Ultra-Low Power and Dependability for IoT Devices

(Invited Paper for *IoT Technologies*)

Jörg Henkel, Santiago Pagani, Hussam Amrouch, Lars Bauer, and Farzad Samie
Chair for Embedded Systems (CES), Karlsruhe Institute of Technology (KIT), Germany
{henkel, pagani, amrouch, lars.bauer, samie}@kit.edu

***Abstract*—Recent advances in technologies have allowed the design of small-size low-power and low-cost devices that can be connected to the Internet, enabling the emerging paradigm of Internet-of-things (IoT). IoT covers an ever-increasing range of applications, e.g., health-care monitoring, smart homes and buildings, etc. In this invited paper, we discuss and summarize the IoT paradigm with a special focus on energy consumption and methodologies for its minimization. Furthermore, we also discuss about reliability in the context of IoT devices. In all, this paper attempts to be a starting point for readers interested in developing energy-efficient IoT devices.**

I. Introduction

Recent advances in technologies of sensors, wireless communication, and embedded processors have enabled the design of low-cost small-size and low-power devices that can be networked or connected to the Internet. These are the key components of the emerging paradigm of Internet-of-things (IoT) [1, 2]. IoT is covering an ever-increasing range of applications, such as health-care monitoring, smart city, smart home, smart building, smart industry, etc.

The IoT is a multidisciplinary paradigm in which many of the objects and *things* that surround us will be networked and connected to the Internet in order to provide new and better services [1, 3]. Recent and ongoing advances in the technologies such as ultra-low power processors, wireless communication, embedded sensors and actuators, Radio Frequency IDentification (RFID), mobile phones, and fog/cloud computing has enabled the emergence of IoT [4]. Although not all those technologies are needed for each and every IoT application, they all enable and facilitate the proliferation of IoT by providing an essential prerequisite [5, 6]. While RFID enables low-cost object identification, and while ultra-low power system-on-chips (SoC) enable portable battery-powered embedded devices, cloud computing and fog computing can be used to offload computations to the local or global servers, providing additional resources for handling large-scale data or performing more complex operations [7, 8]. The IoT system consists of different computation and storage layers, as shown in Figure 1.

Connectivity (wired or wireless) is what distinguishes embedded IoT devices from conventional embedded devices. In a broader sense and vision, IoT is a global infrastructure of *heterogeneous*, *networked* embedded objects and devices [9]. Communication ability, and in particular the Internet connectivity, allows devices and smart objects (also known as machines) to communicate and interact with (i) other machines and devices, or (ii) humans [5, 10, 11].

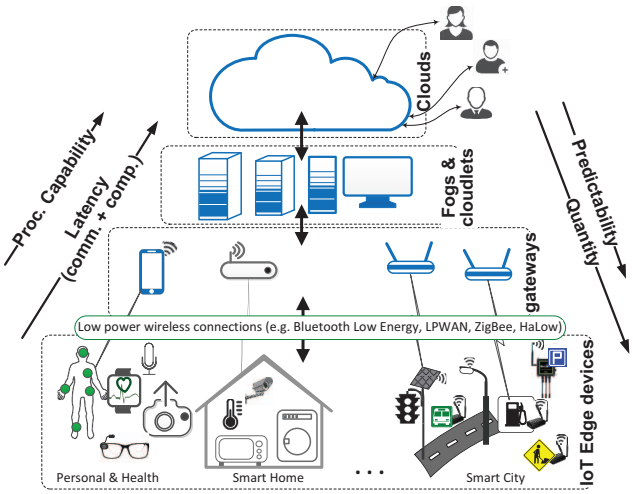

Fig. 1: Computation layers in IoT systems and their properties (figure from [1]).

II. Energy Efficiency on IoT

Every core doing some computation, every Analog-to-Digital Converter (ADC) acquiring data, every wireless communication, consumes power (with unit: Watt [W]), and this power consumption changes through time, e.g., a core executing a certain application will consume different amounts of power at different points in time. The amount of power consumed by the different parts of the IoT device at a given time point depends on several parameters, e.g., the technology scaling node, the architecture of the core, the power mode of the different elements and peripherals (e.g., active, idle, sleep, off, etc.), the voltage and frequency settings, the temperature on the core, the instructions being executed, etc. Contrarily, energy is the integration of power through time (with units: Joule [J] or Watt second [W s]). Namely, when the power consumption of the IoT device is plotted with respect to time, the energy consumed between two time points is equal to the area below the power curve between the two time points. Therefore, energy is associated to a time window, e.g., the intervals between the periodic updates of sensor data.

Many IoT devices are battery-operated or rely on energy harvesters with limited energy sources due to either their portability requirements, or the low cost of installation and maintenance. Therefore, ultra-low power consumption and efficient energy management are critical design objectives both in hardware design and software application, aiming at reducing the overall energy consumption while satisfying the Quality-of-Service (QoS) constraints. The functionality on an

978-3-9815370-8-6/17/$31.00 ©2017 IEEE

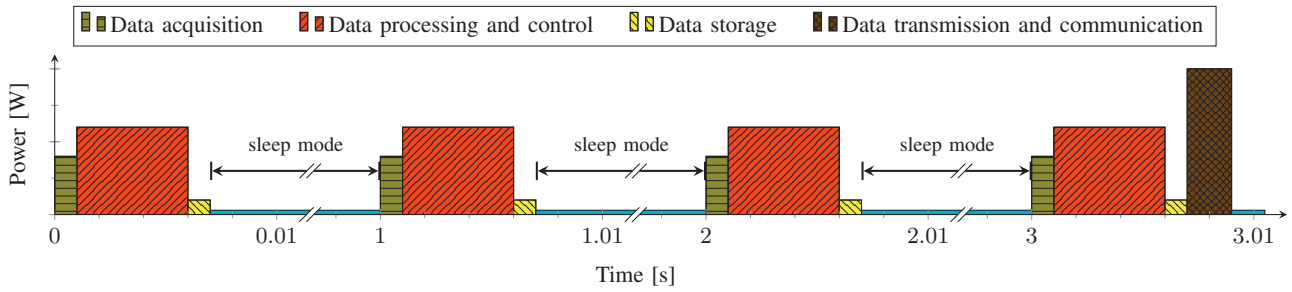

Fig. 2: Abstract example of the operation of an IoT device with respect to time, including the different stages of operation, their normal sequence, and some possible power consumption values. In this example, the sensor data is updated with a period of 1 s.

IoT device mainly includes the following four stages:

- Data acquisition (sensing the physical phenomena).
- Data processing and control.
- Data storage.
- Data transmission and communication.

Ultra-low power and energy-efficient design have to be considered in all these stages, requiring low power and compressed-sensing, low-power microcontrollers, emerging technologies for low power memories, and novel low-power wireless technologies [1]. For example, Figure 2 depicts an abstract example of the operation of an IoT device with respect to time, including these four stages, their normal sequence, and some possible power consumption values. In Figure 2, we can observe that the data to transmit can be stored in memory for a few periods, and only then transmitted. Naturally, we could have the case that data is immediately transmitted after processing it. The decision whether to use one approach or other will depend on the memory size and energy consumption for buffering the data, and its relationship with the energy consumption for waking-up the wireless transmitter and putting it back to sleep (which has non-negligible overheads).

Besides energy-efficient hardware, software optimization for low power consumption also has a great potential. In fact, studies showed and concluded that large energy savings can be achieved when the energy efficiency is addressed in software design [12, 13].

The input data rate in IoT applications is usually relatively low. For instance in the health-care monitoring domain, the sampling rate of bio-signals is only a few kilohertz or even less. In other application domains, such as smart homes or smart buildings, the input data rate might be much less (i.e., a few hundred hertz). Therefore, in order to preserve energy, most IoT devices operate on a very low duty cycle, i.e., remain in sleep mode for most of the time (as already shown in Figure 2). The low duty-cycle in IoT devices calls for the following two approaches, one in hardware design and the other in software design:

- **Hardware design:** Given that the IoT device spends most of its time in the sleep mode, its power consumption in the sleep mode needs to be reduced significantly. Emerging technologies with low leakage power, including e.g. non-volatile processors, Ferroelectric Random Access Memory (FRAM), and Fully-Depleted Silicon-On-Insulator (FD-SOI), are promising solutions for VLSI design in IoT. In addition, leakage power can be further reduced by choosing the process technology with high threshold voltage.
- **Software design:** The IoT application, control functions, and communication tasks need to be optimized for fast execution time in order to reduce the time spent in active mode. One approach is to run the software at the highest frequency (at the given operating voltage). Scheduling of different tasks (main application, communication, etc.) must aim at further reducing the duty cycle (i.e., the active mode). Further details about this are discussed in Section III.

III. Energy Efficiency Through Dynamic Voltage and Frequency Scaling (DVFS) for IoT

In regards to the voltage and frequency settings, cores can be provided with Dynamic Voltage and Frequency Scaling (DVFS) capabilities. For a core to stably support a specific frequency, its supply voltage has to be adjusted above a minimum value. This minimum voltage value is frequency dependent, and higher frequencies require higher minimum voltages. Correspondingly, for a given voltage setting, a core should be executed below a maximum frequency. Specifically, as shown in [14], the relationship between the supply voltage of a core and the maximum frequency for stable execution can be modeled as shown in Eq. (1),

$$f_{\text{stable}} = k \cdot \frac{(V_{\text{dd}} - V_{\text{th}})^2}{V_{\text{dd}}} \tag{1}$$

where V_{dd} is the supply voltage, f_{stable} is the maximum stable frequency for this V_{dd} value, V_{th} is the threshold voltage for the given technology, and k is an architecture-dependent fitting factor. For a given V_{dd}, running at frequencies lower than f_{stable} is stable, but energy inefficient. For example, Figure 3 uses Eq. (1) to model the stable voltage and frequency relationship of a 28 nm x86-64 processor developed in [15].

As detailed in [16], the power consumption of a CMOS core can be modeled as shown in Eq. (2),

$$P(V_{\text{dd}}, f, T, t) = u(t) \cdot C_{\text{eff}} \cdot {V_{\text{dd}}}^2 \cdot f + V_{\text{dd}} \cdot I_{\text{leak}}(V_{\text{dd}}, T) + P_{\text{ind}} \tag{2}$$

where $u(t)$ represents the instantaneous activity factor of the core at time t, C_{eff} represents the effective switching capacitance of the core, V_{dd} represents the supply voltage, f represents the execution frequency of the core, I_{leak} represents

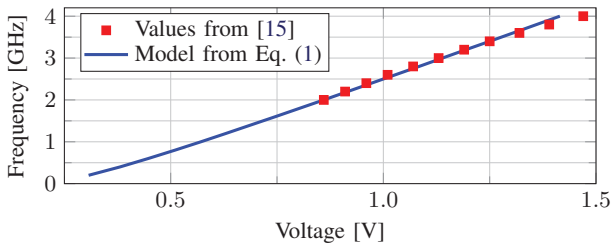

Fig. 3: Supply voltage vs. maximum stable frequency modeled according to Eq. (1) for the experimental results of the 28 nm x86-64 processor developed in [15].

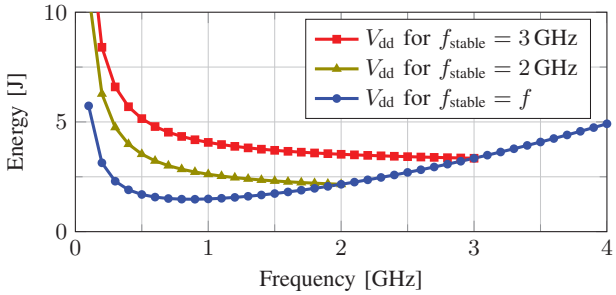

Fig. 4: Energy consumption examples for a core executing $\Delta c = 10^9$ compute cycles according to the energy model from Eq. (4), considering that $\overline{u}(\Delta t) \cdot C_{\text{eff}} = 2.3 \frac{\text{W}}{\text{V}^2 \cdot \text{GHz}}$, that we approximate $I_{\text{leak}}(V_{\text{dd}}, T) = 0.6\,\text{A}$, and $P_{\text{ind}} = 0.4\,\text{W}$. The values for the model from Eq. (4) are for a 22 nm *out-of-order* Alpha 21264 core, based on simulations conducted with gem5 [17] and McPAT [18] for an *x264* application from the PARSEC benchmark suite [19] running a single thread.

the leakage current (which depends on the supply voltage and the temperature of the core T, such that high temperatures cause high leakage currents), and P_{ind} represents an independent power consumption which can be attributed to maintaining the core in execution mode (i.e., a constant offset which corresponds to the voltage and frequency independent part of the power consumption). In Eq. (2), $u(t) \cdot C_{\text{eff}} \cdot {V_{\text{dd}}}^2 \cdot f$ represents the dynamic power consumption on the core (mainly generated by switching activities), while $V_{\text{dd}} \cdot I_{\text{leak}}(V_{\text{dd}}, T)$ represents the leakage power consumption (mainly generated by leakage currents). In other words, there is a convex and increasing relationship between the supply voltage of a core and its power consumption, and a linear relationship between the execution frequency of the core and its power consumption. Furthermore, the average power consumed during time window Δt, can be expressed as shown in Eq. (3),

$$\overline{P}(V_{\text{dd}}, f, T, \Delta t) = \overline{u}(\Delta t) \cdot C_{\text{eff}} \cdot {V_{\text{dd}}}^2 \cdot f + V_{\text{dd}} \cdot I_{\text{leak}}(V_{\text{dd}}, T) + P_{\text{ind}} \quad (3)$$

where $\overline{u}(\Delta t)$ represents the average activity factor of the core during time window Δt.

With regards to energy consumption, if during time window Δt a core executes Δc compute cycles running at frequency f, then the energy consumed during this time window can be computed as $\overline{P}(V_{\text{dd}}, f, T, \Delta t) \cdot \frac{\Delta c}{f}$, which according to Eq. (3) is expressed as shown in Eq. (4).

$$E(V_{\text{dd}}, f, T, \Delta t) = \overline{u}(\Delta t) \cdot C_{\text{eff}} \cdot {V_{\text{dd}}}^2 \cdot \Delta c + [V_{\text{dd}} \cdot I_{\text{leak}}(V_{\text{dd}}, T) + P_{\text{ind}}] \frac{\Delta c}{f} \quad (4)$$

According to Eq. (4), Figure 4 presents energy consumption examples for executing $\Delta c = 10^9$ compute cycles, considering three different voltage configurations. In the first and second case, the value of V_{dd} is kept constant at 1.14 V and 0.86 V, respectively, which according to Eq. (1) and Figure 3, is the minimum value required to execute at 3 GHz and 2 GHz for this type of core, respectively. In the third case, according to Eq. (1) and Figure 3, the value of V_{dd} is always set to the minimum value required to execute at the frequency of interest (i.e., the frequency in the x-axis of Figure 4).

There are two very important conclusions that can be inferred from Eq. (4) and Figure 4. The first conclusions is that, on IoT platforms where voltage scaling *is* possible (*not a common case*), then the energy-efficient option is to reduce the execution frequency as much as possible while setting the supply voltage precisely to the minimum value for stable execution. The reason for doing this is that, given that when we reduce the frequency we can also reduce the voltage, then the savings in dynamic energy consumption are more significant than the increments of the leakage and independent energy (due to the prolonged execution time). Nonetheless, there is a point below which further reductions to the frequency is no longer energy efficient, and this occurs when the savings in dynamic energy consumption (due to voltage reduction) become less significant than the increments of the leakage and independent energy. Contrarily, the second conclusion is that, on IoT platforms where voltage scaling *is not* possible (*the most common case*), then the energy-efficient option is to *race-to-idle*, which implies executing cores as fast as possible in order to reduce the execution time and also reduce the energy consumption. The justification now is that, when the voltage is fixed, according to Eq. (4), the dynamic energy consumption remains constant while the leakage and independent energy consumption is reduced by decreasing the execution time. Naturally, in either case, the selected frequency should be high enough that the performance requirements of the applications are always satisfied.

IV. Approximate Computing on IoT Devices

The emerging paradigm of approximate computing leverages inherent resilience of some applications and relaxes the requirement of exact equivalence between the specification and implementation in order to gain more efficiency. Most of IoT applications are interacting with the physical world with noisy input data [1]. Therefore, they are inherently dealing with approximation. For instance, the first level of approximation happens in the ADC which introduces a quantization error. Moreover, the ADC uses a voltage-reference in its conversion process. In the battery-powered IoT devices, the voltage reference may change as the battery voltage changes, which leads to an inaccurate input from the ADC. Although these applications tolerate some errors, the final output or Quality-of-Service (QoS) should be in a certain range (defined by the user or system designer).

The error tolerance property of the IoT applications can be exploited to trade the output quality for other computational effort (e.g., energy consumption, performance, memory usage, etc.). Some open challenges exist in the domain of approximate computing for IoT. The tolerable error can be exploited at different hardware components and different software parts:

- **Data acquisition**: The quality of input data, during the data acquisition, is determined by the resolution and sampling rate. For instance, an electrocardiogram (ECG) captures the electrical signal of the heart at 360 samples per second, with an 11-bit resolution in [20]. Compressed Sensing (CS) is a novel technique in which the signal can be reconstructed from much fewer samples than Nyquist theory, at the cost of accuracy loss. As long as the input data has the *sparseness* property (see [21] for details), a smaller number of samples can capture the required information, and therefore CS can reduce the volume of collected data without significant loss of information. Most IoT applications have the data sparsity property and can benefit from the CS paradigm. In health monitoring applications and wireless body sensor network, CS has been studied and investigated extensively [22–25]. When the IoT application tolerates error, reducing the quality of the input data is one way to take advantage of it in order to reduce the energy consumption or delay.
- **Data processing**: The approximation can also be done in the underlying hardware, by designing inexact hardware units, or software implementation of the data processing stage for specific arithmetic operations. For instance, some computation-intensive operations such as multiplication, Discrete Cosine Transform (DCT), Discrete Fourier Transform (DFT), etc., have shown great potentials for the approximation [26].
- **Data storage**: The tolerable error can be exploited at the memory unit (or storage) to reduce the size of the required memory, to reduce the number of memory accesses, or to reduce its energy consumption. One example of approximation in memory is presented in [27] for an ultra-low power biosignal processor.
- **Data transmission**: Given that the wireless communication is power-hugry, reducing the amount of transmitted data is a key concern. Novel data compression techniques can take advantage of the specific properties of IoT data and further reduce the power consumption of the device for transmission. Lossy compression techniques fall into the approximation category. In [28], we have presented an approximate compressor for the IoT-based biomedical health-care monitoring devices. By accepting a small amount of error in the amplitude of the sample of the biosignal (given by the user or system designer based on its application requirements), we are able to encode the data using shorter code-words which results in an improvement of the compression ratio. The proposed technique does not need any modification or extension in the hardware. Moreover, the computation overhead of this technique is negligible.

Hybrid schemes, where multiple stages exploit approximation, are also possible and promising. The main challenge here is to decide 1) at which stage, and 2) how much approximation should be applied in order to minimize the computational effort while meeting the QoS requirements.

V. Beyond the IoT Embedded Device

As shown in Figure 1, the ultra-low power and edge IoT devices are usually connected to the gateways. The main functionality of gateways is to enable seamless integration of low-power wireless networks of IoT devices such as Bluetooth Low Energy (BLE) or HaLow with other networks (e.g., cellular network, LAN, etc.). However, they also provide local data processing service at the edge of the network in the Edge Computing paradigm [29]. In the next layers, the Fog and Cloud computing provide massive storage and high performance computing for Big Data in IoT.

Considering the other computation layers in IoT, the task of data processing can be partitioned between different entities or exclusively assigned to a specific layer. In the emerging paradigm of edge computing, the IoT data is processed and acted upon close to the edge of network (i.e., embedded devices and gateways).

VI. Reliability Considerations for IoT

In a sense, voltage reduction is a double-edged sword. While, on the one hand, reducing voltage provides considerable power and energy savings (as motivated earlier in Section III), on the other hand, it reduces the reliability which leads to increasing the susceptibility to different kinds of failures. In general, failures can be divided into two main categories as follows:

- **Time-Independent Failures:** They are independent of the operation time of the circuit and mainly due to soft errors, which can be caused by either intrinsic voltage noise and/or extrinsic radiation (i.e., energetic particles).
- **Time-Dependent Failures:** They are dependent on the operation time of the circuit and mainly due to aging effects such as Bias Temperature Instability and Hot Carrier-induced Degradation (HICD), which are able to alter the electrical characteristics of transistors over time, e.g., such as the threshold voltage (V_{th}) [30].

In order to demonstrate the impact of voltage reduction on reliability, we study, as an example, a 6-T SRAM cell under different operating voltages, particularly, from $0.5\,\text{V}$ to $1.2\,\text{V}$. The Predictive Technology Model (PTM) [31], along with the compact modeling of MOSFETs from Berkeley (BSIM [32]) were jointly employed to perform the required SPICE simulations at $45\,\text{nm}$ technology node.

Resiliency against noise: The voltage noise, which inherently exists in any chip, can jeopardize the data integrity of SRAM cells. The resiliency of an SRAM against noise mainly depends on its operating voltage. To quantify the SRAM resiliency, the Static Noise Margin (SNM) is typically used, which expresses the resiliency of SRAM against the effects of parasitic-coupling (e.g., bit-line coupling) and intrinsic noise sources like thermal noise. In practice, larger SNM results in higher resiliency against noise and vice versa. In Figure 5, we show how voltage reduction considerably reduces the SNM which, in turn, can lead to a higher probability of noised-induced failures. For instance, reducing the voltage from $1.2\,\text{V}$ to $0.9\,\text{V}$ and $0.5\,\text{V}$ leads to reducing SNM by 20% and 60%, respectively.

Resiliency against radiation: When a particle strikes an SRAM cell, the deposited charge due to its kinetic energy can

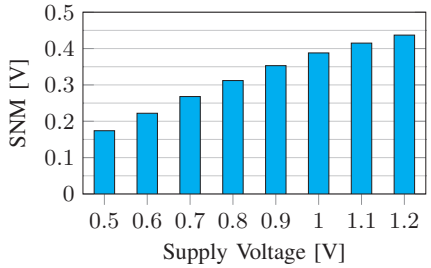

Fig. 5: Impact of V_{dd} on the resiliency of SRAMs against noise. Note that smaller V_{dd} results in smaller SNM and hence *less* reliability.

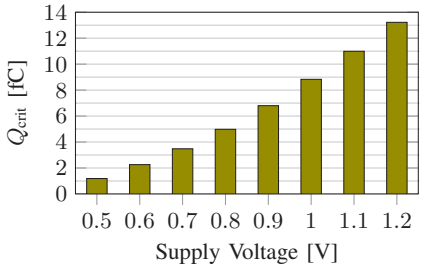

Fig. 6: Impact of V_{dd} on the resiliency of SRAMs against radiation. Note that smaller V_{dd} results in smaller Q_{crit} and hence *less* reliability.

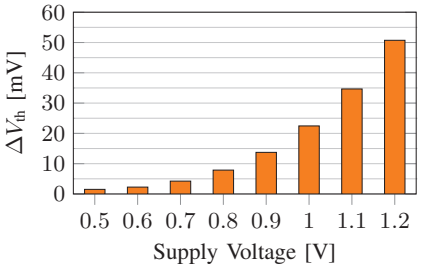

Fig. 7: Impact of V_{dd} on the resiliency of SRAMs against aging. Note that smaller V_{dd} results in smaller ΔV_{th} and hence *better* reliability.

corrupt the stored value if the resulting charge is higher than the critical charge (Q_{crit}) of the SRAM. Similar to the SNM that represents the SRAM resiliency against *intrinsic* noise, Q_{crit} represents the SRAM's resiliency against *extrinsic* noise (i.e., particles-induced charges). In Figure 6, we show how voltage reduction also considerably reduces the resiliency of SRAMs against radiation. As it can be noticed, lowering the voltage, e.g., from 1.2 V to 0.9 V and 0.5 V, can reduce Q_{crit} by 50% and 90%, respectively. In turn, smaller Q_{crit} results in higher probability of failure as even weak particles existing at the sea level (like neutrons) might become sufficient to cause soft errors.

Resiliency against aging: When the electric fields are applied during the operation of MOSFET transistors, different kinds of defects due to aging mechanisms can be generated. Such defects can accumulate over time, leading to an increase in the delay of a transistor and hence slowing down its speed. In fact, the amount of generated defects is primarily determined by the strength of the applied electric fields which are driven by the operating voltage. Therefore, when the voltage is reduced, the impact of aging mechanisms become less significant. In Figure 7, we demonstrate how lowering the voltage results in a smaller aging-induced threshold voltage increase in PMOS transistors. As can be seen, when the voltage reaches 0.5 V, aging can merely increase V_{th} by 1.5 mV. The results in Figure 7 have been estimated using a physics-based aging model [33], which was developed for the purpose of studying aging in the scope of voltage scaling.

In summary: Unlike aging effects, which become smaller at lower voltages, failures due to noise and radiation significantly become higher. In general, IoT devices can profit from voltage scaling with respect to power and energy, but at the same time, they can suffer from a significant reliability degradation. *Hence, when selecting the operating voltage of processors, designers need to find good trade-offs between power/energy saving and reliability.*

VII. Conclusions

Given that many IoT devices are battery-operated or rely on energy harvesters with limited energy sources, ultra-low power consumption and efficient energy management are critical design objectives both in hardware design and software application. A main research challenge is therefore to reduce the overall energy consumption of the IoT device while satisfying the Quality-of-Service (QoS) or performance constraints. In this paper, we have summarized the generalities about energy minimization for IoT devices, and we have discussed several techniques that can be employed for such a purpose. Moreover, we have also discussed several reliability issues which should nonetheless not be ignored. In all, this paper is a good starting point for readers interested in developing energy-efficient IoT devices.

Acknowledgements

This work was partly supported by the German Research Foundation (DFG) as part of the Transregional Collaborative Research Centre *Invasive Computing* [SFB/TR 89] http://invasic.de, and the priority program *Dependable Embedded Systems* [SPP 1500] http://spp1500.itec.kit.edu.

References

[1] F. Samie, L. Bauer, and J. Henkel, "IoT Technologies for Embedded Computing: A Survey", in *CODES+ISSS*, 2016.
[2] L. Atzori, A. Iera, and G. Morabito, "The Internet of Things: A survey", *Computer networks*, vol. 54, no. 15, 2010.
[3] D. Bandyopadhyay and J. Sen, "Internet of things: Applications and challenges in technology and standardization", *Wireless Personal Communications*, vol. 58, no. 1, pp. 49–69, 2011.
[4] B. Guo, D. Zhang, Z. Yu, Y. Liang, Z. Wang, and X. Zhou, "From the Internet of Things to embedded intelligence", *World Wide Web*, vol. 16, no. 4, 2013.
[5] D. Miorandi, S. Sicari, F. De Pellegrini, and I. Chlamtac, "Internet of things: Vision, applications and research challenges", *Ad Hoc Networks*, vol. 10, no. 7, pp. 1497–1516, 2012.
[6] I. Lee and K. Lee, "The Internet of Things (IoT): Applications, investments, and challenges for enterprises", *Business Horizons*, 2015.
[7] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the Internet of Things", in *MCC workshop on Mobile cloud computing*, 2012, pp. 13–16.
[8] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of Things (IoT): A vision, architectural elements, and future directions", *Future Generation Computer Systems*, vol. 29, no. 7, 2013.
[9] A. Gluhak, S. Krco, M. Nati, D. Pfisterer, N. Mitton, and T. Razafindralambo, "A survey on facilities for experimental internet of things research", *IEEE Communications Magazine*, vol. 49, no. 11, pp. 58–67, 2011.
[10] G. Wu, S. Talwar, K. Johnsson, N. Himayat, and K. D. Johnson, "M2M: From mobile to embedded internet", *IEEE Communications Magazine*, vol. 49, no. 4, pp. 36–43, 2011.
[11] L. Da Xu, W. He, and S. Li, "Internet of things in industries: A survey", *IEEE Transactions on Industrial Informatics*, vol. 10, no. 4, pp. 2233–2243, 2014.

[12] V. Tiwari, S. Malik, A. Wolfe, and M. T.-C. Lee, "Instruction level power analysis and optimization of software", in *Technologies for wireless computing*, 1996, pp. 139–154.
[13] A. Chatzigeorgiou and G. Stephanides, "Energy issues in software design of embedded systems", in *International Conference on Applied Informatics*, 2002.
[14] N. Pinckney, K. Sewell, R. G. Dreslinski, D. Fick, T. Mudge, D. Sylvester, and D. Blaauw, "Assessing the performance limits of parallelized near-threshold computing", in *the 49th Design Automation Conference (DAC)*, 2012, pp. 1147–1152.
[15] A. Grenat, S. Pant, R. Rachala, and S. Naffziger, "5.6 adaptive clocking system for improved power efficiency in a 28nm x86-64 microprocessor", in *IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*, 2014, pp. 106–107.
[16] J. Henkel, H. Khdr, S. Pagani, and M. Shafique, "New trends in dark silicon", in *Proceedings of the 52nd ACM/EDAC/IEEE Design Automation Conference (DAC)*, June 2015, pp. 119:1–119:6.
[17] N. Binkert, B. Beckmann, G. Black, S. K. Reinhardt, A. Saidi, A. Basu, J. Hestness, D. R. Hower, T. Krishna, S. Sardashti, R. Sen, K. Sewell, M. Shoaib, N. Vaish, M. D. Hill, and D. A. Wood, "The gem5 simulator", *ACM SIGARCH Computer Architecture News*, vol. 39, no. 2, pp. 1–7, May 2011.
[18] S. Li, J.-H. Ahn, R. Strong, J. Brockman, D. Tullsen, and N. Jouppi, "McPAT: An integrated power, area, and timing modeling framework for multicore and manycore architectures", in *Proceedings of the 42nd IEEE/ACM International Symposium on Microarchitecture (MICRO)*, 2009, pp. 469–480.
[19] C. Bienia, S. Kumar, J. P. Singh, and K. Li, "The PARSEC benchmark suite: Characterization and architectural implications", in *Proceedings of the 17th International Conference on Parallel Architectures and Compilation Techniques (PACT)*, 2008, pp. 72–81.
[20] T. N. Gia, M. J. A.-M. Rahmani, T. Westerlund, P. Liljeberg, and H. Tenhunen, "Fog computing in healthcare Internet-of-Things: A case study on ECG feature extraction", in *Int'l Conf. on Computer and Information Technology (CIT)*, 2015, pp. 356–363.
[21] Z. Zhang, Y. Xu, J. Yang, X. Li, and D. Zhang, "A survey of sparse representation: algorithms and applications", *IEEE Access*, vol. 3, pp. 490–530, 2015.
[22] S. Li, L. Da Xu, and X. Wang, "Compressed sensing signal and data acquisition in wireless sensor networks and Internet of Things", *IEEE Transactions on Industrial Informatics*, vol. 9, no. 4, pp. 2177–2186, 2013.
[23] D. Bortolotti, B. Milosevic, A. Bartolini, E. Farella, and L. Benini, "Quantifying the benefits of compressed sensing on a WBSN-based real-time biosignal monitor", in *Design, Automation & Test in Europe Conference (DATE)*, 2016, pp. 732–737.
[24] R. Ren, X. Chen, X. Hu, Y. Wang, and S. Xia, "Compressed sensing–based method for electrocardiogram monitoring on wireless body sensor using binary matrix", *International Journal of Wireless and Mobile Computing*, vol. 8, no. 2, pp. 114–121, 2015.
[25] J. Sheng, C. Yang, and M. C. Herbordt, "Hardware-efficient compressed sensing encoder designs for wbsns", in *High Performance Extreme Computing Conference (HPEC)*, 2015, pp. 1–7.
[26] J. Han and M. Orshansky, "Approximate computing: An emerging paradigm for energy-efficient design", in *European Test Symposium (ETS)*, 2013, pp. 1–6.
[27] 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 *ISLPED*, 2014, pp. 45–50.
[28] F. Samie, L. Bauer, and J. Henkel, "An approximate compressor for wearable biomedical healthcare monitoring systems", in *CODES+ISSS*, 2015, pp. 133–142.
[29] F. Samie, V. Tsoutsouras, S. Xydis, L. Bauer, D. Soudris, and J. Henkel, "Computation Offloading and Resource Allocation for Low-power IoT Edge Devices", in *IEEE World Forum on Internet of Things (WF-IoT)*, 2016.
[30] H. Amrouch, V. van Santen, T. Ebi, V. Wenzel, and J. Henkel, "Towards interdependencies of aging mechanisms", in *Computer-Aided Design (ICCAD), IEEE/ACM International Conference on*, Nov 2014, pp. 478–485.
[31] "Predictive Technology Model", http://ptm.asu.edu/.
[32] "Bsim4 manual", http://www-device.eecs.berkeley.edu/bsim/Files/BSIM4/BSIM480/BSIM480_Manual.pdf.
[33] V. M. van Santen, H. Amrouch, N. Parihar, S. Mahapatra, and J. Henkel, "Aging-aware voltage scaling", in *Proceedings of the Conference on Design, Automation & Test in Europe (DATE)*, 2016, pp. 576–581.