Scalable Effort Hardware Design: Exploiting Algorithmic Resilience for Energy Efficiency

Vinay K. Chippa[†], Debabrata Mohapatra[†], Anand Raghunathan[†], Kaushik Roy[†] and Srimat T. Chakradhar[*]
[†] School of Electrical and Computer Engineering, Purdue University
[*] Systems Architecture Department, NEC Laboratories America
{vchipp,dmohapat,raghunathan,kaushik}@purdue.edu, chak@nec-labs.com

ABSTRACT

Algorithms from several interesting application domains exhibit the property of inherent resilience to "errors" from extrinsic or intrinsic sources, offering entirely new avenues for performance and power optimization by relaxing the conventional requirement of exact (numerical or Boolean) equivalence between the specification and hardware implementation.

We propose *scalable effort hardware design* as an approach to tap the reservoir of algorithmic resilience and translate it into highly efficient hardware implementations The basic tenet of the scalable effort design approach is to identify mechanisms at each level of design abstraction (circuit, architecture and algorithm) that can be used to vary the computational effort expended towards generation of the correct (exact) result, and expose them as control knobs in the implementation. These scaling mechanisms can be utilized to achieve improved energy efficiency while maintaining an acceptable (and often, near identical) level of quality of the overall result. A second major tenet of the scalable effort design approach is that fully exploiting the potential of algorithmic resilience requires synergistic cross-layer optimization of scaling mechanisms identified at different levels of design abstraction.

We have implemented an energy-efficient SVM classification chip based on the proposed scalable effort design approach. We present results from post-layout simulations and demonstrate that scalable effort hardware can achieve large energy reductions (1.2X-2.2X with no impact on classification accuracy, and 2.2X-4.1X with modest reductions in accuracy) across various sets. Our results also establish that cross-layer optimization leads to much improved energy vs. quality tradeoffs compared to each of the individual techniques.

Categories and Subject Descriptors

B.7.1 [**INTEGRATED CIRCUITS**]: VLSI (Very large scale integration)

General Terms

Algorithms, Design

Keywords

Scalable Effort, Approximate Computing, Low Power Design, Support Vector Machines, Recognition, Mining

1. INTRODUCTION

Several application domains that are of growing interest in general-purpose and embedded computing, such as Recognition and Data Mining [1], exhibit the interesting property of high inherent algorithmic resilience to "errors" from both extrinsic and intrinsic sources. This resilience can be attributed to several factors [2]. These algorithms are designed to process large amounts of input data that has significant redundancy and may frequently contain significant imperfections or noise. The algorithms themselves are often statistical and aggregative, implying that errors can easily get averaged down or averaged out. The algorithms typically use iterative, successive refinement techniques, which imparts them with a self-healing nature since subsequent iterations may correct errors introduced in previous iterations. Frequently, these algorithms do not have a single golden result; instead, they may produce any one of multiple solutions that are equally acceptable (*e.g.*, a training algorithm that is fed the same data set in a different order may produce a different but equivalent model). Finally, the usage model of these algorithms is such that the user is conditioned to accept less-than-perfect results (even the best known algorithm does not achieve 100% accuracy).

Hardware implementations of inherently resilient algorithms offer entirely new avenues for performance and power optimization by relaxing the conventional requirement of exact (numerical or Boolean) equivalence between the specification and implementation. While several approaches to leveraging algorithmic resilience in both software and hardware implementations have been recently proposed and shown significant promise [3, 4, 5, 6, 7], we believe that fully exploiting the "reservoir of resilience" requires a systematic design approach.

In this work we propose one such systematic approach wherein the notion of *scalable effort* is embodied into the design process at different layers (or levels of abstraction). We demonstrate this approach through the design of an energy-efficient scalable-effort hardware implementation for Support Vector Machines, a popular Machine Learning algorithm. Scalable effort hardware design involves (i) identifying mechanisms at each level of abstraction that can be used to vary the effort expended by the hardware in order to accurately implement the computations in the algorithm, and (ii) exposing them as control knobs that can be used (after fabrication) to achieve energy efficiency while maintaining acceptable overall quality of the result. At the circuit-level, we utilize voltage over-scaling as a mechanism to control the effort expended in order to correctly compute the outputs of computational blocks within the clock period, thereby trading accuracy of the result for the energy consumed to compute it. At the architecture level, we utilize dynamic precision control as the mechanism to vary the computational effort expended. Finally, at the algorithm level, we utilize significance-driven algorithmic truncation in order to achieve an energy vs. accuracy tradeoff. While each of these knobs has significant and interesting effects, we demonstrate that much greater gains can be achieved by synergistically co-optimizing across the different levels.

We have implemented an energy-efficient SVM classification chip based on the scalable effort design approach. We present a wide range of experimental results from post-layout simulations, demonstrating that scalable effort design achieves significant reductions in energy compared to conventional implementations (1.2X-2X with no loss in classification accuracy, and 2.2X-4.1X with a moderate loss in classification accuracy). We also establish the benefits of a layered approach to scalable effort design and and cross-layer optimization of the control knobs that govern the efficiency *vs.* accuracy tradeoff. We believe that the proposed approach has potential to significantly extend the performance and energy-efficiency of hardware implementations of algorithms in various existing and emerging application domains.

2. MOTIVATION

The motivation for our work stems from the observation that algorithms in application domains such as recognition and mining possess a very high degree of resilience to "errors" from both extrinsic and intrinsic sources. The above observation naturally leads to the question of *whether the hardware implementations of these algorithms need to be "perfect"*? Specifically, do the hardware implementations of these algorithms need to maintain exact (numerical or Boolean) equivalence to their specification? We suggest that the answer to these questions is *no*, and address the problem of how to utilize the inherent algorithmic resilience so as to achieve maximum savings in energy consumption.

In order to empirically illustrate the algorithmic resilience described above, we consider Support Vector Machines [8], one of the most popularly used Machine Learning algorithms. We focus on SVM-based classification of handwritten digit images from the MNIST database [9]. The SVM classification almost entirely consists of dot-product computations, which can be expressed as MAC operations. In order to emulate the effect of an "imperfect" hardware implementation, we injected random errors in the outputs of each of the MAC operations performed in the SVM classification algorithm [1] and evaluated the impact on the classification accuracy. The results are presented in Figure 1. The x-axis represents the rate of error injection, and the different curves correspond to different experiments where the errors were restricted to different ranges of least significant bits (within the accumulated result of each MAC operation). It can be seen that, when errors are injected in the 15 LSBs (even with a 100% probability), they do not have any impact on the final classification accuracy. Similarly injecting errors with probabilities of 10^{-6} in upto 28 of the 32 bits has negligible impact on classification accuracy. These results suggest that hardware design techniques that trade off the accuracy of the computations for energy savings are worth investigating.

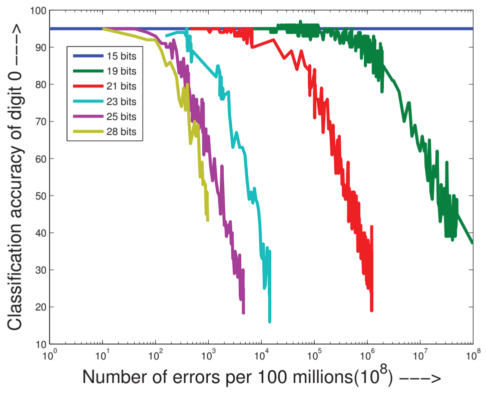

Figure 1: The accuracy of SVM classification for handwritten digit recognition in the presence of errors in the computations

[1]As explained in Section 5.1, the SVM classification almost entirely consists of dot-product computations, which can be expressed as MAC operations.

3. RELATED WORK

A significant body of previous work shares the philosophy of exploiting the inherent resilience of algorithms in certain application domains to achieve various benefits. In this section, we acknowledge and describe the closely related efforts that have inspired our own, and place our contributions in their context.

The use of imperfect or unreliable components in computing has a long history and can be traced back to the work of Von Neumann [10], which proposed a model for computation using unreliable components. In his work on adaptive computers [11], Breuer proposed graceful degradation as a mechanism to deal with component failures. Over two decades later, the use of imprecise computation was explored to achieve improved performance in the context of real-time systems [12].

ANT (Algorithmic Noise Tolerance) [5] was one of the earliest proposals to leverage the inherent error resilience of algorithms in the context of hardware implementation, and aimed to achieve both energy efficiency and tolerance to deep submicron noise. Subsequent efforts proposed variants of this basic approach, and applied it to signal, image, and video processing algorithms [13, 14, 15].

The concept of Probabilistic CMOS or PCMOS, wherein each transistor and logic gate displays a probabilistic rather than deterministic behavior, was proposed as an energy-efficient alternative to traditional always-correct computational models [16]. This has led to a significant body of research on probabilistic and approximate computation, which is summarized in [4].

The concept of utilizing the inherent resilience of multimedia applications to tolerate defects and improve fabrication yields was proposed in [17], and led to new approaches to manufacturing test [18].

Other recent efforts have exploited algorithmic error resilience for process variation tolerance and low power design through the use of design techniques such as significance driven computation [6, 19].

ERSA (Error Resilient System Architecture) [7] is a programmable multi-core architecture for deeply scaled technologies with unreliable components, that combines one reliable processor core with a large number of unreliable cores. ERSA exploits the inherent error resilience of probabilistic applications, which overlap with the domains considered in our work.

We have used Support Vector Machines [8], one of the most widely used machine learning algorithms, as a vehicle to demonstrate our concepts. Various hardware implementations of Support Vector Machines [20, 21] have been optimized for either performance or area, but the error resiliency of the algorithm has not been fully explored.

The term scalable effort was inspired by the work on best effort computing [2], where error resiliency has been used to achieve parallel scalability of software implementations on multi-core computing platforms [3]. Furthermore, the basic SVM architecture used in our work was inspired by the MAPLE parallel architecture [22], where an array of variable precision processing elements were utilized to achieve high performance with relatively low power consumption.

The primary contribution of this work is a systematic design approach for hardware implementation of algorithms that exhibit high levels of inherent resilience. Unlike previous efforts that only focus on one dimension (*e.g.*, voltage overscaling), the scalable effort approach espouses synergistic scaling along multiple dimensions that correspond to different levels of design abstraction (circuit, micro-architecture, and algorithm). This leads to higher energy savings through better utilization of the reservoir of algorithmic resilience, as borne out by the experimental results presented in this paper.

4. BACKGROUND

Support Vector Machines (SVMs), a family of algorithms for supervised learning, are widely used for applications that require classification (where data must be classified into a number of pre-determined categories) and regression (where a model is built to capture the relationship between a dependent variable and one or more independent variables) [2]. In classification, given a labeled training data set (each data point is a multi-dimensional vector of features), the SVM training algorithm is used to generate a hyperplane in the feature space that distinguishes the classes with maximal separation (see Figure 2). The hyperplane is fully specified by a subset of the training data that are called the support vectors.

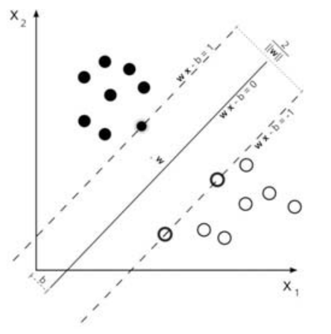

Figure 2: Illustration of SVM-based classification (Source: Wikipedia)

Once the hyperplane is generated, any unlabeled datum can be assigned a label based on it's position relative to the hyperplane. Generation of the hyperplane from the labeled data is called training and assigning a label to a new data point is called testing [3].

The computation performed for classifying a new unlabeled data point is given by

$$\text{Label} = \text{sign}(\sum_{i=1}^{N} Kernel(\vec{x}.s\vec{v}_i) * \alpha_i * y_i - b) \qquad (1)$$

where $\vec{x}$ is a vector representing the data to be classified, $s\vec{v}_i$ is the i-th support vector, b is the offset and α_i and y_i are the Lagrangian coefficient (computed during training) and the label of the i-th support vector, respectively. *Kernel* is a non-linear function that greatly enhances the capability of SVMs by enabling the construction of non-linear classifiers. $s\vec{v}_i$, α_i and b values are generated by the SVM training algorithm, details of which can be found in [8]. SVMs are used in a wide range of applications involving image and video analysis (*e.g.*, face or object detection, handwriting recognition, content-based image retrieval, speech recognition), and have demonstrated great promise in several other application domains [23].

5. SCALABLE EFFORT HARDWARE FOR SVM CLASSIFICATION

In this section, we describe the scalable effort design approach and illustrate it through application to SVM classification. In a broad sense, scalable effort hardware provides mechanisms to modulate the effort expended in performing the computations that constitute the algorithm. The specific definition of effort varies depending on the level of design abstraction at which it is modulated. We next briefly describe the overall design of the SVM classification chip and detail how the principles of scalable effort design were applied at the circuit, architecture, and algorithm levels.

5.1 Overview

Figure 3 presents a dedicated hardware architecture for SVM classification. We used a systolic array architecture (similar to [22]) to implement the computation of dot products between support vectors and test vectors, which dominates the workload of SVM classification. From Equation 1,

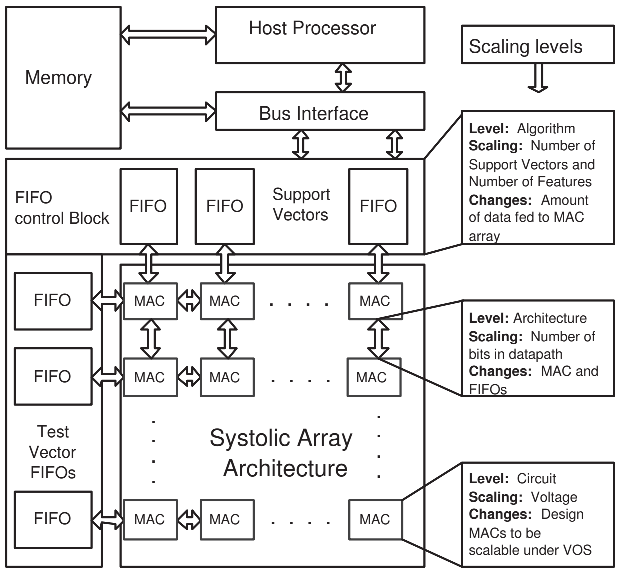

Figure 3: Block diagram of SVM architecture

we can see that with N support vectors and a feature size of d, classifying a single data point will require $N \cdot d + 2N$ multiplications, $N \cdot d - N + 1$ additions and N kernel function computations. Of these, $N \cdot d$ multiplications and $N \cdot d - N$ additions are due to the dot product of the test vector with the support vectors.

The architecture consists of two arrays of FIFOs from which data is streamed to a 2-dimensional array of MAC units. The support vectors and test vectors are pushed into their respective FIFO's. Support vectors are streamed from top to bottom, and test vectors are streamed from left to right, through the MAC array. Each MAC unit computes the dot product between a unique (support vector, test vector) pair by processing one dimension per cycle in the nominal case and accumulating the result. Once the dot product operation is complete, the dot product values are read out in a raster scan order by the host processor, which takes this data and performs the remaining computation required to compute the label.

We would like to mention that our focus in this work is not on the base architecture itself; rather, we focus on the application of scalable effort at different levels of abstraction to this architecture. The effort expended by the SVM classification hardware can be scaled at the circuit, architecture, and algorithm levels. The proposed design has provisions to scale the effort at all these levels, each of which is described in detail in the following subsections.

5.2 Algorithm Level Scaling

At the algorithm level, we identify parameters that control the amount of computation performed and vary them to tradeoff decreased computational complexity for the quality of the result. Rather than indiscriminately eliminate computations, we prioritize computations based on their likely impact on the quality of the result to achieve a better tradeoff.

In the case of Support Vector Machines, our analysis in Section 5.1 shows that the complexity depends upon both N and d, *i.e.*, the number of support vectors and the number of features per vector. Figure 4 depicts the dot product operations involved in SVM classification in matrix form and illustrates how the scaling is performed along these two dimensions. In order to achieve the best accuracy vs. energy tradeoff, support vectors are considered in the order of their significance. One way of quantifying the significance of sup-

[2]We focus on classification, however the proposed concepts apply to SVM based regression as well.

[3]Although we only demonstrate our ideas for SVM testing, the core computation in training is essentially the same and the proposed concepts apply to training as well.

port vectors is based on the values of α_i (Lagrangian coefficients that are computed during training). This approach can be further extended using the concept of RSVMs [24], where training is adapted to generate support vector sets of different sizes, from which we choose during testing.

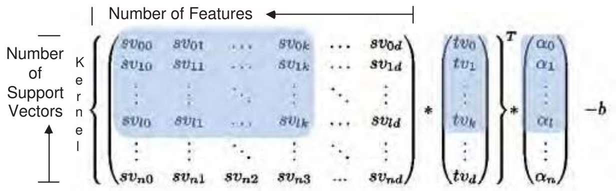

Figure 4: SVM classification in matrix form and illustration of algorithmic scaling

When we scale the number of support vectors, we are basically reducing the number of dot product computations per classification. We can also cut down the number of MAC operations per dot product, e.g., scaling the dimensionality of the vectors. Generally, different features (or dimensions) have very different effects on the accuracy of classification. If the features are arranged in the order of their importance, the dot product can be computed in that order and the computation can be stopped at any point to get an approximate output.

5.3 Architecture Level Scaling

At the architecture level, we scale effort by scaling the precision with which the variables of the algorithm are represented and the corresponding operations are performed. The variables can be the inputs of the algorithm, like the test vectors and α, or they can be the intermediate values like outputs of the multiplier and the accumulator. The number of bits needed to represent variables varies from data set to data set. In context of SVM's, the effect of precision scaling has been studied in [20, 22]. In this section, we explain different methods of realizing precision scaling. However our focus is on translating precision scaling into energy efficiency, and combining this with other scalable effort mechanisms to maximize the overall energy efficiency of the SVM computation.

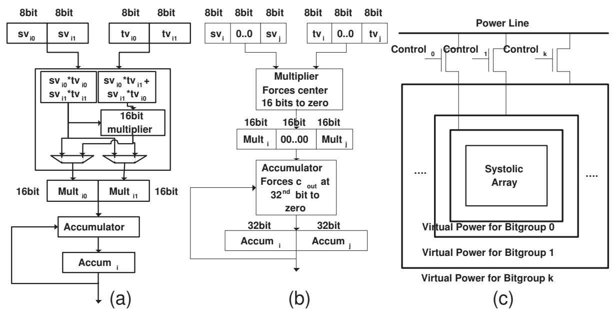
8bit 8bit 8bit 8bit
sv i0 sv i1 tv i0 tv i1
sv i0*tv i0 sv i1*tv i1
sv i0*tv i1 + sv i1*tv i0
16bit multiplier
16bit Mult i0 Mult i1 16bit
Accumulator
Accum i
(a)
8bit 8bit 8bit 8bit 8bit 8bit
sv i 0..0 sv j tv i 0..0 tv j
Multiplier Forces center 16 bits to zero
16bit 16bit 16bit
Mult i 00..00 Mult j
Accumulator Forces c out at 32nd bit to zero
32bit 32bit
Accum i Accum j
(b)
Power Line
Control 0 Control 1 Control k
Systolic Array
Virtual Power for Bitgroup 0
Virtual Power for Bitgroup 1
Virtual Power for Bitgroup k
(c)

Figure 5: (a) Data packing without zero padding. (b) Data packing with zero padding. (c) Power gating

The simplest approach to precision scaling at the architecture level is to utilize only a sub-set of the data path and shut down the rest for energy efficiency. While this approach saves energy, the hardware that is shut down is unutilized. Whenever possible, it is preferable to pack multiple data points into a single word and use the same hardware to process multiple data at the same time (similar to SIMD instructions in modern processors), simultaneously increasing the overall throughput and energy efficiency of the system. Multi-precision operation can be achieved in two ways - with segmented units that incur some hardware overhead, or through suitable data packing with padding (at virtually no hardware overhead).

Figure 5(a) shows the block diagram of a segmented MAC unit that supports multi-precision operation. In this case, the multiplier needs to be able to operate differently for different precision levels. For an architecture supporting 24 bit input data, we can pack one 16 bit, three 8 bit or six 4 bit data types into a single input. We improve the throughput in this case at the cost of some hardware overhead.

Figure 5(b) shows the block diagram of a MAC unit implementing padding, which requires only minimal amount of control logic on top of the regular MAC. Multiple data are packed into a single word with sufficient padding of '0's, such that the middle bits of the multiplier output are 0 and the accumulator never sees carry propagation across the boundaries separating the distinct results in the output. In other words, the guard bits ensure that the two MAC operations do not interfere with each other. For a data path that supports 24-bit input words, we can pack one 16 bit, two 8 bit or two 4 bit data. In this case, we are under-utilizing the hardware (due to the padding bits) but require almost no hardware overhead.

In both the aforementioned methods, there are cases where we will not utilize the hardware completely and it would be advantageous to shut down the parts of the circuit that are not being used. Therefore, we have implemented power gating, where we connect groups of bit lines of the data path to different voltage domains, each of which can be turned off independently using sleep transistors. Figure 5(c) illustrates this concept.

5.4 Circuit Level Scaling

At the circuit level, scalable effort is synonymous with voltage over-scaling. Voltage over-scaling differs from traditional voltage scaling in that we do not scale the clock frequency, thereby intentionally causing the critical paths to violate the clock period. We do not have to take strict corrective measures to preserve the accuracy of computation as the applications that we consider are inherently error resilient. Such implementations can be categorized under the approximate arithmetic paradigm which is an active area of research [4, 6, 13]. In order to perform approximate computing at the circuit level, we visualize computations in terms of "meta" operations, which in the case of SVM is the dot product. Scalable effort at the circuit level essentially translates into a dot product value whose accuracy scales with the degree of voltage over-scaling. While any given implementation can be subject to voltage over-scaling, appropriate design techniques need to be used in order to obtain a better energy vs. accuracy tradeoff.

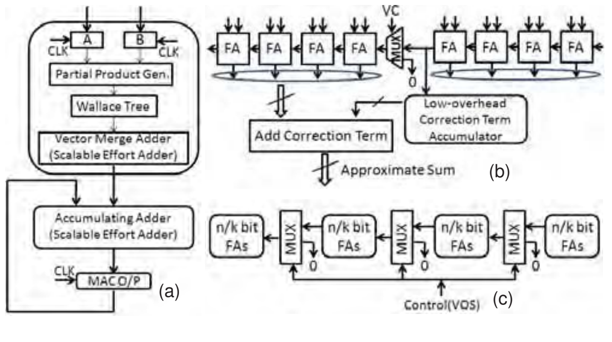
A B
CLK CLK
Partial Product Gen.
Wallace Tree
Vector Merge Adder (Scalable Effort Adder)
Accumulating Adder (Scalable Effort Adder)
CLK
MAC O/P
(a)
VC
FA FA FA FA MUX FA FA FA FA
0
Add Correction Term
Low-overhead Correction Term Accumulator
Approximate Sum
(b)
n/k bit FAs MUX n/k bit FAs MUX n/k bit FAs MUX n/k bit FAs
0 0 0
Control(VOS)
(c)

Figure 6: (a) Scalable MAC circuit. (b) Correction term based implementation. (c) Segmented Implementation

Figure 6 shows the design of a voltage scalable MAC for dot product computation. At an atomic level the MAC unit consists of partial product generator followed by a merge adder and an accumulator. In order to implement a scalable effort MAC, we apply a segmentation based technique to the adders in the MAC hardware. The technique involves segmenting the adder into smaller bit width adders. The point of segmentation can be adaptively controlled based on the degree of voltage scaling. This serves two primary purposes: a) reduces critical path of the adder, allowing aggressive voltage scaling at the cost of approximation and b) prevents harmful glitch propagation across the adder from arbitrarily corrupting the MSB bits. However, application of the standalone segmentation technique incurs significant error in the dot product due to its accumulative nature over multiple cycles. We address this issue by introducing a low overhead correction circuit that keeps track of the carries across sections of the segmented adder that have been ignored in each cycle by means of small bit width counters. In the event of overflow in any one of the carry counters, a correction term is added to adjust the accumulator value based on these counter outputs. It is to be noted that even though this adds few extra cycles to the dot product computation, it can be ignored for practical purposes since the feature size of data (number of MAC cycles for one dot product computation) is typically large. The scalable effort MAC incorporating segmentation with error correction scheme incurs roughly 25% area overhead. However, this area overhead, purely due to the MAC, is significantly amortized over the entire SVM design and can be considered negligible. Another important design issue is determining the group size k during segmentation. Higher degree of segmentation (higher k) implies greater opportunity for VOS due to critical path reduction, at the expense of greater area and power overhead due to the carry correction counters. Hence, a judicious approach needs to be taken while determining the group size during segmentation. In our design k=$n/4$ was used to achieve a balanced tradeoff between the degree of VOS and the overhead incurred.

Techniques such as segmentation augmented by correction term addition are independent of the underlying adder and multiplier architectures and hence can be applied to other algorithms involving these basic arithmetic operations. Moreover, the proposed segmentation technique is distinct from pure truncation in the sense that we utilize all the input bits in computing the result as opposed to discarding some of them, thereby achieving higher accuracy.

6. EXPERIMENTAL METHODOLOGY

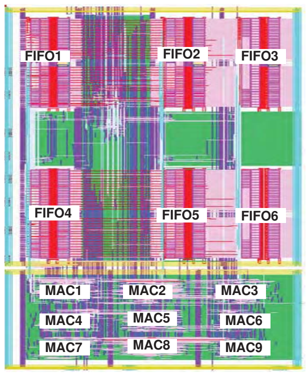

Figure 7: Scalable-effort SVM classifier

In order to demonstrate the proposed design approach, we realised a scalable effort hardware implementation of SVM classification, and evaluated it for energy consumption and accuracy on various data sets. We have implemented the full layout in two different process technologies - IBM 90nm and 45nm. The 45nm technology implementation has been taped out but the chips are not available as of this time, hence we present results based on post-layout simulations.

The layout of the hardware is presented in figure 7. The design flow used for our implementation consists of Synopsys Design Compiler [25] for logic synthesis and Cadence SOC Encounter [26] for physical design. The parasitics annotated net list extracted from the layout was used for power simulations using the Synopsys Nanosim [27] switch-level simulator.

Since circuit level simulations for large data sets and numerous combinations of circuit, micro-architecture, and algorithmic scaling "levels" requires unacceptable simulation times, we used software simulations (on large traces) to obtain classification accuracy and hardware simulations (for smaller representative traces) to obtain energy values.

We used the MILDE framework from NEC Labs, Princeton to perform software simulations [28]. The data sets used for validation are explained in detail in Table 1.

Table 1: Data sets used for our evaluations

Data set	Source	Features	Classes
`Adult`	UCI database	14	2
`MNIST`	NEC MiLDe	784	10
`NORB`	NEC MiLDe	5184	5
`Checkerboard`	Artificial	2	2

Reflecting the effects of the different scaling mechanisms in the software simulations required us to perform careful modeling as explained below. To obtain the accuracy values for different voltage levels, we characterized the MAC for different voltage levels to construct functional models, and used these functional models in software. These models break down the MAC operation into the bit-level and reflect the impact of segmentation and limited carry propagation on the computed result. To obtain accuracy values of the algorithm for different precision levels, the LSB bits of the variables in the software were truncated before/after the appropriate computations. For algorithm level scaling, The SVM implementation in MiLDe was modified to consider only Support Vectors with α values greater than a pre-specified threshold during classification.

7. RESULTS

In this section, we present results that demonstrate the potential of scalable effort hardware design. Scalable effort hardware can be characterized by the energy *vs.* accuracy trade off curve that it can achieve. We first report the overall energy savings that were obtained through scalable-effort hardware at different levels of accuracy. Next, we compare the cross-layer optimization of scaling mechanisms *vs.* the application of individual techniques.

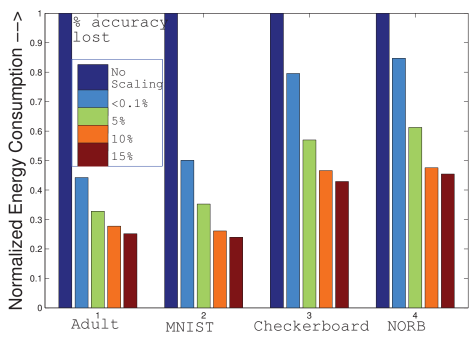

Figure 8: Normalized energy consumption vs accuracy loss for different data sets

Figure 8 shows the normalized energy consumption at different accuracy levels for several data sets. Energy is normalized to the base case in which no scaling techniques are applied [4]. We are able to obtain 1.2X-2.2X energy savings without any significant loss of accuracy. If a minimal loss of 5% in classification accuracy is acceptable, energy savings of 1.6X-3X are possible. For moderate accuracy loss of 15%, the energy savings grow to 2.2X-4.1X.

Figure 9 illustrates the benefits of cross layer optimization in scalable effort design. Significantly improved energy *vs.* quality trade offs were obtained by utilizing the optimal combinations of all the scaling mechanisms as compared

[4]Note that the base case was well-optimized - clocked at close to the critical path, and the data path was configured to operate at the minimum precision dictated by the input data set.

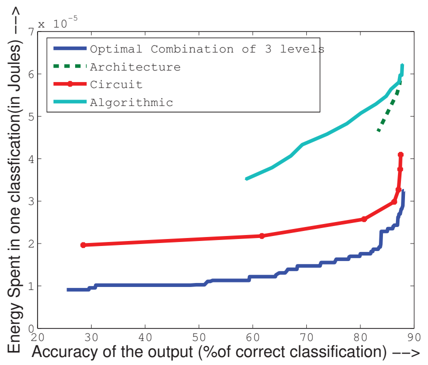

Figure 9: Energy vs accuracy for different levels of scaling(`MNIST` data set)

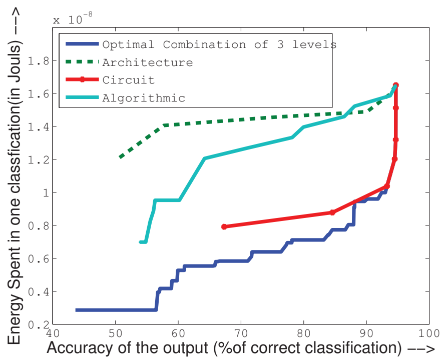

Figure 10: Energy vs accuracy for different levels of scaling (`Checkerboard` data set)

to the application of individual scaling techniques. Figure 9 shows the tradeoffs for the `MNIST` data set. To evaluate a qualitatively different data set, we also obtained accuracy *vs.* energy trade offs for the artificially generated `Checkerboard` data set and present the results in Figure 10. These results reinforce our initial claim that cross layer optimization has considerably greater potential to tap into the reservoir of algorithmic resilience and can achieve significantly improved energy-accuracy trade offs over uni-dimensional approaches.

8. SUMMARY AND CONCLUSIONS

We presented a new systematic approach based on the concept of scalable effort hardware, for the design of efficient hardware implementations for algorithms that demonstrate inherent error resilience. Scalable effort design is based on the identification of mechanisms at each level of design abstraction (circuit, architecture and algorithm) that can be used to vary the computational effort expended towards generation of the correct (exact) result. These mechanisms can be utilized to achieve improved energy efficiency (or performance) while maintaining an acceptable level of quality of the overall result. While each of these knobs has significant impact, we demonstrated that much greater gains can be achieved through synergistic cross-layer optimization.

Although we have focused on a representative and popular Machine Learning algorithm, namely Support Vector Machines, we believe that the proposed concepts are general and can be applied to realize high-performance or energy-efficient implementations of a wide range of Recognition and Data Mining algorithms. Our focus on these application domains is fueled by our belief that the constituent algorithms have unsurpassed levels of algorithmic error resilience. However, the property of error resilience (and therefore our design approach) also applies to traditional application domains such as DSP and multi-media (image/audio/video) processing, which have largely been the focus of related work.

Acknowledgment: The authors would like to thank Patrick Ndai for his inputs on circuit-level scaling, and Igor Durdanovic for his help with the MiLDe software and data sets.

9. REFERENCES

[1] P. Dubey. A platform 2015 workload model recognition, mining and synthesis moves computers to the era of tera. White paper, Intel Corp., 2005.

[2] S. T. Chakradhar and A. Raghunathan. Best-effort Computing: Re-thinking Parallel Software and Hardware. In *Proc. ACM/IEEE Design Automation Conf.*, June 2010.

[3] Jiayuan Meng, Srimat Chakradhar, and Anand Raghunathan. Best-effort parallel execution framework for recognition and mining applications. *Parallel and Distributed Processing Symposium, International*, pages 1–12, 2009.

[4] Krishna V. Palem et al. Sustaining moore's law in embedded computing through probabilistic and approximate design: retrospects and prospects. In *Proc. Int. Conf. on Compilers, Architecture and Synthesis for Embedded Systems*, pages 1–10, 2009.

[5] Rajamohana Hegde and Naresh R. Shanbhag. Energy-efficient signal processing via algorithmic noise-tolerance. In *Proc. Int. Symp. on Low Power Electronics and Design*, pages 30–35, 1999.

[6] Debabrata Mohapatra, Georgios Karakonstantis, and Kaushik Roy. Significance driven computation: A voltage-scalable, variation-aware, quality-tuning motion estimator. In *Proc. Int. Symp. Low Power Electronics and Design*, Aug. 2009.

[7] J. Bau, R. Hankins, Q. Jacobson, S. Mitra, B. Saha, and Adl A. Tabatabai. Error resilient system architecture (ERSA) for probabilistic applications. In 3rd Wkshp. on System Effects of Logic Soft Errors (SELSE), April 2007.

[8] Vladimir N. Vapnik. *The nature of statistical learning theory.* Springer-Verlag New York, Inc., New York, NY, USA, 1995.

[9] Yann Lecun and Corinna Cortes. The mnist database of handwritten digits.

[10] J. Von Neumann. Probabilistic logics and the synthesis of reliable organisms from unreliable components, 1956.

[11] M. A. Breuer. Adaptive computers. *Information and Control*, 11(4):402–422, October 1967.

[12] K. J. Lin, S. Natarajan, and J. W. S. Liu. Imprecise results: Utilizing partial computations in real-time systems. In *Proc. IEEE Real-Time Systems Symposium*, 1987.

[13] Naresh Shanbhag. Reliable and energy-efficient digital signal processing. In *Proc. Design Automation Conference*, pages 830–835, 2002.

[14] R. Hegde and N.R. Shanbhag. A low-power digital filter IC via soft DSP. In *Proc. IEEE Conf. Custom Integrated Circuits*, pages 309–312, 2001.

[15] Girish Vishnu Varatkar and Naresh R. Shanbhag. Error-resilient motion estimation architecture. *IEEE Trans. VLSI Systems*, 16(10):1399–1412, 2008.

[16] Krishna V. Palem. Energy aware algorithm design via probabilistic computing: From algorithms and models to Moore's law and novel (semiconductor) devices. In *Proc. Int. Conf. on Compilers, Architecture and Synthesis for Embedded Systems*, pages 113–116, 2003.

[17] M.A. Breuer. Multi-media applications and imprecise computation. In *Proc. 8th Euromicro Conf. on Digital System Design*, pages 2–7, Aug.-3 Sept. 2005.

[18] Tong-Yu Hsieh, Kuen-Jong Lee, and M.A. Breuer. An error rate based test methodology to support error-tolerance. *Reliability, IEEE Transactions on*, 57(1):204–214, March 2008.

[19] Nilanjan Banerjee, Georgios Karakonstantis, and Kaushik Roy. Process variation tolerant low power DCT architecture. In *Proc. Design, Automation, and Test Europe*, April 2007.

[20] D. Anguita, A. Ghio, S. Pischiutta, and S. Ridella. A hardware-friendly support vector machine for embedded automotive applications. In *Proc. International Joint Conference on Neural Networks*, pages 1360–1364, Aug. 2007.

[21] Soumyajit Dey, Monu Kedia, Niket Agarwal, and Anupam Basu. Embedded support vector machine : Architectural enhancements and evaluation. *Int. Conf. VLSI Design*, pages 685–690, 2007.

[22] Srihari Cadambi, Igor Durdanovic, Venkata Jakkula, Murugan Sankaradass, Eric Cosatto, Srimat Chakradhar, and Hans Peter Graf. A massively parallel FPGA-based coprocessor for support vector machines. *IEEE Symp. Field-Programmable Custom Computing Machines*, pages 115–122, 2009.

[23] http://www.clopinet.com/isabelle/projects/svm/applist.html.

[24] Yuh jye Lee and Olvi L. Mangasarian. RSVM: Reduced support vector machines. In *Data Mining Institute, Computer Sciences Department, University of Wisconsin*, pages 00–07, 2001.

[25] Design compiler. *Synopsys Inc.*

[26] SOC Encounter. *Cadence Inc.*

[27] Nanosim. *Synopsys Inc,.*

[28] Milde. *www.nec-labs.com.*