Statistical Information Processing

Extending the Limits of Approximate Computing *

Sharad Malik Naveen Verma
Princeton University

Subhasish Mitra
Stanford University

Naresh Shanbhag
University of Illinois at Urbana-Champaign

Abstract

Application-level tolerance to approximations is commonly exploited in approximate computing research today. This paper focuses on how this can be leveraged towards tolerance to non-idealities in circuit/device fabrics (variations, defects). Errors due to the computation fabrics can arise either due to reduction of design margins, e.g., to push the limits of energy efficiency and throughput in CMOS, or due to intrinsically-high defect rates and variability in late-CMOS and non-mature post-CMOS fabrics. An architectural framework referred to as *statistical information processing* is presented, whereby algorithmic approaches, both explicit and implicit, are employed with the objective of maximizing this leverage while meeting application-level requirements. To realize this objective, this paper highlights: (i) data-error tolerant accelerators based on techniques from statistical signal processing and machine learning; (ii) effective management of catastrophic control errors; (iii) appropriate programming models; and (iv) accurate microarchitectural and architectural error modeling based on realistic device-level fault models.

1. Introduction

Approximate computing is motivated by the robustness of emerging data-centric applications to approximations in computation, data representation, communication and storage, and that acceptable approximations can significantly reduce energy consumption and/or enhance throughput. Indeed, much of the recent work in this area focuses on signal-processing, computer-vision, and inference applications. In this paper, we begin by asserting that the design of signal-processing and communication systems, in particular those for embedded energy-constrained platforms such as smart phones and others, have long recognized this application-level opportunity and have employed approximations (e.g., precision optimization, algorithmic complexity reduction, etc.) to most efficiently address application-level metrics such as bit-error rate (BER) and signal-to-noise ratio (SNR). This paper proposes to extend the domain of approximate computing by leveraging this application-level tolerance to approximations towards tolerance to non-idealities in circuit/device fabrics, e.g., reduced design margins, variations, and defects, which result in stochastic fabrics. We propose an architectural framework referred to as *statistical information processing* (SIP), which employs algorithmic approaches to maximize this leverage while meeting application-level requirements.

The circuit/device fabrics used to implement computing systems have always been stochastic. However, by-and-large, computing systems to date have been designed to produce deterministic outputs by relying on low-level mechanisms for hiding stochastic behaviors (e.g., noise margins for digital gates), all of which impose costs on system resources (energy, area, performance). Today, the increase in stochastic behaviors in advanced and emerging technologies has made the expense of low-level mechanisms unacceptable. Coupled with the application-level tolerance to approximations, this begs the question of whether stochastic behaviors in circuits/devices are best handled at the lowest levels as in the past, or instead at higher levels, where aggregate behaviors and application-level tolerance can be leveraged to a much greater extent.

How algorithmic approaches can be incorporated in the architectures to enable the SIP vision is a question that has many facets. We envision a SIP architecture to comprise a large number of data error-tolerant accelerators managed by an error-tolerant controller. In the sections below, we consider the following: 1) data error-tolerance in accelerators, ranging from statistical estimation and detection techniques to machine learning and inference; 2) effective management of control-flow errors; 3) programming models necessary for mapping statistical applications to SIPs; and 4) modeling of faults and errors, from fabrics to architectures.

2. Data Error-Tolerant Accelerators

We have proposed the use of algorithm-level techniques to compensate for data errors that arise due to reduced design margins and/or increased nanoscale variabilities and defects. These techniques achieve error compensation either explicitly, i.e., via an explicit error compensator [Statistical Error Compensation (SEC)] much as a communication receiver which compensates for channel errors, or implicitly via the use of adaptation and learning that arise in the implementation of machine-learning kernels [Data-Driven Hardware Resiliency (DDHR)]. We refer to these as Shannon-inspired or communications-inspired, as in both cases statistical inference is employed for error compensation.

The SEC techniques such as algorithmic noise-tolerance (ANT) [5], stochastic sensor network-on-a-chip (SSNOC) [7], soft NMR [6], and likelihood processing (LP) [2] all employ parametric models of signal and noise to compensate for errors. SEC consists of permitting the circuit fabric to exhibit errors beyond the lim-

* This work was supported in part by Systems on Nanoscale Information fabriCs (SONIC), one of the six SRC STARnet Centers, sponsored by MARCO and DARPA.

CONF 'yy, Month d–d, 20yy, City, ST, Country.

http://dx.doi.org/10.1145/nnnnnnn.nnnnnnn

its of application-level error-tolerance so that catastrophic loss in application-level metrics is observed, and then employing error-compensation techniques based on statistical estimation and detection to restore the observed performance to within the desired envelope. SEC techniques have shown to be effective in the presence of both dynamic errors caused by timing violations as a result of reduced design margining, e.g., voltage overscaling or soft errors due to single event upsets, as well as errors caused by defects and variations. Our approach has already been proved in theory and practice with measured results from integrated circuit implementations. The latter [1, 7] demonstrate a 600×-to-2000× enhancement in robustness (ability to handle error rates of up to 80%) and 4×-to-6× enhancement in energy efficiency over conventional deterministic/approximate designs.

Machine-learning algorithms enable data-driven methods for creating robust models of processes for which strong analytical models do not exist. This attribute gives rise to a class of methods for realizing error-tolerant computing systems. DDHR exploits machine learning to model variances caused not only due to application-level data, but also due to computational errors [11].

Emulating randomly-located stuck-at faults in digital blocks, we have observed that resulting computational errors essentially manifest as perturbations in the statistical distributions of the data being analyzed. With a strong machine-learning classifier, the new distributions can be learned in the form of an "error-aware model", enabling high inference performance even in the presence of severe fault rates. Research in DDHR has led to both an understanding of the achievable performance levels as well as system algorithms and architectures for practical implementation. With regards to system algorithms, methods have been demonstrated for fully embedded training of the error-aware model. Further, the DDHR concept has been extended to enable error-tolerance within the inference kernel itself [12]. These ideas have been employed in systems demonstrating high inference performance in fault-affected digital circuits [11], energy-efficient analog circuits [15], and variation-prone emerging technologies [8].

3. Managing Catastrophic Control Errors

While applications may be tolerant to data errors, control errors can be catastrophic. They can lead to an application crashing, hanging or executing code inconsistent with the application algorithm. Similarly, errors in operand addressing can lead to segmentation errors. One approach to manage this fragility is to enforce complete fault tolerance through expensive circuit protection mechanisms. However, our work [13] has shown that even in error-tolerant media applications, greater than 60% of instructions impact control flow or operand addressing. Thus, protecting all logic to execute these instructions would be prohibitive. Our work has considered two alternate low-cost solutions to address this. In the ERSA (Error-Resilient System Architecture) approach, the main control thread runs on a fully protected strongly reliable core (SRC), while worker threads run on mostly unprotected relaxed reliability cores (RRCs) [3]. For applications where a large number of worker threads can be spawned, the amortized overhead of the SRC is low. A subsequent work (the YMM Partially Protected Unit or PPU) focuses on reducing the protection costs of even a single core and shows how, with minimal microarchitecture support, control errors can be managed [13]. Both ERSA and the YMM-PPU follow a principle of execution-on-a-leash, by allowing control errors, but reining in large deviations using program information and architectural support.

4. Programming Models

Appropriate programming models help leverage application error-tolerance in error-tolerant execution. For data error-tolerance, this can be through annotations specifying error-tolerant data or functions, e.g., [9]. In our work, we take this a step-further to see how programming models can be utilized to manage control-flow errors. ERSA uses a `do-all` programming model to identify worker threads to be spawned on the RRCs [3]. The YMM-PPU uses the Stream-It programming model [10] to demarcate frame processing boundaries. This allows storing expected control flow as a sequence of frame processing steps, and abandoning the current frame processing by jumping to the next frame [13, 14] during large control flow errors. This exploits the error-tolerance of individual frames.

5. Fault and Error Models

Low-level physical effects result in faults in the physical computing fabric (devices, interconnect) with corresponding errors up the layers of abstraction. Understanding this propagation across the layers through accurate modeling is critical to developing architectural solutions to manage them. Error injection continues to be the dominant approach for this purpose. Hence, choosing the correct error injection technique is of primary importance. Many low-level (e.g., flip-flop-level) error injection techniques require long execution times and significant memory. On the other hand, high-level error injections at the architecture or memory levels are fast but can be inaccurate. Hence, it is essential to quantify the inaccuracies of such error injections, and to analyze the sources of these inaccuracies. Our results [4] demonstrate that inaccuracies associated with "high-level" error injection can be up to an order of magnitude on average, ranging from an underestimation by 0.07× to an overestimation by 45× in terms of observed erroneous outcome rate. Such inaccuracies can result in overly pessimistic design decisions, or optimistic design choices that may not meet application requirements. A detailed error propagation analysis explains how existing high-level error injection techniques directly model less than 1% of errors at the hardware level, resulting in such inaccuracies.

Such analyses are essential because they create new research opportunities. Fast error injection techniques combining accurate low-level simulators and fast high-level simulators (including statistical and machine learning techniques) may be used to accurately quantify the robustness of large-scale systems. They also help initiate new dialog between various research communities (e.g., system software, architecture, design automation) on cost-effective ways of designing, verifying, and qualifying robust systems.

6. Implications on Architectures

There are two main insights that drive our view on statistical information processors: (i) there is a rich set of error-tolerant accelerators that can be designed using signal processing and machine-learning techniques and (ii) control with current fine grained instruction sets is very fragile, but control errors can be managed to through limited protection and microarchitectural support. This points to architectures that are accelerator-rich and have coarse-grained instructions/macros that can utilize them. While there is an emerging trend for accelerator-rich architectures motivated by energy/execution-time benefits, our work shows the added benefit of error-tolerance. Further, in coarse-grained instruction architectures, the control flow instructions are clearly separated from data-computation and thus easier to protect, either partially or completely. Effective fault and error models, together with analysis techniques spanning various layers of abstraction, play a crucial role in understanding these trade-offs. We are currently designing a statistical information processor based on this learning from our foundational work briefly discussed above.

References

[1] R. A. Abdallah and N. R. Shanbhag. A 14.5 fj/cycle/k-gate, 0.33 v ecg processor in 45nm cmos using statistical error compensation. In *Custom Integrated Circuits Conference (CICC), 2012 IEEE*, pages 1–4. IEEE, 2012.

[2] R. A. Abdallah and N. R. Shanbhag. Robust and energy-efficient multimedia systems via likelihood processing. *IEEE Tran. on Multimedia*, 15(2), February 2013.

[3] H. Cho, L. Leem, and S. Mitra. Ersa: Error resilient system architecture for probabilistic applications. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 31(4):546–558, 2012.

[4] H. Cho, S. Mirkhani, C.-Y. Cher, J. A. Abraham, and S. Mitra. Quantitative evaluation of soft error injection techniques for robust system design. In *Design Automation Conference (DAC), 2013 50th ACM/EDAC/IEEE*, pages 1–10. IEEE, 2013.

[5] R. Hegde and N. R. Shanbhag. Soft digital signal processing. *IEEE Tran. on VLSI Systems*, 9(6), December 2001.

[6] E. P. Kim and Shanbhag. Soft n-modular redundancy. *IEEE Tran. on Computers*, 61(3), March 2012.

[7] E. P. Kim, D. J. Baker, S. Narayanan, N. R. Shanbhag, and D. L. Jones. A 3.6-mw 50-mhz pn code acquisition filter via statistical error compensation in 180-nm cmos. *IEEE Tran. on VLSI Systems*, 23(3), March 2015.

[8] W. Rieutort-Louis, T. Moy, Z. Wang, S. Wagner, J. C. Sturm, and N. Verma. A large-area image sensing and detection system based on embedded thin-film classifiers. In *Intl Solid-State Circuits Conf. Tech. Dig. of Papers*, pages 292–293, Feb. 2015.

[9] A. Sampson, W. Dietl, E. Fortuna, D. Gnanapragasam, L. Ceze, and D. Grossman. Enerj: approximate data types for safe and general low-power computation. In *Programming Language Design and Impl.*, 2011.

[10] W. Thies, M. Karczmarek, and S. Amarasinghe. Streamit: A language for streaming applications. In *Compiler Construction*, pages 179–196. Springer, 2002.

[11] Z. Wang, K. H. Lee, and N. Verma. Overcoming computational errors in low-power sensing platforms through embedded machine-learning kernels. *IEEE Tran. on VLSI Systems*, PP(99), Aug. 2014. .

[12] Z. Wang, R. Schapire, and N. Verma. Error-adaptive classifier boosting (EACB): Exploiting data-driven training for highly fault-tolerant hardware. In *Int. Conf. on Acoustics, Speech and Signal Processing*, pages 3884–3888, 2014.

[13] Y. Yetim, M. Martonosi, and S. Malik. Extracting useful computation from error-prone processors for streaming applications. In *Design, Automation Test in Europe Conference Exhibition (DATE), 2013*, March 2013.

[14] Y. Yetim, S. Malik, and M. Martonosi. Commguard: Mitigating communication errors in error-prone parallel execution. In *Proceedings of the Twentieth International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS '15, 2015.

[15] J. Zhang, Z. Wang, and N. Verma. A matrix-multiplying ADC implementing a machine-learning classifier directly with data conversion. In *Intl Solid-State Circuits Conf. Tech. Dig. of Papers*, pages 332–333, Feb. 2015.