Approximate Computing: An Emerging Paradigm For Energy-Efficient Design

Jie Han
Department of Electrical and Computer Engineering
University of Alberta
Edmonton, AB, Canada
Email: jhan8@ualberta.ca

Michael Orshansky
Department of Electrical and Computer Engineering
University of Texas at Austin
Austin, TX, USA
Email: orshansky@utexas.edu

***Abstract*— Approximate computing has recently emerged as a promising approach to energy-efficient design of digital systems. Approximate computing relies on the ability of many systems and applications to tolerate some loss of quality or optimality in the computed result. By relaxing the need for fully precise or completely deterministic operations, approximate computing techniques allow substantially improved energy efficiency. This paper reviews recent progress in the area, including design of approximate arithmetic blocks, pertinent error and quality measures, and algorithm-level techniques for approximate computing.**

Keywords—approximate computing, probabilistic computing, stochastic computation, adder, multiplier, low-energy design

I. Imprecision Tolerance and Energy Reduction

Energy-efficiency has become the paramount concern in design of computing systems. At the same time, as the computing systems become increasingly embedded and mobile, computational tasks include a growing set of applications that involve media processing (audio, video, graphics, and image), recognition, and data mining. A common characteristic of the above class of applications is that often a perfect result is not necessary and an approximate or less-than-optimal result is sufficient. It is a familiar feature of image processing, for example, that a range of image sharpness/resolution is acceptable. In data mining, simply a good output of, say, a search result, is hard to distinguish from the best result. Such applications are imprecision-tolerant.

There may be multiple sources of imprecision-tolerance [1]: (1) perceptual limitations: these are determined by the ability of the human brain to 'fill in' missing information and filter out high-frequency patterns; (2) redundant input data: this redundancy means that an algorithm can be lossy and still be adequate; and (3) noisy inputs.

The primary purpose of this paper is to review the recent developments in the area of *approximate computing* (AC). The term spans a wide set of research activities ranging from programming languages [2] to transistor level [3]. The common underlying thread in these disparate efforts is the search for solutions that allow computing systems to trade energy for quality of the computed result. In this paper we focus on the solutions that involve rethinking of how *hardware* needs to be designed. To this end, we start with an overview of several related computing paradigms and review some recently proposed approximate arithmetic circuits. Then, error metrics are introduced and algorithm-level designs are discussed. Finally, a brief summary is given.

II. Overview of Error-Resilient Paradigms

A. Approximate Computing

Here we distinguish the work on approximate computing from related but conceptually distinct efforts in probabilistic/stochastic computing. The distinctive feature of AC is that *it does not involve* assumptions on the stochastic nature of any underlying processes *implementing* the system. It does, however, often utilize *statistical* properties of data and algorithms to trade quality for energy reduction. Approximate computing, hence, employs deterministic designs that produce imprecise results.

B. Stochastic/Probabilistic Computing

Stochastic computing (SC) is a different paradigm that uses random binary bit streams for computation. SC was first introduced in the 1960s for logic circuit design [4, 5], but its origin can be traced back to von Neumann's seminal work on probabilistic logic [6]. In SC, real numbers are represented by random binary bit streams that are usually implemented in *series* and in *time*. Information is carried on the statistics of the binary streams. von Neumann's gate multiplexing technique is a special type of SC, in which redundant binary signals are implemented in *parallel* and in *space*. Both forms of SC have been the focus of recent studies [7 - 15]. SC offers advantages such as hardware simplicity and fault tolerance [8]. Its promise in data processing has been shown in several applications including neural computation [8], stochastic decoding [9, 10], fault-tolerance and image processing [11], spectral transforms [12], linear finite state machines [13] and reliability analysis [14, 15]. The notions of stochastic computation have been extended to the regime of error-resilient designs at the system, architecture and application levels [16, 17, 18]. A recent review on SC is given in [19].

A related body of work has been called *probabilistic computing*. This approach proposes exploiting intrinsic probabilistic behavior of the underlying circuit fabric, most explicitly, the stochastic behavior of a binary switch under the influence of thermal noise. Based on this principle, in [20, 21], probabilistic CMOS (PCMOS) family of circuits is proposed. An introduction to the philosophy of probabilistic computing is given in [22].

978-1-4673-6377-8/13/$31.00 ©2013 IEEE

III. Approximate Arithmetic Circuits

A. Approximate Full Adders

In several approximate implementations, multiple-bit adders are divided into two modules: the (accurate) upper part of more significant bits and the (approximate) lower part of less significant bits. For each lower bit, a single-bit approximate adder implements a modified, thus inexact function of the addition. This is often accomplished by simplifying a full adder design at the circuit level, equivalent to a process that alters some entries in the truth table of a full adder at the functional level.

1) Approximate mirror adders (AMAs): A mirror adder (MA) is a common yet efficient adder design. Five approximate MAs (AMAs) have been obtained from a logic reduction at the transistor level, i.e., by removing some transistors to attain a lower power dissipation and circuit complexity [3]. A faster charging/discharging of the node capacitance in an AMA also incurs a shorter delay. Hence, the AMAs tradeoff accuracy for energy, area and performance.

2) Approximate XOR/XNOR-based adders (AXAs): The AXAs are based on a 10-transistor adder using XOR/XNOR gates with multiplexers implemented by pass transistors. The three AXAs in [23] show attractive operational profiles in performance, hardware efficiency and power-delay product (PDP), while with a high accuracy (AXA3 in Fig.1). Although the use of pass transistors causes a reduced noise margin, the AXAs are useful when a lower accuracy can be tolerated, with significant improvements in other design metrics.

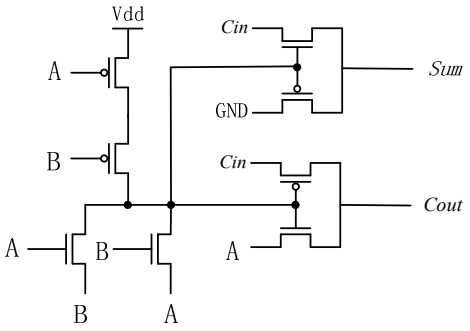

Fig. 1. Approximate XNOR-based Adder 3 (AXA3) with 8 transistors [23].

3) Lower-part-OR adder (LOA): In the LOA [24], an OR gate is used to estimate the sum of each bit at the approximate lower part, while an AND gate is used to generate the carry-in for the accurate upper part when both inputs to the most significant bit adder in the lower part are '1.' The LOA achieves an approximate but efficient operation by ignoring most carries in the less-significant lower part of an adder.

B. Multiple-Bit Approximate Adders

Current microprocessors use one of the fast parallel adders such as the carry look-ahead (CLA). The performance of parallel adders, however, is bounded by a logarithmic delay, that is, the critical path delay is asymptotically proportional to log(n) in an n-bit adder [25, 26]. Sub-logarithmic delays can however be achieved by the so-called speculative adders.

1) Speculative and variable latency adders: A speculative adder exploits the fact that the typical carry propagation chain is significantly shorter than the worst-case carry chain by using a limited number of previous input bits to calculate the sum (e.g. look-ahead k bits) [25]. If k is the square root (or independent) of n, the delay of this adder is reduced to the order of half of the logarithmic delay (or asymptotic constant). In [26], this design is treated in more detail as an almost correct adder (ACA) and developed into a variable latency speculative adder (VLSA) with error detection and recovery.

2) Error tolerant adders: A series of the so-called error tolerant adders (ETAs) are proposed in [27-30]. ETAII truncates the carry propagation chain by dividing the adder into several sub-adders; its accuracy is improved in ETAIIM by connecting carry chains in a few most significant sub-adders [27]. ETAIV further enhances the design by using an alternating carry select process in the sub-adder chain [29].

3) Speculative carry select and accuracy-configurable adders: The speculative adder in [31] employs carry chain truncation and carry select addition as a basis in a reliable variable latency carry select adder (VLCSA). The accuracy-configurable adder (ACA) enables an adaptive operation, either approximate or accurate, configurable at runtime [32].

4) Dithering adder: The result produced by the ETA adder is a bound on the accurate result. Depending on the fixed carry-in value, an upper or lower bound can be produced. That led to the idea of a *dithering* adder (Fig. 2), useful in accumulation, in which subsequent additions produce opposite-direction bounds such that the final result has a smaller overall error variance (Fig. 3) [33].

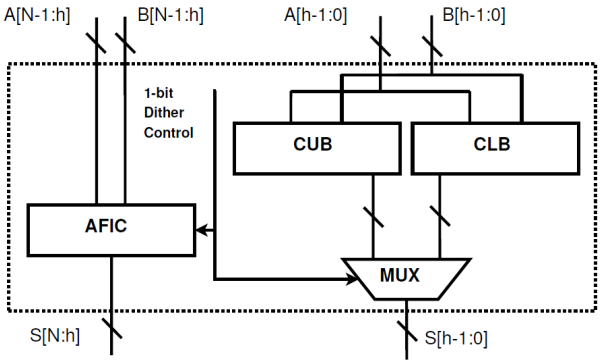

Fig. 2. Dithering adder produces alternating uppper or lower bounds on the accurate sum, resulting in reduced error variance in accumulation [33].

Fig. 3. At the same energy: bounding (left) and dithering (right) adders.

C. Approximate Multipliers

In contrast to the study of adders, the design of approximate multipliers has received relatively little attention. In [25, 34] approximate multipliers are considered by using the speculative adders to compute the sum of partial products; however, the straightforward application of approximate adders in a multiplier may not be efficient in terms of trading off accuracy for savings in energy and area. For an approximate multiplier, a key design aspect is to reduce the critical path of adding the partial products. Since multiplication is usually implemented by a cascaded array of adders, some less significant bits in the partial products are simply omitted in [24] and [35] (with some error compensation mechnisms), and thus some adders can be removed in the array for a faster operation. In [36], a simplified 2×2 multiplier is used as the building block in a larger multiplier for an efficient computation. An efficient design using input pre-processing and additional error compensation is proposed for reducing the critical path delay in a multiplier [37].

D. Approximate Logic Synthesis

Approximate logic synthesis has been considered for the design of low-overhead error-detection circuits [38]. In [33], approximate adders are synthesized for optimizing the quality-energy tradeoff. For a given function, a two-level synthesis approach is used in [39] to reduce circuit area for an error rate threshold. In [40], a multi-level logic minimization algorithm is developed to simplify the design and minimize the area of approximate circuits. Automated synthesis of approximate circuits is recently discussed in [41] for large and complex circuits under error constraints.

IV. Metrics for Approximate Computing

A. Error Rate/Frequency and Error Significance/Magnitute

In light of the advances in approximate computing, performance metrics are needed to evaluate the efficacy of approximate designs. Due to the deterministic nature of approximate circuits, the traditional metric of reliability, defined as the probability of system survival, is not appropriate for use in evaluating the quality of a design. To address this, several metrics have been used for quantifying errors in approximate designs. *Error rate* (ER) is the fraction of incorrect outputs out of a total number of inputs in an approximate circuit [42]; it is sometimes referred to as *error frequency* [33]. *Error significance* (ES) refers to the degree of error severity due to the approximate operation of a circuit [42]. ES has been considered as the numerical deviation of an incorrect output from a correct one [39], the Hamming distance of the two vectors [32], and the maximum *error magnitude* of circuit outputs [33]. The product of ER and ES is used in [40] and [43] as a composite quality metric for approximate designs. Other common metrics include the relative error, average error and error distribution.

B. Error Distance for Approximate Adders

Recently, the above metrics have been generalized to a new figure of merit, *error distance* (ED), for assessing the quality of approximate adders [44]. For an approximate design, ED is defined as the arithmetic distance between an inexact output and the correct output for a given input. For example, the two erroneous values '01' and '00' have an ED of 1 and 2 with respect to the correct number '10'. The *mean error distance* (MED) (or mean absolute error in [45]) considers the averaging effect of multiple inputs, while the *normalized error distance* (NED) is the normalization of MED for multiple-bit adders. The MED is useful in measuring the implementation accuracy of a multiple-bit adder, while the NED is a nearly invariant metric, that is, independent of the size of an adder, so it is useful when characterizing the reliability of a specific design. Moreover, the product of power and NED can be utilized for evaluating the tradeoff between power consumption and precision in an approximate design (Fig. 4). To emphasize the significance of a particular metric (such as the power or precision), a different measure with more weight on this metric can be used for a better assessment of a design according to the specific requirement of an application. These metrics are also applicable to probabilistic adders such as those in [46, 47, 48], and provide effective alternatives to an application-specific metric such as the peak signal-to-noise ratio (PSNR).

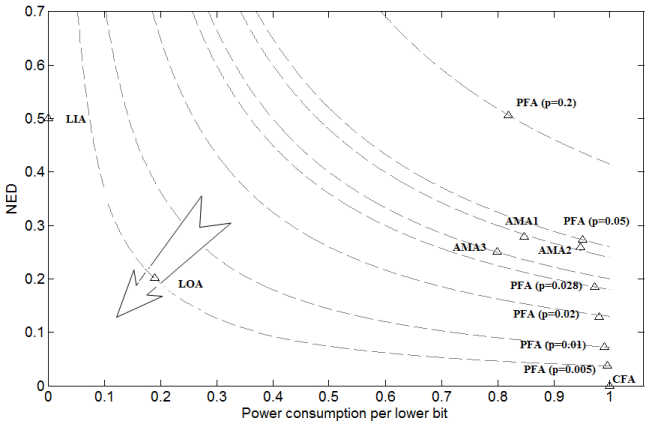

Fig. 4. Power and precision tradeoffs as given by the power consumption per bit and the NED of a full adder design [44]. The product of power per bit and NED is shown by a dashed curve. A better design with a more efficient power and precision tradeoff is along the direction pointed by the arrow.

V. Algorithm-Level Approximate Computing Techniques

Significant potential exists in using the techniques of approximate computing at the algorithm level.

A. Approximate Computing and Incremental Refinement

The notion of approximate signal processing was developed in [49, 50]. The authors introduce a central concept of *incremental refinement,* which is the property of certain algorithms, such that the iterations of an algorithm can be terminated earlier to save energy in exchange for incrementally lower quality. The principle is demonstrated on the FFT-based maximum-likelihood detection algorithm [49, 50]. It was further shown in [51-54] that several signal processing algorithms – that include filtering, frequency domain transforms and classification – can be modified to exhibit the incremental refinement property and allow favorable energy-quality trade-offs, i.e. the ones that permit energy savings in exchange for small quality degradation.

Similar principles are applied in [55] to trading energy and result optimality in an implementation of a widely used machine learning algorithm of support vector machines (SVMs). It is found that the number of support vectors correlates well with the quality of the algorithm while also impacting the algorithm's energy consumption. Reducing the number of support vectors reduces the number of dot product computations per classification while the dimensionality of the support vectors determines the number of multiply-accumulate operations per dot product. An approximate output can be computed by ordering the dimensions (features) in terms of their importance, computing the dot product in that order and stopping the computation at the proper point.

B. Dynamic Bit-Width Adaptation

For many computing and signal processing applications, one of the most powerful and easily available knobs for controlling the energy-quality trade-off is changing the operand bit-width. Dynamic, run-time adaptation of effective bit-width is thus an effective tool of approximate computing. For example, in [56] it is used for dynamic adaptation of energy costs in the discrete-cosine transform (DCT) algorithm. By exploiting the properties of the algorithm, namely, the fact that high-frequency DCT coefficients are typically small after quantization and do not impact the image quality as much as the low-frequency coefficients, lower bit-width can be used for operations on high frequency coefficients. That allows significant, e.g. 60%, power savings at the cost of only a slight, 3dB of PSNR, degradation in image quality.

C. Energy Reduction via Voltage Overscaling

In a conventional design methodology, driven by static timing analysis, timing correctness of all operations is guaranteed by construction. The design methodology guarantees that every circuit path regardless of its likelihood of excitation must meet timing. When V_{DD} is scaled even slightly, large timing errors occur and rapidly degrade the output signal quality. This rapid quality loss under voltage scaling significantly reduces the potential for energy reduction. However, because voltage scaling is the most effective way to reduce digital circuit energy consumption, many techniques of approximate computing seek ways to over-scale voltage below a circuit's safe lower voltage. They differ in how they deal with the fact that the voltage is not sufficient to guarantee timing correctness on all paths.

One possible strategy is to introduce correction mechanisms such that the system is able to tolerate timing errors induced by voltage-overscaling (VOS). In [57-60], this approach is developed under the name *algorithmic noise tolerance*, specifically targeting DSP-type circuits, such as filters. The energy reduction is enabled by using lower voltage on a main computing block and employing a simpler error correcting block that runs at a higher voltage and is thus, error-free, to improve the results impacted by timing errors of the main block. For instance, in [57] the simpler block is a linear forward predictor that estimates the current sample of the filter output based on its past samples.

Another class of approaches focuses on modifying the base implementation of a DSP algorithm to be more VOS-friendly. This can be done at several levels of design hierarchy. The principle behind most efforts is to *identify computations that need to be protected* and those that can tolerate some errors.

In [61], the idea of identifying hardware building blocks that demonstrate more graceful degradation under voltage overs-scaling is pursued. The work studies several commonly-encountered algorithms used in multimedia, recognition, and mining to identify their underlying computational kernels as *meta-functions*. For each such application, it is found that there exists a computational kernel where the algorithm spends up to 95% of its computation time, and therefore consumes the corresponding amount of energy. The following meta-functions (computational kernels) were identified: (1) for the motion estimation, it is the L1-norm, or the sum of absolute differences computation, (2) for support vector machine classification algorithm it is the dot product, and (3) for the mining algorithm of K-means clustering, it is a L1-norm or L2-norm computation. Importantly, all identified meta-functions use the accumulator which becomes the first block to experience time-starvation under voltage over-scaling. By making the accumulator more VOS-friendly, using dynamic segmentation and delay budgeting of chained units, the quality-energy trade-offs are improved for each of the above meta-functions.

Even in the same block not all computations may be equally important for the final output. In [62], the significant/insignificant computations of the sum of absolute difference algorithm, which is a part of a video motion estimation block, are identified directly based on their PSNR impact. The significant computations are then protected under VOS, by allowing them two clock cycles for completion, while the insignificant computations are allowed to produce an occasional error. A delay predictor block is used to predict the input patterns with a higher probability of launching critical paths.

It is also crucial to control sources of errors that have the potential to be spread and amplified within the flow of the algorithm [63]. For example, the 2-D inverse discrete cosine transform (IDCT) algorithm has two nearly identical sequentially executed matrix-multiplication steps. A timing error in step 1 will generate multiple output errors in the second step because each element is used in multiple computations of step 2. Therefore, it is important to prevent errors in the early steps under scaled V_{DD}. This can be achieved by allocating extra timing margins to critical steps. If the overall latency for the design needs to remain constant, an important element of protecting the early algorithm steps is a re-allocation strategy that shifts timing budgets between steps.

Different strategies are possible for dealing with errors that result from overscaling. In some designs, the results produced by blocks subject to timing errors are not directly accepted. Rather, computation is terminated early and intermediate results impacted by timing errors are ignored entirely [64, 65]. From the point of view of gate-level design, such techniques still guarantee timing correctness of all digital operations.

Alternatively, a design may directly accept the results of erroneous computation, providing, of course, that the magnitude of error is carefully controlled [63]. This *timing error acceptance* strategy gives up on guaranteeing the worst-case timing correctness but aims to keep global signal quality from severe degradation. A significant reduction of quality

loss under V_{DD} scaling is enabled by reducing the occurrence of early timing errors with large impact on quality by using operand statistics and by reducing error by dynamic reordering of accumulations. The first innovation of this effort is enabling *error control through knowledge of operand statistics*. When V_{DD} is scaled down, large magnitude timing errors are very likely to happen in the addition of small numbers with opposing signs. Such additions lead to long carry chains and are the timing-critical paths in the adder. The worst case for carry propagation occurs in the addition of −1 and 1. In 2's complement representation, this operation triggers the longest possible carry chain and, thus, experiences timing errors first. In the 2D-IDCT algorithm, the additions that involve small valued, opposite-sign operands occur in the processing of high-frequency components. This is because the first 20 low frequency components contain about 85% or more of the image energy. The introduced technique uses an adder with a bit-width smaller than required by other considerations to process high-frequency small-magnitude operands. Two objectives are achieved by using such adders: the magnitude of quality loss is reduced and its onset is delayed. Large-valued operands, of course, require a regular-width adder. The second technique is based on a reduction of the cumulative quality loss resulting from multiple additions, such as accumulations, which are a key component and optimization target of many DSP algorithms, and, specifically, of the IDCT. The key observation is that if positive and negative operands are accumulated separately and added only in the last step, the number of error-producing operations is reduced to one last addition that involves operands with opposite sign. At the same time, the operands involved in this last addition are guaranteed to be larger in absolute value than any individual opposite-sign operands involved in the original sequence of additions. This guarantees that the reordered accumulation will result in a smaller quality loss under scaled timing. The results of using the introduced techniques on two test images are shown in Fig. 5.

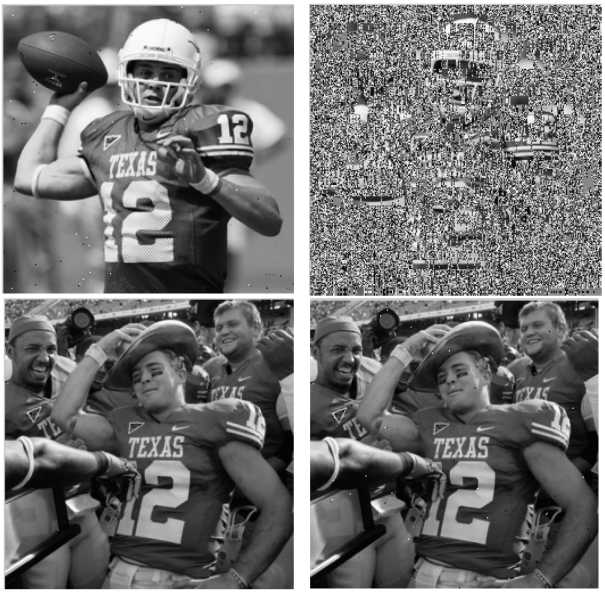

Fig. 5. Upper images are produced by a conventional IDCT with scaled V_{DD}. Techniques of [63] improve image quality for the same scaled V_{DD} in the lower images.

VI. SUMMARY

In this paper, recent progress on approximate computing is reviewed, with a focus on approximate circuit design, pertinent error metrics, and algorithm-level techniques. As an emerging paradigm, approximate computing shows great promise for implementing energy-efficient and error-tolerant systems.

REFERENCES

[1] R. Venkatesan, A. Agarwal, K. Roy, and A. Raghunathan, "MACACO: Modeling and analysis of circuits for approximate computing," in Proc. ICCAD, pp. 667–673, November 2011.

[2] H. Esmaeilzadeh, A. Sampson, L. Ceze and D. Burger, "Architecture support for disciplined approximate programming," in Proc. Intl. Conf. Architectural Support for Programming Languages and Operating Systems, pp. 301-312, 2012.

[3] V. Gupta, D. Mohapatra, A. Raghunathan and K. Roy, "Low-Power Digital Signal Processing Using Approximate Adders," IEEE Trans. CAD of Integrated Circuits and Systems, 32(1), pp. 124-137, 2013.

[4] W.J. Poppelbaump, C. Afuso and J.W. Esch, "Stochastic computing elements and systems," Proc. Fall Joint Comp. Conf., pp. 631-644, 1967.

[5] B.R. Gaines, "Stochastic computing systems," Advances in Information Systems Science, vol. 2, pp. 37-172, 1969.

[6] J. von Neumann, "Probabilistic logics and the synthesis of reliable organisms from unreliable components," Automata Studies, Shannon C.E. & McCarthy J., eds., Princeton University Press, pp. 43-98, 1956.

[7] J. Han, J. Gao, Y. Qi, P. Jonker, J.A.B. Fortes. "Toward Hardware-Redundant, Fault-Tolerant Logic for Nanoelectronics," IEEE Design and Test of Computers, vol. 22, no. 4, pp. 328-339, July/August 2005.

[8] B. Brown and H. Card, "Stochastic neural computation I: Computational elements," IEEE Trans. Computers, vol. 50, pp. 891–905, Sept. 2001.

[9] C. Winstead, V.C. Gaudet, A. Rapley and C.B. Schlegel, "Stochastic iterative decoders," Proc. Intl. Symp. Info. Theory, pp. 1116-1120, 2005.

[10] S.S. Tehrani, S. Mannor and W.J. Gross, "Fully parallel stochastic LDPC decoders," IEEE Trans. Signal Processing, vol. 56, no. 11, pp. 5692-5703, 2008.

[11] W. Qian, X. Li, M.D. Riedel, K. Bazargan and D.J. Lilja, "An architecture for fault-tolerant computation with stochastic logic," IEEE Trans. Computers, vol. 60, pp. 93–105, Jan. 2011.

[12] A. Alaghi and J.P. Hayes. "A spectral transform approach to stochastic circuits," in Proc. ICCD, pp. 315-321, 2012.

[13] P. Li, D. Lilja, W. Qian, M. Riedel and K. Bazargan, "Logical computation on stochastic bit streams with linear finite state machines." IEEE Trans. Computers, in press.

[14] J. Han, H. Chen, J. Liang, P. Zhu, Z. Yang and F. Lombardi, "A stochastic computational approach for accurate and efficient reliability evaluation," IEEE Trans. Computers, in press.

[15] H. Aliee and H.R. Zarandi, "A fast and accurate fault tree analysis based on stochastic logic implemented on field-programmable gate arrays," IEEE Trans. Reliability, vol. 62, pp. 13–22, Mar. 2013.

[16] N. Shanbhag, R. Abdallah, R. Kumar and D. Jones, "Stochastic computation," in Proc. DAC, pp. 859-864, 2010.

[17] J. Sartori, J. Sloan and R. Kumar, "Stochastic computing: embracing errors in architecture and design of processors and applications," in Proc. 14th IEEE Intl. Conf. on Compilers, Architectures and Synthesis for Embedded Systems (CASES), pp. 135-144, 2011.

[18] H. Cho, L. Leem, and S. Mitra, "ERSA: Error resilient system architecture for probabilistic applications," IEEE Trans. CAD of Integrated Circuits and Systems, vol. 31, no. 4, pp. 546-558, 2012.

[19] A. Alaghi and J.P. Hayes, "Survey of stochastic computing," ACM Trans. Embedded Computing Systems, 2012.

[20] S. Cheemalavagu, P. Korkmaz, K.V. Palem, B.E.S. Akgul and L.N. Chakrapani, "A probabilistic CMOS switch and its realization by exploiting noise," in Proc. IFIP-VLSI SoC, pp. 452-457, Oct. 2005.

[21] J. George, B. Marr, B.E.S. Akgul, and K.V. Palem, "Probabilistic arithmetic and energy efficient embedded signal processing," in Proc. Intl. Conf. on Compilers, architecture and synthesis for embedded systems (CASES), pp. 158-168, 2006.

[22] K. Palem and A. Lingamneni, "What to do about the end of Moore's law, probably!" In Proc. DAC, pp. 924-929, 2012.

[23] Z. Yang, A. Jain, J. Liang, J. Han and F. Lombardi, "Approximate XOR/XNOR-based adders for inexact computing," submitted to IEEE Conf. on Nanotechnology, 2013.

[24] H.R. Mahdiani, A. Ahmadi, S.M. Fakhraie, C. Lucas, "Bio-inspired imprecise computational blocks for efficient vlsi implementation of soft-computing applications," IEEE Trans. Circuits and Systems I: Regular Papers, vol. 57, no. 4, pp. 850-862, April 2010.

[25] S.-L. Lu, "Speeding up processing with approximation circuits," Computer, vol. 37, no. 3, pp. 67-73, 2004.

[26] A.K. Verma, P. Brisk and P. Ienne, "Variable latency speculative addition: A new paradigm for arithmetic circuit design," in Proc. DATE, pp. 1250-1255, 2008.

[27] N. Zhu, W.L. Goh and K.S. Yeo, "An enhanced low-power high-speed adder for error-tolerant application," in Proc. ISIC'09, pp. 69–72, 2009.

[28] N. Zhu, W.L. Goh, W.Zhang, K.S. Yeo and Z.H. Kong, "Design of low-power high-speed truncation-error-tolerant adder and its application in digital signal processing," IEEE Trans. VLSI Systems, 18 (8): 1225–1229, August 2010.

[29] N. Zhu, W.L. Goh, G. Wang and K.S. Yeo, "Enhanced low-power high-speed adder for error-tolerant application," in Proc. IEEE Intl. SoC Design Conf., pp. 323–327, 2010.

[30] N. Zhu, W.L. Goh and K.S. Yeo, "Ultra low-power high-speed flexible probabilistic adder for error-tolerant applications," in Proc. Intl. SoC Design Conf., pp. 393–396, 2011.

[31] K. Du, P. Varman and K. Mohanram, "High performance reliable variable latency carry select addition," in Proc. DATE, pp. 1257–1262, 2012.

[32] A.B. Kahng and S. Kang, "Accuracy-configurable adder for approximate arithmetic designs," in Proc. DAC, pp. 820-825, 2012.

[33] J. Miao, K. He, A. Gerstlauer and M. Orshansky "Modeling and synthesis of quality-energy optimal approximate adders," in Proc. ICCAD, pp. 728, 2012.

[34] J. Huang, J. Lach, and G. Robins, "A methodology for energy-quality tradeoff using imprecise hardware," in Proc. DAC, pp. 504-509, 2012.

[35] K. Y. Kyaw, W. L. Goh and K. S. Yeo, "Low power high-speed multiplier for error-tolerant application," in *Proc. IEEE Intl. Conf. of Electron Devices and Solid-State Circuits (EDSSC)*, pp. 1–4, 2010.

[36] P Kulkarni, P Gupta and M Ercegovac, "Trading accuracy for power with an underdesigned multiplier architecture," in *Proc. 24th Intl. Conf. on VLSI Design*, pp. 346–351, January 2011.

[37] C. Liu, J. Han and F. Lombardi, "A high-performance and efficient design of approximate multipliers," technical report, Univ. of Alberta.

[38] M.R. Choudhury and Kartik Mohanram, "Approximate logic circuits for low overhead, non-intrusive concurrent error detection," in Proc. DATE, pp. 903-908, 2008.

[39] D. Shin and S.K. Gupta, "Approximate logic synthesis for error tolerant applications," in Proc. DATE, pp. 957-960, 2010.

[40] D. Shin and S.K. Gupta, "A new circuit simplification method for error tolerant applications," in Proc. DATE, pp. 1-6, 2011.

[41] S. Venkataramani, A. Sabne, V. Kozhikkottu, K. Roy and A. Raghunathan, "SALSA: systematic logic synthesis of approximate circuits," in Proc. DAC, pp. 796-801, 2012.

[42] M.A. Breuer, "Intelligible test techniques to support error-tolerance," in Proc. IEEE Asian Test Symposium, pp. 386-393, 2004.

[43] I. Chong, H. Cheong, and Antonio Ortega, "New quality metric for multimedia compression using faulty hardware," in Int'l Workshop on Video Processing and Quality Metrics for Consumer Electronics, 2006.

[44] J. Liang, J. Han, F. Lombardi, "New metrics for the reliability of approximate and probabilistic adders," IEEE Trans. Computers, in press.

[45] J. Huang and J. Lach, "Exploring the fidelity-efficiency design space using imprecise arithmetic," in Proc. ASPDAC, pp. 579-584, 2011.

[46] M.S.K. Lau, K.V. Ling, Y.C. Chu and A. Bhanu, "A general mathematical model of probabilistic ripple-carry adders," in Proc. DATE, pp. 1100–1105, 2010.

[47] J. Kim and S. Tiwari, "Inexact computing for ultra low-power nanometer digital circuit design," in IEEE/ACM Intl. Symp. on Nanoscale Architectures (NANOARCH), pp. 24-31, 2011.

[48] Z.M. Kedem, V.J. Mooney, K.K. Muntimadugu and K.V. Palem, "An approach to energy-error tradeoffs in approximate ripple carry adders," in Proc. 17th IEEE/ACM Intl. Symp. on Low-power electronics and design, pp. 211-216, 2011.

[49] S.H. Nawab, A.V. Oppenheim, A.P. Chandrakasan, J.M. Winograd, J.T. Ludwig, "Approximate signal processing," VLSI Signal Processing, 15:177-200, 1997.

[50] J.T. Ludwig, S.H. Nawab and A. Chandrakasan, "Low-power digital filtering using approximate processing," IEEE Journal of Solid-State Circuits, 31(3):395-400, 1996.

[51] A. Sinha, A. Wang, and A.P. Chandrakasan, "Algorithmic transforms for efficient energy scalable computation," Intl. Symp. on Low Power Electronics and Design (ISLPED), pp. 31-36, 2000.

[52] A. Sinha and A. Chandrakasan, "Energy aware software," Intl. Conf. on VLSI Design, pp. 50-55, 2000.

[53] J. Goodman, A. Dancy, and A.P. Chandrakasan, "An energy/security scalable encryption processor using an embedded variable voltage DC/DC converter," IEEE Journal of Solid-State Circuits, 33(11):1799-1809, 1998.

[54] A. Sinha and A.P. Chandrakasan, "Energy efficient filtering using adaptive precision and variable voltage," IEEE Intl. ASIC/SOC Conference, pp. 327-331, 1999.

[55] V. Chippa, D. Mohapatra, A. Raghunathan, K. Roy, and S. Chakradhar, "Scalable effort hardware design: exploiting algorithmic resilience for energy efficiency," In Proc. DAC, pp. 555-560, 2010.

[56] J. Park, J. Choi, and K. Roy, "Dynamic bit-width adaptation in DCT: an approach to trade off image quality and computation energy," IEEE Trans. VLSI Systems, vol. 18, no. 5, pp. 787-793, May 2011.

[57] R. Hegde and N.R. Shanbhag, "Soft digital signal processing," IEEE Trans. VLSI Systems (TVLSI), 9(6):813–823, December, 2001.

[58] B. Shim and N.R. Shanbhag, "Energy-efficient soft error-tolerant digital signal processing," IEEE Trans. VLSI Systems, 14(4):336-348, 2006.

[59] B. Shim, S. Sridhara, and N. Shanbhag, "Reliable low-power digital signal processing via reduced precision redundancy," IEEE Trans. VLSI Systems (TVLSI), 12(5): 497-510, 2004.

[60] G. Varatkar and N. R. Shanbhag, "Energy-efficient motion estimation using error tolerance," Intl. Symp. on Low Power Electronics and Design (ISLPED), pp.113-118, 2006.

[61] D. Mohapatra, V.K. Chippa, A. Raghunathan, and K. Roy, "Design of voltage-scalable meta-functions for approximate computing," Proc. DATE, pp.1-6, 2011.

[62] D. Mohapatra, G. Karakonstantis, and K. Roy. 2009, "Significance driven computation: a voltage-scalable, variation-aware, quality-tuning motion estimator," in Proc. of ACM/IEEE Intl. Symp. on Low Power Electronics and Design (ISLPED), pp. 195-200, 2009.

[63] K. He, A. Gerstlauer, and M. Orshansky, "Controlled timing-error acceptance for low energy IDCT design," in Proc. DATE, pp. 1-6, 2011.

[64] G. Karakonstantis, D. Mohapatra, K. Roy, "System level DSP synthesis using voltage overscaling, unequal error protection & adaptive quality tuning," IEEE Workshop on Signal Processing Systems (SIPS), pp. 133-138, 2009.

[65] N. Banerjee, G. Karakonstantis, and K. Roy, "Process variation tolerant low power DCT architecture," in Proc. DATE, pp. 630-635, 2007.