*, 16(1):414–454, 2014.

[76] V. Raghunathan, S. Ganeriwal, M. Srivastava, and C. Schurgers. Energy efficient wireless packet scheduling and fair queuing. *ACM Transactions on Embedded Computing Systems (TECS)*, 3(1):3–23, 2004.

[77] V. Raghunathan, A. Kansal, J. Hsu, J. Friedman, and M. Srivastava. Design considerations for solar energy harvesting wireless embedded systems. In *Proceedings of the 4th international symposium on Information processing in sensor networks*, page 64. IEEE Press, 2005.

[78] M. C. Ramon. Intel galileo and intel galileo gen 2. In *Intel® Galileo and Intel® Galileo Gen 2*, pages 1–33. Springer, 2014.

[79] A.-R. Sadeghi, C. Wachsmann, and M. Waidner. Security and privacy challenges in industrial internet of things. In *Proceedings of the 52nd Annual Design Automation Conference*, page 54. ACM, 2015.

[80] G. K. Saha. Software based fault tolerance: a survey. *Ubiquity*, 2006(July):1, 2006.

[81] F. Samie, L. Bauer, and J. Henkel. An approximate compressor for wearable biomedical healthcare monitoring systems. In *Proceedings of the 10th International Conference on Hardware/Software Codesign and System Synthesis*, pages 133–142. IEEE Press, 2015.

[82] A. Sampson, J. Nelson, K. Strauss, and L. Ceze. Approximate storage in solid-state memories. *ACM Transactions on Computer Systems (TOCS)*, 32(3):9, 2014.

[83] E. Sánchez-Sinencio. Smart nodes of internet of things (iot): a hardware perspective view & implementation. In *Proceedings of the 24th edition of the great lakes symposium on VLSI*, pages 137–138. ACM, 2014.

[84] D. N. Serpanos and A. G. Voyiatzis. Security challenges in embedded systems. *ACM Transactions on Embedded Computing Systems (TECS)*, 12(1s):66, 2013.

[85] 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 *2014 ACM/IEEE 41st International Symposium on Computer Architecture (ISCA)*, pages 97–108. IEEE, 2014.

[86] D. Shin and S. K. Gupta. Approximate logic synthesis for error tolerant applications. In *Proceedings of the Conference on Design, Automation and Test in Europe*, pages 957–960. European Design and Automation Association, 2010.

[87] D. Shin and S. K. Gupta. A new circuit simplification method for error tolerant applications. In *2011 Design, Automation & Test in Europe*, pages 1–6. IEEE, 2011.

[88] A. K. Singh, M. Shafique, A. Kumar, and J. Henkel. Mapping on multi/many-core systems: survey of current and emerging trends. In *Proceedings of the 50th Annual Design Automation Conference*, page 1. ACM, 2013.

[89] S. P. Singh and R. Maini. Comparison of data encryption algorithms. *International Journal of Computer Science and Communication*, 2(1):125–127, 2011.

[90] G. Stitt, R. Lysecky, and F. Vahid. Dynamic hardware/software partitioning: a first approach. In *Proceedings of the 40th annual Design Automation Conference*, pages 250–255. ACM, 2003.

[91] H. Sundmaeker, P. Guillemin, P. Friess, and S. Woelfflé. *Vision and challenges for realising the Internet of Things*, volume 20. EUR-OP, 2010.

[92] K. Swaminathan, R. Mukundrajan, N. Soundararajan, and V. Narayanan. Towards resilient micro-architectures: Datapath reliability enhancement using stt-mram. In *2011 IEEE Computer Society Annual Symposium on VLSI*, pages 236–241. IEEE, 2011.

[93] S. Tilak, N. B. Abu-Ghazaleh, and W. Heinzelman. A taxonomy of wireless micro-sensor network models. *ACM SIGMOBILE Mobile Computing and Communications Review*, 6(2):28–36, 2002.

[94] M. Tory and T. Möller. Rethinking visualization: A high-level taxonomy. In *Information Visualization, 2004. INFOVIS 2004. IEEE Symposium on*, pages 151–158. IEEE, 2004.

[95] O. S. Unsal, I. Koren, and C. M. Krishna. Towards energy-aware software-based fault tolerance in real-time systems. In *Low Power Electronics and Design, 2002. ISLPED'02. Proceedings of the 2002 International Symposium on*, pages 124–129. IEEE, 2002.

[96] S. Venkataraman, A. Kumar, J. Schlachter, and C. Enz. Designing inexact systems efficiently using elimination heuristics. In *Proceedings of the 2015 Design, Automation & Test in Europe Conference & Exhibition*, pages 758–763. EDA Consortium, 2015.

[97] H. J. Visser, A. C. Reniers, and J. A. Theeuwes. Ambient rf energy scavenging: Gsm and wlan power density measurements. In *Microwave Conference, 2008. EuMC 2008. 38th European*, pages 721–724. IEEE, 2008.

[98] W. Wang and T. Dey. A survey on arm cortex a processors. *http:" www. cs. virginia. edu1˜ skadron*, 2011.

[99] Y. Wang, Y. Liu, S. Li, D. Zhang, B. Zhao, M.-F. Chiang, Y. Yan, B. Sai, and H. Yang. A 3us wake-up time nonvolatile processor based on ferroelectric flip-flops. In *ESSCIRC (ESSCIRC), 2012 Proceedings of the*, pages 149–152. IEEE, 2012.

[100] Z. Xiong, X. Wu, S. Cheng, and J. Hua. Lossy-to-lossless compression of medical volumetric data using three-dimensional integer wavelet transforms. *IEEE transactions on medical imaging*, 22(3):459–470, 2003.

[101] J. Yiu. *The Definitive Guide to ARM® Cortex®-M3 and Cortex®-M4 Processors*. Newnes, 2013.

[102] W.-k. Yu, S. Rajwade, S.-E. Wang, B. Lian, G. E. Suh, and E. Kan. A non-volatile microcontroller with integrated floating-gate transistors. In *2011 IEEE/IFIP 41st International Conference on Dependable Systems and Networks Workshops (DSN-W)*, pages 75–80. IEEE, 2011.

[103] M. Zaharia, M. Chowdhury, T. Das, A. Dave, J. Ma, M. McCauley, M. J. Franklin, S. Shenker, and I. Stoica. Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing. In *Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation*, pages 2–2. USENIX Association, 2012.

[104] C. Zhang, F. Vahid, and W. Najjar. A highly configurable cache architecture for embedded systems. In *Computer Architecture, 2003. Proceedings. 30th Annual International Symposium on*, pages 136–146. IEEE, 2003.

[105] S. Zhang, M. Kang, C. Sakr, and N. Shanbhag. Reducing the energy cost of inference via in-sensor information processing. *arXiv preprint arXiv:1607.00667*, 2016.

[106] L. Zhou and H.-C. Chao. Multimedia traffic security architecture for the internet of things. *Network, IEEE*, 25(3):35–40, 2011.

[107] R. Zhuang, S. A. DeLoach, and X. Ou. Towards a theory of moving target defense. In *Proceedings of the First ACM Workshop on Moving Target Defense*, pages 31–40. ACM, 2014.

[108] M. Zwerg, A. Baumann, R. Kuhn, M. Arnold, R. Nerlich, M. Herzog, R. Ledwa, C. Sichert, V. Rzehak, P. Thanigai, et al. An 82μa/mhz microcontroller with embedded feram for energy-harvesting applications. In *2011 IEEE International Solid-State Circuits Conference*, 2011.

Tosiron Adegbija received his M.S and Ph.D in Electrical and Computer Engineering from the University of Florida in 2011 and 2015, respectively and his B.Eng in Electrical Engineering from the University of Ilorin, Nigeria in 2005. He is currently an Assistant Professor of Electrical and Computer Engineering at the University of Arizona, USA. His research interests are in computer architecture, with emphasis on adaptable computing, low-power embedded systems design and optimization methodologies, and microprocessor optimizations for the Internet of Things (IoT). He received the best paper award at the Ph.D forum of IEEE Computer Society Annual Symposium on VLSI in 2014.

Anita Rogacs received her M.S. and Ph.D in Mechanical Engineering from Stanford University and her B.S. in Mechanical Engineering from San Jose State University. She is recipient of National Science Foundation and Sandia National Laboratories Research Fellowships. At HP Labs she built Life Sciences Labs, which is now home to research efforts in areas of plasmonics, Raman spectroscopy, microfluidics, analytical chemistry, molecular biology and data mining. She is also the HP Labs Principle Investigator of the CRADA collaboration with the Federal Food and Drug Administration, which aims to develop screening methods using Surface Enhanced Raman Spectroscopy (SERS) sensors.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCAD.2017.2717782, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems

Chandrakant Patel is currently Chief Engineer and Senior Fellow of HP Inc. Patel has led HP Labs in delivering innovations in chips, systems, data centers, storage, networking, print engines and software platforms. He is a pioneer in thermal and energy management in data centers, and in the application of information technology for available energy management at the scale of cities. Patel is an ASME and an IEEE Fellow, and has been granted 151 patents and published more than 150 papers. An advocate of a return to fundamentals, he has served as an adjunct faculty member in engineering at Chabot College, U.C. Berkeley Extension, San Jose State University and Santa Clara University. In 2014, Patel was elected to the Silicon Valley Engineering Hall of Fame.

Ann Gordon-Ross received the B.S. and Ph.D. degrees in computer science and engineering from the University of California, Riverside, USA, in 2000 and 2007, respectively. She is currently an Associate Professor of Electrical and Computer Engineering with the University of Florida, USA, and a member of the NSF Center for High Performance Reconfigurable Computing with the University of Florida. She is also a Faculty Advisor of the Women in Electrical and Computer Engineering and the Phi Sigma Rho National Society for Women in Engineering and Engineering Technology, and an active member of the Women in Engineering Proactive Network. Her research interests include embedded systems, computer architecture, low-power design, reconfigurable computing, dynamic optimizations, hardware design, real-time systems, and multicore platforms. She was a recipient of the CAREER Award from the National Science Foundation in 2010, best paper awards at the Great Lakes Symposium on VLSI in 2010 and the IARIA International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies in 2010, and the Best Ph.D. Poster at IEEE Computer Society Annual Symposium on VLSI in 2014. She is very active in promoting diversity in STEM fields, and has been a Guest Speaker at several international workshops/conferences on this topic, organizes workshops, and participates in local outreach programs at local K-12 schools.