This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

Overcoming Computational Errors in Sensing Platforms Through Embedded Machine-Learning Kernels

Zhuo Wang, *Student Member, IEEE*, Kyong Ho Lee, *Student Member, IEEE*, and Naveen Verma, *Member, IEEE*

***Abstract*— We present an approach for overcoming computational errors at run time that originate from static hardware faults in digital processors. The approach is based on embedded machine-learning stages that learn and model the statistics of the computational outputs in the presence of errors, resulting in an error-aware model for embedded analysis. We demonstrate, in hardware, two systems for analyzing sensor data: 1) an EEG-based seizure detector and 2) an ECG-based cardiac arrhythmia detector. The systems use a small kernel of fault-free hardware (constituting <7.0% and <31% of the total areas respectively) to construct and apply the error-aware model. The systems construct their own error-aware models with minimal overhead through the use of an embedded active-learning framework. Via an field-programmable gate array implementation for hardware experiments, stuck-at faults are injected at controllable rates within synthesized gate-level netlists to permit characterization. The seizure detector demonstrates restored performance despite faults on 0.018% of the circuit nodes [causing bit error rates (BERs) up to 45%], and the arrhythmia detector demonstrates restored performance despite faults on 2.7% of the circuit nodes (causing BERs up to 50%).**

***Index Terms*— Embedded sensing, fault tolerance, hardware resiliency, machine learning, run-time error correction.**

I. Introduction

POWERFUL algorithms have emerged from the domain of machine learning that enable data-driven methods for modeling and analyzing application signals. The strength of machine learning is that it enables the creation of high-order models from previously observed data. This is valuable to embedded sensing applications for two reasons [1]–[3]: 1) it enables analysis models that are well-suited for classification, recognition, and mining functions, which are becoming increasingly prominent [4] and 2) it overcomes the need for analytical models, which are often inviable for the physically-complex embedded signals of interest. The focus of this paper is to extend the potential benefits of machine learning, beyond enabling analysis models for application signals, to mitigating the need for design margining against a significant class of hardware faults, namely static faults [e.g., opens/shorts caused by lithographic defects, logic gate static noise margin (SNM) violations caused by transistor variability, and so on]. They impose substantial costs on system resources (area, energy, and so on) due to the need for design margining [5]–[8]. While such faults can be impossible to predict and can manifest as severe, complex, and highly transient errors, we show that machine-learning (ML) methods can enable the construction of an analysis model in the presence of the resulting errors, overcoming the impact of static faults. We describe architectures that utilize this principle toward system-level resilience with little overhead in terms of energy and hardware. The approach is referred to as data-driven hardware resilience (DDHR), and the resulting model is referred to as an error-aware model. Preliminary simulations for DDHR were presented in [9]. Here, we present hardware experiments along with an information-based analysis of the approach. While it is expected that such an approach can benefit system resources by alleviating the need for design margining, the precise benefit is highly dependent on the implementation and thus difficult to systematically characterize; for quantitative characterization, we thus focus our analysis by considering system performance directly with respect to fault rates. Further, we propose an architecture that enables on-line construction of the error-aware model and also enables assessment of whether such a model can successfully restore system performance. The specific contributions of this paper are as follows.

1) We demonstrate in hardware, via field-programmable gate array (FPGA) emulation, the proposed architecture utilizing DDHR in two systems for classification: 1) an EEG-based seizure detector and 2) an ECG-based cardiac-arrhythmia detector. FPGA implementation permits characterization through the injection of hardware faults on randomized nodes in the circuit at controllable rates. We show that DDHR imposes low operational overhead and is able to restore performance to the level of a fault-free system yet in the presence of BERs up to 45% and 50%, respectively [due to faults on >20 (out of 110k) circuit nodes and >480 (out of 18k) circuit nodes].
2) We present an architecture that includes a subsystem for training the error-aware model entirely within the platform at run time. Model training requires two items:

Manuscript received May 23, 2013; revised December 21, 2013 and June 13, 2014; accepted July 21, 2014. This work was supported in part by Semiconductor Research Corporation, in part by the National Science Foundation under Grant CCF-1253670, in part by MARCO, and in part by the Defense Advanced Research Projects Agency, through the Center for Future Architectures Research and Systems on Nanoscale Information Fabrics.

Z. Wang and N. Verma are with the Department of Electrical Engineering, Princeton University, Princeton, NJ 08544 USA (e-mail: zhuow@princeton.edu; nverma@princeton.edu).

K. H. Lee was with the Department of Electrical Engineering, Princeton University, Princeton, NJ 08544 USA, and now with Samsung Research America, Dallas, TX, USA (e-mail: kyongho.l@samsung.com).

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/TVLSI.2014.2342153

1063-8210 © 2014 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

1) training data, whose statistics correctly represent variances both due to application signals and errors and 2) class labels for the training data, from which a decision function can be specified. While the training data is readily available from the fault-affected hardware itself, we show how training labels can be effectively estimated within the architecture. Further, we show how the impact of the training subsystem within the architecture can be minimized using an active-learning algorithm, which selects the optimal instances for training from among the sensed data. We show that performance very close to an ideally trained error-aware model is thus achieved with a small fraction of the sensed data.

3) We show that thanks to flexible data-driven modeling, the performance of DDHR is ultimately limited by the mutual information (MI) exhibited by the error-affected data, rather than bit-error metrics. MI is a fundamental information metric derived from the Shannon entropy that measures how much information is shared between two variables (in this case, the error-affected data and the system output). An important insight that emerges is that MI is often retained despite large bit-level errors. This suggests that substantial error-tolerance can be achieved through DDHR. We propose a proxy to MI that can be computed on-line to predict when the hardware faults will be too severe to overcome through DDHR.

The remainder of this paper is organized as follows. Section II presents background, both on error-tolerant architectures and on ML frameworks for embedded data analysis. Section III presents an overview of DDHR, as well as the fault types targeted, and describes a system architecture that incurs minimal hardware overhead for model training. Section IV presents the experimental approach and Section V presents results from the hardware experiments. Finally, Section VI presents the conclusion.

II. Background

The DDHR follows the trend of seeking system-level approaches to hardware resilience. The background in this area as well as the ML approach taken by DDHR is presented below.

A. Architectures for Hardware Resilience

The CMOS manufacturing has always been susceptible to defects, causing hardware faults that have traditionally resulted in direct yield loss. In the past, defects have been managed through design margining. However, technology scaling as well as transistor operation in aggressive low-energy and/or high-performance regimes has dramatically exacerbated the sources of hardware faults. Lithography errors, process fluctuations (random dopant fluctuations, and so on), and device degradations (hot-carrier injection, and so on) [10], [11] have progressed to levels where the overheads of traditional design margining are too severe [5]. As a result, system- or application-level approaches to hardware resilience have gained substantial interest due to their potential to perform better than design margining [8], [12]. The research in this area can generally be categorized into two approaches: error correction and error tolerance.

Error correction attempts to undo the effects of hardware faults typically by detecting and correcting computational errors. While error detection and correction has been widely applied within specific circuits, such as static random-access memories [13], approaches such as Razor [14] have extended the idea to CPUs through microarchitectural enhancements in the processor pipeline for error detection and replay. Such an approach can improve efficiency by permitting voltage/frequency scaling limited by the overhead of replay rather than the overhead of margining against the first occurrence of errors [15].

Error tolerance allows the system to make errors, but attempts to minimize system-level impact; most importantly, this requires ensuring correct functionality. This often incurs reduced overhead by exploiting the idea that many emerging applications inherently exhibit some error tolerance [8]. While simple voting systems based on redundancy [16] incur high hardware and power costs, algorithmic enhancements that exploit the statistics of the errors have shown to substantially improve efficiency; an illustrative example is algorithmic noise tolerance, wherein computations are performed by redundant blocks that explicitly have differing error statistics [17]. Other approaches that utilize redundancy efficiently include N-modular redundancy (NMR) and NMR with soft voting [18]. Aside from application-specific architectures based on redundancy, error-tolerant principles have also been applied to programmable platforms. Here, the critical concern is ensuring error-free control flow while permitting errors for data computation. Error resilient system architecture employs hardware asymmetry, wherein control-flow tasks are delegated to a reliable (fault-free) core, while data computation is delegated to relaxed reliability cores [19]. Approaches have also applied asymmetry on a finer grain, for instance, by synthesizing a CPU core to have increased timing slack on specific paths [20] or by introducing hardware support to provide error protection of specific instructions [21].

These approaches have variously demonstrated efficient handling of BERs on the order of 0.1% caused due to faults in the circuit [19]. The work presented here aims to achieve much higher levels of error tolerance through data-driven statistical modeling capabilities introduced by ML kernels. It has been previously noted that frameworks based on statistical modeling inherently yield some degree of resilience against low-order bit errors [22]. However, by exploiting flexible data-driven modeling capabilities, we aim to approach a more fundamental limit set by the level of information retained within the error-affected data for an analysis of interest. Since ML kernels are increasingly being employed within applications, such capabilities can incur very little operational overhead.

B. Embedded Analysis Using Machine-Learning Kernels

As mentioned, machine learning is gaining popularity for the analysis of embedded signals because it enables the construction of strong models from previous observations of

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

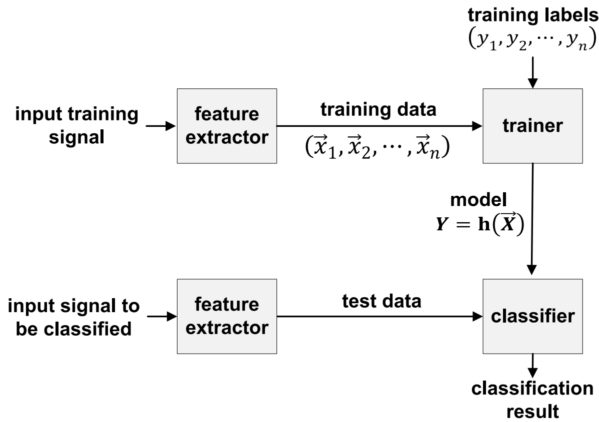


Fig. 1. Illustration of the SVM, which is a supervised machine-learning framework for classification.

the data, thus precluding the need for rigorous analytical modeling. A critical premise in machine learning is that to maintain the integrity of the model, the data to be analyzed must exhibit the same statistics as the previous observations. This forms the basis for the proposed approach and it directs the types of faults that can ultimately be handled.

While a wide range of machine learning frameworks exist, we focus on a particular framework to illustrate the DDHR concept within applications for sensor-data classification, an increasingly prominent function in embedded applications [4]. The support vector machine (SVM) is a supervised machine-learning framework for classification that has gained popularity due to its efficiency [1]–[3], [23] and its ability to form robust models with minimal training data [24], [25]. Fig. 1 shows the SVM framework. Supervised learning algorithms consist of two phases: 1) training, wherein a model for classification is constructed using training data and training-data labels that specify class membership and 2) testing, wherein the constructed model is applied within a decision function to classify in-coming data in real time. During both phases, the data is provided in the form of a feature vector. In general, feature extraction is an important step that reduces the complexity of the application data through processing to extract the critical parameters believed to be most informative for classification. For SVM training, the labels $(y_1, y_2, \ldots, y_n)$ are typically binary parameters provided by an expert. These are then used with the training feature vectors $(\vec{x}_1, \vec{x}_2, \ldots, \vec{x}_n)$ to effectively establish a hyperplane that forms a decision boundary for classification in the feature space. The hyperplane is represented by choosing instances of the training feature vectors that lie near the edges of the class distributions and then maximizing the marginal distance between instances from the different classes. The chosen instances are called the support vectors and are thus used in the classification decision function

$$K(\vec{x}, \vec{z}) = \exp\left(-\frac{\|\vec{x} - \vec{z}\|^2}{2\sigma^2}\right). \quad (1)$$

Fig. 2(a) shows the decision function and an effective decision boundary (in a 2-D feature space) for a binary classifier; $\vec{x}$ represents the feature data to be classified, $\vec{x}_j$, y_j

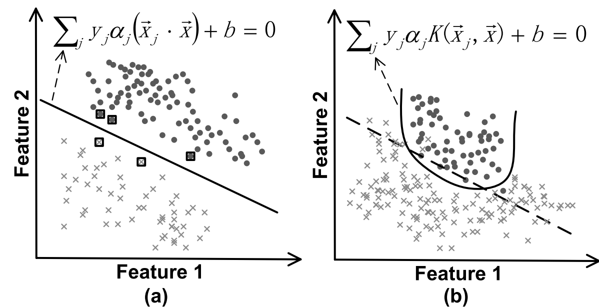


Fig. 2. SVM decision function and effective decision boundary in the feature space using (a) linear kernel and (b) nonlinear kernel to increase the flexibility of the effective decision boundary.

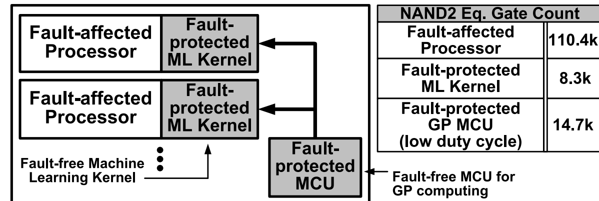


Fig. 3. Architecture where a small kernel of fault-protected hardware (shown shaded) enables resilient system-level performance; estimates of block gate counts from our system are shown (MCU used is MSP430 from OpenCores [26]).

represents the support vectors and their labels, and α_j and b represent parameters obtained from training. A critical strength of the SVM is its ability to construct highly flexible decision boundaries, enabling classification of data with complex distributions in the feature space. This flexibility, beyond the linear decision function shown in Fig. 2(a), is achieved by introducing a nonlinear kernel in the decision function, which has the effect of mapping the feature vectors to higher dimensionality. As shown in Fig. 2(b), where the nonlinear kernel is represented by K, this enables greater flexibility in the effective decision boundary, thus improving discrimination between the classes. A commonly applied kernel function is the radial-basis function kernel that uses an exponential non-linearity as shown in (1), yielding a mapping to infinite dimensionality for a very high degree of flexibility.

III. Details of the Approach

The DDHR aims to achieve error resilience over a large, fault-affected architecture by leveraging a small kernel of fault-protected hardware; the impact of the fault-protected hardware is meant to be minimal both in hardware (area) and energy. Fig. 3 shows the architecture for the systems we demonstrate. The fault-affected blocks consist of digital processors for performing feature extraction during real-time operation. The fault-protected blocks consist of: 1) ML kernels for performing SVM classification during real-time operation and 2) a small, general-purpose microcontroller (MCU) for infrequent or one-time training of the error-aware model, which is not required during real-time operation. The MCU is implemented via an OpenMSP core [26].

A. System Rationale

Employing design margining in the usual way to address hardware faults imposes costs, both real and in terms of system

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

	Seizure Detection*	Arrhythmia Detection**
Feature Extractor w/ MCU	39.5 µJ	88.4 µJ
Feature Extractor w/ Acc.	3.33 µJ	1.61 µJ
SVM Classifier w/ Acc.	1.11 µJ	0.202 µJ

* Energy for each feature vector with one channel of EEG; RBF kernel is used for SVM
** Energy for each feature vector with ECG; Poly kernel with low-energy formulation is used for SVM

Fig. 4. Comparison of classification energy with MCU and with accelerator based system architecture for EEG seizure detector and ECG arrhythmia detector. Numbers are derived from [30].

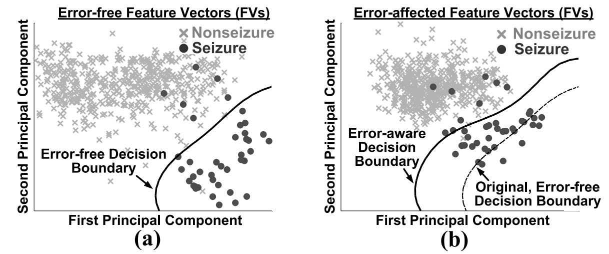


Fig. 5. Feature vectors for a seizure-detection system (shown in two-dimensions via PCA for visualization). (a) Distribution variances are only due to the application signals and are modeled by the original decision boundary. (b) Variances are also due to computational errors, making a new decision boundary necessary by training on the error-affected data.

resources, e.g., area (to address defects due to nanoscale lithography [27], [28]), energy (to address logic-gate SNM violations due to transistor variability in low-voltage designs [29]), and so on. The aim of DDHR is to enable hardware relaxations by tolerating errors in the outputs of the fault-affected blocks.

With regards to area, as an example, in Fig. 3, we show the gate counts for the various blocks in one of the demonstrated systems (EEG-based seizure detector, whose experimental performance is presented in Section V-A). The gate counts are derived from register-transfer level (RTL) synthesis to an application specified integrated circuit (ASIC) standard-cell library. As shown, the fault-affected blocks dominate the architecture, making their tolerance to errors the critical concern. Generally speaking, the MCU can be small, thanks to the need for low performance (MIPS) and simple instruction set, which is made possible since it does not perform real-time functions (as described in Section III-C, the MCU implements model training, including temporary classification for training-label generation and decision-boundary fitting to an acquired training set). The gate counts presented for the fault-affected processor and the fault-protected ML kernel both scale with the number of sensor channels (which can be as high as 18 for the seizure-detection system); the gate count for the MCU, on the other hand, does not scale with the sensor channels and can thus be further amortized. The objective of the architecture is thus to address the hardware-dominating fault-affected blocks by leveraging the comparatively small fault-protected kernel.

With regards to energy, as an example, in Fig. 4, we show the estimated energy for various computations in the demonstrated systems (EEG-based seizure detector and ECG-based arrhythmia detector). The energy estimates are derived from silicon measurements of a fabricated custom CMOS IC [30] that integrates a CPU (based on the same OpenMSP core) and an accelerator, which can be configured as an SVM classifier or finite-impulse response (FIR) filter. The energies for a hardware-based (accelerator) implementation of both the feature extractor and SVM classifier are shown. These constitute real-time computations in the proposed architecture and are mapped to fault-affected and fault-protected blocks, respectively. In addition to these, the architecture requires nonreal-time computations for training the error-aware model via fault-protected hardware. This is achieved using the MCU. As described in Section III-C, this training itself requires error-free feature extraction and SVM classification to derive training label estimates. In Fig. 4, we thus also provide the energy of a software-based implementation (on the MCU) for the feature extractor. This energy is 12× and 55× higher than the implementation mapped to fault-affected hardware and is thus only viable for the temporary, nonreal-time computations during training. As an example, for the permanent faults considered in this paper, training is required only once. The objective of the architecture is thus to address the fault-affected blocks used for on-going real-time computations by leveraging the amortized energy of the MCU for model training.

B. Concept of Error-Aware Modeling and Types of Faults Addressed

The key to overcoming errors is the ML stages, which utilize decision functions constructed at run time using the outputs of the error-affected processors. As mentioned, an SVM is used as the ML stage for classification. For a given instance of a fault-affected processor, the statistics of the output data is generated both by variances in the application signals and by the specific faults leading to computational errors. Accordingly, using data from the fault-affected processor itself, an error-aware model that suitably represents the manifestations of the errors can be effectively constructed. The problem of how to construct such a model at run time for a given instance of the system is discussed in Section III-C.

In Fig. 5, we show the concept of an error-aware model. An actual feature data is shown from hardware measurement of the seizure-detection system described in Section V-A [though the actual data is of much higher dimensionality, principal component analysis (PCA) has been used in the plots to aid visualization]. Fig. 5(a) shows the feature vectors extracted from a fault-free system as well as the decision boundary derived from the error-free data. Fig. 5(b) shows the feature vectors derived using a fault-affected system. As shown, the feature distributions are altered due to the computational errors. The original decision boundary, drawn as a dotted line, misclassifies many seizure vectors. On the other hand, an error-aware model, constructed from the error-affected feature vectors is drawn as a solid line and suggests that classification performance can be restored.

The proposed approach to error-aware modeling has important implications on the types of faults that can be addressed by the system. In general, a requirement with ML algorithms is that the statistics between the training set and the testing

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

data must remain consistent. As a result, faults that are static or that can be tracked through training can be addressed by the proposed approach, while dynamic faults that do not preserve the statistics generally cannot. It is important to note here that the errors generated by static faults can be, and typically are, highly transient. What is critical is that the statistics of the errors does not change between training and testing.

Indeed, many of the critical faults concerning advanced CMOS in low-energy operating regimes are of such a nature and can be addressed. These include opens and shorts caused by defects in nanoscale lithography [31], logic gate SNM violations at low voltages caused due to transistor variations[1] [32], and so on. It is also worth noting that many faults that critically limit the adoption of post-CMOS devices are also static. For instance, carbon nanotube circuits suffer from nanotube-density variations, where the critical effect is complete loss of channel nanotubes in logic gates, preventing functional output transitions [33]. For the experimental evaluations described in Section V-A, we employ a stuck-at-1/0 fault model; this appropriately represents the mentioned faults [29], [34]. Previous work exploring the statistics generated by timing violations [6], [35] suggests that such faults can likely also be addressed by the approach; however, our experiments do not model such faults, and characterization of their effects is not covered in this paper.

It is also noted that in addition to computational errors due to digital faults, a practical system will also be affected by noise in the sensor and analog front end. In general, the ML algorithms will employ training using data from a typical acquisition subsystem and will attempt to construct an inference model that maximizes the margin against such noise in the data. In the case of an SVM, for instance, this is achieved by finding a decision boundary that maximizes the marginal distance from the classes. Examples of practical systems incorporating analog front ends with a back-end classifier (based on an SVM) include [36]–[38].

C. Active-Learning System for On-Line Model Construction

As mentioned above, error-aware modeling relies on training data from the given instance of a fault-affected processor itself. We thus focus on an approach where the models are constructed dynamically at run time as sensor data is being processed. While training data for this is readily available within the system, training-data labels are also required in a supervised learning framework such as an SVM. Here, we propose a system wherein the error-aware model can be constructed on line without the need for external labeling.

The system is shown in Fig. 6. It has two aspects: 1) the permanent fault-affected system, whose operation dominates the architecture and whose outputs will also be used as the training feature vectors and 2) a temporary fault-protected auxiliary system implemented on the MCU that is used to estimate the training labels and perform model training. The auxiliary system estimates training labels by implementing

[1]While the device-level variations are static, such faults can potentially have some dependence on temperature and voltage; this could necessitate training over the various conditions.

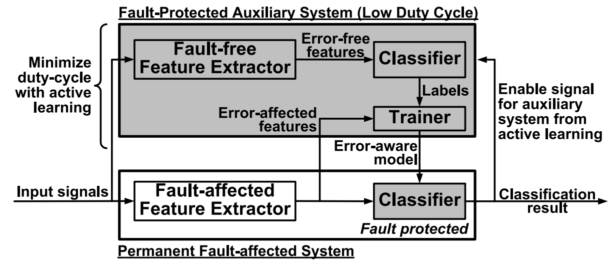


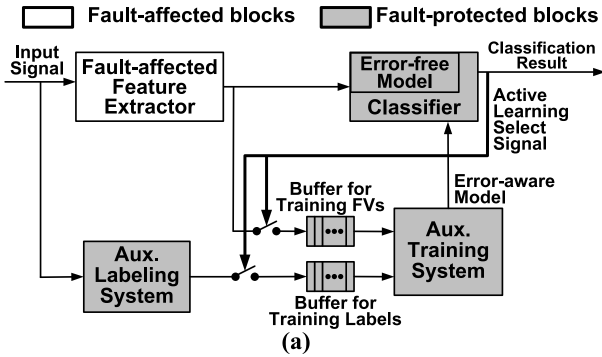
Fig. 6. System for self-construction of error-aware models by estimating training labels using an auxiliary system with low duty cycle, enabled by active learning (shaded blocks are fault protected).

error-free feature extraction and classification. For this classification, a model can be used that addresses the expected statistics generated due to the application signals alone; such a model is generic across instances of the system and can be derived *a priori* since it is not impacted by the errors caused by a particular instance of a fault-affected processor. Though the auxiliary system does not provide perfect labels, it provides labels that are accurate up to the classification performance that is achievable by an error-free system. As described in Section V-A, we find that estimated labels thus enable performance very close to that of perfect labels. As mentioned above, the implementation of error-free feature extractor on the MCU incurs a high energy cost; thus, this is only done during one-time or infrequent training to generate the estimated labels, allowing the energy to be amortized over the real-time operation of the fault-affected processor. Note, in this paper, where we consider permanent faults, training is performed only once. In general, various metrics can be considered for detecting a change in the data statistics to trigger retraining; for instance, changes in the rate with which feature vectors fall near the classification boundary can be monitored (i.e., by computing histograms of the marginal distance [39]). However, such approaches for triggering retraining are not covered in this paper.

As shown in Fig. 6, the auxiliary system performs classification on input data in parallel with processing through the fault-affected processor. The training labels and training data are thus provided by the auxiliary system and fault-affected processor, respectively, to an embedded trainer that can operate at low speed to construct an error-aware model. While in the demonstrated systems, the auxiliary system and the trainer are implemented by the fault-protected MCU, in voltage-scaled systems (such as [15]), these can be implemented via a low-error mode.

In addition to the need for infrequent training, active learning is used within the architecture to further minimize the duty cycle and hardware complexity of the auxiliary system. Active learning [40] is an approach where the optimal training data is selected from an available pool to reduce the training and labeling effort required. A common metric used to select training instances is the marginal distance from an initial decision boundary [39]. This metric is implicitly computed by the classification of kernels shown in Fig. 2 (the marginal distance, thus computed, is actually compared with a threshold to make classification decisions). In this paper, we also

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.



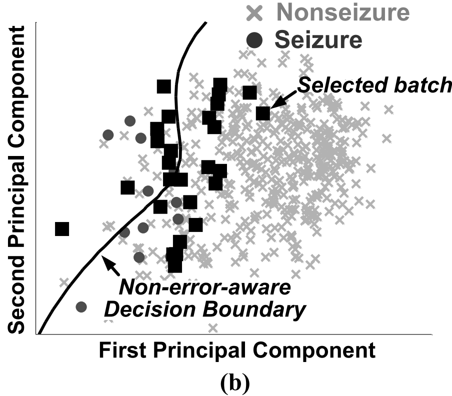


Fig. 7. (a) Active-learning system, where the permanent, fault-affected system chooses data to be used for training. (b) Error-affected feature vectors selected during one iteration using a marginal-distance criterion. Note that though other feature vectors appear closer to the decision boundary, this is in fact an artifact of PCA.

experiment with a randomized learner, wherein the training instances are selected at random. The active-learning architecture used in our system is shown in Fig. 7(a). An original model derived from error-free data serves as the initial decision boundary within the fault-protected classifier of the permanent system. The permanent system thus intrinsically derives the marginal distance metric used for active learning. Training is performed in iterations. As described below, this reduces the hardware requirements. During each iteration, the instances of feature data that fall within an established marginal distance are committed to a buffer along with their auxiliary-system-derived training labels; Fig. 7(b) shows the actual instances selected during an iteration from our hardware experiments. Following each iteration of training, a new error-aware model is loaded in the classifier of the permanent system. We show in Section V-B that active learning enables model convergence with very little training data. This minimizes the duty cycle of the auxiliary system as well as the amount of data that should be buffered for each iteration; i.e., with the seizure-detection and arrhythmia-detection systems, a buffer size of 2 kB enables model convergence in just 1 and 100 iterations, respectively.

D. Performance Limit of Error-Aware Models and a Computable Metric for Assessment

As we show in Section V-B, error-aware modeling is able to overcome very high levels of bit errors in the processed data.

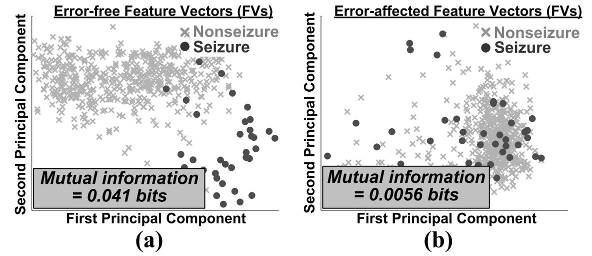


Fig. 8. Feature vector distributions for (a) baseline case without errors and (b) case with errors, where the MI in the resulting data is degraded.

In fact, with a flexible data-driven-modeling stage, neither the error rates nor magnitudes are critical. Rather, the bit-level changes that occur can be viewed as altering the way information is encoded within the feature data. The resulting encoding, though unpredictable, is learned through training with the error-affected data, thus leaving the performance limited fundamentally by how well the resulting encoding retains information for distinguishing class membership. A direct metric for the information for classification is MI. Given feature data X and class label Y, MI uses the Shannon entropy both to represent the overall uncertainty of the class label $H(Y)$ as well as the conditional uncertainty of the class $H(Y|X)$, dependent on the feature data. For discrete random variables, the Shannon entropy is defined as in (2) and (3). It should be noted that $p(x)$ in (3) implies a probability distribution over all values that the feature data can take over a multidimensional feature space. We describe in Section V-B how this is computed numerically for the data obtained from our hardware experiments

$$\mathbf{H(Y)} = -\sum_{y=-1,1} p(y)\log_2 p(y) \quad (2)$$

$$\mathbf{H(Y|X)} = -\sum_{x\in X} p(x) \sum_{y=-1,1} p(y|x)\log_2 p(y|x). \quad (3)$$

After computing entropies, MI is then given by $I(X;Y) = H(Y) - H(Y|X)$; i.e., it is the amount by which the uncertainty of the class labels is reduced given the feature data. To illustrate the implication of MI, Fig. 8 shows feature vector distributions taken from our measurements both in the absence of errors and in the presence of errors, for a case where the errors cannot be overcome. As shown, the errors result in an encoding of the feature data that degrades the separation between the class distributions. Measurements in Section V-B confirm that the performance of the error-aware model strongly corresponds with MI.

Though MI is a precise metric for information, the problem is that it cannot be computed on line, because it requires separating the data distributions according to class membership, Y (3). We suggest a proxy to MI (PMI), where the data is instead separated according to estimated class membership YE, i.e., $I(X;\text{YE}) = H(\text{YE}) - H(\text{YE}|X)$. As in the case of labels for model training, our system estimates class membership using the temporary error-free auxiliary system. Since $H(\text{YE}|X)$ then represents the conditional uncertainty dependent on class declarations by a fault-free system, $I(X;\text{YE})$ can

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

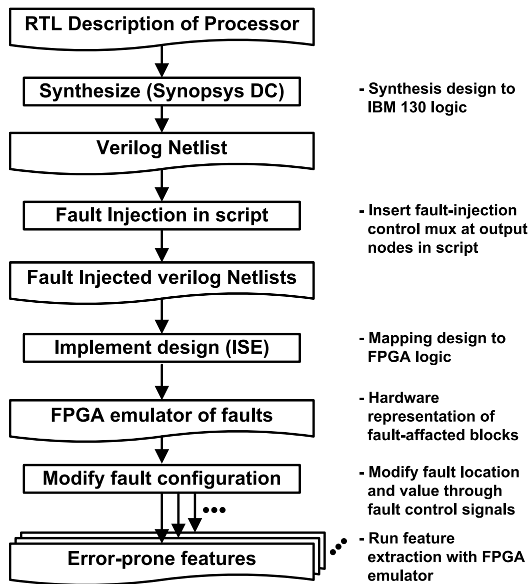


Fig. 9. FPGA based experimentation and demonstration flow to enable controllable scaling of fault rate as well as multiple gate-level implementations for randomized fault locations within the circuit.

be viewed as the information to discriminate feature data with performance corresponding to a fault-free system. We show in Section V-B that $I(X; \text{YE})$ is as an excellent proxy with strong correlation to $I(X; Y)$. It thus enables on-line assessment of an error-aware model's ability to overcome faults even before model convergence has been achieved.

IV. Experimental Approach

The experimental approach is driven by two major considerations: 1) emulating relevant faults in a way that accurately represents the physical effects that are of concern at the technological level, yet doing so in a way that enables evaluations on the algorithmic level targeted by our approach and 2) enabling controllable and suitably randomized fault injection to enable characterization of the approach. While the faults of interest have been the focus of substantial modeling efforts at the transistor/gate levels, system-level simulations, which are required to address the algorithmic approach, are not viable at the transistor/gate levels over large datasets [9], [35]. To overcome this, we focus on FPGA emulation. This allows us to represent hardware faults on a logical level, where the correspondence with technological sources is well established while enabling rapid system-level evaluation compared with simulation approaches. The fault model we employ is stuck-at-1/0 faults. While this model has been widely used to address a range of faults, previous work also explores and suggests its validity for the particular faults of interest in our experiments (opens/shorts due to nanoscale lithography, logic gate SNM violations in low-voltage regimes due to transistor variability) [29], [31], [32], [41].

Fig. 9 shows the FPGA-based fault-injection and hardware-experimentation flow. It starts with RTL of the system and involves synthesis into a gate-level netlist using an ASIC standard-cell library. Given the netlist, faults are injected

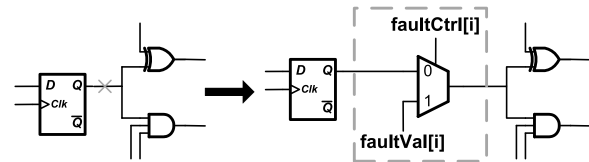


Fig. 10. Logical representation of stuck-at-0/1 faults achieved by introducing multiplexers on randomly-selected outputs; the multiplexers are controllable to set the fault rate, location, and type (i.e., stuck-at-0/1).

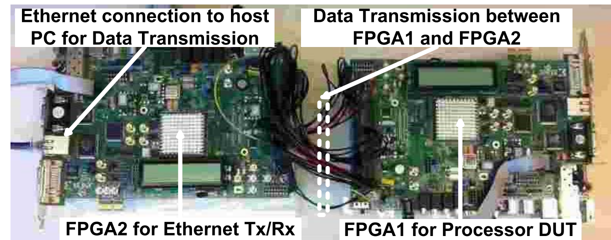


Fig. 11. FPGA test setup (with Xilinx ML509 Virtex 5 boards) for the DUT system and an Ethernet transceiver (enabling data exchange with PC).

by editing the netlist to introduce multiplexers on randomly-selected output nodes. As shown in Fig. 10, the multiplexers can be configured via the faultCtrl signal to assert a logic 0/1 at the output, depending on the faultVal signal; stuck-at-0/1 faults are thus represented. Following the introduction of multiplexers, a fault-control module is introduced to enable control of the faultCtrl and faultVal signals following FPGA mapping. Using this approach, faults can be injected at a controllable rate, and multiple instances of the circuit can be tested at each fault rate to consider the impact of various fault locations within the circuit. In our experiments, we perform tests on five instances of the fault-affected circuit for each fault rate. The faults are static for each test, set by configuring static values for the faultCtrl and faultVal signals. The final netlist, with circuits for injecting faults (multiplexers and fault-control module), is then implemented on an FPGA, designated as the device-under-test (DUT) FPGA. A second FPGA serves simply as an Ethernet transceiver to stream input data from a host PC and to capture output data to the host PC for post processing and analysis. Fig. 11 shows the setup, wherein two Xilinx Virtex 5 FPGAs are used.

V. Experimental Demonstrations and Results

This section presents the applications and systems used to demonstrate the DDHR concept. It then presents results from the hardware experiments. Finally, the results are analyzed using the metric of MI.

A. Applications for Demonstration

Two application systems for sensor-data classification are used to demonstrate the DDHR concept: EEG-based seizure detection and ECG-based cardiac-arrhythmia detection. The architectural and technical details are presented below.

1) EEG-Based Seizure Detector: The processor for the seizure-detection system is shown in Fig. 12(a). The system is based on a state-of-the-art detection algorithm

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

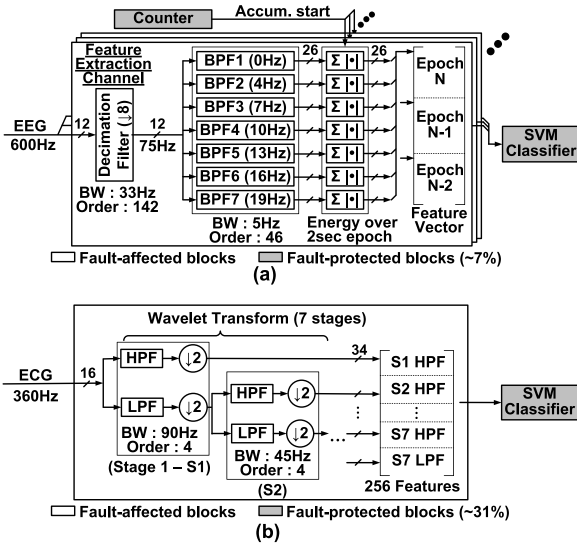


Fig. 12. Processors for (a) EEG-based seizure detection and (b) ECG-based cardiac arrhythmia detection used in our demonstrations (for the seizure detector, two of the shown channels are used). Both designs are implemented and synthesized from RTL, with the shaded blocks protected during fault injection (these account for $\sim$7% for the seizure detector and $\sim$31% for the cardiac arrhythmia detector).

presented in [42]. The processor uses EEG signals acquired from the scalp of an epileptic patient to detect the onset of clinical seizures. Up to 18 channels (i.e., EEG acquisition locations) can be used depending on the patient. For our demonstration, real EEG data, available from the CHB-MIT scalp EEG database [43], [44] are used. This includes expert annotations corresponding to the time of electrographic onset as well as the segment duration of a seizure. The clinical metrics used to evaluate performance are defined in [42]: 1) sensitivity measures the true positive rate of seizure detection; 2) false alarms measures the number of false positives; and 3) latency measures the delay in seizure detection with respect to the expert-identified onset.

For feature extraction, each EEG channel is downsampled by eight using a decimation filter. The spectral energy of each downsampled channel is then extracted from seven frequency bands by accumulating the output of a corresponding bandpass FIR filter over a 2-s epoch. This gives seven features per channel. The features from each epoch are concatenated with the features from two previous epochs, giving a total of 21 features per channel. In the system demonstration, two EEG channels are employed, thus giving a 42-D feature vector. The SVM then uses the feature vector to classify, corresponding to a seizure or nonseizure.

The RTL code is developed and synthesized for the system. The SVM and control block, which corresponds to a counter, are fault protected (i.e., explicitly omitted during netlist editing). These blocks account for approximately 7% of the total circuit. For the FPGA implementation, the fault-affected circuit consists of approximately 110.4k nodes, and faults are injected on 4 to 40 nodes, corresponding to a fault rate from 0.0036% to 0.036% of the fault-prone circuit.

2) ECG-Based Arrhythmia Detector: The processor for the arrhythmia-detection system is shown in Fig. 12(b).

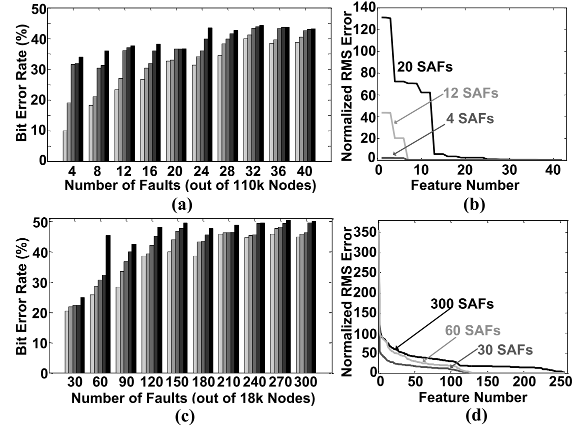


Fig. 13. Bit-error statistics of the computed feature vectors, showing the BERs for different fault levels and the RMS of the feature errors (normalized to the RMS of the true feature values) for (a) and (b) seizure-detection system and (c) and (d) arrhythmia-detection system.

Feature extraction of ECG signals corresponds to a discrete wavelet transform [45]. For our demonstration, real patient ECG data is obtained from the MIT-BIH arrhythmia database [44], [46]. This includes expert annotated labels for normal and arrhythmia beats. The metrics used for evaluating the performance are the true-positive (TP) rate and the false positive rate. For comparison across test cases, training is performed by fixing the true negative rate to 95%.

For feature extraction, a 256-point sequence is used from the ECG. The samples are processed through a seven-stage wavelet transform as shown in Fig. 12(b), resulting in 256 coefficients for each sequence. The feature vectors, thus extracted, are classified with a fault-protected SVM classifier. The fault-prone feature-extraction block accounts for 68% of the total nodes in the architecture. Faults are injected on up to 4k nodes, which correspond to $\sim$22% of the fault-prone circuit.

B. Hardware Results

This subsection presents the hardware results, starting with the bit-error statistics and then the overall performance of both the baseline and DDHR systems.

1) Bit-Error Statistics: The injected faults are observed to cause both high BERs and large error magnitudes, while also exhibiting highly irregular statistical distributions. The measured bit-error statistics for the feature vectors computed by the two demonstration systems are shown in Fig. 13. In Fig. 13(a) and (c), we observe that the feature vector BERs are severe in both application cases, span from 10% to 45% for the seizure detector and from 20% to 50% for the cardiac-arrhythmia detector. Fig. 13(b) and (d) selects three cases corresponding to different fault levels, where the DDHR successfully restores system performance. The plots show the RMS of the feature-error values normalized to the RMS of the true feature values, for each feature (i.e., as in the expression shown below). These plots illustrate that the performance is restored even in cases where the error magnitudes far exceed the actual feature values themselves (it is of the note that

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

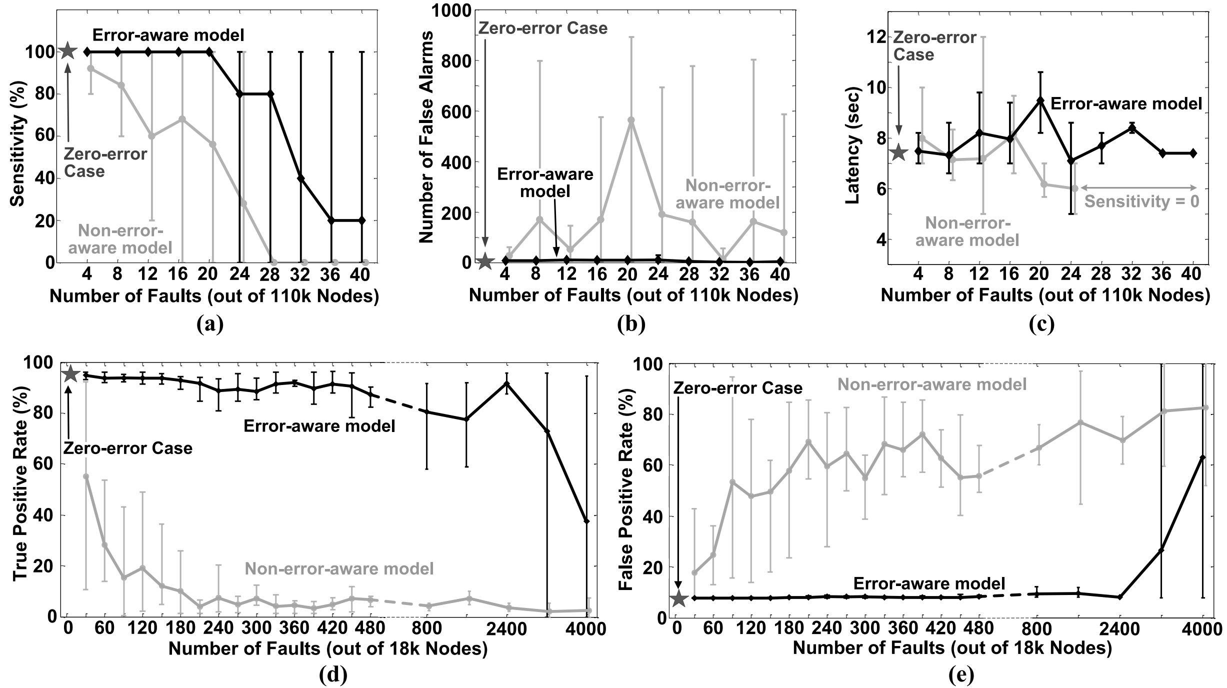


Fig. 14. Performance of the systems with respect to the fault rates, for the cases with and without DDHR (five instances of the system are tested at each fault rate). (a)–(c) Seizure detector performance. (d) and (e) Cardiac-arrhythmia detector performance (where the true-negative rate is set at 95% for all test cases). DDHR consistently restores system performance up to a fault level of 20 nodes for the seizure detector and 480 nodes for the cardiac-arrhythmia detector.

the features are deliberately sorted in order of descending normalized RMS error to aid visualization)

$$\frac{\text{RMS(error)}}{\text{RMS(true value)}} = \frac{\sqrt{\sum_{\mathbf{i=1}}^{\mathbf{M}} (\mathbf{F}_{\text{EA}} - \mathbf{F}_{\text{TV}})^2 / \mathbf{M}}}{\sqrt{\sum_{\mathbf{i=1}}^{\mathbf{M}} \mathbf{F}_{\text{TV}}^2 / \mathbf{M}}} \tag{4}$$

where

- F_{EA} value of error-affected feature;
- F_{TV} true value (error-free) of feature;
- M total number of feature vectors.

2) Overall Performance of DDHR Systems: Despite the large bit-level errors observed, we observe that the performances of the seizure and arrhythmia detectors are substantially restored, thanks to DDHR. To qualitatively show the impact, Fig. 15 first plots histograms of the output from the SVM classifier for one of the seizure-detector test cases (i.e., with faults on 12 nodes). The first histogram considers a baseline detector with no errors, showing the system's ability to strongly discriminate between the classes; the second histogram considers a detector with errors but without DDHR, showing loss of ability to discriminate the classes; and the third plot considers a detector with errors but with an error-aware model through DDHR, showing restored ability to discriminate between classes.

For quantitative overall system performance, Fig. 14(a)–(c) shows the results for the seizure detector, and Fig. 14(d) and (e) shows the results for the cardiac-arrhythmia detector. In both cases, the performance of a system without DDHR is compared to that with DDHR (i.e., using an error-aware model), as the number of injected faults is

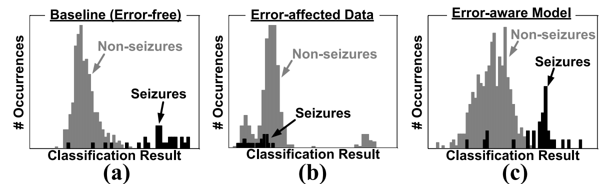


Fig. 15. Output histograms from the SVM classifier for seizure-detector test case. (a) Case of the baseline detector without errors. (b) Case of a detector with errors (due to twelve faults), but without DDHR. (c) Case with errors (due to twelve faults) but with DDHR. The errors initially degrade the separation between the class distributions, but DDHR restores the separation enabling classification.

increased (recall that five instances of the system are tested at each fault rate). While the systems without DDHR exhibit immediate performance degradation, also showing large variance across the cases at each rate, the systems with DDHR exhibit restored performance to the error-free levels for all instances tested up to a fault level of 20 nodes for the seizure detector and 480 nodes for the cardiac-arrhythmia detector. This suggests that substantial performance improvement can be achieved by DDHR in the presence of severe errors caused by hardware faults. Beyond the mentioned fault levels, many of the tested instances exhibit restored performance; however, degradations are also observed, making it valuable to have a mechanism that can predict when DDHR will be effective in cases of severe fault levels, as discussed below.

3) Error-Aware Model Construction: As mentioned in Section III-C, the active-learning architecture is intended to minimize the effort of constructing an error-aware model,

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

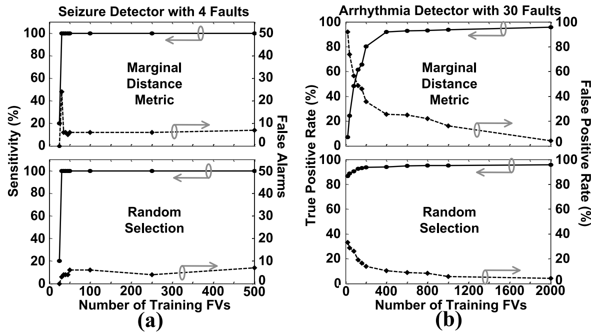


Fig. 16. Convergence of the error-aware model is achieved with minimal training data within an active-learning and random-learning architecture. The two cases show model convergence within (a) seizure-detection system at a fault level of four nodes and (b) arrhythmia-detection system at a fault level of thirty nodes (very similar profiles are observed at the other fault-rates tested).

namely, by minimizing the training-data buffer size and the duty-cycle required of the fault-protected auxiliary system for training-label estimation. Fig. 16(a) and (b) shows the rate of convergence of the model for representative test cases, by plotting the system performance with respect to the number of training vectors used, for the seizure-detection and arrhythmia-detection systems respectively. In both cases, the first plot shows active learning, using the marginal-distance metric for data selection, while the second plot shows the case where training data is selected randomly. In general, a variety of different metrics might be used for minimizing the data selected for on-line training, and the results show that model convergence can, in practice, be achieved with very few training vectors, (i.e., well below 2 k). As a result, the energy and hardware requirements (MCU performance, data buffer size, and so on) can be minimal.

4) MI and On-Line Assessment: As mentioned in Section III-D, the performance of DDHR is expected to be fundamentally limited by the amount of information that is preserved for class discrimination in the error-affected feature vectors. To numerically compute MI as in (2) and (3), probability distributions must be derived for the feature data. To do this, we discretize the entire multidimensional feature space into independent bins; this avoids complications associated with statistical dependencies between the features. Accordingly, for a dimensionality of d and a discretization of each feature into n values, the total number of bins is n^d. The challenge with this is that for reasonable discretization, a large number of bins result due to the high dimensionality (i.e., seizure detection involves dimensionality of 42 and arrhythmia detection involves dimensionality of 256). As a result, we estimate MI by reducing the dimensionality through PCA to $d = 8$ for the seizure detector and $d = 4$ for the cardiac-arrhythmia detector, and we discretize with $n = 2$ (for a total of 256 bins) and $n = 5$ (for a total of 625 bins), respectively; numerically these are shown to yield good estimates of the MI, as observed by the tests wherein the n and d value are scaled.

Due to the need to specify the class label for each feature vector, MI is not computable on-line. As a result, Section III-D proposed a proxy to MI (PMI), which is based on the estimated class labels derived by a fault-free auxiliary system. Fig. 17 shows scatter plots of the TP rate with DDHR versus both MI and PMI for the seizure detector [Fig. 17(a) and (b)] and the cardiac-arrhythmia detector [Fig. 17(c) and (d)]. For the seizure detector, the TP rate is used in this analysis rather than the usual metrics [i.e., from Fig. 14(a)–(c)] since it has direct correspondence with the classifier output, whereas the usual metrics, which have greater application relevance, involve additional derivation [42]. The first point we see is that MI is a strong indicator for the overall performance achieved with DDHR; an MI beyond 0.02 and 0.5 bits, respectively, consistently yields restored performance. The second point we see is that PMI shows excellent correspondence with MI. Thus, by computing PMI and setting thresholds appropriately, we see that architectures can be enabled that detect when their own faults are too severe to overcome using DDHR.

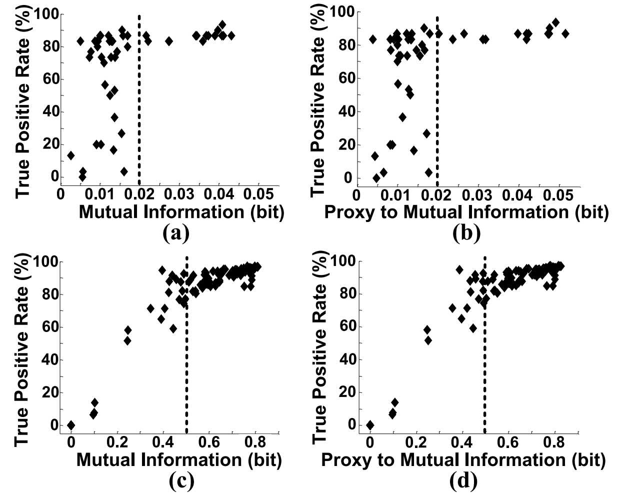


Fig. 17. Scatter plots of TP rate versus MI and PMI, achieved through DDHR within the (a) and (b) seizure-detection system and (c) and (d) arrhythmia-detection system (note that parameters are intentionally set for a true negative rate of 98% for the seizure detector and 95% for cardiac-arrhythmia detector to facilitate comparison across all cases). A threshold of 0.02 bits and 0.5 bits can be used, respectively, to indicate when DDHR will successfully restore system performance.

VI. Conclusion

This paper presents the concept of data-driven hardware resilience (DDHR) for overcoming computational errors due to faults in a digital processor. DDHR uses embedded ML stages to model the statistics of processed data in the presence of errors due to faults in the processing hardware. A system architecture for sensor-data classification is presented that utilizes DDHR and enables on-line training of an error-aware model entirely within the system with minimal overhead. To minimize the model-training overhead, the architecture employs the concept of active learning to construct a reduced data set for training using the error-affected data derived from the fault-affected processor itself. To enable model-training entirely within the system, the architecture avoids the need for external labeling of the training data using a temporary, nonreal-time auxiliary system for error-free classification as a means of generating estimated labels. Although such an

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

auxiliary system is energy intensive, it is only required for model training, which is performed infrequently, allowing the energy overhead to be greatly amortized over the real-time operation. The experimental data is presented (based on FPGA emulation), showing that the system is able to restore performance even when severe bit-level errors occur. Analysis based on MI shows that system performance is set more fundamentally by the level of information retained in the error-affected data. A key insight that thus emerges is that information is often retained even in the presence of high levels of bit errors. While DDHR takes an indirect approach to exposing and modeling system errors (i.e., by focusing on the data statistics generated due to the combined effects of errors and application signals), this has the advantage of enabling greater error tolerance through a dependence on the precise way, in which errors affect the specific distributions involved.

REFERENCES

[1] F. M. Khan, M. G. Arnold, and W. M. Pottenger, "Hardware-based support vector machine classification in logarithmic number systems," in *Proc. IEEE Int. Symp. Circuits Syst.*, vol. 5. May 2005, pp. 5154–5157.

[2] K. H. Lee and N. Verma, "A 1.2–0.55 V general-purpose biomedical processor with configurable machine-learning accelerators for high-order, patient-adaptive monitoring," in *Proc. ESSCIRC*, Sep. 2012, pp. 285–288.

[3] J. Park, J. Kwon, J. Oh, S. Lee, J.-Y. Kim, and H.-J. Yoo, "A 92-mW real-time traffic sign recognition system with robust illumination adaptation and support vector machine," *IEEE J. Solid-State Circuits*, vol. 47, no. 11, pp. 2711–2723, Nov. 2012.

[4] P. Dubey, "Recognition, mining and synthesis moves computers to the era of tera," *Technol. Intel Mag.*, vol. 9, no. 2, pp. 1–10, Feb. 2005.

[5] M. Horowitz, E. Alon, D. Patil, S. Naffziger, R. Kumar, and K. Bernstein, "Scaling, power, and the future of CMOS," in *IEEE Int. Electron Devices Meeting (IEDM) Tech. Dig.*, Dec. 2005, pp. 7–15.

[6] T. Austin, V. Bertacco, D. Blaauw, and T. Mudge, "Opportunities and challenges for better than worst-case design," in *Proc. Asia South Pacific Design Autom. Conf.*, 2005, pp. 2–7.

[7] E. Karl *et al.*, "A 4.6 GHz 162 Mb SRAM design in 22 nm tri-gate CMOS technology with integrated active VMIN-enhancing assist circuitry," in *IEEE Int. Solid-State Circuits Conf. Dig. Tech. Papers (ISSCC)*, Feb. 2012, pp. 230–232.

[8] N. R. Shanbhag *et al.*, "The search for alternative computational paradigms," *IEEE Des. Test. Comput.*, vol. 25, no. 4, pp. 334–343, Jul./Aug. 2008.

[9] N. Verma, K. H. Lee, K. J. Jang, and A. Shoeb, "Enabling system-level platform resilience through embedded data-driven inference capabilities in electronic devices," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Mar. 2012, pp. 5285–5288.

[10] M. Bohr, "The new era of scaling in an SoC world," in *IEEE Int. Solid-State Circuits Conf.-Dig. Tech. Papers (ISSCC)*, Feb. 2009, pp. 23–28.

[11] J. W. McPherson, "Reliability challenges for 45 nm and beyond," in *Proc. 43rd Annu. Design Autom. Conf.*, 2006, pp. 176–181.

[12] N. R. Shanbhag, R. A. Abdallah, R. Kumar, and D. L. Jones, "Stochastic computation," in *Proc. 47th Design Autom. Conf.*, Jun. 2010, pp. 859–864.

[13] C. W. Slayman, "Cache and memory error detection, correction, and reduction techniques for terrestrial servers and workstations," *IEEE Trans. Device Mater. Rel.*, vol. 5, no. 3, pp. 397–404, Sep. 2005.

[14] D. Ernst *et al.*, "Razor: A low-power pipeline based on circuit-level timing speculation," in *Proc. 36th Annu. IEEE/ACM Int. Symp. Microarchit.*, Dec. 2003, pp. 7–18.

[15] 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.

[16] A. L. Hopkins, Jr., T. B. Smith, III, and J. H. Lala, "FTMP—A highly reliable fault-tolerant multiprocess for aircraft," vol. 66, no. 10, pp. 1221–1239, Oct. 1978.

[17] N. Shanbhag, "Reliable and energy-efficient digital signal processing," in *Proc. 39th Annu. Design Autom. Conf.*, 2002, pp. 830–835.

[18] E. P. Kim and N. R. Shanbhag, "Soft N-modular redundancy," *IEEE Trans. Comput.*, vol. 61, no. 3, pp. 323–336, Mar. 2012.

[19] L. Leem, H. Cho, J. Bau, Q. A. Jacobson, and S. Mitra, "ERSA: Error resilient system architecture for probabilistic applications," in *Proc. Design, Autom. & Test Eur. Conf. & Exhibit. (DATE)*, Mar. 2010, pp. 1560–1565.

[20] J. Sartori and R. Kumar, "Architecting processors to allow voltage/reliability tradeoffs," in *Proc. 14th Int. Conf. Compil., Archit. Synthesis Embedded Syst.*, 2011, pp. 115–124.

[21] Y. Yetim, M. Martonosi, and S. Malik, "Extracting useful computation from error-prone processors for streaming applications," in *Proc. Design, Autom. & Test Eur. Conf. & Exhibit. (DATE)*, Mar. 2013, pp. 202–207.

[22] 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. 47th Design Autom. Conf.*, Jun. 2010, pp. 555–560.

[23] N. Cristianini and J. Shawe-Taylor, *An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods*. Cambridge, U.K.: Cambridge Univ. Press, 2000.

[24] R. Akbani, S. Kwek, and N. Japkowicz, "Applying support vector machines to imbalanced datasets," in *Proc. 15th Eur. Conf. Mach. Learn.*, Sep. 2004, pp. 39–50.

[25] T. Joachims, *Text Categorization With Support Vector Machines: Learning With Many Relevant Features*. New York, NY, USA: Springer-Verlag, 1998.

[26] O. Girard. (2010). *Openmsp430 Project*. [Online]. Available: http://opencore.org

[27] M. Karthikeyan *et al.*, "A 65-nm random and systematic yield ramp infrastructure utilizing a specialized addressable array with integrated analysis software," *IEEE Trans. Semicond. Manuf.*, vol. 21, no. 2, pp. 161–168, May 2008.

[28] P. Vijayakumar, V. B. Suresh, and S. Kundu, "Lithography aware critical area estimation and yield analysis," in *Proc. IEEE Int. Test Conf. (ITC)*, Sep. 2011, pp. 1–8.

[29] J. Kwong and A. P. Chandrakasan, "Variation-driven device sizing for minimum energy sub-threshold circuits," in *Proc. Int. Symp. Low Power Electron. Design*, 2006, pp. 8–13.

[30] K. H. Lee and N. Verma, "A low-power processor with configurable embedded machine-learning accelerators for high-order and adaptive analysis of medical-sensor signals," *IEEE J. Solid-State Circuits*, vol. 48, no. 7, pp. 1625–1637, Jul. 2013.

[31] X. Shi, S. Hsu, J. F. Chen, C. M. Hsu, R. J. Socha, and M. V. Dusa, "Understanding the forbidden pitch phenomenon and assist feature placement," *Proc. SPIE*, vol. 4689, pp. 985–996, Jul. 2002.

[32] N. Verma, J. Kwong, and A. P. Chandrakasan, "Nanometer MOSFET variation in minimum energy subthreshold circuits," *IEEE Trans. Electron Devices*, vol. 55, no. 1, pp. 163–174, Jan. 2008.

[33] J. Zhang *et al.*, "Robust digital VLSI using carbon nanotubes," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 31, no. 4, pp. 453–471, Apr. 2012.

[34] A. Sreedhar, A. Sanyal, and S. Kundu, "On modeling and testing of lithography related open faults in nano-CMOS circuits," in *Proc. Design, Autom. & Test Eur. (DATE)*, Mar. 2008, pp. 616–621.

[35] R. A. Abdallah, Y.-H. Lee, and N. R. Shanbhag, "Timing error statistics for energy-efficient robust DSP systems," in *Proc. Design, Autom. & Test Eur. Conf. & Exhibit. (DATE)*, Mar. 2011, pp. 1–4.

[36] N. Verma, A. Shoeb, J. Bohorquez, J. Dawson, J. Guttag, and A. P. Chandrakasan, "A micro-power EEG acquisition SoC with integrated feature extraction processor for a chronic seizure detection system," *IEEE J. Solid-State Circuits*, vol. 45, no. 4, pp. 804–816, Apr. 2010.

[37] M. A. B. Altaf, J. Tillak, Y. Kifle, and J. Yoo, "A 1.83 μJ/classification nonlinear support-vector-machine-based patient-specific seizure classification SoC," in *IEEE Int. Solid-State Circuits Conf. Dig. Tech. Papers (ISSCC)*, Feb. 2013, pp. 100–101.

[38] A. Shoeb, D. Carlson, E. Panken, and T. Denison, "A micropower support vector machine based seizure detection architecture for embedded medical devices," in *Proc. Annu. Int. Conf. IEEE Eng. Med. Biol. Soc. (EMBC)*, 2009, pp. 4202–4205.

[39] G. Schohn and D. Cohn, "Less is more: Active learning with support vector machines," in *Proc. ICML*, 2000, pp. 839–846.

[40] D. Angluin, "Queries and concept learning," *Mach. Learn.*, vol. 2, no. 4, pp. 319–342, Apr. 1988.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

[41] S. Ghosh and F. J. Ferguson, “Estimating detection probability of interconnect opens using stuck-at tests,” in *Proc. 14th ACM Great Lakes Symp. VLSI*, 2004, pp. 254–259.

[42] A. H. Shoeb and J. V. Guttag, “Application of machine learning to epileptic seizure detection,” in *Proc. 27th Int. Conf. Mach. Learn. (ICML)*, 2010, pp. 975–982.

[43] A. H. Shoeb, “Application of machine learning to epileptic seizure onset detection and treatment,” Ph.D. dissertation, Dept. Health Sci. Technol., Massachusetts Inst. Technol., Cambridge, MA, USA, 2009.

[44] A. L. Goldberger *et al.*, “Physiobank, physiotoolkit, and physionet components of a new research resource for complex physiologic signals,” *Circulation*, vol. 101, no. 23, pp. e215–e220, 2000.

[45] E. D. Übeyli, “ECG beats classification using multiclass support vector machines with error correcting output codes,” *Digit. Signal Process.*, vol. 17, no. 3, pp. 675–684, May 2007.

[46] G. B. Moody and R. G. Mark, “The impact of the MIT-BIH arrhythmia database,” *IEEE Eng. Med. Biol. Mag.*, vol. 20, no. 3, pp. 45–50, May/Jun. 2001.

Zhuo Wang (S’13) received the B.S. degree in microelectronics from Peking University, Beijing, China, in 2011, and the M.S. degree from the Department of Electrical Engineering, Princeton University, Princeton, NJ, USA, in 2013, where he is currently pursuing the Ph.D. degree.

His current research interests include robust system design, in particular, 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 faults affecting the platform.

Kyong Ho Lee (S’10) received the B.S. degree from the Korea Advanced Institute of Science and Technology, Daejeon, Korea, in 2004, the M.S. degree from Stanford University, Stanford, CA, USA, in 2009, and the Ph.D. degree from Princeton University, Princeton, NJ, USA, in 2013, all in electrical engineering.

He is currently with Samsung Research America, San Jose, CA, USA, as a Hardware Engineer. His current research interests include ultralow-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.

Dr. Lee was a co-recipient of Qualcomm Innovation Fellowship in 2011.

Naveen Verma (S’04–M’09) 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.

He has been an Assistant Professor of Electrical Engineering with Princeton University, Princeton, NJ, USA, since 2009. On the circuit level, his focus spans low-voltage digital logic and SRAMs, low-noise analog instrumentation and data-conversion, and integrated power management. His current research interests include ultralow-power integrated circuits and systems with an emphasis on sensing applications, with particular importance of 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.

Prof. Verma was a co-recipient of the 2008 International Solid-State Circuits Conference (ISSCC) Jack Kilby Award for Outstanding Student Paper, the 2006 DAC/ISSCC Student Design Contest Award, and the Alfred Rheinstein Junior Faculty Award at Princeton, and the 2013 National Science Foundation CAREER Award.