DESIGN OF ROBUST ULTRA-LOW-POWER PLATFORM FOR IN-SILICON MACHINE LEARNING

BY

SAI ZHANG

DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Electrical and Computer Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2016

Urbana, Illinois

Doctoral Committee:

Professor Naresh R. Shanbhag, Chair
Professor Minh N. Do
Professor David T. Blaauw, University of Michigan
Assistant Professor Lav R. Varshney

ABSTRACT

The rapid development of machine learning plays a key role in enabling next generation computing systems with enhanced intelligence. Present day machine learning systems adopt an "intelligence in the cloud" paradigm, resulting in heavy energy cost despite state-of-the-art performance. It is therefore of great interest to design embedded ultra-low power (ULP) platforms with in-silicon machine learning capability. A self-contained ULP platform consists of the energy delivery, sensing and information processing subsystems. This dissertation proposes techniques to design and optimize the ULP platform for in-silicon machine learning by exploring a trade-off that exists between energy-efficiency and robustness. This trade-off arises when the information processing functionality is integrated into the energy delivery, sensing, or emerging stochastic fabrics (e.g., CMOS operating in near-threshold voltage or voltage overscaling, and beyond CMOS devices).

This dissertation presents the Compute VRM (C-VRM) to embed the information processing into the energy delivery subsystem. The C-VRM employs multiple voltage domain stacking and core swapping to achieve high total system energy efficiency in near/sub-threshold region. A prototype IC of the C-VRM is implemented in a 1.2 V, 130 nm CMOS process. Measured results indicate that the C-VRM has up to 44.8% savings in system-level energy per operation compared to the conventional system, and an efficiency ranging from 79% to 83% over an output voltage range of 0.52 V to 0.6 V.

This dissertation further proposes the Compute Sensor approach to embed information processing into the sensing subsystem. The Compute Sensor eliminates both the traditional sensor-processor interface, and the high-SNR/high-energy digital processing by moving feature extraction and classification functions into the analog domain. Simulation results in 65 nm CMOS show that the proposed Compute Sensor can achieve a detection accuracy

greater than 94.7% using the Caltech101 dataset, which is within 0.5% of that achieved by an ideal digital implementation. The performance is achieved with 7× to 17× lower energy than the conventional architecture for the same level of accuracy.

To further explore the energy-efficiency vs. robustness trade-off, this dissertation explores the use of highly energy efficient but unreliable stochastic fabrics to implement in-silicon machine learning kernels. In order to perform reliable computation on the stochastic fabrics, this dissertation proposes to employ statistical error compensation (SEC) as an effective error compensation technique. This dissertation makes a contribution to the portfolio of SEC by proposing embedded algorithmic noise tolerance (E-ANT) for low overhead error compensation. E-ANT operates by reusing part of the main block as estimator and thus embedding the estimator into the main block. System level simulation results in a commercial 45 nm CMOS process show that E-ANT achieves up to 38% error tolerance and up to 51% energy savings compared with an uncompensated system.

This dissertation makes a contribution to the theoretical understanding of stochastic fabrics by proposing a class of probabilistic error models that can accurately model the hardware errors on the stochastic fabrics. The models are validated in a commercial 45 nm CMOS process and employed to evaluate the performance of machine learning kernels in the presence of hardware errors. Performance prediction of a support vector machine (SVM) based classifier using these models indicates that the probability of detection P_{det} estimated using the proposed model is within 3% for timing errors due to voltage overscaling when the error rate $p_\eta \leq 80\%$, within 5% for timing errors due to process variation in near threshold-voltage (NTV) region ($0.3\,\mathrm{V} - 0.7\,\mathrm{V}$) and within 2% for defect errors when the defect rate p_{saf} is between 10^{-3} and 20%, compared with HDL simulation results.

Employing the proposed error model and evaluation methodology, this dissertation explores the use of distributed machine learning architectures, named classifier ensemble, to enhance the robustness of in-silicon machine learning kernels. Comparative study of distributed architectures (i.e., random forest (RF)) and centralized architectures (i.e., SVM) is performed in a commercial 45 nm CMOS process. Employing the UCI machine learning repository as input, it is determined that RF-based architectures are significantly more robust than SVM architectures in presence of timing errors in the NTV region ($0.3\,\mathrm{V} - 0.7\,\mathrm{V}$).

Additionally, an error weighted voting technique that incorporates the timing error statistics of the NTV circuit fabric is proposed to further enhance the robustness of RF architectures. Simulation results confirm that the error weighted voting technique achieves a P_{det} that varies by only 1.4%, which is 12× lower compared to centralized architectures.

To my parents, and my advisor

ACKNOWLEDGMENTS

First and foremost, my deepest thanks go to my parents, and my advisor. My parents have always put me before themselves, and supported me with the greatest love and care that anyone can hope for. My advisor, Prof. Naresh Shanbhag, has not only been the teacher and supervisor for my research, but also the mentor for my life. It is he who showed me the path towards being an independent researcher with his passion, deep knowledge, patient guidance and high standards. He demonstrates to me the importance of strong motivation, systematic research, and logical reasoning. I believe these valuable lessons, along with many others that I have learned from him, will continuously benefit me throughout the rest of my career. I am grateful to have my parents and my advisor, to whom I dedicate my thesis. I would also like to sincerely thank Prof. David Blaauw, Prof. Lav Varshney, and Prof. Minh Do for their many insightful comments and input and for agreeing to be on my committee. Their suggestions have greatly helped improve this dissertation. Additionally, I would like to thank Jane Tu for her help with implementing the FIR filter core in the Compute VRM project in Chapter 2, and Mingu Kang and Charbel Sakr for providing simulation models for the Compute Sensor project in Chapter 3. I would also like to give my sincere acknowledgments to my research group colleagues, Yingyan Lin, Charbel Sakr, Sujan Gonugondla, Mingu Kang, Ameya Patil, and Dr. Yongjune Kim; my research group alumni, Dr. Rami Abdallah and Dr. Eric Kim; and my colleagues at UIUC, notably Dr. Talegaonkar, Dr. Kairouz, and Guanghua Shu for their valuable feedback, input, and generous help with improving this dissertation. I gratefully acknowledge past and present support from Texas Instruments, and Systems on Nanoscale Information fabriCs (SONIC), one of the six SRC STARnet Centers, sponsored by MARCO and DARPA. Finally, I would like to thank my friends at Illinois, especially the Chinese Volleyball Team and the Formosa Volleyball Enthusiasts group who made me feel at home. Because of them, I was able to keep my physical well-being, and stay focused on my research.

CONTENTS

LIST OF TABLES

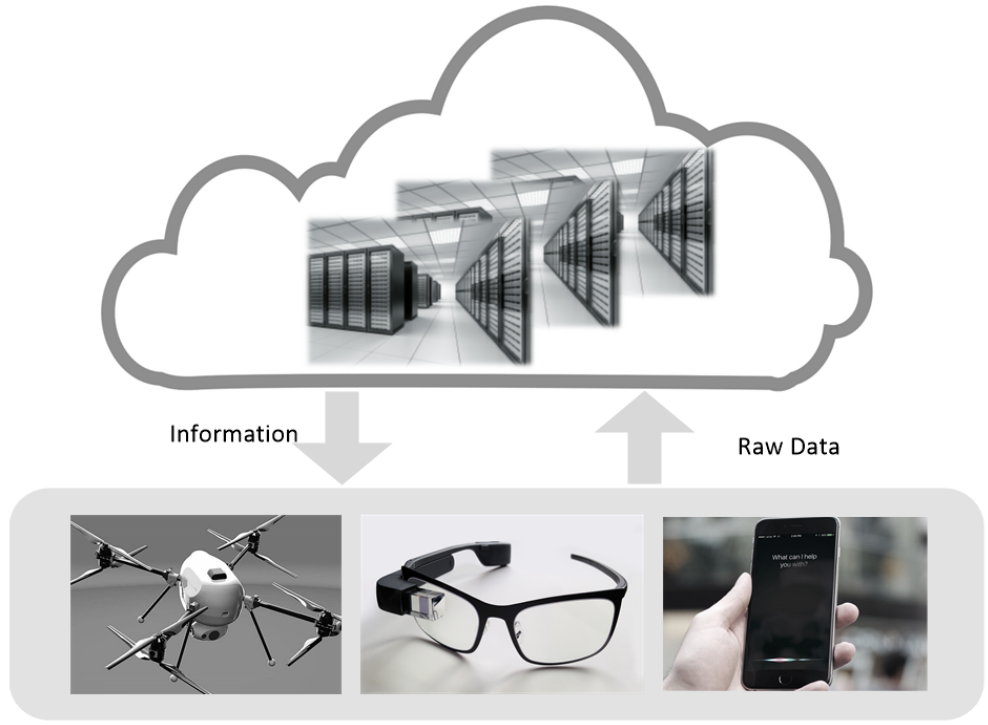
LIST OF FIGURES

Chapter 1

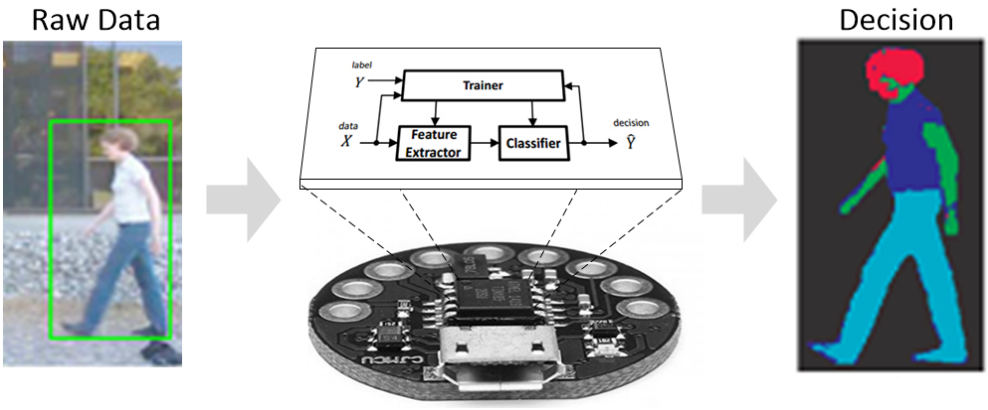
INTRODUCTION

Machine learning based systems are transforming the way humans interact with the physical world and have found wide applications in computer vision, data mining, healthcare, and more. In many areas such as object recognition [1], machines have begun to exceed human performance due to machine learning algorithms' capability to learn complex correlations from large volumes of data. However, this state-of-the-art performance comes at the price of heavy energy cost. For example, Google's AlphaGo system [2] that beat the human Go champion employs 1202 CPUs and 176 GPUs, and consumes more than four-orders-of-magnitude higher power compared with the much cited $\sim$20W power consumption of the human brain. As a result of the intensive energy cost, most of the current machine learning systems adopt an "intelligence in the cloud" paradigm as shown in Fig. 1.1(a). In this paradigm, a large volume of sensory data is transferred from mobile devices to data centers, where the bulk of machine learning algorithms are implemented on CPU and GPU-based clusters. The extracted inference models are then transferred from the cloud back to the device. This voluminous data transmission to the cloud leads to significant energy and latency costs. Indeed, recent projections [3] indicate that the traffic to the cloud consumes 9$\times$ more energy than that in the data center itself, and can account for 2% of the global electricity consumption [4]. Therefore, there is a clear need for small form factor ultra-low-power (ULP) platforms with inference capability so that the generated data can be processed to obtain decisions locally (see Fig. 1.1(b)). Achieving this goal requires energy efficient in-silicon implementation of machine learning systems.

A self-contained ULP platform for in-silicon machine learning consists of sensing, information processing, and energy delivery subsystems (see Fig. 1.2). Figure 1.2 shows a conventional architecture for embedded vision applications. Image data is first acquired via



(a)



(b)

Figure 1.1: The need for in-silicon machine learning: (a) current machine learning systems are implemented in the cloud, requiring transmission of voluminous amount of raw data, and (b) ultra-low-power (ULP) platforms with in-silicon machine learning can potentially process raw data to generate decisions locally.

an active pixel sensor (APS) array whose analog pixel values are sensed sequentially, and converted into digital samples via analog-to-digital converters (ADCs), and then streamed out to a back-end digital processor which implements feature extraction and classification function to obtain the final decision. A digital trainer block computes the hyperparameters of the feature extractor and classifier. The energy delivery subsystem converts the voltage from the supply (battery or energy harvester) to the voltage level suitable for information processing. The limited energy source and the computational complexity of learning algorithms make energy efficiency one of the primary design objectives. Several challenges arise in designing and optimizing such a system:

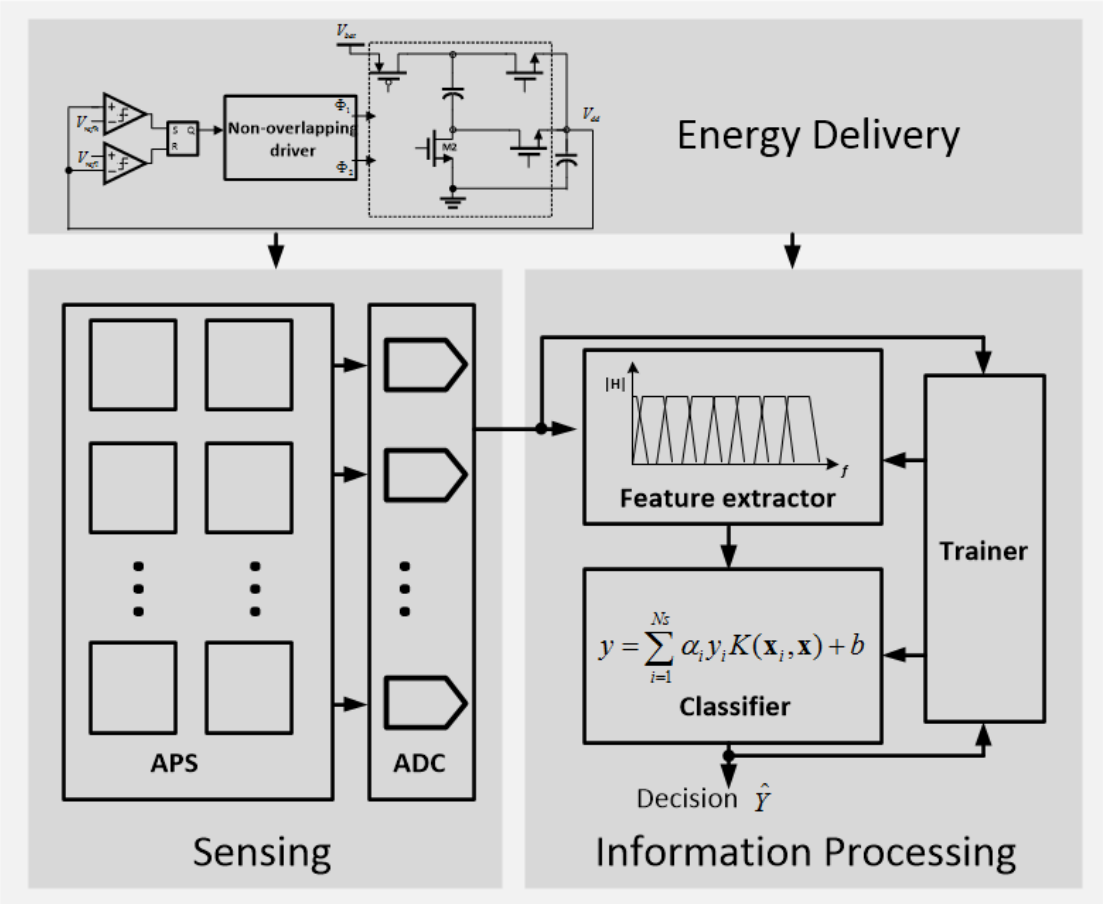


Figure 1.2: Architecture of the ULP platform for in-silicon machine learning.

- **The energy delivery challenge**: The energy delivery subsystem typically consists of one or more voltage regulator modules (VRMs) to convert the voltage from energy

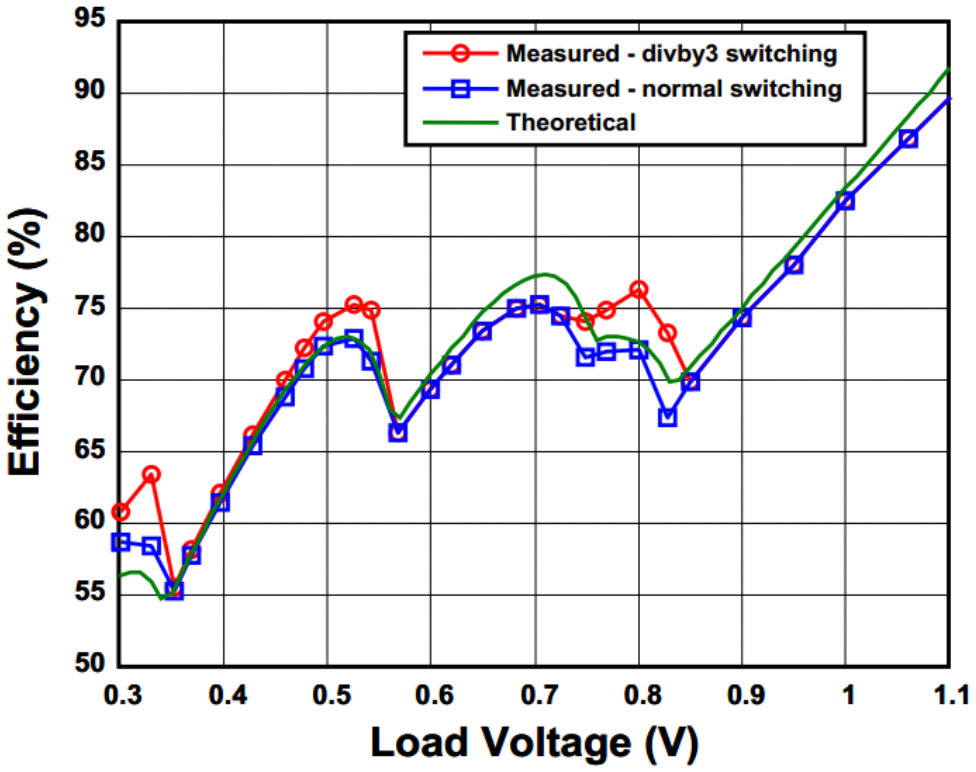
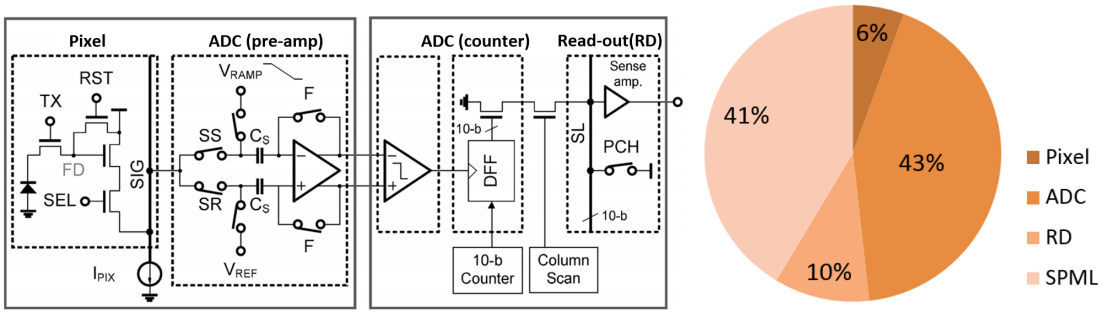


Figure 1.3: The voltage conversion efficiency of VRMs tends to decrease as the conversion ratio increases.

source into voltage of operation. While designing VRM with high efficiency of $\eta > 90\%$ is feasible for output voltage $V_{dd} \geq 1\,\mathrm{V}$, it is increasingly difficult to maintain such high efficiency for ULP platforms that need to operate with scaled voltages for reduction of computation energy. As shown in Fig. 1.3, the efficiency of VRM tends to decrease as the output load voltage decreases [5]. This poor VRM efficiency will offset the energy savings provided via low-voltage design techniques such as sub/near-threshold voltage design.

- **The communication challenge:** The physical separation between the sensing and information processing subsystems leads to a large interface energy. Such a separation is made unavoidable because sensing is intrinsically an analog process while information processing is intrinsically digital in the conventional architecture employing digital signal processors. The energy required to move the data over the sensor-processor

interface (see Fig. 1.4(a)) comprising the ADC, the read-out (RD) circuitry and the interconnect to the digital processor, can account for more than 50% of the total energy as shown in Fig. 1.4(b).

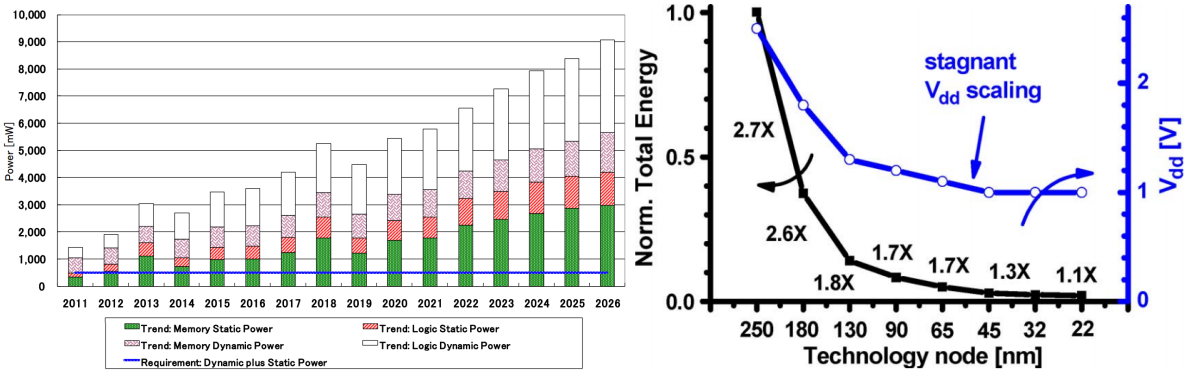


(a) (b)

Figure 1.4: The communication challenge [6] : (a) the sensing front-end, and (b) energy breakdown in a 65nm CMOS of an embedded vision system consisting of active pixel sensor (APS) array as the sensing front-end and principal component analysis (PCA) and support vector machine (SVM) digital signal processor as the back-end.

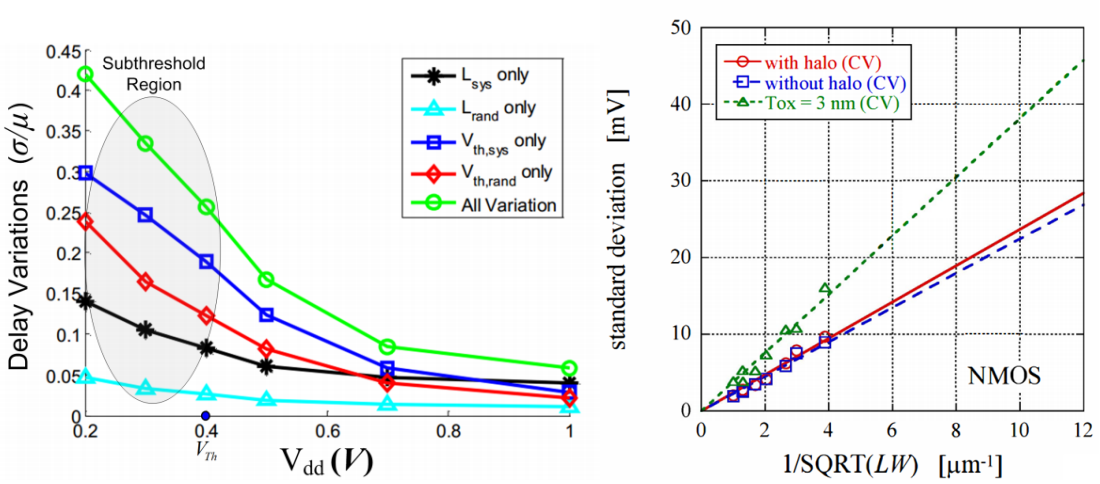
- **The computation challenge**: As projected in Fig. 1.5(a), the power consumption of portable electronics is expected to keep increasing. This increase will be accelerated if machine learning and inference capabilities were to be integrated in-silicon. Indeed, many machine learning algorithms are computationally intensive, e.g., more than 666 million MACs are required to process one 227×227 image (13k MACs/pixel) in AlexNet [7], one of the state-of-art deep learning algorithms. To reduce energy, the conventional approach is to rely on continuous scaling of supply voltage and feature size. However, this trend of scaling has stagnated as shown in Fig. 1.5(b) [8].

- **The robustness challenge:** One way to further reduce energy consumption is to employ near/subtheshold design where the voltage is aggressively scaled down to 200 mV $\sim$ 500 mV. However, the resultant energy savings come with an increase in delay variation as shown in Fig. 1.6(a) [10]. In addition, process scaling leads to increased process-voltage-temperature (PVT) variation, leakage, soft-errors and noise (see Fig.1.6(b) [11]). This is becoming a growing concern for reliable computing.



(a) (b)

Figure 1.5: The computation challenge: (a) the total power for embedded devices keeps increasing [9], and (b) supply voltage scaling for CMOS process is stagnant below 45 nm [8].



(a) (b)

Figure 1.6: The robustness challenge: (a) delay variation increases as supply voltage scales from super to sub-threshold region [10], and (b) standard deviation of threshold voltage increases as technology scales [11].

An estimated energy breakdown between the energy delivery, sensing, and information processing subsystems is very helpful to identify the limitations in the conventional architectures. Employing published works in the literature, it is safe to assume that 5%-20% of the energy is consumed in the energy delivery subsystem as power conversion loss under low voltage operations. The energy breakdown between sensing and information processing subsystems highly depends on the application, algorithm, and circuit architecture employed.

As shown in Fig. 1.4(b), the sensing subsystem accounts for 59% of the system energy (excluding the energy delivery loss), and most of this energy is consumed in the interface circuitry. At the same time, information processing accounts for 41% of the total system energy when employing PCA and SVM kernels as the digital backend. This suggests that the interface circuitry and the information processing subsystem are the dominant sources of energy consumption in the conventional architecture.

In the following part of this chapter, we provide an overview of related work to address these challenges, and finally present our approach to solve these problems.

1.1 Related Work

This section provides an overview about related work in the design of energy delivery and sensing subsystems, low power design, and robust system design.

1.1.1 Energy Delivery

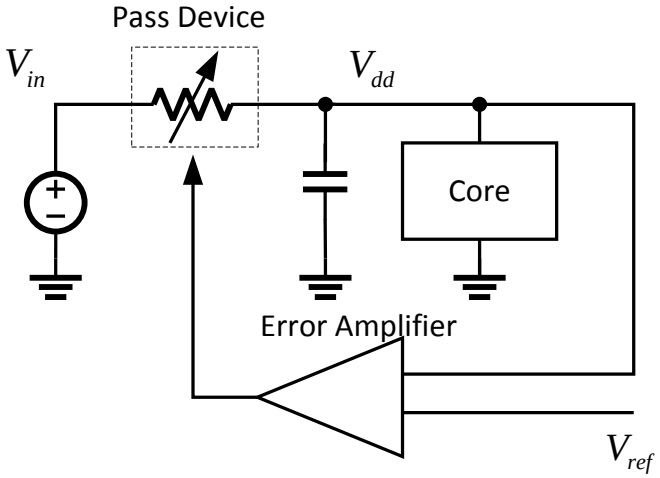
There are three commonly used VRM topologies: 1) linear regulator, 2) switching converter, and 3) switched capacitor voltage regulator module (SC-VRM), as shown in Fig. 1.7. The linear regulator (Fig. 1.7(a)) uses high-gain amplifier and series-shunt feedback to regulate the voltage to a desired reference level. A major problem with the linear regulator is that its efficiency is determined by the ratio V_{out}/V_{in} where V_{out} and V_{in} are the output and input voltage, respectively. Hence, the linear regulator has poor efficiency at low output voltage. A switching converter (Fig. 1.7(b)) employes duty cycle controlled switches to convert a DC voltage into a pulse train, followed by an LC low-pass filter to extract its DC component. However, the off-chip inductor increases the form factor of the system and thus prohibits its use in form-factor constrained ULP platforms. The switched capacitor VRM (SC-VRM) employs a capacitor array to store and transfer charge. Voltage conversion is achieved by transferring charge using duty cycle controlled power switches. SC-VRM can have a high efficiency when the output voltage is close to the ideal conversion output, but efficiency decreases rapidly as the output deviates from the ideal output voltage. Its compactness and

capability to achieve high conversion ratio make SC-VRM an ideal candidate for use in ULP platforms.

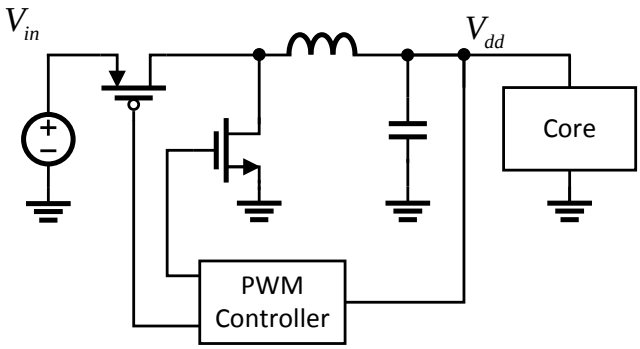
Design of high-efficiency SC-VRM for ULP platforms is made challenging due to the large step-down ratio [12, 5] and the light load conditions ($\approx 10\,\text{nA}$). SC-VRMs with pulse frequency modulation (PFM) control [5] and capacitance modulation [12] have been employed to boost light load efficiency up to 74%. A hybrid converter [13] has been proposed to address energy delivery for loads in the range of 5 nA-to-500 nA, with efficiency up to 56%. Ultra low power clock generation and level shifter are employed to reduce the switching loss, which degrades light load efficiency. To mitigate the conversion ratio problem, the stacked voltage domain approach [14] has been proposed where multiple cores are connected in series to lower the step-down ratio. However, this approach needs push and pull linear regulators in order to compensate for voltage fluctuations in the intermediate supply nodes caused by current mismatch. Most of the conventional approaches target reducing converter losses in existing topologies. However, SC-VRM includes intrinsic charge sharing loss [15], gate drive loss, and other overhead, that tends to severely degrade the VRM light load efficiency. Thus, alternative architectures are needed to increase the VRM efficiency for ULP platforms.

1.1.2 Sensing Subsystem

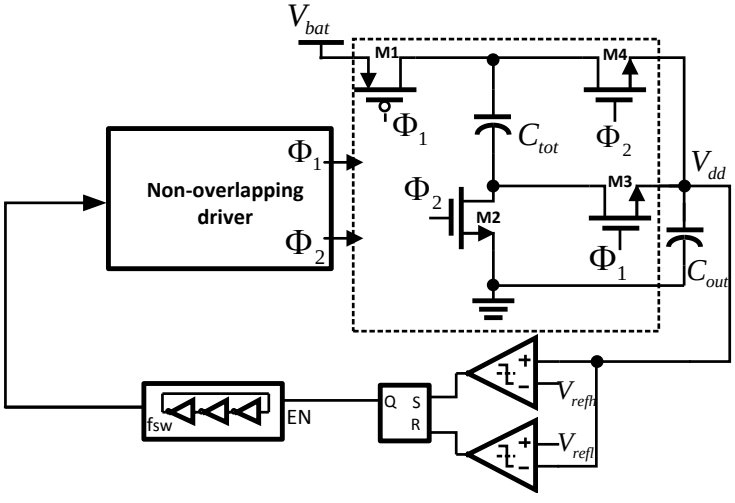
The sensing subsystem contains various types of sensors, such as image, biomedical, or chemical sensors, to convert physical signals into electrical signals for further processing. In applications such as CMOS image sensor based embedded vision where sensing and information processing co-exist, the interface energy between the two subsytems can dominate. One approach to address the resulting communication challenge is to tightly integrate sensing and computation. Previous work in integrating computation into the CMOS image sensor array falls into one of two categories. In the first, the pixel architecture is modified (see Fig. 1.8(a)) to enable simple computations such as 2D convolution [16], image filtering [17, 18], compressive image sensing [19], matrix transformations [20], Gaussian pyramid [21], and image decomposition [22]. These approaches suffer from a loss in *fill-factor* or the spatial resolution because the modified pixel occupies an area that can be as high as 8× greater



(a)

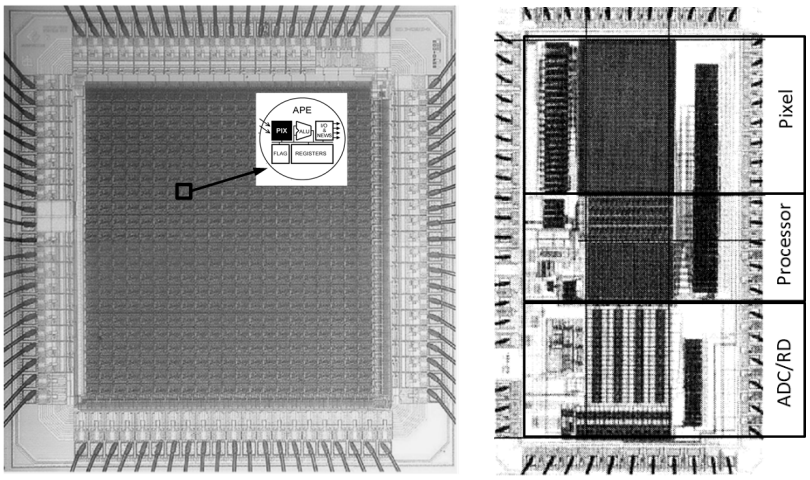


(b)



(c)

Figure 1.7: Three commonly used DC-DC converter topologies: (a) linear regulator, (b) switching converter, and (c) SC-VRM.



(a) (b)

Figure 1.8: Integrated sensing and computing falls into two categories: (a) modified pixel architecture leading to a loss of fill factor [17], and (b) attaching digital/analog processor in APS peripheral for low level filtering functions [23].

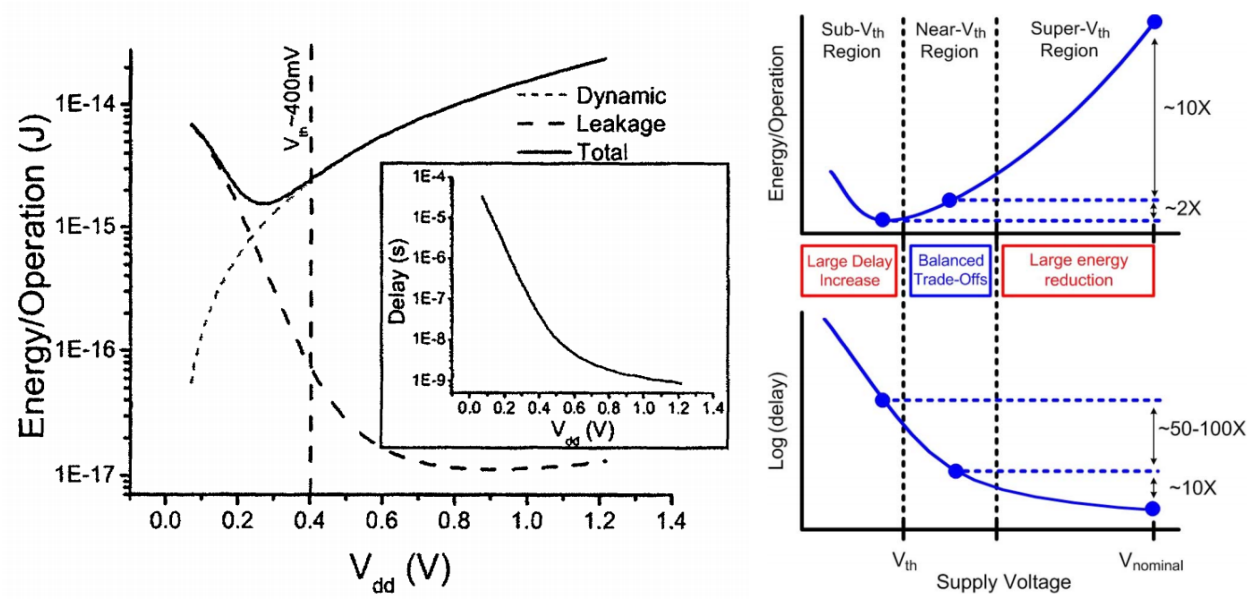
than the standard pixel architecture. In the second, very simple analog processing functions are embedded in the periphery of the APS array (see Fig. 1.8). These include convolution [23], random projections for compressed sensing [24], and difference of Gaussian (DOG) [25]. The main limitation of these approaches is the absence of learning capabilities since only low level image processing algorithms such as filtering are supported. This lack of learning prevents the system from adapting model parameters to compensate for the non-idealities in the hardware platform such as the non-linearity and noise in the sensor and the peripheral circuitry.

1.1.3 Digital Low Power Design

Conventional approaches for power reduction rely on device level techniques such as feature size scaling and body biasing [26]; circuit level techniques such as transistor sizing [27] and voltage scaling [28]; and architectural level techniques such as algorithm transformation [29], clock/power gating [30, 31], dynamic voltage and frequency scaling (DVFS) [32]. As CMOS technology scales into sub-10 nm, these traditional knobs such as supply voltage, frequency, and threshold voltage for energy reduction are becoming ineffective due to leakage as well

as power density concerns. Moreover, the conventional techniques adopt a worst-case design methodology to ensure error free operation. The resulting large margin limits the achievable energy efficiency.

The work of Calhoun et al. [33] and Zhai et al. [34] leads to the discovery of the minimum energy operation point (MEOP) in digital integrated circuits which arises from the trade-off between dynamic energy E_{dyn} and leakage energy E_{leak} in sub-threshold domain (see Fig. 1.9(a)). Supply voltage scaling results in quadratic reduction in E_{dyn}, and an exponential increase in the delay and thus in E_{leak}. The resulting MEOP is defined via the tuple (V_{dd}^*, f_{clk}^*, E_{op}^*) where V_{dd}^* is the optimum supply voltage, f_{clk}^* is the optimum clock frequency, and E_{op}^* is the optimum energy per operation. Operating circuits in sub-threshold might lead to severe performance loss due to the increased delay. To compensate for the performance loss, researchers have also proposed near-threshold operation where the supply voltage is scaled down to $400 - 500\,\text{mV}$ [8] (see Fig. 1.9(b)). Near-threshold computing offers $10\times$ energy benefits with relatively small performance loss and is considered a good trade off between super and sub-threshold operations. Both system level studies and IC implementations exist for near/sub-threshold computing. Markovic et al. [35] study the impact of activity factor and various design parameters on near-threshold operation and propose suitable logic families for near-threshold design. The work concludes that near-threshold operation can provide a $10\times$ throughput increase with a 20% energy increase relative to the MEOP. Dreslinski et al. [36] study architectural optimization of parallel chip multi-processors (CMP) operating in near-threshold. Kwong [37] provides a design methodology for sub-threshold logic with emphasis on device sizing. Processors operating in near/sub-threshold [38, 39, 40, 41] as well as custom digital signal processing (DSP) kernels [42, 43] have also been proposed. The wide adoption of sub/near-threshold design is limited due to the performance loss and increased PVT variations [11]. The conventional approach employs transistors up-sizing and extensive verification using computer aided design (CAD) design flow [37]. This worst case design methodology limits the achievable energy efficiency.



(a) (b)

Figure 1.9: Near/Sub-threshold operation: (a) MEOP exists due to the balance between dynamic and leakage energies [33], and (b) near-threshold computing offers trade-off between throughput and energy efficiency [8].

1.1.4 Robust System Design

Defects and errors originating from various sources necessitate robust system design for next generation ULP platforms. Errors can be caused by imperfections in fabrication such as scratches from wafer mishandling, mask misalignment and over/under-etching [44]. These imperfections, referred to as defects, can cause unpredictable open or short circuits in the fabricated chip, leading to circuit stuck-at-faults. In addition, near/sub-threshold computing presents new challenges for robust system design. The increased PVT variations [8] lead to higher probability of timing errors.

Robust system design dates back to the work of von Neumann [45] who showed that reliable networks can be designed with a cascade of three input majority gates, if the component probability of failure $p_e < 0.0073$, and that reliable computation is impossible if $p_e \geq \frac{1}{6}$. Techniques for various design abstractions have been proposed and are summarized next.

At the circuit level, yield enhancement routing [46] and floor planning [47] techniques

have been proposed. These techniques target improving yield at manufacturing time. In [48], Markov random field (MRF) logic is proposed to enhance noise immunity under low voltage operation. The proposed logic is able to achieve great robustness improvement, but the overhead is prohibitive. Techniques such as transistor sizing [37] and body biasing [49] have been proposed to reduce variations in sub/near-threshold designs. Circuit hardening techniques [50] have been proposed to mitigate single event transients (SET) on logic circuits.

At logic and microarchitecture level, conventional robust system design methods employ redundancy based approaches. For example, in N modular redundancy (NMR), the design is robustified by replicating the module N times followed by majority voting to obtain the final results. NMR incurs large (N fold) area and energy overhead, thus is not suitable for use in low power platforms. In [51, 52], RAZOR is proposed as a low overhead microarchitecture level technique for detecting timing errors. RAZOR employs a specially designed shadow latch to detect late-arriving signals and a recovery scheme to re-execute the erroneous instruction. RAZOR is able to achieve 44% energy savings over the worst case design point while operating close to point of first failure (PoFF) with an error rate of 0.1% [51]. RAZOR's deterministic error compensation makes it well-suited for applications where 100% correctness is important. Emerging machine learning applications have a relaxed notion of correctness that can potentially be utilized to enhance robustness and improve energy efficiency.

In contrast to logic or circuit level techniques, algorithm and system level techniques can take advantage of application level performance metrics to enhance system level robustness. Emerging machine learning applications employ performance metrics that are statistical in nature [53]. For example, the feature extractors consisting of filter banks employ signal-to-noise ratio (SNR) as the design metric, and the classification engines employs true positive rate (TP rate), false positive rate (FP rate) or detection rate (P_{det}) as the design metric [54, 55]. Statistical metrics result in inherent error tolerance to small magnitude errors. In data driven hardware resiliency (DDHR) [54], stuck-at faults and various system non-idealities are treated as feature inputs into the machine learning algorithm. Through training with error affected features, the resulting classification/regression engine compensates for these errors. The resulting engine can thus perform correct classification and regression in

the presence of errors. In [56], adaptive boosting is employed to train in the presence of hardware errors. Data driven methods are effective at handling static errors such as stuck-at faults. However, the data driven nature of these approaches requires the error statistics to be the same during training and testing, which might not be true for dynamic errors such as timing errors.

Statistical error compensation (SEC) [53, 57, 58, 61, 60] (see Fig. 1.10) is a class of system level error compensation techniques that utilize signal and error statistics. SEC employs detection and estimation theory to compensate for the errors in the main computation block, and thus can tolerate a much higher error rate compared with logic level techniques. Algorithmic noise-tolerance (ANT) [57] employs an explicit estimator block to compensate for the most significant bit (MSB) first errors in the main DSP block. ANT has been shown to provide up to 65% energy savings with little loss of performance. Stochastic sensor-network-on-a-chip (SSNOC) [58] employs statistically similar decomposition and robust estimation theory to compensate for errors and achieves up to 5.8× energy savings. Soft NMR [61] makes explicit use of error probability mass functions (PMFs) to provide up to 10× improvement in robustness with 35% energy savings. Likelihood processing [60] utilizes bit-level error statistics to perform inference and has been shown to provide up to 14× improvement in robustness with 25% energy savings. In general, circuit level error resiliency techniques [51, 62] enable operation close to point of first failure (PoFF) or in the low error rate ($<$0.1%) regime. In comparison, system level error resiliency techniques such as SEC [53, 57, 63, 61] can operate in the high error rate ($>$10%) regime. Previous studies have shown that a reduced precision replica ANT (RPR-ANT) protected ECG processor [64] and MRF stereo matching block [65] can be fully functional at an error rates of 58% and 21.3%, respectively. However, the improved robustness in RPR-ANT comes at the price of 30% [64] to 40% [65] complexity overhead due to the use of explicit estimator blocks. Thus, improved SEC techniques need to be developed for them to be applicable to more complex signal processing and inference kernels.

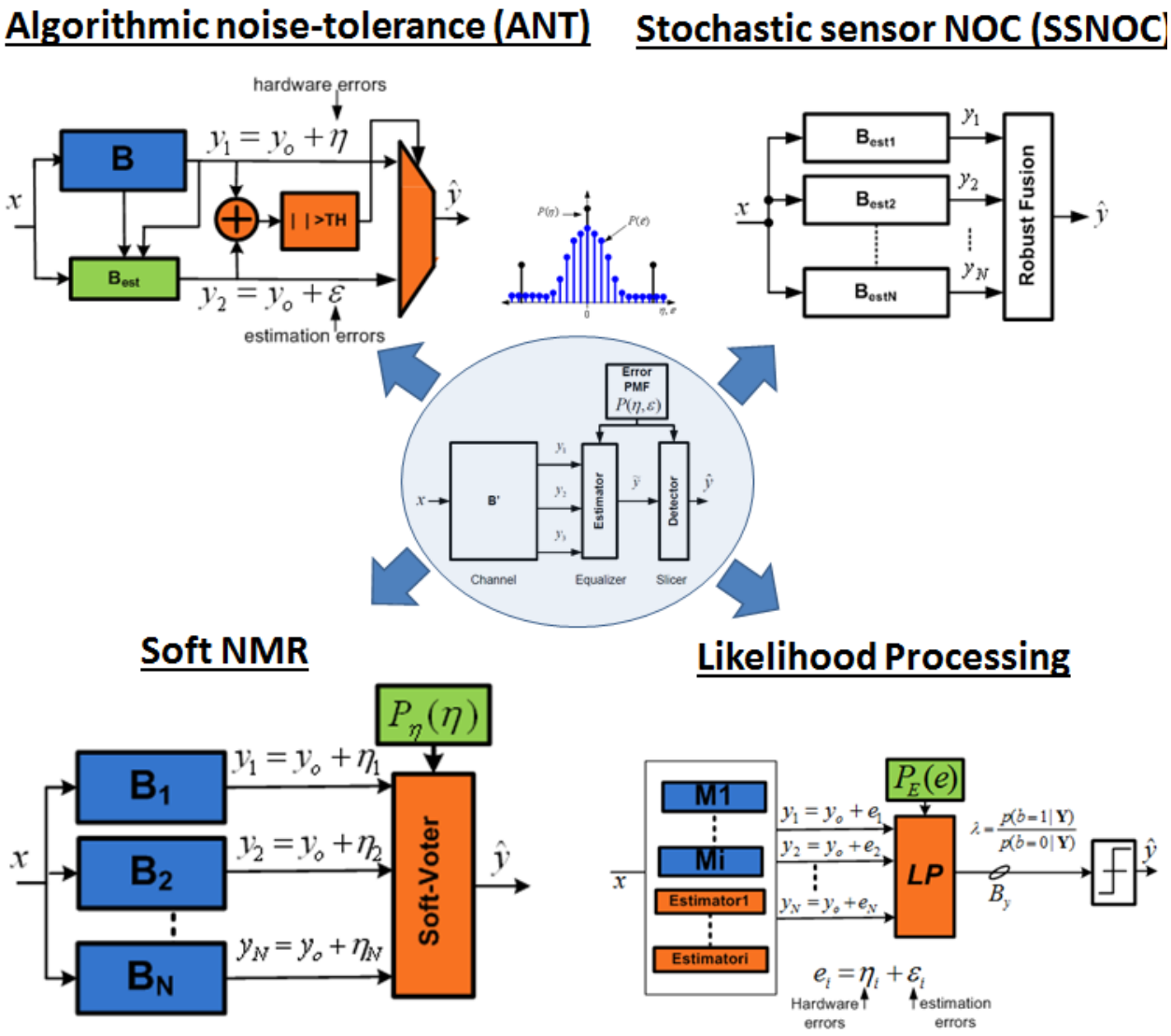


Figure 1.10: SEC techniques (a) ANT [57], (b) SSNOC [58], (c) soft NMR [59] and (d) likelihood processing [60].

1.2 Dissertation Contributions and Organization

The design of ULP platforms for machine learning applications is challenging due to the energy delivery, communication and the tightly coupled computation and robustness challenges. Conventional design approaches optimize the energy delivery, sensing, and information processing separately. In this dissertation, we tackle these problems by (1) embedding information processing into the energy delivery and sensing subsystem to eliminate the voltage conversion loss and interface overhead, and (2) computing at the limits of energy efficiency and thus robustness, and employing SEC techniques to compensate for the resultant hardware errors. The major contributions and organization of the dissertation are summarized as follows:

Chapter 2 presents the C-VRM approach where the information processing subsystem is embedded into the energy delivery subsystem. The C-VRM employs multiple voltage domain stacking and core swapping to achieve high total system energy efficiency in near/sub-threshold region. A prototype IC incorporating a C-VRM and an SC-VRM supplying energy to an 8-tap fully folded FIR filter core is implemented in a 1.2 V, 130nm CMOS process. Measured results indicate that the C-VRM has up to 44.8% savings in system-level energy per operation compared to the SC-VRM system, and an efficiency ranging from 79% to 83% over an output voltage range of 0.52 V to 0.6 V.

Chapter 3 presents an in-sensor computing architecture which (mostly) eliminates the sensor-processor interface by embedding information processing into the noisy sensor fabric in analog and retraining the hyperparameters in order to compensate for non-ideal computations. The resulting architecture, referred to as the Compute Sensor - a sensor that computes in addition to sensing - represents a radical departure from the conventional. A Compute Sensor for image data is designed by embedding both feature extraction and classification functions in the analog domain in close proximity to the CMOS active pixel sensor (APS) array. Significant gains in energy efficiency are demonstrated using behavioral and energy models in a commercial semiconductor process technology.

Chapter 4 presents embedded algorithmic-noise tolerance (E-ANT), a new low overhead SEC technique aiming at enhancing the robustness and energy efficiency of signal processing

and machine learning kernels. E-ANT operates by reusing part of the main block operation as estimation and thus embedding the estimator block into the main block. Such embedding can be achieved at various levels. At the architecture level, we propose ARCH-ANT, which uses data path decomposition to embed the reduced precision replica estimator into the main block. At the algorithm level, we propose ALG-ANT, which employs additional optimization constraints during algorithm to architecture mapping to design incremental refinement architectures. System level simulation results in commercial 45nm process shows large energy efficiency and robustness improvement.

Chapter 5 presents several probabilistic error models for machine learning kernels implemented on low-SNR circuit fabrics where errors arise due to voltage overscaling (VOS), process variations, or defects. Four different variants of the additive error model are proposed that describe the error PMF. Analytical expressions for the error PMF are derived. Performance prediction of a support vector machine (SVM) based classifier using these models indicates that when comparing Monte Carlo with HDL simulations, probability of detection P_{det} estimated using the model is within 3% for VOS error when the error rate $p_\eta \leq 80\%$, within 5% for process variation error and within 2% for defect errors when the defect rate (the percentage of circuit nets subject to stuck-at-faults) p_{saf} is between 10^{-3} and 0.2.

Chapter 6 presents the design of error-resilient machine learning architectures by employing a distributed machine learning framework referred to as *classifier ensemble* (CE). CE combines several simple classifiers to obtain a strong one. In contrast, centralized machine learning employs a single complex block. The random forest (RF) and the support vector machine (SVM), which are representative techniques from the CE and centralized frameworks, respectively, are compared. Employing the breast cancer data set in the UCI machine learning repository and architectural-level error models in a commercial 45 nm CMOS process, it is determined that RF-based architectures are significantly more robust than SVM architectures in the presence of timing errors due to process variations in near-threshold voltage (NTV) regions ($0.3\,\mathrm{V} - 0.7\,\mathrm{V}$). Additionally, an error weighted voting technique that incorporates the timing error statistics of the NTV circuit fabric is proposed to further enhance the robustness of RF architectures. Simulation results confirm that the error weighted voting achieves a P_{det} that varies by only 1.4%, which is 12× lower than SVM.

Chapter 7 concludes this dissertation and provides directions for future research activities.

Chapter 2

COMPUTE VRM

The emerging applications in machine learning require the design of ULP platforms with limited energy supply. Energy per operation (E_{op}) of such systems is equal to the energy extracted from the battery per operation E_{bat} and is given by $E_{bat} = E_{op} = E_{vrm} + E_{core}$, where E_{vrm} and E_{core} are the energy consumption per instruction by the VRM and the core, respectively. The conventional approach is to design the VRM to maximize its efficiency η at a pre-specified core supply voltage V_{dd} and core/load current I_{load} (see Fig. 2.1).

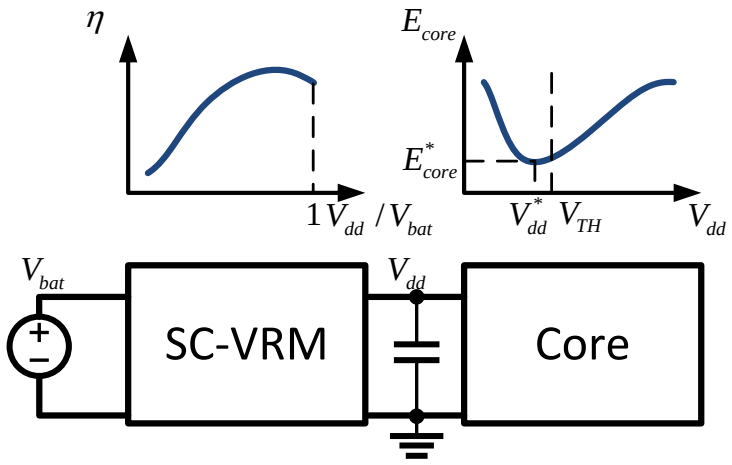


Figure 2.1: Conventional design approach addresses VRM design and core design separately. Due to the tradeoff between dynamic and leakage energy, minimum energy operation point (MEOP) of compute cores usually lies in near or sub-threshold regime. However, the resulting high conversion ratio often results in poor VRM efficiency.

Sub/near-threshold computing (NTC) has been proposed [66, 67] to minimize E_{core} by operating the core close to its minimum energy operating point (MEOP) where V_{dd} is regulated in the 400 mV-to-600 mV range. The battery voltage V_{bat} is typically in the range of 1.2 V to 3.6 V [68]. This large gap between V_{bat} and V_{dd} requires the voltage regulator module (VRM)

to achieve a high step-down ratio. Among the VRM topologies, the switched capacitor VRM (SC-VRM) is attractive as it can achieve high conversion ratio and is amenable to on-chip integration [15, 69]. Design of high-efficiency VRM for ULP platforms is challenging due to the large step-down ratio [5, 12] and light load conditions due to NTC.

Various approaches have been proposed to address the efficiency issue in SC-VRM under light load conditions. SC-VRM with pulse frequency modulation (PFM) control [5, 12] has been employed to boost light load efficiency up to 74%. A hybrid converter [13] has been proposed to address energy delivery for loads in the range of 5 nA-to-500 nA, with efficiency up to 56%. The stacked voltage domain approach [14] has been proposed where multiple cores are connected in series to lower the step-down ratio. This approach needs push and pull linear regulators in order to compensate for voltage fluctuation in the intermediate supply nodes caused by current mismatch.

In this chapter, we propose the compute VRM (C-VRM) which exploits the similarity between charge transfer in an SC-VRM and CMOS logic. Computation in CMOS occurs via transfer of charge between supply/ground nodes and capacitive output nodes. This transfer is controlled by MOS transistor switches. Energy delivery in a SC-VRM occurs in a similar manner with power switches controlling the transfer of charge from the battery to the core. The C-VRM exploits this similarity by replacing the power switches in a SC-VRM with CMOS compute cores. In doing so, the proposed C-VRM provides the following advantages: (1) eliminates driver loss, bottom plate capacitor loss, and charge transfer loss to enhance voltage conversion efficiency, and (2) seamlessly integrates energy delivery and computation to provide a unified platform that enables the minimization of total system (VRM+core) energy E_{op}. The C-VRM concept is validated by: (a) developing energy models for the C-VRM and the SC-VRM, and employing these in system simulations to evaluate the benefits of C-VRM in energy per operation E_{op} and efficiency η, and (b) implementing a prototype IC incorporating a C-VRM and a SC-VRM supplying energy to an 8-tap folded FIR filter core in a 1.2 V, 130 nm CMOS process to verify the benefits of C-VRM via measured results.

The rest of the chapter is organized as follows: Section 2.1 describes the background of the conventional SC-VRM and develops energy models to evaluate its efficiency. Section 2.2 presents the C-VRM and develops energy models to compare with the conventional SC-

VRM. Section 2.3 describes the prototype IC consisting of both the conventional SC-VRM system and the C-VRM. Test results in Section 2.4 demonstrate the improvement in energy and converter efficiency. Conclusions and future work are addressed in Section 2.5.

2.1 Background

This section reviews the design of a conventional SC-VRM system. An energy model is derived to reveal the fundamental loss mechanisms in a SC-VRM. A core energy model is also derived for use in system simulations in the following sections.

2.1.1 Intrinsic Loss in SC-VRM

A block diagram of a 2:1 SC-VRM is shown in Fig. 2.2(a), where a set of charge transfer capacitors and switches are connected in different configurations in each clock phase to convert the voltage. Since the charge transfer procedure involves direct connection of voltage sources and capacitors, the current will be impulsive and lead to an intrinsic charge transfer loss E_{CTL}. As pointed out in [70, 71], E_{CTL} depends on the operational domain of the SC-VRM, i.e., complete charge, partial-charge, or no-charge. In near/sub-threshold operation with light load, driver loss will degrade light load efficiency severely and should be minimized. Thus, we assume that the SC-VRM is operating in the complete charge operation domain so as to maximize the charge transferred to the output per converter switching cycle. Figure 2.2(b) illustrates the source of E_{CTL} in the context of a simple 1:1 SC-VRM where the charge is transferred from V_{bat} to V_{dd} with a flying capacitor C_{sc}. In Fig. 2.2(b), it can be shown that in each phase ($\Phi 1$ and $\Phi 2$), the energy loss due to charge sharing is $\frac{1}{2}C_{sc}(V_{bat} - V_{dd})^2$. Thus, the intrinsic charge transfer loss during every switching cycle is:

$$E_{CTL} = C_{sc}(V_{bat} - V_{dd})^2 \tag{2.1}$$

Note that (2.1) does not depend on switch resistances $R1$ and $R2$. Therefore, E_{CTL} in (2.1) represents a fundamental loss mechanism in a conventional SC-VRM. The C-VRM extracts

useful computation from this loss and thereby improves E_{op}.

2.1.2 SC-VRM Energy Model

In order to evaluate the converter efficiency under different load conditions, a power model for the SC-VRM is necessary. For simplicity of analysis, we choose a 2:1 ladder SC-VRM as shown in Fig. 2.2(a). However, the analysis method can be extended to a higher conversion ratio. There are four major loss components in the conventional SC-VRM:

2.1.2.1 Charge Transfer Loss (E_{CTL})

As with any SC-VRM, there is the loss E_{CTL} during each charge transfer. In [15, 72], SC-VRM is modeled as an ideal transformer (see Fig. 2.2(c)) representing a conversion ratio of N, and E_{CTL} is captured by a series resistance R_{ctl}, and is given by:

$$E_{CTL} = I_{core}^2 R_{ctl} T_{sw} \tag{2.2}$$

where I_{core} is the load current and T_{sw} is the switching period. Substituting $I_{core} = \frac{2C_{sc}\Delta V}{T_{sw}}$ and $R_{ctl} = \frac{1}{4C_{sc}f_{sw}}$ into (2.2), we get:

$$E_{CTL} = \frac{I_{core}^2}{4C_{sc}f_{sw}} T_{sw} = C_{sc}(\Delta V)^2 \tag{2.3}$$

where f_{sw} is the switching frequency of the SC-VRM, and ΔV is the difference between $\frac{V_{bat}}{N}$ (ideal output) and regulated V_{dd}. Note that (2.1) and (2.3) are identical when $\Delta V = V_{bat} - V_{dd}$.

2.1.2.2 Gate Drive Loss (E_{GDL})

The SC-VRM requires explicit power switches to transfer charge. A driver circuit, such as a super buffer, is therefore needed, resulting in additional losses. Assuming the gate capacitance of the power switch is C_{switch}, the gate drive loss per instruction E_{GDL} can be

expressed as:

$$E_{GDL} = C_{switch} V_{bat}^2 f_{sw} / f_{clk-S} \tag{2.4}$$

where E_{GDL} is calculated per one core clock period $T_{clk-S} = 1/f_{clk-S}$, and f_{clk-S} is the core clock frequency. This definition of E_{GDL} allows a direct comparison with the energy consumption of the core E_{core}.

2.1.2.3 Bottom Plate Capacitor Loss (E_{BPCL})

The bottom plate capacitor C_{bottom} is the parasitic capacitor between the bottom plate of C_{sc} and the substrate (see Fig. 2.2(a)). C_{bottom} scales with the area of C_{sc} and can be as high as 5% of C_{sc} [73]. Since the bottom plate of the C_{sc} is not always grounded, during every switching cycle, C_{bottom} will be charged and discharged, as shown in Fig. 2.2(a). Assuming the ratio of C_{bottom} to C_{sc} is γ, this will lead to an energy loss given by:

$$E_{BPCL} = \gamma C_{sc} V_{dd}^2 f_{sw} / f_{clk-S} \tag{2.5}$$

2.1.2.4 Control Loss (E_{CL})

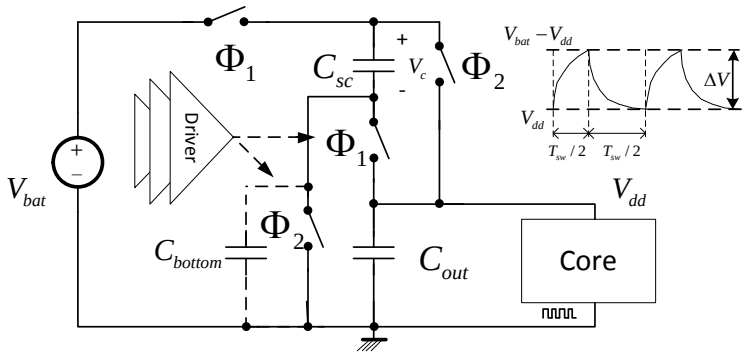
Control loss represents a constant loss in the SC-VRM and will degrade light load efficiency. Assuming the effective load capacitance of control circuit is C_{ctrl}, and the control circuit frequency is f_{ctrl}, the control loss can be expressed as:

$$E_{CL} = C_{ctrl} V_{bat}^2 f_{ctrl} / f_{clk-S} \tag{2.6}$$

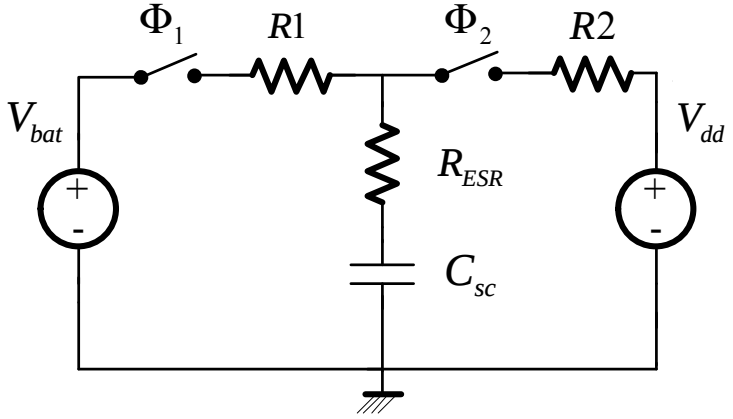
2.1.3 Core Energy Model

There are two types of energy consumption in a core operating in near/sub-threshold region: dynamic energy and leakage energy. A unified model that accounts for both components has been proposed in [74]:

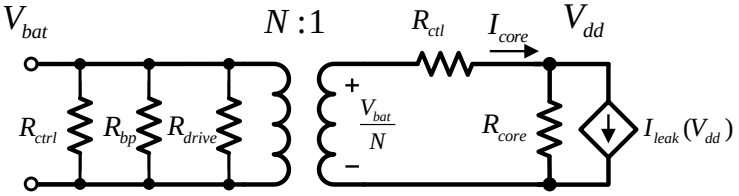
$$E_{core} = \alpha C_{core} V_{dd}^2 + V_{dd} I_{leak}(V_{dd}) \frac{1}{f_{clk}} \tag{2.7}$$



(a)



(b)



(c)

Figure 2.2: Conventional SC-VRM architecture: (a) block diagram of a 2:1 SC-VRM, (b) charge sharing loss mechanism, and (c) a simplified energy transfer model.

$$I_{leak}(V_{dd}) = \mu C_{ox}\frac{W}{L}(m-1)V_T^2 e^{\frac{-V_t}{mV_T}} e^{\frac{-\eta_d V_{dd}}{mV_T}}(1-e^{\frac{-V_{dd}}{V_T}}) \tag{2.8}$$

where α is the core activity factor, C_{core} is the load capacitance in the core, V_{dd} is the supply voltage, V_t is the threshold voltage, V_T is the thermal voltage, μ is the carrier mobility, C_{ox} is the gate capacitance per W/L, m is a constant related with sub-threshold slope factor, and η_d is the drain induced barrier lowering (DIBL) coefficient. This model captures the trade-off between the dynamic and leakage energy, which leads to the MEOP [74], as defined via the 3-tuple $(E^*_{core}, V^*_{dd}, f^*_{clk})$, where E^*_{core} is the energy at MEOP, V^*_{dd} is the optimum voltage, and f^*_{clk} is the energy optimum frequency. The core is modeled as a resistor R_{core} in parallel with a leakage current source $I_{leak}(V_{dd})$ (see Fig. 2.2(c)).

2.2 C-VRM System Design

This section presents the system design of the proposed C-VRM. An analytical energy model for the C-VRM is developed to compare its efficiency with the conventional SC-VRM system.

2.2.1 Principle of Operation of the C-VRM

C-VRM utilizes computational cores as switches to perform computation and transfer charge. The compute cores are used as distributed power switches and perform the dual functions of energy transfer and information processing.

The proposed C-VRM operates in principle the same as an interleaved SC-VRM. To illustrate this, Fig. 2.3 describes the operation of an interleaved 2:1 SC-VRM and a 2:1 C-VRM. For the interleaved SC-VRM, in the first phase (Φ_1), charge is stored in the flying capacitor C_1 and released by C_2; in the second phase (Φ_2), charge is released by C_1 and stored in C_2. The 2:1 C-VRM implements the same charge transfer function described above but without explicit power switches/drivers. In Φ_1, the core in the high voltage domain (CH) is clock gated while the core in low voltage domain (CL) is active. Thus, charge is stored in C_1 and released by C_2. In Φ_2 , CL is clock gated while CH is active, so that charge is released

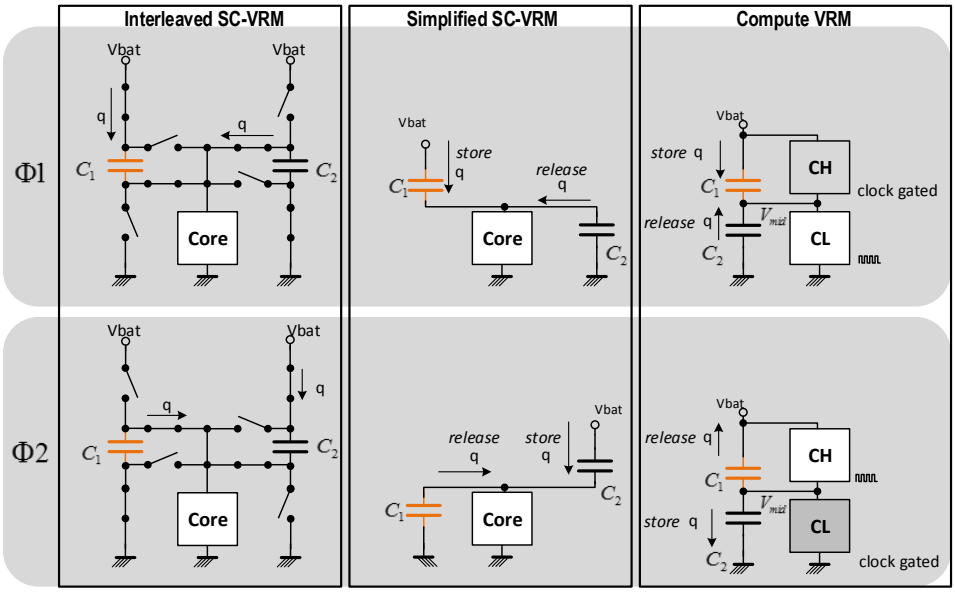


Figure 2.3: The C-VRM principle for $N = 2$.

by C_1 and stored in C_2. The C-VRM achieves improved energy efficiency compared to the conventional SC-VRM as the losses associated with the driver, bottom plate capacitor, and intrinsic charge transfer are eliminated. Furthermore, it incorporates computation as an intrinsic part of its energy delivery functionality.

The out-of-phase operation of CH and CL (core swapping) requires data transfer between two voltage domains, as shown in Fig. 2.4. At the end of Φ_1 and Φ_2, data is transferred between CL and CH by adding an extra core swapping cycle. The core swapping has negligible effect on total throughput, so long as the swap frequency is low compared to C-VRM core clock frequency f_{clk-C}. To ensure this condition, CH and CL employ continuous voltage and frequency scaling (CVFS), where f_{clk-C} tracks the decaying voltage (V_{CH} or V_{CL}) across the active core. The voltage of the intermediate node (V_{mid}) is permitted to vary by 80 mV-200 mV. As a result, V_{CH} and V_{CL} varies between 500 mV and 700 mV, and f_{clk-C} tracks the instantaneous voltage by employing an on-chip critical path replica (CPR) oscillator.

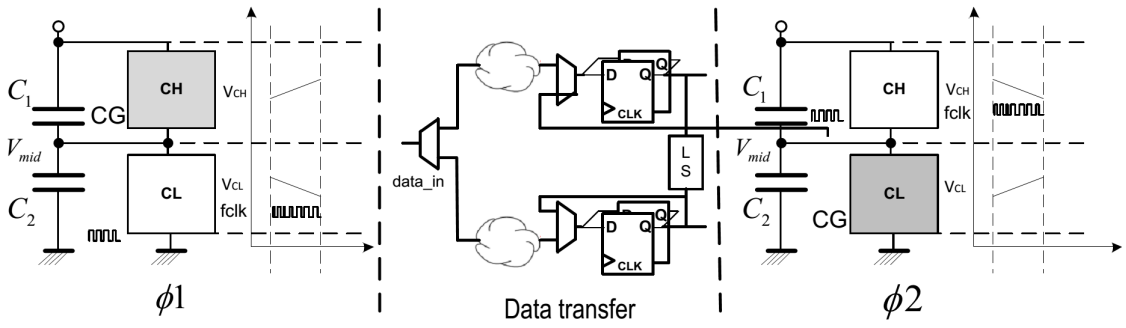


Figure 2.4: Data transfer in the C-VRM during core swapping.

2.2.2 C-VRM Energy Model

An energy model of the C-VRM is necessary to compare its system level energy consumption E_{op} with that of the SC-VRM system. The C-VRM eliminates driver loss and charge transfer loss associated with the conventional SC-VRM system. However, the variable core voltage results in data transfer loss and increased core energy. There are three major energy components in the C-VRM: core energy (E_{core-C}), data transfer loss (E_{DTL}), and control loss (E_{CL-C}), all of which need to be characterized.

Since the V_{dd} across each core during its operation varies, E_{core-C} is time varying, as shown in Fig. 2.5. Assume that M operations are completed during M clock cycles that comprise the active period. In the m^{th} ($m = 1, 2, ...M$) cycle, the average voltage across the core is denoted as $V_{dd}(m)$ and the clock period during the m^{th} operation is denoted as $T_{clk-C}(m)$. The supply voltage V_{dd} drops from a pre-defined voltage $V_{dd}(0)$ to another pre-defined voltage $V_{dd}(M)$ over M clock cycles, as shown in Fig. 2.5. In the test chip, $V_{dd}(0)$ and $V_{dd}(M)$ are chosen to be 500 mV and 700 mV, respectively, and the value of M ranges from 96 to 131 depending on the core activity factor α. We also assume that CH and CL see the same voltage profile ($V_{dd}(m)$) during active periods in order to simplify the analysis. Thus, E_{core-C}, E_{DTL}, and E_{CL-C} can be calculated as

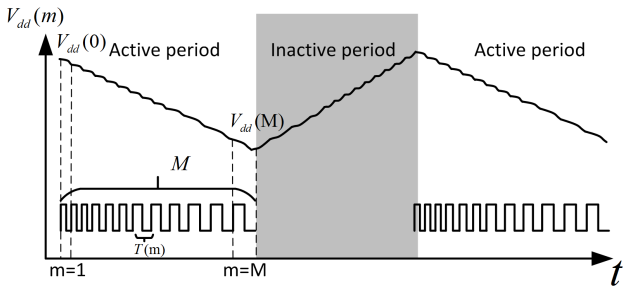


Figure 2.5: The variable supply voltage $V_{dd}(m)$ results in a time varying clock period $T_{clk-C}(m)$.

$$E_{core-C} = \frac{1}{M}\sum_{m=1}^{M}[\alpha C_{core}V_{dd}^2(m) + I_{leak}(m)V_{dd}(m)]T_{clk-C}(m) \tag{2.9}$$

$$E_{DTL} = \frac{C_{reg-C}V_{bat}^2}{M} \tag{2.10}$$

$$E_{CL-C} = \frac{C_{ctrl-C}V_{bat}^2 f_{ctrl-C}}{f_{clk-C}} \tag{2.11}$$

where α is the core activity factor, $I_{leak}(m)$ is the leakage current at the supply voltage of $V_{dd}(m)$, C_{reg-C} is the total load capacitance of data transfer logic, C_{ctrl-C} is the load capacitance of the control circuitry, f_{ctrl-C} is the equivalent control frequency, and f_{clk-C} is the core clock frequency. Figure 2.6 shows a C-VRM with N cores (thus N voltage domains). From the principle of charge conservation, the following set of equations holds:

$$Q = \alpha C_{core}V_{dd}(m) \tag{2.12}$$

$$Q\frac{N-1}{N} = \frac{C_{sc}}{N}V_{dd}(m-1) - \frac{C_{sc}}{N}V_{dd}(m) \tag{2.13}$$

where (2.12) describes the charge consumed by the core in the m^{th} clock cycle, and (2.13) describes the charge conservation at node **a** in Fig. 2.6. Equations (2.12) and (2.13) can be used to solve for $V_{dd}(m)$ $(m = 1..., M)$ to obtain:

$$V_{dd}(m) = [\frac{C_{sc}}{(N-1)\alpha C_{core} + C_{sc}}]^m V_{dd}(0) \tag{2.14}$$

Next, substituting $m = M$ in (2.14), we solve for M as follows:

$$M = \frac{\ln \frac{V_{dd}(M)}{V_{dd}(0)}}{\ln[\frac{C_{sc}}{(N-1)C_{core}+C_{sc}}]} \tag{2.15}$$

The clock period $T_{clk-C}(m)$ is obtained as the average of the critical path delays at $V_{dd}(m)$ ($T_d(V_{dd}(m))$) and $V_{dd}(m-1)$ ($T_d(V_{dd}(m-1))$):

$$T_{clk-C}(m) = \frac{T_d(V_{dd}(m)) + T_d(V_{dd}(m-1))}{2} \tag{2.16}$$

Therefore, the E_{op} of the conventional SC-VRM system is given by:

$$E_{op-SC} = E_{core} + E_{CTL} + E_{GDL} + E_{BPCL} + E_{CL} \tag{2.17}$$

where E_{core}, E_{GDL}, E_{BPCL}, and E_{CL} are defined in (2.2)-(2.7). Similarly, the E_{op} of the C-VRM is obtained as:

$$E_{op-C} = E_{core-C} + E_{DTL} + E_{CL-C} \tag{2.18}$$

where E_{core-C}, E_{DTL}, and E_{CL-C} are defined in (2.9)-(2.11), and V_{dd}, M and $T_{C,CLK}$ are obtained from (2.14)-(2.16). Measured results from a prototype test chip in 130 nm CMOS (see Fig. 2.21) indicate that (2.17) and (2.18) accurately models the energy consumption of the SC-VRM and the C-VRM, respectively. Energy saving can be obtained if $E_{op-C} < E_{op-SC}$. Next, we determine conditions under which the C-VRM is more energy efficient, as compared to an SC-VRM system.

2.2.3 C-VRM System Design

In the rest of this chapter, we will assume that the C-VRM has $N = 2$ cores to simplify the analysis. System simulations are performed to compare the energy efficiencies of the C-VRM and the SC-VRM system. The battery voltage V_{bat} is assumed to be 1.2 V. For the SC-VRM, we use an $N = 2$ ladder topology as shown in Fig. 2.2(a), with flying capacitor C_{sc} chosen to be 500 pF. C_{switch} is chosen such that the SC-VRM is operating in slow switching

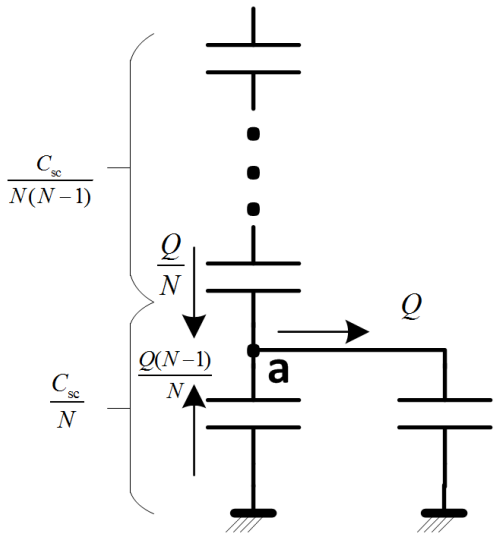


Figure 2.6: The principle of charge conservation in the C-VRM.

limit (SSL) and fast switching limit (FSL) boundary to balance shunt and series losses. The bottom plate capacitance C_{bottom} is assumed to be 2% of C_{sc} and C_{ctrl} is assumed to be 1% of C_{sc}. For the C-VRM, we also choose the $N = 2$ topology as shown in Fig. 2.3. For fairness of comparison, we constrain the total charge transfer capacitance $(C_1 + C_2)$ in the C-VRM to equal C_{sc} in the SC-VRM. The 500 pF capacitor is split equally between C_1 and C_2. Each 250 pF capacitance supplies one of the cores. We also keep the control loss of the C-VRM the same as the SC-VRM. We assume the same 100 pF C_{core} for both the SC-VRM system and the C-VRM. The average activity factor α is assumed to be 0.3. The switching frequency f_{sw} is swept to generate V_{dd} in the range of 0.42 V to 0.6 V. Energy losses and core energy are calculated via the energy model developed in previous sections.

To compare energy efficiency of the SC-VRM and C-VRM, we define the effective V_{dd} ($V_{dd,eff}$) as the V_{dd} under which the MAC core in the SC-VRM will give the same throughput as the MAC core in the C-VRM, i.e., the SC-VRM clock period $T_{clk-S}(V_{dd,eff})$ equals the average C-VRM clock period:

$$T_{clk-S}\left(V_{dd,eff}\right) = \frac{1}{M}\sum_{i=1}^{M} T_{clk-C}\left(i\right) \tag{2.19}$$

where $T_{clk-C}(i)$ is the C-VRM clock period in the i^{th} cycle. Substituting (2.15), (2.16) into

(2.19) for $N = 2$, we obtain:

$$T_{clk-S}\left(V_{dd,eff}\right) = \frac{ln[\frac{C_{sc}}{C_{core}+C_{sc}}]}{ln[\frac{V_{dd}(M)}{V_{dd}(0)}]} \sum_{i=1}^{M} \frac{T_d(V_{dd}\left(i\right)) + T_d\left(V_{dd}\left(i-1\right)\right)}{2} \tag{2.20}$$

Substituting (2.14) into (2.20), we obtain the relation between $V_{dd,eff}$ and $V_{dd}(0)$ and $V_{dd}(M)$ as follows:

$$T_{clk-S}\left(V_{dd,eff}\right) = \frac{ln[\frac{C_{sc}}{C_{core}+C_{sc}}]}{ln[\frac{V_{dd}(M)}{V_{dd}(0)}]} \sum_{i=1}^{M} \frac{T_d(\left[\frac{C_{sc}}{\alpha C_{core}+C_{sc}}\right]^{i} V_{dd}(0)) + T_d\left(\left[\frac{C_{sc}}{\alpha C_{core}+C_{sc}}\right]^{i-1} V_{dd}(0)\right)}{2} \tag{2.21}$$

Under the assumption that CH and CL see the same voltage profile $V_{dd}(m)$ during their active periods, we can substitute $\frac{V_{bat}}{2} + \Delta V$ and $\frac{V_{bat}}{2} - \Delta V$ into (2.21) to obtain:

$$\begin{aligned} T_{clk-S}\left(V_{dd,eff}\right) \;=\; & \frac{ln[\frac{C_{sc}}{C_{core}+C_{sc}}]}{ln[\frac{\frac{V_{bat}}{2}-\Delta V}{\frac{V_{bat}}{2}+\Delta V}]} \times \\ & \sum_{i=1}^{M} \frac{T_d(\left[\frac{C_{sc}}{\alpha C_{core}+C_{sc}}\right]^{i} (\frac{V_{bat}}{2} + \Delta V)) + T_d\left(\left[\frac{C_{sc}}{\alpha C_{core}+C_{sc}}\right]^{i-1} (\frac{V_{bat}}{2} + \Delta V)\right)}{2} \end{aligned}$$

For near threshold operation, it is difficult to obtain a precise analytical expression for this delay. Simulation results were used to extract the delay vs. V_{dd} curve of the critical path and $V_{dd,eff}$ can be solved numerically for different values of ΔV. Figure 2.7 shows that while the actual voltage range might go beyond $V_{bat}/2$ (0.6 V in the simulation), the effective voltage is less than the ideal output voltage $V_{bat}/2$.

In the first experiment, we assume C_{reg-C} is only 1% of C_{core} so that the data transfer overhead is small. Figure 2.8(a) shows the different energy components for the conventional SC-VRM as a function of V_{dd}. We denote $E_{SHUNT} = E_{GDL} + E_{BPCL} + E_{CL}$ since driver loss, bottom capacitance loss and control loss can all be denoted as parallel equivalent resistors in Fig. 2.2(c). From Fig. 2.8(a), we can see that as V_{dd} increases, E_{CTL} increases due to reduced ΔV according to (2.3); but E_{SHUNT} increases due to increased f_{sw}. In the super-

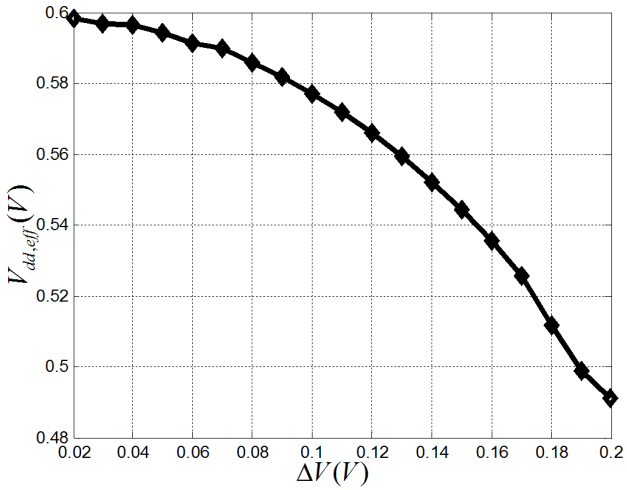


Figure 2.7: The effective voltage $V_{dd,eff}$ as a function of $\Delta V = V_{bat}/2 - V_{dd}(M)$.

threshold region, as V_{dd} decreases, E_{core} decreases because dynamic energy dominates. As V_{dd} further decreases to sub/near-threshold region, E_{core} increases due to the exponential increase of propagation delay. Due to the trade-off between E_{SHUNT}, E_{CTL} and E_{core}, the system MEOP (S-MEOP) voltage $V^*_{dd,S-MEOP}$ is around 0.46 V. The E_{op} increases as the V_{dd} deviates from $V^*_{dd,S-MEOP}$.

Figure 2.8(b) shows the different energy components of the C-VRM as a function of V_{dd}, where E_{DTL} and E_{CL-C} are lumped together as E_{LOSS} for simplicity. The V_{dd} value for the C-VRM is the $V_{dd,eff}$ defined in (2.19). Figure 2.8(b) illustrates that compared with the conventional SC-VRM system, the C-VRM has a higher E_{core} due to its variable voltage operation. However, the C-VRM eliminates E_{GDL}, E_{BPCL} and E_{CTL} associated with the SC-VRM system. Furthermore, Fig. 2.8(b) indicates that E_{LOSS} becomes higher when V_{dd} is close to the ideal output ($\frac{1}{2}V_{bat} = 0.6$ V) due to the increased data transfer frequency. E_{LOSS} also increases as the core enters sub-threshold region due to increased delay.

Figure 2.8(c) compares the E_{op-SC} and E_{op-C}, as defined in (2.17) and (2.18), and shows that the C-VRM has lower E_{op} compared with the SC-VRM system across the entire operating point from 0.42 V to 0.6 V. Large energy savings can be achieved either at high V_{dd} (close to ideal output of 0.6 V) due to the elimination of E_{GDL} and E_{BPCL}, or when V_{dd} is further reduced beyond $V^*_{dd,S-MEOP}$ of 0.46 V due to the elimination of E_{CTL}.

Figure 2.8(d) shows the efficiency comparison of the SC-VRM system and the C-VRM.

This figure illustrates that the C-VRM can maintain high efficiency ($\eta_{C-VRM} > 93\%$) across the operating range from 0.42 V to 0.6 V, while the SC-VRM can only achieve $\eta_{SC-VRM} \approx 80\%$ at around 0.54 V. As V_{dd} deviates from this efficiency maximum voltage, η_{SC-VRM} drops quickly due to increased E_{SHUNT} or E_{CTL}.

The energy benefit of the C-VRM depends on the assumption that the data transfer loss E_{DTL} is small. This assumption holds if the core swapping frequency f_{swap} is small compared to f_{clk-C}, and C_{reg-C} is small. To illustrate this point, we perform the same set of experiments as in Fig. 2.8 but with C_{reg-C} increased to 10% of C_{core}. Figure 2.9 shows the resulting E_{op} and η. Figure 2.9(a) shows that when E_{DTL} is large, it is possible that the $E_{op-C} > E_{op-SC}$. However, energy savings are preserved when V_{dd} is close to the ideal output of $\frac{1}{2}V_{bat} = 0.6$ V or when V_{dd} is in far below $V^*_{dd,S-MEOP}$ in sub-threshold. Figure 2.9(b) shows that when E_{DTL} is large, η_{C-VRM} decreases dramatically when V_{dd} increases due to the increased core swapping frequency. Therefore, to achieve maximum energy savings, E_{DTL} of the C-VRM needs to be kept to a minimum.

2.3 C-VRM Prototype IC Design

A prototype IC was designed in a 1.2 V, 130 nm CMOS process to compare the SC-VRM system and the C-VRM. This section describes the prototype IC.

2.3.1 Chip Architecture

To enable a direct comparison between the SC-VRM system and the C-VRM, we fix the charge transfer capacitor (C_{sc} in Fig. 2.2(a) and $C_1 + C_2$ Fig. 2.4) to 250 pF and employ an 8-bit multiply-accumulator (MAC) as the core in both systems. The SC-VRM system and the C-VRM are optimized to supply an I_{core} up to 1 mA at a nominal V_{dd} of 500 mV . To reduce E_{DTL}, a folded MAC architecture is adopted.

Figure 2.10 shows the top level chip architecture. The chip consists of a 2:1 SC-VRM system, a 2:1 C-VRM and a test block. The 2:1 SC-VRM system consists of a ladder SC-VRM delivering energy to a core. The core is a MAC with an 8 bit array multiplier and a

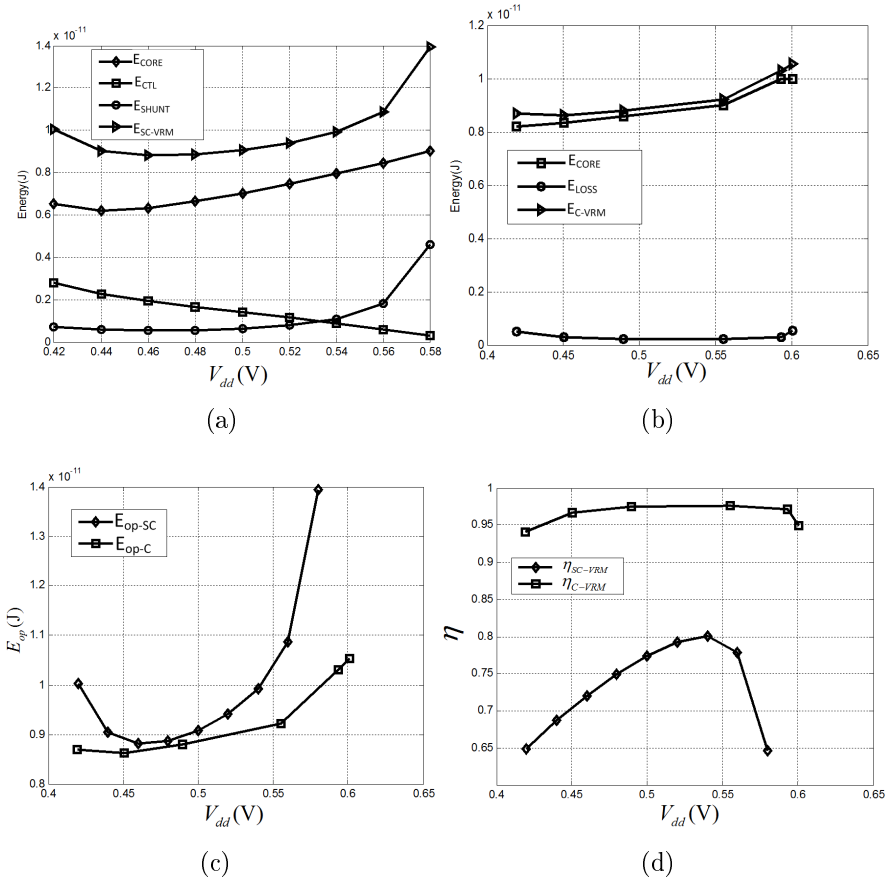


Figure 2.8: Comparison of SC-VRM system and C-VRM with 1% data transfer overhead: (a) energy vs. output V_{dd} of SC-VRM, (b) energy vs. V_{dd} of C-VRM, where E_{DTL} and E_{CL-C} were lumped together as E_{LOSS}, (c) E_{op} comparison of SC-VRM and C-VRM, and (d) efficiency comparison of SC-VRM and C-VRM.

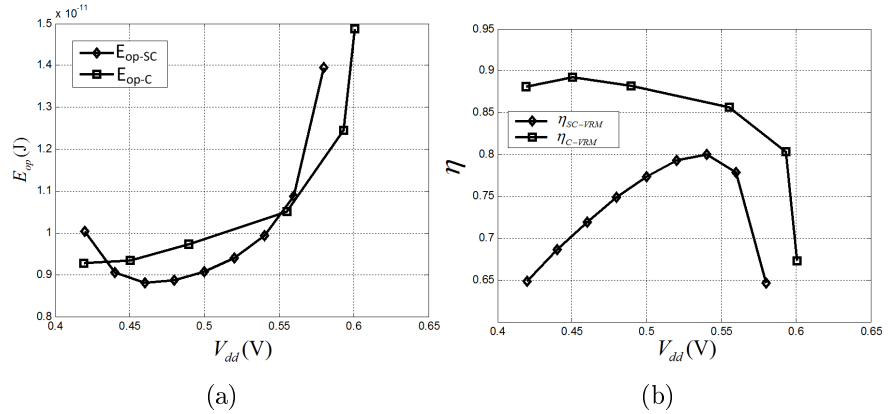


Figure 2.9: System comparison between SC-VRM system and C-VRM with 10% data transfer overhead: (a) E_{op} comparison of SC-VRM and C-VRM, and (b) efficiency comparison of SC-VRM and C-VRM.

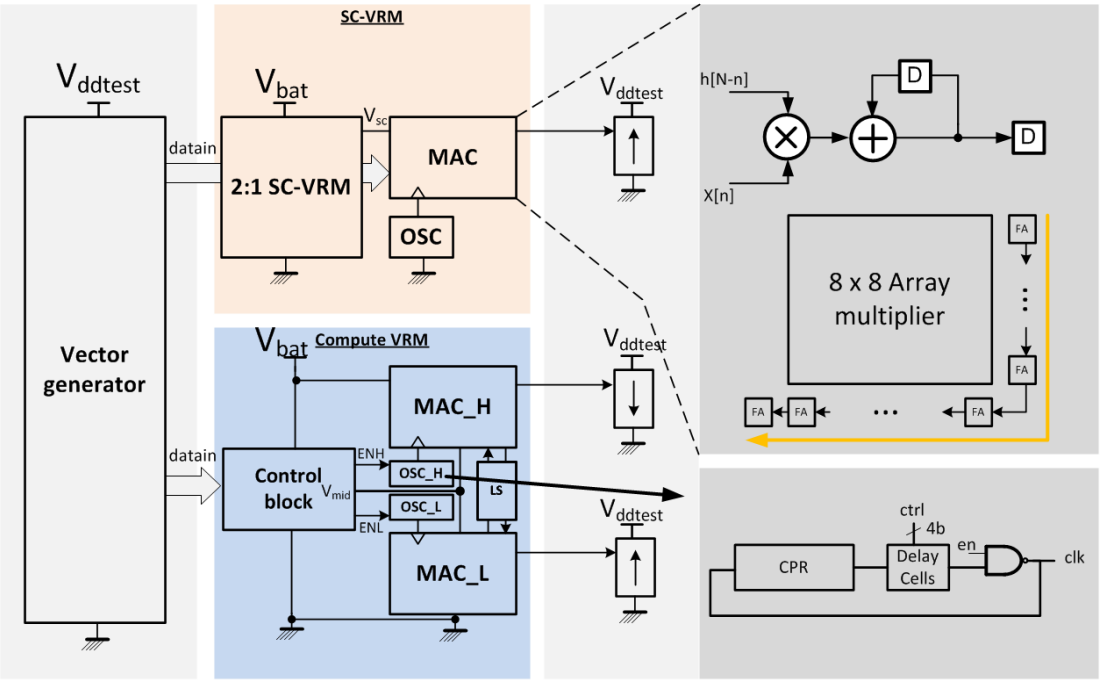
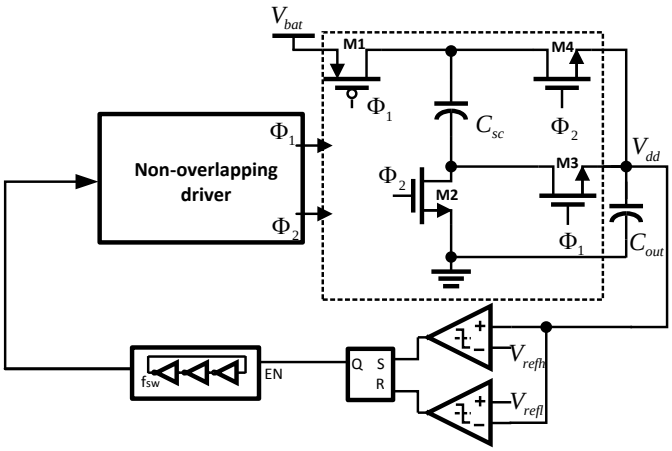


Figure 2.10: The C-VRM prototype IC architecture.

ripple carry adder. It is configured as an 8-tap folded FIR filter. The 2:1 C-VRM consists of cores MAC_H and MAC_L, which are identical to the core in the 2:1 SC-VRM system. A CPR oscillator with tunable delay is designed to continuously scale the core frequency f_{clk-C} with V_{dd}. The test block consists of a vector generator to feed input data to the cores and level shifters to transfer output data for off-chip processing.

2.3.2 SC-VRM Design

Figure 2.11 shows the detailed architecture of the 2:1 SC-VRM, which has a ladder topology containing four power switches and one 250 pF on-chip MIM flying capacitor C_{sc}. C_{sc} is chosen to supply maximum I_{core} of 1 mA with maximum f_{sw} of 10 MHz. The transistor M1 and M2 are chosen to be PMOS and NMOS, respectively, to remove the threshold voltage drop. M3 and M4 are chosen to be NMOS because the regulated output V_{dd} is always lower than $\frac{1}{2}V_{bat}$. The power switches are sized to balance shunt and series loss according to [15, 75]. Figure 2.11 also shows the control loop of the SC-VRM. A hysteresis PFM control [76] is realized via a strong ARM comparator and a current starved oscillator. The output



C_{sc}	250 pF
M1	100 µ/0.12 µ
M2	50 µ/0.12 µ
M3	100 µ/0.12 µ
M4	100 µ/0.12 µ

Figure 2.11: The 2:1 ladder SC-VRM.

of the oscillator is passed through the driver circuitry and converted to non-overlapping two-phase clock signals.

Figure 2.12 shows the circuit diagram of the strong ARM comparator and the current starved oscillator. We adopt the dynamic comparator to avoid steady state current, which degrades light load efficiency. In the pre-charge phase, MP1-MP4 pre-charges the output node and the drains of MN3 and MN4 to V_{dd}. In the evaluation phase, the drain of MN3 and MN4 are discharged at different rates according to input V_{ip} and V_{in}, respectively. If V_{ip} is higher than V_{in}, MN1 will turn on prior to MN2, and the positive feedback formed by MN1, MN2, MP5 and MP6 will discharge V_{on} and charge V_{op} back to V_{dd}. The pre-charge transistors are kept to a minimum size to reduce the load capacitance. The input pair and cross coupled inverter MN1, MN2, MP5 and MP6 are sized to trade off speed and offset. The current starved oscillator contains mirror transistors MN1-MN3 and MP1-MP3. Transistors MN4-MN6 and MP4-MP6 form a 3 stage ring oscillator. They are sized to minimize the power consumption while providing sufficient driver speed.

Figure 2.13 shows the non-overlapping circuit with an embedded driver. The complementary clock signal is fed into a cross coupled NOR gate to add dead time, t_p, between $\Phi 1$ and $\Phi 2$. The t_p is adjusted by changing the number of the buffer chain stages. A superbuffer is added at the end of the buffer chain to drive the power switches.

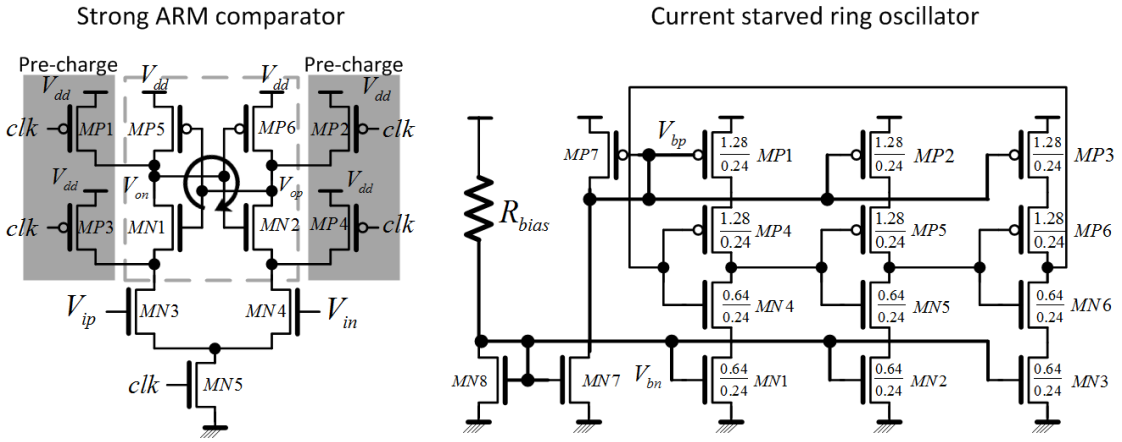


Figure 2.12: The strong ARM comparator and the current starved oscillator employed in the 2:1 SC-VRM.

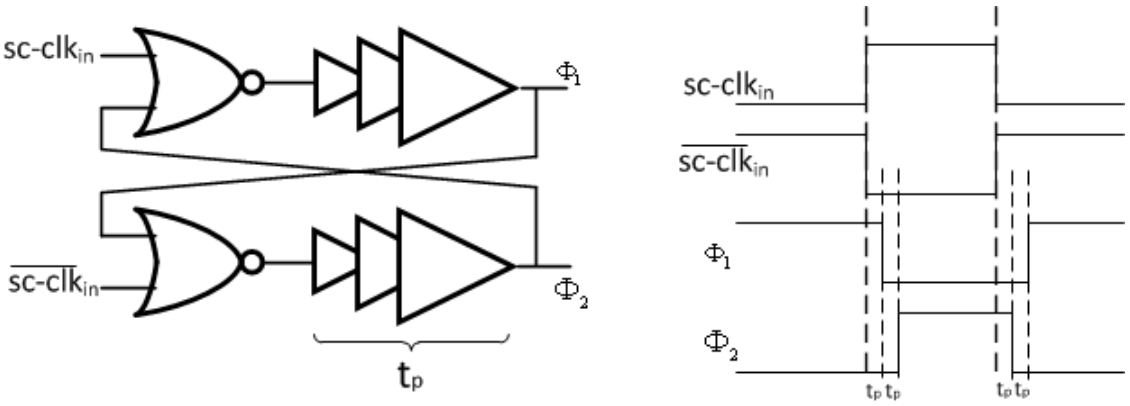


Figure 2.13: Non-overlapping driver employed in the 2:1 SC-VRM.

2.3.3 C-VRM Design

Figure 2.14 shows the design of the C-VRM. The 2:1 C-VRM contains MAC_H and MAC_L as the two compute cores, level shifters (LS) for data transfer, and a control block to switch the compute cores from active to inactive modes.

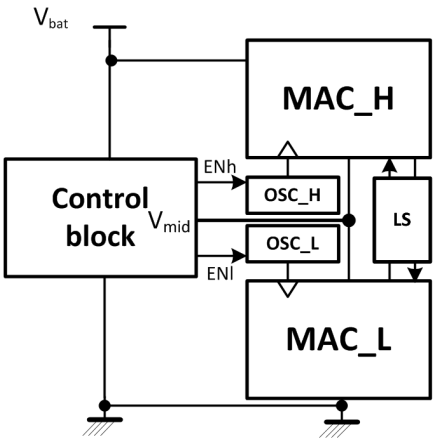


Figure 2.14: The 2:1 C-VRM block diagram.

Figure 2.15(a) shows the circuit diagram of the control block. The shaded blocks operate in the low voltage domain, while the unshaded blocks operate in the high voltage domain. The control block consists of an RC delay based frequency detector, a latch and a pulse generator for core swapping. Figure 2.15(b) shows the operation of the frequency detection block. During the pre-charge phase, C_2 is connected to V_{mid}; during the evaluation phase, C_2 is discharged through the RC circuit formed by R_2 and C_2, with discharging time determined by $\frac{1}{2f_{clk}}$. If during this period, V_a drops below the threshold, V_b will rise to V_{mid} and state of the latch (**ENl**) will be set to 0, disabling MAC_L. The pulse generator is realized via NOR **ENl** and a delayed version of the signal, so that during the 1-0 transition of **ENl**, a pulse is generated. The pulse will force the state of latch in high voltage domain (**ENh**) to be 1 through the pulldown transistor M1, thus enabling MAC_H by turning **ENh** to 1.

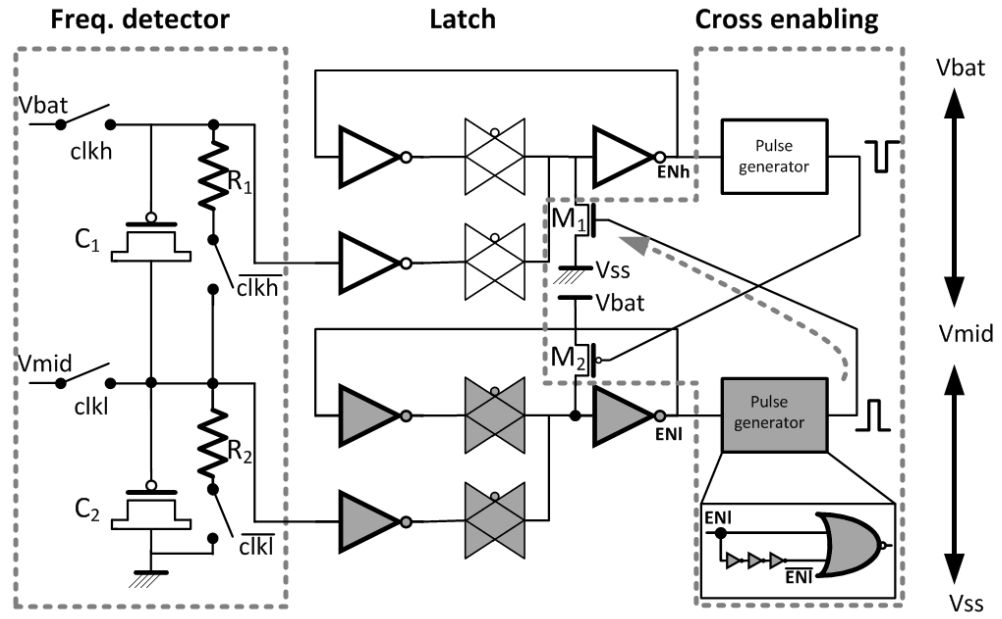
Figure 2.16 shows the level shifter used in the C-VRM. The level shifter will perform bidirectional transfer of the data between MAC_H and MAC_L. The conventional level shifter design shown in Fig. 2.16(a) is not suitable for two reasons: (1) two different circuit topologies are needed to perform high-to-low and low-to-high level conversion, respectively,

and (2) there are direct paths current through MN1, MP1 or MN2, MP2 during shift. Both of these will result in additional E_{DTL} in (2.18). Therefore, we adopt a capacitor coupling based dynamic level shifter in Fig. 2.16(b). Figure 2.16(b) also shows the operation during high-to-low data transfer. In the pre-charge phase, the capacitor C_{ls} is pre-charged to V_{mid}. When MAC_H is disabled by changing **ENh** from 1 to 0, a one clock cycle pulse **shift_hl** is inserted before **ENl** goes high. This will turn on the transmission gate and shift $data\_h$ from MAC_H to MAC_L. After the shift operation, C_{ls} is charged to V_{mid} before the next operation. The dynamic level shifter achieves bidirectional shifting and removes the direct path loss associated with the conventional design.

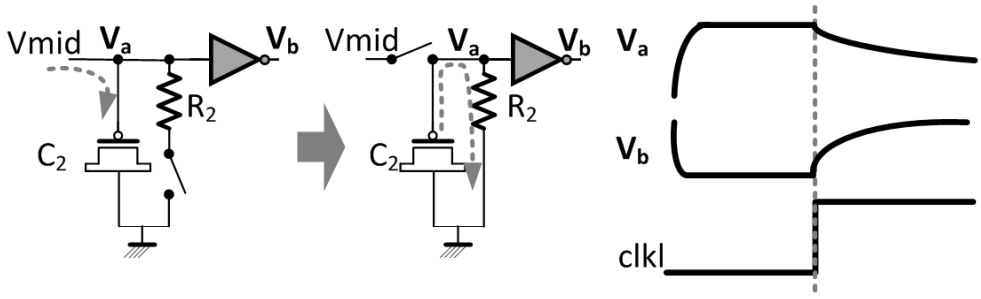
2.3.4 CPR Oscillator

The CPR oscillator we employed in this chapter is similar to the one used in [62], which is an inverter chain based ring oscillator with tunable delay cells (See Fig. 2.17(a)). A chain of inverters are used instead of a direct mapping of the core critical path components to provide a near-50% duty cycle clock. The number of inverters is calculated based on a first-order approximation using the Elmore delay formula for a resistor-capacitor network. In this design, 68 inverters are used to replicate the critical path. To account for PVT variation, a tuning circuit is added to adjust the delay margin provided by the CPR oscillator to ensure the clock period is greater than the actual critical path. A digital control is chosen over voltage control for simplicity and reliable bias in different voltage domains. Figure 2.17(b) shows the detail of the inverter delay cell. Each delay cell consists of a long path and a shortpath, the selection of the paths is controlled by MUX. Each delay cell uses a single-bit control to minimize the capacitive loading at the output of the delay cell. In the design, 4 delay cells are added to provide tuning range of 16 inverter delays.

Figure 2.18(a) shows the postlayout simulation of MAC unit critical path delay and the CPR oscillator clock period in different process corners. It can be seen that the CPR oscillator is able to track the MAC unit critical path and ensure that the clock period is larger than the MAC unit critical path delay across all process corners. Figure 2.18(b) shows the CPR oscillator clock period with different tuning settings. The 8 delay cells

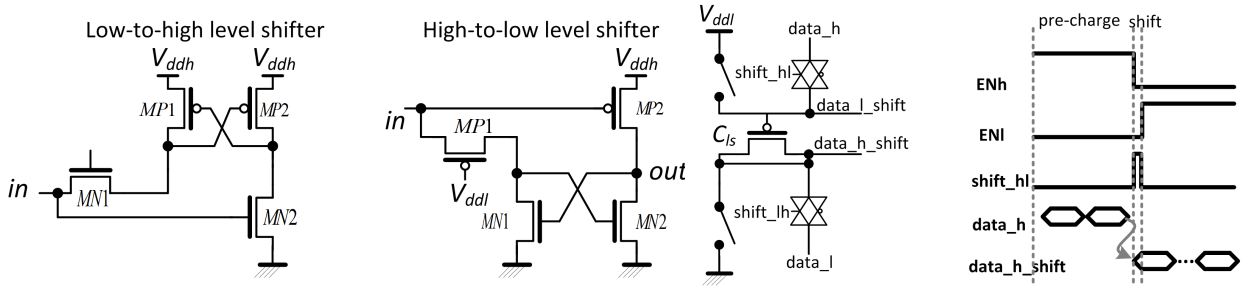


(a)



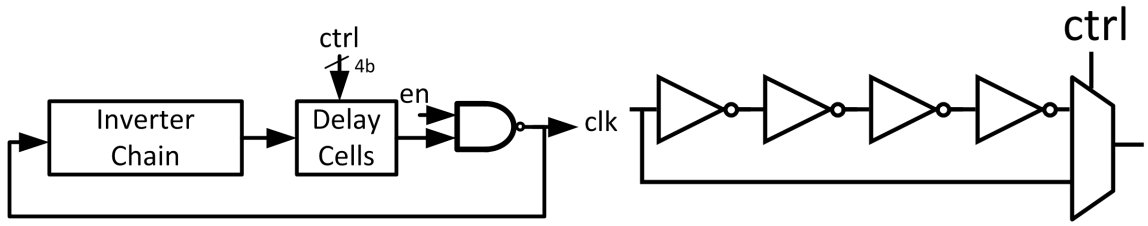
(b)

Figure 2.15: Control block of the C-VRM: (a) circuit schematic, and (b) principle of operation.



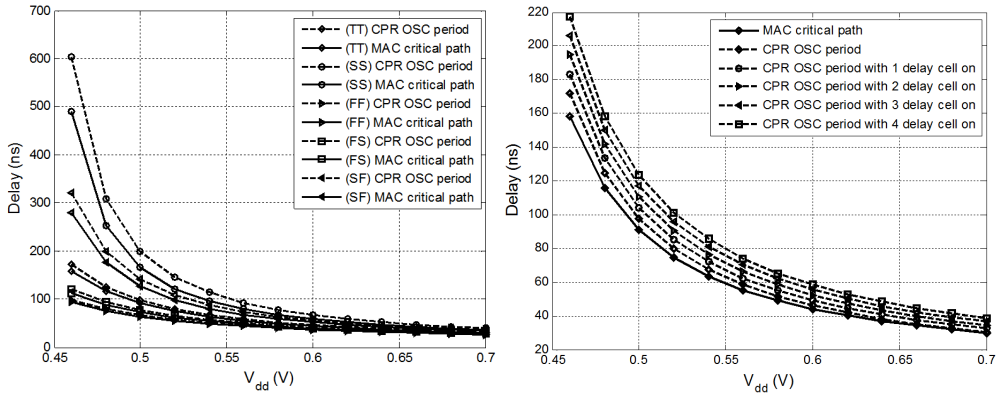
(a) (b)

Figure 2.16: Bidirectional level shifter: (a) conventional design of low-to-high and high-to-low level shifters, and (b) the proposed capacitor coupling based dynamic level shifter.



(a) (b)

Figure 2.17: Architecture of: (a) the CPR oscillator, and (b) the tunable delay cell.



(a) (b)

Figure 2.18: Post layout simulations showing: (a) the CPR oscillator clock period and MAC critical path delay, and (b) the CPR oscillator clock period with different delay cells selected.

provide sufficient large tuning range to account for the delay variations.

2.4 Test Results

Figure 2.19 shows the operation of the SC-VRM during startup and steady state. During steady state, PFM control is employed to scale the switching frequency f_{sw} with I_{core} to reduce the driver and bottom plate capacitance loss.

Figure 2.20 shows the operation of the C-VRM including core swapping and data transfer. The startup circuitry for the C-VRM is implemented on the board. When the MAC core voltage in one voltage domain decreases to 500 mV, core swapping is performed by employing

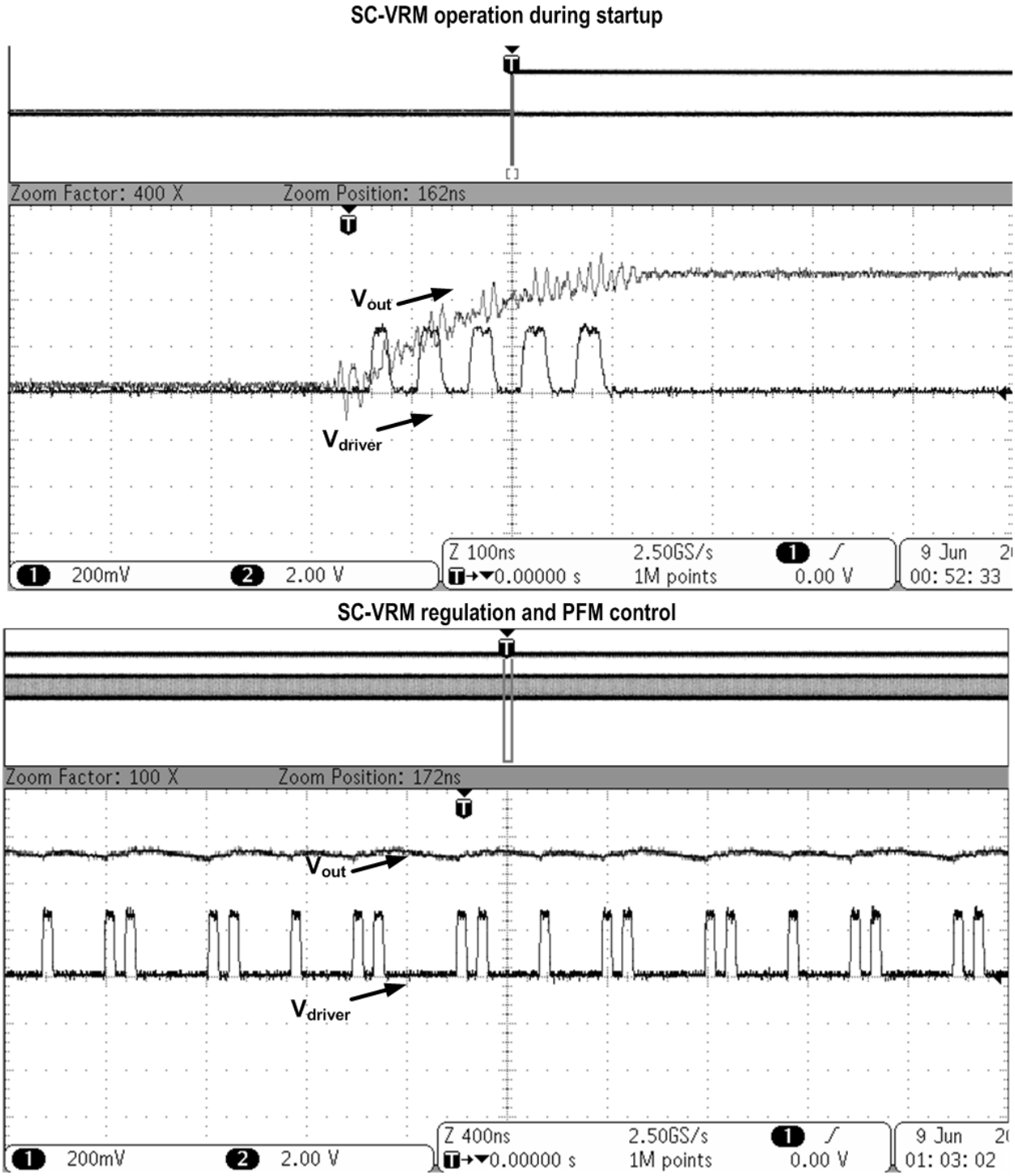


Figure 2.19: Measured SC-VRM operation during start up and steady state.

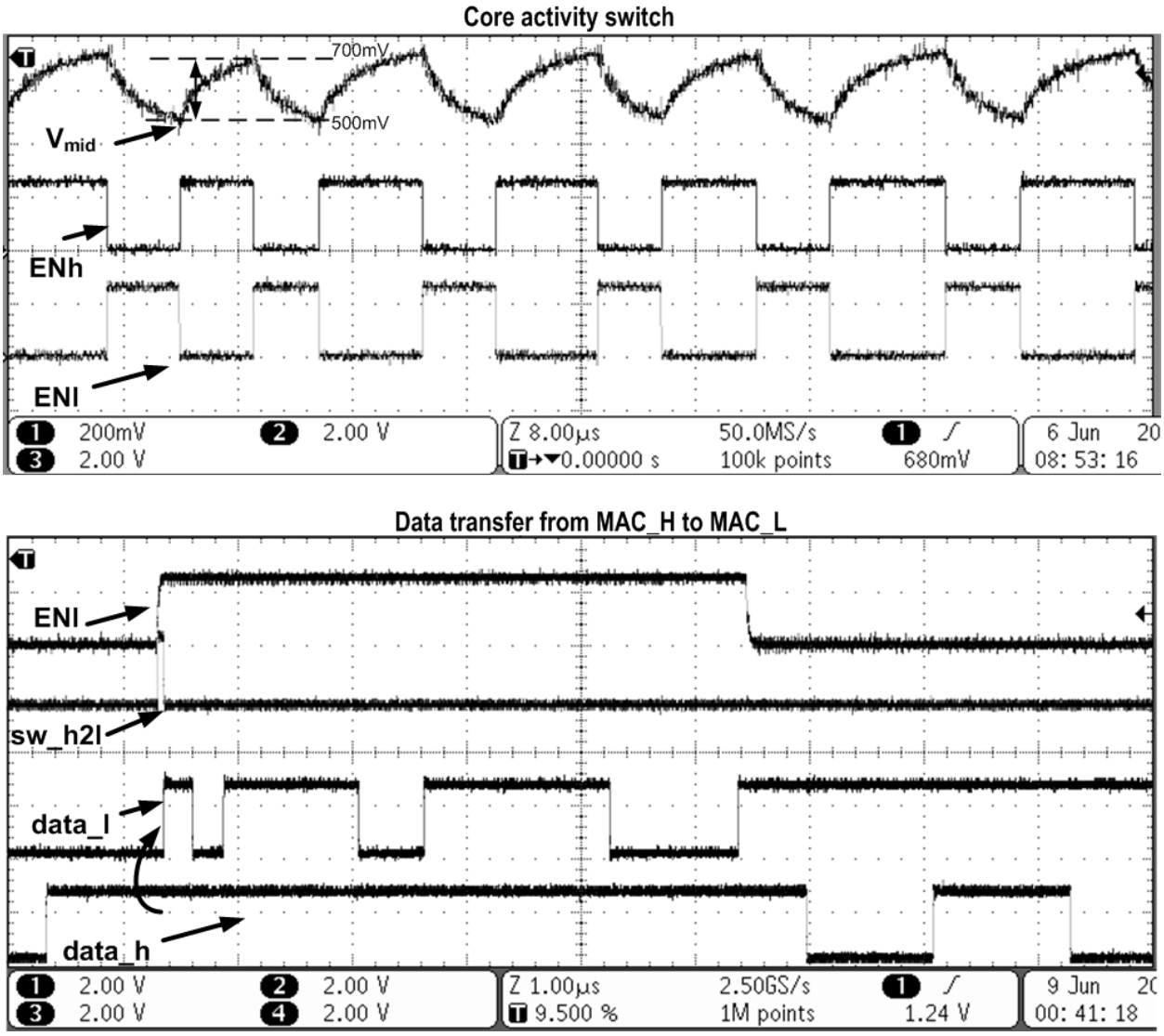


Figure 2.20: Measured C-VRM core swapping and data transfer.

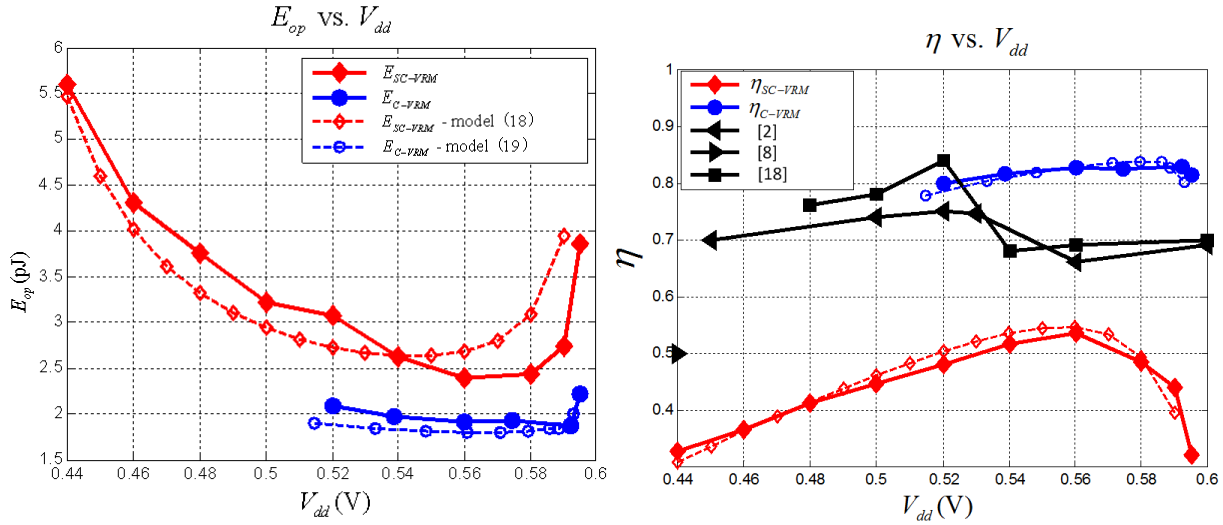
the on-chip generated enable signal **ENh** and **ENl** for MAC_H and MAC_L, respectively. During data transfer, when MAC_L is enabled by **ENl,** a one cycle switching signal **sw_h2l** is generated to shift data from the high voltage to the low voltage domain.

Figure 2.21 compares the measured E_{op} and the efficiency of the fabricated 2:1 SC-VRM system and the 2:1 C-VRM. The simulated results according to the energy model in Section 2.2 are also shown as dashed lines to demonstrate the accuracy of the model in Section 2.2. The efficiency vs. V_{dd} is obtained by changing the reference voltage for the SC-VRM, and by changing the RC time constant of the frequency detector in the C-VRM controller. To

perform efficiency measurements for the C-VRM, the compute core in the C-VRM, and the controller and level shifter of the C-VRM are provided separate supply pins, which allows the loss (control loss and data transfer loss) current I_{loss} and the core current I_o to be measured. The efficiency is then calculated using $\eta = I_o/(I_o + I_{loss})$. As shown in Fig. 2.21(a), the C-VRM has lower E_{op} across all measured V_{dd} from 0.52 V to 0.59 V. Since C-VRM has a continuously varying voltage, the V_{dd} is the effective output voltage delivering the same throughput (11 MHz-to-20 MHz) as the SC-VRM system. V_{dd} of C-VRM cannot extend below 0.52 V due to the limitations of the capacitively coupled level shifter. The SC-VRM system has high system energy overhead both in high V_{dd}, due to increased driver loss E_{GDL}, and in low V_{dd}, due to control loss E_{CL} and low f_{clk-C}. This is also indicated by the system level simulations in Fig. 2.8(a). The absence of the driver circuits and the use of a low power frequency detection scheme enables the C-VRM to achieve a maximum of 44.8% energy savings compared to the SC-VRM system. Figure 2.21(b) shows that the C-VRM achieves efficiency $> 79\%$ across the entire tested V_{dd} range. As a comparison, the SC-VRM has a peak efficiency of only 54% due to E_{SHUNT} and E_{CTL}. Figure 2.21(b) also plots the efficiency of previously published SC-VRM designs operating at power levels of 1μW-100s of μW range from [67, 13, 77].

Table 2.1 shows the design specifications and performance of C-VRM compared with previously published works. The system energy per instruction/K-gate is calculated by dividing the system EPI by the estimated number of gates for [66] and [67]. For [12] and [13], since no compute cores are included on-chip, the system energy per instruction/K-gate is estimated using the core EPI and gate count of our design and the efficiency reported in [12] and [13]. The comparison in Fig. 2.21(b) and in Table 2.1 shows that the C-VRM achieves the highest efficiency (83%) in the designed power level and the lowest EPI/k-gate of 0.79 pJ.

Figure 2.22 shows the die photo of the test chip, which is fabricated in a 1.2 V, 130 nm CMOS process and has an area of 2 mm $\times$ 2 mm. Note that the SC-VRM requires 2 nF off-chip decoupling capacitor. This capacitor, if integrated on-chip, will present 0.49 mm^2 additional area.



(a) (b)

Figure 2.21: The C-VRM test chip measurement results: (a) E_{op} comparison, and (b) efficiency comparison.

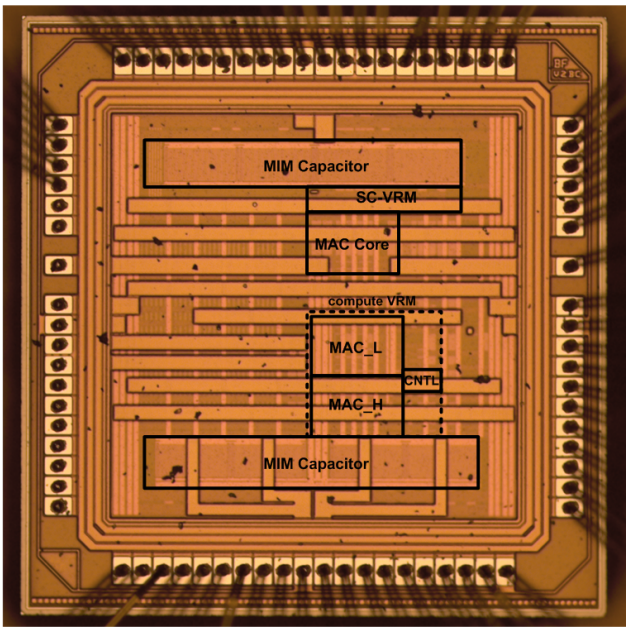


Figure 2.22: Die photo of the test chip.

Table 2.1: Comparison with Previously Published Work

	[67]	[12]	[13]	[66]	C-VRM
Technology	65 nm	45 nm	130 nm	180 nm	130 nm
Conversion Ratio	1/3, 1/2, 2/3, 3/4, 1	2/3	1/5	1/6	1/2
VRM Topology	SC-VRM	SC-VRM	SC-VRM+LDO	SC-VRM+LDO	Compute VRM
Input Voltage	1.2 V	1.8 V	3.6 V	3.6 V	1.2 V
Output power /current level	$1-500\,\mu$W	100 μA-9 mA	2.5 nW-254 nW	550 pW-7.7 μW	$1-60\,\mu$A
C_{sc}	600 pF	534 pF	800 pF	NA	250 pF
C_{out}	NA	700 pF	NA	NA	0
Maximum driver switching freq.	15 MHz	30 MHz	2 KHz	1.2 MHz	No driver
Efficiency	75% @V_{dd}=0.5 V I_o= 100 μA	55% @V_{dd} = 0.9 V I_o= 200 μA	56% @V_{dd} = 0.44 V I_o= 300 nA	41.6% @V_{dd}= 0.4 V	83% @V_{dd} = 0.5 V-0.7 V I_o= 40 μA
E_{op}/k-gate	1.09 pJ	1.19 pJ[1]	1.17 pJ[1]	0.88 pJ	0.79 pJ

[1]System energy per instruction is calculated based on core power of the filter core in this work and the reported efficiency

2.5 Conclusions

In this chapter, we propose the C-VRM, a unified architecture for energy delivery and computation, to overcome the intrinsic loss and drive circuit overhead of the conventional SC-VRM. The C-VRM employs multiple voltage domain stacking, core swapping, and CVFS to achieve high energy efficiency in the sub/near-threshold region. This work shows that by combining the compute core and the energy delivery block, the system energy efficiency can be significantly improved. It opens the possibilities of embedding more sophisticated computational blocks into SC-VRM and other switching power delivery blocks. Further study can be explored to develop C-VRM architectures based on multi-ratio SC-VRM or multi-phase SC-VRM.

Chapter 3

COMPUTE SENSOR

In the previous chapter, the energy delivery efficiency is improved by embedding information processing into the VRM. In this chapter, we explore a similar approach which embeds information processing into the sensing circuits to drastically alleviate the communication challenge between the sensing and information processing subsystems. Specifically, we present an in-sensor computing architecture which (mostly) eliminates the sensor-processor interface and thus resolves the communication challenge by embedding inference computations in the noisy sensor fabric in analog, and retraining the hyperparameters in order to compensate for non-ideal computations. The resulting architecture, referred to as the Compute Sensor - a sensor that computes in addition to sensing - represents a radical departure from the conventional architecture. We show that a Compute Sensor for image data can be designed by embedding both feature extraction and classification functions in the analog domain in close proximity to the CMOS active pixel sensor (APS) array. Significant gains in energy-efficiency are demonstrated using behavioral and energy models in a commercial semiconductor process technology. In the process, the Compute Sensor creates a unique opportunity to develop machine learning algorithms for information extraction from data on a noisy underlying computational fabric.

Figure 3.1(a) shows a conventional architecture of an embedded vision system. Image data is first acquired via an M_r row $\times$ M_c column active pixel sensor (APS) array whose analog pixel values are sensed sequentially in a row-wise fashion, and then converted into digital samples by the sample-and-hold (S/H) and the analog-to-digital converter (ADC), and then streamed out by the read-out (RD) circuitry to a back-end digital processor which implements feature extraction and classification function to obtain the final decision $\hat{y}$. A digital trainer block computes the hyperparameters in supervised learning mode. This phys-

ical separation between sensing and processing subsystems is unavoidable because sensing is intrinsically an analog process while information processing is intrinsically digital. Excluding the energy delivery loss, the energy dissipation in such a system is dominated by two sources:

- The energy required to move the data over the sensor-processor interface comprising the ADC, RD and the interconnect to the digital processor, i.e., *communication energy*, and
- The energy consumed in processing the data using digital circuits which by nature are high signal-to-noise ratio (SNR), i.e., *computational energy*.

We employed energy data from [6] for a CMOS image sensor consisting of a 32×32 APS array and the associated interface circuits, and estimated the computational energy needed to implement a principal component analysis (PCA) [78] engine and a support vector machine (SVM) [79] in a 65 nm CMOS process operating at a throughput of 32 frames/s. This analysis indicates that the communication and computational energies are approximately 53% and 41%, respectively, for a combined total of 94%. An impactful solution to the energy problem needs to reduce both components of energy - communication and computational energy.

In this chapter, we propose the Compute Sensor shown in Figure 3.1(b) - a sensory system that senses *and* processes the sensed data thereby integrating both data acquisition and information extraction functionalities. The Compute Sensor architecture consists of a data processing engine and a training engine. The data processing engine is a cascade of: (1) the APS array which is identical to the conventional architecture, (2) a bit-line processor (BLP) whose physical dimensions are matched to that of the APS array in order to perform pixel-wise operation such as sample-and-hold (S/H), scaling, and absolute difference but no ADC, (3) a cross bit-line processor (CBP) to perform data dimensionality reduction operations such as dot product, filtering, sum-of-absolute difference (SAD), mean square, followed by an ADC that operates on the reduced dimensionality data and feeds it into (4) the residual digital processor (RDP) which implements very simple digital computations needed to obtain the final decision $\hat{y}$. Unlike the conventional architecture, both the BLP and CBP in the Compute Sensor operate in the analog domain. The trainer is digital.

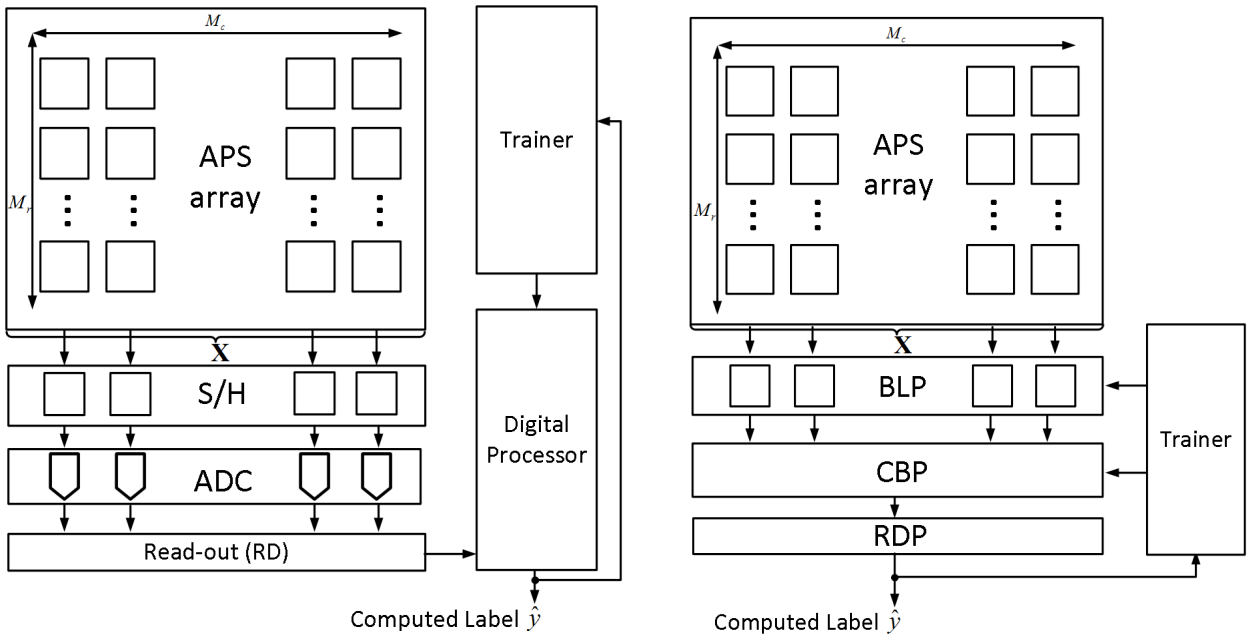


Figure 3.1: A typical embedded vision platform: (a) conventional architecture, and (b) the proposed Compute Sensor architecture.

The Compute Sensor eliminates both the traditional sensor-processor interface, and the high-SNR/high-energy digital processing by moving feature extraction and classification functions into the analog domain in close proximity to the APS array. The Compute Sensor leverages the intrinsic ability of machine learning algorithms to extract information from noisy and often incomplete data to provide robust inference in presence of non-ideal computations. We demonstrate a Compute Sensor that incorporates a PCA-based feature extractor and a support vector machine (SVM). Using circuit characterized behavioral and energy models in a 65 nm CMOS process, we show that the Compute Sensor is able to achieve a detection accuracy greater than 94.7% using the Caltech101 dataset [80], which is within 0.5% of that achieved by an ideal digital implementation. Furthermore, the Compute Sensor is able to compensate for variations in the electrical parameters of the transistors in the APS array caused by finite tolerances of the semiconductor manufacturing process by retraining in presence of these non-idealities. As a result the Compute Sensor consumes 7× to 17× less energy than the conventional architecture for the same level of accuracy. Thus, this paper highlights the potential for conducting algorithmic research that accounts for platform resource-constraints such as energy, storage, and computation.

The rest of the chapter is organized as follows. Section 3.1 presents the necessary background and establishes notation. Section 3.2 presents the Compute Sensor architecture incorporating PCA and SVM algorithms, and the behavioral and energy models in a 65 nm CMOS processes. Simulation results are shown in Section 3.3, and discussions are provided in Section 3.4.

3.1 Background

3.1.1 Active Pixel Sensor

Solid state imaging devices can be classified into two categories, i.e., charge coupled device (CCD) sensor and CMOS sensor. The CMOS image sensor has gained much popularity due to its low voltage, low power operation, and compatibility with standard CMOS technologies [81]. APS is by far the most widely employed CMOS image sensor architecture due to the speed advantage and low noise. The architecture of the 3-transistor (3T) APS and associated read-out (RD) circuit is shown in Fig.3.2(a) where each APS consists of a photodiode (PD) as the optical detector and three transistors for readout. A rolling shutter operation where the pixel array is exposed row by row is typically employed for the 3T-APS array, and the timing diagram is shown in Fig. 3.2(b). During the reset phase, M_{RST} is on and the charge integrated on the photodiode is removed. During the integration phase, M_{RST} is off and the photodiode converts light into current, discharging the parasitic capacitor C_{PD}. During the readout phase, M_{SEL} is on, and the signal voltage V_{SIG} is sampled to sampler S-SIG.

3.1.2 Principle Component Analysis (PCA)

PCA is a widely used method for dimensionality reduction. This reduction is accomplished by projecting the data vector $\mathbf{x}_n \in \mathcal{R}^M$ ($n = 1, ..., N$, is the sample index and N is the total number of samples in the dataset) onto a set of orthonormal principal components $\boldsymbol{\alpha}_k \in \mathcal{R}^M, k = [1, ..., K]$. These principal components are the top K variance maximizing eigenvectors of the sample covariance matrix $\sum_{n=1}^{N} \mathbf{x}_n \mathbf{x}_n^T$ [78]. Hence, the reduced dimension

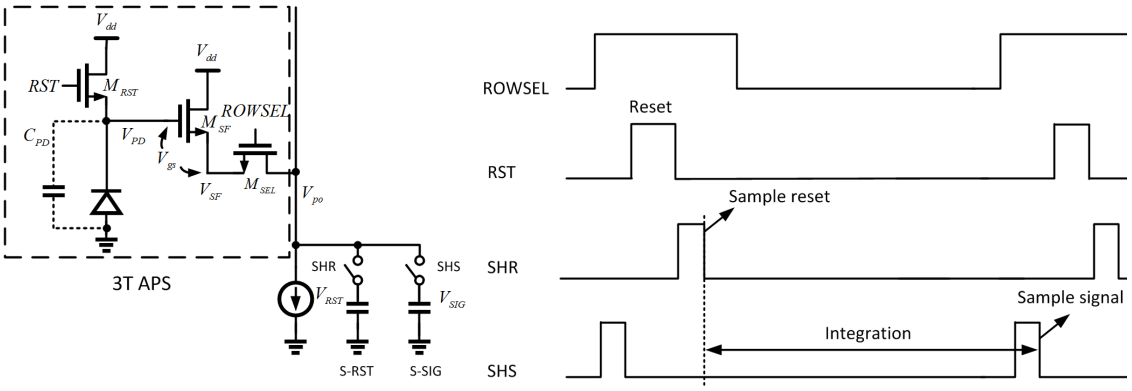


Figure 3.2: 3T APS: (a) pixel architecture and associated RD circuit, and (b) timing diagram in rolling shutter operation.

(feature) vectors $\mathbf{f} \in \mathcal{R}^K$ are obtained as:

$$\mathbf{f} \quad = \quad \mathbf{A}\mathbf{x} \tag{3.1}$$

where $\mathbf{A} = [\boldsymbol{\alpha}_1, ..., \boldsymbol{\alpha}_K]^T \in \mathcal{R}^{K \times M}$ is the eigenmatrix, and $\mathbf{x} \in \mathcal{R}^M$ is the *test* data vector. In the CMOS image sensor shown in Figure 3.1(a), $\mathbf{x}$ is obtained from the APS array and $M = M_r M_c$, where M_r and M_c are the number of rows and number of columns, respectively, in the APS array.

3.1.3 Support Vector Machine (SVM)

The SVM [79] is a popular supervised learning method for classification and regression. In SVM, the trained model is represented by:

$$y_o = \mathbf{w}_s^T \mathbf{f} - b \tag{3.2}$$

where $\mathbf{w}_s \in \mathcal{R}^K$ is the optimum weight vector, and $\mathbf{f} \in \mathcal{R}^K$ is the *test* feature vector. It can be shown that the optimum weight vector $\mathbf{w}_s$ can be described in terms of feature vectors that lie on the margins, i.e., support vectors:

$$\mathbf{w}_s = \sum_{n=1}^{N_s} \beta_n y_n \mathbf{f}_{s,n} \tag{3.3}$$

where y_n, N_s and $\mathbf{f}_{s,n}$ are the label, the number of support vectors, and the n^{th} support vector, respectively. The SVM's classification accuracy is denoted by $p_c = Pr\{\hat{y} = y\}$, where $\hat{y} = sgn(y_o)$ is the computed label and y is the true label.

3.2 The Compute Sensor

This section presents the proposed Compute Sensor architecture for implementing the PCA and SVM, along with architectural level functional and energy models in a 65 nm CMOS process. These models are employed to study the effectiveness of retraining on compensating for analog non-idealities, and for estimating the energy consumption.

3.2.1 Architecture

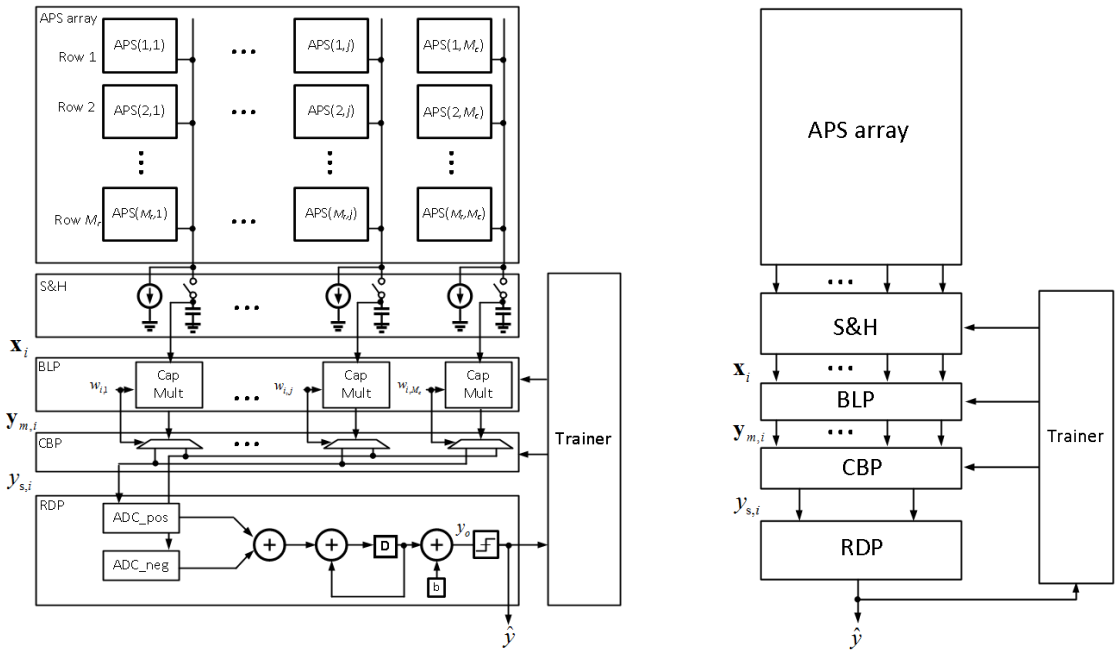


Figure 3.3: Compute Sensor implementing PCA and SVM: (a) the architecture, and (b) the behavioral model.

The general Compute Sensor architecture in Figure 3.1(b) enforces a specific sequence of functions - acquire data in the APS array of size $M = M_r M_c$, bit-line processing, cross

bit-line processing, followed by residual digital processing. Bit-line processing involves scalar operations while the cross bit-line processing results in dimensionality reduction. For example, bit-line operations could be the product of scalar data values and scalar weights, while cross bit-line processing would sum up these scalar products to generate a dot product. In the following, we remember that $M = M_c M_r$. Keeping these architectural constraints in mind, we exploit the linearity of the PCA and SVM computations in (3.1) and (6.7) to combine them as follows:

$$y_o = \mathbf{w}_s^T \mathbf{A}\mathbf{x} - b = \mathbf{w}^T \mathbf{x} - b \tag{3.4}$$

where $\mathbf{w}^T = \mathbf{w}_s^T \mathbf{A} = \left[\mathbf{w}_1^T, \ldots, \mathbf{w}_{M_r}^T\right] \in \mathcal{R}^{1\times M}$, $\mathbf{w}_i \in \mathcal{R}^{M_c}$, and $\mathbf{x} \in \mathcal{R}^M$, where $\mathbf{x}^T = \left[\mathbf{x}_1^T, \ldots, \mathbf{x}_{M_r}^T\right]$ and $\mathbf{x}_i \in \mathcal{R}^{M_c}$. The composite weight vector $\mathbf{w}$ can be obtained directly via SVM training methods. The data acquisition in the APS array occurs sequentially in a row-by-row fashion. In order to accommodate this constraint, we rewrite (3.4) as follows:

$$y_o = \left[\mathbf{w}_1^T, \ldots, \mathbf{w}_{M_r}^T\right] \begin{bmatrix} \mathbf{x}_1 \\ \vdots \\ \mathbf{x}_{M_r} \end{bmatrix} - b = \sum_{i=1}^{M_r} \mathbf{w}_i^T \mathbf{x}_i - b \tag{3.5}$$

This simple step enables us to implement multiplication operations involved in computing the dot product $\mathbf{w}_i^T \mathbf{x}_i$ (3.5) in the BLP consisting of an array of M_c capacitive multipliers, and the addition of these products in a charge sharing-based adder in the CBP. The Compute Sensor's classification accuracy $p_c = Pr\{\hat{y} = y\}$ is calculated in the same manner as that in the conventional system.

3.2.2 Behavioral and Energy Models

Behavioral models describe the input-output relationship of the various blocks constituting the Compute Sensor while accounting for circuit non-idealities. These models can be employed in system simulations to estimate the performance of algorithms implemented on the Compute Sensor. In this chapter, the noise sources included in the behavior model are: (1) spatial threshold mismatch in the APS array, (2) temporal noise in the APS array, and (3)

non-linearities in the BLP. Figure 3.3(b) shows that the first two stages of the Compute Sensor - the APS array and the S/H blocks - map light energy incident on the i^{th} row of pixels to $\mathbf{x}_i$ as follows:

$$\mathbf{x}_i = x_{max}\mathbf{1} - \gamma \mathbf{I}_i + \boldsymbol{\eta}_{s,i} + \boldsymbol{\eta}_{a,i} \tag{3.6}$$

where $\mathbf{x}_i \in \mathcal{R}^{M_c}$ is a discrete-time continuous-amplitude voltage representation of the luminous exposure $\mathbf{I}_i$ incident on the i^{th} row of pixels, x_{max} is the maximum output, $\mathbf{1}$ is a column vector with all ones, γ is the conversion gain, $\boldsymbol{\eta}_{s,i} \in \mathcal{R}^{M_c}$ is a vector of samples from $\mathcal{N}(0, \sigma_s^2)$ representing the impact of spatial mismatch in device parameters across the APS array, and $\boldsymbol{\eta}_{a,i} \in \mathcal{R}^{M_c}$ is a vector of samples from $\sim \mathcal{N}(0, \sigma_n^2)$ representing the thermal noise in the APS array. To derive this model, we note that during the APS operation, the exposed PD voltage is first sampled on the sampler in S&H block in Fig. 3.3(a) by selecting the associated word line (WL). When the i^{th} row is selected, the voltage on the j^{th} sampler V_{SIG} can be expressed as:

$$V_{SIG,j} = V_{PDrst} - V_{gs0} - [\frac{\kappa_1}{C_{PD}} - \kappa_2(V_{gs0} - V_{gs1})]I_{i,j} + \Delta V_{th,j} + V_{n,j} \tag{3.7}$$

where V_{PDrst} is the voltage of the PD after reset, V_{gs0} and V_{gs1} are the gate to source voltage of M_{SF} (see Fig. 3.2(a)) in dark and highest illumination condition, C_{PD} is the parasitic capacitance at the PD node, $\Delta V_{th,j}$ is the threshold mismatch, $V_{n,j}$ is the output referred RD noise, and κ_1 and κ_2 are fitting parameters. The model in (3.6) can be derived by noting that:

$$x_{max} = V_{PDrst} - V_{gs0} \tag{3.8}$$

$$\gamma = [\frac{\kappa_1}{C_{PD}} - \kappa_2(V_{gs0} - V_{gs1})] \tag{3.9}$$

$$\tag{3.10}$$

and the threshold mismatch $\Delta V_{th,j}$ and noise $V_{n,j}$ are modeled as normally distributed random variables with variances σ_s^2 and σ_n^2, respectively.

The BLP scales each pixel value by the weight $w_{i,j}$ using a mixed-signal capacitive multi-

plier [82] as follows:

$$\mathbf{y}_{m,i} = \rho_0(x_{max}\mathbf{1} - \mathbf{x}_i) * \mathbf{w}_i + \rho_1\mathbf{x}_i + \rho_2\mathbf{w}_i + \boldsymbol{\eta}_{m,i} \tag{3.11}$$

where $*$ represents the element-wise product of two vectors, $\rho_0 \tilde{} \rho_2$ captures the non-linearity due to charge sharing based computation, and $\boldsymbol{\eta}_{m,i}$ is a vector of samples from $\mathcal{N}(0, \sigma_m)$ representing the impact of reset mismatches. To derive this model, we first show the operation principle of the capacitive multiplier. In the Compute Sensor, the scaling operation between input $\Delta V_{SIG,i,j} = V_{PDrst} - V_{gs0} - V_{SIG,i,j}$ and weight $w_{i,j}$ is realized in the bit-line processor employing a mixed-signal capacitive multiplier as shown in Fig. 3.4(a). We next drop the index (i, j) and denote the analog voltage and B_p-b digital weight as ΔV_{SIG} and $w = \sum_{i=0}^{B_p-1} p_i 2^{-(B_p-i)}$, respectively, for notational simplicity. The capacitive multiplier employs successive charge sharing to obtain a voltage V_m of:

$$V_m \;\; = \;\; V_{pre} - w(V_{pre} - (V_{PDrst} - V_{gs0}) - \Delta V_{SIG}) \tag{3.12}$$

By choosing $V_{pre} = V_{PDrst} - V_{gs0}$, the voltage drop ΔV_m is thus:

$$\Delta V_m = V_{pre} - V_m = w\Delta V_{SIG} \tag{3.13}$$

To account for the nonlinearity due to charge sharing based operation and the mismatch in the reset transistors, the following model is employed:

$$\Delta V_m = \rho_0 \Delta V_{SIG} w + \rho_1 V_{SIG} + \rho_2 w + \eta_m \tag{3.14}$$

The behavior in (3.11) can be obtained by noting that $y_m = \Delta V_m$ and $\Delta V_{SIG} = V_{PDrst} - V_{gs0} - V_{SIG} = x_{max} - x$.

After the BLP, the CBP uses charge sharing-based circuits to sum up the elements of $\mathbf{y}_{m,i}$ and obtain the dot product $\mathbf{w}_i^T\mathbf{x}_i$:

$$y_{s,i} = \mathbf{1}^T\mathbf{y}_{m,i} \tag{3.15}$$

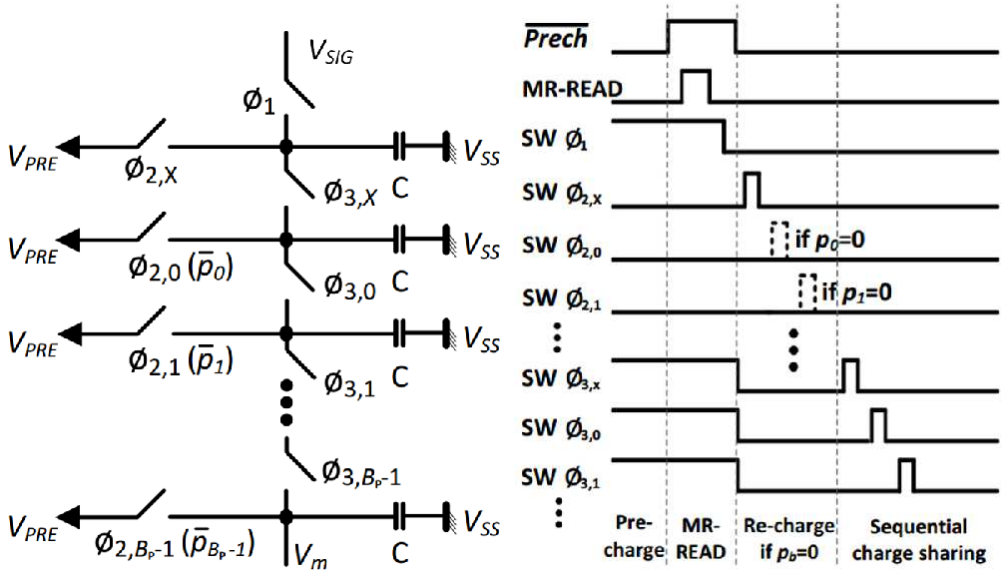


Figure 3.4: Capacitive multiplier (a) architecture and (b) timing diagram.

The residual digital processor maintains a running sum of the row-wise dot products in order to compute the output $y_o = \sum_i y_{s,i} - b$ in (3.5) followed by $\hat{y} = sign(y_o)$ as the computed label. Equations (3.6)-(3.15) describes the behavior of the Compute Sensor. Table 3.1 lists the model parameters values in a 65nm CMOS process. These equations can be employed to estimate the system behavior of the Compute Sensor.

The Compute Sensor's energy consumption per decision, i.e., in processing one $M_r \times M_c$ image, is given by:

$$E_{CS} = M_r M_c (E_p + E_m) + M_r (2E_{adc} + 2E_{add}) + E_{add} \tag{3.16}$$

where E_p, E_m, E_{adc}, and E_{add} are the energy consumptions of the pixel, capacitive multiplier, the ADC, and a digital adder, respectively.

The conventional system needs to convert all pixel values into the digital domain then process digitally. The energy consumption per decision is given by:

$$E_{conv} = M_c M_r (E_p + E_{adc} + E_{rd}) + M_c M_r E_{mac} \tag{3.17}$$

where E_{rd} and E_{mac} are the energy per readout and multiply-accumulate (MAC) operation, respectively. The behavioral and energy model will be employed in Section 3.3 to evaluate the system performance and energy savings. The energy savings from Compute Sensor are evident from (3.16) and (3.17). The key savings arise from having the ADC operate on row-wise dot products giving rise to the multiplicative factor of $2M_r$ as compared to the factor of M_cM_r ($M_c >> 2$) for the conventional system. The second source of energy savings arises from the analog domain multiplication in the Compute Sensor compared to digital domain because $E_{mac} \approx 3E_m$ or $4E_m$.

3.3 Simulation Results

We first validate the behavioral and energy models described in Section 3.2 using the parameters of a 65 nm CMOS process. The system performance and energy savings achieved by Compute Sensor are estimated using these models. In the following, the conventional system is assumed to be operating with noise-free data and ideal digital computations.

The Compute Sensor architecture in this study consists of a 32×32 APS array, a capacitive multiplier array with $5b$ weight, a 8 b column ADC array, and 16 b addition in the digital domain. The conventional digital implementation has an identical APS array and ADC, but employs a digital MAC with 8 b input, 5 b weight, and 32 b output. These precisions are the minimum needed for the conventional architecture to achieve a classification accuracy $p_c = 95\%$. The face and non-face images extracted from the Caltech101 dataset [80] consisting of 32×32 gray-scale images are employed. During the system simulation, a linear mapping from the pixel values to the luminous exposure $\mathbf{I}_i$ is employed and used in (3.6).

3.3.1 Model Validation

Table 3.1 lists the parameter values for the behavioral model in (3.6) obtained by curve fitting to the results of circuit simulations of a standard 3-transistor APS. The model is found to match detailed circuit simulations to within 5.2% when the pixel output $x_{i,j}$ lies in the interval $[0.2, 0.9]$ as shown in Fig. 3.5(a). The standard deviation of spatial mismatch σ_s

was found to lie in the interval $[1.62 \times 10^{-2}, 2 \times 10^{-2}]$ using Monte Carlo circuit simulations and is shown in Fig. 3.5(b). The standard deviation of output referred noise σ_n was found to lie in the interval $[7 \times 10^{-4}, 7.5 \times 10^{-4}]$. The model parameters in (3.11) were obtained using the methodology in [82] and are also listed in Table 3.1.

Table 3.1: Model Parameters in 65 nm CMOS

$x_{max}(V)$	$\gamma(V/(lx \cdot s))$	$\sigma_s(V)$	$\sigma_n(V)$
0.9	4.39×10^{-5}	2×10^{-2}	7.5×10^{-4}
ρ_0	ρ_1	$\rho_2(V)$	$\sigma_m(V)$
0.93	1.2×10^{-2}	6.68×10^{-4}	1.6×10^{-2}

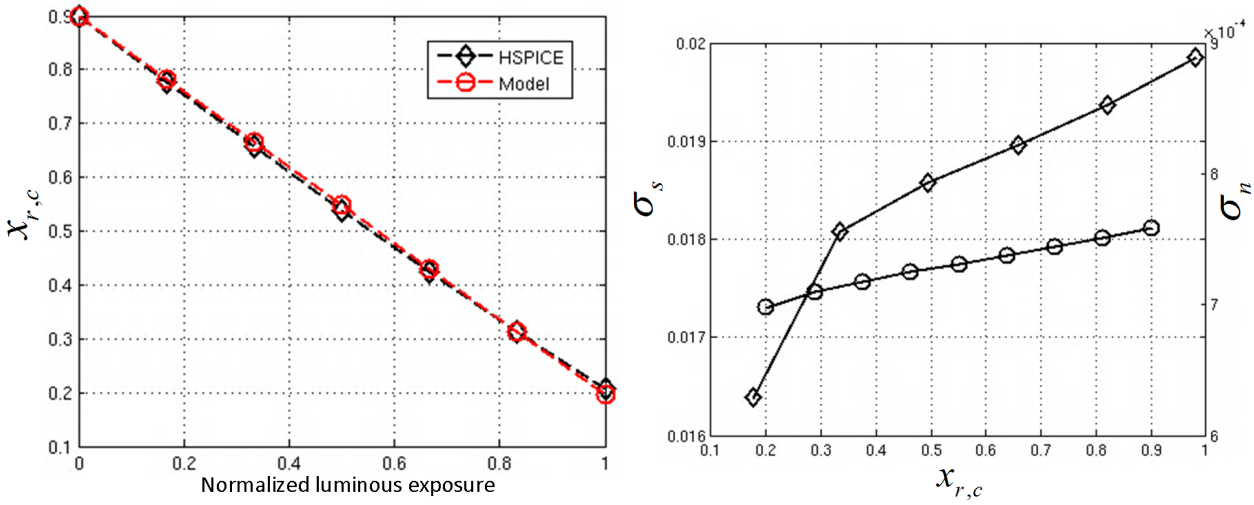


Figure 3.5: Model characterization and validation: (a) linearity, and (b) standard deviation of mismatch and noise.

3.3.2 Classification Accuracy

The classification accuracy of Compute Sensor is evaluated employing the models in Section 3.3.1 with parameters from Table 3.1.

Figure 3.6(a) shows that the Compute Sensor is able to achieve a classification accuracy $p_c = 94.7\%$ at the nominal values of spatial mismatch $\sigma_s = 2 \times 10^{-2}$, multiplier mismatch $\sigma_m = 2 \times 10^{-2}$, and noise $\sigma_n = 7.5 \times 10^{-4}$. This accuracy is very close to the value of 95% achieved by the ideal digital implementation. In fact, the Compute Sensor is able to

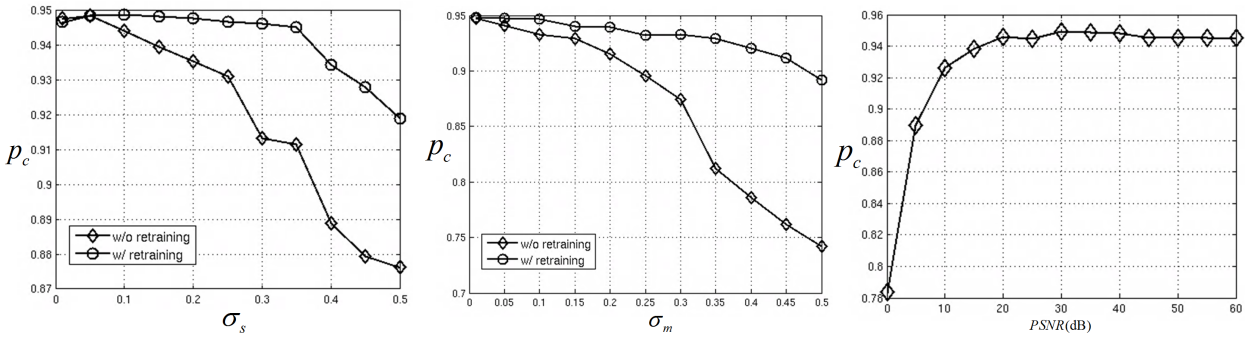


Figure 3.6: Classification accuracy of the Compute Sensor wrt.: (a) APS spatial mismatch, (b) capacitive multiplier mismatch, and (c) input peak signal-to-noise ratio ($PSNR$).

maintain $p_c \geq 94\%$ when σ_s is increased to 0.1, which is 5× more than the nominal value, using the hyperparameters obtained with the nominal value of σ_s. Any further increases in σ_s lead to a large reduction in p_c. For example, p_c decreases to 87% when σ_s increases to 0.5. Next, we retrain the Compute Sensor with data generated in the presence of spatial mismatch. Figure 3.6(a) shows that the Compute Sensor achieves a $p_c = 92\%$ when $\sigma_s = 0.5$ after retraining. This clearly indicates the effectiveness of retraining in order to compensate for spatial mismatch in the APS array. Retraining can also be employed to address the computational errors due to capacitive multiplier mismatch $\boldsymbol{\eta}_m$. A similar study was conducted to observe the impact of multiplier mismatch σ_m as shown in Figure 3.6(b) where σ_s and σ_n were set at their nominal values. This figure shows that the Compute Sensor achieves $p_c = 90\%$ in the presence of $\sigma_m = 0.5$ with retraining which is a significant improvement over case when retraining was not employed.

Classification accuracy is a function of the input peak signal-to-noise ratio $PSNR$ defined as $PSNR = 20\log_{10}\frac{x_{max}}{\sigma_n}$. A $PSNR = 61\,\text{dB}$ is obtained with the nominal values of $x_{max} = 0.9$, σ_s, σ_n. At this value of $PSNR$, a classification accuracy of 94.7% is achieved. Figure 3.6(c) shows that the Compute Sensor's classification accuracy decreases to 78% as the $PSNR$ reduces to 0 dB.

To further understand the performance of Compute Sensor, PCA is performed on the feature vectors obtained from the behavior model in Section 3.2.2. Figure 3.7(a) shows the distribution of the feature vectors when circuit non-idealities are absent. In this case, the SVM chooses a hyperplane that successfully separates the two classes. However, in the

presence of spatial and multiplier mismatch ($\sigma_s = \sigma_m = 0.3$), the feature vectors shift as shown in Figure 3.7(b). The classification accuracy falls if the original hyperplane is used. However, retraining with the new set of feature vectors enables the Compute Sensor to obtain a new separating hyperplane with a commensurate improvement in the classification accuracy as shown in Figure 3.7(c). These results indicate that the Compute Sensor may need to adapt to changing environmental conditions such as temperature in order to ensure that the optimal separating hyperplane is generated and employed for classification.

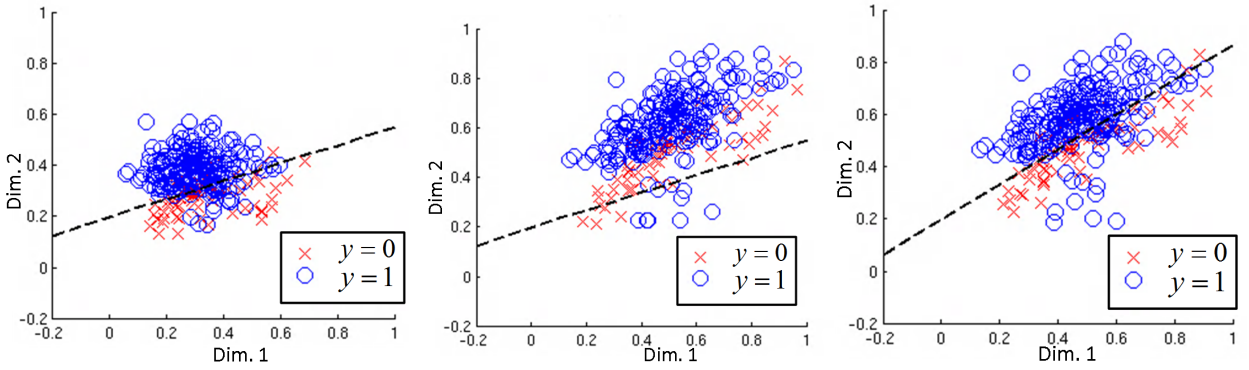


Figure 3.7: The feature distribution and SVM separation hyper-plane when: (a) $\sigma_s = \sigma_m = 0$ without retraining, (b) $\sigma_s = \sigma_m = 0.3$ without retraining, and (c) $\sigma_s = \sigma_m = 0.3$ with retraining.

3.3.3 Energy Savings

In order to compare the energy consumption of the Compute Sensor with the conventional architecture, we employ the energy numbers in Table 3.2 which are based on circuit simulation and published energy numbers from [6, 83].

Table 3.2: Energy per Pixel Processing in 65 nm CMOS

$E_p(pJ)$	$E_{adc}(pJ)$	$E_{rd}(pJ)$	$E_m(pJ)$	$E_{mac}(pJ)$	$E_{add}(pJ)$
2.69	20.5	5	0.77	3.2	0.1

Figure 3.8(a) shows that the proposed Compute Sensor consumes 6.2× less energy compared with conventional implementation. The main source of energy savings is due to the elimination of the per bit-line ADC and RD energy and from the use of analog dot product

computations in the Compute Sensor. For example, the 1024-length dot product in analog consumes 0.79 nJ, which is 4.1× less than the 3.28 nJ needed by the digital implementation.

We also studied the energy savings as a function of the array size as shown in Figure 3.8(b). Indeed, the energy savings increases from 6.2× to 11× as the APS array size increases from 32×32 to 512×512. This is because the Compute Sensor performs ADC operations row-wise on dot products, as compared to the conventional architecture which performs ADC operation pixel-wise on scalars.

Another opportunity to reduce energy consumption is to reduce the APS current. However, doing so will degrade the input $PSNR$. Specifically, one of the fundamental noise sources in APS is the thermal noise, whose noise power is described by:

$$\sigma_n^2 = kT/C \tag{3.18}$$

where k is the Boltzmann's constant, T is the temperature, and C is the sampling capacitance. The bandwidth of the APS can be approximated by:

$$B = \frac{g_m}{C} = \frac{I_{aps}}{V_{ov}C} \tag{3.19}$$

where I_{aps} is the current consumption of the APS array, V_{ov} is the overdrive voltage and is a fixed parameter chosen during the design. A fundamental trade-off between noise and bandwidth (thus speed) can be seen from (3.18) and (3.19). For fixed bandwidth, reducing C will allow smaller I_{aps} thus lower energy, but will increase the noise variance σ_n^2 per (3.18). More specifically, the $PSNR$ is related to the current I_{aps} via:

$$PSNR = 20log_{10}(x_{max}/\sigma_n) \propto 10log_{10}(I_{aps}) \tag{3.20}$$

and the energy of the APS is related with I_{aps} via:

$$E_{pix} = V_{dd}I_{aps}T_{pix} \tag{3.21}$$

where V_{dd} and T_{pix} is the supply voltage and the pixel access time, respectively. The degraded

$PSNR$ may be acceptable if retraining is employed as suggested in Figure 3.6(c). This figure shows that the Compute Sensor is able to achieve less than 1% performance drop from the ideal digital performance of $p_c = 95\%$ for $PSNR \geq 20\,\text{dB}$. This relaxed $PSNR$ requirement allows the APS array current to be reduced for additional energy savings. Figure 3.8(c) shows that the energy savings increases to 17× as the $PSNR$ decreases from the 61 dB to 20 dB.

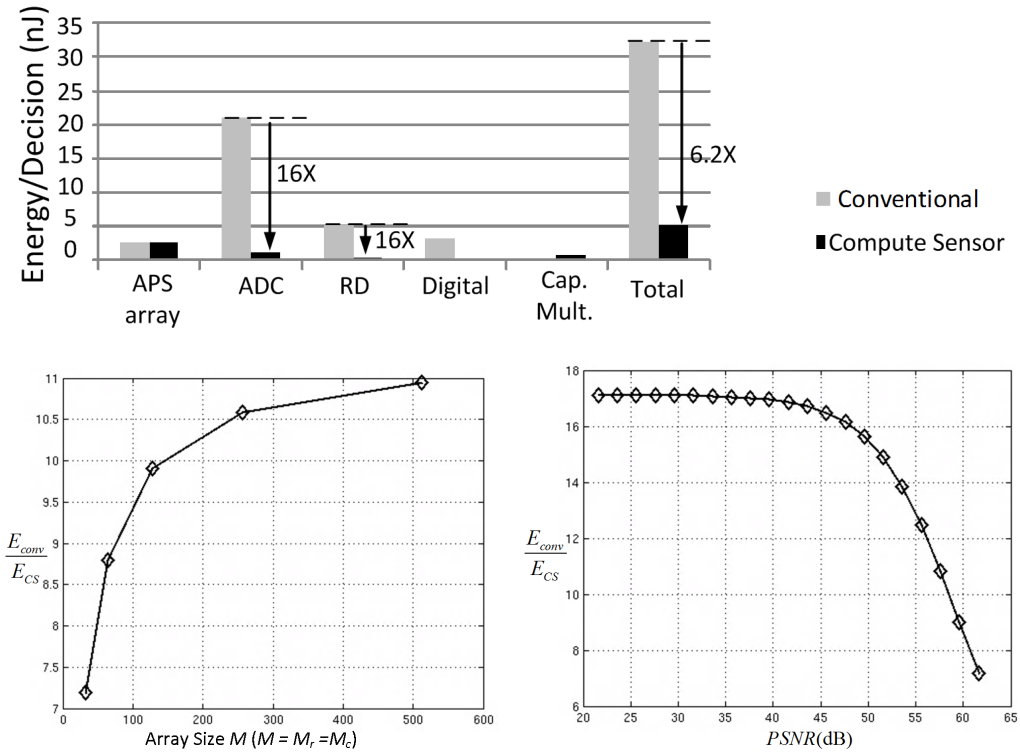


Figure 3.8: Energy per decision: (a) energy breakdown, (b) energy savings vs. APS size, and (c) energy savings vs. $PSNR$.

3.4 Discussion

We have shown the benefits of embedding information processing functionality into the sensory substrates. We note that such embeddings are made possible due to the intrinsic ability of machine learning algorithms to adapt to noise. Behavioral models such as those in Section 3.2.2 can be employed to develop a variety of machine learning algorithms for Compute

Sensor style architectures. We believe that more powerful machine learning algorithms including deep neural networks, ensemble methods such as bagging and boosting, decision trees, and random forest, can also potentially be embedded into the Compute Sensor. The huge design space spanned by the Compute Sensor encompassing algorithms, architectures, circuits, and sensors, can be a challenge when searching for energy-optimal implementations. Another formidable challenge that we hope to address in the future is to design programmable Compute Sensor architectures whereby a variety of algorithms can be mapped on to the same platform. Silicon prototypes are necessary to demonstrate the benefits of Compute Sensor in real world applications. An initial characterization chip (see Fig. 3.9) containing 8 different types of CMOS APS arrays in 65 nm CMOS has been taped-out. The chip allows flexible control of the APS supply voltage, bias current, readout timing and pulse widths. It will be employed to characterize the APS model as well as the trade-off between performance and energy consumption, and for future integration with various learning kernels.

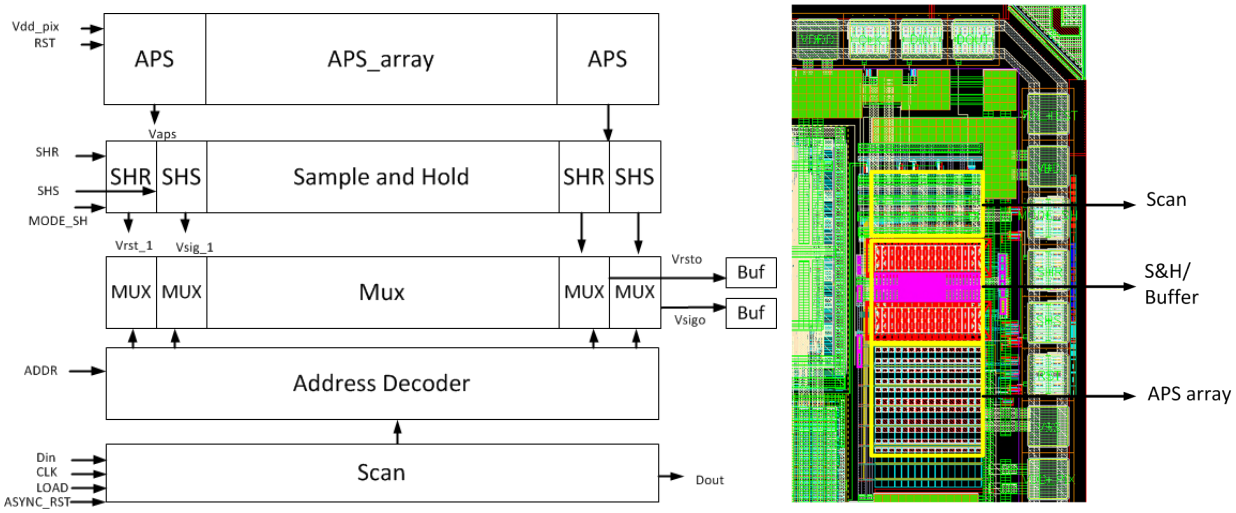


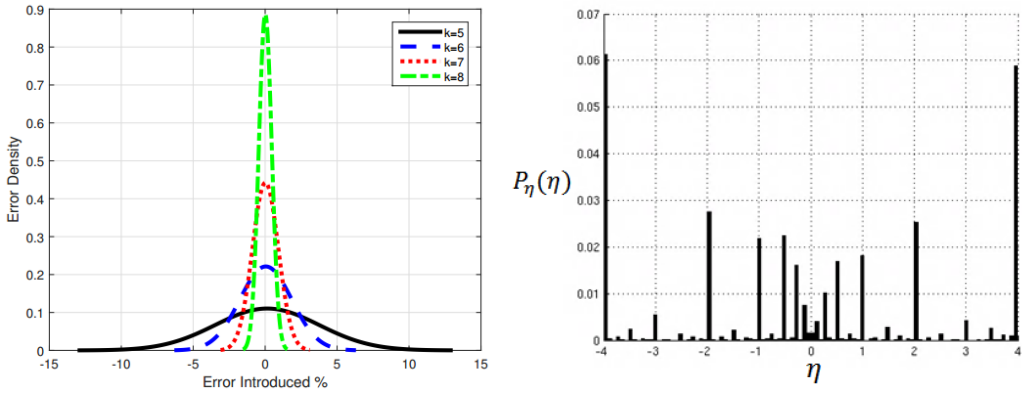
Figure 3.9: Compute Sensor characterization chip in 65 nm CMOS: (a) chip architecture, and (b) chip layout.

Chapter 4

EMBEDDED ALGORITHMIC-NOISE TOLERANCE

In previous chapters, the energy efficiency of in-silicon machine learning kernels is improved by integrating information processing into the energy delivery and sensing fabrics thereby eliminating the fundamental losses associated with the conventional architectures or the communication overhead between sensing and computing. The resultant energy efficiency vs. robustness trade-off that is exploited by retraining the hyper parameters in the machine learning algorithms to compensate for the circuit level non-idealities. Such an energy efficiency vs. robustness trade-off also arises by implementing information processing subsystems on stochastic fabrics, i.e., low SNR fabrics due to operating in different regime or new devices. Examples of stochastic fabrics include CMOS circuits operating with overscaled supply voltage, near/subthreshold voltage (NTV) CMOS [84] and emerging nanoscale devices such as CNFET [85], spin [86], and others. As pointed out in Charpter 1, these stochastic fabrics have the potential to achieve high energy efficiency, but are subject to various kinds of hardware errors.

Hardware errors in stochastic fabrics have unique properties and should be distinguished from the input noise and approximation errors such as those in approximate computing (AC) literature [87, 88, 89]. Input noise in the feature vectors occurs during the data acquisition process. Although it has been shown that many machine learning algorithms are robust to noise [90, 91], and that adding noise during the training might even improve the performance [92, 93], it is always assumed that the noise power is much smaller than the signal power [92], and that the computation is error-free. Approximation errors occur when complex operations/circuits are replaced with simpler approximated ones and thus are static errors. Both logic level AC [87, 88, 89], and algorithmic level AC [94, 95] have been proposed to improve the energy efficiency of machine learning algorithms. However, the



(a) (b)

Figure 4.1: The error probability mass function of: (a) the approxiamtion errors in an approximate multiplier in [96], and (b) the hardware errors (timing errors) of an $8b$ mutliplier operating with scaled voltage of $V_{dd} = 0.7\,\mathrm{V}$. Unlike the approximation errors, the hardware errors are both dynamic and large-magnitude.

performance improvement relies solely on the inherent algorithmic robustness, thus limiting the approximation errors to be of small magnitude. Furthermore, the computation at the circuit level is also assumed to be error-free. In contrast, hardware errors occur during the computations on the circuit fabrics. These errors are complex functions of the circuit state, inputs, architecture, and the process technology, and can be both dynamic and large-magnitude (see Fig. 4.1). This is particularly the case if the errors are timing errors in DSP data path circuits [53], since these errors are most significant bit (MSB) errors and can directly lead to decision failures [53]. As a result, hardware errors are usually far more detrimental to the system performance compared with the input noise and approximation errors, and cannot be compensated for via the inherent robustness of machine learning algorithms. Therefore, statistical error compensation techniques are needed to detect and compensate for these errors.

Starting from this chapter, we will explore the use of error resiliency techniques to compensate for the hardware errors on the stochastic fabrics, so that large energy savings can be achieved without loss of system level performance. Error resiliency techniques have been proposed [53] to enhance energy efficiency by reducing design margins and compensating for the resultant errors. Large design margins arise from the need to provide robustness in the

presence of process, voltage, and temperature variations [97], and represent an energy overhead as high as 3×-to-4× [98]. The key to the use of error resiliency for energy reduction is that such techniques need to be low overhead and yet effective in compensating for high error rates. Classical fault-tolerance techniques such as N-modular redundancy (NMR) rely on replication of the main computation block and as a result are ineffective for the purposes of energy reduction. Hence, low overhead error resiliency techniques such as RAZOR [99, 100], error-detection sequential (EDS) [101], and confidence driven computing (CDC) [102] have been proposed to enhance energy efficiency. These techniques employ rollback based error correction, and are suitable when operating close to point of first failure (PoFF). Unlike the rollback based techniques, statistical error compensation (SEC) [53] is a class of system level error compensation techniques that utilizes signal and error statistics and hence is particularly well-suited for signal processing and machine learning systems. These techniques include algorithmic noise tolerance (ANT), soft NMR, and stochastic sensor network on a chip (SSNOC) [53], and have been shown to compensate for error rates ranging from 0.21 to 0.89, with a combined error detection and correction overhead ranging from 5% to 30% resulting in energy savings ranging from 35% to 72%.

ANT [53] is a specific SEC technique that has been shown to be effective in compensating for high error rates in signal processing and machine learning kernels. For example, the reduced precision replica (RPR) ANT technique and prediction based ANT was employed to compensate for error rates of $0.27 \sim 0.58$ in an ECG processor [103, 104] while delivering the required application-level performance. The overhead in ANT ranges from 5% to 30% [103] due to the use of explicit estimator blocks in error compensation. This overhead, though small compared to other techniques, limits the achievable systems level energy efficiency to $28\% \sim 41\%$.

In this chapter, we propose embedded algorithmic-noise tolerance (E-ANT), a new class of statistical error compensation (SEC) techniques aiming to reduce the error compensation complexity associated with conventional SEC techniques. E-ANT operates by reusing part of the main block as an estimator and thus embedding it into the main block. At the architectural level, we propose ARCH-ANT, which employs data path decomposition (DPD) to embed the RPR estimator into the main block. At the algorithmic level, we propose ALG-

ANT, which employs additional optimization constraints during algorithm to architecture mapping to design incremental refinement architectures. The logic overhead of E-ANT is reduced to below 8% from the 20%-44.1% [57, 65] overhead associated with conventional ANT system. To evaluate the improved robustness and energy savings of the proposed technique, ARCH-ANT and ALG-ANT are applied to the design of an EEG seizure classification system consisting of a frequency selective filter bank as the feature extractor and a support vector machine (SVM) as the classifier. Simulation results in a commercial 45 nm CMOS process show that ARCH-ANT can compensate for error rates up to 0.38, and ALG-ANT can compensate for error rates up to 0.41, while maintaining a true positive rate $p_{tp} > 0.9$ and a false positive rate $p_{fp} \leq 0.01$. This error tolerance is employed to reduce energy via the use of voltage overscaling (VOS). ARCH-ANT and ALG-ANT are able to achieve up to 51% and 44% energy savings, respectively.

The rest of the chapter is organized as follows. Section 4.1 describes the background of the ANT technique and the SVM EEG classification system architecture. Section 4.2 presents the principle of ARCH-ANT and ALG-ANT. Section 4.3 presents the design optimization of ARCH-ANT and ALG-ANT compute kernels and their application to the SVM EEG classification system. Conclusions are presented in Section 4.4.

4.1 Background

4.1.1 Conventional ANT

Conventional ANT incorporates a main block (**M**) and an estimator (**E**) as shown Fig. 4.2(a). The **M**-block implements the algorithm of interest and is conventionally error-free. In ANT, the **M**-block is permitted to make errors, which are then compensated for by the rest of the blocks in Fig. 4.2(a) including the **E**-block. In RPR ANT, the **E**-block is obtained by reducing the precision of the **M**-block. The **M**-block is subject to large magnitude errors η (e.g., timing errors due to critical path violations which typically occur in the MSBs) while the **E**-block is subject to small magnitude errors e (see Fig. 4.2(b), e.g., due to quantization noise in the LSBs), i.e.:

$$y_a = y_o + \eta \tag{4.1}$$

$$y_e = y_o + e \tag{4.2}$$

where y_o, y_a, and y_e are the error-free, the **M,** and **E**-block outputs, respectively. ANT exploits the difference in the statistics of η and e to detect and compensate for errors to obtain the final corrected output $\hat{y}$ as follows:

$$\hat{y} = \begin{cases} y_a \text{ if } |y_a - y_e| \leq T_h \\ y_e \quad \text{otherwise} \end{cases} \tag{4.3}$$

where T_h is an application dependent threshold parameter chosen to maximize the performance of ANT. In this paper, T_h is chosen to equal $max(|y_o - y_e|)$ as this ensures that the **M**-block output y_a will always be selected [103] when the output is error free. The error rate p_η is defined as:

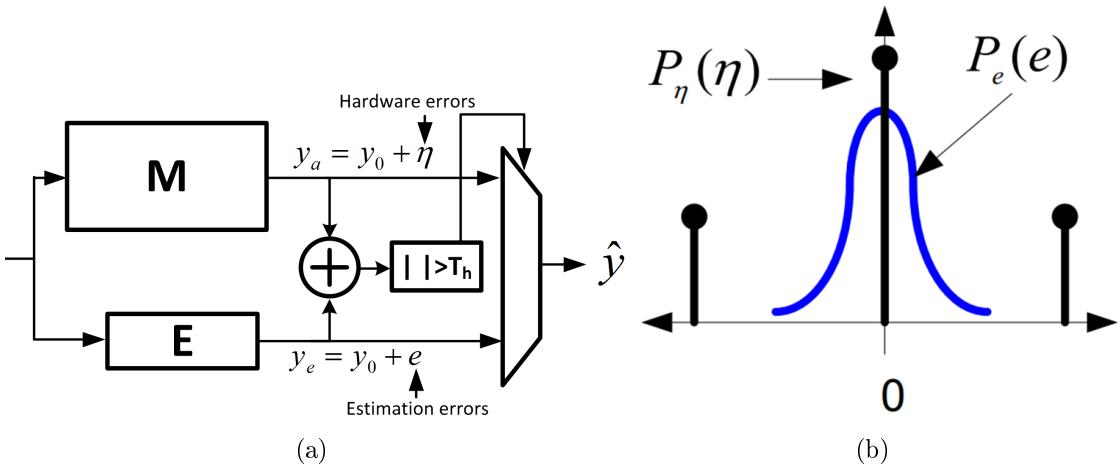
$$p_\eta = 1 - P_\eta(0) = Pr\{\eta \neq 0\} \tag{4.4}$$

where $P_\eta(\cdot)$ is the error probability mass function (PMF) of η. The errors η are most conveniently obtained by applying voltage overscaling (VOS) where the supply voltage V_{dd} is scaled as follows:

$$V_{dd} = K_{vos} V_{dd-crit} \tag{4.5}$$

where K_{vos} is the voltage overscaling factor, and $V_{dd-crit}$ is the minimum voltage needed for error free operation in the **M**-block. Note that for the ANT system to work properly, the **E**-block is not permitted to make large magnitude errors such as those arising from timing violations. This helps maintain the difference in the error statistics at the output of the **M** and **E**-block as shown in Fig. 4.2(b).

The performance improvement achieved by ANT can be evaluated by employing a system level metric such as the signal-to-noise ratio (SNR). Assume that the error-free output in



(a) (b)

Figure 4.2: Algorithmic noise-tolerance (ANT): (a) conventional architecture, and (b) the error statistics in the main (**M**) and estimator (**E**) blocks.

Fig. 4.2(a) is expressed as:

$$y_o = s + n_s \tag{4.6}$$

where s and n_s represent the signal and noise components in the error-free output y_o, respectively. At the application level, one is interested in the ratio of the signal power σ_s^2 to the noise powers at the outputs of the **M**, the **E**-block, and the ANT system. Thus, the following application level SNRs can be defined:

$$SNR_{M,a} = 10log_{10}(\frac{\sigma_s^2}{\sigma_{n_s}^2 + \sigma_\eta^2}) \tag{4.7}$$

$$SNR_{E,a} = 10log_{10}(\frac{\sigma_s^2}{\sigma_{n_s}^2 + \sigma_e^2}) \tag{4.8}$$

$$SNR_{ANT,a} = 10log_{10}(\frac{\sigma_s^2}{\sigma_{n_s}^2 + \sigma_{n_r}^2}) \tag{4.9}$$

where σ_s^2, $\sigma_{n_s}^2$, σ_η^2, σ_e^2, $\sigma_{y_o}^2$ and $\sigma_{n_r}^2$ are the variances of the signal s, noise n_s, **M**-block hardware error η, **E**-block estimation error e, error-free output y_o, and residual error $n_r = y_o - \hat{y}$, respectively. It is also of interest to evaluate how 'noisy' the circuit fabric is with respect to an error-free (conventional) architecture. By definition, the output of such an architecture

is y_o. Thus, we define the circuit level SNRs as follows:

$$SNR_{M,c} = 10log_{10}(\frac{\sigma^2_{y_o}}{\sigma^2_{\eta}}) \quad (4.10)$$

$$SNR_{E,c} = 10log_{10}(\frac{\sigma^2_{y_o}}{\sigma^2_{e}}) \quad (4.11)$$

$$SNR_{ANT,c} = 10log_{10}(\frac{\sigma^2_{y_o}}{\sigma^2_{n_r}}) \quad (4.12)$$

If error detection is ideal, then $n_r \in \{0, e\}$, and its probability mass function (PMF) $P_{\eta_r}(n_r)$ is given by:

$$P_{\eta_r}(n_r) = \begin{cases} 1 - p_\eta & \text{if } n_r = 0 \\ p_\eta & \text{if } n_r = e \end{cases}$$

and

$$\sigma^2_{n_r} = p_\eta \sigma^2_e$$

where p_η is the error rate of the **M**-block defined in (4.4). Therefore, (4.9) and (4.12) can be expressed as:

$$SNR_{ANT,a} = 10log_{10}(\frac{\sigma^2_s}{\sigma^2_n + p_\eta \sigma^2_e}) \quad (4.13)$$

$$SNR_{ANT,c} = 10log_{10}(\frac{\sigma^2_{y_o}}{p_\eta \sigma^2_e}) \quad (4.14)$$

Since e is the small magnitude LSB error and η is the large magnitude MSB error, $p_\eta \sigma^2_e \ll \sigma^2_e \ll \sigma^2_\eta$. This further implies that $SNR_{ANT,a} \gg SNR_{E,a} \gg SNR_{M,a}$, and $SNR_{ANT,c} \gg SNR_{E,c} \gg SNR_{M,c}$. Thus, the output SNR of the ANT system is significantly greater than the SNR at the output of either the **M** or **E**-block. This phenomenon occurs in spite of the fact that the ANT system output $\hat{y} \in \{y_a, y_e\}$ (see Fig. 4.2(a)), i.e., $\hat{y}$ equals the output of either the **M** or **E**-block. The reason for this unique feature of ANT is that it exploits the difference in the error statistics (see Fig. 4.2(b)) at the output of the **M** and **E**-block.

4.1.2 EEG Classification System using SVM

Portable health monitoring is an important class of applications that can benefit from the design of energy efficient machine learning kernels. It has been shown [105] that epileptic seizures can be efficiently detected by analyzing the EEG signal using an SVM kernel. The EEG seizure classification system [105] shown in Fig. 4.3(a) consists of a frequency selective filter bank to extract signal energy in the $0-20\,\text{Hz}$ range and a SVM classifier. The filter bank has passband of $3\,\text{Hz}$ with a transition band of $1.5\,\text{Hz}$. Eight channels are employed to cover the entire frequency range [105].

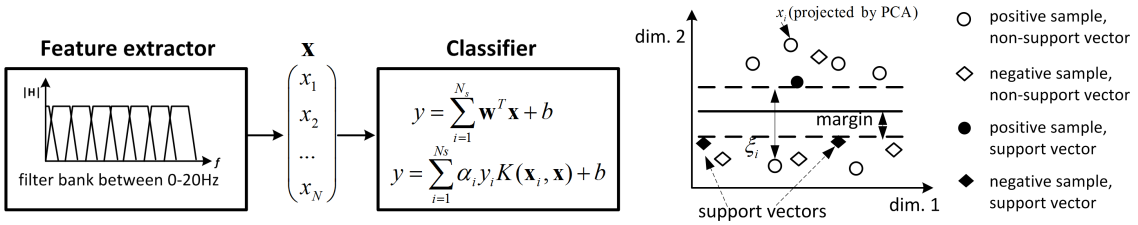
SVM [79] is a popular supervised learning method for classification and regression. An SVM operates by first training the model (the training phase) followed by classification (the classification phase). During the training phase, feature vectors with labels are used to train the model. During the classification phase, the SVM produces a predictive label when provided with a new feature vector. The SVM training can be formulated as the following optimization problem to determine the maximum margin classifier [79] (see Fig. 4.3(b)):

$$\begin{aligned} &\min \tfrac{1}{2}\|\mathbf{w}\|^2 + C\sum_i \xi_i \\ &s.t. \\ &y_i(\mathbf{w}^T\mathbf{x}_i - \text{b}) \geq 1 - \xi_i \\ &\xi_i \geq 0 \end{aligned} \tag{4.15}$$

where C is the cost factor, ξ_i is the soft margin, $\mathbf{x}_i$ is the feature vector, y_i is the label corresponding to the feature vector $\mathbf{x}_i$, $\mathbf{w}$ is the weight vector, and b is the bias. The trained model is represented by:

$$y = \mathbf{w}_o^T\mathbf{x} - b \tag{4.16}$$

where $\mathbf{w}_o$ are the optimized weights. It can be shown that the optimum weights are represented as a linear combination of the feature vectors that lie on the margins (see Fig. 4.3(b)), i.e., support vectors:



(a) (b)

Figure 4.3: EEG seizure classifier with SVM: (a) system architecture, and (b) principle of SVM.

$$\mathbf{w}_o = \sum_{n=1}^{N_s} \alpha_n y_n \mathbf{x}_{s,n} \tag{4.17}$$

where N_s and $\mathbf{x}_{s,n}$ are the number of support vectors and n^{th} support vector, respectively. The linear model can thus be represented as:

$$y = \sum_{n=1}^{N_s} \alpha_n y_n \mathbf{x}_{s,n}^T \mathbf{x} - b \tag{4.18}$$

The linear SVM in (4.18) can be easily extended into non-linear SVM by employing the kernel trick [79], resulting in:

$$y = \sum_{n=1}^{N_s} \alpha_n y_n K(\mathbf{x}_{s,n}, \mathbf{x}) - b \tag{4.19}$$

where $K(\mathbf{x}_{s,n}, \mathbf{x})$ is a kernel function. Popular kernel functions include polynomial, radial basis function (RBF), and others [106].

4.2 Proposed E-ANT

E-ANT reuses part of the main block $\mathbf{M}$ to generate an estimate of its error free output y_o. This is in contrast to conventional SEC techniques, where an explicit estimator is required. Such embedding of the estimator can be performed either at the architectural or the algorithmic level. At the architectural level, data path decomposition can be employed to transform an existing architecture into an error resilient architecture, leading to the proposed

ARCH-ANT technique. At the algorithm level, traditional algorithm transforms search over the design space for optimum parameters suitable for hardware implementations. Additional training/optimization constraints can be employed to trade off performance and error resiliency, leading to the proposed ALG-ANT technique. Both techniques will be presented in this section.

4.2.1 ARCH-ANT

In RPR ANT, the **M** and **E**-blocks process the same data but with different precisions. This redundancy can be exploited to embed the **E-**block into the **M**-block via DPD. In particular, DPD decomposes the **M**-block into MSB and LSB components, and employs the output of the MSB component as an estimate of the error-free **M**-block output y_o. By ensuring that the critical path of the MSB block is always shorter than that of the **M**-block, the requirements on the error statistics (see Fig. 4.2(b)) on the **M** and **E**-block are satisfied.

Let $y_a = f(x)$ denote the **M**-block functionality, where x and y_a are the input and output of the **M**-block, respectively. A B_x-bit input $x = x_0 x_1 ... x_{B_x-1}$ can be written in 2's complement form [107], as follows:

$$x \ = -x_0 + \sum_{i=1}^{B_x-1} x_i 2^{-i} = x_M + x_L 2^{-(B_{msb}-1)} \tag{4.20}$$

where x_M is the value of B_{msb} MSB bits, and x_L is the value of $B_x - B_{msb}$ LSB bits, as shown below:

$$x_M = -x_0 + \sum_{i=1}^{B_{msb}-1} x_i 2^{-i} \tag{4.21}$$

$$x_L = \sum_{i=B_{msb}}^{B_x-1} x_i 2^{-(i-B_{msb}+1)} \tag{4.22}$$

Therefore, the **M**-block output is expressed as:

$$y_a = f(x) = f(x_M + x_L 2^{-(B_{msb}-1)})$$

In E-ANT, we decompose $f(x)$ as follows:

$$\begin{aligned} y_a &= f(x) \\ &= f(x_M + x_L 2^{-(B_{msb}-1)}) \\ &= g(f_M(x_M), f_L(x_M,\ x_L)) \end{aligned} \tag{4.23}$$

where $f_M(x_M)$ and $f_L(x_M,\ x_L)$ are functions that are combined by the operator $g(\cdot)$ to generate the final output y_a. Since this decomposition utilizes the finite precision nature of arithmetic units, it is referred to as DPD. We show that DPD exists if $f(x)$ is n-times differentiable or can be piecewise approximated. An E-ANT system can be obtained via DPD by ensuring that: (1) the critical path of $f_M(x_M)$ is shorter than that of $g(f_M(x_M), f_L(x_M,\ x_L))$, and (2) $f_M(x_M)$ generates an estimate y_e of the error-free output y_o. The operation of DPD based E-ANT is described as follows:

$$\begin{aligned} y_a &= g(f_M(x_M), f_L(x_M,\ x_L)) \\ y_e &= f_M(x_M) \\ \hat{y} &= \begin{cases} y_a \text{ if } |y_a - y_e| \leq T_h \\ y_e \quad \text{otherwise} \end{cases} \end{aligned}$$

where T_h is the error detection threshold as in (4.3). Next, we describe several methods to achieve DPD.

4.2.2 DPD via Taylor Expansion

Taylor expansion can be employed to achieve DPD. If $f(x)$ is n-times differentiable in the input range $x \in [x_l, x_u]$. The DPD for $f(x)$ using Taylor expansion is given by:

$$f(x) \approx f_M(x_M) + f_L(x_M, x_L) \tag{4.24}$$

where

$$\begin{aligned}
f_M(x_M) &= f(x_0) + \sum_{k=1}^{n}\sum_{i=0}^{k}[\frac{f^{(k)}(x_0)}{k!}\binom{k}{i}(-x_0)^{k-i}]x_{i,M} \\
f_L(x_M, x_L) &= \sum_{k=1}^{n}\sum_{i=0}^{k}[\frac{f^{(k)}(x_0)}{k!}\binom{k}{i}(-x_0)^{k-i}]x_{i,L} \\
x_{i,M} &= x_M^i \\
x_{i,L} &= \sum_{j=0}^{i-1}\binom{i}{j}x_M^j(x_L 2^{-(B_{msb}-1)})^{i-j}
\end{aligned}$$

where x_M and x_L are defined in (4.21) and (4.22), respectively.

As a special case, when a first order Taylor expansion is employed at $x_0 = \frac{1}{2}(x_l + x_u)$, i.e., at center of the input dynamic range, (4.24) simplifies into:

$$f(x) \approx f(x_0) + f'(x_0)(x - x_0) \tag{4.25}$$

where $f'(x_0)$ is the first order derivative of $f(x)$ at x_0. Substituting (4.20) into (4.25), we obtain the DPD of $f(x)$ as follows:

$$\begin{aligned}
f(x) &\approx f(x_0) + f'(x_0)(x_M + x_L 2^{-(B_{msb}-1)} - x_{0,M} - x_{0,L}2^{-(B_{msb}-1)}) \\
&= f_M(x_M) + f_L(x_L)2^{-(B_{msb}-1)}
\end{aligned} \tag{4.26}$$

where

$$
\begin{aligned}
f_M(x_M) &= f(x_0) + f'(x_0)(x_M - x_{0,M}) \\
f_L(x_L) &= f'(x_0)(x_L - x_{0,L})
\end{aligned}
$$

and $f_M(x_M)$ can be used as the **E**-block. Note that in the decomposition in (4.26), only x is decomposed into MSB and LSB components and the factor $f'(x_0)$ remains in full precision. If a simpler **E**-block is required, $f'(x_0)$ can also be decomposed into MSB and LSB parts, as shown in Section 4.2.4. The pivot point x_0 should be chosen such that the error metric, e.g., the mean square error, between the original and the E-ANT kernel is minimized.

4.2.3 DPD via Piecewise Linear (PWL) Approximation

The PWL approximation can be employed when $f(x)$ ($x \in [x_l, x_u]$) is non-differentiable or the input dynamic range is large.

The PWL approximation employs $N+1$ points $(x_k, f(x_k))$ where $x_k = x_l + \frac{k}{N}(x_u - x_l)$ and $k = 0, 1, ..., N$ to approximate $f(x)$ as:

$$
f(x) \approx \sum_{k=1}^{N} p_k(x)
$$

$$
p_k(x) = \begin{cases} a_k x + b_k & x_k \leq x < x_{k+1} \\ 0 & otherwise \end{cases} \tag{4.27}
$$

where $x_0 = x_l$, $x_N = x_u$, $a_k = \frac{f(x_{k+1}) - f(x_k)}{x_{k+1} - x_k}$, and $b_k = \frac{x_{k+1} f(x_k) - x_k f(x_{k+1})}{x_{k+1} - x_k}$. Each segment $p_k(x)$ can be decomposed by noting that for a linear function $p(x)$, substituting for x from (4.20), we have

$$
p(x) = p(x_M + x_L 2^{-(B_{msb}-1)}) = p(x_M) + p(x_L) 2^{-(B_{msb}-1)} \tag{4.28}
$$

Therefore, substituting (4.20) into (4.27), we obtain:

$$p_k(x) = \begin{cases} p_{k,M}(x_M) + p_{k,L}(x_L)2^{-(B_{msb}-1)} & x_k \leq x < x_{k+1} \\ 0 & otherwise \end{cases} \tag{4.29}$$

where

$$\begin{aligned} p_{k,M}(x_M) &= a_k x_M + b_k \\ p_{k,L}(x_L) &= a_k x_L \end{aligned}$$

and $p_{k,M}(x_M)$ can be employed as the **E-**block.

Note that other piecewise approximation methods such as spline interpolation [108] where each segment is approximated with a low order polynomial can also be employed for DPD. Each low order polynomial can be decomposed in a manner similar to (4.24).

Next, we apply DPD to obtain E-ANT architectures for arithmetic units and compute kernels commonly used in signal processing and machine learning.

4.2.4 E-ANT Arithmetic Unit Architectures

4.2.4.1 E-ANT Adder

The output of a two-operand adder is given by:

$$y_a = x_1 + x_2$$

where x_1 and x_2 are the input operands. We first decompose the operands into MSB and LSB components according to (4.20):

$$\begin{aligned} x_1 &= x_{1M} + x_{1L}2^{-(B_{msb}-1)} \\ x_2 &= x_{2M} + x_{2L}2^{-(B_{msb}-1)} \end{aligned}$$

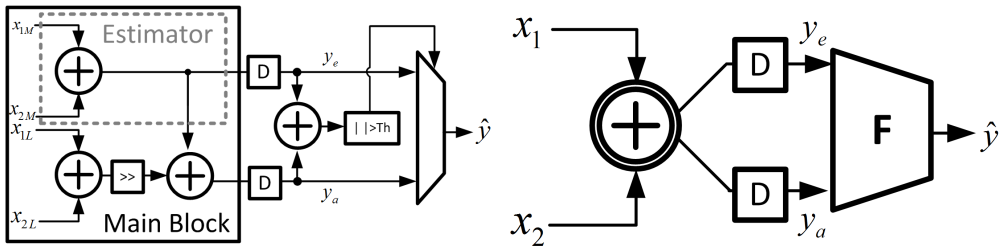


Figure 4.4: E-ANT Adder: (a) DFG, and (b) symbol.

where x_{iM} and x_{iL} are defined in (4.21) and (4.22), respectively.

Since addition is a linear function, DPD can be easily obtained from (4.28) as follows:

$$\begin{aligned} y_a &= x_{1M} + x_{2M} + x_{1L}2^{-(B_{msb}-1)} + x_{2L}2^{-(B_{msb}-1)} \\ &= f_M + f_L 2^{-(B_{msb}-1)} \end{aligned} \tag{4.30}$$

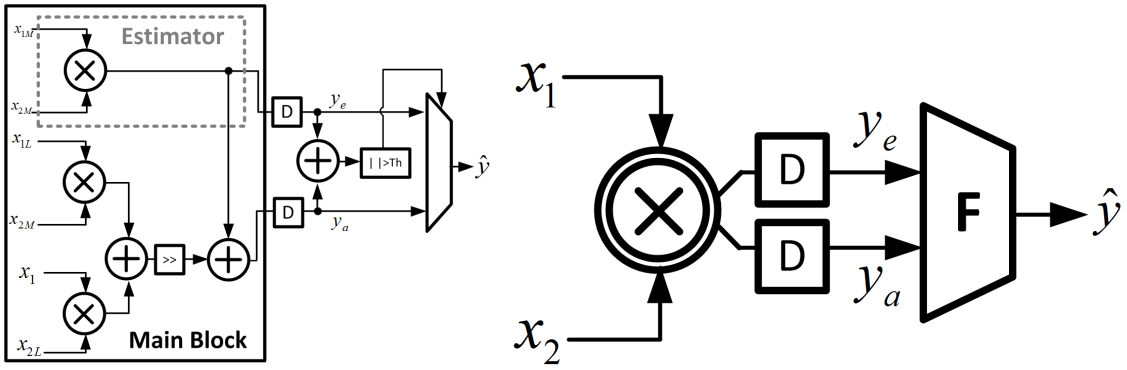
where $f_M = x_{1M} + x_{2M}$ and $f_L = x_{1L} + x_{2L}$. The data flow graph (DFG) and the symbol of the E-ANT adder are shown in Fig. 4.4(a) and Fig. 4.4(b), respectively.

4.2.4.2 E-ANT Multiplier

Employing the DPD in (4.21)-(4.23), the E-ANT multiplier can be derived as follows:

$$\begin{aligned} y_a &= x_1 x_2 \\ &= (x_{1M} + x_{1L}2^{-(B_{msb}-1)})(x_{2M} + x_{2L}2^{-(B_{msb}-1)}) \\ &= f_M + f_L 2^{-(B_{msb}-1)} \end{aligned} \tag{4.31}$$

where $f_M = x_{1M}x_{2M}$ and $f_L = x_{1L}x_{2M} + x_1 x_{2L}$. Figure 4.5(a) and Fig. 4.5(b) show the DFG and symbol of the E-ANT multiplier.



(a) (b)

Figure 4.5: E-ANT multiplier: (a) DFG, and (b) symbol.

4.2.4.3 E-ANT Multiply-accumulator (MAC)

MAC operation is described as:

$$y_a[n] = x[n]w[n] + y_a[n-1] \tag{4.32}$$

We first decompose $x[n]$, $w[n]$ and $y[n-1]$ according to (4.20):

$$x[n] = x_M[n] + x_L[n]2^{-(B_{msb}-1)} \tag{4.33}$$

$$w[n] = w_M[n] + w_L[n]2^{-(B_{msb}-1)} \tag{4.34}$$

$$y_a[n-1] = y_{a,M}[n-1] + y_{a,L}[n-1]2^{-2(B_{msb}-1)} \tag{4.35}$$

The E-ANT MAC can be obtained by substituting (4.33)-(4.35) into (4.32), and employing (4.30)-(4.31) to decompose $y_a[n]$ as follows:

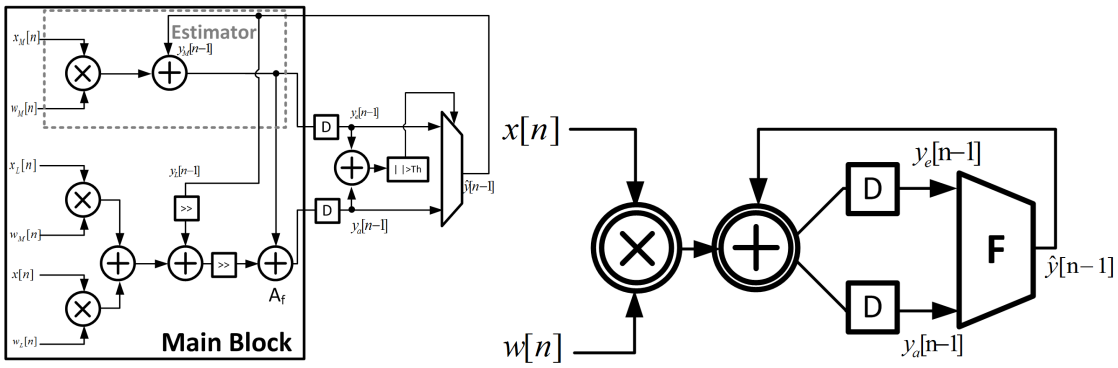


Figure 4.6: E-ANT MAC unit: (a) DFG, and (b) symbol.

$$
\begin{aligned}
y_a[n] &= x_M[n]w_M[n] + y_{a,M}[n-1] + (x_L[n]w_M[n] + (x_M[n] + x_L[n]2^{-(B_{msb-1})})w_L[n] \\
&\quad + y_{a,L}[n-1]2^{-(B_{msb}-1)})2^{-(B_{msb}-1)} \\
&= x_M[n]w_M[n] + y_{a,M}[n-1] + (x_L[n]w_M[n] + x[n]w_L[n] \\
&\quad + y_{a,L}[n-1]2^{-(B_{msb}-1)})2^{-(B_{msb}-1)} \\
&= f_M + f_L 2^{-(B_{msb}-1)} + y_{a,L}[n-1]2^{-(B_{msb}-1)})2^{-(B_{msb}-1)}
\end{aligned}
$$

where $f_M = x_M[n]w_M[n] + y_{a,M}[n-1]$ and $f_L = x_L[n]w_M[n] + x[n]w_L[n] + y_{a,L}[n-1]2^{-(B_{msb}-1)}$. Figure 4.6(a) and Fig. 4.6(b) show the DFG and the symbol of the E-ANT MAC.

4.2.5 E-ANT Signal Processing and Machine Learning Kernels

Complex E-ANT kernels can be derived by employing the E-ANT arithmetic units derived in section 4.2.4.

4.2.5.1 E-ANT FIR Filter

One of the most important kernels in information processing is filtering/convolution. We can derive an E-ANT FIR filter by employing (4.30)-(4.31) as follows:

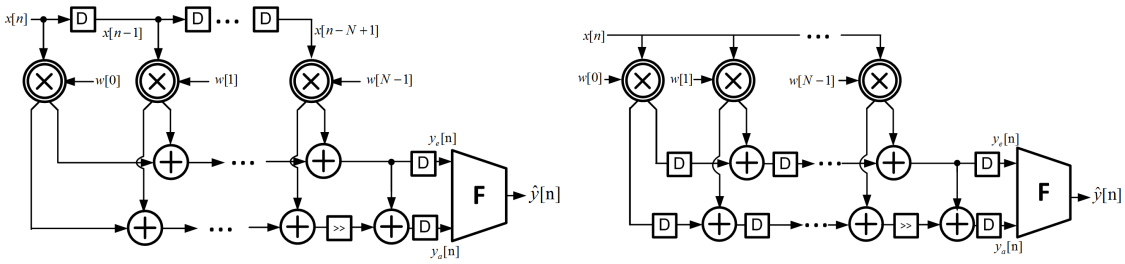


Figure 4.7: E-ANT FIR filter: (a) the DFG of direct form FIR filter, and (b) the DFG of transposed form FIR filter.

$$y_a[n] = \sum_{i=0}^{N-1} w[i]x[n-i] = \sum_{i=0}^{N-1} f_{iM} + \sum_{i} f_{iL} 2^{-(B_{msb}-1)}$$

where $f_{iM} = x_M[n-i]w_M[i]$ and $f_{iL} = x_L[n-i]w_M[i] + x[n-i]w_L[i]$ for $i = 0...N-1$. Figure 4.7 shows the DFGs of the direct form and transposed form E-ANT FIR filter where we make use of the symbols in Fig. 4.4(b), 4.5(b), and 4.6(b) to simplify the DFGs.

4.2.5.2 E-ANT Fast Fourier Transform (FFT) Butterfly Unit (BU)

BU is the main data processing unit in FFT processors. A general BU implements the following function:

$$\begin{aligned} &y_{1r,a} = x_{1r} + x_{2r}, y_{1i,a} = x_{1i} + x_{2i} \\ &d = x_1 - x_2 \\ &y_{2r,a} = d_r W_r - d_i W_i, y_{2i,a} = d_r W_i + d_i W_r \end{aligned}$$

where x_1 and x_2 are the inputs, y_1 and y_2 are the outputs, and W is the twiddle factor. The real and imaginary parts are denoted by r and i subscripts, respectively. E-ANT FFT BU can be derived as shown in Table 4.1.

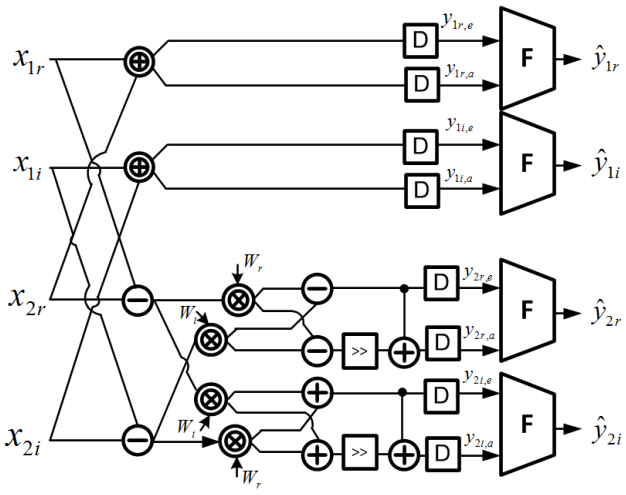


Figure 4.8: DFG of the E-ANT FFT butterfly unit.

Table 4.1: DPD for FFT BU

$y_{1r,a} = x_{1r} + x_{2r} = y_{1r,M} + y_{1r,L}2^{-(B_{msb}-1)}$,where $y_{1r,M} = x_{1r,M} + x_{2r,M}$ and $y_{1r,L} = x_{1r,L} + x_{2r,L}$.
$y_{1i,a} = x_{1i} + x_{2i} = y_{1i,M} + y_{1i,L}2^{-(B_{msb}-1)}$, where $y_{1i,M} = x_{1i,M} + x_{2i,M}$ and $y_{1i,L} = x_{1i,L} + x_{2i,L}$.
$y_{2r,a} = d_rW_r - d_iW_i = y_{2r,M} + y_{2r,L}2^{-(B_{msb}-1)}$, where $y_{2r,M} = d_{r,M}W_{r,M} - d_{i,M}W_{i,M}$ and $y_{2r,L} = d_{r,L}W_{r,M} + d_rW_{r,L} - (d_{i,L}W_{i,M} + d_iW_{i,L})$.
$y_{2i,a} = d_rW_i + d_iW_r = y_{2i,M} + y_{2i,L}2^{-(B_{msb}-1)}$, where $y_{2i,M} = d_{r,M}W_{i,M} + d_{i,M}W_{r,M}$ and $y_{2i,L} = d_{r,L}W_{i,M} + d_rW_{i,L} + d_{i,L}W_{r,M} + d_iW_{r,L}$

The DFG of the E-ANT FFT BU is shown in Fig. 4.8.

4.2.5.3 E-ANT Exponential Kernel

Exponential kernel (e^{-x}) is a critical component in many machine learning algorithms such as kernel SVM [105, 109], Gaussian mixture model [110], and others [111, 112]. Taylor expansion in Section 4.2.2 and PWL approximation in Section 4.2.3 can be employed to obtain E-ANT exponential kernels.

Assuming that the input dynamic range is scaled to [0,1], a 2^{nd} order Taylor expansion

leads to:

$$y_a = e^{-x} \approx e^{-x_0} - e^{-x_0} \times (x - x_0) + \frac{e^{-x_0}(x - x_0)^2}{2} \tag{4.36}$$

When $x_0 = 0.5$, (4.36) simplifies to

$$y_a \approx ax^2 + bx + c \tag{4.37}$$

where $a = 0.3033$, $b = -0.9098$, and $c = 0.9856$. The Taylor expansion based E-ANT exponential block can thus be derived from (4.24) as follows:

$$y_a \approx f_M + f_L 2^{-(B_{msb}-1)}$$

where

$$\begin{aligned} f_M &= ax_M^2 + bx_M + c \\ f_L &= a(2x_M x_L + x_L^2 2^{-(B_{msb}-1)}) + bx_L \end{aligned}$$

The DFG is shown in Fig. 4.9(a).

Alternatively, PWL approximation can be employed to obtain an E-ANT exponential kernel. Assume that two linear functions on $[0, 0.5]$ and $[0.5, 1]$ are used to approximate the exponential function on the interval $[0, 1]$; then according to (4.27):

$$y_a = e^{-x} \approx \begin{cases} a_1 x + b_1 & 0 \le x < 0.5 \\ a_2 x + b_2 & 0.5 \le x < 1 \end{cases}$$

where $a_1 = -0.7869$, $b_1 = 1$, $a_2 = -0.4773$ and $b_2 = 0.8452$. We first decompose a_i, b_i $(i = 1, 2)$ according to (4.20):

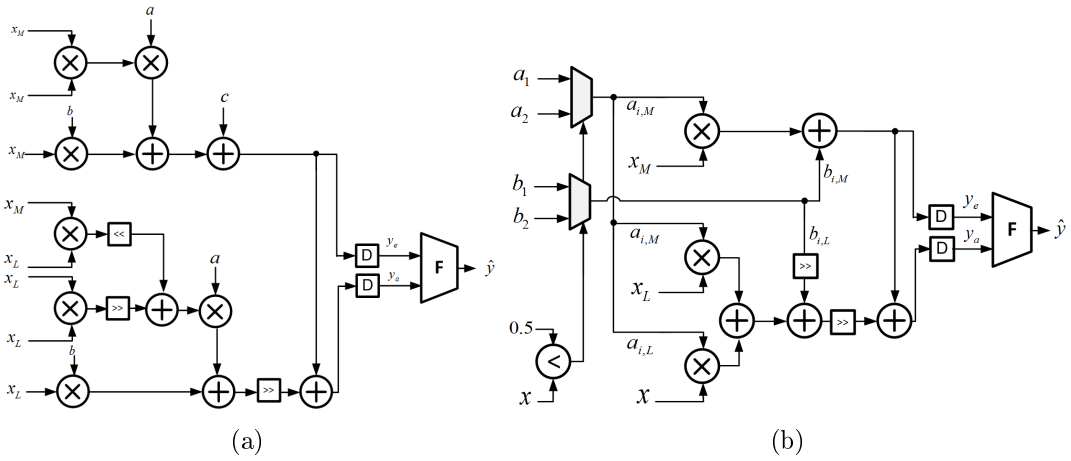


Figure 4.9: DFG of the E-ANT exponential kernel: (a) Taylor expansion based, and (b) PWL approximation based.

$$a_i = a_{i,M} + a_{i,L} 2^{-(B_{msb}-1)}$$

$$b_i = b_{i,M} + b_{i,L} 2^{-2(B_{msb}-1)}$$

Since each segment is linear, they can be decomposed by using (4.28):

$$y_{a,i} = a_i x + b_i = f_{i,M} + f_{i,L} 2^{-(B_{msb}-1)}$$

where

$$\begin{aligned} f_{i,M} &= a_{i,M} x_M + b_{i,M} \\ f_{i,L} &= a_{i,M} x_L + a_{i,L} x + b_{i,L} 2^{-(B_{msb}-1)} \end{aligned}$$

The resultant E-ANT exponential kernel has a reconfigurable architecture where different approximations are chosen according to the input values. The DFG of the PWL based E-ANT exponential kernel is shown in Fig. 4.9(b).

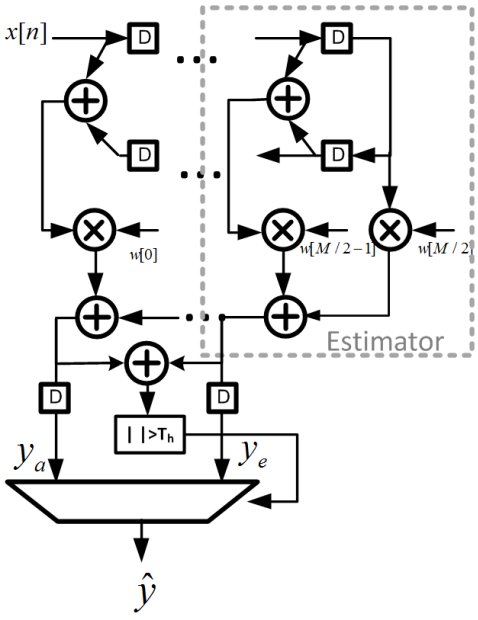


Figure 4.10: ALG-ANT FIR filter structure.

4.2.6 ALG-ANT

Low complexity estimation can also be achieved by changing the algorithm into an incremental refinement structure through algorithm transformation. Unlike architecture level techniques, ALG-ANT is algorithm specific. We next derive two ALG-ANT techniques - one for the FIR filter kernel and another for the dot product kernel.

4.2.7 ALG-ANT FIR Filter Kernel

The FIR filter is a commonly used kernel in signal processing and machine learning. The conventional FIR filter design method employs algorithms such as the weighted least square (WLS) method [113], which formulates the filter design as an optimization problem. Let $H(e^{j\omega})$ and $H_d(e^{j\omega})$ denote the designed and the ideal filter frequency responses, respectively, and $W(e^{j\omega})$ be a non-negative error weighting function. The WLS method minimizes the L_2 norm of the weighted difference between $H(e^{j\omega})$ and $H_d(e^{j\omega})$ as follows:

$$\min \frac{1}{2\pi} \int_{-\pi}^{\pi} [W(e^{j\omega})H(e^{j\omega}) - W(e^{j\omega})H_d(e^{j\omega})]^2 d\omega \tag{4.38}$$

Let $\mathbf{h} = [h[0], ..., h[M]]^T$, $\mathbf{d} = [d[0], ..., d[N-1]]^T$ be the pulse response of the $(M+1)$-tap filter $H(e^{j\omega})$ and the IDFT of $W(e^{j\omega})H_d(e^{jw})$, respectively, and let the N by $M+1$ matrix $\mathbf{W}$ be defined as $\mathbf{W}[n,l] = w[n-l]$ where $w[n]$ is the IDFT of $W(e^{j\omega})$. The optimization can be reduced to:

$$\min \|\mathbf{Wh} - \mathbf{d}\|^2 \tag{4.39}$$

and has solution $\mathbf{h}^* = \mathbf{W}^\dagger \mathbf{d}$, where $\mathbf{W}^\dagger$ is the Moore-Penrose pseudo inverse.

In ALG-ANT, the optimization in (4.39) is modified to include architectural level constraints. In particular, we employ the filter architecture in Fig. 4.10 where the center $M+1-2K_f$ filter taps are employed to obtain the estimator output $y_e[n]$ (see Fig. 4.2(a)). Here K_f is a design parameter that determines the estimator length. The rationale for using the center taps of an FIR filter to obtain an estimate of its final output $y_o[n]$ is that for linear phase FIR filter, the center taps of the filter can provide a good estimate of the filter response [114]. Doing so embeds the estimator completely into the main block. To achieve this, we reformulate the objective function in (4.39) as follows:

$$\min(1-\gamma)\|\mathbf{Wh} - \mathbf{d}\|^2 + \gamma \left\| \tilde{\mathbf{W}}\mathbf{h} \right\|^2 \tag{4.40}$$

where $\tilde{\mathbf{W}} = \begin{bmatrix} I_{K_f \times K_f} & 0_{K_f \times \hat{K_f}} & 0_{K_f \times K_f} \\ 0_{\hat{K_f} \times K_f} & 0_{\hat{K_f} \times \hat{K_f}} & 0_{\hat{K_f} \times K_f} \\ 0_{K_f \times K_f} & 0_{K_f \times \hat{K_f}} & I_{K_f \times K_f} \end{bmatrix}$, $\hat{K_f} = M + 1 - 2K_f$ and the parameter γ $(0 \leq \gamma < 1)$ is used to control the relative strength of the two optimization terms. Doing so constrains the magnitude of the outer taps of the filter. The optimization in (4.40) can be solved by setting the derivative of the loss function in (4.40) to zero, resulting in the following filter:

$$\mathbf{h}^* = ((1-\gamma)\mathbf{W}^T\mathbf{W} + \gamma\tilde{\mathbf{W}}^T\tilde{\mathbf{W}})^{-1}((1-\gamma)\mathbf{W}^T\mathbf{d}) \tag{4.41}$$

where $\mathbf{h}^*$ is the optimum ALG-ANT filter coefficients. In practice, γ and K_f are design parameters that can be employed to trade off the two optimization terms in (4.40). A large

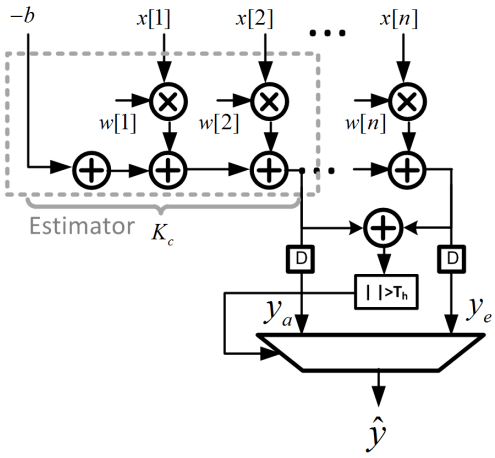


Figure 4.11: ALG-ANT linear SVM: the dot product result is unaltered when the order of the multiply-accumulates (MACs) is varied.

γ will weigh more on the estimator design, leading to a more accurate estimator. However, a large γ tends to decrease the performance of the main block since the resulting filter deviates from the ideal filter $\mathbf{d}$. A small K_f (thus larger estimator length) will lead to a more accurate estimator because more coefficients can be employed, but a small value of K_f will limit the amount by which VOS can be applied before the estimator begins to exhibit large magnitude timing violations. Thus, in this design, as expected, the accuracy of the estimator and the main block trade off with each other, and so does the extent of VOS that can be applied. In Section 4.3.6, K_f is determined by the error rate p_η (thus K_{vos}) and γ is optimized via a grid search.

4.2.8 ALG-ANT Dot Product Kernel

We next derive ALG-ANT for the dot product kernel, another widely used kernel in machine learning. The dot product kernel is employed in the linear SVM (see Fig. 4.3(a)), which provides good classification performance and results in a particularly simple architecture [115]. In the dot product kernel (see Fig. 4.11), the input vector $\mathbf{x}$ and the weight vector $\mathbf{w}$ are multiplied element-wise and the resulting products are added up.

One observation in (4.16) is that it is only the final dot product that contributes to the classification result, not the order in which computation is done. This suggests that we

can implement the dot product kernel via dimension reordering (DR) which will reorder the dimensions of the inputs $\mathbf{x}$ and $\mathbf{w}$ in the classification engine and use more important weights first during the dot product evaluation.

The reordered weight vector $\hat{\mathbf{w}}$ can be calculated via a simple sorting operation:

$$\hat{\mathbf{w}} = [\hat{w}_1, \hat{w}_2, ...\hat{w}_n]$$

where $|\hat{w}_i| \geq |\hat{w}_j|$ for $i < j$. In other words, we reorder the calculation of the dot product according to the importance of weights w_i in these dimensions. The resulting incremental refinement architecture enables us to employ the intermediate stage output as the estimator output $y_e[n]$, as shown in Fig. 4.11. As the estimator length K_c increases, the classification results will improve but the extent to which VOS can be applied will reduce. This trade-off is explored in the next section.

4.3 Simulation Results

This section presents the design optimization of the proposed ARCH-ANT and ALG-ANT technique, and shows the simulation results in a 45 nm CMOS process when they are applied to an SVM EEG classification system.

4.3.1 Methodology

Figure 4.12(a) shows the evaluation methodology employed to quantify system-level performance metrics and to estimate system-level energy consumption that integrates circuit, architecture, and system level design variables. The methodology consists of two parts: 1) *system-level error injection*, and 2) *system-level energy estimation*. Comparison of the proposed E-ANT with conventional approach (no error compensation) and retraining based approach in [54] is done using a commercial 45 nm CMOS process.

System-level error injection is done as follows:

1. Characterize delay vs. V_{dd} of basic gates such as AND and XOR using HSPICE for

$0.2\,\mathrm{V} \leq V_{dd} \leq 1.2\,\mathrm{V}$.

2. Develop structural Verilog HDL models of key kernels needed in the EEG classification system using the basic gates characterized in Step 1. These kernels are a 12 b input, 8 b coefficient, and 16 b output, 44-tap FIR filter (used in the FE) and a 8 b input, 8 b coefficient, and 19 b output vector-matrix multiplication kernel (used in the polynomial kernel SVM CE).

3. HDL simulations of these kernels were conducted at different voltages by including the appropriate voltage-specific delay numbers obtained in Step 1 into the HDL model. The error PMFs of these kernels and error rates p_η are obtained for different supply voltages (and thus voltage overscaling factor K_{vos}).

4. System performance evaluation and design optimization are done by injecting errors into a fixed point MATLAB-model of the EEG classification system. The errors are obtained by sampling the error PMFs obtained in Step 3.

Figure 4.12(c) shows the error PMF $P_\eta(\eta)$ of the 44-tap low pass FIR filter used in the FE at $V_{dd} = 0.9\,\mathrm{V}$ ($f_{clk} = 76\,\mathrm{MHz}$) which corresponds to a $K_{vos} = 0.75$ and an error rate $p_\eta = 0.05$, and Fig. 4.12(d) shows the error rate p_η increases from 10^{-5} at $V_{dd} = 1.15\,\mathrm{V}$ to 0.99 at $V_{dd} = 0.5\,\mathrm{V}$ as the voltage scales down.

System-level energy estimation is done as follows:

1. Obtain a full adder (FA) count N_{FA} of the kernel being analyzed.

2. Conduct a one-time characterization of the energy consumption of a FA incorporating both dynamic and leakage energies as follows:

$$E_{FA} = C_{FA}V_{dd}^2 + V_{dd}I_{leak}(V_{dd})\frac{1}{f_{clk}} \tag{4.42}$$

with

$$I_{leak}(V_{dd}) = \mu C_{ox}\frac{W}{L}(m-1)V_T^2 e^{\frac{-V_t}{mV_T}} e^{\frac{-\eta_d V_{dd}}{mV_T}}(1 - e^{\frac{-V_{dd}}{V_T}}) \tag{4.43}$$

where C_{FA} is the effective load capacitance of the FA and is extracted from HSPICE, V_{dd} is the supply voltage, V_t, V_T, μ, C_{ox}, and η_d are the threshold voltage, the thermal

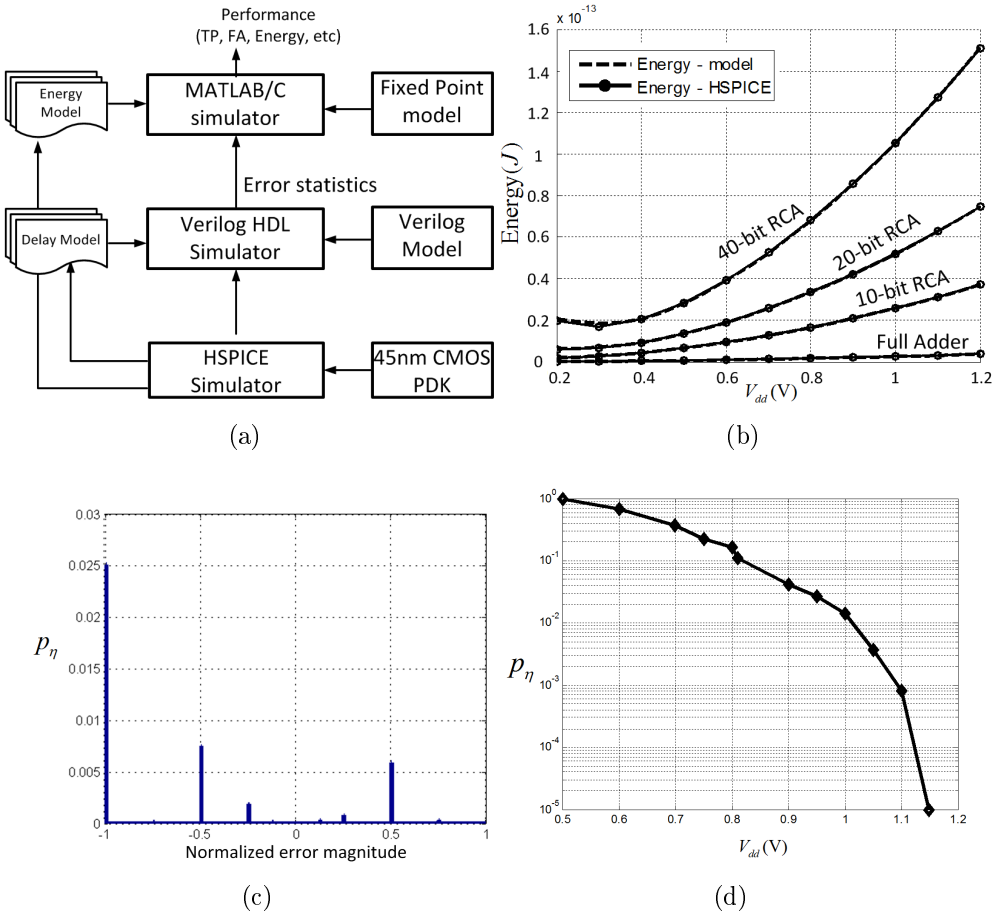


Figure 4.12: Evaluation methodology: (a) simulation setup, and (b) comparison of the energy model and HSPICE simulations in a 45 nm CMOS process, (c) error PMF at $V_{dd} = 0.9\,\text{V}$, and (d) error rate p_η vs. V_{dd} for the 44-tap low pass filter employed in the FE, the CHB-MIT EEG data set [54] is employed as input.

voltage, the carrier mobility, the gate capacitance per unit W/L, and the drain induced barrier lowering (DIBL) coefficient, respectively, obtained from the process files, and m is a constant related to the sub-threshold slope factor and is a fitting parameter.

3. The energy estimate of the kernel is obtained as $E_{op} = N_{FA}E_{FA}$.

Figure 4.12(b) shows the modeling results of the FA and ripple carry adder (RCA) for various bit widths demonstrating the accuracy and scalability of the energy model. The energy model is within 5% (for $0.2\,\text{V} \leq V_{dd} \leq 1.2\,\text{V}$) of circuit simulation results.

Algorithm 1 Energy optimization algorithm for E-ANT

1. Initialize $K^*_{vos} = 1$, $B^*_{msb} = 0$, E^*_{op} = energy of conventional MAC, MSE_{req} = specified MSE requirement.
2. $K_{vos} = K_{vos} - \Delta, B_{msb} = 0$. Obtain maximum E-block precision B_{max} to ensure error-free E-block operation.
3. $B_{msb} = B_{msb} + 1$. If $B_{msb} > B_{max}$, then exit, else compute MSE according to (23).
4. If $MSE < MSE_{req}$, then calculate energy $E(K_{vos})$ according to (21), else go to step 3
5. If $E^*_{op} > E(K_{vos})$, then $E^*_{op} = E(K_{vos})$, and $B^*_{msb} = B_{msb}$
6. Go to step 2

4.3.2 ARCH-ANT Design Optimization

The methodology in Fig. 4.12(a) is employed to perform optimization for the E-ANT kernels proposed in Sect. 4.2, and the E-ANT MAC kernel in Fig. 4.6(a) is used as an example. Since we adopt VOS to obtain different error rates, the parameters to be optimized are the voltage overscaling factor K_{vos} and the **E**-block bit width B_{msb}, where we assume that $B_{x,msb} = B_{w,msb} = B_{msb}$. The optimization framework is general enough to include the case when $B_{x,msb} \neq B_{w,msb}$. A grid search algorithm is employed to systematically determine the optimum setting K^*_{vos} and B^*_{msb} satisfying the performance metric, as shown in Algorithm 1 below. We adopt mean squared error (MSE) with respect to the floating point kernel as the performance metric:

$$MSE = E(\hat{y} - y_{fl})^2 \tag{4.44}$$

where $\hat{y}$ and y_{fl} indicate the E-ANT and floating point output, respectively. The maximum **E**-block length B_{max} under which the **E-**block does not make errors is determined by K_{vos}. The optimization routine gives the optimum E-ANT configuration, including K^*_{vos}, B^*_{msb} and minimum energy E^*_{op}, at the output.

Algorithm 1 is employed to optimize E-ANT MAC for 8b and 16b precision with MSE requirements of $10^{-2} \sim 10^{-5}$. The iso-MSE plots in the B_{msb} and p_η plane (see Fig. 4.13(a) and Fig. 4.13(c)) indicate that the optimum B_{msb} increases as the error rate p_η increases because a higher precision **E**-block is needed to compensate for the **M**-block errors. Figure 4.13(b) shows that the 8 bit E-ANT MAC achieves energy savings of 16% $\sim$ 69%, while as the 8 bit ANT MAC fails to achieve energy savings at the tight MSE requirement of

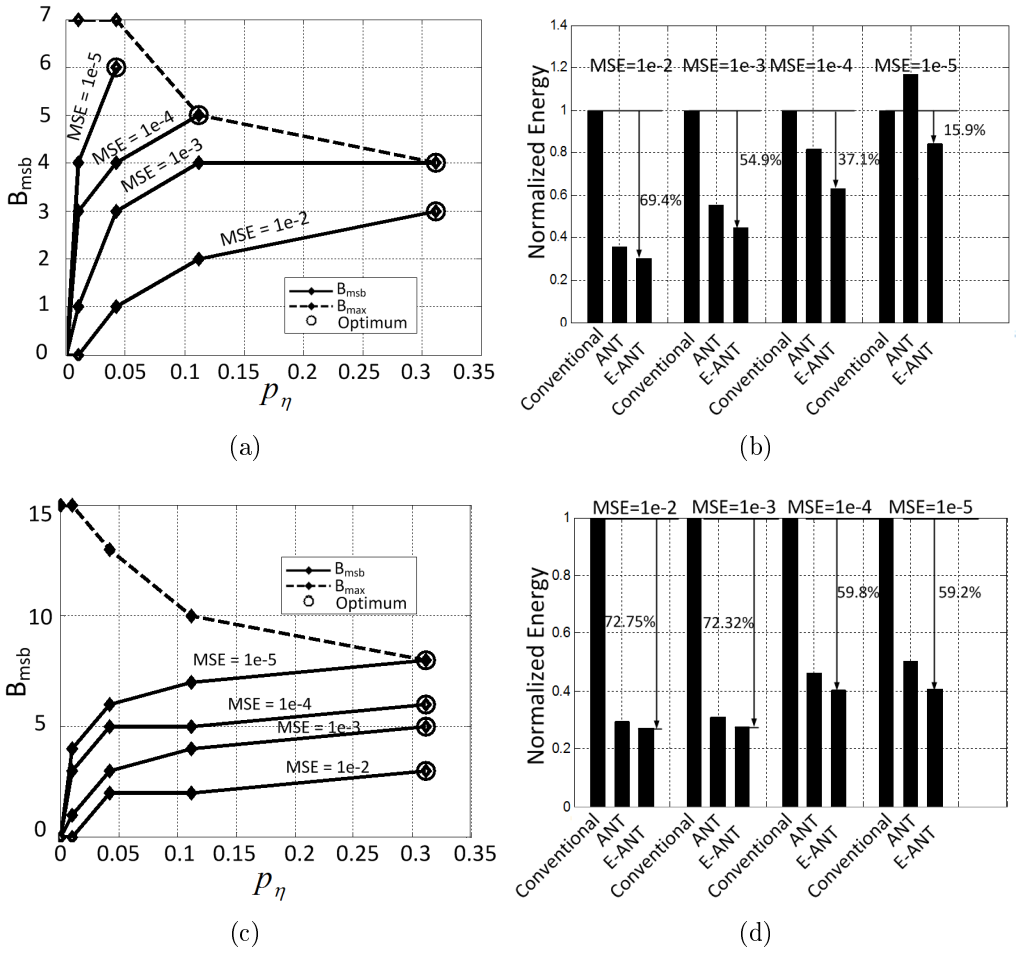


Figure 4.13: Optimization of E-ANT MAC; the dashed line illustrates that the maximum **E**-block precision B_{max} decreases as p_η increases, indicating that to ensure error-free **E**-block operation, the **E**-block bit width is upper bounded. The solid lines show the optimum B_{msb} configuration for each p_η at different MSE requirements, with the circle marker indicating the (B^*_{msb}, p^*_η) pair achieving the MSE requirements with minimum E_{op}: (a) optimization results of an 8×8 E-ANT MAC for different MSE requirements, (b) normalized energy of an 8×8 conventional MAC, ANT MAC and E-ANT MAC, (c) optimization results of a 16×16 E-ANT MAC for different MSE requirements, and (d) normalized energy of a 16×16 conventional MAC, ANT MAC, and E-ANT MAC.

10^{-5} due to **E**-block overheads. Figure 4.13(d) shows that the 16 bit E-ANT MAC achieves 59% energy savings compared with the conventional MAC. The overhead of the E-ANT architecture is below 8% compared with the 36.4% and 13.9% overheads for the 8 bit and 16 bit ANT architecture, respectively.

Figure 4.14 shows that the energy savings increase as the MSE requirement increases for a

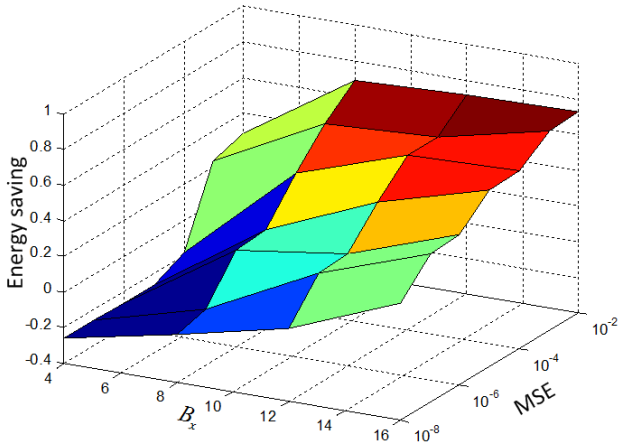


Figure 4.14: Energy savings vs. input precision and MSE.

fixed B_x. This is because a larger MSE requirement allows the MAC to operate at a higher p_η and can thus reduce the **E**-block overheads. This is also confirmed in Fig. 4.13(b) and Fig. 4.13(d). Additionally, the energy savings increase as B_x increases for a fixed MSE requirement because a large B_x tends to tolerate more LSB errors, thus enabling the MAC to operate at a higher p_η.

4.3.3 ARCH-ANT System Performance

To evaluate the performance of E-ANT, the filter kernel in the FE and the vector-matrix multiplication kernel in the SVM CE shown in Fig. 4.15 are implemented employing the E-ANT MAC as shown in Fig. 4.6, and are characterized via the procedure described in Section 4.3.2. For the filter bank in the FE, we use an input of 12 bit, with the MSB 8 bit taken as the **E**-block. For the CE, the input precisions of the two MACs are chosen to be 8 bit, and **E**-block precisions are 4 bit.

We employ the CHB-MIT EEG data set [54] to train the SVM and use leave-one-out cross validations to evaluate the system performance. The system performance metric employed is the true positive (TP) rate p_{tp} and false positive/alarm (FP) rate p_{fp}, defined as:

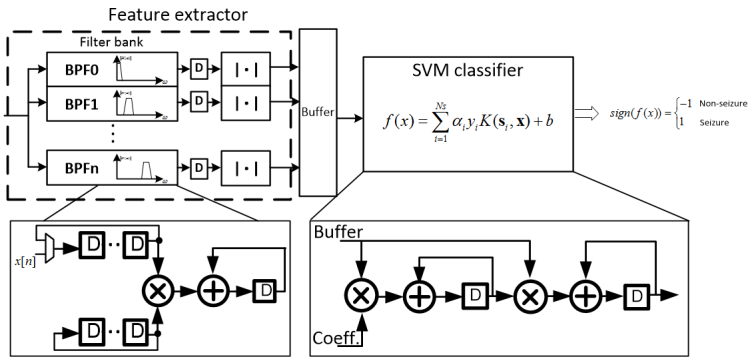
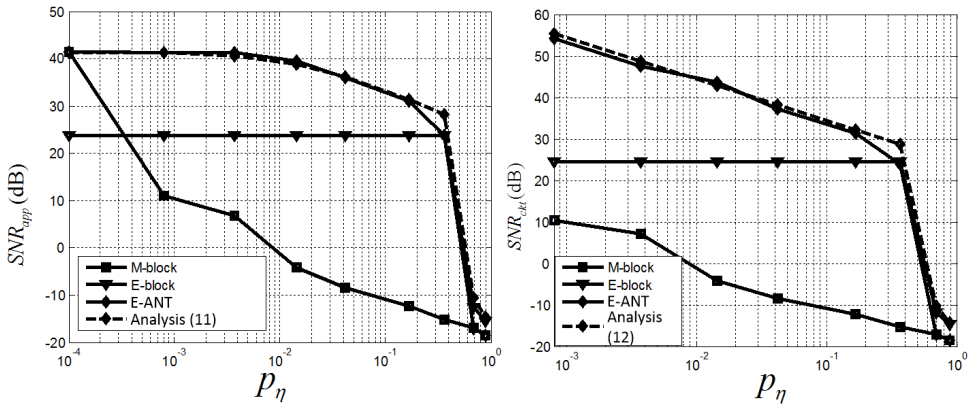


Figure 4.15: Second order polynomial kernel SVM EEG classification system architecture.

$$p_{tp} = \frac{TP}{TP+FN}$$
$$p_{fp} = \frac{FP}{FP+TN}$$

where TP, FN, FP, and TN are the number of true positives, false negatives, false positives, and true negatives, respectively. A good classifier achieves high values of p_{tp} (> 0.9) at a small constant false alarm rate p_{fp} (<0.01).

Three implementations are considered: the uncompensated system (denoted as CONV), the system which performs retraining with erroneous features, similar to the one proposed in [54] (denoted as RETRAIN), and the system with E-ANT (denoted as E-ANT). In the retraining method [54], the classifier is trained with features extracted in the presence of VOS errors. Unlike in the retraining method [54] where the CE needs to be error-free, E-ANT can tolerate errors in both the FE and CE. Therefore, two setups are considered in our experiment: (1) errors in FE only, and (2) errors in both FE and CE. The maximum value of the error rate p_η for which $p_{tp} > 0.9$ and $p_{fp} < 0.01$ is referred to as the error tolerance $p_{\eta-max}$ of the architecture. In the first setup, $p_{\eta-max}$ is the error rate in the FE, and in the second setup, $p_{\eta-max}$ is the maximum of the error rate in the FE and CE.



(a) (b)

Figure 4.16: SNR at the output of the FE: (a) application level SNR, and (b) circuit level SNR.

4.3.4 SNR Performance

Figure 4.16(a) shows that the improvement in the application level SNR (see (4.7)-(4.9)) achieved by E-ANT at the FE output is significant. In particular, the SNR of the **M-**block (also the SNR of the conventional system with errors), $SNR_{M,a}$, drops catastrophically from 42 dB to 10 dB for values of p_η as low as 8×10^{-4}. The SNR of the **E-**block, $SNR_{E,a}$, is constant at 23 dB for $p_\eta \leq 0.42$. This is because the **E-**block makes small magnitude estimation errors e. For $p_\eta > 0.42$, $SNR_{E,a}$ drops catastrophically as the **E-**block also starts to make large magnitude timing errors. In contrast, the ANT system SNR, $SNR_{ANT,a}$, is at least 10 dB higher than either $SNR_{E,a}$ or $SNR_{M,a}$ for values of p_η as high as 0.1, and approaches the **E-**block SNR as p_η increases.

The circuit level SNRs (see (4.10) - (4.12)) also exhibit a similar trend in Fig. 4.16(b). Furthermore, the SNR analysis in Section 4.1.1 is validated by plotting (4.13) and (4.14) in Fig. 4.16(a) and Fig. 4.16(b), respectively.

4.3.5 Classification Performance and Energy Savings

As shown in Fig. 4.17(a), p_{tp} drops sharply as circuit error rate increases in CONV system where no SEC is applied. The RETRAIN system does slightly better than the CONV system because the classifier is retrained to adapt to the error affected features. However, $p_{\eta-max}$ is

still only around 10^{-3}. This is due to the fact that the errors under investigation are timing errors. Unlike the stuck-at faults in [54], timing errors are dynamic and depend on the state of circuit, so the error pattern observed during training might not be the same as during the test. In contrast, when E-ANT is applied, p_{tp} degrades gracefully as p_η increases. As a result, the E-ANT system can achieve $p_{\eta-max}$ as high as 0.38. Figure 4.17(a) also shows that the p_{tp} is always lower than 0.9 when only **E**-block is employed, i.e., the **E**-block on its own is unable to meet the performance specifications. Similarly, when errors present in both FE and CE (Fig. 4.17(b)), both the CONV system and RETRAIN system achieve $p_{\eta-max} < 10^{-3}$, while E-ANT achieves a $p_{\eta-max}$ of 0.17. These are of 2 orders of magnitude (errors in FE only) and 3 orders of magnitude (errors in both FE and CE) greater than the existing systems. The receiver operating characteristic (ROC) curve at $p_{\eta-max}$ is shown in Fig. 4.17(c) when errors are in FE only, and in Fig. 4.17(d) when errors are in both FE and CE. In both experiments, the ROC of the CONV as well as the RETRAIN system approaches the ROC of a random classifier which outputs ± 1 with equal probability, while the ROC of the E-ANT system (w/ or w/o retraining) remains close to the ROC of an ideal classifier.

Principle component analysis (PCA) is performed on the feature vectors to understand the reason why CONV system fails but E-ANT system is able to maintain good performance. Figure 4.17(e) shows that when no SEC is applied, circuit errors have two effects on the feature vectors: (1) errors make it harder to separate the positive and negative samples, and (2) the entire feature space is shifted due to the accumulation block in the FE. The SVM fails to correctly perform classification without knowledge of the error statistics. Figure 4.17(f) shows that the large magnitude errors are compensated and converted to small residual errors when E-ANT is applied. This will cause a very small shift in the feature space. As a result, the SVM classifier can still perform correct classification. One way to improve E-ANT further is to incorporate retraining. In this method, the classifier is trained employing features that are subject to residual errors after the correction via E-ANT. However, as shown in Fig. 4.17(a), the improvement is minor due to the fact that the residual errors are typically small.

Table 4.2 compares the $p_{\eta-max}$, FE energy/feature (E_F), and CE energy/decision (E_C)

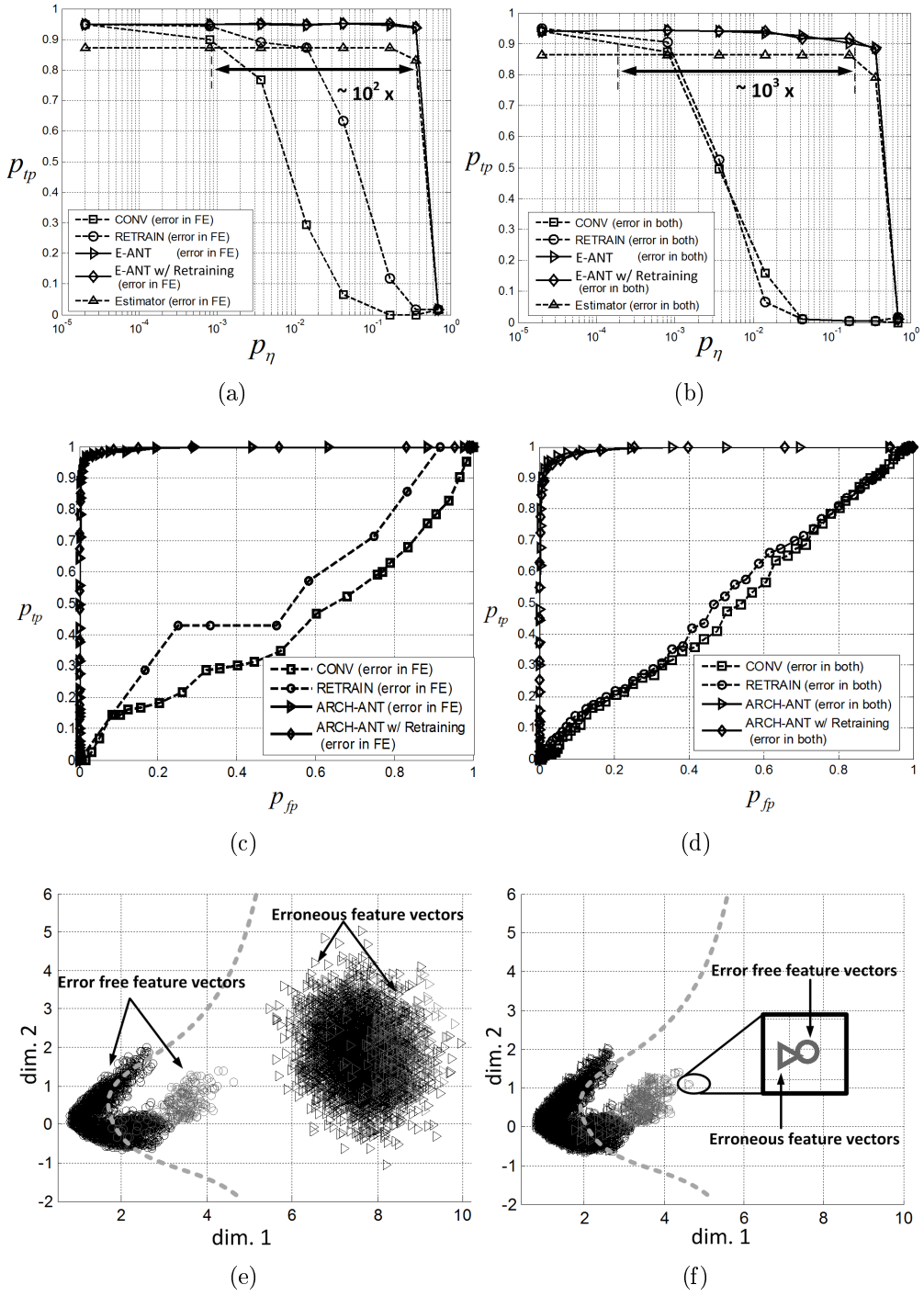


Figure 4.17: Simulation results: (a) p_{tp} of CONV, RETRAIN and E-ANT with $p_{fp} = 0.01$ when errors are in FE only, (b) p_{tp} of CONV, RETRAIN and E-ANT with $p_{fp} = 0.01$ when errors are in both FE and CE, (c) ROC curve of CONV, RETRAIN and E-ANT at $p_{\eta-max}$ when errors are in FE only, (d) ROC curve of CONV, RETRAIN and E-ANT at $p_{\eta-max}$ when errors are in both FE and CE, (e) PCA results of error-free and erroneous features for CONV, and (f) PCA results of error-free and erroneous features for E-ANT.

of the three systems. The E-ANT system can achieve $p_{\eta-max}$ of 0.38 when errors are in FE only, and 0.17 when errors are in both FE and CE. When VOS is applied for energy saving, the E-ANT system is able to achieve 51% energy savings when errors are in FE only compared with the CONV system. When both FE and CE are in error, the E-ANT system is able to achieve 43% and 29% energy savings in the FE and CE, respectively.

4.3.6 ALG-ANT Design Optimization

The FE and CE in an ALG-ANT based system have a number of design parameters that need to be selected for optimal system performance. For the FE, (4.40) indicates that K_f determines the estimator complexity. Hence, K_f places a lower bound on the supply voltage because the estimator needs to be free of timing violations. Similarly, γ indicates how closely the main block approximates the ideal frequency response. Thus, the accuracies of the estimator and the main block trade off with each other, which suggests that an optimum value for γ and K_f, i.e., γ^* and K_f^*, exists. To explore the trade-off between main block and estimator performance, the application level ALG-ANT filter SNR is defined. Let y_o, $\hat{y}$ denote the error-free main filter output and ALG-ANT filter output, and let y_d denote the error-free ideal filter (with coefficient $\mathbf{d}$) output. The ALG-ANT filter SNR is defined as

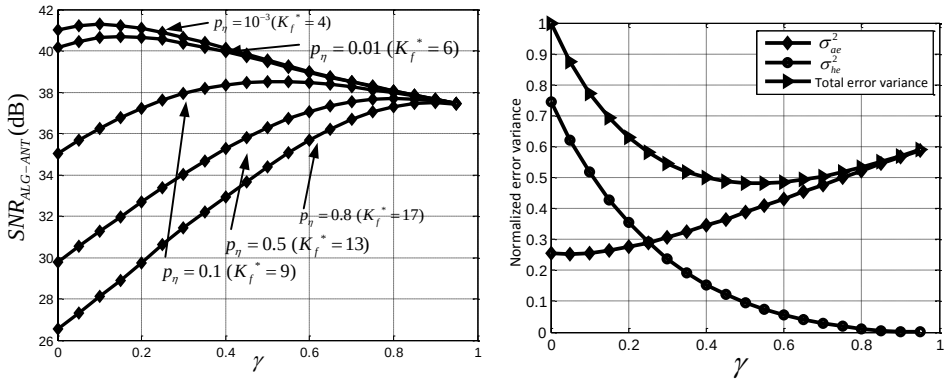
$$SNR_{ALG-ANT} = 10log_{10}(\frac{\sigma_{y_o}^2}{\sigma_{ae}^2 + \sigma_{he}^2}) \tag{4.45}$$

where $\sigma_{ae}^2 = E(y_o - y_d)^2$ is the variance of approximation error and $\sigma_{he}^2 = E(y_o - \hat{y})^2$ is the variance of hardware error.

In order to determine these SNR-optimum values, K_f^* is first determined by choosing the maximum estimator length at a given supply voltage V_{dd}, and hence error rate p_η (thus K_{vos}). In particular, K_f^* increases with p_η as shown in Fig. 4.18(a). Next, γ^* is obtained via sweeping its value and observing the $SNR_{ALG-ANT}$. Figure 4.18(a) shows that when p_η is low, i.e. K_f^* is small, γ^* is small because the approximation error σ_{ae}^2 dominates. On the other hand, when p_η is high, i.e., K_f^* is large, γ^* is large because the hardware error σ_{he}^2 dominates, and the optimization procedure will strive for a more accurate estimator, as

Table 4.2: ARCH-ANT Performance and Energy Comparison

	Errors in FE only		**Errors in both FE and CE**		
	$p_{\eta-max}$	Energy savings in FE	$p_{\eta-max}$	Energy savings in FE	Energy savings in CE
Conventional	8×10^{-4}	NA	2×10^{-4}	NA	NA
Error resilient retraining	3×10^{-3}	7%	8×10^{-4}	13%	12%
E-ANT	0.38	51%	0.17	43%	29%



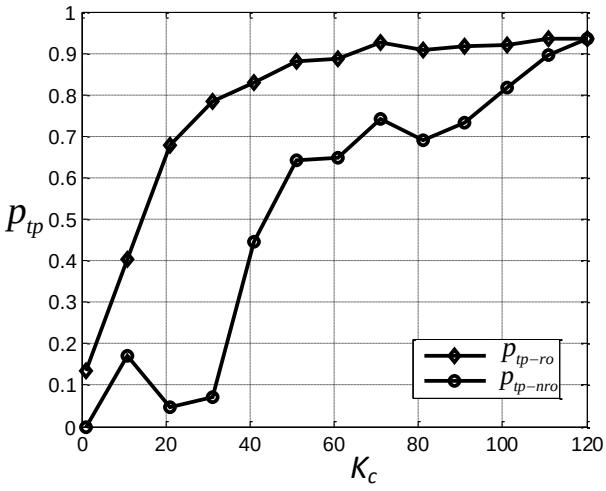
(a) (b)

Figure 4.18: ALG-ANT applied to the filter design problem: (a) $SNR_{ALG-ANT}$ vs. γ for various p_η, and (b) approximation error σ^2_{ae}, hardware error σ^2_{he} and total error $(\sigma^2_{ae} + \sigma^2_{he})$ vs. γ at $p_\eta = 0.1$, the CHB-MIT EEG data set [54] is employed as input, error variances are normalized w.r.t. total error variance at $\gamma = 0$.

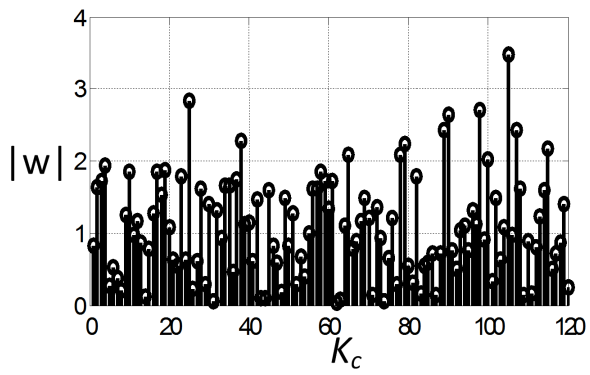
shown in (4.40). Figure 4.18(b) shows this trade-off for a specific value of p_η, where it can be seen that as γ increases, σ^2_{ae} increases because the overall filter no longer minimizes the difference between ideal filter and main block; at the same time, σ^2_{he} decreases because the estimator gives better approximations.

For the linear SVM, DR is applied. We employ the CHB-MIT EEG data set [54] to train the SVM and use leave-one-out cross validations to evaluate the classifier performance. The system performance metric employed is the true positive (TP) rate p_{tp} and false positive/alarm (FP) rate p_{fp}, as defined in Sect. 4.3.3.

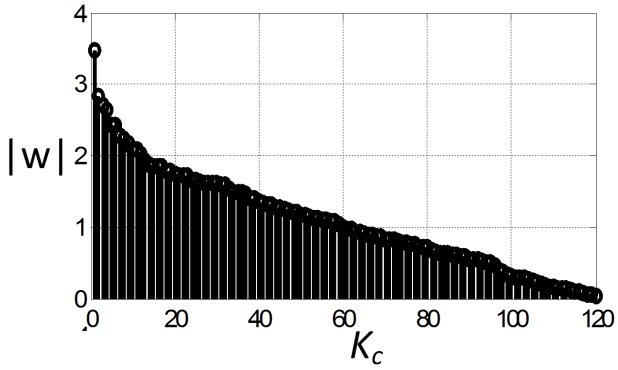
Figure 4.19 studies the impact of DR in the SVM classifier in an error-free condition and a $p_{fp} \leq 0.01$. It indicates that the TP rate in the absence of DR (p_{tp-nro}) increases non-monotonically with K_c (the estimator complexity). In particular, $p_{tp-nro} \leq 0.5$ for $K_c \leq 45$, and $p_{tp-nro} \geq 0.9$ only when $K_c \geq 112$. In contrast, when DR is employed the TP rate p_{tp-ro} increases monotonically with K_c, and $p_{tp-ro} \geq 0.9$ when $K_c \geq 64$, which is 43% smaller than when DR is not used. Note that DR needs to be performed only once during the training and thus does not incur overhead during classification. Figure 4.19(b) shows that without DR, the large magnitude weights are scattered across the dimensions, leading to poor classification results unless the value of K_c is sufficiently large. DR uses the important weights first (see Fig. 4.19(c)), and thus can produce acceptable results with much smaller



(a)



(b)



(c)

Figure 4.19: Comparison of classification results with and without DR; feature vectors extracted with FE are employed as input: (a) p_{tp} (with $p_{fp} \leq 0.01$) of the SVM classifier vs. estimator length K_c where the estimator is directly obtained by using the first K_c taps of the dot product kernel; the results with DR are denoted as p_{tp-ro}, while the results of directly using the reduced dimension classifier is denoted as p_{tp-nro}. (b) The weights without DR. (c) The weights with DR.

values of K_c.

4.3.7 ALG-ANT System Performance

The system architecture of the SVM EEG classification system is shown in Fig. 4.20 where the feature extractor employs the design parameters from [54, 55], with an input of 12 b for the filter bank and 8 b bit for the SVM classifier. Three architectures are considered:

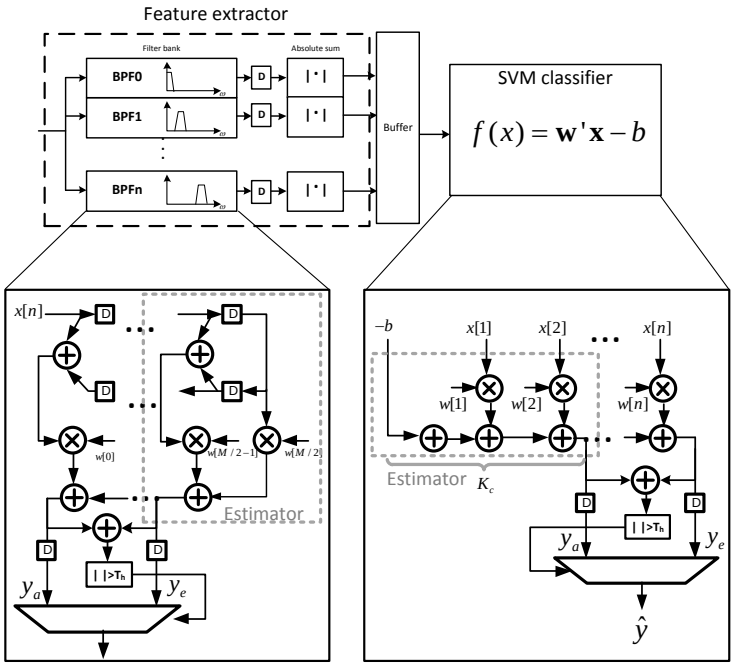


Figure 4.20: ALG-ANT based SVM EEG classification system architecture.

the conventional classifier (denoted as CONV), the classifier with retraining [54] (denoted as RETRAIN), and the classifier with ALG-ANT (denoted as ALG-ANT). In the retraining method [54], the classifier is trained with features extracted in the presence of VOS errors. Unlike in the retraining method [54] where CE needs to be error free, ALG-ANT can tolerate errors in both FE and CE. Therefore, two setups are considered in our experiment: (1) errors in FE only and (2) errors in both FE and CE. The max value of error rate p_η for which $p_{tp} > 0.9$ and $p_{fp} < 0.01$ is referred to as the error tolerance metric $p_{\eta-max}$ of the architecture. In the first setup, $p_{\eta-max}$ is the error rate in the FE, and in the second setup, $p_{\eta-max}$ is the maximum of the error rate in FE and CE.

Figure 4.21(a) shows that when errors are in FE only, p_{tp} for the conventional system drops sharply and $p_{\eta-max}$ is as low as 1.5×10^{-4}. Retraining does slightly better as the classifier is retrained to adapt to the error affected features. However, the error tolerance $p_{\eta-max}$ is below 10^{-3}. This is most likely due to the fact that unlike stuck-at faults studied in [54], timing errors due to VOS are dynamic and depend on the state of circuit. In contrast, when ALG-ANT is applied, p_{tp} has a graceful degradation as p_η increases and $p_{\eta-max}$ is improved to 0.41. The performance of the conventional ANT system was found to be similar to the

ALG-ANT system, and thus is not shown. Figure 4.21(a) also shows that the p_{tp} is always lower than 0.9 when only estimator is employed. Similarly, when errors present in both FE and CE (see Fig. 4.21(b)), ALG-ANT classifier achieves $p_{\eta-max} = 0.19$. These are both 3 orders of magnitude greater than the existing systems.

Principle component analysis (PCA) is performed on the feature vectors to understand the reason why the conventional system fails and ALG-ANT is able to maintain good performance. Figure 4.21(c) shows that in the conventional system, circuit errors have two effects on the feature vectors: (1) errors make it harder to separate the positive and negative samples, and (2) the entire feature space is shifted. The SVM fails to correctly perform classification without knowledge of the error statistics. Figure 4.21(d) shows the large magnitude error is compensated and converted to small residual errors when ALG-ANT is applied, which will cause a very small shift in the feature space. As a result, the SVM classifier can still perform correct classification.

Table 4.3 compares the error tolerance $p_{\eta-max}$, feature extraction energy/feature (E_F), and classification energy/decision (E_C) of three classifiers. When VOS is applied for energy savings, compared with the conventional classifier, the ALG-ANT classifier is able to achieve 44.3% energy savings when errors are in FE only. When both FE and CE are in error, the ALG-ANT classifier is able to achieve 37.1% and 36.9% energy savings in the FE and CE, respectively. The energy savings are due to: (1) the elimination of an explicit estimator, and (2) the scaling of supply voltage.

4.4 Conclusions

In this chapter, we propose E-ANT, where the estimator is embedded into the main block via proper architecture and algorithm level transforms, resulting in a low overhead architecture with the same error compensation functionality. At the architecture level, ARCH-ANT uses data path decomposition to embed a reduced precision replica estimator into the main block. The data path decomposition is general and can be derived for a wide class of compute kernels. At the algorithm level ALG-ANT employs additional optimization constraints during the algorithm to architecture mapping to embed the estimator into the main block. The

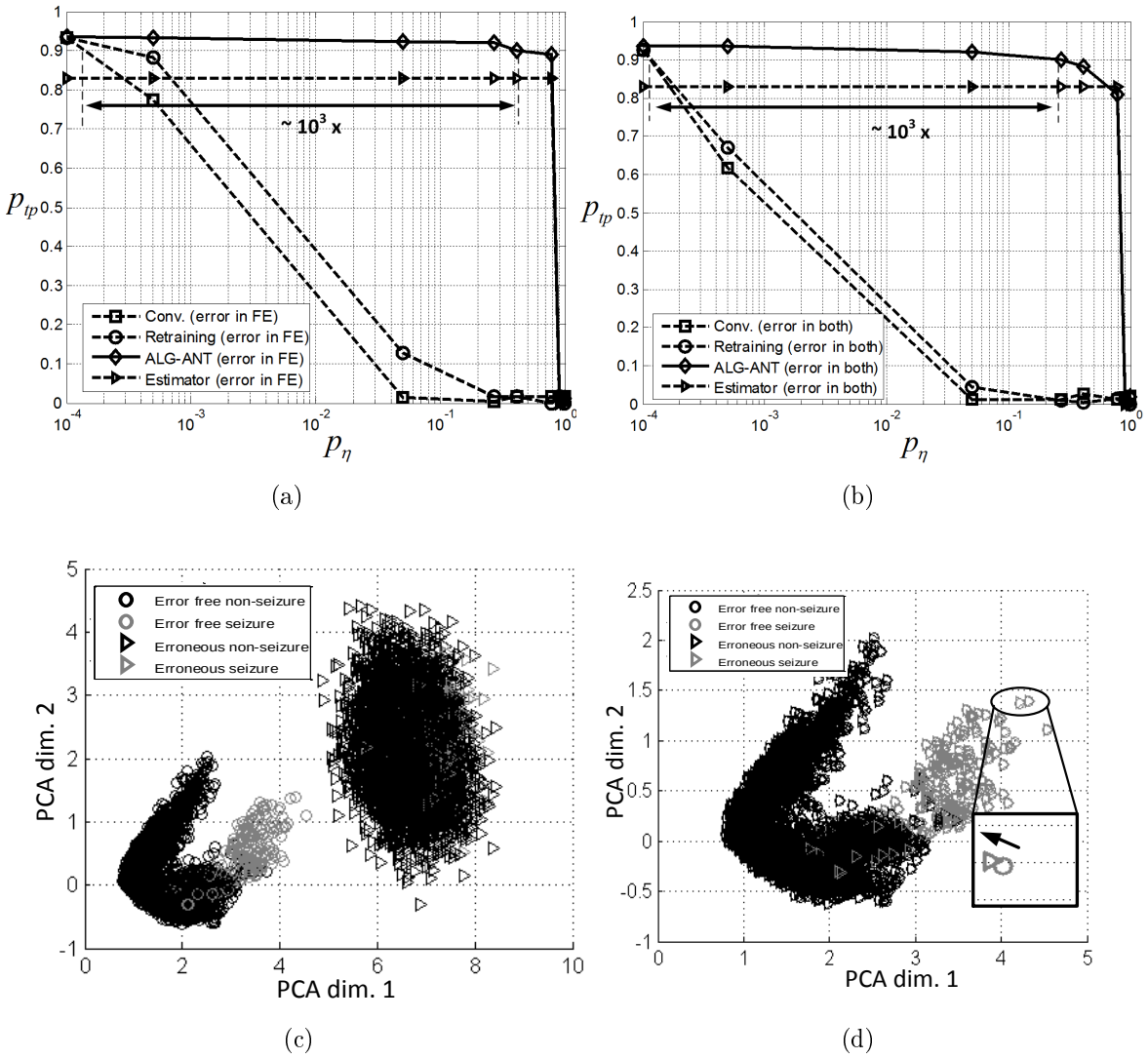


Figure 4.21: Simulation results: (a) p_{tp} of conventional, retraining, and ALG-ANT classifier with $p_{fp} \leq 0.01$ when errors are in feature extractor only, (b) p_{tp} of conventional, retraining, and ALG-ANT system with $p_{fp} \leq 0.01$ when errors are in both the feature extractor and the classifier, (c) PCA results of error free and erroneous features for conventional classifier, and (d) PCA results of error free and erroneous features for ALG-ANT classifier.

result is a single architecture that can be used to obtain both the estimator and main block outputs as in ANT systems. The effectiveness of the proposed ARCH-ANT and ALG-ANT technique has been demonstrated through the design of a SVM EEG seizure classification system where simulation results in a commercial 45 nm CMOS process show that ARCH-ANT achieves up to 38% error tolerance and up to 50.6% energy savings compared with an uncompensated system. ALG-ANT achieves up to 41% error tolerance and up to 44.3%

energy savings compared with uncompensated system.

This work shows that by exploring architectural and algorithmic level transforms, it is possible to design architectures that are inherently error resilient without explicit estimator blocks. It opens up a few research directions to extend or generalize existing SEC techniques. In particular, E-ANT techniques for other SEC techniques such as SSNOC [63] and soft-NMR [61] can be derived. Moreover, with the adoption of near/sub-threshold voltage design and continued scaling of the CMOS process, PVT-induced and defect-induced errors are becoming a growing concern for the design of ULP platforms. E-ANT techniques, and in general SEC techniques, can be applied in the near-threshold region to enhance system robustness in the presence of these new error models.

Table 4.3: ALG-ANT Performance and Energy Comparison

	Errors in FE only		**Errors in both FE and CE**		
	$p_{\eta-max}$	Energy Savings in FE	$p_{\eta-max}$	Energy Savings in FE	Energy Savings in CE
Conventional	1.5×10^{-4}	NA	10^{-4}	NA	NA
Retraining	3×10^{-4}	5.1%	10^{-4}	0	0
ALG-ANT	0.41	44.3%	0.19	37.1%	36.9%

Chapter 5

PROBABILISTIC ERROR MODELS FOR MACHINE LEARNING KERNELS IMPLEMENTED ON STOCHASTIC NANOSCALE FABRICS

Systematic design of ML kernels on stochastic nanoscale fabrics requires one to efficiently predict the behavior of such implementations. For this, high-level error models of key ML building blocks need to be developed. The error models of such kernels need to capture the stochastic behavior of the underlying fabric such as voltage overscaling (VOS), process variations, and defects. Compact analytical models of kernel behavior in the presence of errors are very desirable as these can be employed to: (1) characterize the inherent error resiliency of ML algorithms, and (2) evaluate the effectiveness of error resiliency techniques in compensating for these errors.

The error behavior of computational kernels can be fully captured in terms of their joint probability mass functions (PMFs). To date, not much work has been done on this topic. Analytical models for logic errors [116], transient errors [117], and timing errors [118] have been proposed. These models focus on obtaining expressions for the error rate/magnitude. In approximate computing, theoretical models have been proposed to model the inaccuracy of circuits [119]. However, the models are architecture specific. Interval-based approaches (interval arithmetic or affine arithmetic) [120] have been proposed to model and propagate PMFs. These approaches need to store the entire error PMF. A lookup table based technique [121] has been proposed to characterize the statistical properties of approximate hardware. These models only capture the standard deviations of basic circuit building blocks rather than the error PMF. In signature analysis based testing, symmetrical error model [122] and independent error model [123] are employed to model the output error PMF. Additionally, in all the models for approximate computing, the errors are due to imprecise but deterministic circuits, not dynamic errors due to VOS and process variations.

In this chapter, we propose a probabilistic additive error model capable of modeling er-

rors due to sources such as VOS, process variations, and defects. Four different variants of the additive error model are studied: additive over **R**eals **E**rror **M**odel with **i**ndependent Bernoulli RVs (REM-i), additive over **R**eals **E**rror **M**odel with **j**oint Bernoulli RVs (REM-j), additive over **G**alois field **E**rror **M**odel with **i**ndependent Bernoulli RVs (GEM-i), and additive over **G**alois field **E**rror **M**odel with **j**oint Bernoulli RVs (GEM-j). Analytical expressions for the error PMFs are derived. Kernel level model validation is accomplished by comparing the Jensen-Shannon divergence D_{JS} between the modeled PMF and the PMFs obtained via HDL simulations in a commercial 45 nm CMOS process of MAC units used in a support vector machine (SVM) to classify the UCI machine learning dataset [124]. Results indicate that at the MAC unit level, D_{JS} for the GEM models are 2 orders of magnitude lower (better) than the REM models for VOS, and 1 order of magnitude lower for process variation errors. However, when considering errors due to defects, D_{JS} for REM-j is between 1 and 2 orders of magnitude lower than the others. Performance (probability of detection P_{det}) prediction of a 2^{nd} order polynomial SVM classifier is conducted using the proposed model and compared with HDL simulations. We find that P_{det} estimated using GEM-j is within 3% for VOS errors when the error rate $p_\eta \leq 80\%$, and within 5% for process variation errors when supply voltage V_{dd} is between 0.3 V and 0.7 V. In addition, P_{det} using REM-j is within 2% for defect errors when the defect rate (the percentage of circuit nets subject to stuck-at-faults) p_{saf} is between 10^{-3} and 0.2.

The rest of the chapter is organized as follows. Section 5.1 describes the framework for the error analysis and the distance measure employed to compare models. Section 5.2 presents the proposed models, and derives analytical expressions for the PMF of the errors. Section 5.3 presents the error characterization/simulation methodology, and model validation results at kernel level and system level. Conclusions are presented in Section 5.4.

5.1 Modeling Framework and Accuracy Measure

5.1.1 Error Modeling Framework

In this chapter, we employ capital letters and small letters to denote a RV Y and its realization y, respectively. The proposed error modeling framework (see Fig. 5.1) captures the spatio-temporal distribution of errors. This is required as certain error sources such as defect and process variations result in an error RV whose PMF is determined by the statistics of the input and the spatial distribution across physical instantiations of the computational block. The following notation is employed in this chapter: let I_k $(k = 1, 2..., M)$ denote the k^{th} instance of the system/kernel subject to errors, and let $x_k[n]$, $y_{o,k}[n]$, $\eta_k[n]$, and $y_{a,k}[n]$ denote the samples corresponding to the input, error free output, error, and the final output of I_k, with time index n, respectively.

5.1.2 Additive Error Models

For notational simplicity, we drop the index n and k. We consider the additive error model as shown in Fig. 5.1:

$$y_a = y_o \oplus_F \eta \tag{5.1}$$

where $\oplus_F$ denotes addition over field F, η is the error and is a realization of the RV N with PMF $P(\eta)$. The models (REM-i,j and GEM-i,j) proposed in this paper focus on modeling $P(\eta)$.

In REM-i,j, addition in (5.1) is taken over the field of reals $\mathbb{R}$. Thus, (5.1) can be written as

$$y_a = y_o + \eta \tag{5.2}$$

where y_a, y_o and η are reals expressed in the *2's complement form.* For example, η is written in the 2's complement form as:

$$\eta = -\eta_0^b + \sum_{i=1}^{B_\eta - 1} \eta_i^b 2^{-i} \tag{5.3}$$

and $\eta_i^b \in \{0, 1\}$ and B_η are the i^{th} bit and bit precision of η, respectively. The 2's complement form of y_a and y_o can be expressed similarly.

In GEM-i,j, addition in (5.1) is taken over the Galois field of 2 (GF(2)). Thus, (5.1) can be written as:

$$\mathbf{y}_a = \mathbf{y}_o \oplus \boldsymbol{\eta} \tag{5.4}$$

where $\mathbf{y}_a, \mathbf{y}_o$ and $\boldsymbol{\eta}$ are the bit vectors representing y_a, y_o and η, respectively, and $\oplus$ is the bitwise XOR operator. For example, $\boldsymbol{\eta}$ is given by the *vectorized form* as:

$$\boldsymbol{\eta} = [\eta_0^b, ...\eta_{B_\eta - 1}^b]^T \tag{5.5}$$

and is a realization of RV $\boldsymbol{N} = [N_0^b, N_1^b, ..., N_{B_\eta - 1}^b]$. The vectorized form of y_a and y_o can be expressed similarly. Note that the 2's complement form and the vectorized form of η are equivalent.

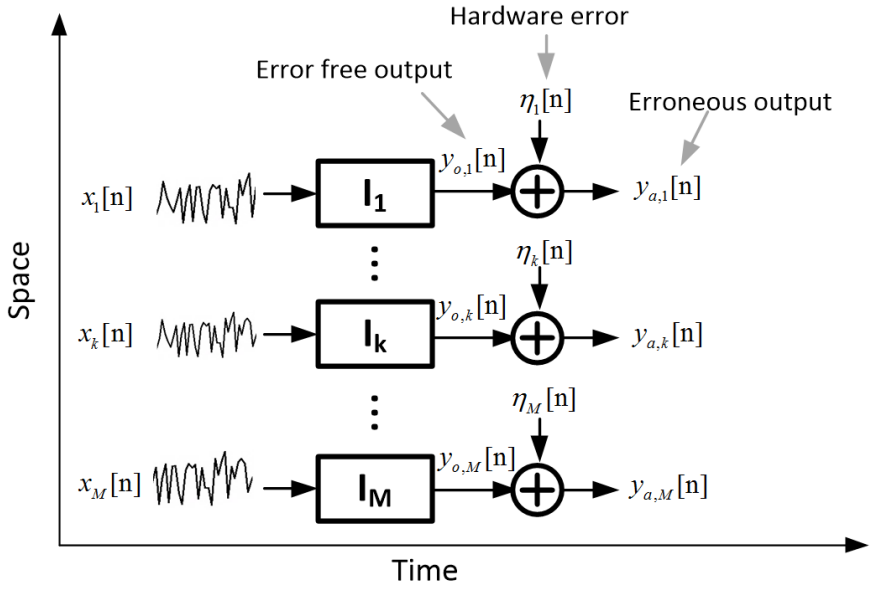


Figure 5.1: Error modeling framework.

5.1.3 Model Accuracy Metric

To quantify model accuracy, we employ the commonly employed Jensen-Shannon (JS) divergence D_{JS} [125] as the measure of the distance between two distributions. The JS divergence between two PMFs P and Q is defined as:

$$D_{JS}(P||Q) = \frac{1}{2}D_{KL}(P||M) + \frac{1}{2}D_{KL}(Q||M) \tag{5.6}$$

where $M(\cdot) = \frac{1}{2}P(\cdot) + \frac{1}{2}Q(\cdot)$, and D_{KL} is the Kullback–Leibler (KL) divergence defined as:

$$D_{KL}(P||Q) = \sum_i P(i) log_2 \left(\frac{P(i)}{Q(i)}\right) \tag{5.7}$$

The reason for choosing the JS divergence as the distance measure is that it is symmetric ($D_{JS}(P||Q) = D_{JS}(Q||P)$) and bounded ($0 \leq D_{JS} \leq 1$) [125], unlike KL divergence.

5.2 Error Model Derivation

The challenges in modeling error N lie in the fact that: (1) N is a discrete RV and is restricted to certain error magnitudes (especially when error rate is low), and (2) the PMF is not smooth. Instead of modeling the error magnitude directly, we propose to model the bits N_i^b of N as joint RVs. Four different variants of the additive error model are studied: REM-i, REM-j, GEM-i, and GEM-j.

5.2.1 REM-i: Additive over Real Error Model with Independent Bernoulli RVs

In REM-i (see (5.2)), N_i^b ($i = 0, 1..., B_\eta - 1$) is modeled as a Bernoulli RV so the PMF of N_i^b can be written as:

$$P_{N_i^b}(x) = \begin{cases} p_i & \text{if } x = 1 \\ 1 - p_i & \text{if } x = 0 \end{cases} \tag{5.8}$$

and N_i^b $(i = 0, 1..., B_\eta - 1)$ are assumed to be independent. Thus, under REM-i, the PMF $P(\eta)$ can be obtained from (5.8) as:

$$P(\eta) = \prod_{i=0}^{B_\eta - 1} p_i^{\eta_i^b} (1 - p_i)^{1 - \eta_i^b} \tag{5.9}$$

Statistical metrics such as the mean and variance of N can be easily derived from (5.3) and (5.9). The modeling complexity, defined as the number of parameters to be estimated, is $O(B_\eta)$ for REM-i, as shown in (5.9).

5.2.2 REM-j: Additive over Real Error Model with Joint Bernoulli RVs

The pairwise covariance between N_i^b and N_j^b can be included to improve the modeling accuracy. In REM-j, the PMF of N is parametrized by the mean vector $\boldsymbol{\mu}_\eta = [p_0, p_1, ..., p_{B_\eta - 1}]^T$ and the covariance matrix $\mathbf{C}_\eta$, where $\mathbf{C}_\eta(i, j) = cov(N_i^b, N_j^b)$ is the covariance between N_i^b and N_j^b for $i, j = 0, 1, ..., B_\eta - 1$.

The dichotomized Gaussian (DG) distribution [126] can be used to obtain the PMF of $\boldsymbol{N}$. It is shown that [126] for any $\boldsymbol{N}$, there exists a latent multivariate Gaussian $\boldsymbol{U} = [U_0, U_1, ..., U_{B_\eta - 1}]^T$ with mean vector $\boldsymbol{\mu}_u$ and covariance matrix $\mathbf{C}_u$ such that after dichotomizing $\boldsymbol{U}$, i.e.

$$\hat{N}_i^b = \begin{cases} 1 & U_i \geq 0 \\ 0 & U_i < 0 \end{cases} \text{ for } (i = 0, 1..., B_\eta - 1) \tag{5.10}$$

the obtained RV $\hat{\boldsymbol{N}} = [\hat{N}_0^b, \hat{N}_1^b .., \hat{N}_{B_\eta - 1}^b]^T$ can have identical first and second order statistics as $\boldsymbol{N}$. Therefore, as shown in Section 5.5, REM-j can be obtained as:

$$\begin{aligned} P(\eta) &= P_{\boldsymbol{N}}([N_0^b, ...N_{B_\eta - 1}^b]^T = [\eta_0^b, ..., \eta_i^b]^T) \\ &= \Phi([0, ...0]^T; \mathbf{D}\boldsymbol{\mu}_u, \mathbf{D}\mathbf{C}_u\mathbf{D}^T) \end{aligned} \tag{5.11}$$

where

$$\mathbf{D} = \begin{pmatrix} (-1)^{\eta_0^b} & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & (-1)^{\eta_{B_\eta - 1}^b} \end{pmatrix} \tag{5.12}$$

and $\Phi([0, ...0]^T; \mathbf{D}\boldsymbol{\mu}_u, \mathbf{D}\mathbf{C}_u\mathbf{D}^T)$ is the CDF of the joint Gaussian with mean $\mathbf{D}\boldsymbol{\mu}_u$ and covariance matrix $\mathbf{D}\mathbf{C}_u\mathbf{D}^T$ evaluated at $[0, 0...0]^T$.

The mean and variance of $\boldsymbol{N}$ can be calculated using (5.3) and (5.11). In REM-j, the parameters to be estimated are the mean vector $\boldsymbol{\mu}_u$ and the covariance matrix $\mathbf{C}_u$. Thus, the modeling complexity is $O(B_\eta^2)$ as shown in (5.11).

5.2.3 GEM-i and GEM-j

The main difference between GEM-i,j (see (5.4)) and REM-i,j (see (5.2)) is that in GEM-i,j, the error is defined using addition over GF(2) instead of real addition. Since the vectorized form $\boldsymbol{N}$ and the 2's complement form N are equivalent, GEM-i,j can be derived in the same manner as REM-i,j. Therefore, the PMF $P(\eta)$ under GEM-i and GEM-j has the same form as in (5.9) and (5.11), respectively. Note that the independent error model [123] is a special case ($p_i = p, \forall i$) of the GEM-i model. The modeling complexities for GEM-i and GEM-j are $O(B_\eta)$ and $O(B_\eta^2)$, respectively.

5.3 Model Validation

The proposed models are validated and compared at both the kernel and system levels. Kernel level validation aims at comparing the JS divergence D_{JS} between the proposed models and the PMFs obtained via HDL simulation. System level validation aims at validating the accuracy of the proposed models in predicting system level performance metric S. As shown in Fig. 5.1, S can be obtained via averaging over the spatio-temporal domain. However, we employ the following procedure in order to evaluate the performance yield: (1) For each instance I_k, the system level performance metric for I_k is obtained by averaging over the input X, i.e., $S_k = E(S|I_k)$, where S_k is a RV, and (2) statistical measures such as the mean

and standard deviation of S_k can be obtained by performing spatial averaging.

5.3.1 Error Characterization and Injection Methodology

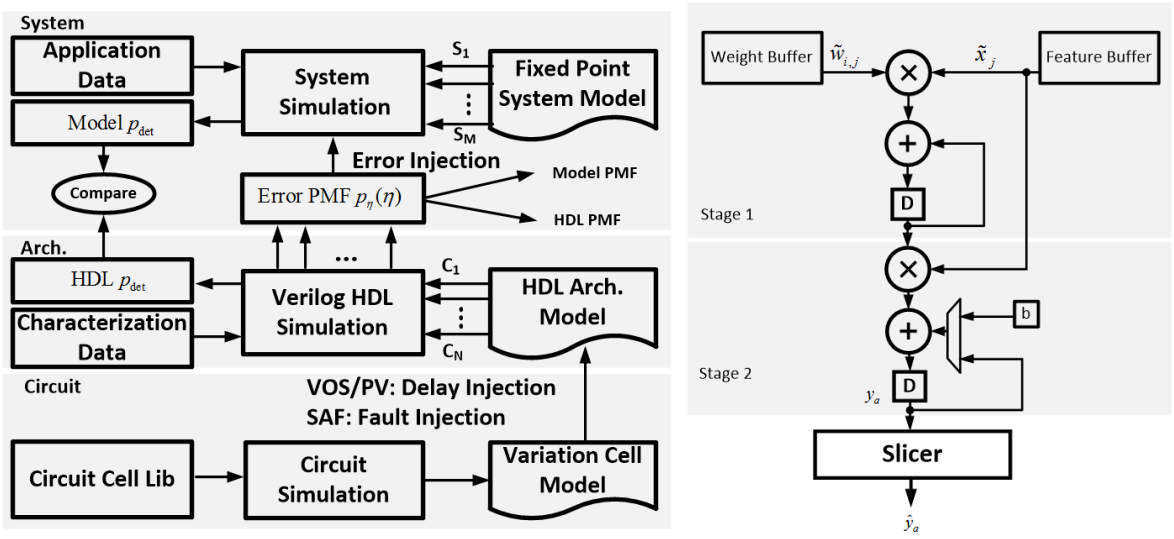


Figure 5.2: Model validation: (a) error characterization and injection methodology, and (b) 2^{nd} order polynomial kernel SVM classifier.

Figure 5.2(a) shows the error characterization and injection methodology for VOS, process variation, and defect errors in a common framework. In this paper, an SVM classifier as shown in Fig. 5.2(b) is employed to validate the models. The SVM classifier consists of two types of multiply accumulator (MAC) kernels: MAC1 is an 8 b input, 8 b coefficient, and 22 b output MAC used in the first stage, and MAC2 is a 10 b input, 8 b coefficient, and 24 b output MAC used in the second stage. Simulation results are obtained using a commercial 45 nm CMOS process.

VOS error characterization and injection are done as follows:

1. Characterize delay vs. V_{dd} of basic gates such as AND and XOR using HSPICE for $0.3\,\text{V} \leq V_{dd} \leq 1.2\,\text{V}$.

2. Develop structural Verilog HDL models for the SVM classifier using the basic gates characterized in Step 1.

3. Run HDL (bit and clock accurate) simulations using a characterization dataset to obtain error samples η and classification accuracy P_{det-h}. The characterization dataset is obtained via sampling with replacement from the application level data to emulate the input statistics. Note that we treate the detection accuracy $p_{det} = P(\hat{Y}_a = c)$ a RV, which we denote as P_{det}.

4. During kernel level validation, analytical models $P(\eta)$ were built using REM-i,j and GEM-i,j (see (5.9) and (5.11)). The JS divergence between the models and the characterized error PMFs were calculated according to (5.6).

5. During system level validation, run fixed-point MATLAB simulations using $P(\eta)$ to inject errors using the UCI dataset to obtain detection accuracy P_{det-s}. Compare P_{det-s} with P_{det-h}.

Process variation error characterization and injection are done as follows:

1. Characterize the gate delay distribution vs. operating voltage V_{dd} of basic gates such as AND and XOR using HSPICE in the NTV range 0.3 V-0.7 V.

2. Implement the SVM architecture using structural Verilog HDL using the basic gates characterized in Step 1.

3. Emulate process variations at NTV by generating multiple (30) architectural instances and assigning random gate delays obtained via sampling the gate delay distributions obtained in Step 1.

4. Kernel and system level model validations were conducted following the same procedure as in the VOS methodology.

Defect error characterization and injection are done as follows:

1. Develop structural Verilog HDL models for the SVM classifeir.

2. During HDL simulation, multiple instances (30) were generated, and defects (stuck-at-one and stuck-at-zero errors) with different defect error rate p_{saf} were injected to randomly selected nets in the Verilog netlist using custom scripts.

3. Kernel and system level model validations were conducted following the same procedure as in the VOS methodology.

5.3.2 Kernel Level Model Validation

Figure 5.3 shows the JS divergence comparison at different voltage overscaling factor $K_{vos} = V_{dd}/V_{dd,crit}$ where $V_{dd-crit}$ is the minimum voltage needed for error free operation. It shows that for VOS errors, GEM-j achieves the lowest D_{JS} which is below 10^{-2} for $0.5 \leq K_{vos} \leq 0.95$, and both GEM-i and GEM-j achieve up to two-orders-of-magnitude smaller D_{JS} compared with REM-i and REM-j. Additionally, Figure 5.3 shows that the D_{JS} of the symmetrical error model [122] and independent error model [123] is higher than the GEM models. For all modeling methods, the PMF modeling accuracy decreases as K_{vos} decreases (thus error rate p_η increases).

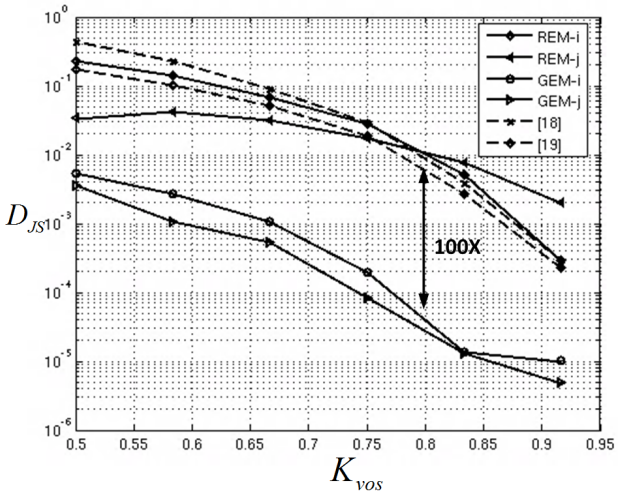


Figure 5.3: JS divergence comparison of the proposed models for VOS errors for MAC1 used in the SVM classifier. The JS divergence is calculated between the proposed models and the error PMFs obtained via HDL simulation. The results for MAC2 are similar.

Figure 5.4 shows that in the case of process variation errors, GEM-j achieves the lowest D_{JS} which is below 0.03 for $0.3\,\mathrm{V} \leq V_{dd} \leq 0.7\,\mathrm{V}$, and both GEM-i and GEM-j achieve up to $10\times$ smaller D_{JS} compared with REM-i and REM-j. Additionally, Figure 5.4 shows that the D_{JS} of the symmetrical error model [122] and independent error model [123] are higher than the GEM models. This is expected due to the fact that process variation errors are

indeed timing errors. For all modeling methods, the PMF modeling accuracy decreases with V_{dd}.

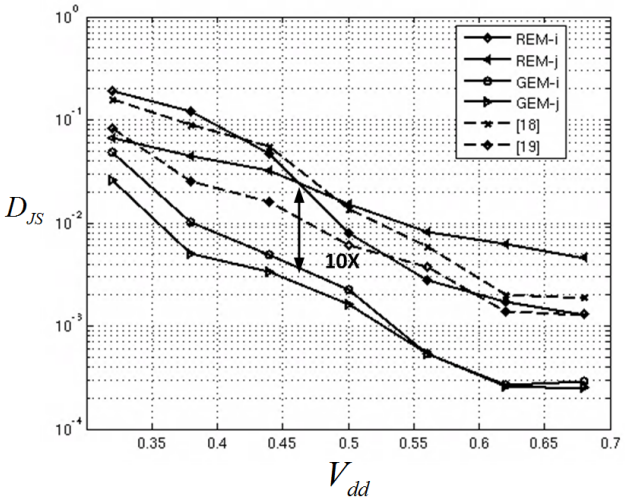


Figure 5.4: JS divergence comparison of the proposed models for process variation errors for MAC1 used in the SVM classifier. The JS divergence is calculated between the proposed models and error PMFs obtained via HDL simulation for $M = 30$ instances. The mean D_{JS} is shown in the figure. The results for MAC2 are similar.

Figure 5.5 shows that unlike timing errors caused by VOS or process variations, in the case of defects, REM-j achieves the lowest D_{JS} which is below 0.05 for $10^{-3} \leq p_{saf} \leq 0.2$ compared with other models. This indicates that the error statistics of defect errors are different from timing errors, and different model should be employed. In addition, Figure 5.5 shows the D_{JS} of the symmetrical error model [122] and independent error model [123] is higher than any of the proposed models. Figure 5.5 also shows that PMF modeling accuracy decreases at higher p_{saf} for all modeling methods.

5.3.3 System Level Simulation

To evaluate the model, we employ the probabilistic models in system simulation and compare them with HDL results following the procedure in Section 5.3.1. We employ the Breast Cancer Wisconsin dataset from the UCI machine learning repository [124] which consists of labeled feature vectors (benign vs. malignant) constructed from digitized images of fine needle aspirates (FNA) of patient tissue, and use the SVM classifier to perform classification.

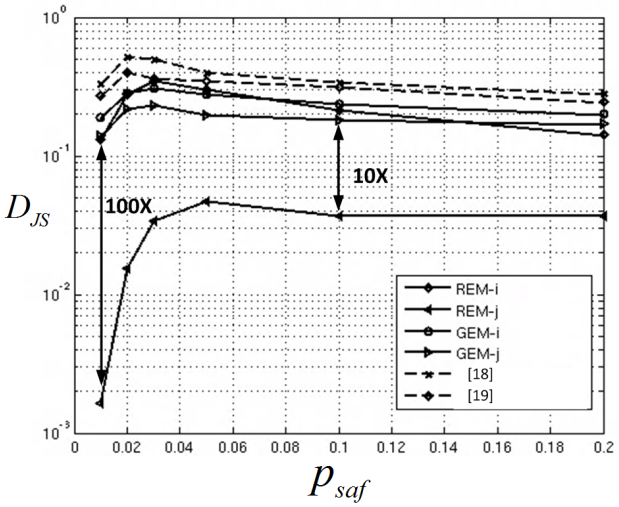


Figure 5.5: JS divergence comparison of the proposed models for defect errors for MAC1 used in the SVM classifier. The JS divergence is calculated between the proposed models and error PMFs obtained via HDL simulation for $M = 30$ instances. The mean D_{JS} is shown in the figure. The results for MAC2 are similar.

Figure 5.6 plots the P_{det} in presence of VOS errors, and shows that GEM-i and GEM-j are more accurate than REM-i and REM-j. The difference between the estimated P_{det} using GEM-j and HDL error statistics is within 3% for error rate $p_\eta \leq 80\%$.

Figure 5.7 shows the distribution of P_{det} in presence of process variation errors, and demonstrates that GEM-i and GEM-j are more accurate than REM-i and REM-j, similar to the case of VOS errors. The difference between the estimated P_{det} using GEM-j and HDL error statistics is within 5% for $0.3\,\mathrm{V} \leq V_{dd} \leq 0.7\,\mathrm{V}$.

Figure 5.8 shows the distribution of P_{det} in presence of defect errors, and demonstrates that REM-j achieves higher accuracy than other models, unlike the case of VOS and process variation errors. The difference between the estimated P_{det} using REM-j and HDL error statistics is within 2% for $10^{-3} \leq p_{saf} \leq 0.2$.

5.4 Conclusion

In this chapter, probabilistic additive models for circuit errors due to VOS, process variation, and defects were proposed to effectively predict the performance of ML kernels in presence

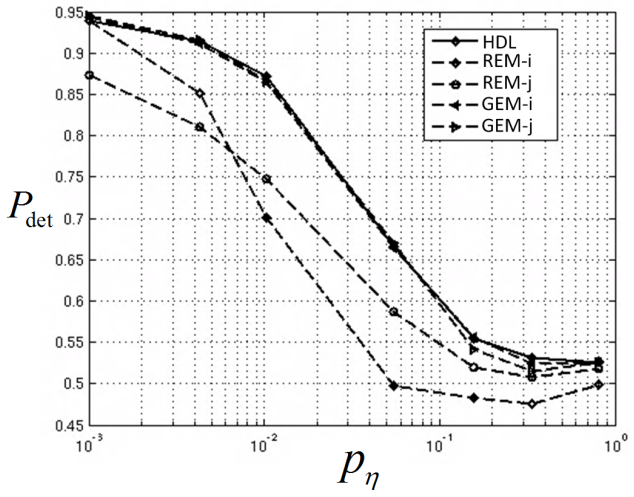


Figure 5.6: System simulation results in presence of VOS errors for the SVM classifier comparing the proposed models with HDL simulation results.

of hardware errors. Four models were compared, and analytical expressions for the PMF were derived. In addition, error characterization/injection methodologies were proposed and employed to validate the models. Kernel level validation showed that the GEM-j is the most accurate for VOS and process variation errors, but REM-j is the most accurate for defect errors. System level simulation using a 2^{nd} order polynomial SVM classifier further confirms the validity of the models.

5.5 Derivation of REM-j

In this section, we derive (5.11). Use vectorized notation $\boldsymbol{\eta}$ in (5.5), the $P(\eta)$ can be expressed as:

$$\begin{aligned} & P(\eta) \\ & = P([N_0^b, ...N_{B_\eta-1}^b]^T = [\eta_0^b, ..., \eta_i^b]^T) \\ & = P_{\boldsymbol{U}}((-1)^{\eta_0} U_0 < 0, .., (-1)^{\eta_{B_\eta-1}} U_{B_\eta-1} < 0) \end{aligned} \tag{5.13}$$

where $\boldsymbol{U} = [U_0, U_1, ..., U_{B_\eta-1}]^T$ is the latent Gaussian that can be dichotomized to obtain $\boldsymbol{\eta}$ according to (5.10). We further define $\hat{\boldsymbol{U}} = \mathbf{D}\boldsymbol{U}$ where $\mathbf{D}$ is defined in (5.12). Hence, the mean and variance of $\hat{\boldsymbol{U}}$ can be calculated as $E(\hat{\boldsymbol{U}}) = \mathbf{D}\boldsymbol{\mu}_u$ and $Cov(\hat{\boldsymbol{U}}) = \mathbf{D}\mathbf{C}_u\mathbf{D}^T$, where

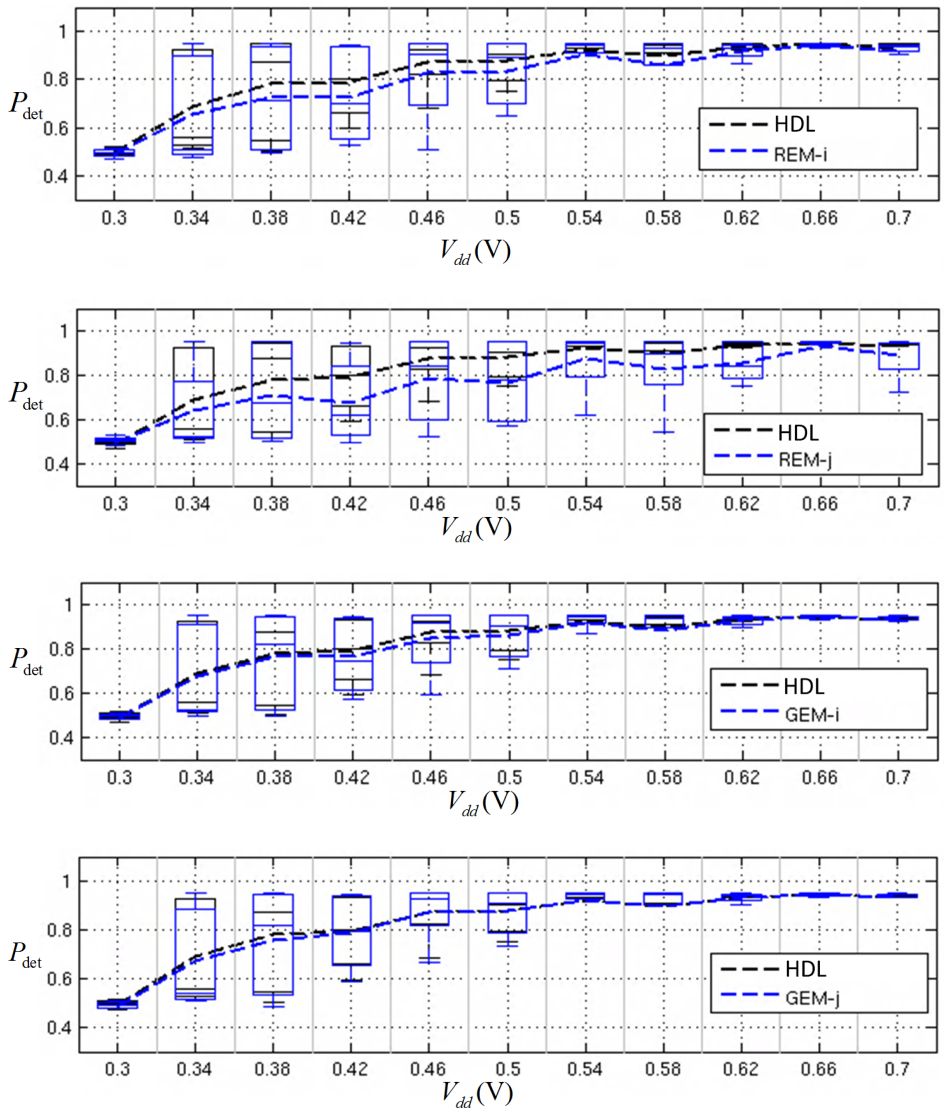


Figure 5.7: System simulation results in presence of process variation errors for the SVM classifier comparing the HDL simulation results with (a) REM-i, (b) REM-j, (c) GEM-i, and (d) GEM-j. Simulations are performed for 30 instances, the box plot shows the median, 25%, and 75% quartile of the prediction accuracy P_{det}, the dashed line shows the median P_{det}.

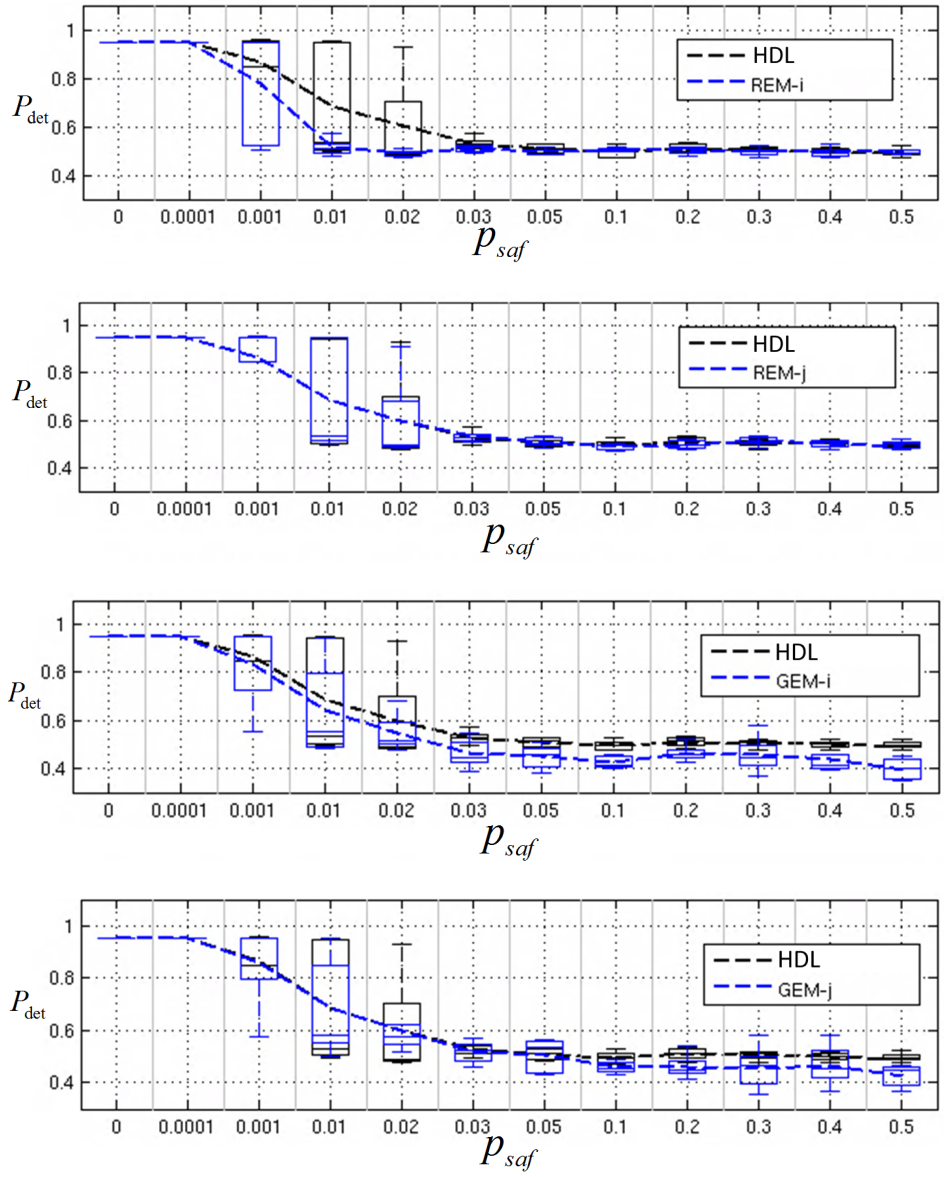


Figure 5.8: System simulation results in presence of defect errors for the SVM classifier comparing the HDL simulation results with (a) REM-i, (b) REM-j, (c) GEM-i, and (d) GEM-j. Simulations are performed for 30 instances, the box plot shows the median, 25%, and 75% quartile of the prediction accuracy P_{det}, the dashed line shows the median P_{det}.

$\boldsymbol{\mu}_u$ and $\mathbf{C}_u$ are the mean vector and covariance matrix of the latent Gaussian $\boldsymbol{U}$. The PMF $P(\eta)$ can then be obtained as:

$$P(\eta) = \Phi([0, ...0]^T; \mathbf{D}\boldsymbol{\mu}_u, \mathbf{D}\mathbf{C}_u\mathbf{D}^T)$$

where $\Phi([0, ...0]^T; \mathbf{D}\boldsymbol{\mu}_u, \mathbf{D}\mathbf{C}_u\mathbf{D}^T)$ is the CDF of the joint Gaussian $\boldsymbol{U}$ evaluated at $[0, 0..., 0]^T$.

Chapter 6

ERROR-RESILIENT MACHINE LEARNING IN NEAR THRESHOLD VOLTAGE VIA CLASSIFIER ENSEMBLE

In this chapter, we present the design of error-resilient machine learning architectures by employing a distributed machine learning framework referred to as *classifier ensemble* (CE). The most common machine learning architecture is the centralized architecture (see Fig. 6.1(a)) where a complex block such as the support vector machine (SVM) is employed to process all the input data. However, the computational complexity of centralized architecture increases dramatically as a function of the non-linearity of the decision boundary [54]. The CE (see Fig. 6.1(b)) is a distributed architecture for machine learning which combines several weak (low-complexity) classifiers to form a strong classifier. CE enables on-chip training due to its distributed nature, and exhibits robustness to feature/label noise. Thus, it is of great importance to compare the robustness and energy efficiency of distributed machine learning architectures designed using CE with centralized architectures such as SVM. Specifically, we hypothesize that architectures based on distributed algorithms are more robust than those based on centralized ones in presence of timing errors due to NTV operations. We compare a CE method - random forest (RF) - with SVM using architectural-level error models [127] in a commercial 45 nm CMOS process on the breast cancer data set in the UCI machine learning repository [124]. We show that RF achieves a detection accuracy (P_{det}) that varies by 3.2% while maintaining a median $P_{det} \geq 0.9$ when operating with a gate level delay variation of 28.9%. This is 5× lower as compared to SVM which exhibits a P_{det} that varies by 16.8% under identical conditions. We further propose a new error weighted voting to enhance the robustness of RF by employing the timing error statistics of the NTV circuit fabric. Simulation results confirm that the proposed method leads to a P_{det} that varies by only 1.4%, which is 12× lower compared to SVM.

The rest of the chapter is organized as follows. Section 6.1 provides the background

for CE, SVM. Section 6.2 describes dedicated architectures for RF and SVM classifiers. Section 6.3 presents simulation results validating the error models in a 45 nm CMOS process, and employs these models to compare the detection accuracy of SVM, RF, and proposed RF with error weighted voting scheme. Conclusions are presented in Section 6.4.

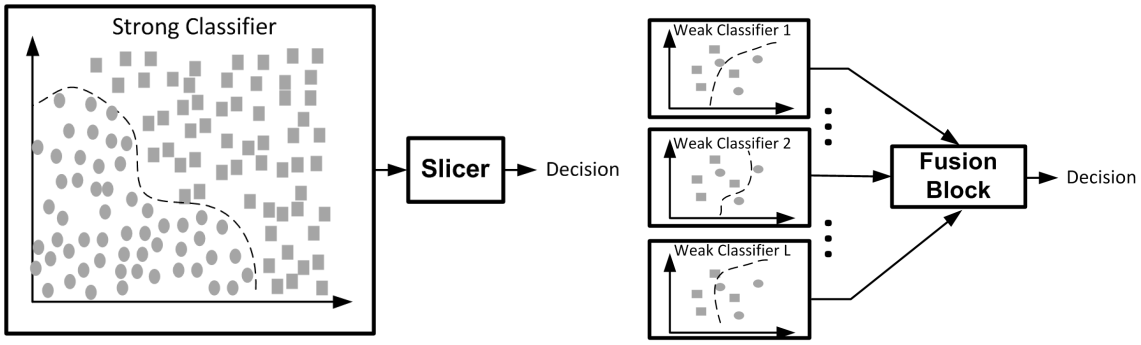


Figure 6.1: Two distinct machine learning frameworks: (a) centralized machine learning, and (b) classifier ensemble.

6.1 Background

6.1.1 Classifier Ensemble (CE)

Classifier ensemble (also referred to as multiple classifier system) has been employed to enhance the performance of single classifier system [128]. A wide variety of CE methods exist. In bootstrap aggregating (bagging) [129], multiple training sets are generated from the original training set via random sampling with replacement, in order to train multiple classifiers. Adaboost [130] is another popular method for ensemble generation. The training samples are re-weighted after each iteration so that the mis-classified samples get higher weights. Other methods such as randomness injection, random subspace and output coding [128] also exist.

RF is a CE method that combines random subspace and bagging, while employing an ensemble of decision trees (DTs) as weak classifiers. It is a popular technique for classification, prediction, and variable selection, and yielded results superior to those of other linear and non-linear predictive modeling techniques [131]. Advantages include parallel training,

robustness to overfitting, ease of design, the capability of getting out-of-bag (OOB) error estimate, and others.

In RF, the training set for each individual DT is generated using bagging. During the training of each DT, a random subset of features is selected, and the best feature is selected to split the DT according to an appropriate criterion. Several variations of RF exist based on the type of DT used as base classifiers. Classification and regression tree (CART) [131] employs the Gini index as a measure of the impurity of nodes. ID3 [132] employs information gain as the criterion. C4.5 [132] improves ID3 by using the information gain ratio.

6.1.2 Support Vector Machine

Support vector machine (SVM) [133] is a popular supervised learning method for classification and regression. SVM operates by first training a model (the training phase) followed by test/classification (the test phase). During the training phase, labeled feature vectors are used to train a model. During the test phase, SVM produces a predictive label when provided with a new (test) feature vector. SVM training can be formulated as the solution to the following optimization problem [133]:

$$\begin{aligned}
&\min \tfrac{1}{2}\|\mathbf{w}\|^2 + C\sum_i \xi_i \\
&s.t. \\
&c_i(\mathbf{w}^T\mathbf{x}_i - \mathrm{b}) \geq 1 - \xi_i \\
&\xi_i \geq 0
\end{aligned}$$

where C is the cost factor, ξ_i is the soft margin, $\mathbf{x}_i$ is the feature vector, c_i is the label corresponding to the feature vector $\mathbf{x}_i$, $\mathbf{w}$ is the weight vector, and b is the bias. It can be shown that the optimum weights are a linear combination of the feature vectors that lie on the margins, i.e., support vectors. Kernel tricks can be employed to realize non-linear decision boundaries [133].

6.2 System Architecture

In this section, we present system architectures for RF and SVM classifiers.

6.2.1 The RF Architecture

The RF classifier is implemented using an ensemble of L two-stage DT classifiers (weak learners) shown in Fig. 6.2(a). The l^{th} DT is trained from a bootstrapped training set $\mathcal{S}_l$ obtained from the original training set $\mathcal{S}$, and processes the M_l-dimensional data vector $\mathbf{x}_l = [x_{l,1}, x_{l,2}, \ldots, x_{l,M_l}]^T$ obtained from the M-dimensional test data vector $\mathbf{x}$ ($M \gg M_l$).

Stage 1 of the l^{th} DT consists of a comparator array that computes $sgn(x_{l,i} - T_{l,i})$ ($l = 1, 2, \ldots, L$, and $i = 1, 2, \ldots, M_l$) where $T_{l,i}$s are the thresholds obtained via training. Stage 2 consists of a look up table (LUT) which encodes the decision of each root-to-leaf path into a 1-bit output $y_{a,l} \in \{0, 1\}$. The outputs of the L DTs are combined via a voter block to generate the final decision. Each DT is trained using the Gini index [131] as the training criterion.

Conventionally, a majority voter is employed to combine the outputs from all DTs as follows:

$$\hat{y}_a = maj(y_{a,1}, y_{a,2}, ..., y_{a,L})$$

where $\hat{y}_a \in \{0, 1\}$ is the majority voter output, and $y_{a,l}$ is the l^{th} DT output given by:

$$y_{a,l} = y_{o,l} \oplus \eta_l$$

where $y_{o,l} \in \{0, 1\}$ is the error-free output and $\eta_l \in \{0, 1\}$ is the timing error of the l^{th} DT. The RF with majority voter is denoted as RF-M. In case of binary classification, the majority voter can be implemented as shown in Fig. 6.2(b).

In order to enhance the robustness of RF in presence of timing errors, we propose an *error weighted voting* scheme where the timing error statistics are incorporated during the decision

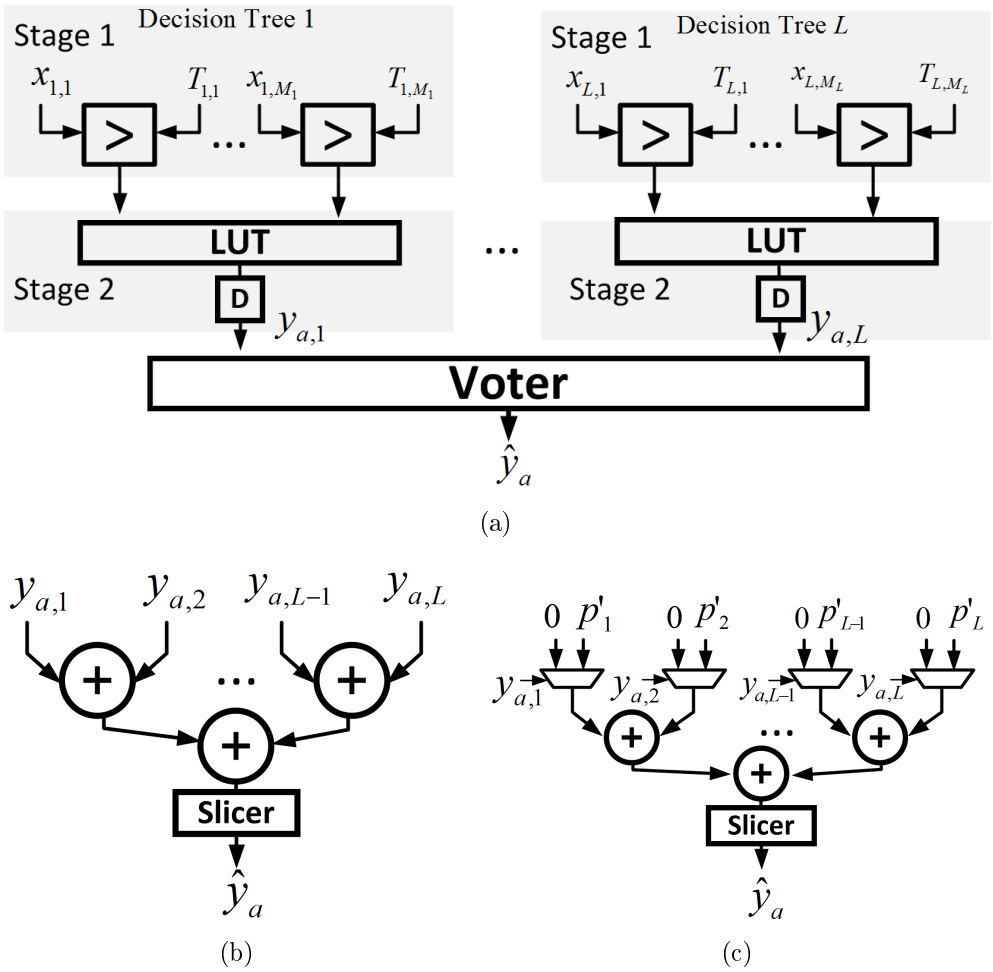


Figure 6.2: System architecture for: (a) the RF classifier with L DTs, (b) the majority voter, and (c) the weighted voter.

process. In order to do so, we employ the maximum-a-posterior (MAP) criterion, i.e.:

$$\hat{y}_a = \arg\max_{\forall c \in \mathcal{C}} P(c|\mathbf{x}) \tag{6.1}$$

where $\mathcal{C}$ is the label set, and $P(c|\mathbf{x})$ is the posterior probability of class label c conditioned on the *test data* $\mathbf{x}$. Thus:

$$
\begin{aligned}
P(c|\mathbf{x}) &= \sum_{l=1}^{L} P(c|R_l, \mathbf{x}) P(R_l|\mathbf{x}) && (6.2)\\
&= \sum_{l=1}^{L} P(c|R_l, \mathbf{x}) P(R_l) && (6.3)\\
&\approx \sum_{l=1}^{L} \mathbf{1}\{y_{a,l} = c\} p_l && (6.4)
\end{aligned}
$$

where $P(c|R_l, \mathbf{x})$ denotes the posterior probability of the class label, R_l is the event of the l^{th} DT being correct during the training phase, $p_l = P(R_l)$ is the probability of the event R_l, and $\mathbf{1}\{\cdot\}$ denotes the indicator function. Equation (6.2) implies (6.3) because the test data $\mathbf{x}$ and event R_l are independent, and (6.3) implies (6.4) because we assume the DT output has a probability mass of 1 at the selected class label. The final decision $\hat{y}_a$ is obtained from (6.1) by choosing the label c that maximizes (6.4). Note that p_l represents the decision accuracy of the l^{th} DT in presence of timing errors.

In the case of binary classification, one can simplify (6.1) using (6.4) into:

$$
\hat{y}_a = \begin{cases} 1 & \text{if } \sum_{l=1}^{L} \mathbf{1}\{y_{a,l} = 1\} p'_l > \frac{1}{2} \\ 0 & \text{otherwise} \end{cases}
$$

where $p'_l = \frac{p_l}{\sum_{l=1}^{L} p_l}$ and the voter can be implemented as shown in Fig. 6.2(c).

To incorporate the timing error statistics of each DT, we express p_l in (6.4) as follows:

$$
p_l = \sum_{\eta_l=0}^{1} P(R_l, \eta_l) = \sum_{\eta_l=0}^{1} P(R_l|\eta_l) P(\eta_l) \tag{6.5}
$$

where $P(R_l|\eta_l)$ is the probability of correct decision of the l^{th} DT conditioned on η_l. The probabilities $P(R_l|\eta_l)$ and $P(\eta_l)$ can be obtained during the training phase for each DT. For a RF binary classifier, (6.5) can be simplified into (see Section 6.5.1):

$$
p_l = P(R_l|\eta_l = 0)(1 - p_{\eta_l}) + (1 - P(R_l|\eta_l = 0)) p_{\eta_l} \tag{6.6}
$$

where $P(R_l|\eta_l = 0)$ can be obtained via performing validation using out-of-bag samples, and $p_{\eta_l} = P(\eta_l \neq 0)$ is the error rate of the l^{th} DT. We denote RF with error weighted voting scheme as RF-EW.

When error rate $p_{\eta_l} = 0$, the error weighted voting scheme reduces to the conventional weighted voter [128] where $p_l = P(R_l|\eta_l = 0)$. The RF with conventional weighted voter is denoted as RF-W.

The performance of RF-EW improves when the DTs exhibit uncorrelated errors, i.e., the DT outputs exhibit diversity in terms of error statistics. It is possible to enhance DT diversity by designing each DT to have different: (1) algorithm (algorithmic diversity), (2) architecture (architectural diversity), and (3) data-path precision (precisional diversity), across the DT ensemble. Precision has a significant impact on the timing error statistics since the hardware errors under investigation are due to timing violations. Therefore, in this paper, the precision of each DT data-path in the RF-EW is randomly assigned uniformly between 4b and 8b, leading to different critical path delays among the DTs, and hence uncorrelated errors.

6.2.2 The SVM Architecture

The centralized machine learning algorithm employed in this paper is a second-order polynomial kernel SVM described as:

$$\begin{aligned} \hat{y}_a &= sgn(y_a) \\ y_a &= \sum_{i=1}^{N} (\beta \mathbf{s}_i^T \mathbf{x} + \gamma)^2 \alpha_i + b \end{aligned} \tag{6.7}$$

where $\mathbf{x} = [x_1, x_2, ..., x_M]^T$ is the M dimensional test data vector, $\mathbf{s}_i = [s_1, s_2, ..., s_M]^T$ is the i^{th} support vector, α_i is the weight associated with $\mathbf{s}_i$, b is the bias, β and γ are parameters of the polynomial kernel, and N is the total number of support vectors (typically $N \gg M$). Direct computation of (6.7) requires $O(NM)$ multiply-accumulate (MAC) operations. The

following reformulation [134] reduces the number of MAC operations to $O(M^2)$:

$$\begin{aligned} y_a &= \tilde{\mathbf{x}}^T \tilde{\mathbf{W}} \tilde{\mathbf{x}} + b \\ \tilde{\mathbf{W}} &= \sum_{i=1}^{N} \alpha_i \tilde{\mathbf{s}}_i \tilde{\mathbf{s}}_i{}^T \end{aligned} \tag{6.8}$$

where $\tilde{\mathbf{W}}$ is a precomputed weight matrix, $\tilde{\mathbf{x}} = \begin{bmatrix} 1 \\ \mathbf{x} \end{bmatrix}$, and $\tilde{\mathbf{s}}_i = \begin{bmatrix} \gamma \\ \beta \mathbf{s}_i \end{bmatrix}$. Figure 6.3 shows a folded SVM architecture implementing (6.8) where Stage 1 computes $\tilde{\mathbf{W}}\tilde{\mathbf{x}}$, and Stage 2 computes the dot product between $\tilde{\mathbf{x}}$ and Stage 1 output, and adds the bias term b.

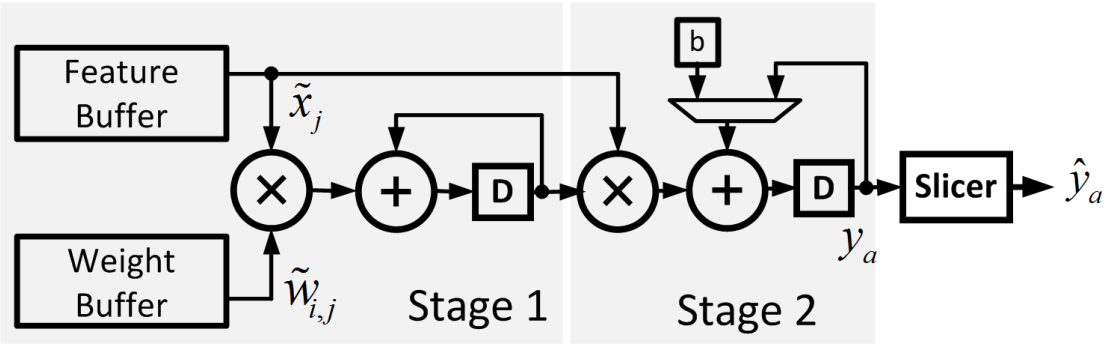


Figure 6.3: System architecture for a second-order polynomial kernel SVM classifier.

6.2.3 System Analysis

The potential robustness improvement achieved by RF can be analyzed by inspecting the generalized error $E\left[\left(C - \frac{1}{L}\sum_{l=1}^{L} \hat{Y}_{a,l}\right)^2\right]$ where C is the label and $\frac{1}{L}\sum_{l=1}^{L} \hat{Y}_{a,l}$ is the RF output where equal weights in the voter are assumed for simplicity of analysis. Here the expectation is taken over the distribution of the label C, the training set $\mathcal{S}$, and the timing error $N_1, ..., N_L$.

We start by deriving the generalized error for a single DT defined as $E[(C - \hat{Y}_a)^2]$ where $\hat{Y}_a$ is the DT output. It can be shown that (see Section 6.5.2):

$$E\left[\left(C - \hat{Y}_a\right)^2\right] = \sigma_C^2 + b^2 + \sigma_{\hat{Y}_a}^2 \tag{6.9}$$

where $\sigma_C^2 = E\big[\big(C - E[C]\big)^2\big]$ is the irreducible error (noise), $b^2 = (E[C] - E[\hat{Y}_a])^2$ is the bias term and $\sigma_{\hat{Y}_a}^2 = E\big[\big(E[\hat{Y}_a] - \hat{Y}_a\big)^2\big]$ is the variance of $\hat{Y}_a$. Such a decomposition identifies the contribution of different error sources and allows one to understand the effect of CE in reducing these errors.

For CE, it can further be shown that the noise $\sigma_{C,RF}^2$ and the bias b_{RF}^2 (corresponding to the first two terms in (6.9)) do not change, i.e., $\sigma_{C,RF}^2 = \sigma_C^2$ and $b_{RF}^2 = b^2$, respectively. However, the output variance σ_{RF}^2 can be expressed as (see Section 6.5.3):

$$\sigma_{RF}^2 = \frac{1}{L}\sigma_{\hat{Y}_a}^2 \tag{6.10}$$

We can see from (6.10) that σ_{RF}^2 is reduced by a factor of $\frac{1}{L}$ from $\sigma_{\hat{Y}_a}^2$, and that for the RF to achieve a lower variance, $\sigma_{\hat{Y}_a}^2$ should be less than L times the variance of the centralized system. The reduction of variance leads to reduced generalized error and mis-classification rate.

6.3 Simulation Results

In section 6.3.1 the detection accuracies of the SVM and RF architectures are compared using the validated error models and methodology from Chapter 5. We employ the Breast Cancer Wisconsin dataset from the UCI machine learning repository [124] which consists of labeled feature vectors (benign vs. malignant) constructed from digitized images of fine needle aspirates (FNA) of patient tissue.

The SVM architecture being considered in this study consists of two types of MACs: Stage 1 employs 8 b input, 8 b coefficient, and 22 b output MACs, and Stage 2 employs 10 b input, 8 b coefficient, and 24 b output MACs. The conventional RF architecture employing majority and weighted voter has Stage 1 consisting of comparator arrays with 8 b input and 8 b thresholds, and LUTs implemented as logic networks during the architecture generation. In the proposed RF-EW, each DT is implemented using a randomly selected precision uniformly distributed between 4 b and 8 b for both the input and the thresholds. In all cases, the parameters of SVM and RF were trained assuming no timing errors. These precisions

were chosen to obtain less than 0.5% degradation in P_{det} compared to a floating point implementation. The complexities of the SVM and RF (with ensemble size $L = 10$) were found to be 1.63K and 1.47K 2-input NAND gate equivalents, respectively.

6.3.1 Comparison of SVM and RF

6.3.1.1 Comparison of Timing Error Rates

We first compare the timing error rates $p_\eta = P(\boldsymbol{\eta} \neq 0)$ of SVM and RF obtained via HDL simulations as the voltage decreases in NTV. Figure 6.4 shows that the median timing error rate $\bar{p}_\eta$ increases by 500× from 2.1×10^{-3} to 0.99, and from 1.1×10^{-3} to 0.61 for SVM and RF, respectively, as the voltage V_{dd} decreases from 0.7 V to 0.3 V, indicating that the RF architecture has up to 4.5× lower timing error rate compared with SVM. The error rate of RF architecture is lower because it has comparator blocks which have a much simpler data path compared with the MAC units in SVM. Figure 6.4 also demonstrates that the gate level delay variation $(\sigma/\mu)_d$ increases by 12× from 2.8% to 33% as the voltage V_{dd} decreases from 0.7 V to 0.3 V.

Next, we employ $P(\boldsymbol{\eta})$ to inject errors in fixed-point MATLAB simulations of SVM and RF architectures to compare their robustness to timing errors in NTV. All comparisons henceforth are in terms of P_{det-s}. Hence, we simplify the subscript and denote the detection accuracy as P_{det}. Four architectures are compared: (1) SVM, (2) RF with majority voter [131] (RF-M), (3) RF with weighted majority voter [128] (RF-W), and (4) RF with the proposed error weighted voter (RF-EW). We will compare the four architectures in terms of median ($\bar{p}_{det}$) and standard deviation ($\sigma_{p_{det}}$) of detection accuracy P_{det}.

6.3.1.2 Comparison of $\bar{p}_{det}$

Figure 6.5(a) shows that RF has higher $\bar{p}_{det}$ than SVM when the ensemble size L is sufficiently large. Specifically, RF-M is able to maintain $\bar{p}_{det} \geq 0.9$ for $(\sigma/\mu)_d \leq 28.9\%$ with $L = 10$, whereas SVM can only maintain the same performance for $(\sigma/\mu)_d \leq 11.7\%$. Additionally, RF-EW achieves up to 3% higher $\bar{p}_{det}$ compared with RF-W and RF-EW, and is able to

maintain $\bar{p}_{det} \geq 0.93$ for $(\sigma/\mu)_d \leq 29.6\%$. Finally, Figure 6.5(a) further shows that RF with $L = 10$ is able to maintain detection performance even at $(\sigma/\mu)_d$ of 28.9%. This indicates that RF architectures have a higher robustness to timing errors compared with SVM in spite of its complexity being lower by 10% when $L = 10$.

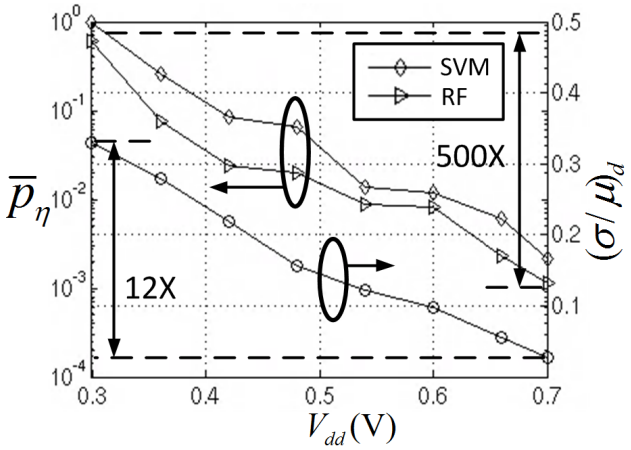
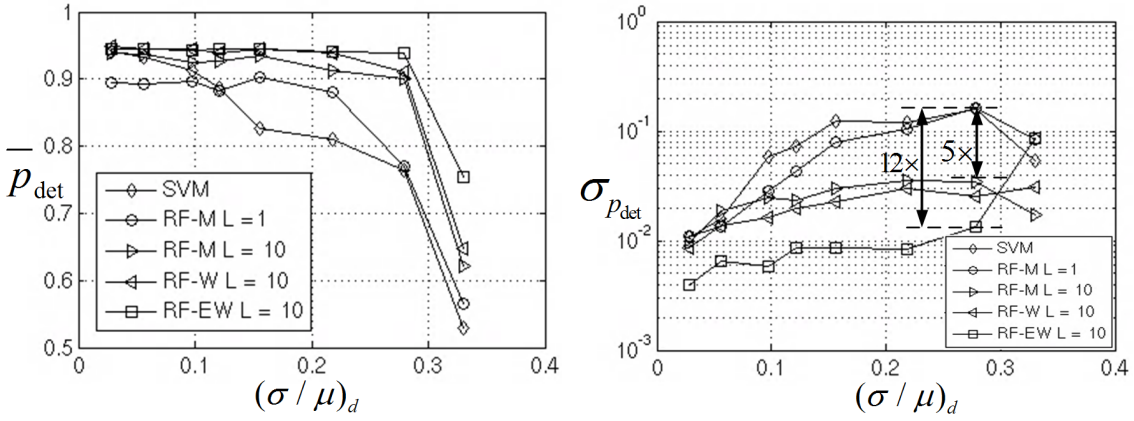


Figure 6.4: Median error rate $\bar{p}_\eta$ and gate level delay variation $(\sigma/\mu)_d$ of SVM and RF architecture in NTV region of $0.3\,\mathrm{V} \leq V_{dd} \leq 0.7\,\mathrm{V}$.



(a) (b)

Figure 6.5: Robustness comparison in: (a) the median detection accuracy $\bar{p}_{det}$, and (b) the standard deviation of detection accuracy σp_{det} for SVM classifier, RF-M with $L = 1$ (i.e. single DT), and RF-M ($L = 10$), RF-W ($L = 10$), and RF-EW ($L = 10$). Simulations were performed over 30 instances.

6.3.1.3 Comparison of $\sigma_{p_{det}}$

Figure 6.5(b) shows that $\sigma_{p_{det}}$ is significantly reduced as L increases. RF-M achieves $\sigma_{p_{det}} \leq 3.5 \times 10^{-2}$ when $L = 10$, which is $5X$ lower compared to SVM or RF-M with $L = 1$. This further demonstrates that distributed architectures are inherently more robust to timing errors than centralized ones. Figure 6.5(b) also shows that RF-EW achieves $\sigma_{p_{det}} \leq 1.4 \times 10^{-2}$ when $(\sigma/\mu)_d \leq 29.6\%$, which is 12× and 3.5× lower compared to SVM and RF-W, respectively. This demonstrates that incorporating timing error statistics into the decision making process enhances robustness. When $(\sigma/\mu)_d \geq 30\%$, $\sigma_{p_{det}}$ of RF-EW is higher than that of RF-M and RF-W because all instances of RF-M and RF-W achieve a low $P_{det} \approx 0.6$, whereas some instances of RF-EW can still achieve a $P_{det} \geq 0.9$, leading to increased $\sigma_{p_{det}}$.

To further understand the robustness improvement achieved by RF, Fig. 6.6 shows that the RF output variance σ^2_{RF} reduces from 0.16 to 0.02 as L increases from 1 to 25 when no precision diversity is employed. The variance reduction is more significant when the ensemble size L is small, and slows down as L further increases. This is because the independence assumption across the DTs is violated for large L. Figure 6.6 also shows that σ^2_{RF} can be further reduced to 0.01 due to more uncorrelated error statistics when precision diversity is employed as in the RF-EW.

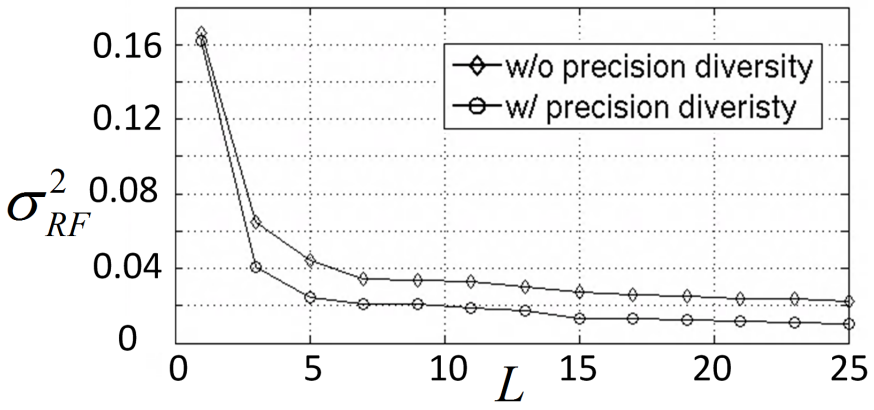


Figure 6.6: The variance of RF output when $(\sigma/\mu)_d = 29\%$.

.

6.4 Conclusion

In this chapter, the inherent robustness of CE and centralized machine learning architectures in presence of timing violations is compared. It is shown that distributed architectures employing CE are inherently more robust to timing errors than centralized ones. Furthermore, it is shown that the algorithm itself can be adapted to further enhance the robustness. Such enhancement is achieved by using error weighted voting during the decision combination, and employing precision diversity in the architecture data path. The results demonstrate that in the CE framework, architectural level information can be incorporated at the system level to achieve enhanced robustness. In the future, architectural and algorithmic level diversity techniques can be employed to improve the robustness of CE. In addition, the robustness of CE in presence of defects errors (stuck-at-faults) can also be evaluated.

6.5 Derivations

6.5.1 Derivation of (6.6)

In this subsection, we derive (6.6). From the theorem of total probability:

$$\begin{aligned} p_l &= P(R_l|\eta_l = 0)P(\eta_l = 0) + P(R_l|\eta_l = 1)P(\eta_l \neq 0) \\ &= P(R_l|\eta_l = 0)(1 - p_{\eta_l}) + P(R_l|\eta_l = 1)p_{\eta_l} \end{aligned} \tag{6.11}$$

We need to show $P(R_l|\eta_l = 1) = 1 - P(R_l|\eta_l = 0)$. In binary classification, the erroneous output of the l^{th} DT can be expressed as:

$$y_{a,l} = c \oplus \eta_l \oplus e_l \tag{6.12}$$

where c, η_l, e_l denote the true label, timing error, and error due to noise in data, respectively. Thus, the event $R_l = \{e_l \oplus \eta_l = 0\}$, and we have:

$$\begin{aligned} P(R_l|\eta_l = 1) &= P(e_l \oplus \eta_l = 0|\eta_l = 1) \\ &= P(e_l = 1|\eta_l = 1) \quad (6.13) \\ &= P(e_l = 1) \quad (6.14) \\ &= P(e_l = 1|\eta_l = 0) \\ &= P(e_l \oplus \eta_l = 1|\eta_l = 0) \\ &= 1 - P(e_l \oplus \eta_l = 0|\eta_l = 0) \\ &= 1 - P(R_l|\eta_l = 0) \quad (6.15) \end{aligned}$$

where (6.13) to (6.14) comes from the independence of e_l and η_l. Substituting (6.15) into (6.11) leads to (6.6) thereby completing the proof of (6.6).

6.5.2 Derivation of (6.9)

In this subsection, we derive (6.9). In deriving the generalized error $E[(C - \hat{Y}_a)^2]$, the expectation is taken over label C, the training set $\mathcal{S}$, and the timing error N. Here $\mathcal{S}$ and N are independent. Without loss of generality, we assume a fixed input $X = \mathbf{x}$ as suggested by [135] for notational simplicity. Thus, (6.9) can be expressed as:

$$\begin{aligned} & E\big[\big(C - \hat{Y}_a\big)^2\big] \\ = \; & E\big[\big(C - E[C] + E[C] - \hat{Y}_a\big)^2\big] \\ = \; & E\big[(C - E[C])^2\big] + E\big[(E[C] - \hat{Y}_a)^2\big] \quad (6.16) \end{aligned}$$

where we use the fact that

$$\begin{aligned} & E\big[\big(C - E[C]\big)\big(E[C] - \hat{Y}_a\big)\big] \\ = \; & E\Big[E\big[(C - E[C])(E[C] - \hat{Y}_a)|S, N\big]\Big] \\ = \; & 0 \end{aligned} \tag{6.17}$$

The first term in (6.16) $E\big[(C - E[C])^2\big]$ is the noise σ_C^2. The second term in (6.16) can be further decomposed as:

$$\begin{aligned} & E\big[\big(E[C] - \hat{Y}_a\big)^2\big] \\ = \; & E\big[\big(E[C] - E[\hat{Y}_a] + E[\hat{Y}_a] - \hat{Y}_a\big)^2\big] && (6.18) \\ = \; & (E[C] - E[\hat{Y}_a])^2 + E\big[\big(E[\hat{Y}_a] - \hat{Y}_a\big)^2\big] && (6.19) \\ = \; & b^2 + \sigma_{\hat{Y}_a}^2 && (6.20) \end{aligned}$$

where in going from (6.18) to (6.19) we use the fact that

$$\begin{aligned} & E\Big[\big(E[C] - E[\hat{Y}_a]\big)\big(E[\hat{Y}_a] - \hat{Y}_a\big)\Big] \\ = \; & \big(E[C] - E[\hat{Y}_a]\big)E\big[E[\hat{Y}_a] - \hat{Y}_a\big] \\ = \; & 0 \end{aligned} \tag{6.21}$$

This completes the proof of (6.9).

6.5.3 Derivation of (6.10)

In this subsection, we derive (6.10). In deriving the generalized error $E[(C - \frac{1}{L}\sum_{l=1}^{L}\hat{Y}_{a,l})^2]$, the expectation is taken over $C, \mathcal{S}$, and the timing error of the L DTs $N_1, \ldots, N_L$. The generalized error can be decomposed similarly to (6.16) and (6.20) as follows:

$$\begin{aligned} & E\big[\big(C - \frac{1}{L}\sum_{l=1}^{L}\hat{Y}_{a,l}\big)^2\big] \\ = \; & \sigma_C^2 + b_{RF}^2 + \sigma_{RF}^2 \end{aligned}$$

where $\sigma_C^2 = E\big[\big(C - E[C]\big)^2\big]$ is the noise term same as the first term in (6.16). Assuming $\hat{Y}_{a,l}$ are i.i.d with the same distribution as in the single DT $\hat{Y}_a$, the bias term b_{RF}^2 can be simplified into:

$$\begin{aligned} b_{RF}^2 &= \big(E[C] - E[\frac{1}{L}\sum_{l=1}^{L}\hat{Y}_{a,l}]\big)^2 \\ &= \big(E[C] - E[\hat{Y}_a]\big)^2 \end{aligned}$$

which is the same as the first term in (6.20), and σ_{RF}^2 can be simplified as follows:

$$\begin{aligned} \sigma_{RF}^2 &= E\Big[\big(E[\frac{1}{L}\sum_{l=1}^{L}\hat{Y}_{a,l}] - \frac{1}{L}\sum_{l=1}^{L}\hat{Y}_{a,l}\big)^2\Big] \\ &= \frac{1}{L^2}E\Big[\big(\sum_{l=1}^{L}(E[\hat{Y}_{a,l}] - \hat{Y}_{a,l})\big)^2\Big] \qquad (6.22) \\ &= \frac{1}{L^2}\sum_{l=1}^{L}E\Big[\big(E[\hat{Y}_{a,l}] - \hat{Y}_{a,l}\big)^2\Big] \qquad (6.23) \\ &= \frac{1}{L}\sigma_{\hat{Y}_a}^2 \qquad (6.24) \end{aligned}$$

where (6.22) to (6.23) comes from the assumption on the independence of $\hat{Y}_{a,l}$. This completes the proof of (6.10).

Chapter 7

CONCLUSION AND FUTURE WORK

Moving towards the age of ubiquitous computing, ULP platforms for in-silicon machine learning will be the key enabler for pervasive intelligence in our daily lives. The quest for energy efficient in-silicon machine learning is made challenging due to the energy delivery, communication, computation and robustness challenges. This dissertation explores design approaches that represent a radical change from the conventional design methodology by embedding computation into the energy delivery, sensing, and emerging stochastic fabrics.

7.1 Dissertation Contributions

The Compute Voltage Regulator Module (C-VRM) has been proposed to embed information processing into the energy delivery subsystem. The C-VRM eliminates the loss associated with conventional switched capacitor based energy delivery circuits. Measured results of a prototype IC show more than 40% savings in system-level energy per operation, and an efficiency ranging from 79% to 83%.

The Compute Sensor approach has been proposed to embed information processing and learning into the sensing front-end. The Compute Sensor eliminates both the traditional sensor-processor interface, and the high-SNR/high-energy digital processing by moving feature extraction and classification functions into the analog domain in close proximity to the APS array. The Compute Sensor designed in 65 nm CMOS is shown to achieve a detection accuracy greater than 94.7% using the Caltech101 dataset [80], which is within 0.5% of that achieved by an ideal digital implementation. The performance is achieved with 7× to 17× lower energy than the conventional architecture for the same level of accuracy.

This dissertation proposes a new SEC technique well-suited for machine learning appli-

cations, i.e., embedded algorithmic-noise tolerance (E-ANT). E-ANT at the architectural and algorithmic level are applied to EEG seizure detection systems. Simulation results in a commercial 45 nm CMOS process show that ARCH-ANT can compensate for error rates up to 0.38, and ALG-ANT can compensate for error rates up to 0.41, while maintaining a true positive rate $p_{tp} > 0.9$ and a false positive rate $p_{fp} \leq 0.01$. This error tolerance is employed to reduce energy via the use of voltage overscaling (VOS). ARCH-ANT and ALG-ANT are able to achieve up to 51% and 44% energy savings, respectively.

This dissertation makes contribution in theoretical analysis of SEC by proposing a class of probabilistic error models that can accurately model the error distribution on the noisy hardware fabrics. The models are validated in a commercial 45 nm CMOS process and employed to evaluate the performance of machine learning kernels in presence of hardware errors. Performance prediction of a support vector machine (SVM) based classifier using these models indicates that when comparing Monte Carlo with HDL simulations, probability of detection P_{det} estimated using the model is within 3% for VOS error when the error rate $p_\eta \leq 80\%$, within 5% for process variation error and within 2% for defect errors when the defect rate (the percentage of circuit nets subject to stuck-at-faults) p_{saf} is between 10^{-3} and 0.2.

Finally, SEC techniques have been extended into distributed machine learning algorithms by proposing error resilient classifier ensemble. Employing the breast cancer data set in the UCI machine learning repository and architectural-level error models in a commercial 45 nm CMOS process, it is determined that RF-based architectures are significantly more robust than SVM architectures in presence of timing errors due to process variations in near-threshold voltage (NTV) regions ($0.3\,\mathrm{V} - 0.7\,\mathrm{V}$). Additionally, an error weighted voting technique that incorporates the timing error statistics of the NTV circuit fabric is proposed to further enhance the robustness of RF architectures. Simulation results confirm that the error weighted voting achieves a P_{det} that varies by only 1.4%, which is 12× lower compared to SVM.

7.2 Future Work

7.2.1 System Design

System optimization and design space exploration play a crucial role in designing energy efficient and robust ULP platforms for in-silicon machine learning. To date, an ad-hoc approach is adopted for the design space exploration. The search for energy minimum realization of in-silicon machine learning systems needs to be done in a systematic manner, much as it is done in the area of low power signal processing and communication systems and ICs [136, 29]. Many machine learning algorithms are inherently formulated as optimization problems, i.e., minimizing certain loss functions subject to constraints such as the form of the functions (linear, quadratic, multi-layer, etc), the sparsity of solutions, and others. This optimization formulation opens many opportunities where the architecture/circuit level performance metrics can be incorporated so as to introduce new constraints in the original optimization problem. Such a resource constrained optimization allows the systematic design of ULP platforms for machine learning. In fact, the ALG-ANT based FIR filter in Chapter 4 is one example where we introduce additional constraints during the WLS filter optimization to enhance the robustness of the design. Some possible constraints to consider are the precision requirements in the data path, the error rate/error statistics in the underlying circuit fabric, the storage/communication cost for data transfer, and others.

7.2.2 Architecture

New architectures are needed for energy minimum realization of in-silicon machine learning systems. Conventionally, the design of in-silicon machine learning is treated as yet another problem of efficient computing. As a result, reliable circuit operations are guaranteed by adopting a worst case design methodology and use of large design margins, limiting the achievable energy efficiency. In contrast, new architectures should embrace both the probabilistic nature of the performance metric, and the statistical behavior in the computing fabric. To this end, the architectures explored in this dissertation offer several future directions to be extended.

In-sensor machine learning points to an opportunity to achieve large energy savings by the elimination of interface overhead. At the same time, the non-idealities resulting from mixed signal implementation require error resilient computing. The proposed Compute Sensor offers a general framework to map different algorithms such as decision trees, kernel SVMs, and ensembles of these classifiers, because the bit-line and cross bit-line processors perform element-wise and dimensionality reduction operations which are common in many of these algorithms. Feature selection techniques, such as the focus-of-attention mechanisms used in many vision algorithms [137], can also be embedded into the sensor. These techniques employ a cascade of simple classifiers, and can rapidly discard images/samples that are not informative. Therefore, further energy savings can be achieved by avoiding the access of non-informative pixels. In addition, programmable in-sensor machine learning architectures can also be explored to configure the bit-line and cross bit-line processors to achieve various inference tasks. Another interesting direction is the combination of in-sensor and recently proposed in-memory computing [82] architectures so that inference task is partitioned between the two systems, completely eliminating the need for conventional von Neumann architectures.

Machine learning in stochastic fabrics, such as those operating with voltage scaling, process variation or defects, offers another opportunity where architectural design can be explored to tolerate device/circuit errors and to enhance energy efficiency. Many machine learning algorithms are iterative and possess inherent tolerance for certain error statistics. New SEC techniques can be explored for in-silicon machine learning systems. Error compensation such as retraining can also be explored to develop architectures that have on-line training capabilities. Such architectures are of great interest in applications where the data statistics are time varying.

7.2.3 Circuit and Devices

Circuit level techniques should be explored in combination with the architectural level techniques. Specifically, machine learning algorithms offer relaxed precision/linearity requirement for the underlying circuit. This new degree of freedom can be explored to design

circuits that have tolerable non-ideality, but at the same time offer large energy benefits. Circuit level techniques that can engineer the error statistics to favor error compensation should also be explored for wide application of SEC techniques. At the device level, many emerging devices such as CNFET [85], spin devices [86], and others have the potential for large energy saving or density improvement, but suffer from various hardware errors such as defects and noise. This inherent statistical behavior offers an opportunity to employ SEC to design reliable systems on emerging beyond CMOS technologies.

7.2.4 Theoretical Limits

Theoretical foundations for in-silicon machine learning need to be investigated. The system performance in presence of errors/non-idealities should be further analyzed to reveal the impact of different error statistics. Performance bounds when hardware constraints are incorporated need to be derived to guide the design of resource-constrained machine learning systems. Such theoretical limits will provide guidance in the quest for energy efficient realizations of in-silicon machine learning systems.

REFERENCES

[1] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "Imagenet large scale visual recognition challenge," *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, 2015. [Online]. Available: http://dx.doi.org/10.1007/s11263-015-0816-y

[2] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis, "Mastering the game of Go with deep neural networks and tree search," *Nature*, vol. 529, pp. 484–503, 2016. [Online]. Available: http://www.nature.com/nature/journal/v529/n7587/full/nature16961.html

[3] J. Baliga, R. W. A. Ayre, K. Hinton, and R. S. Tucker, "Green cloud computing: Balancing energy in processing, storage, and transport," *Proceedings of the IEEE*, vol. 99, no. 1, pp. 149–167, Jan 2011.

[4] M. K. Weldon, *The Future X Network A Bell Labs Perspective*. CRC Press, 2016.

[5] Y. Ramadass and A. Chandrakasan, "Voltage scalable switched capacitor DC-DC converter for ultra-low-power on-chip applications," in *Power Electronics Specialists Conference, 2007. PESC 2007. IEEE*, June 2007, pp. 2353–2359.

[6] J. Choi, S. Park, J. Cho, and E. Yoon, "An energy/illumination-adaptive CMOS image sensor with rconfigurable modes of operations," *IEEE Journal of Solid-State Circuits*, vol. 50, no. 6, pp. 1438–1450, June 2015.

[7] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems 25*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2012, pp. 1097–1105. [Online]. Available: http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf

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

[9] "International technology roadmap 2011 edition," ITRS, Tech. Rep., 2011. [Online]. Available: http://www.itrs.net/Links/2011ITRS/2011Chapters/2011SysDrivers.pdf

[10] S. Hanson, B. Zhai, D. Blaauw, D. Sylvester, A. Bryant, and X. Wang, "Energy optimality and variability in subthreshold design," in *Low Power Electronics and Design, 2006. ISLPED'06. Proceedings of the 2006 International Symposium on*, Oct 2006, pp. 363–365.

[11] K. Tsuji, K. Terada, R. Takeda, T. Tsunomura, A. Nishida, and T. Mogami, "Threshold voltage variation extracted from MOSFET C-V curves by charge-based capacitance measurement," in *Microelectronic Test Structures (ICMTS), 2012 IEEE International Conference on*, March 2012, pp. 82–86.

[12] Y. Ramadass, A. Fayed, B. Haroun, and A. Chandrakasan, "A 0.16mm2 completely on-chip switched-capacitor DC-DC converter using digital capacitance modulation for LDO replacement in 45nm CMOS," in *Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2010 IEEE International*, 2010, pp. 208–209.

[13] M. Wieckowski, G. K. Chen, M. Seok, D. Blaauw, and D. Sylvester, "A hybrid DC-DC converter for sub-microwatt sub-1V implantable applications," in *VLSI Circuits, 2009 Symposium on*, June 2009, pp. 166 –167.

[14] S. Rajapandian, K. Shepard, P. Hazucha, and T. Karnik, "High-tension power delivery: operating 0.18 mu;m CMOS digital logic at 5.4V," in *Solid-State Circuits Conference, 2005. Digest of Technical Papers. ISSCC. 2005 IEEE International*, 2005, pp. 298–599.

[15] M. Seeman and S. Sanders, "Analysis and optimization of switched-capacitor DC-DC converters," in *Computers in Power Electronics, 2006. COMPEL '06. IEEE Workshops on*, July 2006, pp. 216 –224.

[16] A. Nilchi, J. Aziz, and R. Genov, "Focal-plane algorithmically-multiplying CMOS computational image sensor," *IEEE Journal of Solid-State Circuits*, vol. 44, no. 6, pp. 1829–1839, June 2009.

[17] P. Dudek and P. J. Hicks, "A general-purpose processor-per-pixel analog SIMD vision chip," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 52, no. 1, pp. 13–20, Jan 2005.

[18] N. Massari, M. Gottardi, L. Gonzo, D. Stoppa, and A. Simoni, "A CMOS image sensor with programmable pixel-level analog processing," *IEEE Transactions on Neural Networks*, vol. 16, no. 6, pp. 1673–1684, Nov 2005.

[19] R. Robucci, J. D. Gray, L. K. Chiu, J. Romberg, and P. Hasler, "Compressive sensing on a CMOS separable-transform image sensor," *Proceedings of the IEEE*, vol. 98, no. 6, pp. 1089–1101, June 2010.

[20] A. Bandyopadhyay, J. Lee, R. W. Robucci, and P. Hasler, "MATIA: A programmable 80 μw/frame CMOS block matrix transform imager architecture," *IEEE Journal of Solid-State Circuits*, vol. 41, no. 3, pp. 663–672, March 2006.

[21] J. Fernandez-Berni, R. Carmona-Galan, and L. Carranza-Gonzalez, "FLIP-Q: A QCIF resolution focal-plane array for low-power image processing," *IEEE Journal of Solid-State Circuits*, vol. 46, no. 3, pp. 669–680, March 2011.

[22] Z. Lin, M. W. Hoffman, N. Schemm, W. D. Leon-Salas, and S. Balkir, "A CMOS image sensor for multi-level focal plane image decomposition," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 55, no. 9, pp. 2561–2572, Oct 2008.

[23] A. Graupner, J. Schreiter, S. Getzlaff, and R. Schuffny, "CMOS image sensor with mixed-signal processor array," *IEEE Journal of Solid-State Circuits*, vol. 38, no. 6, pp. 948–957, June 2003.

[24] Y. Oike and A. E. Gamal, "CMOS image sensor with per-column ADC and programmable compressed sensing," *IEEE Journal of Solid-State Circuits*, vol. 48, no. 1, pp. 318–328, Jan 2013.

[25] Y. Ni and J. Guan, "A 256 x 256 pixel smart CMOS image sensor for line-based stereo vision applications," *IEEE Journal of Solid-State Circuits*, vol. 35, no. 7, pp. 1055–1061, July 2000.

[26] S. Martin, K. Flautner, T. Mudge, and D. Blaauw, "Combined dynamic voltage scaling and adaptive body biasing for lower power microprocessors under dynamic workloads," in *Computer Aided Design, 2002. ICCAD 2002. IEEE/ACM International Conference on*, Nov 2002, pp. 721–725.

[27] M. Borah, R. Owens, and M. Irwin, "Transistor sizing for low power CMOS circuits," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 15, no. 6, pp. 665–671, Jun 1996.

[28] A. Chandrakasan, S. Sheng, and R. Brodersen, "Low-power CMOS digital design," *Solid-State Circuits, IEEE Journal of*, vol. 27, no. 4, pp. 473–484, Apr 1992.

[29] M. Goel and N. Shanbhag, "Dynamic algorithm transforms for low-power reconfigurable adaptive equalizers," in *Signal Processing, IEEE Transactions on*, vol. 47, no. 10, Oct 1999, pp. 2821–2832.

[30] M. Donno, A. Ivaldi, L. Benini, and E. Macii, "Clock-tree power optimization based on RTL clock-gating," in *Design Automation Conference, 2003. Proceedings*, June 2003, pp. 622–627.

[31] S.-H. Chen and J.-Y. Lin, "Implementation and verification practices of DVFS and power gating," in *VLSI Design, Automation and Test, 2009. VLSI-DAT '09. International Symposium on*, April 2009, pp. 19–22.

[32] W. Kim, M. Gupta, G.-Y. Wei, and D. Brooks, "System level analysis of fast, per-core DVFS using on-chip switching regulators," in *High Performance Computer Architecture, 2008. HPCA 2008. IEEE 14th International Symposium on*, Feb 2008, pp. 123–134.

[33] B. Calhoun, A. Wang, and A. Chandrakasan, "Modeling and sizing for minimum energy operation in subthreshold circuits," *Solid-State Circuits, IEEE Journal of*, vol. 40, no. 9, pp. 1778–1786, Sept 2005.

[34] B. Zhai, D. Blaauw, D. Sylvester, and K. Flautner, "Theoretical and practical limits of dynamic voltage scaling," in *Design Automation Conference, 2004. Proceedings. 41st*, July 2004, pp. 868 –873.

[35] D. Markovic, C. C. Wang, L. P. Alarcon, T. T. Liu, and J. M. Rabaey, "Ultralow-power design in near-threshold region," *Proceedings of the IEEE*, vol. 98, no. 2, pp. 237–252, Feb 2010.

[36] R. Dreslinski, B. Zhai, T. Mudge, D. Blaauw, and D. Sylvester, "An energy efficient parallel architecture using near threshold operation," in *Parallel Architecture and Compilation Techniques, 2007. PACT 2007. 16th International Conference on*, Sept 2007, pp. 175–188.

[37] J. Y. S. Kwong, "A sub-threshold cell library and methodology," M.S. thesis, Massachusetts Institute of Technology, Cambridge, 2006.

[38] B. Zhai, S. Pant, L. Nazhandali, S. Hanson, J. Olson, A. Reeves, M. Minuth, R. Helfand, T. Austin, D. Sylvester, and D. Blaauw, "Energy-efficient subthreshold processor design," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 17, no. 8, pp. 1127–1137, Aug 2009.

[39] S. Luetkemeier, T. Jungeblut, M. Porrmann, and U. Rueckert, "A 200mV 32b subthreshold processor with adaptive supply voltage control," in *Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2012 IEEE International*, Feb 2012, pp. 484–486.

[40] J. Kwong, Y. Ramadass, N. Verma, M. Koesler, K. Huber, H. Moormann, and A. Chandrakasan, "A 65nm sub-Vt microcontroller with integrated SRAM and switched-capacitor DC-DC converter," in *Solid-State Circuits Conference, 2008. ISSCC 2008. Digest of Technical Papers. IEEE International*, Feb 2008, pp. 318–616.

[41] S. Vangal, S. Jain, and V. De, "A solar-powered 280mV-to-1.2V wide-operating-range IA-32 processor," in *IC Design Technology (ICICDT), 2014 IEEE International Conference on*, May 2014, pp. 1–4.

[42] A. Wang and A. Chandrakasan, "A 180-mV subthreshold FFT processor using a minimum energy design methodology," *Solid-State Circuits, IEEE Journal of*, vol. 40, no. 1, pp. 310–319, Jan 2005.

[43] N. Reynders and W. Dehaene, "Variation-resilient sub-threshold circuit solutions for ultra-low-power digital signal processors with 10MHz clock frequency," in *ESSCIRC (ESSCIRC), 2012 Proceedings of the*, Sept 2012, pp. 474–477.

[44] I. Koren and Z. Koren, "Defect tolerance in VLSI circuits: techniques and yield analysis," *Proceedings of the IEEE*, vol. 86, no. 9, pp. 1819–1838, Sep 1998.

[45] J. von Neumann, "Probabilistic logics and the synthesis of reliable organisms from unreliable components," *Automata Studies*, pp. 43–98, 1956.

[46] E. Huijbregts, H. Xue, and J. Jess, "Routing for reliable manufacturing," *Semiconductor Manufacturing, IEEE Transactions on*, vol. 8, no. 2, pp. 188–194, May 1995.

[47] Z. Koren and I. Koren, "On the effect of floorplanning on the yield of large area integrated circuits," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 5, no. 1, pp. 3–14, March 1997.

[48] K. Nepal, R. Bahar, J. Mundy, W. Patterson, and A. Zaslavsky, "Designing logic circuits for probabilistic computation in the presence of noise," in *Design Automation Conference, 2005. Proceedings. 42nd*, June 2005, pp. 485–490.

[49] N. Jayakumar and S. Khatri, "A variation-tolerant sub-threshold design approach," in *Design Automation Conference, 2005. Proceedings. 42nd*, June 2005, pp. 716–719.

[50] M. Zhang and N. Shanbhag, "A CMOS design style for logic circuit hardening," in *Reliability Physics Symposium, 2005. Proceedings. 43rd Annual. 2005 IEEE International*, April 2005, pp. 223–229.

[51] D. Ernst, N. S. Kim, S. Das, S. Pant, R. Rao, T. Pham, C. Ziesler, D. Blaauw, T. Austin, K. Flautner, and T. Mudge, "Razor: a low-power pipeline based on circuit-level timing speculation," in *Microarchitecture, 2003. MICRO-36. Proceedings. 36th Annual IEEE/ACM International Symposium on*, Dec 2003, pp. 7–18.

[52] N. Jayakumar and S. Khatri, "A variation-tolerant sub-threshold design approach," in *Design Automation Conference, 2005. Proceedings. 42nd*, June 2005, pp. 716–719.

[53] N. Shanbhag, R. Abdallah, R. Kumar, and D. Jones, "Stochastic computation," in *Design Automation Conference (DAC), 2010 47th ACM/IEEE*, June 2010, pp. 859–864.

[54] 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 *Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on*, March 2012, pp. 5285–5288.

[55] S. Zhang and N. Shanbhag, "Embedded error compensation for energy efficient DSP systems," in *Signal and Information Processing (GlobalSIP), 2014 IEEE Global Conference on*, Dec 2014, pp. 30–34.

[56] Z. Wang, R. Schapire, and N. Verma, "Error-adaptive classifier boosting (EACB): Exploiting data-driven training for highly fault-tolerant hardware," in *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*, May 2014, pp. 3884–3888.

[57] B. Shim, S. Sridhara, and N. Shanbhag, "Reliable low-power digital signal processing via reduced precision redundancy," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 12, no. 5, pp. 497–510, May 2004.

[58] G. Varatkar, S. Narayanan, N. Shanbhag, and D. Jones, "Variation-tolerant, low-power PN-code acquisition using stochastic sensor NOC," in *Circuits and Systems, 2008. ISCAS 2008. IEEE International Symposium on*, May 2008, pp. 380–383.

[59] E. Kim and N. Shanbhag, "Soft N-Modular Redundancy," *Computers, IEEE Transactions on*, vol. 61, no. 3, pp. 323–336, March 2012.

[60] R. Abdallah and N. Shanbhag, "Robust and energy-efficient DSP systems via output probability processing," in *Computer Design (ICCD), 2010 IEEE International Conference on*, Oct 2010, pp. 38–44.

[61] E. Kim and N. Shanbhag, "Soft NMR: Analysis and application to DSP systems," in *Acoustics Speech and Signal Processing (ICASSP), 2010 IEEE International Conference on*, March 2010, pp. 1494–1497.

[62] K. Bowman, J. Tschanz, S. Lu, P. Aseron, M. Khellah, A. Raychowdhury, B. Geuskens, C. Tokunaga, C. Wilkerson, T. Karnik, and V. De, "A 45 nm resilient microprocessor core for dynamic variation tolerance," *Solid-State Circuits, IEEE Journal of*, vol. 46, no. 1, pp. 194–208, Jan 2011.

[63] E. Kim, D. Baker, S. Narayanan, N. Shanbhag, and D. Jones, "A 3.6-mW 50-MHz PN code acquisition filter via statistical error compensation in 180-nm CMOS," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. PP, no. 99, pp. 1–1, 2014.

[64] R. Abdallah and N. Shanbhag, "An energy-efficient ECG processor in 45-nm CMOS using statistical error compensation," *Solid-State Circuits, IEEE Journal of*, vol. 48, no. 11, pp. 2882–2893, Nov 2013.

[65] J. Choi, E. Kim, R. Rutenbar, and N. Shanbhag, "Error resilient MRF message passing architecture for stereo matching," in *Signal Processing Systems (SiPS), 2013 IEEE Workshop on*, Oct 2013, pp. 348–353.

[66] G. Chen, M. Fojtik, D. Kim, D. Fick, J. Park, M. Seok, M.-T. Chen, Z. Foo, D. Sylvester, and D. Blaauw, "Millimeter-scale nearly perpetual sensor system with stacked battery and solar cells," in *Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2010 IEEE International*, Feb 2010, pp. 288–289.

[67] J. Kwong, Y. Ramadass, N. Verma, and A. Chandrakasan, "A 65 nm sub-Vt microcontroller with integrated SRAM and switched capacitor DC-DC converter," *Solid-State Circuits, IEEE Journal of*, vol. 44, no. 1, pp. 115 –126, Jan. 2009.

[68] M. Stojcev, M. Kosanovic, and L. Golubovic, "Power management and energy harvesting techniques for wireless sensor nodes," in *Telecommunication in Modern Satellite, Cable, and Broadcasting Services, 2009. TELSIKS '09. 9th International Conference on*, Oct. 2009, pp. 65 –72.

[69] J. Kimball and P. Krein, "Analysis and design of switched capacitor converters," in *Applied Power Electronics Conference and Exposition, 2005. APEC 2005. Twentieth Annual IEEE*, vol. 3, 2005, pp. 1473–1477 Vol. 3.

[70] M. Evzelman and S. Ben-Yaakov, "Average modeling technique for switched capacitor converters including large signal dynamics and small signal responses," in *Microwaves, Communications, Antennas and Electronics Systems (COMCAS), 2011 IEEE International Conference on*, Nov 2011, pp. 1–5.

[71] M. Evzelman and S. Ben-Yaakov, "Optimal switch resistances in switched capacitor converters," in *Electrical and Electronics Engineers in Israel (IEEEI), 2010 IEEE 26th Convention of*, Nov 2010, pp. 000 436–000 439.

[72] S. Ben-Yaakov and M. Evzelman, "Generic and unified model of Switched Capacitor Converters," in *Energy Conversion Congress and Exposition, 2009. ECCE 2009. IEEE*, 2009, pp. 3501–3508.

[73] R. Aparicio and A. Hajimiri, "Capacity limits and matching properties of integrated capacitors," *Solid-State Circuits, IEEE Journal of*, vol. 37, no. 3, pp. 384 –393, Mar 2002.

[74] B. Calhoun, A. Wang, and A. Chandrakasan, "Modeling and sizing for minimum energy operation in subthreshold circuits," *Solid-State Circuits, IEEE Journal of*, vol. 40, no. 9, pp. 1778–1786, Sept 2005.

[75] D. El-Damak, S. Bandyopadhyay, and A. Chandrakasan, "A 93% efficiency reconfigurable switched-capacitor DC-DC converter using on-chip ferroelectric capacitors," in *Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2013 IEEE International*, Feb 2013, pp. 374–375.

[76] M. Seeman and R. Jain, "Single-bound hysteretic regulation of switched-capacitor converters," Mar. 31 2011, US Patent App. 12/566,730. [Online]. Available: http://www.google.com/patents/US20110074371

[77] N. Krihely, S. Ben-Yaakov, and A. Fish, "Efficiency optimization of a step-down switched capacitor converter for subthreshold," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 21, no. 12, pp. 2353–2357, 2013.

[78] J. Karhunen and J. Joutsensalo, "Generalizations of principal component analysis, optimization problems, and neural networks," *Neural Networks*, vol. 8, no. 4, pp. 549 – 562, 1995.

[79] V. Vapnik, "An overview of statistical learning theory," *Neural Networks, IEEE Transactions on*, vol. 10, no. 5, pp. 988–999, Sep 1999.

[80] L. Fei-Fei, R. Fergus, and P. Perona, "Learning generative visual models from few training examples: an incremental bayesian approach tested on 101 object categories," in *Computer Vision and Pattern Recognition Workshop, 2004. CVPRW '04. Conference on*, June 2004, pp. 178–178.

[81] H.-S. P. Wong, "CMOS image sensors-recent advances and device scaling considerations," in *Electron Devices Meeting, 1997. IEDM '97. Technical Digest., International*, Dec 1997, pp. 201–204.

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

[83] M. Horowitz, "Computing's energy problem (and what we can do about it)," in *2014 IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*, Feb 2014, pp. 10–14.

[84] R. Dreslinski, M. Wieckowski, D. Blaauw, D. Sylvester, and T. Mudge, "Near-Threshold Computing: Reclaiming Moore's Law Through Energy Efficient Integrated Circuits," *Proceedings of the IEEE*, vol. 98, no. 2, pp. 253–266, Feb 2010.

[85] R. Martel, V. Derycke, J. Appenzeller, S. Wind, and P. Avouris, "Carbon nanotube field-effect transistors and logic circuits," in *Design Automation Conference, 2002. Proceedings. 39th*, 2002, pp. 94–98.

[86] K. Roy, M. Sharad, D. Fan, and K. Yogendra, "Beyond charge-based computation: Boolean and non-Boolean computing with spin torque devices," in *Low Power Electronics and Design (ISLPED), 2013 IEEE International Symposium on*, Sept 2013, pp. 139–142.

[87] V. Gupta, D. Mohapatra, S. P. Park, A. Raghunathan, and K. Roy, "IMPACT: IMPrecise adders for low-power approximate computing," in *Low Power Electronics and Design (ISLPED) 2011 International Symposium on*, Aug 2011, pp. 409–414.

[88] S. Venkataramani, A. Sabne, V. Kozhikkottu, K. Roy, and A. Raghunathan, "SALSA: Systematic logic synthesis of approximate circuits," in *Design Automation Conference (DAC), 2012 49th ACM/EDAC/IEEE*, June 2012, pp. 796–801.

[89] S. Venkataramani, K. Roy, and A. Raghunathan, "Substitute-and-simplify: A unified design paradigm for approximate and quality configurable circuits," in *Design, Automation Test in Europe Conference Exhibition (DATE), 2013*, March 2013, pp. 1367–1372.

[90] X. Zhu and X. Wu, "Class noise vs. attribute noise: A quantitative study of their impacts," *Artif. Intell. Rev.*, vol. 22, no. 3, pp. 177–210, Nov. 2004. [Online]. Available: http://dx.doi.org/10.1007/s10462-004-0751-8

[91] A. Atla, R. Tada, V. Sheng, and N. Singireddy, "Sensitivity of different machine learning algorithms to noise," *J. Comput. Sci. Coll.*, vol. 26, no. 5, pp. 96–103, May 2011. [Online]. Available: http://dl.acm.org/citation.cfm?id=1961574.1961594

[92] C. M. Bishop, "Training with noise is equivalent to Tikhonov regularization," *Neural Comput.*, vol. 7, no. 1, pp. 108–116, Jan. 1995. [Online]. Available: http://dx.doi.org/10.1162/neco.1995.7.1.108

[93] Y. Raviv and N. Intrator, "Bootstrapping with noise: An effective regularization technique," *Connection Science*, vol. 8, pp. 355–372, 1996.

[94] J. Mengte, A. Raghunathan, S. Chakradhar, and S. Byna, "Exploiting the forgiving nature of applications for scalable parallel execution," in *Parallel Distributed Processing (IPDPS), 2010 IEEE International Symposium on*, April 2010, pp. 1–12.

[95] S. Venkataramani, A. Raghunathan, J. Liu, and M. Shoaib, "Scalable-effort classifiers for energy-efficient machine learning," in *Proceedings of the 52Nd Annual Design Automation Conference*, ser. DAC '15. New York, NY, USA: ACM, 2015. [Online]. Available: http://doi.acm.org/10.1145/2744769.2744904 pp. 67:1–67:6.

[96] S. Hashemi, R. I. Bahar, and S. Reda, "Drum: A dynamic range unbiased multiplier for approximate applications," in *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, ser. ICCAD '15. Piscataway, NJ, USA: IEEE Press, 2015. [Online]. Available: http://dl.acm.org/citation.cfm?id=2840819.2840878 pp. 418–425.

[97] S. Borkar, T. Karnik, S. Narendra, J. Tschanz, A. Keshavarzi, and V. De, "Parameter variations and impact on circuits and microarchitecture," in *Design Automation Conference, 2003. Proceedings*, June 2003, pp. 338–342.

[98] K. Nepal, R. Bahar, J. Mundy, W. Patterson, and A. Zaslavsky, "Techniques for designing noise-tolerant multi-level combinational circuits," in *Design, Automation Test in Europe Conference Exhibition, 2007. DATE '07*, April 2007, pp. 1–6.

[99] D. Ernst, S. Das, S. Lee, D. Blaauw, T. Austin, T. Mudge, N. S. Kim, and K. Flautner, "RAZOR: circuit-level correction of timing errors for low-power operation," *Micro, IEEE*, vol. 24, no. 6, pp. 10–20, Nov 2004.

[100] D. Blaauw, S. Kalaiselvan, K. Lai, W.-H. Ma, S. Pant, C. Tokunaga, S. Das, and D. Bull, "RAZOR II: In situ error detection and correction for PVT and SER tolerance," in *Solid-State Circuits Conference, 2008. ISSCC 2008. Digest of Technical Papers. IEEE International*, Feb 2008, pp. 400–622.

[101] J. Tschanz, K. Bowman, C. Wilkerson, S.-L. Lu, and T. Karnik, "Resilient circuits; Enabling energy-efficient performance and reliability," in *Computer-Aided Design - Digest of Technical Papers, 2009. ICCAD 2009. IEEE/ACM International Conference on*, Nov 2009, pp. 71–73.

[102] C.-H. Chen, D. Blaauw, D. Sylvester, and Z. Zhang, "Design and evaluation of confidence-driven error-resilient systems," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 22, no. 8, pp. 1727–1737, Aug 2014.

[103] R. Abdallah and N. Shanbhag, "An energy-efficient ECG processor in 45-nm CMOS using statistical error compensation," *Solid-State Circuits, IEEE Journal of*, vol. 48, no. 11, pp. 2882–2893, Nov 2013.

[104] M. Chen, J. Han, Y. Zhang, Y. Zou, Y. Li, and X. Zeng, "An error-resilient wavelet-based ECG processor under voltage overscaling," in *Biomedical Circuits and Systems Conference (BioCAS), 2014 IEEE*, Oct 2014, pp. 628–631.

[105] J. Yoo, L. Yan, D. El-Damak, M. Altaf, A. Shoeb, and A. Chandrakasan, "An 8-channel scalable EEG acquisition SoC with patient-specific seizure classification and recording processor," *Solid-State Circuits, IEEE Journal of*, vol. 48, no. 1, pp. 214–228, Jan 2013.

[106] C. J. C. Burges, "A tutorial on support vector machines for pattern recognition," *Data Mining and Knowledge Discovery*, vol. 2, pp. 121–167, 1998.

[107] K. K. Parhi, *VLSI Digital Signal Processing Systems: Design and Implementation*. New York: Wiley, 1999, a Wiley-Interscience publication. [Online]. Available: http://opac.inria.fr/record=b1097682

[108] M. T. Heath, *Scientific Computing: An Introductory Survey*, 2nd ed., E. M. Munson, Ed. McGraw-Hill Higher Education, 1996.

[109] A. Ganapathiraju, J. Hamaker, and J. Picone, "Applications of support vector machines to speech recognition," *Signal Processing, IEEE Transactions on*, vol. 52, no. 8, pp. 2348–2355, Aug 2004.

[110] H. Hu, M.-X. Xu, and W. Wu, "GMM supervector based SVM with spectral features for speech emotion recognition," in *Acoustics, Speech and Signal Processing, 2007. ICASSP 2007. IEEE International Conference on*, vol. 4, April 2007, pp. IV–413–IV–416.

[111] S. Lawrence, C. Giles, A. C. Tsoi, and A. Back, "Face recognition: A convolutional neural-network approach," *Neural Networks, IEEE Transactions on*, vol. 8, no. 1, pp. 98–113, Jan 1997.

[112] D. Wedge, D. Ingram, D. McLean, and Z. Bandar, "On global-local artificial neural networks for function approximation," *Neural Networks, IEEE Transactions on*, vol. 17, no. 4, pp. 942–952, July 2006.

[113] A. V. Oppenheim, R. W. Schafer, and J. R. Buck, *Discrete-time Signal Processing (2Nd Ed.)*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1999.

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

[115] K. Lee, S.-Y. Kung, and N. Verma, "Low-energy formulations of support vector machine kernel functions for biomedical sensor applications," *Journal of Signal Processing Systems*, vol. 69, no. 3, pp. 339–349, 2012.

[116] M. Abbas, M. Ikeda, and K. Asada, "Statistical model for logic errors in CMOS digital circuits for reliability-driven design flow," in *Design and Diagnostics of Electronic Circuits and Systems, 2006 IEEE*, April 2006, pp. 145–146.

[117] K. Lingasubramanian and S. Bhanja, "An error model to study the behavior of transient errors in sequential circuits," in *VLSI Design, 2009 22nd International Conference on*, Jan 2009, pp. 485–490.

[118] Y. Liu, T. Zhang, and K. Parhi, "Computation error analysis in digital signal processing systems with overscaled supply voltage," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 18, no. 4, pp. 517–526, April 2010.

[119] L. Li and H. Zhou, "On error modeling and analysis of approximate adders," in *Computer-Aided Design (ICCAD), 2014 IEEE/ACM International Conference on*, Nov 2014, pp. 511–518.

[120] J. Huang and J. Lach, "Exploring the fidelity-efficiency design space using imprecise arithmetic," in *Design Automation Conference (ASP-DAC), 2011 16th Asia and South Pacific*, Jan 2011, pp. 579–584.

[121] W.-T. Chan, A. Kahng, S. Kang, R. Kumar, and J. Sartori, "Statistical analysis and modeling for error composition in approximate computation circuits," in *Computer Design (ICCD), 2013 IEEE 31st International Conference on*, Oct 2013, pp. 47–53.

[122] K. Iwasaki and F. Arakawa, "An analysis of the aliasing probability of multiple-input signature registers in the case of a 2m-ary symmetric channel," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 9, no. 4, pp. 427–438, Apr 1990.

[123] T. Williams, W. Daehn, M. Gruetzner, and C. Starke, "Aliasing errors in signature in analysis registers," *Design Test of Computers, IEEE*, vol. 4, no. 2, pp. 39–45, April 1987.

[124] D. N. A. Asuncion, "UCI machine learning repository," 2007. [Online]. Available: http://www.ics.uci.edu/∼mlearn/MLRepository.html

[125] J. Lin, "Divergence measures based on the Shannon entropy," *Information Theory, IEEE Transactions on*, vol. 37, no. 1, pp. 145–151, Jan 1991.

[126] J. Macke, P. Berens, A. Ecker, A. Tolias, and M. Bethge, "Generating spike trains with specified correlation coefficients," *Neural Computation*, vol. 21, no. 2, pp. 397–423, Feb 2009.

[127] S. Zhang and N. Shanbhag, "Probabilistic error models for machine learning kernels implemented on stochastic nanoscale fabrics," in *Design, Automation Test in Europe Conference Exhibition, 2016. DATE '16*, March 2016.

[128] T. G. Dietterich, "Ensemble methods in machine learning," in *Proceedings of the First International Workshop on Multiple Classifier Systems*, ser. MCS '00. London, UK, UK: Springer-Verlag, 2000, pp. 1–15.

[129] L. Breiman, "Bagging Predictors," *Mach. Learn.*, vol. 24, no. 2, pp. 123–140, Aug. 1996.

[130] Y. Freund and R. E. Schapire, "Experiments with a new boosting algorithm," in *Proceedings of the Thirteenth International Conference on Machine Learning (ICML 1996)*, L. Saitta, Ed. Morgan Kaufmann, 1996. [Online]. Available: http://www.biostat.wisc.edu/ kbroman/teaching/statgen/2004/refs/freund.pdf pp. 148–156.

[131] L. Breiman, "Random Forests," *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, Oct. 2001.

[132] J. R. Quinlan, *C4.5: Programs for Machine Learning*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1993.

[133] C. Cortes and V. Vapnik, "Support-Vector Networks," *Mach. Learn.*, vol. 20, no. 3, pp. 273–297, Sep. 1995. [Online]. Available: http://dx.doi.org/10.1023/A:1022627411411

[134] K. Lee, S. Kung, and N. Verma, "Low-energy formulations of support vector machine kernel functions for biomedical sensor applications," *Signal Processing Systems*, vol. 69, no. 3, pp. 339–349, 2012. [Online]. Available: http://dx.doi.org/10.1007/s11265-012-0672-8

[135] P. Domingos, "A unified bias-variance decomposition and its applications," in *In Proc. 17th International Conf. on Machine Learning*. Morgan Kaufmann, 2000, pp. 231–238.

[136] K. K. Parhi, "VLSI Digital Signal Processing Systems: Design and Implementation." New York: Wiley, 1999, a Wiley-Interscience publication. [Online]. Available: http://opac.inria.fr/record=b1097682

[137] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, vol. 1, 2001, pp. I–511–I–518 vol.1.