Energy-Quality Scalable Integrated Circuits and Systems: Continuing Energy Scaling in the Twilight of Moore's Law

Massimo Alioto, *Fellow, IEEE*, Vivek De, *Fellow, IEEE*, and Andrea Marongiu, *Member, IEEE*

***Abstract*— This paper aims to take stock of recent advances in the field of energy-quality (EQ) scalable circuits and systems, as promising direction to continue the historical exponential energy downscaling under diminished returns from technology and voltage scaling. EQ-scalable systems explicitly trade off energy and quality at different levels of abstraction and sub-systems, dealing with "quality" as an explicit design requirement, and reducing energy whenever the application, the task, or the dataset allow quality degradation (e.g., vision and machine learning). A general framework for EQ-scalable systems based on the concept of quality slack is presented along with scalable architectures. A taxonomy of techniques to trade off energy and quality, a VLSI perspective, and possible quality control strategies are then discussed. The state of the art is surveyed to put the advances in its different sub-fields into a unitary perspective, emphasizing the on-going and prospective trends. At the component level, the generality of the EQ-scaling concept is shown through several examples, ranging from logic to analog circuits, to memories, data converters, and accelerators. Interesting implications of the joint adoption of EQ scaling and machine learning are also discussed, suggesting that their synergy gives ample room for further energy and performance improvements. From a level of abstraction viewpoint, EQ scaling is discussed from the circuit level to architectures, the hardware–software interface, the programming language, the compiler level, and run-time adaptation. Several case studies are discussed to put EQ scaling in the context of real-world applications.**

***Index Terms*— Energy-efficient integrated circuits, VLSI, machine learning, signal processing, architectures, software.**

I. Introduction

ENERGY efficiency of integrated circuits is well known to be a crucial design constraint in a very wide range of applications spanning the entire spectrum of performance and form factor targets. In cloud and datacenter-scale computing (e.g., using servers and GPUs as components), energy efficiency directly determines the achievable performance under the power budget available from the grid and allowed by heat removal considerations [1]–[6], and has a major impact on datacenter profit margin [6] and [7]. In portable electronics (e.g., laptops, smartwatches), better energy efficiency enables longer battery lifetime or smaller form factor, in addition to enhanced functionality [8], [9]. At the edge of the Internet of Things (IoT, e.g. sensor nodes), the same observations apply to an even greater extent, in view of the even stronger importance of lifetime, form factor, and cost [10]–[13].

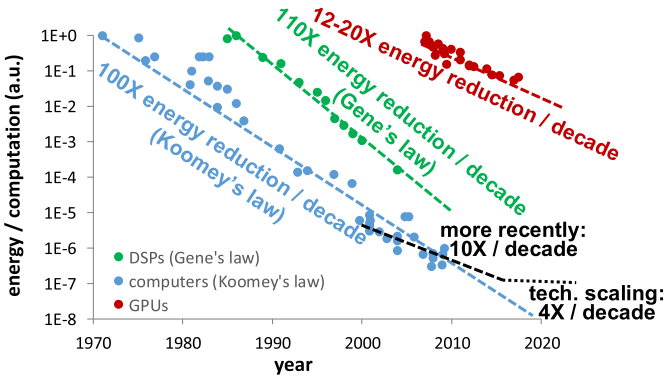


Fig. 1. Energy per computation trend vs year for general-purpose computers, DSPs and GPUs.

In the last few decades, electronic systems have witnessed an exponential energy reduction in all main sub-systems, ranging from processing, to interfaces at the sensor boundary (e.g., data converters) and the wireless boundary (i.e., radios). In regard to processing, technology scaling and Moore's law have driven energy efficiency improvements, accounting for more than 90% of the energy savings at each generation until year 2000s [6]. Fig. 1 shows that energy for general-purpose computers has decreased at the pace of 100× every decade, as per Koomey's law [6]. Very similar trend was observed in Digital Signal Processors (DSPs), according to Gene's law [14]. In the last two decades, the energy trend has slowed down substantially to 10× per decade, and has been outpaced by the 12-2× improvement per decade of GPUs, although this is still much slower than Koomey's law. Fig. 1 also shows that pure technology scaling will bring to up to 4× energy reduction in the decade ahead, given the small number of

Manuscript received November 1, 2018; accepted November 12, 2018. Date of publication November 15, 2018; date of current version December 11, 2018. This work was supported in part by the Singapore Ministry of Education under the Grant MOE2014-T2-1-161 and Grant MOE2014-T2-2-158 and in part by the NUS Hybrid-Integrated Flexible Electronic Systems Program. This paper was recommended by Editor E. Alarcon. *(Corresponding author: Massimo Alioto.)*

M. Alioto is with the Department of Electrical and Computer Engineering, National University of Singapore, Singapore 117583 (e-mail: massimo.alioto@nus.edu.sg).

V. De is with the Circuit Technology Research Group, Intel Labs, Hillsboro, OR 97124 USA (e-mail: vivek.de@intel.com).

A. Marongiu is with the Department of Computer Science and Engineering, University of Bologna, 40126 Bologna, Italy (e-mail: a.marongiu@unibo.it).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/JETCAS.2018.2881461

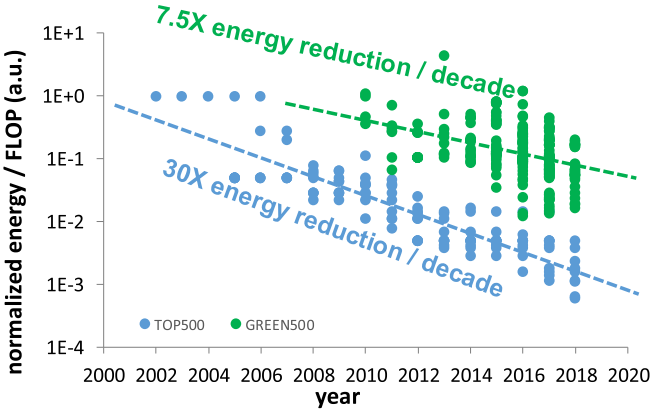


Fig. 2. Energy per floating point operation in the ten fastest supercomputers in the world vs year (TOP500 ranking [17]), and the most efficient supercomputers (GREEN500 ranking [16]).

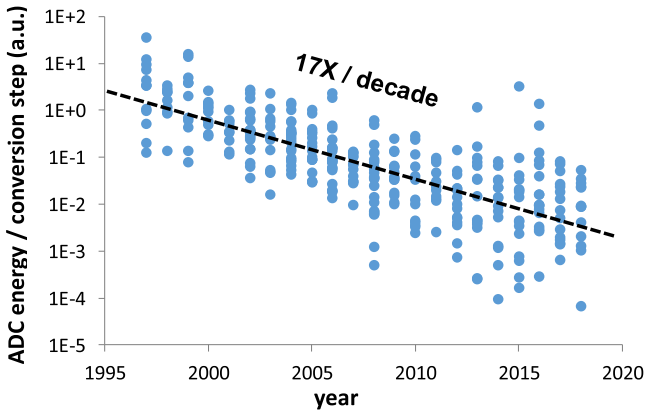


Fig. 3. Energy per conversion vs year in ADCs, as extracted from [19].

upcoming CMOS generations, and being the energy gain lower than 20% per generation even under the most optimistic and now outdated predictions [15].

The energy trend at the larger scale of supercomputing, Fig. 2 shows that the energy per floating point operation has decreased at the pace of 7.5× per decade in supercomputers designed for energy efficiency (i.e., belonging to the GREEN500 ranking [16]). For supercomputers designed with performance as first objective [17], a faster 30× energy saving per decade can be observed, as expectable from the initially larger energy gap between the two classes of computers. As a result, the energy of the ten fastest supercomputers is only 1.6-2.2× worse than the most energy-efficient one in the GREEN500 list. This confirms that energy efficiency is an essential prerequisite to achieve high performance, and is the main performance bottleneck. At the scale of mobile computing, 27× energy reduction per decade has been achieved to sustain performance improvements, under the power limit imposed by comfort standards and considerations on skin temperature of such devices [9], [18].

Exponential energy reductions have been observed also at the physical boundaries of the processing sub-system (e.g., sensing and communications). As an example, Fig. 3 shows a 17× per decade energy reduction trend for Analog-to-Digital Converters (ADCs), as extracted from [19].

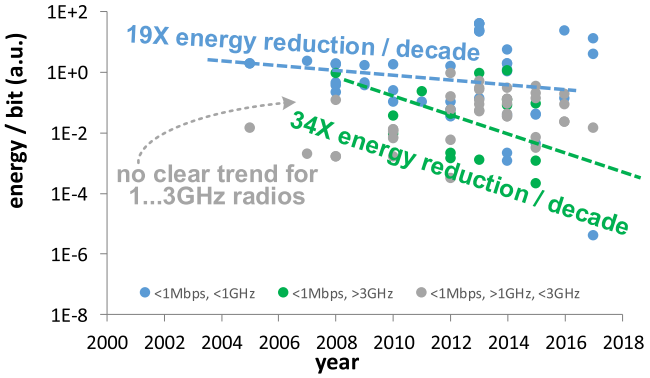


Fig. 4. Energy per bit in ultra-low power radios, as extracted from [20].

Ultra-low power radios with sub-GHz carrier have followed a similar trend (19× per decade) from Fig. 4, whereas the more recent interest and advances in millimeter-wave radios have driven the energy per bit by 34× per decade, as extracted from [20]. In both cases, processing has generally contributed to such scaling through the widespread adoption of digitally-assisted techniques [21], [22]. In other words, the slowed-down energy trend in the processing sub-system also limits the scaling of other sub-systems.

From the above considerations, technology scaling needs to be supplemented with significant and coordinated advances at all levels of abstraction [23]. Similar considerations hold for voltage scaling, which has been extensively leveraged in the last two decades. Indeed, 0.6-V operation is already available for commercial processors and standard cell libraries, which is rapidly approaching the transistor threshold and hence leaving very limited room for further voltage scaling [24]. Similarly, parallelism is no longer providing the energy gains that was used to deliver, especially in the vast portion of applications whose workload is not naturally parallelizable (e.g., portable electronics, sensor nodes). For example, the number of simultaneously active cores in smartphone platforms is well known to have remained essentially constant in the last few generations.

In summary, the historical 100×/decade energy downscaling trend in computation will not be solely sustained by relying on continued technology scaling, more aggressive voltage scaling or highly parallel architectures. This is a general fact in both compute-dominated systems, due to the recently slower energy per computation trend, and in systems with significant sensing and communications power cost, due to the inherently slower energy scaling. In the decade ahead, novel design dimensions will need to be introduced to trade off energy and lower it, whenever the related specifications can be relaxed. In this paper, and in the entire special issue that it belongs to, the energy-quality tradeoff is explored as very promising dimension at several levels of abstraction.

This paper is structured as follows. Section II introduces basic concepts on EQ-scalable systems and quality. A VLSI perspective is provided in Section III, which presents the concept of quality slack, a taxonomy of techniques enabling EQ scaling, and its role as generalization of conventional runtime error detection and correction schemes. Section IV provides

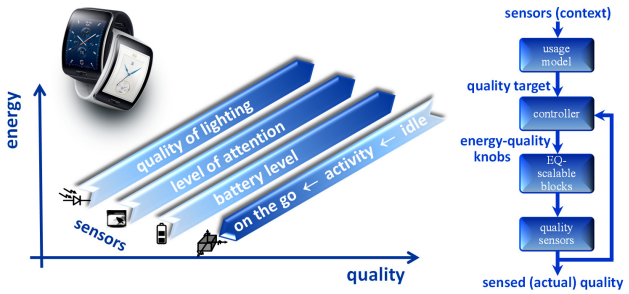


Fig. 5. Opportunities to scale down quality and energy in a wearable or portable device delivering video content, under poor quality of lighting, low level of user's attention, low battery level, high level of physical activity.

an application-centric perspective on EQ scaling, whereas a hardware design-centric view is introduced in Section V. Case studies of EQ-scalable circuits and architectures are reviewed in Section VI, whereas EQ scaling at the algorithm level and the interesting convergence with machine learning are discussed in Section VII. Software-level aspects on EQ scaling from language constructs to run-time layers are discussed in Section VIII. Conclusions are drawn in Section IX.

II. Energy-Quality Scaling and Basic Concepts

As motivating example, let us consider a smartphone or other portable or wearable device (e.g., smart glass) that is delivering a video stream. As in Fig. 5, the quality of the video content delivery can be degraded to a certain extent to save energy, whenever [2]

1) the quality of lighting is poor, as sensed by a light sensor
2) the user is not paying much attention, e.g. by performing contextual analysis of the imager and microphone input
3) the battery is drained out, and extending its lifetime becomes a priority
4) the user is on the go or is exercising, as sensed by motion sensors.

In other words, the built-in sensors can be used to derive the usage context in real time, and set the video quality target at the minimum that does not compromise the user experience, hence minimizing the energy per frame (e.g., by scaling aggressively the supply voltage and have occasional timing faults and have some pixels to be a bit off color). As in Fig. 5, the quality target is managed by a quality controller, which generates the appropriate energy-quality (EQ) knobs for the silicon sub-systems that are involved in the processing and the visualization of the content. In other words, the sub-systems are energy-quality scalable in the sense that the quality can be intentionally and individually degraded, and energy can be saved as a result. The actual quality degradation obtained through the coordinated adjustment of such knobs needs to be sensed (see quality sensors in Fig. 5), and compared to the quality target by the controller, establishing a feedback loop.

In general, energy-quality scalable circuits and systems are able to dynamically degrade the quality to the minimum level that is acceptable for the considered application, and save energy whenever quality is degraded. The concept of quality is defined by the output fidelity to a reference, desirable, exact or correct output, and its definition is application-dependent [25]. Several metrics are available to quantify it unambiguously [2]. As exemplified by Table I, the quality target in Fig. 5 can be quantified by application-level quality metrics such as accuracy, sensitivity and specificity in binary classification tasks [26]; Peak Signal-to-Noise Ratio (PSNR) or other perceptive metrics such as Structural Similarity (SSIM) in video processing [27], and several others. Being defined in well-established expertise domains, these metrics are already well defined and widely used in the relevant communities and applications. Using the above quality metrics, examples of EQ knobs are provided in Table II for several components and sub-systems.

TABLE I
Examples of Quality Metrics in Applications and Integrated Circuit Sub-Systems

application / task / data type / sub-system	quality (Q) metric
classification	misclassification rate, accuracy, sensitivity, specificity [28]
video processing	Peak Signal-to-Noise Ratio (PSNR) Structural Similarity (SSIM) [29]
audio processing	Signal-to-Noise Ratio [30]
event detection (e.g., IoT sensors)	false alarm rate missed event detection rate [31]
feature extraction	percentage of good matches [32]
data compression	compression ratio [33]
compressed sensing data	signal reconstruction error [34]
data security	key space size [35]
random key generation	entropy [36]
Physically Unclonable Function	bit stability, uniqueness [37]
visual object tracking	tracking length [38]
unsupervised learning	perplexity [39]
approximate computation	mean error distance (MED) [40]
AD and DA data conversion	effective resolution [41]
watermarking	Komparator [42]

III. Quality Slack and Taxonomy

A. Quality Slack: A Vlsi Perspective

Energy-quality scaling takes advantage of the fact that the quality required by the application level is usually lower than the maximum quality allowed by the considered system. Similar to the concept of timing slack in digital circuits, this means that a quality slack can be defined as the difference between the maximum quality available in the system, and the actual quality that is demanded by the application. Conventional non-EQ scalable systems have fixed quality, which is hence pessimistically margined to the maximum required across all possible usage scenarios, and conditions of a given design. In particular, the quality slack is dictated by four separate contributions [2], as summarized in Fig. 6 and exemplified for a vision system making sense of the scene:

1) RESILIENCY: the design (e.g., timing, voltage) is margined pessimistically to assure that no fault occurs (e.g., timing in logic, bit flipping in memory bitcell).

TABLE II
EXAMPLES OF ENERGY-QUALITY (EQ) TRADEOFF AT COMPONENT LEVEL (Q METRIC DEFINES IF HIGHER Q IMPLIES WORSE/BETTER QUALITY)

sub-system	knob	quality (Q) metric	EQ tradeoff when knob is increased	
			E	Q
digital	arithmetic precision [40]	PSNR*	↑	↑
analog	bias current [43]	input-referred noise**	↑	↓
ADC, DAC	resolution [44]	effective no. of bits*	↑	↑
SRAM	selective assist/ECC [45]	bitcell failure rate**	↑	↓
DRAM	refresh period [46]	bitcell failure rate**	↑	↑
STT-MRAM memory read	early read time termination [47]	bit read error rate**	↑	↓
switched-cap DC-DC converter	frequency, time interleaving [48]	voltage ripple**	↑	↓
Network on Chip	error threshold [49]	data approximation quality*	↑	↑
wireless transceiver	LNA bias current [50]	bit/packet error rate**	↑	↓
data security	crypto-key bitwidth [51]	key space size in brute-force attacks*	↑	↑
sensor bus	approximate encoding of time differences [52]	measurement error**	↑	↑
sensor channel acquisition	no. of channels considered [53]	event detection accuracy*	↑	↑
image sensor	saliency-driven resolution [54]	foreground object detection error**	↑	↓
wake-up trigger	activity-sampling rate [55]	false positives**	↑	↓
digital filter	precision & voltage scaling [56]	SNR*	↑	↑
digital filter w/ ANT	over-scaled supply voltage [57]	output SNR of digital filter*	↓	↓
Discrete Fourier Transform	approximate DFT for 5G beamforming [58]	accuracy in beam angle*	↑	↑
Discrete Cosine Transform	DCT coefficient bitwidth [59]	peak signal-to-noise ratio* (PSNR)	↑	↑
feature extraction	feature ranking / selection [53]	event detection accuracy*	↑	↑
video feature extraction	EQ knobs derived from algorithm [60]	percentage of good matches	↑	↑
Support Vector Machines	# of support vectors, # features/vector [62]	classification accuracy*	↑	↑
neural network (no overfitting)	# of weights [63]	accuracy*	↑	↑
compressive sensing data representation	# of sparse coefficients [64]	SNR of reconstructed signal*	↑	↑
stochastic computing	sequence length [65]	signal error**	↑	↓

* Higher values mean better quality
** Higher values mean worse quality

Fig. 6. Quality slack in practical design scenarios, and its main components.

As a contrasting example, occasional faults might actually be acceptable in vision systems, as the presence of pixels with slightly different color does not affect the high-level understanding of the scene.

2) APPLICATION/TASK: in a non-EQ scalable platform, quality is pessimistically margined for the most demanding task or application that will be run. For example, higher quality might be required in tasks involving colors as opposed to edges, or face verification as opposed to object detection.
3) CONTEXT/USAGE: if quality cannot be scaled, it needs to be set to the highest required by the different usage contexts. For example, in a vision system the detection of an event of interest certainly requires the quality to be raised to perform more accurate analysis of the successive frames, as opposed to periods where no event is detected. Similar considerations hold for low lighting conditions versus better lighting.
4) DATASET: quality can be relaxed with the dataset allows. For example, scenes and images with higher contrast, strong texture, and high-frequency information are less sensitive to inaccuracies [66].

From the above considerations, EQ allows to take advantage of the quality slack that is allowed by the specific die and the instantaneous operating conditions, and reduce energy accordingly. In other words, EQ scaling requires dynamic adaptation to the variations in the considered die, and the time-varying conditions (task under execution, context and dataset). Conceptually, this is quite similar to conventional run-time voltage-frequency adaptation to process and time-varying environmental variations and workload, as opposed to design-time margining.

The above adaptation naturally involves several levels of abstraction to maximize the opportunities to degrade the quality in the intermediate levels, while assuring a given quality target at the application level. In other words, EQ scaling goes beyond the traditional design approach based on compartmentalization into rigid levels of abstraction. Across-level optimization permits to relax the design constraints at each level, and eliminate the quality slack that is artificially inserted at each level by pessimistically translating the application-level specifications into requirements at each level of abstraction. This justifies why quality metrics at intermediate levels of abstraction (e.g., bit error rate at the physical layer of a communication link) must be replaced by application-level metrics as discussed in the previous section.

From a control strategy perspective, each sub-system needs to be individually EQ scalable, and expose its unique EQ knobs, which are then dynamically co-optimized to bring the quality to the minimum allowed. Accordingly, the main

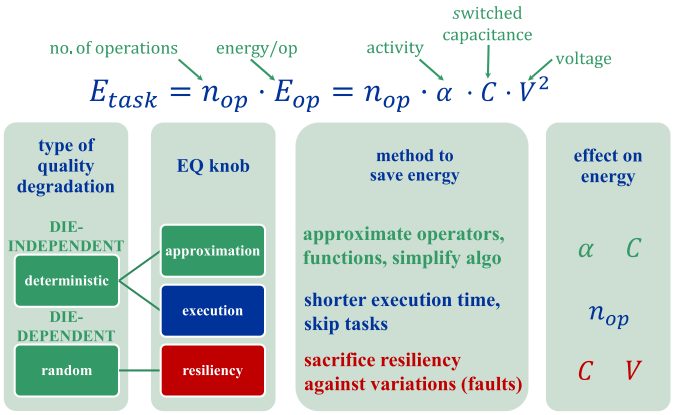


Fig. 7. Taxonomy of methods to trade off energy and quality [2].

goal of EQ-scalable systems is to 1) determine the application-level quality target, 2) inject component-level EQ knobs leading to a favorable energy-quality tradeoff (at various levels, e.g., from circuit to architecture), and 3) optimally tune EQ knobs across components to minimize energy for the targeted quality.

B. A Taxonomy of Knobs to Trade Off Energy and Quality

The EQ knobs allowing dynamic energy-quality tradeoff can be categorized according to the dependence of the associated quality degradation on the specific die where the task at hand is being executed [2]. EQ knobs that determine the same quality degradation across dice are not affected by process/voltage/temperature (PVT) variations, and introduce either approximations or execution time reduction. Instead, knobs with die-dependent quality degradation are directly impacted by PVT variations, as summarized in Fig. 7 and exemplified for a digital sub-systems.

Approximation has been extensively used to reduce quality and energy through algorithmic simplifications, gate-level and circuit-level approximations of operators and functions. The abundant literature on the sub-area of approximate computing covers a wide range of approximate adders [67]–[77] and multipliers with shortened carry chain [78]–[83], which respectively bring an energy reduction 1.5-3.7× and 1.5-3.2×. Similarly, approximate digital signal processing modules have been explored, such as FIR filters [56], multiply-and-accumulate units [62], [84], arithmetic processors [85], and Discrete Cosine Transform accelerators [59], with energy savings ranging from 2× to 4.1×. Various frameworks have been proposed to automatically synthesize approximate functions with quality target defined at design time. As few examples, some of these frameworks introduce an approximate replica to detect failures of the main unit [86], identification of minterms leading to minimum error rate [87], probabilistic pruning for both Boolean and arithmetic functions [88], identification and merge of pairs of similar signals [89], error magnitude- and frequency-aware synthesis [90], general approach for sequential [91] and combinational circuits [92]. Under such frameworks, energy savings of 1.2-7.5× have been demonstrated. When applied to deep learning engines, adjustable precision leads to energy savings from 1.8× to 4.8× [93]–[99].

Reduction in the execution time has also been explored to reduce the energy required by tasks. For example, execution time has been reduced via order reduction in digital filters [100], and sub-sampling of input data and vector basis [62]. More in general, early termination of iterative refinement algorithms and loop perforation have been widely exploited (see, e.g., [101]). To improve the energy benefits of early termination, "anytime" algorithms have been explored to guarantee monotonic quality increase over execution time [102]. In addition to the obvious energy reduction offered by shorter execution, such approaches also allow to scale down the supply voltage linearly to fit the original performance target (e.g., latency), saving energy quadratically. As execution time reduction has been investigated for several decades from an algorithm viewpoint, the available body of knowledge can be immediately reused in the design of EQ-scalable integrated systems.

As third and last category, EQ knobs determining die-dependent quality tuning by trading off energy and resiliency against PVT variations (i.e., the design margin). To enhance the energy savings, design margin can be completely eliminated [103], [104], or even made negative [45], leading to occasional errors that are acceptable as long as they are kept within bounds [57]. For example, negative margin is achieved when overscaling the clock frequency (supply voltage), so that the circuit operates at larger frequency (lower voltage) than allowed by the setup timing constraint of logic. Similar considerations apply to the bitcell stability margin in memories, where the supply voltage is set lower than the minimum voltage V_{min} guaranteeing perfect integrity in read, write and retention. The adoption of negative margin indeed permits to achieve better energy efficiency through faster operation at same voltage (lower voltage at same speed) in logic, under frequency (voltage) overscaling in logic. In memory circuits, this permits to improve the energy efficiency, thanks to the voltage reduction associated with voltage overscaling. This energy saving comes at the cost of lower quality, due to the occasional errors that alter the processing outcome. Interestingly, such EQ scaling based on resiliency reduction needs to be supported by circuits that can sense the quality (i.e., detect errors), and control it to keep it within assigned bounds, as discussed in the next subsection. Also, as discussed in the next section, this tradeoff needs to be properly managed to achieve true energy efficiency improvements.

C. Eq Scaling as Generalization of Error Detection and Correction Approaches

From a historical perspective, EQ scaling based on resiliency reduction actually represents the continuation of the long-term evolution of how errors and design margin are managed in digital sub-systems. This is shown in Fig. 9, which summarizes the design options allowed by systems having (or not having) the ability to sense/correct errors, and in platforms where occasional error are allowed, as in the several applications in Sections II and IV. Conventional systems having no ability to capture errors and not allowing any error need to prevent errors across all chips, and hence

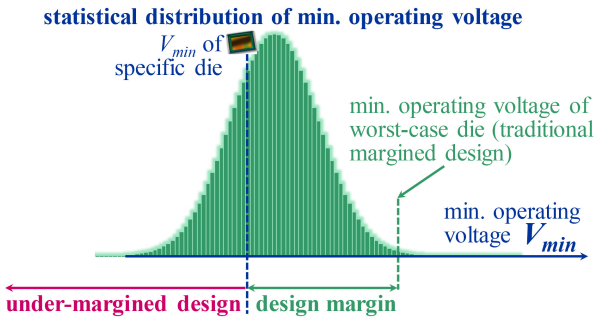


Fig. 8. Taxonomy of methods to trade off energy and quality [2].

	capable of error detection/correction	
bounded errors ↓	**N**	**Y**
not allowed	traditional (margin > 0)	EDAC (margin ~ 0)
allowed	faulty (margin < 0)	**EQ scaling (margin < 0)**

Fig. 9. Available design approaches, depending on whether occasional errors are tolerable, and whether they can be caught through detection and corrected (or at least kept within limits).

rely on traditional be margined for the worst case across PVT variations, as in Fig. 9. If errors are allowed and occur, but no mechanism is available to detect errors, the resulting errors and quality become unpredictable and uncontrollable, thus leading to faulty designs (as highlighted in red in Fig. 9). Systems not allowing errors in the output and having the capability to sense and control them at run time belong to the broad category of systems with Error Detection And Correction (EDAC), such as Razor [105], Razor II [106], EDS [103], iRazor [104] and RazorSRAM [107]. EDAC systems sense the timing margin at run time by detecting timing failures in situ, so that the system can be tuned to operate at nearly-zero design margin (i.e., the clock cycle is kept at the very minimum, without additional margin). This is crucial in both high-speed (due to the crucial importance of margin elimination to maximize speed) and ultra-low power systems, due to the strong PVT variations and design margin determined by aggressive voltage scaling [13], [108].

Let us now consider the fourth quadrant in Fig. 9 (bottom-right), which is associated with applications that can actually tolerate errors and platforms being able to sense and control errors. This quadrant is associated with systems having negative margin, where errors are kept within assigned bounds, instead of being completely suppressed as in margined and EDAC systems. In other words, this quadrant is associated with energy-quality scalable systems, which are hence the natural continuation in the evolution of digital system design that previously moved from margined to EDAC systems.

From the above considerations, energy-quality scaling simply generalizes EDAC methods to error-tolerant applications. From a design viewpoint, there are some important differences between EQ-scalable and EDAC systems, as expected by observing that the former ones need to keep errors within bounds, instead of fully suppressing them in the latter case. From an error detection perspective, full error coverage is needed by EDAC systems to avoid the disastrous consequences of escaped errors (e.g., wrong instruction being committed in a microprocessor). On the other hand, EQ-scalable systems are allowed to have occasionally escaped errors, as long as the quality can be estimated from the detected errors. For example, this allows approximate or under-designed quality sensing, as opposed to EDAC systems that require error detection to be completely error-free. This suggests that the significant area/energy overhead of error detection in EDAC systems [103]–[105] can be generally reduced in EQ-scalable systems, as they allow substantially simpler and more area- and energy-efficient circuits for detection. Furthermore, error correction/control in EQ-scalable designs can be substantially simplified compared to EDAC designs. Indeed, EDAC systems require errors to be corrected very quickly, and typically in a few clock cycles (e.g., before an incorrect instruction outcome is committed and written back to the memory). On the contrary, EQ-scalable systems need to maintain the long-term average of errors within assigned quality bounds, as the quality is actually determined by aggregate errors, rather than individual ones. As an example, quality in computer vision systems is related to the composition of several errors within the same video frame, rather than individual pixels. Similarly, in a neural network accelerator the quality is associated with the output of a multitude of neurons, instead of critically depending on the output of each specific neuron. Accordingly, the responsiveness of error correction and control in EQ-scalable systems can be drastically relaxed compared to EDAC systems (e.g., at the time scale of a frame or classification completion, rather than few clock cycles). Again, this translates into a drastic simplification of the error correction circuits.

Overall margin reduction down to zero has historically led to energy reductions of 2× or less (see, e.g., [105]–[107]). The adoption of a negative margin has led to more pronounced energy reductions ranging from 2.2× to 5× (see, e.g., [45], [115]–[117]). It is important to observe that the three design dimensions in Fig. 7 enabling energy-quality scaling are generally orthogonal, as they affect very different design aspects. Indeed, approximation is related to the implementation of sub-systems in a system architecture, the execution time is more related to algorithmic considerations and run-time task termination, and resiliency is associated with PVT variation handling. Overall, this means that the energy gains coming from these three dimensions can be composed to achieve better energy efficiency than allowed by each single dimension. This suggests that, in spite of energy reductions in the order of units for each individual dimension, the joint

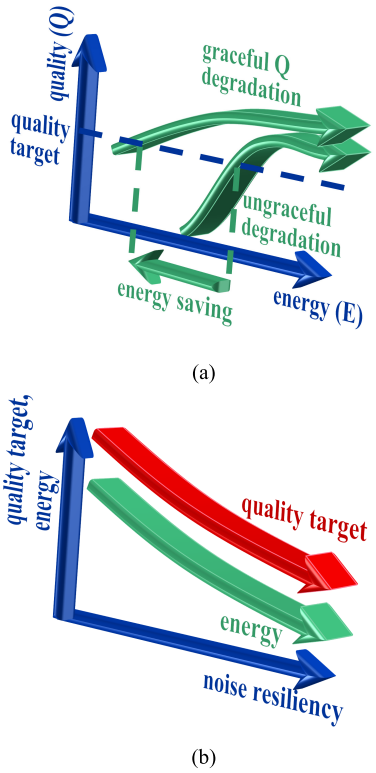


Fig. 10. a) Energy-quality scaling, and importance of graceful quality degradation to achieve significant energy savings under a given quality target, b) higher noise resiliency allows for more degraded quality target, and hence lower energy.

adoption of multiple dimensions can potentially lead to more than an order of magnitude energy reduction.

IV. Properties of Energy-Quality Scaling: Application Viewpoint

Opportunities to reduce energy under EQ scaling are enhanced in applications, systems and algorithms that have higher noise resiliency. Indeed, higher resiliency implies the ability to withstand a stronger degradation in the internal quality, while having a limited effect on the output quality target as summarized in Fig. 10a. In detail, more graceful quality degradation pushes energy towards lower values, when a quality target is assigned. In turn, this affects the choice of the most appropriate EQ knobs to be inserted in each subsystem to achieve EQ scalability. Indeed, effective EQ knobs need to substantially reduce energy, while affecting quality to a much lower extent. Accordingly, the insertion or the selection of EQ knobs requires a deep understanding at multiple levels of abstraction, from the application level (e.g., quality, complexity) down to the architectural/circuit level (e.g., energy and area penalty associated with knob insertion). Further design implications of graceful degradation assurance at the circuit level are discussed in the next subsection.

Generally, graceful quality degradation is enabled by the inherent ability of applications, algorithms and systems to be resilient against noise (or other random sources), errors, inaccuracies, including approximations and simplifications. Noise resiliency is defined as the property of dynamic systems that state deviations lead to bounded deviations of the output. Higher resiliency means that state deviations have little effect on the output, allowing larger quality degradation in internal signals for a given quality degradation at the output. In other words, higher resiliency allows for more pronounced quality degradation as in Fig. 10b, which in turn translates into lower energy from Fig. 10a. Beyond the intrinsic properties of applications, noise resiliency and hence energy efficiency can be further improved with several methods rejecting bounded inaccuracies, such as Error Correcting Codes (ECC) or online learning, at the cost of hardware and energy/power overhead.

Examples of applications, systems and algorithms with various degrees of noise resiliency are provided in Fig. 11. In this figure, the leftmost category refers to the case with zero noise resiliency, which occurs when any state deviation or inaccuracy irreversibly compromises the fidelity of the output, and the cumulative error diverges. Such extreme sensitivity is dictated by either the divergence of the state evolution (e.g., transition to the wrong state in a Finite State Machine), or by intrinsic properties that tend to amplify the effect of state changes (e.g., confusion and diffusion properties in cryptography). Similarly, the control flow of Von Neumann architectures is generally not able to withstand state deviations, and hence inaccuracies. As another example, the output of error detecting codes (e.g., Cyclic redundancy check) is purposely made very sensitive to change in the internal state and to the input, to reveals the effect of errors in digital words.

Systems and applications with limited resiliency are able to bear some bounded state perturbation, although the latter limits the time span or the complexity range within which the output trajectory deviation is within assigned bounds (i.e., quality target). As an example, the presence of noise limits the time span within which the output trajectory of synchronized chaotic systems can be predicted [109], [110]. Analogously, noise fundamentally limits the scalability of quantum computers, and the maximum number of effective qubits that can be feasibly integrated [111], and their fundamental feasibility to more pessimistic views [111].

Overall, a very wide range of prominent applications are able to counteract the effect of state deviations, as occurs in the general class of applications, systems and algorithms for data dimensionality reduction (orange column in Fig. 11). This is for example the case of feature extraction, audio/video compression, and compressive sensing, and many other functions that aim to reduce data dimensionality (e.g., Principal Component Analysis [113]). This is somewhat expected, as dimensionality reduction aims to aggregate related data points, and is hence relatively robust against inaccuracies in individual points.

Higher resiliency is achieved by higher-level data-driven tasks in which noise is inherently handled (light green column in Fig. 11), such as pattern recognition, classification and detection (e.g., in neural networks not experiencing overfitting), all perceptual tasks (as human perception is inherently imperfect), acquisition and processing of physical signals (as noise invariably affects the signal being considered), as well as statistical processing (as the overall is set by the data sets, not by individual data points).

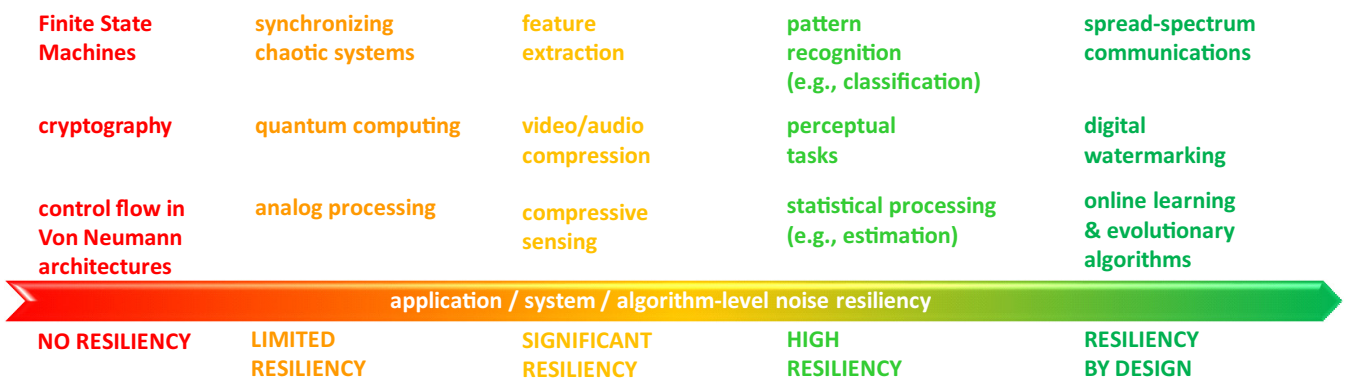


Fig. 11. Examples of applications, systems and algorithms by degree of resiliency against noise and inaccuracies.

Even higher resiliency is generally offered by systems that are resilient by design (dark green column in Fig. 11), as the function that they perform explicitly requires to reject significant noise, or can even benefit from the presence of noise. For example, this is the case of spread-spectrum communications (where the bandwidth is intentionally enlarged to improve noise resiliency), or adaptive/distilled compressive sensing (which achieves increasingly more accurate measurements when adding moderate amounts of noise). Other examples are digital watermarking algorithms (which are designed to be robust), and fuzzy extractors to extract stable keys in spite of noise and changing conditions, among the others. Error Correcting Codes (ECC) represent another class of systems that are resilient by design, and can make their output insensitive to inaccuracies, under bounded errors. On the other hand, error detection (e.g., CRC) is very sensitive to errors, and is hence very different from error correction (ECC) techniques. This is expectable from the fact that they are designed for very different response to noise and errors (i.e., respectively highlight and hide errors).

Interestingly, the expanding field of EQ-scalable design is converging with the wide interest in machine learning. Indeed, the noise resilience of machine learning circuits/algorithms can be generally leveraged to achieve graceful quality degradation. In addition, their learning ability allows even more aggressive circuit/architectural simplifications, as they can be compensated with off-chip retraining, or on-chip in-field training. This makes the synergistic adoption of EQ scaling and machine learning very promising, beyond the simple forms of EQ scaling that have been explored to date (e.g., precision scaling – see Section VII).

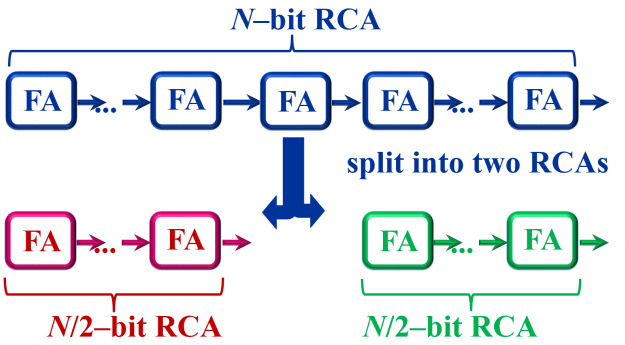
Based on the above considerations, the applications that can take advantage of EQ scaling have at least one of the following properties:

1) DATA INTENSIVE: EQ scaling is applicable only to the data path, and not to the control path of processing systems (as control flow faults would not be acceptable). Hence, the overall energy savings of EQ scaling is weighted by the portion of the system energy associated with the data path, as quantifiable by Amdahl's law [2]. Accordingly, only systems whose energy is dominated by data acquisition and processing truly benefit from EQ scaling.
2) STATISTICAL: statistical algorithms are affected by the collective dataset, and are highly resilient to inaccuracies in individual data points. Hence, statistical processing-based systems and applications are inherently resilient, and can hence benefit from EQ scaling.
3) SOFT COMPUTING: this class of algorithms is inherently tolerant of inaccuracies (e.g., evolutionary computation, fuzzy logic, machine learning, probabilistic reasoning). The resulting noise resiliency makes the related applications and systems very well suited for EQ scaling.
4) HUMAN PERCEPTION: human perception is highly robust against inaccuracies. Hence, tasks related to human perception (e.g., video streaming) or its emulation (e.g., brain-inspired computing) are naturally well suited for EQ scaling.
5) PHYSICAL SIGNALS: applications and systems dealing with physical signals inevitably deal with noise, and hence are necessarily noise resilient. Hence, they are generally well suited to extract the energy benefits of EQ scaling.
6) BUILT-IN REDUNDANCY: applications and systems dealing with data having intrinsic redundancy are inherently robust against noise (e.g., GPS, video streams in view of spatial and temporal correlation). This class is again well suited for EQ scaling.

As examples of applications that fulfill some of the above criteria, relevant applications that can leverage EQ scaling are machine learning-based accelerators (e.g., deep learning), computer vision, audio processing, speech/gesture/face recognition and others, multimedia, wearables, smart biosensors, physical data acquisition/processing, data analytics, web search, recommendation systems, human-computer natural interfaces, ubiquitous surveillance, smart sensors for the IoT, among many others. The prominence of such applications, and the trend towards more data-intensive and perceptive systems clearly makes EQ scaling very promising and even more appealing in the future.

V. Properties of Energy-Quality Scaling: Hardware Design Viewpoint

Among the other advantages, EQ scaling permits to make design tradeoffs more favorable than traditional designs.



precision →	original RCA	one half RCA (no reuse)	two half RCAs (reuse)
energy/op	1	½	½
performance (throughput)	1	2	4
area/op	1	1	½

Fig. 12. Example of EQ-scalable N-bit Ripple Carry Adder (RCA) where precision is reduced from N down to $N/2$. The resulting energy per operation, the throughput and the area per operation are reported in the table at the bottom in two cases (one where the other half RCA is not reused, the other one where the two halved adders are simultaneously used for two additions).

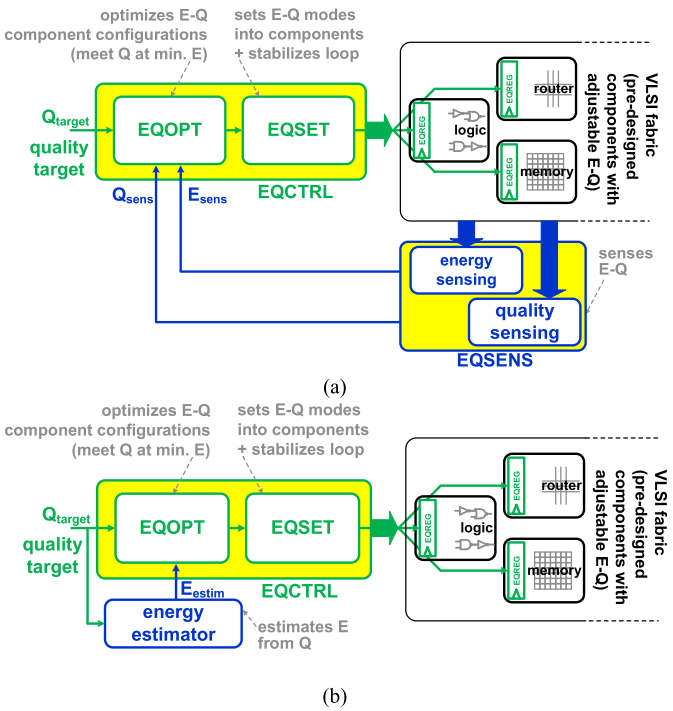


Fig. 13. Block diagram of a) closed-loop and b) open-loop run-time EQ knob control.

As representative example [2], let us consider a simple Ripple Carry Adder (RCA), which consists of N cascaded full adders [114]. Its delay and energy are approximately proportional to N, and can be reduced by reducing the precision of the addition. For simplicity, let us consider the reduction of the precision from N down to $N/2$, which reduces energy at the cost of lower output quality. From Fig. 12, both energy and delay (i.e., throughput) are improved by a factor of two, whereas no improvement in area efficiency is observed when the other half RCA is not utilized. On the other hand, when the other half RCA is used to perform two $N/2$-bit additions at a time, the throughput improves to a factor of four instead of two (thanks to parallel operation), and area efficiency improves by a factor of two. In other words, energy efficiency, performance and area efficiency are simultaneously and significantly improved. This is very different from conventional design of digital sub-systems where these design requirements are invariably conflicting. In other words, this simple example shows that EQ scaling can actually relax the interdependence of requirements, and make the overall tradeoff more favorable.

Once EQ knobs are inserted into the main sub-systems, EQ-scalable systems require the optimization of such EQ knobs to minimize the energy and meet the assigned quality target. To this aim, energy and quality need to be sensed in each sub-system, and a control strategy to optimize the EQ knobs at run time. Quality sensing at low area and energy overhead is still in its infancy, and significant room for innovation is available.

Regarding the EQ knob control, open-loop and closed-loop strategies can be adopted. In closed-loop approaches, energy and quality sensors provide real-time data on the EQ status of each sub-system to a feedback controller (EQOPT Fig. 13a). The latter optimizes the EQ knobs to minimize the overall energy under the input quality constraint. Then, the optimal EQ knobs are distributed to all sub-systems by the compensator (EQSET), which determines the transient with which knobs are adjusted to assure closed-loop stability. On the other hand, open-loop schemes in Fig. 13b do not need energy and quality sensing, with the energy being estimated from the assigned quality (through EQ pre-characterization), and sometimes from the incoming inputs. In open-loop schemes, EQSET simply adjusts the EQ knobs to have a well-behaved transition between EQ states. Open-loop schemes are simpler than closed-loop ones and reduce the energy overhead for control, although they cannot track the minimum energy as accurately as closed-loop schemes. In other words, the choice is mainly influenced by the tradeoff between control overhead and accuracy in reaching the true minimum energy.

To allow seamless SoC integration, the detailed energy-quality management based on the assigned knobs needs to be internally managed by each module. As depicted in Fig. 14, each combination of input knobs generated by determines a specific energy-quality tradeoff. Meaningful EQ modes are a small and discrete subset of the possible EQ combinations, which represents their Pareto frontier that minimize energy for a given quality. The number of available EQ modes can be fairly small, as it is not generally advantageous to tune the quality with very fine granularity. The available EQ modes are pre-characterized for each Intellectual Property (IP) or module, and are attached to the IP similarly to the timing information associated with SoC component libraries. Such EQ modes are conceptually equivalent to the power modes of IPs in the design of SoCs, with the fundamental difference that energy is traded off with quality rather than performance. The selected EQ mode is set by the controller by writing the corresponding

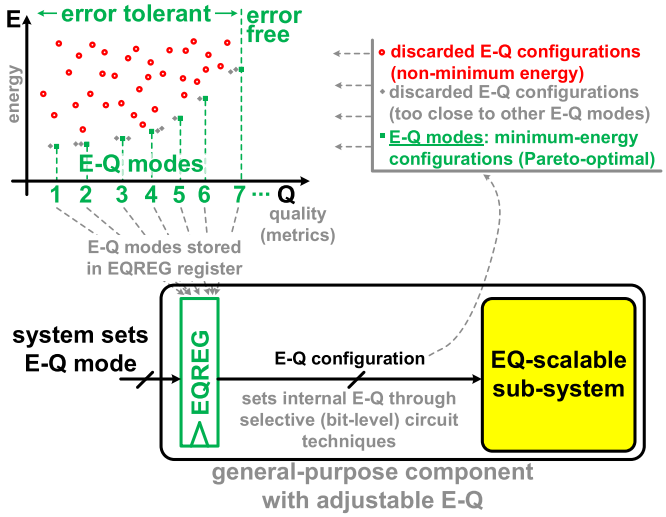


Fig. 14. EQ modes within each module are selected as Pareto-optimal EQ points. In each module, the EQ modes are internally translated into the detailed EQ knobs that are associated with them.

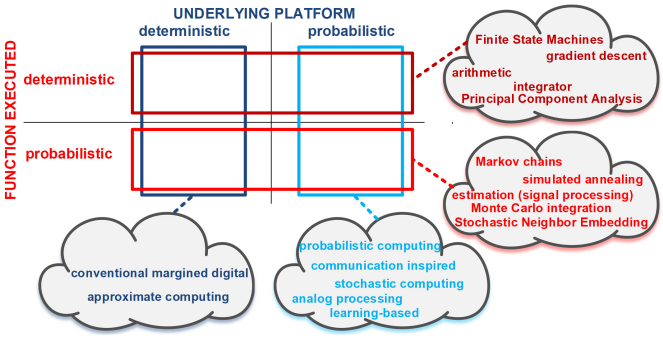


Fig. 15. Categorization of EQ-scalable functions vs underlying platforms [25].

label into the EQ register (EQREG) within the module, as depicted in Fig. 14. Then, the internal knob values associated with the EQ mode within each module are derived through simple look-up tables. In this way, the controller does not need to have visibility into each module, as it only needs to set the EQ mode and rely on the internal capability of the module to set its own knobs accordingly. In turn, this permits to keep SoC integration simple, as the simple boundary at EQREG in Fig. 14 hides the complexity of the detailed EQ knob tuning from the system. Interestingly, the EQ modes in this figure can be selected appropriately and ordered by energy, to ensure that energy and quality monotonically change across the different values associated with the EQREG values. Overall, this makes the system-level EQ knob optimization problem inherently convex, which assures rapid convergence and minimal optimization complexity [118].

Several strategies to control EQ knobs at run time have been proposed to date. For example, the open-loop RUMBA framework [119] introduces a lightweight quality estimation and correction error prediction via an inexpensive predictor. The latter is based on the observation of the inputs, and a machine learning model that combines linear regression and decision trees. The error correction consists in the re-execution of computations with large error, and in the dynamic adjustment of the error threshold to manage the EQ tradeoff. A 2.2-3.2× lower energy is achieved.

The Algorithmic Noise Tolerance (ANT) open-loop framework [116] was introduced to enable controlled voltage overscaling, whose main effect is to introduce timing errors. In turn, such errors translate into degraded signal-to-noise ratio of the signals being processed. This is a "Shannon-inspired" computing approach, as it effectively treats overscaled digital circuits like a noisy channel. The error due to overscaling is bounded via an estimator (e.g., replica with reduced precision), whose output replaces the output of the main module when the error is estimated to exceed a given threshold. An energy reduction by 2.5-5× was shown in [116], thanks to the graceful quality degradation enabled by the estimator.

The closed-loop significance-driven computation was proposed in [62] and [120], and introduces an error thresholding methodology for run-time EQ tuning under voltage overscaling. Once quality-critical computations (i.e., with higher error significance) are identified, the system extends the clock cycle operation to two cycles to allow their correct completion.

The more general "dynamic effort scaling" framework [121] was proposed to manage the EQ tradeoff across multiple levels of abstraction, from circuit to algorithm. In the example presented in [121], a Support Vector Machine processor was made EQ scalable by introducing several EQ knobs: number of support vectors at the algorithm level, arithmetic precision at the architectural level, and amount of voltage overscaling at the circuit level. Quality sensors were introduced at the same levels of abstraction: distance from decision boundary at the algorithm level, error in the comparison with maximum precision at the architectural level, and timing sensors at the circuit level to detect timing failures. A proportional-integral-derivative (PID) controller was used to assure closed-loop stability.

VI. EQ Scaling From Circuits to Architectures and Case Studies

A. General Considerations on Approaches With Deterministic and Random Quality Degradation

EQ-scalable designs can be categorized according to the function that they perform, and the underlying platform that implements it [25]. As summarized in Fig. 15, functions and platforms can have either a deterministic or a probabilistic behavior. In the following, these four categories are discussed in terms of EQ scaling, and are mapped to the broad classes in Fig. 7.

Platforms with deterministic behavior hide the randomness associated with the uncertainty in the manufacturing process, environmental conditions (e.g., voltage, temperature), noise, and other mechanisms causing erratic response (e.g., soft errors). This assure a repeatable behaviour across margined silicon dice implementing the same design, in spite of the presence of variations, thanks to the negligibly small failure rate guaranteed by the design margin. EQ-scalable systems

based on approximate computing certainly belong to this category. Indeed, determinism is assumed on the underlying platform, and approximations do not diminish the circuit-level design margin (as they are applied at the architectural or the gate level). Accordingly, approximate computing is typically mapped on deterministic platforms executing deterministic functions. The related quality degradation falls into the "approximation" category in Fig. 7. Very similar considerations can be made on the "execution time" category in the same figure.

On the other hand, probabilistic platforms do not completely hide the uncertainty that is inherent in their physical implementation [25]. Such uncertainty is explicitly dealt with in data representation and processing, and needs to be controlled to maintain adequate amount of information (e.g., SNR). Accordingly, EQ-scalable probabilistic platforms fall into the category of "random" quality degradation in Fig. 7, where different dice have different response to given stimuli. This is the case of platforms adopting negative design margin, where occasional errors are allowed to occur. As another example of probabilistic platforms, communication-inspired digital sub-systems have been explored to assure a lower-bounded output SNR in the presence of occasional errors due to voltage/frequency overscaling, i.e. treating digital circuits like a noisy channel (see, e.g., [57], [116], [117]). As another example, stochastic computing was proposed several decades ago [121] to encode information in the form of 0/1 probability in streams of random bits. This drastically simplifies the implementation of several fundamental operators, which are generally reduces to bit-wise operations (e.g., multiplier becomes a simple AND gate). In stochastic computing platforms, the signal quality can be dynamically adjusted according to the length of the stream [123], as longer streams allow finer discretization of the 0/1 probability at the cost of larger energy. As another example, probabilistic computing has been introduced to explicitly deal with logic gates with non-perfectly deterministic behavior [124], whose output is governed by probabilities rather than deterministic values. Another fundamental class of EQ-scalable probabilistic platforms, learning-based approaches are being extensively investigated to extract information from input data at run time, building models that are increasingly accurate over time (see, e.g., [125]). Interestingly, online learning permits to discriminate and suppress the effect of noise, circuit non-idealities, and other irrelevant information.

As general property of probabilistic systems based on negative design margin, quality degrades ungracefully due to the rapid increase in the bit error rate when the design margin is reduced. This is shown in Figs. 16a-b for logic circuits, in Fig. 16c for on-chip memories, and Fig. 16d for on-chip links, as fundamental sub-systems of today's SoCs. For all of them, the BER degrades exponentially when overscaling frequency or the supply voltage, as exemplified in Figs. 16a-d [103], [106], [126], [127]. Such exponential quality degradation strongly limits the allowed amount of overscaling for practicalquality targets, and hence allows very limited energy savings. To achieve more significant energy reductions, the quality degradation needs to be made graceful

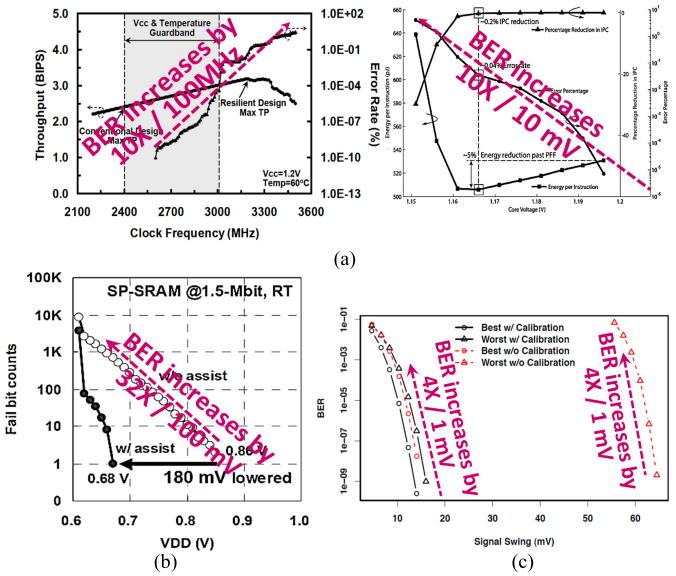


Fig. 16. Exponential error increase occurs when reducing the design margin in a) logic circuits under frequency [103] (left) and voltage overscaling [106] (right), b) SRAM memories with 6-transistor bitcell under voltage overscaling [126], c) low-swing on-chip links with overscaled swing [127].

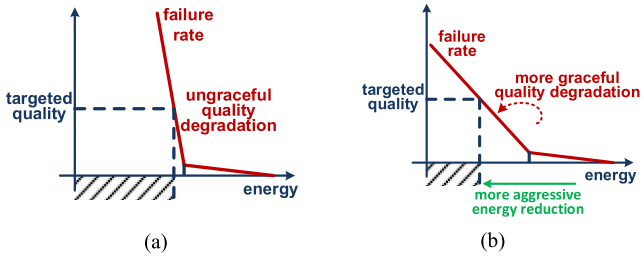


Fig. 17. Qualitative trend of failure rate and quality degradation when EQ knob is adjusted to reduce energy in a design with negative margin in a) conventional design with disgraceful (exponential) quality degradation at lower energy, b) truly EQ-scalable design with graceful quality degradation (more aggressive energy scaling is allowed for given quality target).

as shown in Figs. 17a-b. In this figure, the supply voltage V_{DD} is assumed to be the EQ knob being adjusted, although the very same considerations hold for any other knob. From the comparison of Figs. 17a-b, graceful quality degradation allows more aggressive change in the adopted EQ knob(s), and hence stronger energy reduction for a given quality target. Since graceful quality degradation is not inherent in practical implementations, EQ scaling fundamentally requires the introduction of techniques that mitigate the quality degradation. In other words, the insertion of techniques that make quality degradation graceful is a crucial task in any EQ-scalable system based on design margin reduction, and needs to be considered in first place, rather than being an afterthought. Design perspectives on the mitigation of the quality degradation are provided for some of the case studies presented in the next subsection.

B. Case Studies

The above challenge of achieving a graceful quality degradation needs to be explicitly addressed to extract the energy benefits of EQ scaling. As fundamental observation on signals

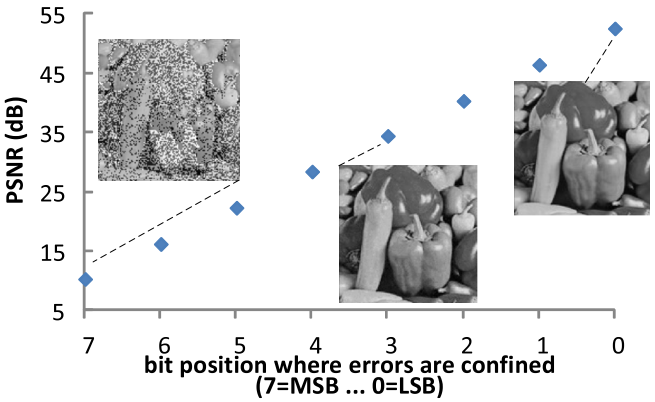


Fig. 18. When storing images on chip, errors in MSBs have a much stronger effect than in LSBs (as measured in [45]).

acquired, processed and stored on chip, let us observe that different bits, words and functions have a different impact on the overall quality and energy.

At the bit level, the numerical representation of physical signals makes Most Significant Bits (MSBs) more important than Least Significant Bits (LSBs) from the quality viewpoint [45], [115], [128]–[131]. This means that energy should be allocated non-uniformly across bits by budgeting more energy for the MSBs (to mitigate the quality degradation), and less energy on LSBs, so that the overall energy can be reduced while preserving graceful quality. Using computer vision as an example [45], errors occurring in LSBs expectedly degrade quality to a minor extent, whereas errors in MSBs make the frames unrecognizable from Fig. 18.

At the word level, luma information is well known to be more important than chroma in the Y′CBCR color space, due to the higher human eye sensitivity to brightness differences rather than color differences, and as exploited in many video coding standards [132]–[134]. The same considerations can be generalized and scaled up at the function level, observing that different portions of a given algorithm implemented on integrated circuits have different impact on the overall quality. For example, in HEVC video coding the inter-picture prediction is well known to have a stronger effect on quality than intra-picture prediction, and then entropy coding and quantization [135]. As another example related to specific sub-systems, video frame quality is weakly affected by errors in frame storage and buffering, when SRAM voltage overscaling is applied to a moderate extent [136], and when the video decoder is enhanced to correct some of the errors generated in the EQ-scalable decoder [136].

The above considerations point to the conclusion that EQ management under negative margin design has to be fine-grain and selective down to the bit level, to extract the potential energy gains of EQ scaling. Bit-level selective techniques are certainly feasible in highly regular sub-systems (e.g., SRAMs, NoCs), as the same control circuitry can be shared among many instances to minimize the related area/energy overhead. Instead, bit-level selective techniques are generally unfeasible in logic, as the cost of logic reconfiguration (e.g., multiplexing) can easily be comparable to the logic that needs to be made EQ scalable.

As a case study among the examples in Table II, the above considerations were incorporated in the design of EQ-scalable SRAMs [45], [115]. In this work, additional energy on MSBs was purposely paid for to improve their resiliency against aggressive voltage scaling, while not spending any additional energy on LSBs. This was done by introducing bit-level selective assist techniques to increase the read/write margin in MSBs (e.g., negative bitline boosting), and introducing non-uniform Error Correcting Codes (ECC) that favor MSBs over LSBs. These techniques can assure error-less operation when applied uniformly to all bits, allow graceful quality degradation when selectively applied. The resulting quality degradation mitigation allows more aggressive voltage scaling [45]. Interestingly, the achieved energy savings are superior compared to simple approximate SRAMs, as the usage of skipped bits as additional ECC check bits to strengthen MSBs allows a quadratic energy reduction, as compared to the linear reduction achieved when LSBs are kept inactive [115]. As a result, the energy per access is reduced by more than $2\times$ compared to the energy gains allowed by voltage scaling, when compared at iso-quality.

Regarding memories, EQ scaling has been also investigated in embedded DRAMs [46], in which the refresh period was dynamically relaxed to reduce the associated consumption at the cost of higher bitcell failure rate due to the capacitor discharge in the leakiest bitcells. EQ scaling of read access in STT-MRAM memories has been explored in [47], where early termination saves energy since the read current is kept ON for a shorter period, at the cost of higher read failure rate due to reduced bitline voltage swing.

From Table II, several other techniques were investigated to introduce EQ scaling at the circuit level. In analog circuits such as amplifiers (see, e.g., [43]), bias currents were lowered to reduce the consumption, at the cost of higher input-referred noise and hence poorer quality in the acquired signal. Similarly, energy-resolution scalable ADCs and DACs were investigated to reduce the energy per conversion by dynamically reducing the resolution (with a typical $2\times$ energy reduction per bit), at the cost of higher quantization noise [44], [138]. In [48], the quality of the output voltage of on-chip switched-cap regulators (e.g., ripple) was sacrificed to improve the overall system energy efficiency.

Regarding circuits for on-chip communications, approximate NoCs were investigated in [49] to trade off quality of the output signal and the energy per bit. EQ scaling was also introduced in circuits for wireless communications [50], with the bias current of the LNA in the receive being tunable to trade off consumption and packet error rate. Data security has also been traded off with energy by dynamically adjusting the bitwidth of the cryptographic key, depending on the demanded level of security [51]. Communication with off-chip sensors via the sensor bus has also been made EQ-scalable, encoding the sensor signal so that approximate time differences across samples can be transmitted, when lower energy is required and under the quality that allows it [52].

In multi-channel sensor systems, the number of enabled channels has been made scalable to trade off the quality (i.e., overall amount of information received from sensors)

and energy per acquisition, which is lowered when fewer channels are activated [53]. In image sensors, the resolution has been made scalable to trade off image quality and energy per frame, and has been adjusted according to the saliency of the scene. In the same area of sensing, the always-on sensor interface for event-driven wake-up in [55] has been made scalable in terms of sampling rate, to trade off the consumption and the false positive rate.

EQ-scalable digital filters were investigated in [56], where more aggressive dynamic voltage scaling is enabled when the precision is down-scaled to trade off energy and quality, and exploit the additional timing slack allowed by the lower precision. Voltage overscaling is introduced in digital filters [57], where the output quality is controlled via an estimator, as discussed in Section V. Transform accelerators were made EQ scalable by introducing approximate Discrete Cosine Transform (DCT) in [58] for 5G beamforming, and dynamically adjusting the DCT coefficient bitwidth in [59].

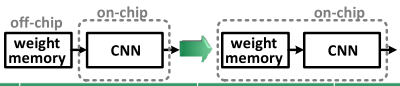
Accelerators for higher-level tasks were also made EQ scalable, as in the case of feature extraction [53] where the features were ranked by importance to dynamically discard the least important ones. The adoption of EQ scaling in always-on systems for computer vision was also shown to reduce the energy in feature extraction down to few tens of pJ/pixel, and power down to tens of μWs instead of mWs [60]. This opens the opportunity to embed vision capabilities in IoT and energy-harvested systems [61].

Machine learning accelerators for classification were also made EQ scalable, as in the case of Support Vector Machines [62], whose number of support vectors and features per vector were dynamically adjusted. Similarly, the number of non-zero weights in neural networks was dynamically increased to improve classification accuracy at the cost of higher energy (assuming that accuracy monotonically improves with a larger number of weights, i.e. the neural network has been designed to counteract overfitting).

At the data representation level, EQ scaling was explored in the context of compressive sensing, where the number of sparse (non-zero) coefficient was dynamically reduced to reduce the energy associated with the signal reconstruction, at the cost of lower SNR [64]. Analogously, stochastic computing accelerators were made EQ scalable by dynamically adjusting the length of the bit stream sequence associated with individual pieces of data [65].

VII. EQ Scaling: Case Studies at the Algorithm Level and Convergence With Machine Learning

Convolutional Neural Networks (CNN) running on GPU or CPU have emerged as the workhorse of a variety of computer vision and machine learning applications in the cloud data centers. However, the memory, compute and the associated area and energy requirements are too big for real-time embedded IoT devices. In particular, the weight memory need to fit on the on-chip SRAM since large DRAMs are not affordable in IoT platforms, as summarized in Fig. 19. The compute engine needs to be orders-of-magnitude more compact and energy-efficient than GPU or CPU, requiring the development of flexible accelerators.



	off-chip DRAM (LP-DDR2)	on-chip SRAM	SRAM vs DRAM
capacity	GB	MB	1,000X
energy/bit	40pJ/bit	30-150fJ/bit	1,000X
max BW	8GB/s	set by design	-
power @ max BW	2.5W	"	-
power @ AlexNet 30fps (3.6GB/s), 16bit/weight	1.15W	1.15mW	1,000X

Fig. 19. Memory requirements in CNN accelerators for edge computing.

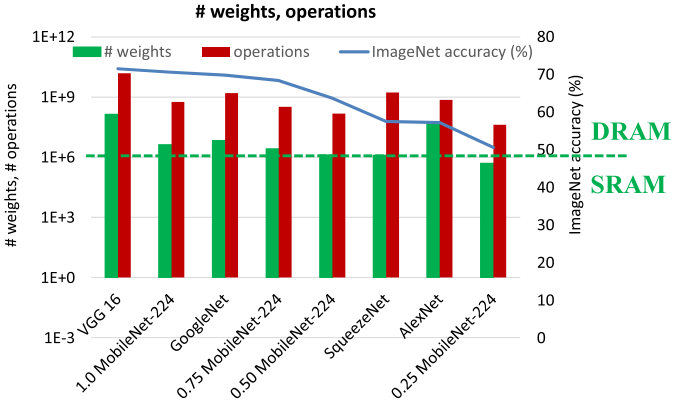


Fig. 20. Number of weights, elementary operations (additions, multiplications), and accuracy under the ImageNet benchmark of state-of-the-art CNNs.

More compact ImageNet architectures such as SqueezeNet [139] and MobileNet [140] reduce the amount of weight memory and associated compute operations required for 224 × 224 pixels image classification applications from GB to MB (see Fig. 20). Then, on-die embedded SRAM in IoT platforms is sufficient and an external DRAM is no longer necessary. Clearly, the classification accuracy degrades significantly as the number of pixels and MobileNet parameters are reduced. However, these quality degradations are graceful and can be effectively exploited for energy-quality tradeoffs in many IoT platforms and applications (see Fig. 21).

Weight compression techniques can further alleviate the memory requirements for neural network implementations in IoT platforms. For example, iterative pruning of connections and retraining helps preserve the most important connections in a deep learning neural network [63] without causing significant degradations in the accuracy of the trained network (see Fig. 22). Furthermore, smart quantization of the weight values with optimal encoding schemes that provided high resolution only in range where the weight values are clustered enables usage of fewer bits per weight value. This can be combined with retraining of the coding scheme to produce further compression without compromising accuracy significantly. This weight sharing approach can reduce weight memory requirements by 20-30× without degrading accuracy.

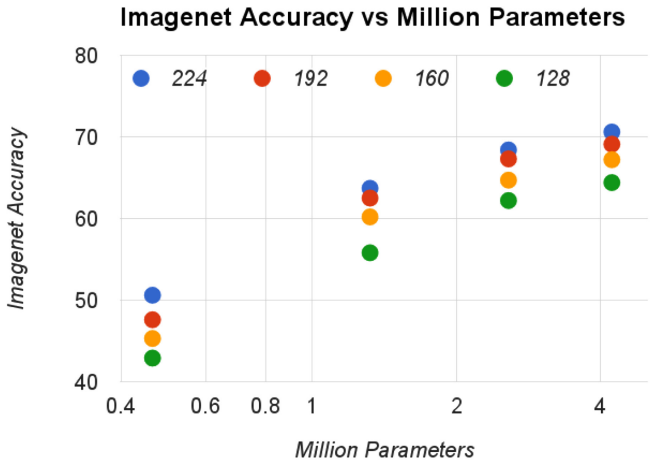


Fig. 21. Accuracy-memory tradeoff in MobileNet shows graceful degradation when memory is shrunk to reduce silicon area and energy [140].

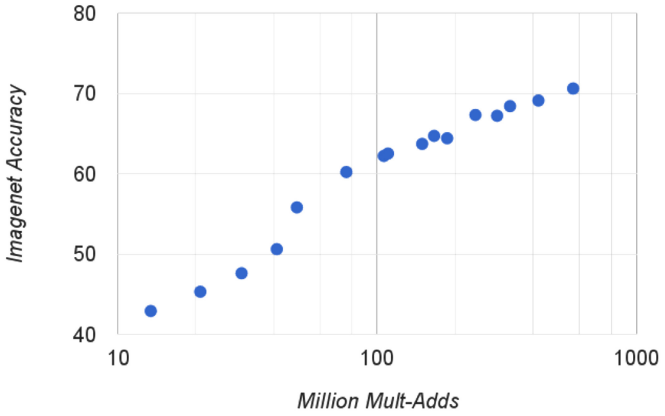


Fig. 23. Accuracy-processing tradeoff in MobileNet shows graceful degradation when the number of operations is reduced to save energy [140].

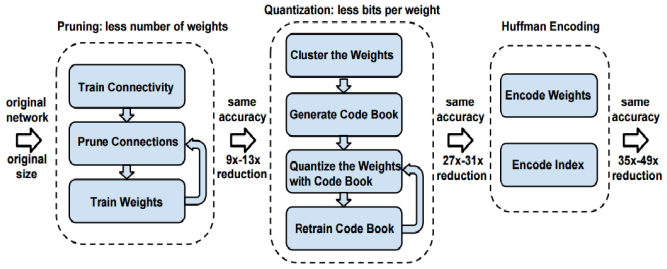


Fig. 22. Summary of energy savings obtained via the techniques in [63].

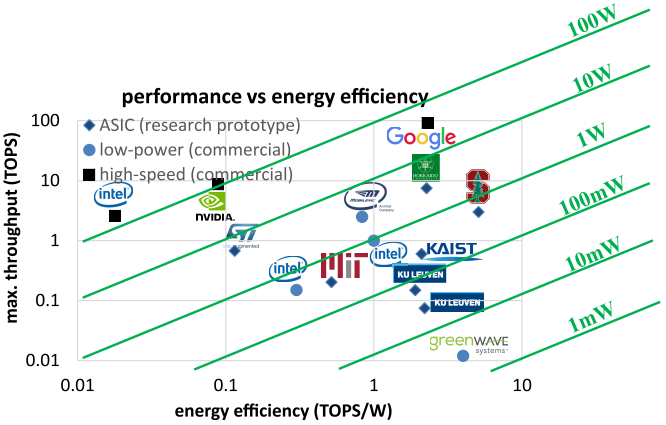


Fig. 24. Performance-energy efficiency tradeoff in state-of-the-art CNN accelerators.

Another 35-50× improvement is possible by using Huffman encoding based compression when many 0's are present.

Computational throughputs of embedded CNN research prototype implementations in IoT platforms range from sub-GOPS to 100 TOPS [93], [95]–[99], with low power commercial CPUs providing a few TOPS and GPUs offering tens of TOPS. Computational energy efficiencies of a few TOPS/W have been demonstrated. Computational area efficiency values span 0.001 to 0.5 TOPS/mm2, and benefit from technology scaling. ImageNet accuracy scales gracefully with the number of compute (multiply-add) operations and saturates beyond a certain point [140], as shown in Fig. 23. Also, architecture and design techniques that improve area efficiency boost both energy efficiency and performance via reduced wire length and logic optimizations for dedicated fixed function accelerators. Better energy efficiency translates directly to higher throughput performance in power-limited IoT platforms (see Fig. 24).

Low energy CNN accelerators utilize scalable precision of weight values, data and computation, and combine them with voltage-frequency scaling to achieve a range of energy efficiencies and performances [97]. Lower precision improves both energy efficiency and operational frequency at a specific voltage – thus, providing additional energy improvement at a fixed frequency target at the expense of accuracy. In addition, using a "memory guarding" scheme, where weight values of 0 are not fetched, enables exploitation of sparsity and provides another 2× improvement in power consumption.

State-of-the-art dedicated and reconfigurable machine learning engines implement a range of architecture and design techniques to achieve energy-quality scalability. Notable examples are: (i) zero input skipping with clock gating [98], (ii) data reuse [98], (iii) run length encoding to provide energy-quality scalability to reduce DRAM traffic [98], (iv) hierarchical processing [94], (v) precision scaling [94], (vi) dynamic fixed point arithmetic with online adaptation of integer/fractional parts [95], (vii) weight quantization using multiplier based on quantization LUT [95], (viii) data fabric reconfiguration [96], (ix) nonlinear weight quantization with local decompression [96], (x) adaptive SRAM [96], and (xi) zero weight indexing [99].

Binary neural networks have been recently developed to achieve extreme scaling of computational complexity and memory requirements with only marginal degradation in accuracy [141], [142] for machine learning applications at the edge of IoT. In-memory neural network processing, without any external data accesses, enabled by the symmetry and simplicity of computation of the binary neural network, improves energy efficiency dramatically as demonstrated by the BRein processor-in-memory (PIM) research prototype

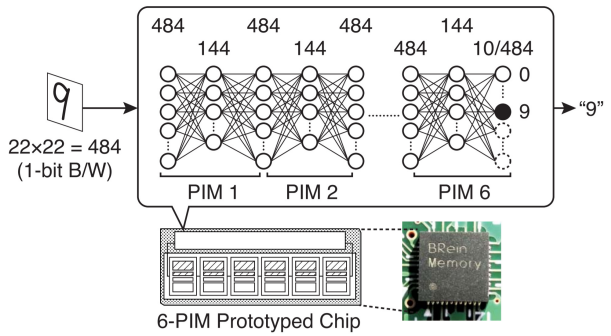


Fig. 25. Example of in-memory computing for binary neural network acceleration [143].

achieving 1.4TOPS at 0.6W [143]. Thus, implementations of intelligence at the edge of IoT can exploit the inherent tolerance to imprecisions and approximations offered by many machine learning and cognition workloads to achieve extreme energy efficiency with compact hardware accelerators and small amounts of on-die memory without significant degradation in quality.

VIII. Software-Level Approaches to EQ Scalability: Language Constructs, Compiler and Run-Time Layers

Several software-level approaches have been proposed to enable EQ scaling, as exemplified in Fig. 25 with some of the most relevant prior art. To exploit the inherent resilience to computation approximation offered at the application level (see Section IV), prior techniques have focused on

1) using unreliable hardware components, or operating the hardware close to the edge of failures (like controlling the voltage scaling in logic circuits, the DRAM refresh rate or the SRAM supply voltage) [144]–[152]
2) reducing the numerical precision of target computations, for example by lowering the precision of floating-point (FP) computations or by implementing FP to fixed-point conversion [153]–[158]
3) deploying unsound code transformations, namely transformations that change the semantics of an original exact program (within tolerable boundaries, see Section III-B). Notable examples include task skipping [159], [160], a run-time technique for best-effort parallel computing which saves energy by dropping tasks (atomic parallel computation units), loop perforation [101], [161]–[163], a compiler technique that skips iterations of time-consuming loops in a controlled manner, allowing to operate at different points of the EQ tradeoff space, and various techniques that can be grouped under the denomination of approximate parallelization. Several transformations can be applied by a compiler to selectively relax strict adherence to language semantics, thus controlling the EQ tradeoff. Among these, various forms of relaxed synchronization in parallel programs have been explored [164]. For example, the barrier placed at the end of parallel loops can often be relaxed, as data generated by loops are not always used immediately [165]. Similarly, shared data locking can be relaxed, in a manner similar to traditional transactional memory [166] or lock elision [167] approaches, under additional guarantees that acceptably accurate results are provided [168].

Different approximate parallelization techniques have been proposed in combination to common and well-understood data parallel patterns (Map, Scatter/Gather, Reduction, Scan, Stencil, and Partition). Typically, the approximation involves computing the outputs operating on a subset of the input array, and predicting the results for the rest of the array, like in the scan pattern. For the reduction pattern, Paraprox [169] uses sampling [170] and adjustment to only compute the reduction of a subset of the data. The reduction node operates on a randomly selected subset of its inputs, simultaneously eliminating the computations that produce the discarded inputs.

The concept of fuzzy computation has been applied to the memoization technique to take advantage of the EQ tradeoff [171]. Approximate memoization has been studied in the context of loop perforation [172], and has been applied to map and scatter/gather patterns where computations are replaced by memory accesses [169]. Stencil and partition patterns assume that adjacent locations in an input array are typically similar in value, thus only a subset of values in the input array is accessed and replicated to construct an approximate version of the array.

In some cases, software approaches have been tailored to the specificity of a particular application domain, which has allowed to better craft the approximation solutions. A representative example is a set of techniques proposed in [159], aimed at improving best-effort system behavior and leveraging unique characteristics of iterative, convergence-based recognition and mining (RM) applications:

1) convergence-based pruning uses converging data structures to speculatively identify computations that have minimal impact on results and eliminate them
2) staged computation attempts to expedite overall convergence rate by considering fewer data points in early stages, and then gradually considering more points in subsequent stages, to refine initial estimates
3) early termination aggregates statistics to estimate accuracy and terminate before absolute convergence
4) sampling selects a random subset of input data, and use it to compute the results (useful when significant redundancy is expected in the input data)
5) dependency relaxation ignores potentially redundant dependencies across iterations. Iterations can then be parallelized, leading to higher degree of parallelism, or coarser parallel granularity.

Each of these approaches has proven effective at reducing energy consumption within the tolerated quality loss, when applied in isolation. However, one single solution typically does not fit all use cases, thus researchers have put significant effort into providing integrated solutions that allow application developers and system designers to dynamically select the best approximation approach, while keeping the desired quality under control. To this end, all the described techniques have been explored as part of more complex software stacks that typically involve different layers.

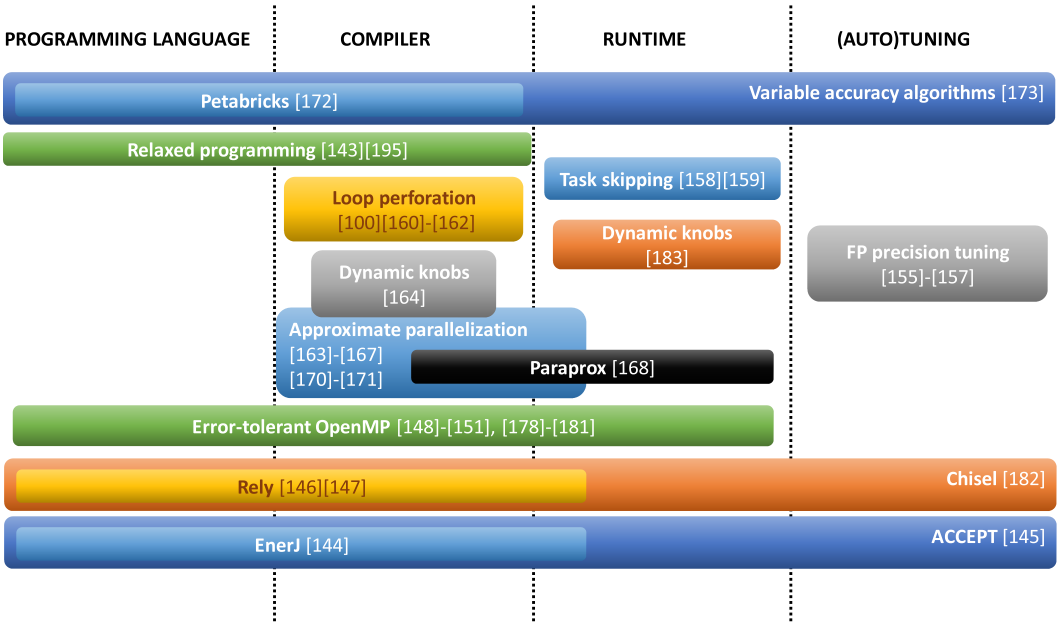


Fig. 26. A taxonomy of main software techniques leveraging EQ scaling.

At the programming language layer, several efforts have been made to extend widely adopted programming models/languages with custom constructs. The latter ones allow application developers to express where and when approximation in their codes could be tolerated. Compilers play different roles in the context of approximate computing, from translating custom constructs into run-time variants of the program (code generation), to providing static guarantees that the transformations are safe (disciplined approximate computation, guaranteeing strong program semantics). Approaches focusing on these two layers are surveyed in Section VIII-A.

While static approaches help limit the negative effects of approximation, some techniques can imply more severe quality degradation that accumulates during program execution. Hence, at the run-time layer several approaches have been studied to dynamically inspect the quality of a program's results and react to it. Compiler and run-time approaches provide convenient support for executing approximate versions of programs, where a certain number of parameters have been tuned according to a specific configuration. However, finding the best configuration is typically an iterative process, for which tuning methodologies and tools are required. (Auto)tuning layers efficiently search for a program's best approximation parameters. Approaches focusing on the latter two layers are surveyed in Section VIII-B.

A. Language and Compiler Layers

Compilers and related tools have played an important role in aiding the deployment of imprecise computation, giving programmers control over the use of approximation. Most often, this has also implied the specification of a custom set of language constructs to achieve this goal. Table III summarizes some of the most eminent examples of such approaches, highlighting the key language constructs and the optimizations/analysis employed.

In some cases, "giving control" means aiding the selection and deployment of algorithmic variants as a means of managing EQ scaling. Multiple selectable implementations represent an effective way of dynamically adjusting the system capabilities to deliver the targeted EQ tradeoff. Researchers have developed several approaches that allow programmers to provide multiple implementations for a given piece of functionality, each occupying a different point in the EQ tradeoff space. This type of approaches typically makes no specific assumptions on the underlying hardware, as the target can be any best-effort technique aimed at reducing energy consumption in both reliable and unreliable systems.

Petabricks [173] is a parallel programming language and compiler that developers can use to provide alternate implementations of a given functionality. It offers extensions that make algorithmic choice a first-class construct of the language. The focus is on performance rather than EQ tradeoff, but the concept can be extended to fit this context. The custom language constructs allow to define inputs and outputs to a function-like entity called a *transform*, in addition to rules to convert the inputs to the outputs.

Later additions to the language focus on supporting variable-accuracy algorithms [174] (e.g., approximation algorithms, iterative methods, data resampling, heuristics) by exposing the tradeoff between time and accuracy to the compiler. Key language constructs are *accuracy_metric*, which specifies a user-defined transform to compute accuracy for an input/output pair; *accuracy_variable* to define algorithm specific parameters that affect the accuracy of the program; *for_enough* to describe a loop where the compiler can change the number of iterations needed for each accuracy level and input size; *verify_accuracy* to direct the compiler to insert a run-time check for the level of accuracy attained. The Petabricks compiler also relies on an autotuning layer to construct optimized algorithms to achieve the targeted accuracy (see Section VIII-B).

TABLE III
EXAMPLES OF LANGUAGE CONSTRUCTS TO ENABLE SELECTION OF ALGORITHMIC VARIANTS AND SAFE APPROXIMATIONS

	Language constructs	Description	ROLE of COMPILER / Enabled compiler optimizations

Petabricks [173]	***ALGORITHMIC CHOICE***
	`Transform [from, through, to]`
	
	VARIABLE ACCURACY
	`accuracy_metric`
	`accuracy_variable`
	`for_enough`
	`verify_accuracy`
		Defines inputs and outputs to a function, plus *rules* to convert the inputs to the outputs.
		
		Define an accuracy metric for an input/output pair.
		Define parameters that influence the accuracy.
		A loop where the number of iterations can be reduced.
		Insert a runtime check for the level of accuracy.
			Aid use of algorithmic variants as a means of controlling EQ
			
			Provides accuracy guarantees:
			• Static guarantees
			• Runtime checking
			• Domain-specific guarantees

Green [175]	`QoS_compute`
	`QoS_approx`
	`QoS_ReCalibrate`
	`approx_loop`
	`approx_function`
		User-provided function for computing the QoS
		Compiler-synthesized functions to approximate/revisit approximation decisions at runtime
		Annotate candidate loops for approximation
		Annotate candidate functions for approximation
			loop and function approximation (users specify algorithmic variants)
			• Loop perforation
			• Function memoization

Rely [147] [148]	***RELIABILITY SPECIFICATIONS***
	`int<0.99*R(x)>`
	
	MEMORY REGION SELECTION
	`Urel`
	
	UNRELIABLE COMPUTATION
	`-.`
	`<.`
		Specify reliability requirements
		
		Allocate data in unreliable memory regions
		
		Specify use of unreliable ALU logic
		(e.g., unreliable subtraction and comparison).
			Verify quantitative reliability for programs that execute on unreliable HW

Chisel [183]	***Inherits Rely constructs*** `<d, r * R(d1) >= D(x1), ..., dn >= D(xn))>`	Specify maximum acceptable difference between the approximate and exact result (***d***). Specify probability that the kernel computes a value within ***d*** of the exact value (***r***)	Solve ILP to select approximate instructions and storage
EnerJ [145]	@Approx, @Precise, @Top Endorse @Approximable @Context	Annotated type can be precise or approximate Cast an approximate value to its precise equivalent. Annotated class can have precise or approx instances. Precision of a type inherits from the enclosing object.	Precise-purity analysis for approximate region selection
Accept [146]	***Inherits EnerJ constructs*** _APPROX ACCEPT_PERMIT ACCEPT_FORBID	Marks a method that may be invoked when the receiver has approximate type. forces a statement to be considered approximate forces a statement to be precise	Safe approximate relaxations • loop perforation • synchronization elision • neural acceleration

Green [175] also exposes a set of language constructs to the programmer, to enable the deployment of algorithmic variants (focusing on loop and function approximation), while keeping quality degradation under control. The custom language constructs allow the definition of user-provided functions to evaluate the program's quality QoS (*QoS_Compute*). This allows to annotate candidate loops and functions for approximation (approx_loop, approx_function), and internal symbols to label compiler-synthesized functions to approximate/revisit approximation decisions at run-time (*QoS_approx*, *QoS_ReCalibrate*).

Other relevant approaches include Eon [176], a coordination language that selects between different implementations based on the system energy state (the Eon run-time executes only the flows that the programmer has marked as suitable for the given energy state). Another approach introduces the tunability interface [177], which allows application developers to specify 1) multiple configurations of an application via compiler directives (a tunable application is then modeled in a virtual execution environment, to determine which configuration is best suited for different system states), and 2) the substitution transformations [170], which allow to select one of the available implementations for a given function, with each implementation offering a different combination of accuracy and resources.

In some other cases, "giving control" means that the targeted computation is mapped on approximate hardware (e.g., providing reliable and unreliable versions of standard arithmetic/logical instructions and memories), while at the same time providing quality/accuracy guarantees [147], [162]. Custom language extensions allow to manually identify unreliable instructions and variables that can be stored in unreliable memories. Often in this case the compiler also

provides support in verifying user-specified correctness properties in relaxed programs and in guaranteeing strong program semantics.

EnerJ [145] provides a simple non-interference guarantee by means of a type system capable of distinguishing data that can tolerate errors from data that require full precision (*@Approx, @Precise*). Typing rules prevent approximate-to-precise information flow, unless explicit endorsement (the *endorse* keywork) is used. *@Approximable* classes can have both precise and approximate instances.

Rely [147], [148] statically determines the probability that values produced by an approximate computation are correct. In other words, the probability that values produced by an approximate computation are correct is evaluated by examining the static data flow of nondeterministic operations. The developer provides a reliability specification, which identifies the minimum required probability with which the kernel must produce a correct result. Similarly, language constructs are provided for memory region specification, and unreliable computation specification. The Rely analysis then verifies that the kernel satisfies the reliability specification for all inputs, under the identified unreliable instructions and variables.

For both EnerJ and Rely, EQ tradeoff exploration is entirely in the developer's hands, as he/she is responsible for identifying the unreliable operations and data in the program. The authors of these tools have thus provided extensions with additional language constructs and a tuning methodology to simplify this task (see Section VIII-B).

Another noteworthy example of compiler-level support to approximate computing is the implementation of dynamic knobs, i.e., configuration parameters that can be changed as the program executes. HELIX-UP [165] offers a set of knobs that control the type of relaxations that can be applied to the application parallelization. For example, a sequential segment knob decides on the amount of parallelism applicable to a given code block. The most optimistic configuration is fully-parallel, with no synchronization between threads executing that code block. The most conservative is fully sequential, with threads treating the code block like a large critical section. In between, there is a wide range of synchronization options that can be selected. The loop-end barrier knob decides on the number of threads that need to synchronize at the end of a parallel loop. In the most optimistic case, there is no barrier implied, whereas in the most conservative case all threads are involved. In between, a configurable number of threads can be selected. Other knobs are used to control grouping iterations (similar to OpenMP loop chunking), loop iteration communication (from a fully DOALL to a fully DOACROSS loop) and SMT (to decide whether to use simultaneous multi-threading to hide the communication latency between cores, or to perform actual computation).

Several authors have proposed extensions to the popular OpenMP parallel programming API as a convenient means to control the use of approximation in programs. In some cases, such extensions are aimed at expressing the minimum quality requirements and the relative importance (significance) of individual computations for the final quality [178]. Such approach operates in conjunction with a significance-aware run-time system that relies on an analytical energy model, to identify the degree of concurrency and approximation that maximizes quality while meeting user-specified energy constraints.

In other prior art, the focus is on systems where energy savings are obtained by abandoning conservative CMOS design practices based on the addition of margins (guard-bands) to absorb worst-case PVT variations (see Section III-B), or by operating the system very close to the point of failure (near-threshold computing). In such conditions, substantial energy savings are possible, but variability-induced timing errors appear, and the problem becomes that of ensuring correct system behavior. In this context, several authors have focused on applications that are amenable to approximation of results, for which some of these errors can be tolerated rather than corrected (without compromising the expected quality). Researchers have shown that embedded shared-memory compute clusters can indeed benefit from very significant energy savings by extending the run-time support to all the main OpenMP parallel constructs (loops, sections [179] and tasks [180], [181]), with features that improve the robustness against variability-induced errors. Rather than extending the language, the EQ scalability features are all automatically handled by the compiler, which invokes custom run-time library functions to monitor the execution (i.e., the quality) in a transparent manner to the programmer, and take appropriate scheduling decisions to minimize the energy. Custom OpenMP directives have been proposed to extend the operation of this framework to accuracy-reconfigurable floating-point units [182]. In this case the additional directive `#pragma omp approximate` is used to identify portions of code suitable for relaxed FPU operation and the `error_significance_threshold` clause is used to specify the tolerated error for the computation involved.

OpenMP extensions to specify which regions of code and what variables are tolerant to approximation have also been proposed, in conjunction with hybrid memory system management for embedded SoCs [149], [150]. Standard-cell memory (SCM) and 6T-SRAM are used to implement L1 on-chip storage, and hardware support is provided to split error-tolerant data, placing the most significant bits (MSB) on the SCM and the least significant bits (LSB) on the 6T-SRAM. This allows extremely low voltage operation while ensuring correct operation by confining possible flip-bit errors to the LSBs only, as the SCM is always reliable.

OpenMP has finally been exploited to program a revisited form of transactional memory system to recover from errors, when aggressive voltage scaling makes the output quality unacceptably low [151], [152]. In this case, a compiler transparently wraps the code inside OpenMP constructs to outline resilient transactions (RTx). The underlying run-time system maintains a core-level error-management policy that optimistically lowers the voltage in small steps to save energy, while dynamically monitoring the output quality to quickly restore the system operating conditions to a safe level, in case the RTx abort rate exceeds the tolerated threshold.

B. Run-Time and Tuning Layers

Static compilers analyze the relaxed programs and provide accuracy guarantees, bounding the negative effects of approximation. The resulting error margin bounds prevent catastrophic errors in the system's control path (like the non-interference property of EnerJ [145]). However, some approximations can be more dangerous than others in terms of their effect on the program quality, which should hence be dynamically monitored. Many of the approaches discussed in Subsection A include a run-time and/or a tuning component that helps developers to achieve such goal. Dynamic quality monitoring and quality debugging tools help programmers understand and control quality degradation.

Offline debugging tools instrument programs to determine the critical data locations, and code points that mostly affect quality, as suitable for early exploration. Online mechanisms dynamically monitor quality and allow programs to adjust approximation levels or re-execute code. Such mechanisms are typically lightweight enough to run in deployed code (unlike offline monitoring/debugging approaches), and adaptively adjust approximation levels, or correct erroneous results that arise at run-time.

Early versions of approximate computing frameworks such as Relax [144] and EnerJ [145] primarily focused on safety properties, thus providing features to identify the parts of a computation that may be subject to error, rather than to monitor the quality of computations. Green [175] uses heuristic control to manage quality, but it does not control or even monitor performance. Optionally, it can invoke user code on sampled executions to assess quality, where the programmer must provide an appropriate monitoring scheme. Petabricks [173] does not have a dynamic control component, but its language extensions and compiler features allow developers to auto-tune variable-accuracy algorithms. Eon [176] uses a heuristic control system to manage the energy consumption of the system but does not directly control quality. Quality-of-service profiling [161] uses offline profiling runs during development to examine the impact of unsound code transformations (loop perforation) on the quality. This allows to identify code that has little influence on output quality, which can be considered by the programmer to relax the code and improve its performance.

More recent approaches and evolutions of the frameworks mentioned above address the issue of quality, which is more concerned with understanding and controlling the degree of error allowable in a program. Accept [146] is an evolution of EnerJ [145], and combines three main techniques: 1) a programmer-compiler feedback loop consisting of an enhanced set of source code annotations compared to EnerJ (see Table III: the *ACCEPT_PERMIT* annotation forces a statement to be considered approximate and *ACCEPT_FORBID* forces it to be precise, forbidding any relaxations involving it), plus an analysis log; 2) a range of automatic program relaxations (loop perforation, synchronization elision, neural acceleration); 3) an auto-tuning system that uses dynamic measurements of candidate program relaxations to output a set of Pareto-optimal versions of the input program that reflect its EQ tradeoff space.

Chisel [183] is an evolution of Rely [148], and automatically selects approximate instructions and data that may be stored in approximate memory, given the exact kernel computation and the associated reliability and/or accuracy specifications. To do so, Chisel relies on an integer linear program formulation aimed at minimizing energy consumption while satisfying the quality requirements. This automates the exploration of the EQ tradeoff space, lifting this responsibility from the developer, who is instead assisted in deriving the reliability specification at the application level. Specifically, the developer provides three inputs: 1) *Sensitivity Metric*, i.e. a function of the difference between the outputs of the original and approximate executions; 2) *Sensitivity Goal*, i.e. the value of the sensitivity metric that satisfied the QoR requirement; 3) *Sensitivity Testing Procedure*, i.e. fault (noise) injection routines to explore the sensitivity of the program's results.

Powerdial [184] transforms static configuration parameters provided by the user into dynamic knobs, and contains a control system that uses the knobs to maintain a given performance level independent of any fluctuation or event that the application experiences at run-time. Powerdial works directly on unmodified and unannotated applications by automatically transforming existing configuration parameters into dynamic knobs, and inserting Application Heartbeats API [185] calls into the application.

Decaf [186] infers operator reliabilities to meet programmer-specified correctness bounds. Decaf's type-based approach (i.e., probability type qualifiers) targets approximate architectures with multiple probabilities. It augments static guarantees with run-time monitoring by specializing functions according to quality demands in calling contexts.

In general, offline quality debugging is useful for understanding quality tradeoffs during development and testing, while online quality monitoring is needed to effectively detect and correct quality degradations in approximate applications [187].

C. Transprecision Computing

Research on approximate computing has matured to the point where several approximate hardware designs and design methodologies are available. This adds to existing compiler optimizations and language tools to explore accuracy-efficiency tradeoffs. So far, a more widespread adoption of such techniques has been barred by the lack of an application-to-hardware framework to manage precision without compromising application quality. Also, there is consensus that the next steps in research should be focused on domains where approximations are already widespread or compulsory [188]. In domains where approximation is a fact of life, approximate computing is already deployed.

Among the most obvious application domains where compulsory approximation is encountered, numerical analysis and scientific computing probably provide the most representative example. Floating-point numbers represent a well-established instance of approximate computing. In scientific computing,

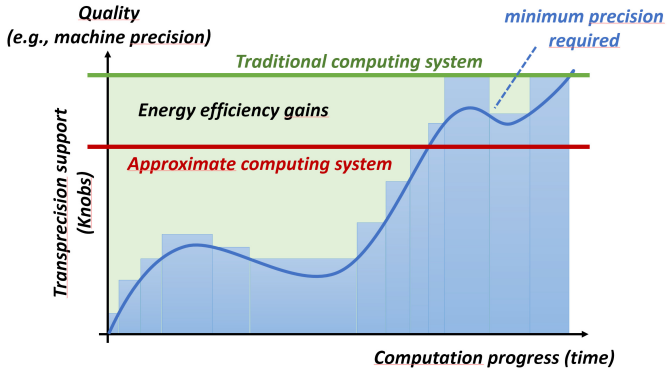


Fig. 27. Transprecision computing principle: fine-grained control of accuracy relaxation to save energy by using just-enough quality.

the practice of bounding floating-point error is just as mature. Concerning the need for tools and approaches to provide an application-to-hardware solution to deploying an approximate computing system, numerical computing represents an extraordinarily thorough case study in high-overhead, manual approaches to deriving strong accuracy properties.

Transprecision Computing [189] is an emerging paradigm and a good example of application-to-hardware solution. Transprecision computing leverages approximation techniques in both hardware (e.g., due to intentionally degraded resiliency against PVT variations as in Section III-B) and software (e.g., due to computation with limited precision). In addition, it focuses on fine-grained control of approximation in space and time through multiple hardware and software EQ knobs. This enables substantial energy savings, compared to more traditional approximate computing systems designed for a fixed quality, as discussed in Section III-A (see also Fig. 27). Indeed, transprecision computing does not deal with error tolerance, nor does it imply reduced precision at the application level. Instead, it aims at disrupting the conservative bounds (i.e., quality margin from Section III-A) of guaranteed numerical precision of each elementary step by designing systems that deliver just-enough quality at any point of time. In short, transprecision computing defines computing architectures that operate with a smooth and wide range of EQ tradeoffs at every compute scale, from ultra-low power platforms for the Internet-of-Things to large high-performance computing centers.

Memory technology plays a central role in transprecision computing. As DRAMs largely contribute to the system power consumption and performance, several techniques have been explored to control approximation at the off-chip memory level [190]. The DRAM retention error behavior, data lifetimes and application robustness are all jointly considered to allow for controlled lowering (or even complete switch-off) of the DRAM refresh rate (see, e.g., [46] in Table II). For a given application-specific access pattern, the memory controller can be configured so that data are optimally mapped into the DRAM banks/ranks/channels, minimizing the row misses and maximizing the bandwidth and energy efficiency.

Another key aspect of transprecision computing is the emphasis on floating-point (FP) computation. The execution of FP operations emerges as a major contributor to the energy consumption of compute-intensive applications with large dynamic range, especially on ultra-low-power embedded platforms. Experimental evidence shows that 50% of the energy consumed by a core and its data memory is related to FP computations [150]. In this context, rather than tolerating errors implied by imprecise hardware or software components, the "just enough quality" requirement of transprecision computing is achieved by controlling bit-width reduction (i.e., approximation) at a fine grain through the computation steps. The adoption of FP formats with a reduced number of bits is a promising opportunity to reduce energy consumption, as not only it allows to simplify the arithmetic circuitry but combined with vectorization can also significantly reduce the memory bandwidth required for data transfers. On the other hand, the adoption of multiple FP types requires proper methodologies to understand which variables experience quality slack (see Section III-A) within an application.

The IEEE Standard for Floating-Point Arithmetic (IEEE 754) describes representation and memory encoding of five binary formats: binary16 (half-precision), binary32 (single-precision), binary64 (double-precision), binary128 (quadruple-precision) and binary256 (octuple-precision). Single-precision and double-precision formats correspond for example to `float` and `double` primitive types in the C language. Quad-precision is sometimes available as `long double`, while octuple-precision is rarely implemented, and smaller formats are typically not supported.

The adoption of smaller formats (16 bits or even less) offers significant potential to reduce the energy consumption of FP computations. Besides the related research effort, the adoption of smaller formats is a major industry trend, especially in the context of deep learning workloads (e.g., in General Purpose GPUS, GPGPUs), which run extremely efficiently on low-precision hardware [191], [192]. GPGPUs with smaller-than-32-bit FP types are now commercially available, supporting for example the binary16 type, and reduced precision types (e.g. mixed-precision multiply-and-add instructions).

An extended FP type system (the *smallFloat* formats) with complete hardware support to enable transprecision computing on low-power embedded processors has been preliminarily explored [193], [194]. This includes two standard IEEE formats (binary32, binary16), and two new formats (binary8, binary16alt). A FP arithmetic unit capable of basic operations on *smallFloat* formats, conversions among old and new FP types and SIMD-style vectorization have been added into the execution stage of a low-power 32-bit RISC-V processor core. Exploration of FP types is performed to find the mapping between program variables and available FP types that make the EQ tradeoff more favorable. The experimental results on several benchmarks for embedded systems show that up to 90% of the FP operations can be safely scaled down to 8-bit or 16-bit formats, which combined with code vectorization has the potential to greatly improve energy efficiency of both compute and memory operations.

Although these preliminary explorations show promising results, several research challenges lie ahead for the transprecision computing paradigm to successfully facilitate the design

of fine-grained EQ scalable systems. Focusing on floating-point calculus, the IEEE-standardized FP formats and the *smallFloat* formats represent only a small subset of all the possible bit-widths and mantissa-exponent bit allocations, and the exploration of custom formats for future HW platforms will need appropriate tools.

As a first challenge, the precision tuning process requires tedious and error-prone program instrumentation and parameter configuration stages. This is true for both the exploration of custom data types (i.e., for new hardware being designed) and for understanding which types (supported by the existing hardware) should be used for different program variables to maximize the EQ tradeoff for a given application. New compiler analysis/transformation passes can be designed, to automate this process and simplify the deployment of transprecision applications. Also, the specialization of well-established compiler optimizations for new data types will be crucial to enable major energy savings (e.g., vectorization).

As a second challenge, FP emulation is typically used for precision tuning, but state-of-the-art emulation libraries [195] were originally designed for different goals. New emulation tools need to be developed that are 1) fast enough to minimize the time required to complete iterative precision tuning; 2) accurate enough to correctly capture the accuracy that a custom HW type would deliver; 3) easy to augment, so that the definition of custom types does not require complex rewrites of the emulation library; 4) informative in terms of the achieved quality, as the convergence time of the search heuristics should be significantly reduced if better insight on the individual variables were propagated to the precision tuning tool.

As a third challenge, alternative approaches should be investigated to the emulation/execution of various program configurations that are less time consuming (e.g., FP data types used for different program variables). Since a single precision tuning process is entirely application- and often input dataset-dependent, moving to a new program will require to repeat the entire process. Machine learning approaches have the potential to simplify and quicken the process by replacing the execution of the program with emulated data types with that of a trained model.

IX. Conclusions

In this paper, energy-quality scalable circuits and systems have been discussed as promising direction to continue the historical exponential energy down-scaling, in spite of the significant Moore's law slowdown. EQ scaling offers new and interesting opportunities to reduce energy by leveraging the application and system noise resiliency, and taking advantage of the quality slack created by the conventional design margin, the application, the usage context, and the dataset. Fundamentally, EQ scaling is grounded on the elimination of artificial boundaries created by the levels of abstraction, and focuses on the application-level quality, rather than pessimistically margining at each level of abstraction.

The literature on energy-quality scalable systems has been reviewed by providing a unitary view, and bridging several independent sub-areas under a common framework. A taxonomy based on the available EQ knobs has been introduced. Several fundamental properties of EQ-scalable systems have been discussed, and exemplified by numerous silicon demonstrations. From a broad perspective, EQ scaling has been shown to be the natural generalization of existing techniques performing error detection and correction at run-time, and to be particularly promising in view of its lower overhead (since no full error coverage is needed, and quality control response does not need to be fast).

Quality control schemes and related examples have been introduced, and it has been shown that proper choice of EQ modes makes the energy minimization problem convex, and hence very simple to perform at run-time. The important role of graceful quality degradation has been discussed, as it ultimately dictates the limits of the achievable energy savings. Graceful degradation has been shown to be achievable through adaptation and selective techniques that adjust the energy-quality trade off down to the function, word and bit level, depending on the granularity of the adopted technique.

EQ scalability has been discussed via numerous case studies at the circuit, gate, architectural level. At the algorithm level, EQ scaling has been shown to be well suited for machine learning-based systems, in view of their inherent resiliency and the graceful dependence of the accuracy on memory size and amount of computation. At software level, the design implications of EQ scaling have been discussed from the point of view of language constructs, compiler and run-time tuning layers. Among the others, Transprecision Computing has been discussed as an approach that holistically optimizes precision across layers via automated control loops.

In perspective, EQ-scalable circuits and systems pose a wide range of challenges that demand further advances, before the potential energy gains are truly extracted. At the sub-system level, innovation is needed in terms of methods to inexpensively insert EQ knobs in all sub-systems, from sensors to analog, processing, power management, algorithms, and software. Versatile quality sensors need to be inexpensively inserted in such sub-systems, and more general methods to make the quality degradation graceful are needed. Novel design paradigms and optimization methodologies are needed to supervise the energy-quality control at run-time, based on an underlying application-to-hardware framework. Across-level design frameworks are also needed to extend the applicability of EQ scaling (e.g., to general-purpose platforms), spanning multiple levels of abstraction and sub-systems. Innovative frameworks and techniques are needed to minimize energy via true adaptation (e.g., on-chip learning), while keeping quality within bounds in a real-time and context-aware fashion. At the software level, new design methodologies, abstractions and compiler support are needed to enable structured and disciplined energy-quality scaling in both general-purpose and application-specific hardware. Highly inter-disciplinary efforts and approaches will be needed to address the above challenges, as they lie at the intersection of circuits, systems, solid-state circuits, CAD, architectures, machine learning and signal processing.

Overall, the general area of EQ scaling has made tremendous progress in the last few years, and has been shown to be able to reduce energy by an order of magnitude or more. Being a natural fit for machine learning, data-intensive workloads and ultra-low power sensing/processing/communications, EQ scaling is expected to play an even more important role in the future. Ultimately, EQ scaling will help extend to the upcoming decade the exponential savings that we have been used to see since the inception of Moore's law.

REFERENCES

[1] M. Alioto, "Guest editorial for the special issue on ultra-low-voltage VLSI circuits and systems for green computing," *IEEE Trans. Circuits Syst., II, Exp. Briefs*, vol. 59, no. 12, pp. 849–852, Dec. 2012.

[2] M. Alioto, "Energy-quality scalable adaptive VLSI circuits and systems beyond approximate computing," in *Proc. IEEE DATE*, Lausanne, Switzerland, Mar. 2017, pp. 127–132.

[3] L. A. Barroso and U. Holzle, "The case for energy-proportional computing," *Computer*, vol. 40, no. 12, pp. 33–37, Dec. 2007.

[4] K. T. Malladi, F. A. Nothaft, K. Periyathambi, B. C. Lee, C. Kozyrakis, and M. Horowitz, "Towards energy-proportional datacenter memory with mobile DRAM," in *Proc. ISCA*, Portland, OR, USA, Jun. 2012, pp. 37–48.

[5] K. Bergman *et al.*, "Exascale computing study: Technology challenges in achieving exascale systems," Defense Adv. Res. Projects Agency Inf. Process. Techn. Office (DARPA IPTO), Tech. Rep., 2008.

[6] J. Koomey, S. Berard, M. Sanchez, and H. Wong, "Implications of historical trends in the electrical efficiency of computing," *IEEE Ann. Hist. Comput.*, vol. 33, no. 3, pp. 46–54, Mar. 2011.

[7] Y. Guo and Y. Fang, "Electricity cost saving strategy in data centers by using energy storage," *IEEE Trans. Parallel Distrib. Syst.*, vol. 24, no. 6, pp. 1149–1160, Jun. 2013.

[8] S. Tarkoma, M. Siekkinen, E. Lagerspetz, and Y. Xiao, *Smartphone Energy Consumption: Modeling and Optimization*. Cambridge, U.K.: Cambridge Univ. Press, 2014.

[9] M. Halpern, Y. Zhu, and V. J. Reddi, "Mobile CPU's rise to power: Quantifying the impact of generational mobile CPU design trends on performance, energy, and user satisfaction," in *Proc. HPCA*, Barcelona, Spain, Mar. 2016, pp. 64–76.

[10] M. Alioto, E. Sánchez-Sinencio, and A. Sangiovanni-Vincentelli, "Guest editorial special issue on circuits and systems for the Internet of Things—From sensing to sensemaking," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 64, no. 9, pp. 2221–2225, Sep. 2017.

[11] J. M. Rabaey, "The swarm at the edge of the cloud—A new perspective on wireless," in *Symp. VLSI Circuits, Dig. Tech. Papers*, Kyoto, Japan, Jun. 2011, pp. 6–8.

[12] S. Borkar, "Truths and Myths of Embedded Computing," in *Proc. 48th Design Automat. Conf. (DAC)*, San Diego, CA, USA, Jun. 2011.

[13] M. Alioto, Ed., *Enabling the Internet of Things: From Integrated Circuits to Integrated Systems*. Cham, Switzerland: Springer, 2017.

[14] L. Karam, I. Alkamal, A. Gatherer, G. A. Frantz, D. V. Anderson, and B. L. Evans, "Trends in multicore DSP platforms," *IEEE Signal Process. Mag.*, vol. 26, no. 6, pp. 38–49, Nov. 2009.

[15] (2015). *International Technology Roadmap for Semiconductors 2.0*. [Online]. Available: http://www.itrs2.net/

[16] *GREEN500 List*. Accessed: Oct. 15, 2018. [Online]. Available: https://www.top500.org/green500/

[17] *TOP500 List*. Accessed: Oct. 15, 2018. [Online]. Available: https://www.top500.org/lists/top500/

[18] B. Egilmez, G. Memik, S. Ogrenci-Memik, and O. Ergin, "User-specific skin temperature-aware DVFS for smartphones," in *Proc. DATE*, Grenoble, France, Mar. 2015, pp. 1217–1220.

[19] B. Murmann. *ADC Performance Survey 1997-2018*. Accessed: Oct. 15, 2018. [Online]. Available: http://web.stanford.edu/~murmann/adcsurvey.html

[20] D. D. Wentzloff. Low Power Radio Survey. [Online]. Available: www.eecs.umich.edu/wics/low_power_radio_survey.html

[21] P. Harpe, A. Baschirotto, and K. A. A. Makinwa, Eds., *High-Performance AD and DA Converters, IC Design in Scaled Technologies, and Time-Domain Signal Processing*. Cham, Switzerland: Springer, 2015.

[22] K. Okada and S. Kousai, Eds., *Digitally-Assisted Analog and RF CMOS Circuit Design for Software-Defined Radio*. Cham, Switzerland: Springer, 2011.

[23] L. Ceze and J. Larus. *Report on ISAT/DARPA Workshop on Accuracy Trade-Offs Across the System Stack for Performance and Energy*. Accessed: Oct. 15, 2018. [Online]. Available: http://homes.cs.washington.edu/~luisceze/publications/ISAT_Accuracy_Trade-Offs_Across_the_System_Stack_for_Performance_and_Energy-PUBLIC-final.pptx

[24] R. G. Dreslinski, M. Wieckowski, D. Blaauw, D. Sylvester, and T. Mudge, "Near-threshold computing: Reclaiming Moore's law through energy efficient integrated circuits," *Proc. IEEE*, vol. 98, no. 2, pp. 253–266, Feb. 2010.

[25] M. Alioto, V. De, and A. Marongiu, "Guest editorial energy-quality scalable circuits and systems for sensing and computing: From approximate to communication-inspired and learning-based," *IEEE J. Emerg. Sel. Topics Circuits Syst.*, vol. 8, no. 3, pp. 361–368, Sep. 2018.

[26] C. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2006.

[27] S. Winkler and P. Mohandas, "The evolution of video quality measurement: From PSNR to hybrid metrics," *IEEE Trans. Broadcast.*, vol. 54, no. 3, pp. 660–668, Sep. 2008.

[28] D. M. W. Powers, "Evaluation: From precision, recall and F-measure to ROC, informedness, markedness & correlation," *J. Mach. Learn. Technol.*, vol. 2, no. 1, pp. 37–63, Jan. 2011.

[29] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: From error visibility to structural similarity," *IEEE Trans. Image Process.*, vol. 13, no. 4, pp. 600–612, Apr. 2004.

[30] V. K. Madisetti, *The Digital Signal Processing Handbook*, 2nd ed. Boca Raton, FL, USA: CRC Press, 2009.

[31] M. C. Kerman, W. Jiang, A. F. Blumberg, and S. E. Buttrey, "Event detection challenges, methods, and applications in natural and artificial systems," Defense Tech. Inf. Center Paper, Tech. Rep., Mar. 2009. [Online]. Available: http://www.dtic.mil/dtic/tr/fulltext/u2/a503477.pdf

[32] R. Szeliski, *Computer Vision: Algorithms and Applications*. Springer, 2011.

[33] M. Nelson and J.-L. Gailly, *The Data Compression Book*, 2nd ed. New York, NY, USA: MIS Press, 1996.

[34] Y. C. Eldar and G. Jutyniok, Eds., *Compressed Sensing: Theory and Applications*. Cambridge, U.K.: Cambridge Univ. Press, 2012.

[35] A. Menezes, P. van Oorschot, and S. Vanstone, *Handbook of Applied Cryptography*. Boca Raton, FL, USA: CRC Press, 1996.

[36] T. Addabbo, M. Alioto, A. Fort, S. Rocchi, and V. Vignoli, "A feedback strategy to improve the entropy of a chaos-based random bit generator," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 53, no. 2, pp. 326–337, Feb. 2006.

[37] R. Maes, *Physically Unclonable Functions: Construction, Properties and Applications*. Cham, Switzerland: Springer, 2013.

[38] L. Čehovin, A. Leonardis, and M. Kristan, "Visual object tracking performance measures revisited," *IEEE Trans. Image Process.*, vol. 25, no. 3, pp. 1261–1274, Mar. 2016.

[39] K. P. Murphy, *Machine Learning: A Probabilistic Perspective*. Cambridge, MA, USA: MIT Press, 2012.

[40] Q. Xu, M. Todd, and S. K. Nam, "Approximate computing: A survey," *IEEE Design Test*, vol. 33, no. 1, pp. 8–22, Feb. 2016.

[41] R. J. van de Plassche, *CMOS Integrated Analog-to-Digital and Digital-to-Analog Converters*, 2nd ed. Cham, Switzerland: Springer, 2003.

[42] P. Le Callet and D. Barba, "A robust quality metric for color image quality assessment," in *Proc. IEEE Int. Conf. Image Process.*, Sep. 2003, pp. 437–440.

[43] J. Xu, P. Harpe, and C. Van Hoof, "An energy-efficient and reconfigurable sensor IC for bio-impedance spectroscopy and ECG recording," *IEEE J. Emerg. Sel. Topics Circuits Syst.*, vol. 8, no. 3, pp. 616–626, Sep. 2018.

[44] L. Freyman, D. Fick, M. Alioto, D. Blaauw, and D. Sylvester, "A 346μm^2 VCO-based, reference-free, self-timed sensor interface for cubic-millimeter sensor nodes in 28 nm CMOS," *IEEE J. Solid-State Circuits*, vol. 49, no. 11, pp. 2462–2473, Nov. 2014.

[45] F. Frustaci, M. Khayatzadeh, D. Blaauw, D. Sylvester, and M. Alioto, "SRAM for error-tolerant applications with dynamic energy-quality management in 28 nm CMOS," *IEEE J. Solid-State Circuits*, vol. 50, no. 5, pp. 1310–1323, May 2015.

[46] R. Giterman, A. Fish, N. Geuli, E. Mentovich, A. Burg, and A. Teman, "An 800-MHz mixed-V_T 4T IFGC embedded DRAM in 28-nm CMOS bulk process for approximate storage applications," *IEEE J. Solid-State Circuits*, vol. 53, no. 7, pp. 2136–2148, Jul. 2018.

[47] H. Farkhani, M. Tohidi, S. Farkhani, J. K. Madsen, and F. Moradi, "A low-power high-speed spintronics-based neuromorphic computing system using real-time tracking method," *IEEE J. Emerg. Sel. Topics Circuits Syst.*, vol. 8, no. 3, pp. 627–638, Sep. 2018.

[48] B. Zimmer *et al.*, "A RISC-V vector processor with simultaneous-switching switched-capacitor DC-DC converters in 28 nm FDSOI," *IEEE J. Solid-State Circuits*, vol. 51, no. 4, pp. 930–942, Apr. 2016.

[49] R. Boyapati, J. Huang, P. Majumder, K. H. Yum, and E. J. Kim, "APPROX-NoC: A data approximation framework for network-on-chip architectures," in *Proc. ISCA*, Toronto, ON, Canada, Jun. 2017, pp. 666–677.

[50] G. Chang, S. Maity, B. Chatterjee, and S. Sen, "A medradio receiver front-end with wide energy-quality scalability through circuit and architecture-level reconfigurations," *IEEE J. Emerg. Sel. Topics Circuits Syst.*, vol. 8, no. 3, pp. 369–378, Sep. 2018.

[51] M. N. Aman, S. Taneja, B. Sikdar, K. C. Chua, and M. Alioto, "Token-based security for the Internet of Things with dynamic energy-quality tradeoff," *IEEE Internet Things J.*, to be published.

[52] S. Behroozi, V. Raghunathan, A. Raghunathan, and Y. Kim, "A quality-configurable approximate serial bus for energy-efficient sensory data transfer," *IEEE J. Emerg. Sel. Topics Circuits Syst.*, vol. 8, no. 3, pp. 379–390, Sep. 2018.

[53] A. Mohammed and A. Demosthenous, "Complementary detection for hardware efficient on-site monitoring of parkinsonian progress," *IEEE J. Emerg. Sel. Topics Circuits Syst.*, vol. 8, no. 3, pp. 603–615, Sep. 2018.

[54] J. W. Wells and A. Chatterjee, "Content-aware low-complexity object detection for tracking using adaptive compressed sensing," *IEEE J. Emerg. Sel. Topics Circuits Syst.*, vol. 8, no. 3, pp. 578–590, Sep. 2018.

[55] G. Rovere, S. Fateh, and L. Benini, "A 2.2-μW cognitive always-on wake-up circuit for event-driven duty-cycling of IoT sensor nodes," *IEEE J. Emerg. Sel. Topics Circuits Syst.*, vol. 8, no. 3, pp. 543–554, Sep. 2018.

[56] A. Sinha and A. P. Chandrakasan, "Energy efficient filtering using adaptive precision and variable voltage," in *Proc. ASIC/SOC*, Sep. 1999, pp. 327–331.

[57] R. Hegde and N. R. Shanbhag, "Energy-efficient signal processing via algorithmic noise-tolerance," in *Proc. ISLPED*, San Diego, CA, USA, Aug. 1999, pp. 30–35.

[58] V. Ariyarathna *et al.*, "Analog approximate-FFT 8/16-beam algorithms, architectures and CMOS circuits for 5G beamforming MIMO transceivers," *IEEE J. Emerg. Sel. Topics Circuits Syst.*, vol. 8, no. 3, pp. 466–479, Sep. 2018.

[59] J. Park, J. Choi, and K. Roy, "Dynamic bit-width adaptation in DCT: An approach to trade off image quality and computation energy," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 18, no. 5, pp. 787–793, May 2010.

[60] A. B. Alvarez, G. Ponnusamy, and M. Alioto, "EQSCALE: Energy-quality scalable feature extraction engine for Sub-mW real-time video processing with 0.55 mm^2 area in 40 nm CMOS," in *Proc. A-SSCC*, Seoul, South Korea, Nov. 2017, pp. 241–244.

[61] A. B. Alvarez, G. Ponnusamy, and M. Alioto, "A sub-mW, sub-mm^2 energy-quality scalable feature extraction accelerator for distributed vision in 40 nm," *IEEE J. Solid-State Circuits*, to be published.

[62] V. K. Chippa, D. Mohapatra, A. Raghunathan, K. Roy, and S. T. Chakradhar, "Scalable effort hardware design: Exploiting algorithmic resilience for energy efficiency," in *Proc. DAC*, Jun. 2010, pp. 555–560.

[63] S. Han, H. Mao, and W. J. Dally. (Feb. 2016). "Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding." [Online]. Available: https://arxiv.org/abs/1510.00149

[64] D. L. Donoho, "Compressed sensing," *IEEE Trans. Inf. Theory*, vol. 52, no. 4, pp. 1289–1306, Apr. 2006.

[65] W. J. Poppelbaum, C. Afuso, and J. W. Esch, "Stochastic computing elements and systems," in *Proc. AFIPS Fall Joint Comput. Conf.*, 1967, pp. 635–644.

[66] S. H. Kim, S. Mukhopadhyay, and M. Wolf, "Modeling and analysis of image dependence and its implications for energy savings in error tolerant image processing," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 30, no. 8, pp. 1163–1172, Aug. 2011.

[67] S.-L. Lu, "Speeding up processing with approximation circuits," *Computer*, vol. 37, no. 3, pp. 67–73, Mar. 2004.

[68] J. George, B. Marr, B. E. S. Akgul, and K. V. Palem, "Probabilistic arithmetic and energy efficient embedded signal processing," in *Proc. CASES*, 2006, pp. 158–168.

[69] A. K. Verma, P. Brisk, and P. Ienne, "Variable latency speculative addition: A new paradigm for arithmetic circuit design," in *Proc. DATE*, 2008, pp. 1250–1255.

[70] N. Zhu, W. L. Goh, and K. S. Yeo, "An enhanced low-power high-speed Adder For Error-Tolerant application," in *Proc. ISIC*, Dec. 2009, pp. 69–72.

[71] N. Zhu, W. L. Goh, W. Zhang, K. S. Yeo, and Z. H. Kong, "Design of low-power high-speed truncation-error-tolerant adder and its application in digital signal processing," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 18, no. 8, pp. 1225–1229, Aug. 2010.

[72] N. Zhu, W. L. Goh, and K. S. Yeo, "Ultra low-power high-speed flexible probabilistic adder for error-tolerant applications," in *Proc. Int. SoC Design Conf.*, Nov. 2011, pp. 393–396.

[73] K. Du, P. Varman, and K. Mohanram, "High performance reliable variable latency carry select addition," in *Proc. DATE*, Mar. 2012, pp. 1257–1262.

[74] A. B. Kahng and S. Kang, "Accuracy-configurable adder for approximate arithmetic designs," in *Proc. DAC*, San Francisco, CA, USA, Jun. 2012, pp. 820–825.

[75] J. Miao, K. He, A. Gerstlauer, and M. Orshansky, "Modeling and synthesis of quality-energy optimal approximate adders," in *Proc. ICCAD*, 2012, pp. 728–735.

[76] Y. Kim, Y. Zhang, and P. Li, "Energy efficient approximate arithmetic for error resilient neuromorphic computing," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 23, no. 11, pp. 2733–2737, Nov. 2015.

[77] H. A. F. Almurib, T. N. Kumar, and F. Lombardi, "Inexact designs for approximate low power addition by cell replacement," in *Proc. DATE*, Dresden, Germany, Mar. 2016, pp. 660–665.

[78] H. R. Mahdiani, A. Ahmadi, S. M. Fakhraie, and C. Lucas, "Bio-inspired imprecise computational blocks for efficient VLSI implementation of soft-computing applications," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 57, no. 4, pp. 850–862, Apr. 2010.

[79] K. Y. Kyaw, W. L. Goh, and K. S. Yeo, "Low-power high-speed multiplier for error-tolerant application," in *Proc. IEEE Intl. Conf. Electron Devices Solid-State Circuits*, Dec. 2010, pp. 1–4.

[80] P. Kulkarni, P. Gupta, and M. Ercegovac, "Trading accuracy for power with an underdesigned multiplier architecture," in *Proc. 24th Int. Conf. VLSI Design*, Jan. 2011, pp. 346–351.

[81] C. Liu, J. Han, and F. Lombardi, "A low-power, high-performance approximate multiplier with configurable partial error recovery," in *Proc. DATE*, Mar. 2014, pp. 1–4.

[82] S. Narayanamoorthy, H. A. Moghaddam, Z. Liu, T. Park, and S. Kim, "Energy-efficient approximate multiplication for digital signal processing and classification applications," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 23, no. 6, pp. 1180–1184, Jun. 2015.

[83] Z. Babić, A. Avramović, and P. Bulić, "An iterative logarithmic multiplier," *Microprocess. Microsyst.*, vol. 35, no. 1, pp. 23–33, Feb. 2011.

[84] H. Kaul *et al.*, "A 1.45 GHz 52-to-162GFLOPS/W variable-precision floating-point fused multiply-add unit with certainty tracking in 32 nm CMOS," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, San Francisco, CA, USA, Feb. 2012, pp. 182–184.

[85] M. J. Schulte and E. E. Swartzlander, "A family of variable-precision interval arithmetic processors," *IEEE Trans. Comput.*, vol. 49, no. 5, pp. 387–397, May 2000.

[86] M. R. Choudhury and K. Mohanram, "Approximate logic circuits for low overhead, non-intrusive concurrent error detection," in *Proc. DATE*, 2008, pp. 903–908.

[87] D. Shin and S. K. Gupta, "Approximate logic synthesis for error tolerant applications," in *Proc. DATE*, Mar. 2010, pp. 957–960.

[88] A. Lingamneni, C. Enz, J.-L. Nagel, K. Palem, and C. Piguet, "Energy parsimonious circuit design through probabilistic pruning," in *Proc. DATE*, May 2011, pp. 1–6.

[89] S. Venkataramani, K. Roy, and A. Raghunathan, "Substitute-and-simplify: A unified design paradigm for approximate and quality configurable circuits," in *Proc. DATE*, Mar. 2013, pp. 1367–1372.

[90] J. Miao, A. Gerstlauer, and M. Orshansky, "Approximate logic synthesis under general error magnitude and frequency constraints," in *Proc. DATE*, Nov. 2013, pp. 1–6.

[91] A. Ranjan, A. Raha, S. Venkataramani, K. Roy, and A. Raghunathan, "ASLAN: Synthesis of approximate sequential circuits," in *Proc. DAC*, 2012, Art. no. 364.

[92] S. Venkataramani, A. Sabne, V. Kozhikkottu, K. Roy, and A. Raghunathan, "SALSA: Systematic logic synthesis of approximate circuits," in *Proc. DAC*, Jun. 2012, pp. 796–801.

[93] K. Ueyoshi *et al.*, "QUEST: A 7.49TOPS multi-purpose log-quantized DNN inference engine stacked on 96 MB 3D SRAM using inductive-coupling technology in 40 nm CMOS," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2018, pp. 216–217.

[94] B. Moons, R. Uytterhoeven, W. Dehaene, and M. Verhelst, "ENVISION: A 0.26-to-10TOPS/W subword-parallel dynamic-voltage-accuracy-frequency-scalable convolutional neural network processor in 28 nm FDSOI," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2017, pp. 246–247.

[95] D. Shin, J. Lee, J. Lee, and H.-J. Yoo, "DNPU: An 8.1TOPS/W reconfigurable CNN-RNN processor for general-purpose deep neural networks," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2017, pp. 240–241.

[96] G. Desoli *et al.*, "A 2.9TOPS/W deep convolutional neural network SoC in FD-SOI 28nm for intelligent embedded systems," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2017, pp. 238–239.

[97] B. Moons and M. Verhelst, "A 0.3–2.6 TOPS/W precision-scalable processor for real-time large-scale ConvNets," in *Proc. VLSI Symp.*, Jun. 2016, pp. 1–2.

[98] Y.-H. Chen, T. Krishna, J. Emer, and V. Sze, "Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Jan. 2016, pp. 262–263.

[99] S. Han *et al.* (2016). "EIE: Efficient inference engine on compressed deep neural network." [Online]. Available: https://arxiv.org/abs/1602.01528

[100] J. T. Ludwig, S. H. Nawab, and A. P. Chandrakasan, "Low-power digital filtering using approximate processing," *IEEE J. Solid-State Circuits*, vol. 31, no. 3, pp. 395–400, Mar. 1996.

[101] S. Sidiroglou-Douskos, S. Misailovic, H. Hoffmann, and M. Rinard, "Managing performance vs. Accuracy trade-offs with loop perforation," in *Proc. ACM SIGSOFT Symp. 13th Eur. Conf. Found. Softw. Eng.*, 2011, pp. 124–134.

[102] R. Mangharam and A. A. Saba, "Anytime algorithms for GPU architectures," in *Proc. IEEE 32nd Real-Time Syst. Symp.*, Nov. 2011, pp. 47–56.

[103] K. A. Bowman *et al.*, "Energy-efficient and metastability-immune resilient circuits for dynamic variation tolerance," *IEEE J. Solid-State Circuits*, vol. 44, no. 1, pp. 49–63, Jan. 2009.

[104] Y. Zhang *et al.*, "iRazor: 3-transistor current-based error detection and correction in an ARM cortex-R4 processor," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2016, pp. 160–162.

[105] D. Ernst *et al.*, "Razor: A low-power pipeline based on circuit-level timing speculation," in *Proc. MICRO-36*, Dec. 2003, pp. 7–18.

[106] S. Das *et al.*, "RazorII: *In situ* error detection and correction for PVT and SER tolerance," *IEEE J. Solid-State Circuits*, vol. 44, no. 1, pp. 32–48, Jan. 2009.

[107] M. Khayatzadeh, M. Saligane, J. Wang, M. Alioto, D. Blaauw, and D. Sylvester, "A reconfigurable dual-port memory with error detection and correction in 28nm FDSOI," in *IEEE IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Jan./Feb. 2016, pp. 310–311.

[108] M. Alioto, "Ultra-low power VLSI circuit design demystified and explained: A tutorial," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 59, no. 1, pp. 3–29, Jan. 2012.

[109] U. Parlitz and L. Junge, "Synchronization of chaotic systems," in *Proc. Eur. Control Conf. (ECC)*, Karlsruhe, Germany, Aug./Sep. 1999, pp. 4637–4642.

[110] T. L. Carroll, "Chaotic systems that are robust to added noise," *AIP Chaos*, vol. 15, no. 1, p. 013901, 2005.

[111] J. Preskill. (2018). "Quantum computing in the NISQ era and beyond." [Online]. Available: https://arxiv.org/abs/1801.00862

[112] K. Moskvitch, "The Argument Against Quantum Computers," *Quantamagazine*, Feb. 2018. [Online]. Available: https://www.quantamagazine.org/gil-kalais-argument-against-quantum-computers-20180207/

[113] I. Jolliffe, *Principal Component Analysis, in International Encyclopedia of Statistical Science*. Cham, Switzerland: Springer, 2011.

[114] N. Weste and D. Harris, *CMOS VLSI Design: A Circuits and Systems Perspective*, 4th ed. Reading, MA, USA: Addison-Wesley, 2011.

[115] F. Frustaci, D. Blaauw, D. Sylvester, and M. Alioto, "Approximate SRAMs with dynamic energy-quality management," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 24, no. 6, pp. 2128–2141, Jun. 2016.

[116] R. Hegde and N. R. Shanbhag, "Soft digital signal processing," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 9, no. 6, pp. 813–823, Jun. 2001.

[117] R. Hegde and N. R. Shanbhag, "A voltage overscaled low-power digital filter IC," *IEEE J. Solid-State Circuits*, vol. 39, no. 2, pp. 388–391, Feb. 2004.

[118] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge, U.K.: Cambridge Univ. Press, 2004.

[119] D. S. Khudia, B. Zamirai, M. Samadi, and S. Mahlke, "Rumba: An online quality management system for approximate computing," in *Proc. ISCA*, Portland, OR, USA, Jun. 2015, pp. 554–566.

[120] D. Mohapatra, G. Karakonstantis, and K. Roy, "Significance driven computation: A voltage-scalable, variation-aware, quality-tuning motion estimator," in *Proc. ISLPED*, Aug. 2009, pp. 195–200.

[121] V. Chippa, A. Raghunathan, K. Roy, and S. Chakradhar, "Dynamic effort scaling: Managing the quality-efficiency tradeoff," in *Proc. DAC*, New York, NY, USA, Jun. 2011, pp. 603–608.

[122] B. R. Gaines, "Stochastic computing systems," in *Advances in Information Systems Science*, J. T. Tou, Eds. Boston, MA, USA: Springer, 1969.

[123] A. Alaghi and J. P. Hayes, "Survey of stochastic computing," *ACM Trans. Embedded Comput. Syst. Special Sect. Probabilistic Embedded Comput.*, vol. 12, no. 2s, May 2013, Art. no. 92.

[124] S. Cheemalavagu, P. Korkmaz, K. V. Palem, B. E. S. Akgul, and L. N. Chakrapani, "A probabilistic CMOS switch and its realization by exploiting noise," in *Proc. IFIP-VLSI SoC*, Oct. 2005, pp. 452–457.

[125] M. Davies *et al.*, "Loihi: A neuromorphic manycore processor with on-chip learning," *IEEE Micro*, vol. 38, no. 1, pp. 82–99, Jan. 2018.

[126] K. Nii *et al.*, "A 45-nm single-port and dual-port SRAM family with robust read/write stabilizing circuitry under DVFS environment," in *Symp. VLSI Circuits Dig. Tech. Papers*, Jun. 2008, pp. 212–213.

[127] J. Chen, "Self-calibrating on-chip interconnects," Ph.D. dissertation, Dept. Elect. Eng., Stanford Univ., Stanford, CA, USA, 2012.

[128] I. J. Chang, D. Mohapatra, and K. Roy, "A priority-based 6T/8T hybrid SRAM architecture for aggressive voltage scaling in video applications," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 21, no. 2, pp. 101–112, Feb. 2011.

[129] N. Gong, S. Jiang, A. Challapalli, S. Fernandes, and R. Sridhar, "Ultra-low voltage split-data-aware embedded SRAM for mobile video applications," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 59, no. 12, pp. 883–887, Dec. 2012.

[130] M. Cho, J. Schlessman, W. Wolf, and S. Mukhopadhyay, "Reconfigurable SRAM architecture with spatial voltage scaling for low power mobile multimedia applications," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 19, no. 1, pp. 161–165, Jan. 2011.

[131] J. Kwon, I. J. Chang, I. Lee, H. Park, and J. Park, "Heterogeneous SRAM cell sizing for low-power H.264 applications," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 59, no. 10, pp. 2275–2284, Oct. 2012.

[132] W. B. Pennebaker and J. L. Mitchell, *JPEG: Still Image Data Compression Standard*. Norwell, MA, USA: Kluwer, 1993.

[133] *Advanced Video Coding for Generic Audiovisual Services*, document H.264, ITU publication, 2017. [Online]. Available: https://www.itu.int/rec/T-REC-H.264

[134] *High Efficiency Video Coding*, document H.265, ITU publication, 2018. [Online]. Available: https://www.itu.int/rec/T-REC-H.265

[135] E. Nogues, D. Menard, and M. Pelcat, "Algorithmic-level approximate computing applied to energy efficient HEVC decoding," *IEEE Trans. Emerg. Topics Comput.*, to be published.

[136] F. J. Kurdahi, A. Eltawil, K. Yi, S. Cheng, and A. Khajeh, "Low-power multimedia system design by aggressive voltage scaling," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 18, no. 5, pp. 852–856, May 2010.

[137] M. A. Makhzan, A. Khajeh, A. Eltawil, and F. J. Kurdahi, "A low power JPEG2000 encoder with iterative and fault tolerant error concealment," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 17, no. 6, pp. 827–837, Jun. 2009.

[138] M. Yip and A. P. Chandrakasan, "A resolution-reconfigurable 5-to-10b 0.4-to-1V power scalable SAR ADC," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, San Francisco, CA, USA, Feb. 2011, pp. 310–311.

[139] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer. (2016). "SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size." [Online]. Available: https://arxiv.org/abs/1602.07360

[140] A. G. Howard *et al.* (2017). "MobileNets: efficient convolutional neural networks for mobile vision applications." [Online]. Available: https://arxiv.org/abs/1704.04861

[141] M. Courbariaux, I. Hubara, D. Soudry, R. El-Yaniv, and Y. Bengio. (Feb. 2016). "Binarized neural networks: Training deep neural networks with weights and activations constrained to +1 or -1." [Online]. Available: https://arxiv.org/abs/1602.02830

[142] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi. (Mar. 2016). "XNOR-Net: ImageNet classification using binary convolutional neural networks." [Online]. Available: https://arxiv.org/abs/1603.05279

[143] K. Ando *et al.*, "BRein memory: A single-chip binary/ternary reconfigurable in-memory deep neural network accelerator achieving 1.4 TOPS at 0.6 W," *IEEE J. Solid-State Circuits*, vol. 53, no. 4, pp. 983–994, Apr. 2018.

[144] M. De Kruijf, S. Nomura, and K. Sankaralingam, "Relax: An architectural framework for software recovery of hardware faults," in *Proc. 37th Annu. Int. Symp. Comput. Archit.*, Saint-Malo, France, 2010, pp. 497–508.

[145] A. Sampson, W. Dietl, E. Fortuna, D. Gnanapragasam, L. Ceze, and D. Grossman, "EnerJ: Approximate data types for safe and general low-power computation," in *Proc. 32nd ACM SIGPLAN Conf. Program. Lang. Design Implement.*, San Jose, CA, USA, 2011, pp. 164–174.

[146] A. Sampson *et al.*, "ACCEPT: A programmer-guided compiler framework for practical approximate computing," Univ. Washington, Tech. Rep., 2015.

[147] M. Carbin, S. Misailovic, and M. C. Rinard, "Verifying quantitative reliability for programs that execute on unreliable hardware," in *Proc. ACM SIGPLAN Int. Conf. Object Oriented Program. Syst. Lang. Appl.*, Indianapolis, IN, USA, 2013, pp. 33–52.

[148] M. Carbin, S. Misailovic, and M. C. Rinard, "Verifying quantitative reliability for programs that execute on unreliable hardware," *Commun. ACM*, vol. 59, no. 8, pp. 83–91, 2016.

[149] G. Tagliavini, D. Rossi, L. Benini, and A. Marongiu, "Synergistic architecture and programming model support for approximate micropower computing," in *Proc. IEEE Comput. Soc. Annu. Symp. VLSI*, Montpellier, France, Jul. 2015, pp. 280–285.

[150] G. Tagliavini, D. Rossi, A. Marongiu, and L. Benini, "Synergistic HW/SW approximation techniques for ultralow-power parallel computing," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 37, no. 5, pp. 982–995, May 2018.

[151] D. Papagiannopoulou, A. Marongiu, T. Moreshet, M. Herlihy, and R. I. Bahar, "Edge-TM: Exploiting transactional memory for error tolerance and energy efficiency," *ACM Trans. Embedded Comput. Syst.*, vol. 16, no. 5, 2017, Art. no. 153.

[152] D. Papagiannopoulou, A. Marongiu, T. Moreshet, L. Benini, M. Herlihy, and I. Bahar, "Playing with fire: Transactional memory revisited for error-resilient and energy-efficient MPSoC execution," in *Proc. 25th Ed. Great Lakes Symp. VLSI*, Pittsburgh, PA, USA, 2015, pp. 9–14.

[153] T. M. Aamodt and P. Chow, "Compile-time and instruction-set methods for improving floating- to fixed-point conversion accuracy," *ACM Trans. Embedded Comput. Syst.*, vol. 7, no. 3, 2008, Art. no. 26.

[154] M. O. Lam, J. K. Hollingsworth, B. R. De Supinski, and M. P. Legendre, "Automatically adapting programs for mixed-precision floating-point computation," in *Proc. 27th Int. ACM Conf. Supercomput.*, New York, NY, USA, 2013, pp. 369–378.

[155] R. Medhat, M. O. Lam, B. L. Rountree, B. Bonakdarpour, and S. Fischmeister, "Managing the performance/error tradeoff of floating-point intensive applications," *ACM Trans. Embedded Comput. Syst.*, vol. 16, no. 5s, 2017, Art. no. 184.

[156] N.-M. Ho, E. Manogaran, W.-F. Wong, and A. Anoosheh, "Efficient floating point precision tuning for approximate computing," in *Proc. 22nd Asia South Pacific Design Automat. Conf. (ASP-DAC)*, Chiba, Japan, Jan. 2017, pp. 63–68.

[157] S. Graillat, F. Jézéquel, R. Picot, F. Févotte, and B. Lathuilière, "PROMISE: Floating-point precision tuning with stochastic arithmetic," in *Proc. 17th Int. Symp. Sci. Comput., Comput. Arithmetic Verified Numerics*, Uppsala, Sweden, 2016, pp. 98–156.

[158] C. Rubio-González *et al.*, "Precimonious: Tuning assistant for floating-point precision," in *Proc. Int. Conf. High Perform. Comput., Netw., Storage Anal.*, Denver, CO, USA, Nov. 2013, pp. 1–12.

[159] J. Meng, S. Chakradhar, and A. Raghunathan, "Best-effort parallel execution framework for recognition and mining applications," in *Proc. IEEE Int. Symp. Parallel Distrib. Process.*, Rome, Italy, May 2009, pp. 1–12.

[160] M. Rinard, "Probabilistic accuracy bounds for fault-tolerant computations that discard tasks," in *Proc. 20th Annu. Int. Conf. Supercomput.*, Cairns, QL, Australia, 2006, pp. 324–334.

[161] S. Misailovic, S. Sidiroglou, H. Hoffmann, and M. Rinard, "Quality of service profiling," in *Proc. 32nd ACM/IEEE Int. Conf. Softw. Eng.*, vol. 1, Cape Town, South Africa, May 2010, pp. 25–34.

[162] S. Misailovic, D. M. Roy, and M. C. Rinard, "Probabilistically accurate program transformations," in *Proc. 18th Int. Conf. Static Anal.*, Venice, Italy, 2011, pp. 316–333.

[163] D. Maier, B. Cosenza, and B. Juurlink, "Local memory-aware kernel perforation," in *Proc. Int. Symp. Code Gener. Optim.*, Vienna, Austria, 2018, pp. 278–287.

[164] L. Renganarayana, V. Srinivasan, R. Nair, and D. Prener, "Programming with relaxed synchronization," in *Proc. ACM Workshop Relaxing Synchronization Multicore Manycore Scalability*, Tucson, AZ, USA, 2012, pp. 41–50.

[165] S. Campanoni, G. Holloway, G.-Y. Wei, and D. Brooks, "HELIX-UP: Relaxing program semantics to unleash parallelization," in *Proc. 13th Annu. IEEE/ACM Int. Symp. Code Gener. Optim.*, San Francisco, CA, USA, 2015, pp. 235–245.

[166] M. Herlihy and J. E. B. Moss, "Transactional memory: Architectural support for lock-free data structures," in *Proc. 20th Annu. Int. Symp. Comput. Archit.*, San Diego, CA, USA, 1993, pp. 289–300.

[167] R. Rajwar and J. R. Goodman, "Speculative lock elision: Enabling highly concurrent multithreaded execution," in *Proc. 34th Annu. ACM/IEEE Int. Symp. Microarchit.*, Austin, TX, USA, Dec. 2001, pp. 294–305.

[168] S. Misailovic, D. Kim, and M. Rinard, "Parallelizing sequential programs with statistical accuracy tests," *ACM Trans. Embedded Comput. Syst.*, vol. 12, no. 2, 2013, Art. no. 88.

[169] M. Samadi, D. A. Jamshidi, J. Lee, and S. Mahlke, "Paraprox: Pattern-based approximation for data parallel applications," in *Proc. 19th Int. Conf. Archit. Support Program. Lang. Operating Syst.*, Salt Lake City, UT, USA, 2014, pp. 35–50.

[170] Z. A. Zhu, S. Misailovic, J. A. Kelner, and M. Rinard, "Randomized accuracy-aware program transformations for efficient approximate computations," in *Proc. 39th Annu. ACM SIGPLAN-SIGACT Symp. Princ. Program. Lang.*, Philadelphia, PA, USA, 2012, pp. 441–454.

[171] C. Alvarez, J. Corbal, and M. Valero, "Fuzzy memoization for floating-point multimedia applications," *IEEE Trans. Comput.*, vol. 54, no. 7, pp. 922–927, Jul. 2005.

[172] S. Chaudhuri, S. Gulwani, R. Lublinerman, and S. Navidpour, "Proving Programs Robust," in *Proc. 19th ACM SIGSOFT Symp., 13th Eur. Conf. Found. Softw. Eng.*, Szeged, Hungary, 2011, pp. 102–112.

[173] J. Ansel *et al.*, "PetaBricks: A language and compiler for algorithmic choice," in *Proc. 30th ACM SIGPLAN Conf. Program. Lang. Design Implement.*, Dublin, Ireland, 2009, pp. 38–49.

[174] J. Ansel, Y. L. Wong, C. Chan, M. Olszewski, A. Edelman, and S. Amarasinghe, "Language and compiler support for auto-tuning variable-accuracy algorithms," in *Proc. 9th Annu. IEEE/ACM Int. Symp. Code Gener. Optim.*, Apr. 2011, pp. 85–96.

[175] W. Baek and T. M. Chilimbi, "Green: A framework for supporting energy-conscious programming using controlled approximation," in *Proc. 31st ACM SIGPLAN Conf. Program. Lang. Design Implement.*, Toronto, ON, Canada, 2010, pp. 198–209.

[176] J. Sorber, A. Kostadinov, M. Garber, M. Brennan, M. D. Corner, and E. D. Berger, "Eon: A language and runtime system for perpetual systems," in *Proc. 5th Int. Conf. Embedded Netw. Sensor Syst.*, Sydney, NSW, Australia, 2007, pp. 161–174.

[177] F. Chang and V. Karamcheti, "A framework for automatic adaptation of tunable distributed applications," *Cluster Comput.*, vol. 4, no. 1, pp. 49–62, 2001.

[178] V. Vassiliadis *et al.*, "A significance-driven programming framework for energy-constrained approximate computing," in *Proc. 12th ACM Int. Conf. Comput. Frontiers*, Ischia, Italy, 2015, Art. no. 9.

[179] A. Rahimi, D. Cesarini, A. Marongiu, R. K. Gupta, and L. Benini, "Improving resilience to timing errors by exposing variability effects to software in tightly-coupled processor clusters," *IEEE J. Emerg. Sel. Topics Circuits Syst.*, vol. 4, no. 2, pp. 216–229, Jun. 2014.

[180] A. Rahimi, A. Marongiu, P. Burgio, R. K. Gupta, and L. Benini, "Variation-tolerant OpenMP tasking on tightly-coupled processor clusters," in *Proc. Conf. Design, Automat. Test Eur.*, Grenoble, France, Mar. 2013, pp. 541–546.

[181] A. Rahimi, D. Cesarini, A. Marongiu, R. K. Gupta, and L. Benini, "Task scheduling strategies to mitigate hardware variability in embedded shared memory clusters," in *Proc. 52nd Annu. Design Automat. Conf.*, San Francisco, CA, USA, 2015, Art. no. 152.

[182] A. Rahimi, A. Marongiu, R. K. Gupta, and L. Benini, "A variability-aware OpenMP environment for efficient execution of accuracy-configurable computation on shared-FPU processor clusters," in *Proc. 9th IEEE/ACM/IFIP Int. Conf. Hardware/Softw. Codesign Syst. Synth.*, Montreal, QC, Canada, Sep./Oct. 2013, pp. 1–10.

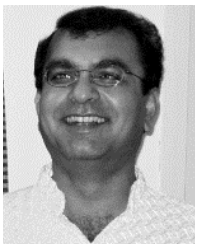
[183] S. Misailovic, M. Carbin, S. Achour, Z. Qi, and M. C. Rinard, "Chisel: Reliability- and accuracy-aware optimization of approximate computational kernels," in *Proc. ACM Int. Conf. Object Oriented Program. Syst. Lang. Appl.*, Portland, OR, USA, 2014, pp. 309–328.

[184] H. Hoffmann, S. Sidiroglou, M. Carbin, S. Misailovic, A. Agarwal, and M. Rinard, "Dynamic knobs for responsive power-aware computing," in *Proc. 16th Int. Conf. Archit. Support Program. Lang. Operating Syst.*, Newport Beach, CA, USA, 2011, pp. 199–212.
[185] H. Hoffmann, J. Eastep, M. D. Santambrogio, J. E. Miller, and A. Agarwal, "Application heartbeats: A generic interface for specifying program performance and goals in autonomous computing environments," in *Proc. 7th Int. Conf. Autonomic Comput.*, Washington, DC, USA, 2010, pp. 79–88.
[186] B. Boston, A. Sampson, D. Grossman, and L. Ceze, "Probability type inference for flexible approximate programming," in *Proc. ACM SIGPLAN Int. Conf. Object-Oriented Program., Syst., Lang., Appl.*, Pittsburgh, PA, USA, 2015, pp. 470–487.
[187] M. Ringenburg, A. Sampson, I. Ackerman, L. Ceze, and D. Grossman, "Monitoring and debugging the quality of results in approximate programs," in *Proc. 20th Int. Conf. Archit. Support Program. Lang. Operating Syst.*, Istanbul, Turkey, 2015, pp. 399–411.
[188] A. Sampson, "The case for compulsory approximation," in *Proc. Workshop Approx. Comput. Across Stack*, Atlanta, GA, USA, 2016, pp. 1–10.
[189] A. C. I. Malossi *et al.*, "The transprecision computing paradigm: Concept, design, and applications," in *Proc. Design, Automat. Test Eur. Conf. Exhib.*, Dresden, Germany, Mar. 2018, pp. 1105–1110.
[190] C. Weis, M. Jung, É. F. Zulian, C. Sudarshan, D. M. Mathew, and N. Wehn, "The role of memories in transprecision computing," in *Proc. IEEE Int. Symp. Circuits Syst.*, Florence, Italy, May 2018, pp. 1–5.
[191] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio, "Binarized neural networks," in *Proc. 30th Int. Conf. Neural Inf. Process. Syst.*, Barcelona, Spain, 2016, pp. 4107–4115.
[192] M. Rusci, D. Rossi, E. Flamand, M. Gottardi, E. Farella, and L. Benini, "Always-ON visual node with a hardware-software event-based binarized neural network inference engine," in *Proc. 15th ACM Int. Conf. Comput. Frontiers*, Ischia, Italy, 2018, pp. 314–319.
[193] G. Tagliavini, S. Mach, D. Rossi, A. Marongiu, and L. Benini, "A transprecision floating-point platform for ultra-low power computing," in *Proc. Design, Automat. Test Eur. Conf. Exhib.*, Dresden, Germany, Mar. 2018, pp. 1051–1056.
[194] S. Mach, D. Rossi, G. Tagliavini, A. Marongiu, and L. Benini, "A transprecision floating-point architecture for energy-efficient embedded computing," in *Proc. IEEE Int. Symp. Circuits Syst.*, Florence, Italy, May 2018, pp. 1–5.
[195] L. Fousse, G. Hanrot, V. Lefèvre, P. Pélissier, and P. Zimmermann, "MPFR: A multiple-precision binary floating-point library with correct rounding," *ACM Trans. Math. Softw.*, vol. 33, no. 2, 2007, Art. no. 13.
[196] M. Carbin, D. Kim, S. Misailovic, and M. C. Rinard, "Proving acceptability properties of relaxed nondeterministic approximate programs," in *Proc. 33rd ACM SIGPLAN Conf. Program. Lang. Design Implement.*, Beijing, China, 2012, pp. 169–180.
[197] O. Tahan and M. Shawky, "Using dynamic task level redundancy for OpenMP fault tolerance," in *Proc. 25th Int. Conf. Archit. Comput. Syst.*, Munich, Germany, 2012, pp. 25–36.
[198] C. Bolchini, A. Miele, and D. Sciuto, "An adaptive approach for online fault management in many-core architectures," in *Proc. Conf. Design, Automat. Test Eur.*, Dresden, Germany, 2012, pp. 1429–1432.
[199] G. Yalcin, O. Unsal, and A. Cristal, "FaulTM: Error detection and recovery using hardware transactional memory," in *Proc. Conf. Design, Automat. Test Eur.*, Grenoble, France, 2013, pp. 220–225.

Massimo Alioto (M'01–SM'07–F'16) was with the University of Siena, a Visiting Professor at Intel Labs–CRL, with the University of Michigan—Ann Arbor, with the University of California at Berkeley, Berkeley, and with EPFL. He is with the National University of Singapore, where he is currently the Director of the Integrated Circuits and Embedded Systems Area and leads the Green IC Research Group.

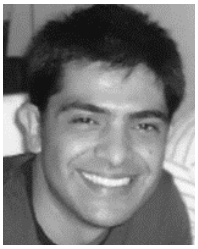
He has authored or co-authored over 250 publications on journals and conference proceedings. He has co-authored of three books from Springer, including *Enabling the Internet of Things: From Integrated Circuits to Integrated Systems* (2017). His primary research interests include ultra-low power VLSI circuits, self-powered systems, near-threshold circuits for green computing, energy-quality scalable systems, circuit techniques for emerging technologies, and hardware-level security, among the others.

Dr. Alioto was a Distinguished Lecturer (2009–2010), and a member of the Board of Governors of the IEEE CAS Society (2015–2020), and the Chair of the VLSI Systems and Applications Technical Committee (2010–2012). He is/was a Conference Technical Program Chair (ISCAS 2022, ICECS, NEWCAS, SOCC, PRIME, VARI, and ICM), a Track Chair (ICCD, ISCAS, ICECS, VLSI-SoC, APCCAS, and ICM), and a TPC Member (ISSCC and ASSCC). He has served as a Guest Editor for numerous IEEE journal special issues, as well as an associate editor for a number of other journals. He is the Editor-in-Chief of the IEEE TRANSACTIONS ON VLSI SYSTEMS (2019–2020) and the Deputy Editor-in-Chief of the IEEE JOURNAL ON EMERGING AND SELECTED TOPICS IN CIRCUITS AND SYSTEMS (2018).



Vivek De (F'11) received the B.Tech. degree in electrical engineering from IIT Madras, Chennai, India, the M.S. degree in electrical engineering from Duke University, Durham, NC, USA, and the Ph.D. degree in electrical engineering from Rensselaer Polytechnic Institute, Troy, NY, USA. He is currently an Intel Fellow and the Director of the Circuit Technology Research Group at Intel Labs. He is responsible for providing strategic technical directions for long-term research in the future circuit technologies and leading energy efficiency research across the hardware stack.

He has 273 publications in refereed international conferences and journals with a citation H-index of 73, and 221 patents issued with 29 more patents filed (pending). He received an Intel Achievement Award for his contributions to an integrated voltage regulator technology. He also received the Best Paper Award at the 1996 IEEE International ASIC Conference, and nominations for Best Paper Awards at the 2007 IEEE/ACM Design Automation Conference (DAC) and the 2008 IEEE/ACM International Conference on Computer-Aided Design. He has also co-authored a paper nominated for the Best Student Paper Award at the 2017 IEEE International Electron Devices Meeting. One of his publications was recognized in the 2013 IEEE/ACM DAC as one of the Top 10 Cited Papers in 50 Years of DAC. Another one of his publications received the Most Frequently Cited Paper Award from the IEEE Symposium on VLSI Circuits at its 30th Anniversary in 2017. He was recognized as a Prolific Contributor to the IEEE International Solid-State Circuits Conference at its 60th Anniversary in 2013 and a Top 10 Contributor to the IEEE Symposium on VLSI Circuits at its 30th Anniversary in 2017. He has served as an IEEE/EDS Distinguished Lecturer in 2011 and an IEEE/SSCS Distinguished Lecturer in 2017–2018. He received the 2017 Distinguished Alumnus Award from IIT Madras.



Andrea Marongiu (M'12) received the M.Sc. degree in electronic engineering from the University of Cagliari, Italy, and the Ph.D. degree in electronic engineering from the University of Bologna, Italy. He has been a Post-Doctoral Research Fellow at ETH Zürich, Switzerland. He is currently an Assistant Professor at the Department of Computer Science and Engineering, University of Bologna.

His main research interests focus on programming models and architectures in the domain of heterogeneous multi- and many-core systems on a chip. This includes language, compiler, runtime, and architecture support to efficient address performance, predictability, energy, and reliability issues in parallel, embedded systems, and hardware-software co-design of accelerator-based MPSoCs. In this field, he has published over 100 papers in international peer-reviewed conferences and journals, books, and book chapters.

He has collaborated and collaborates with several international research institutes and companies and serves or has served as a TPC member for several international conferences and workshops in his field (DATE, SCOPES, MCSoC, EUC, FPL, and DASIP).