J Sign Process Syst (2015) 78:49–62
DOI 10.1007/s11265-014-0931-y

Hardware Specialization in Low-power Sensing Applications to Address Energy and Resilience

Zhuo Wang · Kyong Ho Lee · Naveen Verma

Received: 20 February 2014 / Revised: 11 July 2014 / Accepted: 15 July 2014 / Published online: 10 August 2014

Abstract This paper explores implications of introducing machine learning capabilities within a hardware-specialized platform for low power embedded sensing applications. Such a platform enables algorithms well suited for analyzing complex sensor signals under strict energy constraints. However, the benefits go further, enabling the effects of errors to be overcome in the presence of hardware faults within the platform. Although errors can result in substantial bit-level perturbations, the approach described views these an alteration on the way that information is encoded within the embedded data. The new information encoding can thus be learned in the form of an error-aware model. The energy implications of hardware-specialized machine-learning kernels are analyzed using a fabricated custom IC, and the hardware-resilience implications are analyzed using an FPGA platform, which permits controllable and randomized injection of logical hardwarefaults.

Keywords Machine learning · Embedded sensing · Fault tolerance · Boosting · FPGA emulation

Z. Wang (✉) · K. H. Lee · N. Verma
Department of Electrical Engineering, Princeton University, Princeton, NJ, 08544, USA
e-mail: zhuow@princeton.edu

N. Verma
e-mail: nverma@princeton.edu

Present Address:
K. H. Lee
Samsung Research America, Dallas, TX, USA
e-mail: kyongho.l@samsung.com

1 Introduction

Advances in sensor technologies and the proliferation of embedded sensors is substantially expanding the scope and scale of computing applications in the future by seeking to derive high-quality inferences from the sensor inputs. In particular, we are seeing the rise of a class of applications, known as recognition, mining and synthesis (RMS) [1]. In these applications, inference refers to the capability to make simple decisions and construct models for predictions, based on previous observations of data. Recently, efficient and powerful algorithms for inference have emerged from the domain of machine learning. In the context of sensing systems, machine-learning algorithms are proven highly beneficial. The reason is that many sensor signals of interest are too complex to model analytically. Data-driven approaches, on the other hand, enable the construction of high-order models. For example, the expression of disease states such as epileptic seizures is intractable to model analytically in sensor signals such as the electroencephalogram (EEG), since the mechanisms governing how seizures appear are both complex and subject to variabilities. Machine-learning algorithms have proven effective in modeling the onset of epileptic seizures in the EEG [2]. The challenge is that, generally speaking, machine-learning algorithms can impose severe requirements on system resources (most notably energy and memory), when considered in the context of low-power sensing devices. In fact, in such devices, resilient operation in the face of increasing hardware faults and non-idealities is also elevating to a dominating concern. The main contribution of this work is exposing structure in the machine-learning algorithms that exists broadly in resource-constrained sensor applications. We describe the critical implications this structure has on the design of embedded sensing platforms, enabling both energy and memory as well as resilience to be substantially addressed by exploiting this structure through hardware specialization.

2 Algorithmic Structure Pertaining to Energy and Resilience

This section examines the typical structure of machine-learning algorithms in embedded sensing applications. In particular, we suggest that this structure can be exploited within a broad class of applications to substantially benefit the overall system energy and resilience.

2.1 Exploiting Structure for Energy Efficiency

A strength of machine-learning algorithms is that they enable the construction of high-order models even when strong analytical representations of the signals (or the processes that generate the signals) do not exist. The challenge is that for many algorithmic kernels, the computational energy scales with the resulting model complexity, thus posing a limiting factor. For example, [3] shows that a large number of support vectors are needed in an ECG cardiac-arrhythmia detection system based on a kernel SVM. Hardware specialization is well known as a technique to substantially reduce computational energy. However, hardware specialization trades programmability for energy efficiency [4], and thus its feasibility depends on the level of programmability required in a class of applications. For this reason, many different approaches to hardware specialization have been taken. For example, to address performance and energy, graphical processing units (GPUs) are commonly used to address complex video- and image-processing functions beyond the level achievable by general-purpose CPUs [5]. GPUs have incorporated a substantial level of programmability to address a large application space. On the other hand, hardware specialization for key fixed functions has also been widely employed. For example, in [6] a hardware-specialized accelerator is used to address sequence comparison in biological-computing applications, wherein analysis of large amounts of data is required. In the case of machine-learning algorithms for embedded-sensor applications we find structure that can be exploited to mitigate the effects of this tradeoff with programmability. As described in [7], the algorithms substantially separate the need for programmability from the need for energy efficiency. Namely, feature extraction depends strongly on the application signals and the inferences of interest, thus requiring a high degree of programmability. On the other hand, the inference functions themselves, which apply the high-order data-driven models, are energy intensive but are performed through specific kernel functions. On a high-level, this implies that the inference kernels can be delegated to hardware accelerators while the feature computations can be delegated to programmable CPUs, preserving both the programmability required and substantially reducing the overall energy of the applications. Section 4 describes the details of such an approach.

2.2 Exploiting Structure for Hardware Resilience

In addition to energy, hardware resilience has elevated to a first-class design challenge. This is brought in by the continued scaling of transistors. With scaling, hardware faults arise due to physical defects caused, for instance, because of sub-wavelength lithography, limited process control of fine features, electrical device variations, etc. [8–10]. Furthermore, candidate technologies to replace CMOS (carbon nanotube transistors, tunneling FETs, etc.) are primarily limited today by fabrication reliability. Taking carbon nanotube transistors (CNTs) as an example, the challenges include tube-density variations causing substantial drive-current variability and the selectivity of semiconducting (versus metallic) tubes causing effective shorts [11]. Such issues limit the scale of systems that can be achieved today [12].

While addressing hardware resilience at the device level has been and continues to be a critical focus for the research community [13–15], the recognition is widely emerging that the challenges cannot be solved at the device level alone. This has led to an active branch of research that explores approaches whereby device-level non-idealities are tolerated through architectural and algorithmic solutions [16–18]. Instead of requiring that circuit components perform deterministic functions, as in the case of traditional circuit design, such approaches assume that the basic circuit components present stochastic behaviors [19, 20]. The architectures and algorithms employed take into consideration such stochastic behaviors and achieve system-level robustness by tolerating error manifestations using various techniques.

A particular technique that is well-suited to machine-learning algorithms is data-driven hardware resilience (DDHR) [21, 22]. While previous work (e.g., [23]) has explored the inherent error resilience of inference functions based on machine-learning kernels, in fact machine-learning algorithms can enable substantially higher levels of resilience. Specifically, DDHR exploits data-driven modeling capabilities to model data *in the presence* of errors generated due to hardware faults. As a result, inference performance is retained so long as the mutual information is retained in the error-affected data. The extent to which DDHR improves system-level resilience, however, depends on algorithmic structure. In particular, it is beneficial for the inference stage to be the final algorithmic stage since errors originating in all preceding stages can thus be addressed. The inference stage itself, requires some particular design attention, as in the case of a computational accelerator (this time to address computational resilience rather than computational energy). Nonetheless, similar to yielding strong leverage for reducing energy in the entire system, the inference stage, appropriately designed can yield high leverage for overcoming errors in the entire system. Section 5 describes the details.

3 Background

Technological scaling has profound impacts on integrated circuits. Strictly speaking, scaling has focused on diminishing the marginal cost of transistors. Architectural innovations have then translated the availability of transistors into gains spanning a wide range of system aspects (performance, energy, feature complexity). Perhaps the most critical drawback of technological scaling, however, has been degraded reliability and/or increased cost in meeting a required level of reliability. Given the trend of leveraging additional transistors to address system challenges, a compelling question that has emerged is whether technological scaling itself can be used, at the architectural or system level, to overcome reliability challenges. In the following subsections, we describe first how technological scaling has addressed energy. Then we describe the challenges technological scaling has brought on in terms of device-level reliability. Finally, we survey approaches that have been used to leverage technological scaling for enhanced reliability; some of these explicitly invoke tradeoff with energy efficiency.

3.1 Technological Scaling to Address Energy

The availability of transistors has enabled energy reductions through several architectural trends. Two of the most prominent trends are parallelism and hardware specialization. Parallelism is establishing itself as a generalizable way to address energy. The underlying idea is that if transistors are readily available, these can be devoted towards blocks that replicate the same function. In order to meet an overall performance requirement, the individual blocks can thus operate more slowly allowing their supply voltage to be lowered, thereby reducing energy. Parallelism has been used successfully with voltage scaling to substantially reduce energy by employing a large number of replicated blocks. For example in [24], a video decoder is designed with 16 parallel slice engines which enables it to operate at a low supply voltage of 0.7V while maintaining a high throughput of 300 million bits per second. Parallelism can be also exploited for energy savings for general purpose computations. Modern GPUs are used not only for graphics functions, but also for replacing a general-purpose CPU for many arithmetic-intensive and memory-intensive computations. This has shown to substantially elevate performance and improve energy efficiency [25].

While parallelism has shown promise for general-purpose and fixed-function computation, hardware specialization has shown orders of magnitude energy reduction for fixed-function computation [7, 26]. With low marginal cost of transistors, many functions can be simultaneously implemented, enabling substantial energy reduction over a large set of computations. The underlying idea is that fixed-function computation eliminates the overheads associated with programmable control and data flow in a processor. In fact, hardware specialization can imply various levels of programmability. For example, specialized-instruction-set processors retain software-level programmability but also incorporate selective instructions that are highly utilized in an application domain in order to avoid the overhead of compiling these from a base instruction set [27]. The greatest energy savings, however, are derived from entirely fixed-function hardware that accelerates high-level computations [7]. Given the tradeoff with programmability and the cost associated with mapping high-level computations to dedicated hardware, two characteristics play a key role in determining the effectiveness of hardware acceleration. First, to mitigate the impact on application-level programmability, it is preferable that applications have structure wherein the underlying computations separate the need for energy efficiency from the need for programmability. If the computations requiring a high degree of programmability are not energy dominating, these can be delegated to a general-purpose CPU, while only the computations requiring high energy efficiency, but limited programmability, can be delegated to hardware accelerators. Second, to mitigate the need to map an excessive number of computations to a hardware accelerator, it is preferable if applications have structure wherein a small number of computations have large leverage on the energy in an application. If the energy in an application is dominated by a small set of computations, these can readily be mapped to hardware accelerators to substantially benefit the overall application energy.

We will show in the coming sections that algorithms for deriving simple inferences from sensor data substantially exhibit these two characteristics. This makes hardware acceleration an effective path for addressing the severe energy constraints faced in these applications. Importantly, in addition to these two characteristics, we also show that the algorithms exhibit a third characteristic, where the computations delegated to hardware acceleration not only have large overall leverage in terms of energy, but can also have large overall leverage in terms of *system-level resilience*.

3.2 Technological Scaling Elevating Reliability Challenges

While substantial and diverse benefits have been derived from technological scaling, integrated circuits have also been faced with substantial challenges. The origins of these have been highly varied. For example, nano-scale lithography suffers from defects due to sub-resolution features [8]. To address this, a range of strategies have been used, including double patterning [28], assist features introduced in photomasks, etc. [29]. Unfortunately, all of these lead to complex design rules and some remaining yield loss, ultimately increasing cost. In addition to this, the impacts of process fluctuations are prominent, both physically in the patterned features [29] and electrically in the fabricated transistors [30].

To reduce the likelihood of chip-level malfunctions due to such effects, it is common practice to introduce engineering margins at the mask-design, circuit-design, subsystem-design, and system-design levels. In all, such margins consume substantial system resources (energy, performance, etc.) [30, 31], leading to increasing efficiency degradations as the offending effects rapidly increase in severity. As described in the following section, the research community has responded by seeking solutions that avoid the need for traditional margining. In all cases, the objective has been to evaluate whether such approaches can enable reliable operation with reduced degradation on system resources (such as energy) than traditional design margining would impose. While the approaches being explored have shown substantial promise, they are all in the research phase. Having not yet been adopted into mainstream design and manufacturing practice, important questions still remain with respect their performance in the context of practical defect sources. Consequently, evaluation of the research ideas with appropriate fault models and fault profiles remains an important subtopic; this idea is revisited in an approach described later in this paper.

3.3 Convergence of Energy and Resilience

The transistors made available through technological scaling enable several avenues for potentially improving resilience at the system level. In fact many of these exploit tradeoffs with energy efficiency.

Perhaps a clear approach to exploiting additional transistors towards reliability is through redundancy. In fact, several architectures to effectively employ redundancy have been reported. For example, N-modular redundancy (NMR) is a strategy where n identical modules are implemented, and the ultimate output is the majority vote over the n modules [32]. Soft NMR takes this idea further by implementing n unreliable, simple-structure estimators. A fusion module is used to infer the correct output from the estimated results statistically [33]. Algorithmic noise tolerance (ANT) [34] takes a different approach to redundancy, wherein two processing units with explicitly diverse error characteristics are used together. The first unit has low-precision but also low error rate, while the second unit has higher precision but also higher error rate. Thus the low-precision unit can be used to detect clear errors in the high-precision unit, and provide corrected approximations to the output, once an error is detected. ANT is an example of an emerging class of systems that leverage a hardware block having low error rate towards enhanced resilience of the overall system. For example, lowpass filter circuit implemented with ANT achieves 81 % energy savings due to aggressive voltage scaling, at a sacrifice of performance loss of less than 0.5 dB [35].

Beyond redundancy, approaches have focused on addressing the resilience of programmable processors both through architectural and microarchitectural approaches. For example, Razor uses low-complexity error-detection registers to detect and correct timing faults in a data-path register with the penalty of one clock cycle for correction [18]. With appropriate support added in the microprocessor pipeline, timing faults induced by aggressive voltage scaling can thus be tolerated. Energy savings of 12.9 % up to 63.6 % were reported for a range of applications, with penalties on instructions per cycle (IPC) of 0.06 % to 5.88 % reduction. Another example is Error Resilient System Architecture (ERSA) [16]. ERSA proposes a multi-core processor architecture where the computational control-flow operations are delegated to a core that is explicitly protected against hardware faults. This can be realized through conservative design and fabrication rules [30] or through increased supply voltage in a voltage-scaling system [36]. On the other hand, data-flow operations are delegated to cores that are fault-prone. Since control flow errors can result in catastrophic system crash, such failures are prevented through an asymmetric architecture. Results with practical applications show that so long as the data-flow errors are not too severe (below 20000 errors per second per core), they can be readily tolerated. Like ANT, ERSA represents an architecture wherein an explicitly fault-protected hardware block is leveraged towards system-level resilience. Later in the paper, we describe yet another approach, termed data-driven hardware resilience (DDHR) which also applies this principle. First, however, we show in the next section how the hardware block exploited towards system-level resilience in DDHR is also exploited towards system-level energy savings to derive simple inferences in low-power sensing devices.

4 Specialization of Machine-learning Kernels for Low Energy

In this section, we first introduce the energy characteristics observed for simple inference algorithms in embedded sensing applications; this is done using a representative case study involving classification. Having identified structure in the algorithms that can be exploited towards hardware specialization, we then present an accelerator-based microprocessor. Finally, we illustrate the energy impact that such an approach enables using two application demonstrations.

4.1 Energy Characteristics of Embedded Machine-learning Algorithms

Machine-learning algorithms enable the construction of high-order analysis models for analytically-intractable signals. Generally, such models do not have compact parametric representations. As a result, the model, in order to enable high-order analysis functions, must itself be represented through a dataset that is large enough to adequately represent statistical attributes of the signals. To illustrate the challenges that this raises for energy, we consider a widely used supervised learning algorithm for classification, called the support vector machine (SVM). In an SVM, training data together with class labels are used during a training phase to construct a decision function (model), which is used for classifying new instances of data. The data is usually represented as feature vectors through a feature-extraction stage. The role of feature extraction is to extract critical parameters from the data to facilitate modeling and classification. In particular, this involves discarding aspects of the data or making redundant and/or superfluous aspects invariant. For example, for the arrhythmia-detection application illustrated in Fig. 1, the extracted features are sampled ECG signal points along with the time intervals between peaks [37]. Following feature extraction, the feature vector can be mapped to a point in the feature space. The feature space is a hyper dimensional space, with dimensionality set by the feature vector. In Fig. 1, principal component analysis (PCA) is applied to reduce the feature space to two dimensions only for visualization; for classification, high-dimensional spaces can be handled by the SVM. During training, the labeled feature vectors form class distribution in the feature space. Those feature vectors lying close to the edges of the class distributions are then retained to represent a decision boundary, which is the classification model applied to subsequent data. The selected feature vectors are called support vectors. The decision boundary is formally represented as Eq. 1. Here $K(\bullet)$ is called kernel function, which is typically a linear, polynomial, or exponential function. $\vec{x_i}$ are support vectors. a_i and b are training parameters.

$$\mathbf{f(x)} = \sum_{\mathbf{i=1}}^{\mathbf{n}} \mathbf{a_i K(x_i, x) - b} \qquad (1)$$

Considering practical applications, what we commonly find is that high-order SVM-classification models are needed along with non-linear kernel functions for adequate performance. As an example, Fig. 2a shows that roughly 11000 support vectors are needed for high performance of an ECG-based arrhythmia detector. Unfortunately, as can be indicated from Eq. 1, the computational complexity for classification scales with the number of support vectors [38]. Figure 2b shows the corresponding energy, profiled on an MSP430 microprocessor [7]. Also shown is the feature-extraction energy, showing that classification can dominate by orders of magnitude. However, it is important to note, that while feature extraction needs to be programmable, in order to address various applications and application signals, SVM classification is performed through kernel computations. This suggests that energy-intensive SVM classification can be delegated to an accelerator with minimal degradation on application-level programmability, while feature extraction can be delegated to a general-purpose CPU with minimal degradation on application-level energy.

4.2 Demonstration of Hardware-specialized Microprocessor

Even with the structure described above, a substantial degree of configurability in the accelerator can substantially benefit the scope of applications enabled. In [7], an accelerator formed for low-level computational hardware is described, wherein high-level inference algorithms can be synthesized. Through this approach, a broad range of feature-extraction and inference computations are supported. In this section, we first describe two applications that are demonstrated on the custom microprocessor from [7], and then we describe experimental results, including energy measurements.

4.2.1 Applications for Demonstration

Two applications are considered for demonstrating the impact that an accelerator-based microprocessor can have: (1) EEG-based epileptic seizure detection; and (2) ECG-based cardiac-arrhythmia detection.

EEG-based Epileptic Seizure Detection For seizure detection, expert-annotated EEG data from the CHB-MIT

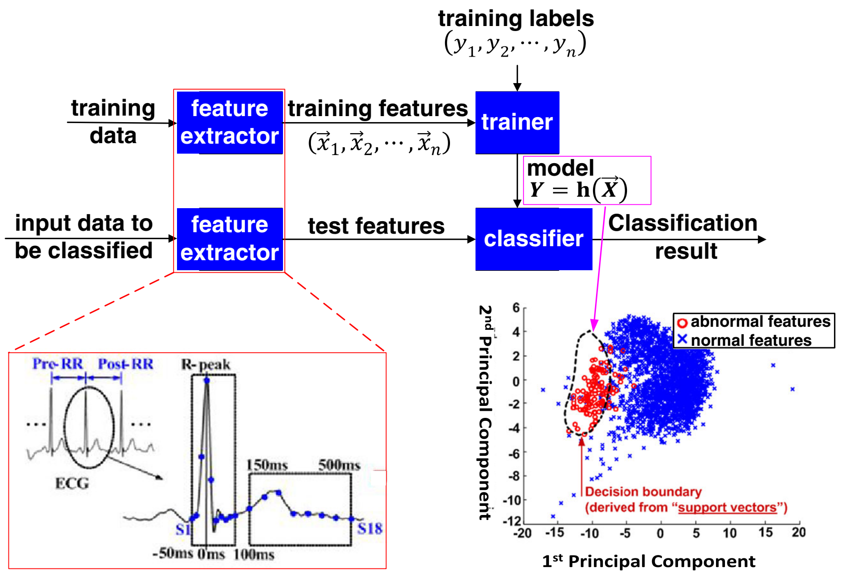


Figure 1 Illustration of SVM algorithm using ECG-based arrhythmia detection as an example.

database is used [39][40]. Up to 18 channels of EEG signals are available, but only two are selected to simplify the demonstration without much sacrifice in performance [41]. The feature extraction algorithm is described in [42]. Each EEG signal (channel) is first band limited to the frequency-range of interest and down-sampled by eight. The down-sampled signal is then presented to a bank of seven band-pass FIR filters with center frequencies ranging from 0 to 19 Hz. The energy in each band is computed by accumulating the absolute value of the output samples over a 2 second epoch. This forms seven features for each channel of EEG, with a total of 14 features for the 2 channels. Finally, the extracted features from two previous epochs are concatenated, constructing a 42 dimensional feature vector. The feature vector, thus extracted, is then presented to an SVM classifier (Fig. 5a). The application-level metrics for evaluating performance are (1) sensitivity, which corresponds to the percentage of correctly detected seizures; (2) false alarms, which corresponds to the total number of incorrectly declared seizure onsets; (3) latency, which corresponds to the average time delay between expert-labeled seizure onset and detector-declared seizure onset [42].

ECG-based Cardiac Arrhythmia Detection For arrhythmia detection, expert-annotated ECG data from the MIT-BIH database is used [39, 40]. The feature extraction algorithm is described in [43]. The ECG signal is presented to a seven-stage wavelet-transform filter bank. This results in a total of 256 data points from each ECG beat, yielding a 256 dimensional feature vector (Fig. 5b). The application-level metrics for evaluating performance are (1) the true positive rate, which corresponds to the percentage of correctly classified normal beats, and (2) the true negative rate, which corresponds to the percentage of correctly classified abnormal beats.

4.3 Measurements and Demonstrations

Figure 3a shows the die photo of the custom accelerator-based microprocessor [7]. The measurements show the

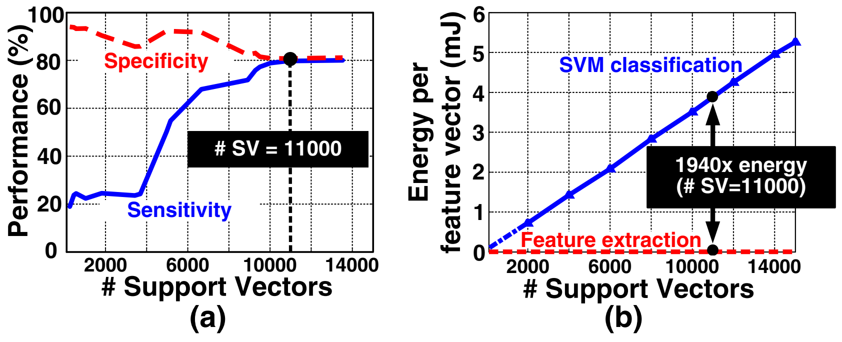
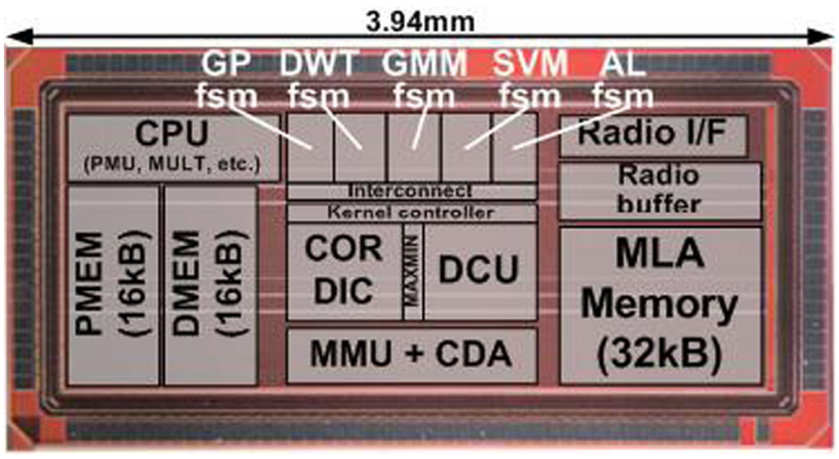


Figure 2 Analysis of ECG-based arrhythmia-detraction application considering (**a**) performance scaling with model complexity (number of support vectors), and (**b**) energy scaling with model complexity (number of support vectors) [7].

Figure 3 **a** Die photo and **b** comparison of energy consumption with and without accelerators for two demonstrated applications.



(a)

	Seizure Detection *			Arrhythmia Detection **		
	w/o acc.	w/ acc.	saving	w/o acc.	w/ acc.	saving
Feature Extractor	39.5 μJ	3.33 μJ	x 11.9	88.4 μJ	1.61 μJ	x 54.9
Classifier	1.44 mJ	1.11 μJ	x 1300	6.04 mJ	150 μJ	x 40.3

* Energy for computing each feature vector with one channel of EEG signal.
** Energy for computing each feature vector with ECG signal.

(b)

energy savings enabled by the accelerator-based system compared to a CPU-only implementation for both applications. As shown, by implementing the inference computations using an accelerator, the *overall* energy dramatically benefits. In the following section, we describe how implementing the inference computations using an explicitly-fault-protected accelerator, the *overall* system-resilience dramatically benefits.

5 Specialization of Machine-learning Kernels for Resilience

The previous section explored asymmetry in the level of impact on energy that the system components have, motivating the hardware specialization of machine-learning kernels. This section explores asymmetry in the level of impact on system-level resilience that system components have, once again motivating the hardware specialization of machine-learning kernels. This time specialization explicitly implies the need to ensure fault protection. The idea is that ensuring the fault protection of a machine-learning stage can enable a high degree of system-level resilience. As an example, below we describe an approach known as data-driven hardware resilience (DDHR) [22]. We point out that further work has also been done showing that in fact only a small portion of the machine-learning stage itself needs to be fault protected [44].

5.1 DDHR Concept

The idea of DDHR is to utilize data-driven training to construct a model for inference from data whose statistics is set by variances due to the application-level signals *and* variances due to errors in the system. The key to DDHR is thus the construction of an error-aware (EA) model. This is achieved by using error-affected data for training that is derived from the fault-affected system. To illustrate the concept, Fig. 4b shows feature-vector data derived from both a fault-free system and a fault-affected system for an EEG-based seizure-detection application (the data is shown projected on two dimensions using PCA to aid visualization); as we describe in the following subsection, the fault-free and fault-protected systems are implemented via FPGA emulation. We see that errors substantially perturb the data distributions. While the original classification model is thus inadequate, we see that a classification model trained using the error-affected data can provide effective classification. It is worth mentioning, however, for data-driven training to work, a critical aspect is that the data statistics of the training set should match those of the testing set. Consequently, the faults that we target must result in stationary statistics (e.g., the approach has not been evaluated in the context of transient or soft faults). Accordingly, with DDHR, hardware faults in a processor can be viewed as altering the statistics of embedded data in the form of new class distributions. While the resulting data statistics

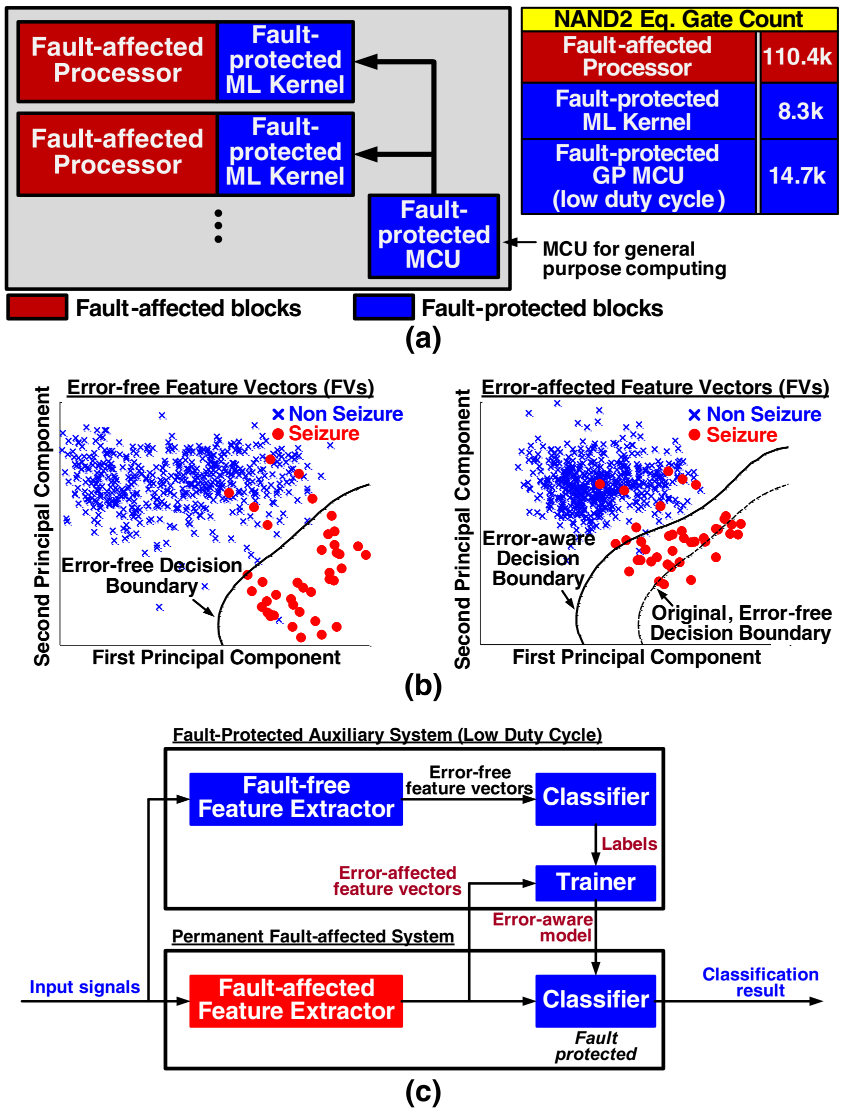

Fault-affected Processor
Fault-protected ML Kernel
Fault-affected Processor
Fault-protected ML Kernel
Fault-protected MCU
MCU for general purpose computing
NAND2 Eq. Gate Count
Fault-affected Processor 110.4k
Fault-protected ML Kernel 8.3k
Fault-protected GP MCU (low duty cycle) 14.7k
Fault-affected blocks
Fault-protected blocks
(a)
Error-free Feature Vectors (FVs)
Non Seizure
Seizure
Error-free Decision Boundary
Second Principal Component
First Principal Component
Error-affected Feature Vectors (FVs)
Error-aware Decision Boundary
Original, Error-free Decision Boundary
(b)
Fault-Protected Auxiliary System (Low Duty Cycle)
Fault-free Feature Extractor
Error-free feature vectors
Classifier
Labels
Trainer
Error-affected feature vectors
Error-aware model
Permanent Fault-affected System
Input signals
Fault-affected Feature Extractor
Classifier
Fault protected
Classification result
(c)

Figure 4 Illustration of DDHR. **a** system architecture with minimum fault-protection, with example gate counts provided with synthesis to ASIC technology of an EEG seizure detection system; **b** construction of an error-aware model training to the variance of data distribution due to hardware faults; **c** on-line training system with an auxiliary system for generating training labels.

are unpredictable, the new statistics can be learned from a fault-affected training set, leading to the error-aware model.

In addition to the basic principle behind DDHR, a practical architecture has also been proposed [22]. In particular, to construct an error-aware model within a supervised learning framework, besides the training data, we also require training-data labels. Typically, ground truths are used for the training-data labels, and these are obtained from an oracle, such as an expert. However, generally, in practical systems, particularly as these scale in volume, such a source may not be viable. In [22], the system is achieved without the need for supervision from an external oracle, and training is performed entirely within the architecture. To accomplish this, estimated labels are used rather than ground truths. These are obtained using a *temporary* auxiliary system, as shown in Fig. 4c. The auxiliary system implements a fault-free system for feature extraction and classification within a fault-protected microcontroller (MCU). Then, during training, data is generated directly by the fault-affected processor, but input data is also provided to the auxiliary system to generate estimated training labels. This means that the estimated labels correspond to the class declarations derived from a *fault-free* system. This enables training that leads to a system whose performance can at best converge with that of a fault-free system. In fact, the use of a strong classifier (such as an SVM with radial-basis function kernel) will guarantee that the trained model can fit the training set, i.e., at least the training-data instances will be classified as they would by a fault-free system. However, the performance generalized to a testing set will depend on how errors affect the distribution of data in the feature space. Nonetheless, experiments show that in practice, this method of generating training labels yields performance close to that achieved with perfect labeling. It is important to note, that the axillary system, implemented on the MCU does consume excessive energy and resources.

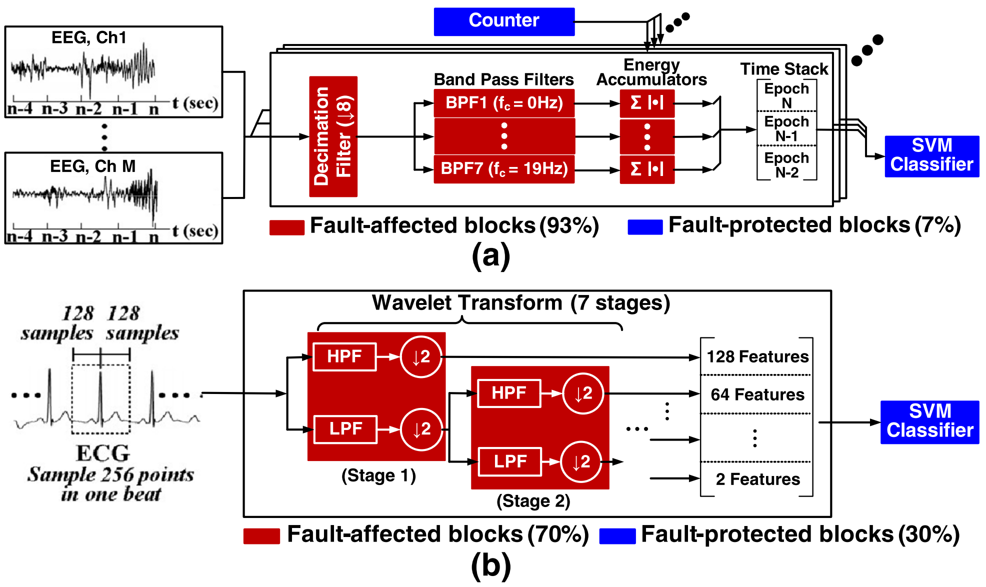


Figure 5 illustration of the applications used for demonstration: **a** EEG-based seizure detector; **b** ECG-based arrhythmia detector.

However, since training is infrequent and not required in real time, this overhead can be mitigated and greatly amortized over the long-term operation of the system.

5.2 DDHR Demonstration

In this section, we discuss the experimental approach for implementing the systems and emulating hardware faults [22].

5.2.1 Applications for Demonstration

The same applications as described in Section 4.2.1 are used for demonstrating the DDHR concept. Here, feature extractors are implemented using accelerators as shown in Fig.5. As described in Section 5.1, the feature-extraction processors are fault-affected, corresponding to 93 % and 70 % of the total circuitry for seizure detector and arrhythmia detector, respectively.

5.2.2 Experimental Approach

To evaluate the DDHR systems, a methodology is needed to controllably inject faults in the circuitry. The data-intensive nature of DDHR training and testing renders simulation-based methods inviable, particularly if low-level simulations (gate-level) are to be targeted to appropriately represent hardware faults. FPGA-based emulation is thus employed.

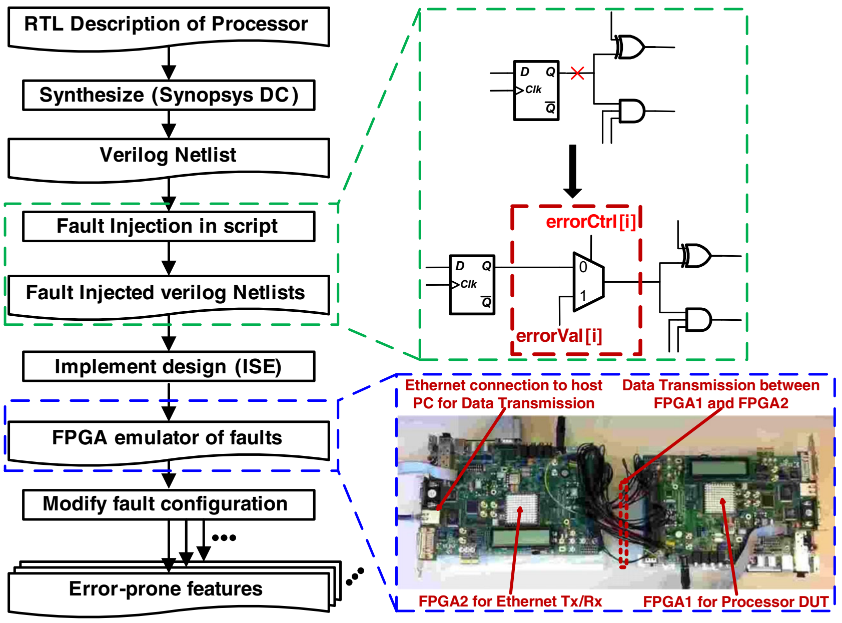

RTL Description of Processor
Synthesize (Synopsys DC)
Verilog Netlist
Fault Injection in script
Fault Injected verilog Netlists
Implement design (ISE)
FPGA emulator of faults
Modify fault configuration
Error-prone features
D Q Clk Q̄
errorCtrl [i]
errorVal [i]
0
1
Ethernet connection to host PC for Data Transmission
Data Transmission between FPGA1 and FPGA2
FPGA2 for Ethernet Tx/Rx
FPGA1 for Processor DUT

Figure 6 FPGA emulation system and stuck-at fault-injection flow.

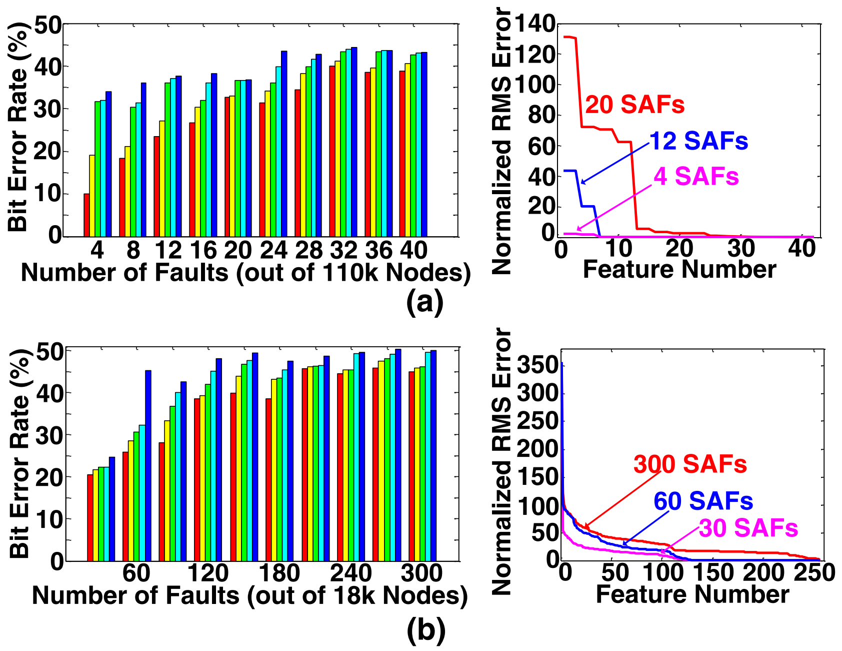


Figure 7 Bit level error statistics for (**a**) seizure detector and (**b**) arrhythmia detector, including (*1*) the bit error rates of the output data for all test cases at different fault levels (*left*), and (*2*) the RMS values of errors normalized to the RMS values of features themselves for three fault levels (*right*). The feature number in the right-hand plots is sorted according to reducing normalized RMS error to aid visualization.

While a range of fault models can be considered, the initial demonstration utilize a stuck-at fault model. Stuck-at faults have been effective for representing physical faults in various contexts [45], and they are also directly representative of physical defect sources of particular concern due to technological scaling, such as lithographic defects [8], threshold-voltage variation inducing logic errors [46], etc.

Figure 6 shows the FPGA emulation and fault-injection flow. Register-transfer level (RTL) descriptions of the processors are first developed and synthesized to generate a Verilog gate-level netlist. An executable script is then edits the synthesized netlist to inject stuck-at faults. A large number of output nodes in the netlist are randomly selected. Multiplexers are injected to control the output net to be driven either by the original signal or by a static logic value of 1 or 0. By setting the error-control signals (multiplexer signals), faults can deliberately be injected in particular circuit nodes, and the injected stuck-at value can be selected by the error value signals. The modified netlist, along with a fault-control module is then mapped to the FPGA for running experiments. In the experiments, the number of nodes affected by faults is scaled to enable characterization. Further, at each fault level, multiple cases of fault-affected nodes are tested in order to emulate randomized fault locations within the circuit. All characterization parameters are controllable simply by modifying the error control signals

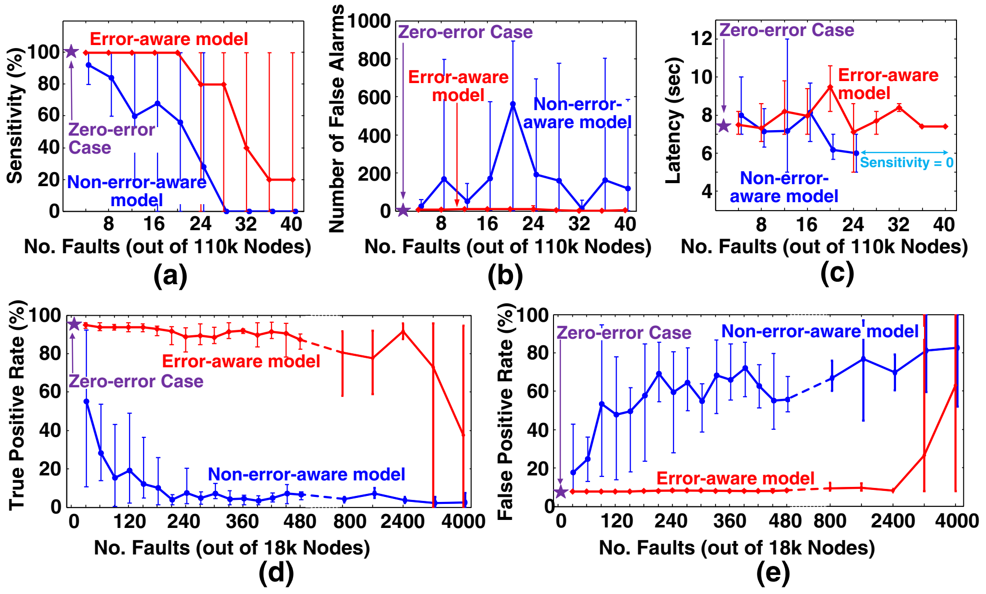


Figure 8 Classification performance comparison with DDHR (in *red*) and without DDHR (in *blue*) for (**a**)–(**c**) seizure detection and (**d**)–(**e**) arrhythmia detection (for arrhythmia detection, training is performed by fixing the true negative rate at 95 % for all test cases to aid comparison).

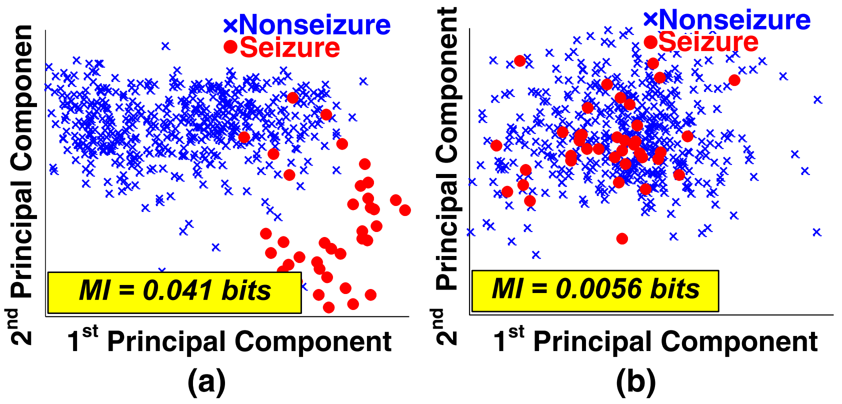


Figure 9 PCA of feature space visualizing the feature distribution for EEG features that are computed with (**a**) fault-free processor and (**b**) fault-affected processor where the performance is not restored. The latter case corresponds to low mutual information.

and error value signals. For interfacing with a host computer, both to stream input data and acquire output data, a second FPGA is used strictly as an Ethernet interface. The FPGA experimental setup is pictured in Fig. 6.

5.3 Results and Discussion

In this section, the FPGA-emulation results for EEG-seizure detection and ECG-arrhythmia detection are presented and some analysis is provided focusing on the fundamental limit of the performance achievable by a DDHR system.

5.3.1 Bit Level Errors due to Hardware Faults

Before presenting the performance of the DDHR systems, we start by analyzing the bit-level errors in the extracted features due to the fault-affected processors. In Fig. 7, we see that the bit error rates (left) are very high for both applications, ranging from 20 % to 50 % for most test cases. Moreover, the bit-level errors occur on high order bits, as shown by the large feature-error magnitudes normalized to the true feature values (right) [the error magnitudes are shown for three fault levels plotted as the normalized root mean square (RMS) value of the errors, calculated as in Eq. 3]. Namely, we see that for many cases, the error values are larger than the true features themselves. However, as shown in the following section, the DDHR is able to restore system performance despite the severe bit-level errors.

$$\frac{\mathbf{RMS(error)}}{\mathbf{RMS(true\ value)}} = \frac{\sqrt{\sum_{\mathbf{i=1}}^{\mathbf{M}}(\mathbf{F_{EA}} - \mathbf{F_{TV}})^2/\mathbf{M}}}{\sqrt{\sum_{\mathbf{i=1}}^{\mathbf{M}}\mathbf{F_{TV}^2}/\mathbf{M}}} \tag{2}$$

$\mathbf{F_{EA}}$: **Value of error-affected feature**
$\mathbf{F_{TV}}$: **True value (error-free) of feature**
M: Total number of feature vectors

5.3.2 DDHR Performance

Figure 8 shows the system performance for both the seizure detector (a)-(c) and the arrhythmia detector (d)-(e). The error-aware model (in red) is evaluated against the non-error-aware model (in blue), by varying the fault level. Further, at each fault level, five test cases are employed. The plot markers indicate average value, while error bars indicate maximal values. For both applications, without an error-aware model, the performance degrades immediately with hardware faults. On the other hand, with an error-aware model within a DDHR system, the performance is restored up to the high fault levels, despite the large bit-level errors observed (Section 5.3.1). For the seizure detector, the performance of all test cases with less than 20 stuck-at faults (0.02 % of total circuit nodes) is consistently restored to the

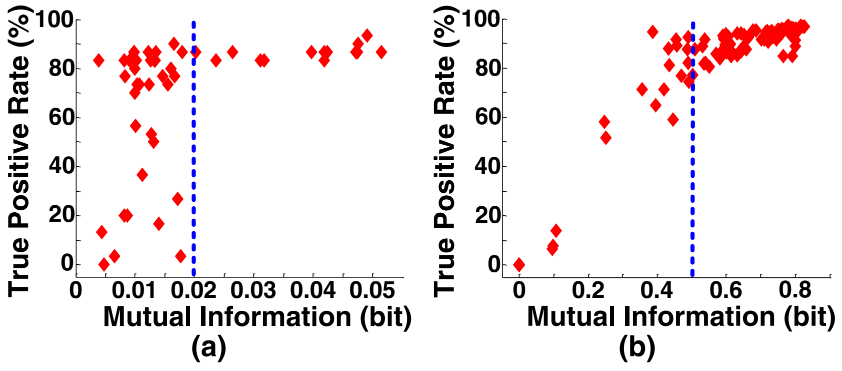


Figure 10 Plotting of the true positive rate of DDHR versus mutual information for (**a**) EEG seizure detector and (**b**) ECG arrhythmia detector.

level of the fault-free system. Even beyond that fault level, many test cases exhibit restored performance. Similarly, for the arrhythmia detector, the performance of all test cases with less than 480 stuck-at faults (3 % of total circuit nodes) is consistently restored to the level of the fault-free system.

The above experiments suggest that the performance of DDHR systems is not limited by bit level errors. Despite the high bit error rates and large bit-level error magnitudes, DDHR is capable of restoring system performance. In order to understand the fundamental limit for DDHR systems, we thus look beyond bit-level error metrics, exploring more fundamental information-level metrics. This is discussed in the following section.

5.4 Mutual Information - a Fundamental Limit

As a fundamental information-level metric, we focus on mutual information. Specifically, we consider the mutual information between the feature vectors derived from the fault-affected processor and the associated class membership. Effectively, this tells us how much information we have about class membership given information about the feature vector (and vice-versa) [47]. As an example, Fig. 9 shows the EEG feature-vector distributions extracted from both a fault-free processor and a fault-affected processor whose performance could not be restored. We see in the latter case that low mutual information implies degraded separation between the classes.

Mutual information is calculated as the reduction of class-label entropy given knowledge of the feature vectors (Eq. 3). Here, X represents feature vectors and Y represents class labels. To calculate $H(Y)$, we need to know the probability distribution of class labels $p(y)$ (Eq. 4), which can be easily estimated from the training samples. To calculate $H(Y|X)$, however, we need to know the probability distribution of feature vectors $p(x)$, together with conditional probability of class labels given feature vectors $p(y|x)$. To estimate this, we first reduce the dimensionality of feature space to d through PCA ($d = 8$ for EEG features and $d = 4$ for ECG features). Then we discretize each feature dimension into n bins ($n = 2$ for EEG features and $n = 4$ for ECG features). We thus get a total of 256 bins for EEG feature space and 625 bins for ECG feature space. Now we can use the discretized bins to estimate $p(x)$ and $p(y|x)$ for calculating $H(y|x)$.

$$\mathbf{I(X;Y) = H(Y) - H(Y|X)} \tag{3}$$

$$\mathbf{H(Y) = -\sum_{y=-1,1} p(y)\log_2 p(y)} \tag{4}$$

$$\mathbf{H(Y|X) = -\sum_{x\in X} p(x) \sum_{y=-1,1} p(y|x)\log_2 p(y|x)} \tag{5}$$

The mutual information between the feature vectors (obtain from experiments) and their class membership is computed for the two applications. To analyze the performance achieved by the DDHR systems, Fig. 10 shows scatter plots of the mutual information with respect to the system performance (true positive rate is shown, and training is performed to set the true negative rate to 95 %). As shown, with DDHR, the achieved performance strongly correlates with mutual information. For seizure detector, mutual information above 0.02 bit indicates consistently good classification performance. While for arrhythmia detector, mutual information above 0.5 bit indicates consistently good performance.

6 Conclusion

Moore's Law has been a key aspect for deriving broad and substantial advantages within CMOS integrated circuits. One consequence has been an explosion in computing applications, particularly making computational systems available ubiquitously in the ambient and mobile domains. In these domains, energy constraints play a critical role. As a result, many architectural approaches have focused on leveraging Moore's Law towards enhanced energy efficiency. A powerful approach in this regard has been hardware specialization. Hardware specialization is well known, and has been particularly effective when a small set of computations have large energy impact on the overall system. These computations can then be mapped to accelerators for substantial overall benefits. Unfortunately, the key challenge brought on by Moore's Law is hardware reliability, due increasingly aggressive fabrication of transistors and interconnects. Just like accelerators exploit asymmetry in energy impact across the operations within an application, we can also envision asymmetry in reliability impact across the operations. We describe an approach known as data-driven hardware resilience (DDHR), which utilizes a machine-learning algorithm to train an inference model to embedded data in the presence of errors caused by hardware faults. We show that, similar to energy, specialization, this time to ensure fault-free operation within the inference kernels, can substantially benefit overall system resilience. In this way, Moore's Law can continue to deliver benefits on the architectural level, including enhanced reliability to address degraded reliability at the device level.

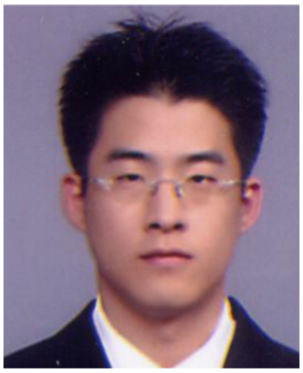
Acknowledgments Support is provided by SRC, NSF (CCF-1253670), as well as Center for Future Architectures Research (C-FAR) and Systems on Nanoscale Information fabriCs (SONIC), two of the six SRC STARnet Centers, sponsored by MARCO and DARPA.

References

1. Dubey, P. (2005). Recognition, mining and synthesis moves computers to the era of tera. *Technology Intel Magazine*, *9*(2), 1–10.
2. Verma, N., Shoeb, A., Bohorquez, J., Dawson, J., Guttag, J., Chandrakasan, A.P. (2010). A micro-power EEG acquisition SoC with integrated feature extraction processor for a chronic seizure detection system. *IEEE Journal of Solid-State Circuits*, *45*(4), 804–816.
3. Lee, K.H., & Verma, N. (2012). A 1.2–0.55 V general-purpose biomedical processor with configurable machine-learning accelerators for high-order, patient-adaptive monitoring. In *Proceedings of the IEEE in ESSCIRC (ESSCIRC)* (pp. 285–288).
4. Fan, K., Kudlur, M., Dasika, G., Mahlke, S. (2009). Bridging the computation gap between programmable processors and hardwired accelerators. In *IEEE 15th International Symposium on High Performance Computer Architecture, 2009. HPCA 2009* (pp. 313–322). IEEE.
5. Manavski, S.A. (2007). CUDA compatible GPU as an efficient hardware accelerator for AES cryptography. In *IEEE International Conference on Signal Processing and Communications, 2007. ICSPC 2007* (pp. 65–68). IEEE.
6. Guerdoux-Jamet, P., & Lavenier, D. (1997). SAMBA: hardware accelerator for biological sequence comparison. *Comput. Appl. Biosci.: CABIOS*, *13*(6), 609–615.
7. Lee, K., & Verma, N. (2013). A low-power processor with configurable embedded machine-learning accelerators for high-order and adaptive analysis of medical-sensor signals. *IEEE Journal of Solid-State Circuits*, *48*(7), 1625–1637.
8. Shi, X., Hsu, S., Chen, F., Hsu, M., Socha, R., Dusa, M. (2002). Understanding the forbidden pitch phenomenon and assist feature placement. In: *Proceedings SPIE* (vol. 4689, pp. 985–996).
9. Bohr, M. (2009). The new era of scaling in an SoC world. In *IEEE Solid-State Circuits Conference-Digest of Technical Papers, 2009. ISSCC 2009* (pp. 23–28). IEEE International.
10. McPherson, J.W. (2006). Reliability challenges for 45nm and beyond. In *Proceedings of the 43rd Annual Design Automation Conference* (pp. 176–181). ACM.
11. Zhang, J., Lin, A., Patil, N., Wei, H., Wei, L., Wong, H., Mitra, S. (2012). Robust digital VLSI using carbon nanotubes. *IEEE Transaction on Computer-aided Design of Integrated Circuits and Systems*, *31*(4), 453–471.
12. Shulaker, M.M., Hills, G., Patil, N., Wei, H., Chen, H.-Y., Wong, H.-S.P., Mitra, S. (2013). Carbon nanotube computer. *Nature*, *501*(7468), 526–530.
13. Chang, L., Choi, Y.-k., Ha, D., Ranade, P., Xiong, S., Bokor, J., Hu, C., King, T.-J. (2003). Extremely scaled silicon nano-CMOS devices. *Proceedings of IEEE*, *91*(11), 1860–1873.
14. Ribes, G., Rafik, M., Roy, D. (2007). Reliability issues for nanoscale CMOS dielectrics. *Microelectronic Engineering*, *84*(9), 1910–1916.
15. Henkel, J., Bauer, L., Dutt, N., Gupta, P., Nassif, S., Shafique, M., Tahoori, M., Wehn, N. (2013). Reliable on-chip systems in the nano-era: lessons learnt and future trends. In: *Proceedings of the 50th Annual Design Automation Conference* (p. 99). ACM.
16. Leem, L., Cho, H., Bau, J., Jacobson, Q.A., Mitra, S. (2010). ERSA: Error resilient system architecture for probabilistic applications. In *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2010* (pp. 1560–1565). IEEE.
17. Shanbhag, N. (2002). Reliable and energy-efficient digital signal processing. In: *Proceedings of the 39th annual Design Automation Conference* (pp. 830–835). ACM.
18. Ernst, D., Kim, N.S., Das, S., Pant, S., Rao, R., Pham, T., Ziesler, C., Blaauw, D., Austin, T., Flautner, K., et al. (2003).
19. Shanbhag, N.R., Mitra, S., Veciana, G.d., Orshansky, M., Marculescu, R., Roychowdhury, J., Jones, D., Rabaey, J.M. (2008). The search for alternative computational paradigms. *IEEE Design and Test of Computers*, *25*(4), 334–343.
20. Shanbhag, N.R., Abdallah, R.A., Kumar, R., Jones, D.L. (2010). Stochastic computation. In: *Proceedings of the 47th Design Automation Conference* (pp. 859–864). ACM.
21. Verma, N., Lee, K.H., Jang, K.J., Shoeb, A. (2012). Enabling system-level platform resilience through embedded data-driven inference capabilities in electronic devices," In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2012* (pp. 5285–5288). IEEE.
22. Wang, Z., Lee, K.H., Verma, N. Overcoming computational errors in low-power sensing platforms through embedded machine-learning kernels. To appear on IEEE Transactions on Very Large Scale Integration (VLSI) systems.
23. Chippa, V.K., Mohapatra, D., Raghunathan, A., Roy, K., Chakradhar, S.T. (2010). Scalable effort hardware design: exploiting algorithmic resilience for energy efficiency. In: *Proceedings of the 47th Design Automation Conference* (pp. 555–560). ACM.
24. Sze, V., & Chandrakasan, A.P. (2011). A highly parallel and scalable CABAC decoder for next generation video coding. In *IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2011* (pp. 126–128). IEEE.
25. Owens, J.D., Houston, M., Luebke, D., Green, S., Stone, J.E., Phillips, J.C. (2008). GPU computing. *Proceedings of IEEE*, *96*(5), 879–899.
26. Kwong, J., & Chandrakasan, A.P. (2011). An energy-efficient biomedical signal processing platform. *IEEE Journal of Solid-State Circuits*, *46*(7), 1742–1753.
27. Sun, F., Ravi, S., Raghunathan, A., Jha, N.K. (2004). Custom-instruction synthesis for extensible-processor platforms, (Vol. 23.
28. Dusa, M., Quaedackers, J., Larsen, O.F., Meessen, J., van der Heijden, E., Dicker, G., Wismans, O., de Haas, P., van Ingen Schenau, K., Finders, J., et al. (2007). Pitch doubling through dual-patterning lithography challenges in integration and litho budgets. In *Advanced Lithography. International Society for Optics and Photonics* (pp. 65200G–65200G).
29. Mansfield, S.M., Liebmann, L.W., Molless, A.F., Wong, A.K. (2000). Lithographic comparison of assist feature design strategies. In *Microlithography 2000. International Society for Optics and Photonics* (pp. 63–76).
30. Horowitz, M., Alon, E., Patil, D., Naffziger, S., Kumar, R., Bernstein, K. (2005). Scaling, power, and the future of CMOS. In *Electron Devices Meeting, 2005. IEDM Technical Digest. IEEE International.* (pp. 7–15). IEEE.
31. Austin, T., Bertacco, V., Blaauw, D., Mudge, T. (2005). Opportunities and challenges for better than worst-case design. In *Proceedings of the 2005 Asia and South Pacific Design Automation Conference* (pp. 2–7). ACM.
32. Shooman, M.L. (2002). *N-modular redundancy. Reliability of Computer Systems and Networks: Fault Tolerance, Analysis, and Design* (pp. 145–201).
33. Kim, E.P., & Shanbhag, N.R. (2012). Soft n-modular redundancy. *IEEE Transactions on Computers*, *61*(3), 323–336.
34. Hegde, R., & Shanbhag, N.R. (1999). Energy-efficient signal processing via algorithmic noise-tolerance. In *Proceedings of the 1999 international symposium on Low power electronics and design* (pp. 30–35). ACM.
35. Hedge, R., & Shanbhag, N.R. (2001). Soft digital signal processing.
36. Pillai, P., & Shin, K.G. (2001). Real-time dynamic voltage scaling for low-power embedded operating systems. In *ACM SIGOPS Operating Systems Review* (vol. 35(5), pp. 89–102). ACM.

37. Maglaveras, N., Stamkopoulos, T., Diamantaras, K., Pappas, C., Strintzis, M. (1998). ECG pattern recognition and classification using non-linear transformations and neural networks: a review. *International Journal of Medical Informatics*, *52*(1), 191–208.
38. Lee, K.H., Kung, S.-Y., Verma, N. (2011). Improving kernel-energy trade-offs for machine learning in implantable and wearable biomedical applications. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2011* (pp. 1597–1600). IEEE.
39. Goldberger, A.L., Amaral, L.A., Glass, L., Hausdorff, J.M., Ivanov, P.C., Mark, R.G., Mietus, J.E., Moody, G.B., Peng, C.-K., Stanley, H.E. (2000). Physiobank, physiotoolkit, and physionet components of a new research resource for complex physiologic signals.
40. Shoeb, A.H. (2009). Application of machine learning to epileptic seizure onset detection and treatment. Ph.D. dissertation, Massachusetts Institute of Technology.
41. Shih, E., & Guttag, J. (2008). Reducing energy consumption of multi-channel mobile medical monitoring algorithms. In *Proceedings of the 2nd International Workshop on Systems and Networking Support for Health Care and Assisted Living Environments* (p. 15). ACM.
42. Shoeb, A.H., & Guttag, J.V. (2010). Application of machine learning to epileptic seizure detection. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)* (pp. 975–982).
43. Übeyli, E.D. (2007). ECG beats classification using multiclass support vector machines with error correcting output codes. *Digital Signal Processing*, *17*(3), 675–684.
44. Wang, Z., & Verma, N. (2014). Error adaptive classifier boosting (EACB): exploiting data-driven training for highly fault-tolerant hardware. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2014*.
45. Schertz, D.R., & Metze, G. (1972). A new representation for faults in combinational digital circuits. *IEEE Transactions on Computers*, *100*(8), 858–866.
46. Rabaey, J.M., Chandrakasan, A.P., Nikolic, B. (2002). In *Digital integrated circuits* (vol. 2). Englewood Cliffs: Prentice hall.
47. Gallager, R.G. (2008). In *Principles of digital communication* (vol. 1). Cambridge: Cambridge University Press.

Zhuo Wang received his B.S. degree in Microelectronics from Peking University, Beijing, China, in 2011. He received his M.S. degree in 2013 and is currently a Ph.D. Candidate in the Department of Electrical Engineering at Princeton University, Princeton, New Jersey, US. His research interest is on robust system design. More specifically, he is interested in designing robust VLSI systems for highly-energy-constrained applications, and how machine-learning techniques can be exploited, not only to model sensor signals, but also hardware errors affecting the platform.



Kyong Ho Lee received the B.S. degree from the Korea Advanced Institute of Science and Technology (KAIST) in 2004, M.S. degree from Stanford University in 2009, and Ph.D. degree from Princeton University in 2013 in electrical engineering. He is a co-recipient of Qualcomm Innovation Fellowship (QInF) 2011.

He is currently working at Samsung Research America, Dallas, as a hardware engineer. His research interests include ultra-low energy circuit design specialized in vision and wearable applications, machine-learning techniques and algorithms for high energy efficiency, deep-learning applications in mobile domain, and sensor-fusion applications.

Naveen Verma received the B.A.Sc. degree in electrical and computer engineering from the University of British Columbia, Vancouver, BC, Canada, in 2003 and the M.S. and Ph.D. degrees in electrical engineering from the Massachusetts Institute of Technology, Cambridge, MA, USA, in 2005 and 2009, respectively. Since July 2009, he has been an Assistant Professor of Electrical Engineering at Princeton University, Princeton, NJ, USA. His research focuses on ultra-low-power integrated circuits and systems with an emphasis on sensing applications. Of particular importance is the use of emerging devices for the creation of functionally diverse systems and the use of advanced signal-analysis frameworks for low-power inference over embedded signals. On the circuit level, his focus spans low-voltage digital logic and SRAMs, low-noise analog instrumentation and data-conversion, and integrated power management. Prof. Verma is corecipient of 2008 ISSCC Jack Kilby Award for Outstanding Student Paper, and 2006 DAC/ISSCC Student Design Contest Award. He is recipient of the Alfred Rheinstein Junior Faculty Award at Princeton and the 2013 NSF CAREER award.