Design of Voltage-Scalable Meta-Functions for Approximate Computing

Debabrata Mohapatra, Vinay K. Chippa, Anand Raghunathan and Kaushik Roy
School of Electrical and Computer Engineering, Purdue University
{dmohapat,vchipp,raghunathan,kaushik}@purdue.edu

***Abstract*—Approximate computing techniques that exploit the inherent resilience in algorithms through mechanisms such as voltage over-scaling (VOS) have gained significant interest. In this work, we focus on *meta-functions* that represent computational kernels commonly found in application domains that demonstrate significant inherent resilience, namely Multimedia, Recognition and Data Mining. We propose design techniques (dynamic segmentation with multi-cycle error compensation, and delay budgeting for chained data path components) which enable the hardware implementations of these meta-functions to scale more gracefully under voltage over-scaling. The net effect of these design techniques is improved accuracy (fewer and smaller errors) under a wide range of over-scaled voltages. Results based on extensive transistor-level simulations demonstrate that the optimized meta-function implementations consume upto 30% less energy at *iso-error* rates, while achieving upto 27% lower error rates at *iso-energy* when compared to their baseline counterparts. System-level simulations for three applications, motion estimation, support vector machine based classification and k-means based clustering are also presented to demonstrate the impact of the improved meta-functions at the application level.**

***Index Terms*—Approximate Computing, Low Power Design, Voltage Over-scaling, Meta-functions.**

I. INTRODUCTION

A promising class of low-power design techniques, which could be collectively classified under the paradigm of "approximate computing", has received significant attention recently. These techniques relax the conventional requirement of perfect equivalence between the specification and hardware implementation, and translate this flexibility into energy efficiency. A number of traditional and emerging application domains, such as multimedia processing, wireless communications, networking, and recognition, mining, and synthesis, demonstrate the property of inherent resilience to "errors" introduced in their implementation. This resilience is due to several factors: (i) these algorithms are designed to deal with real-world noisy input data, (ii) they frequently process large data sets with significant redundancy, (iii) they utilize statistical or probabilistic computations, and (iv) a small amount of error in their outputs cannot be discerned due to limited human perception.

Several examples exist of techniques that embody the approximate computing approach [1], [2], [3], [4], which we describe further in the next section. A related class of techniques [5], [6] rely on detection followed by correction or prediction followed by multicycle operation respectively to avoid timing errors, allowing circuits to operate at "zero margins" even in the presence of process, voltage, and temperature variations. Most of these techniques utilize voltage over-scaling (VOS) in order to obtain energy efficiency - the voltage is scaled to a point where the delays of some paths in the logic will exceed the clock period, thereby introducing "errors". *A common underlying assumption for all these techniques is that the circuit blocks used in the hardware implementations scale gracefully under VOS,* i.e.*, the probability of errors or magnitude of errors introduced under VOS is relatively small.* However, this is not always the case, especially for circuits that have been aggressively optimized for performance and thereby contain a large number of paths whose delays are close to the critical path [7]. If, contrary to the assumption, the building blocks do not scale gracefully, then the algorithm will fail in providing an acceptable output, limiting the extent to which VOS can be applied.

In this paper, we consider common computational kernels used in multimedia, recognition, and mining algorithms, and propose design techniques to make the hardware implementations of these meta-functions behave more gracefully, *i.e.*, generate fewer or smaller errors, under VOS. We concentrate on meta-functions since they dominate the energy and execution time of algorithms, and also provide scope for new design optimizations that improve scalability under VOS. The major contributions of this work are as follows:

- We identify the computational kernels in representative multimedia, recognition, and mining algorithms, and analyze their behavior under VOS. We consider coarse-grained meta-functions that are iteratively executed across multiple cycles, which facilitates some of the proposed design techniques.
- We optimize the meta-function implementations using two basic techniques: (i) Dynamic segmentation with multi-cycle error compensation, and (ii) Delay budgeting of chained units, for improved scalability under VOS.
- Using a combination of transistor-level simulation and system-level simulation, we analyze the benefits of the proposed design techniques, and also demonstrate how optimizations at the meta-function level translate into benefits at the application level in terms of improved energy vs. output quality tradeoffs.

The optimized implementations of the meta-functions can be used as a drop in replacement for their conventional

This material is based upon work supported in part by the National Science Foundation under Grant No. 1018621

counterparts in a wide range of algorithms. Moreover, the design techniques used for optimizing the meta-functions are generic in nature, hence, they can be also applied to the design of other meta-functions that involve similar operations.

The rest of the paper is organized as follows. Section II briefly presents the related work in this area and places our contribution in this context. Section III identifies the computational kernels that are considered as meta-functions in our work. Section IV describes in detail the design techniques used for optimizing the meta-functions. Simulation results highlighting the impact of optimizations at the meta-function level and at the algorithm level are presented in Section V, while Section VI concludes the paper.

II. Related Work

ANT (Algorithmic Noise Tolerance) [1] was one of the earliest proposals to leverage the inherent error resilience of algorithms in the context of hardware implementation of signal, image, and video processing algorithms, improving both energy efficiency and tolerance to deep sub-micron noise. The concept of Probabilistic CMOS or PCMOS, wherein each logic gate displays a probabilistic rather than deterministic behavior, was proposed as an energy efficient alternative to traditional always-correct computational models [4]. ERSA (Error Resilient System Architecture) [2] 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. A Significance Driven Design (SDD) approach for process variation tolerance and low power design that identifies and protects significant computations and data in a preferential manner has been proposed in [8], [3]. While a large body of research has tried to exploit the inherent resilience of algorithms, most of them are based on the implicit assumption that the underlying hardware building blocks scale gracefully under VOS. The primary contribution of this work is a systematic design approach that aims at improving the scalability of the underlying basic building blocks of multimedia, recognition, and mining algorithms, under VOS. Unlike previous efforts that solely rely on the inherent error resiliency of the algorithms, coupled with the implicit assumption regarding scaling behavior of underlying components, this work goes a step further by identifying the computational kernels in the algorithm and proposing techniques for improving their robustness under VOS. The dynamic segmentation and error compensation technique exploits the multi-cycle aggregative nature of the computational kernels that we consider, and delay budgeting exploits the difference in the error characteristics of chained components.

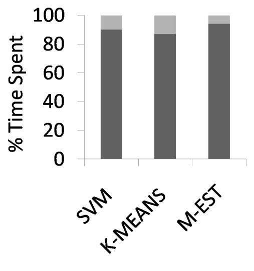


Fig. 1: Computational kernels of MRM algorithms.

III. Meta-functions in Multimedia, Recognition and Mining (MRM) Applications

The primary objective of our work is to facilitate the approximate computing paradigm through the design of hardware "building blocks" that demonstrate more graceful degradation under VOS. We consider few representative algorithms and identifies their underlying computational kernels as the meta-functions. In this work, the three algorithms considered are 1) Motion estimation which is the workhorse of MPEG [9] video compression standard, 2) Support Vector Machine (SVM) classification [10] and 3) K-means clustering [11], which are widely used in the fields of Recognition and Data Mining. For each of these three algorithms, we used software profiling to discover where the algorithm spends most of its computation time. The results of our profiling are shown in Figure 1. It is evident from Figure 1 that in each case there exists a computational kernel where the algorithm spends 85-95% of its computation time. In the case of the motion estimation algorithm, the meta-function is identified to be the L1-norm which is the sum of absolute differences (SAD). For SVM classification, dot product, which is essentially a series of multiply and accumulate operations, is identified to be the dominant meta-function. Depending on the distance criterion used in K-means clustering, the dominant meta-function can either be the L1-norm (SAD) or the L2-norm which is the Euclidean distance between two vectors. Mathematical representations of all three meta-functions for two N dimensional vectors $\mathbf{X}$ and $\mathbf{Y}$ are given in Figure 2. Baseline hardware implementations of these meta-functions are shown in Figure 3.

$$L1-NORM(\mathbf{X},\mathbf{Y}) = \sum_{i=1}^{N} |x_i - y_i|$$

$$DOTPRODUCT(\mathbf{X},\mathbf{Y}) = \sum_{i=1}^{N} x_i * y_i$$

$$L2-NORM(\mathbf{X},\mathbf{Y}) = \sum_{i=1}^{N} (x_i - y_i)^2$$

Fig. 2: Mathematical representation of meta-functions.

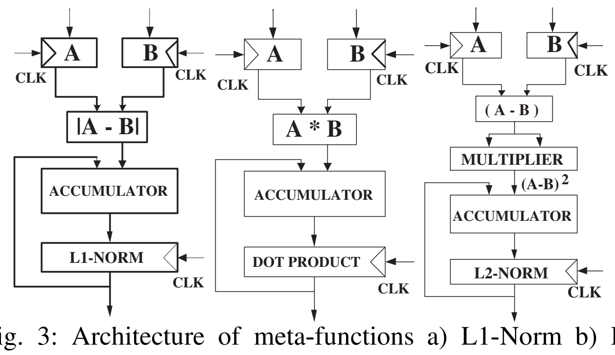


Fig. 3: Architecture of meta-functions a) L1-Norm b) Dot Product and c) L2-Norm.

From the results obtained in Figure 1, it is logical to assume that improving the scalability of the underlying meta-functions would naturally lead to a better energy efficiency *vs.* output quality tradeoff at the algorithm level. We next present design techniques to achieve this objective.

IV. Scalable meta-function design

VOS is a very effective technique for reducing power due to the superlinear dependence of both dynamic and leakage power on supply voltage. VOS 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. However, since the applications themselves are inherently error resilient, the necessity to take strict corrective measures to preserve the accuracy of computation can be waived. Note that while any given implementation can be subject to VOS, appropriate design techniques need to be used in order to obtain a better energy *vs.* output quality tradeoff. In this section, we present two basic design techniques to optimize the meta-functions identified in Section III.

A. Dynamic Segmentation and Error Compensation (DSEC)

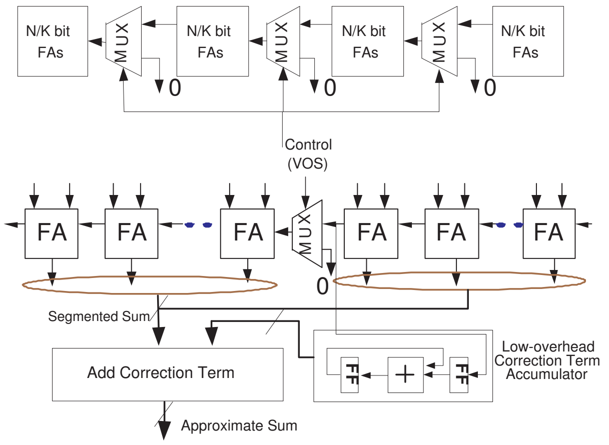
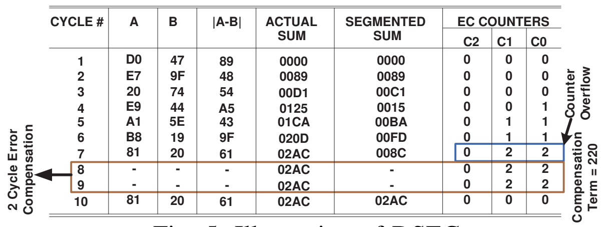


Fig. 4: Dynamic segmentation with error compensation.

Figure 4 shows the design of a voltage scalable accumulator which is an integral part of all three meta-functions under consideration [1]. Dynamic segmentation involves dividing the adder in the accumulator into smaller bit width adders by bit-slicing the data path. Depending on the degree of VOS, the degree of segmentation can be dynamically controlled based on control inputs to the multiplexers. Under nominal V_{DD}, the multiplexers select the original carry and feed it to the subsequent stages. Note that each adder stage can be implemented using any adder architecture. When operating under VOS, the adder stages are isolated from each other due to a carry of zero being forced by the multiplexers. The segmentation based approach serves two primary purposes: a) it reduces the critical path of the adder, allowing aggressive voltage scaling at the cost of approximation, and b) it prevents harmful glitch propagation across the adder stages from arbitrarily corrupting the output bits. However, application of the standalone segmentation technique incurs significant error in the meta-function computation due to its accumulative nature across multiple cycles. In order to address this issue, we augment the dynamic segmentation scheme with a multi-cycle error compensation technique. Error compensation involves addition of a low-overhead correction circuitry 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. When there is an overflow in any one of the carry counters, a "correction cycle" is introduced to adjust the accumulator value based on these counter values.

[1]We focus on the accumulator since it is the component that directly feeds the flip-flops. Therefore, the effect of VOS can be viewed as the accumulator not having sufficient time to compute its output within the clock period.



CYCLE #	A	B	|A-B|	ACTUAL SUM	SEGMENTED SUM	EC COUNTERS		
						C2	C1	C0
1	D0	47	89	0000	0000	0	0	0
2	E7	9F	48	0089	0089	0	0	0
3	20	74	54	00D1	00C1	0	0	0
4	E9	44	A5	0125	0015	0	0	1
5	A1	5E	43	01CA	00BA	0	1	1
6	B8	19	9F	020D	00FD	0	1	1
7	81	20	61	02AC	008C	0	2	2
8	-	-	-	02AC	-	0	2	2
9	-	-	-	02AC	-	0	2	2
10	81	20	61	02AC	02AC	0	0	0

2 Cycle Error Compensation
Counter Overflow
Compensation Term = 220

Fig. 5: Illustration of DSEC.

The dynamic segmentation with error compensation technique is illustrated with an example in Figure 5. Let us consider a 16-bit adder that is segmented into 4 parts, each of 4 bits. The carries across the segmented stages are tracked by three 1-bit counters C_0, C_1, and C_2, which track carries into the 5^{th}, 9^{th} and 13^{th} bit slice of the adder, respectively. The sequence of inputs shown in Figure 5 causes simultaneous counter overflows in C_0 and C_1 in cycle 7. Once an overflow is detected in any of the counters, the error compensation scheme kicks in by allocating two cycles for addition of the compensating term which in the above example is $0x220$. The rationale behind allocating two cycles instead of one is that we do not want any error in addition of the correction term since it would defeat the purpose of the error compensation scheme. It is to be noted that since the same adder is utilized for addition of the correction term, the multiplexers need to propagate the correct carries to subsequent stages, which is achieved by setting the appropriate control signals. Note that due to multi-cycle operation, the error compensation operation uses the non-segmented adder (actual carry propagated by multiplexers) and is error free.

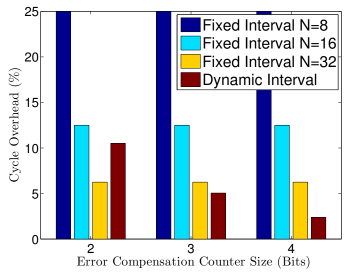


Fig. 6: Cycles overhead due to DSEC.

There are three aspects that need to be addressed in relation to the segmentation and error compensation circuitry: a) increase in critical path due to segmentation muxes, b) the cycles overhead associated with the correction term addition and c) power consumption due to the additional logic. The increase in critical path in DSEC which is purely due to the segmentation muxes (correction term accumulator does not lie in critical path) is addressed by operating the DSEC based design at the same clock period as the original design. The rationale behind such an approach is to absorb any additional errors in DSEC due to clock period violation in the overall error due to VOS. Effectively, there in *no* clock period overhead in the DSEC design. Moreover, due to the additional error compensation circuitry, DSEC exhibits better error scaling trends even when operated at clock period identical to the original design.

Figure 6 shows the cycles overhead due to correction term addition as a function of the error compensation counter bit width. We compare our proposed dynamic error compensation scheme to the fixed interval scheme in which the correction term is added every fixed number of cycles (N). The results were obtained by running 12 CIF [9] benchmark video sequences for the motion estimation algorithm for fixed $N = [8, 16, 32]$ cycles and dynamic cases. It can be observed that the dynamic scheme clearly outperforms the fixed interval scheme in terms of cycles overhead [2]. In addition, dynamic update eliminates the hazard of erroneous compensation due to counter overflow associated with the fixed interval update. Increasing the counter bit width allows us to perform error compensation less frequently, thereby reducing the cycles penalty, which is also beneficial from an energy perspective. However, increasing the counter bit width increases the power consumption of the error compensation circuitry, and needs to be performed judiciously.

The other design issue concerns power consumption due to the additional logic. At nominal V_{DD} without any preventive measures, the error compensation circuitry results in approximately 50% power overhead. In order to address this issue, we take advantage of the fact that the critical path in the error compensation circuitry is much shorter than the overall clock period. Hence, due to the large slack available, we can scale down the supply voltage in the error compensation logic without affecting its functionality. Furthermore, the switching power of the correction logic can be reduced by clock gating. Combining voltage scaling with clock gating for the error compensation circuitry allows us to reduce the associated power overhead from 50% to 13%. This overhead is outweighed by the ability to perform more aggressive VOS.

B. Delay Budgeting (DB)

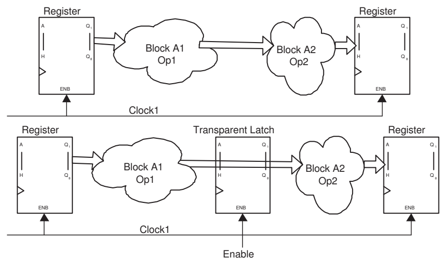


Fig. 7: Delay budgeting for chained data path components.

The second design technique for implementing meta-functions with improved scalability is based on the concept of delay budgeting for chained data path components. Chaining of arithmetic units within a clock cycle is a commonly used technique owing to the fact that the delay of two chained arithmetic units is often much lower than the sum of the delays of the individual units [12].

[2]The dynamic interval scheme appears to perform worse than the fixed interval scheme for $N = 32$ when the counter bit width is 2. However, this is misleading since in the $N = 32$ case the counter often overflows prematurely (in less than 32 cycles) resulting in a significant fraction of the carries going uncompensated. This loss in accuracy is not captured within the plot, which purely reflects the cycles overhead.

Let us consider an example of two chained arithmetic units A_1 followed by A_2 performing operations OP_1 and OP_2 respectively, as shown in Figure 7. Under VOS, with increase in gate delay, the first block A_1 is given implicit priority over the second block A_2 in terms of time available for performing its computation (since the outputs of A_1 can change arbitrarily late while the outputs of A_2 will be latched at the end of the clock period). This may be sub-optimal from the point-of-view of overall accuracy of the meta-function output, since the second block may be penalized to a point where its accuracy degrades drastically.

Figure 8 shows the scaling trend of individual arithmetic units under VOS. It can be observed that nature of resilience to errors under VOS varies with arithmetic units. Arithmetic units such as the 16-bit and 32-bit RCA scale more gracefully compared to the 8-bit subtractor and the Wallace multiplier. This serves as the primary motivation behind the proposed delay budgeting technique. Since different blocks have different error characteristics when they are provided with less-than-necessary time to compute their results, the available time (clock period) should be distributed between the chained components so as to minimize the cumulative error introduced.

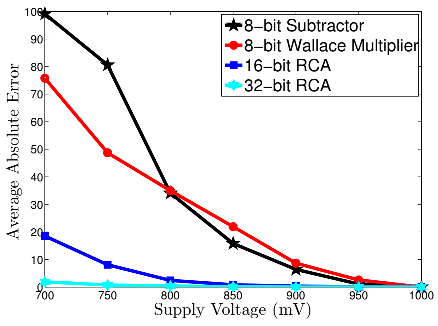
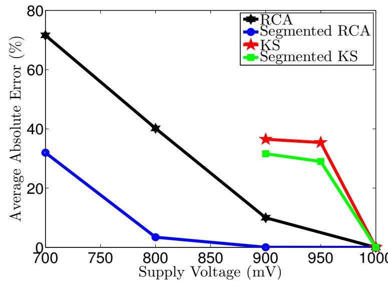
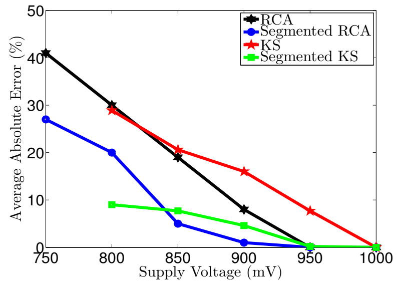


Fig. 8: Error scaling in individual data path components.

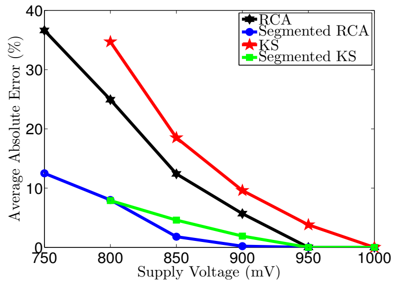
The delay budgeting scheme takes advantage of differences in scalability of error with VOS for different components, to perform an equitable distribution of delay among the chained units to achieve maximum accuracy in the meta-function result. Assuming that A_1 has better error-versus-V_{DD} characteristic than A_2, we propose the introduction of transparent latches in between the two chained arithmetic units for better apportioning of the clock period between A_1 and A_2. Depending on the control signal to the latches, the output of A_1 can be frozen at different instants of time within a clock period. This allows A_2 more time for completion of its computation, which improves the accuracy of the overall meta-function result. In the case that there are more than two chained units (such as in the L2-norm which consists of adder, multiplier and adder in the chain), we treat the last block separately and consider first adder along with multiplier as a joint unit. The two key issues that need to be addressed in the delay budgeting scheme are the energy overhead due to the transparent latches and the generation of appropriate control signals to the latches for freezing the output of A_1 at the correct instant of time. The energy overhead due to the transparent latches can be outweighed if the scalability of the meta-function under VOS



(a) L1-Norm



(b) L2-Norm



(c) Dot Product

Fig. 9: Error *vs.* voltage plots for different meta-functions using dynamic segmentation

is sufficiently improved (the meta-function can be operated at a lower V_{DD} than the conventional case, thereby offsetting the energy overhead due to the latches). The second issue pertaining to generation of control signals for the latches can be addressed in multiple ways. A chain of inverters used to delay the original clock provides multiple tapping points, each of which corresponds to a different time instant for freezing the outputs of A_1. Depending on the degree of VOS, a multiplexer is used to select the appropriate version of the delayed clock to budget the total delay between A_1 and A_2.

V. Experimental Results

In this section, we present results of transistor-level and system-level simulations that demonstrate the benefits of the proposed design techniques at the meta-function level and application level, respectively. We implemented the proposed scalable meta-functions in IBM 90nm CMOS technology. Verilog RTL descriptions were synthesized using Synopsys Design Compiler to obtain a gate-level netlist, which was then converted to a HSPICE netlist. The error rate versus voltage results for each meta-function were obtained by simulating the transistor level netlist for real sequences of input data using NanoSim. Since circuit-level simulations are slow, obtaining both application-level quality (accuracy) and energy values from circuit-level simulators is infeasible. Therefore, we used software simulations based on error profiles extracted from transistor-level simulations to obtain application-level output quality, while transistor-level simulations on sampled test benches were used to obtain energy values. The system quality results for Motion Estimation were obtained by running 12 benchmark CIF [9] video sequences. The MiLDe [13] machine learning toolkit was used for running SVM classification on the MNIST [14] data set. Phoenix [15] and the ADULT [16] data set were used for K-means clustering algorithm.

TABLE I: EDEP for different meta-function designs.

$Meta-Function$	RCA	DSEC RCA	KS	DSEC KS
L1-norm	7.48	2.44	16.00	14.00
Dot product	14.8	3.37	39.50	8.45
L2-norm	32.70	14.50	68.70	19.80

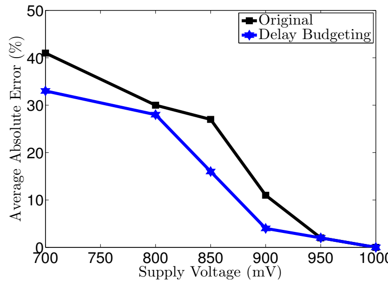
A. *Meta-Function Level Impact*

Figures 9 and 10 show the average absolute error rate versus V_{DD} for for all three meta-functions using the DSEC and delay budgeting techniques. Both linear (ripple carry adder or RCA) and logarithmic (KoggeStone or KS) adders were used. Note that even though results for RCA and KS are plotted on the same graph, the intent in these graphs is not to compare across different arithmetic units. Each of the cases (using RCA and KS) have been synthesized to their own specific critical path delay. It can be observed that DSEC outperforms the original design in terms of improved scaling in error rate with VOS. The proposed segmentation technique performs well irrespective of the type of arithmetic units (linear or logarithmic) employed. Similar results are shown in Figure 10 for the delay budgeting technique.

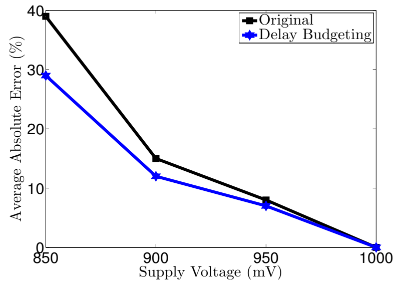
Traditionally, energy-delay product has been extensively used to compare different architectures. In order to compare the various meta-function implementations we use a slightly modified metric — *energy-delay-error product* (EDEP) — that not only takes into account energy and delay but also the error rates at scaled V_{DD}. The EDEP metric (normalized to $10^{-21} J.s.\%$) for various meta-function implementations is shown in Table I. The results clearly show the improved quality of meta-functions designed using the proposed techniques.

B. *Application Level Impact*

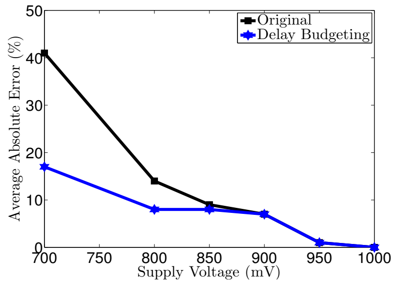
In this section, we evaluate the energy versus quality at the application level for three representative algorithms, namely Motion Estimation, SVM classification and K-means clustering. System-level simulation results for these algorithms uti-



(a) L1-Norm

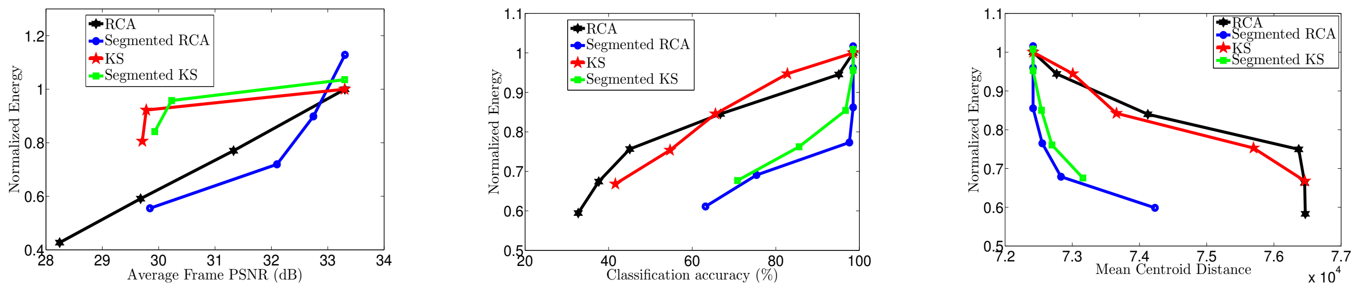


(b) L2-Norm



(c) Dot Product

Fig. 10: Error *vs.* voltage plots for different meta-functions using delay budgeting



(a) Motion Estimation (b) SVM Classification (c) K-means Clustering

Fig. 11: Application-level results: Normalized energy vs. output quality

TABLE II: Area overhead of different meta-functions.

Area(μm^2)	L1-norm	Dot product	L2-norm
RCA	1976	4100	4727
DSEC RCA	2917(47.6%)	5962(45.4%)	6649(40.6%)
KS	2551	4996	5904
DSEC KS	3384(32.7%)	6539(30.9%)	7826(32.6%)
Base DB	2192	4577	5053
Proposed DB	2327(6.2%)	4939(7.9%)	5467(8.2%)

lizing the proposed scalable implementation of L1-Norm, Dot product and L2-Norm, respectively, are shown in Figure 11. The energy values are obtained considering an underlying architecture that implements the algorithms. Since transistor-level simulations take into account VOS in logic implementing the meta-functions, we use them in conjunction with the profiling results provided in Figure 1 to estimate system level energy savings. In the case of Motion Estimation, we observe an improvement in average frame Peak Signal to Noise Ratio (PSNR) of upto 2dB at iso-energy, while energy savings upto 17% are obtained at iso-quality (PSNR). Similarly, SVM quality measured in terms of classification accuracy improves upto 52% at iso-energy, and energy savings upto 30% are achieved for iso-accuracy. We have used the mean distance-to-centroid as a measure of algorithm level quality for K-means clustering, where a smaller mean distance implies higher accuracy of clustering. We obtain 30% energy savings at iso-accuracy and 6% improvements in mean distance at iso-energy.

Table II presents the area overhead incurred by different meta-function designs. small to moderately high area overheads (6.2%∼47.6%) are incurred at the meta-function level. However, the area overhead, is significantly amortized to ∼10% when the meta-functions are incorporated in complete hardware architectures for motion estimation, SVM and K-means. The increase in critical path due to the additional circuitry varies from 1.6%∼ 20.8% depending on the meta-function implementation. However, for the sake of fairness in comparison, we obtain error and energy values at *iso-frequency*. This is achieved by operating the DSEC and delay budgeting meta-functions at the clock period of the baseline case, thereby violating its own clock period constraints. As shown in Figures 9 and 10, such an approach does not degrade the output error even at nominal V_{DD} due to the relatively low probability of occurrence of critical path activating inputs.

VI. Conclusions

We presented systematic techniques to design meta-functions that exhibit more graceful scaling behavior under VOS for use in approximate computation. Since these building blocks are the workhorses of the algorithms under consideration, improving their scalability significantly enhances the energy versus quality tradeoff at the algorithm level. Moreover, the proposed design techniques are independent of underlying architecture of the constituent arithmetic units, and can be applied to meta-functions different from those considered in this work. Therefore, we believe that the proposed concepts can be applied to realize highly energy-efficient implementations of a wide range of Multimedia, Recognition and Data Mining algorithms.

References

[1] 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.
[2] J. Bau *et al.* Error resilient system architecture (ERSA) for probabilistic applications. In 3rd Wkshp. on System Effects of Logic Soft Errors (SELSE), April 2007.
[3] Debabrata Mohapatra *et al,*. Significance driven computation: A voltage-scalable, variation-aware, quality-tuning motion estimator. In *ISLPED*, Aug. 2009.
[4] Krishna V. Palem. Energy aware algorithm design via probabilistic computing: From algorithms and models to Moore's law and novel (semiconductor) devices. In *Proc. CASES*, pages 113–116, 2003.
[5] D. Ernst *et al.* Razor: Circuit-level correction of timing errors for low-power operation. *IEEE Micro*, 24(6):10–20, 2004.
[6] Swaroop Ghosh, Swarup Bhunia, and Kaushik Roy. Crista: A new paradigm for low-power, variation-tolerant, and adaptive circuit synthesis using critical path isolation. *IEEE Trans. on CAD*, 26(11):1947–1956, 2007.
[7] Andrew B. Kahng, Seokhyeong Kang, Rakesh Kumar, and John Sartori. Slack redistribution for graceful degradation under voltage overscaling. In *Proc. Asia and South Pacific Design Automation Conference*, pages 825 –831, Jan. 2010.
[8] Nilanjan Banerjee, Georgios Karakonstantis, and Kaushik Roy. Process variation tolerant low power DCT architecture. In *DATE*, April 2007.
[9] MPEG Standard. *www.mpeg.org*.
[10] Vladimir N. Vapnik. *The nature of statistical learning theory*. Springer-Verlag New York, Inc., New York, NY, USA, 1995.
[11] J. B. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability*, pages 281–297, 1967.
[12] J. Rabaey, A. Chandrakasan, and B. Nikolic. *Digital Integrated Circuits*. Prentice-Hall, Inc., New York, NY, USA, 2003.
[13] MiLDe Machine Learning Toolkit. *www.nec-labs.com*.
[14] Yann Lecun and Corinna Cortes. The MNIST database of handwritten digits.
[15] C. Ranger *et al.* Evaluating mapreduce for multi-core and multiprocessor systems. In *Proc. Int. Symp. High Performance Computer Architecture*, pages 13–24, 2007.
[16] A. Asuncion and D. J. Newman. UCI machine learning repository, 2007.