Stochastic Computing

Stochastic Computing

John Sartori and Rakesh Kumar

University of Illinois at Urbana-Champaign
Urbana, IL 61801
USA
{*sartori2@illinois.edu;rakeshk@illinois.edu*}

now
the essence of knowledge

Boston – Delft

Foundations and Trends® in Electronic Design Automation

Published, sold and distributed by:
now Publishers Inc.
PO Box 1024
Hanover, MA 02339
USA
Tel. +1-781-985-4510
www.nowpublishers.com
sales@nowpublishers.com

Outside North America:
now Publishers Inc.
PO Box 179
2600 AD Delft
The Netherlands
Tel. +31-6-51115274

The preferred citation for this publication is J. Sartori and R. Kumar, Stochastic Computing, Foundations and Trends® in Electronic Design Automation, vol 5, no 3, pp 153–210, 2011

ISBN: 978-1-60198-508-8

Foundations and Trends® in Electronic Design Automation

Volume 5 Issue 3, 2011

Editorial Scope

Foundations and Trends® in Electronic Design Automation will publish survey and tutorial articles in the following topics:

- System Level Design
- Behavioral Synthesis
- Logic Design
- Verification
- Test
- Physical Design
- Circuit Level Design
- Reconfigurable Systems
- Analog Design

Information for Librarians
Foundations and Trends® in Electronic Design Automation, 2011, Volume 5, 4 issues. ISSN paper version 1551-3939. ISSN online version 1551-3947. Also available as a combined paper and online subscription.

Foundations and Trends® in
Electronic Design Automation
Vol. 5, No. 3 (2011) 153–210
© 2011 J. Sartori and R. Kumar
DOI: 10.1561/1000000021

now
the essence of knowledge

Stochastic Computing

John Sartori and Rakesh Kumar

University of Illinois at Urbana-Champaign, 1308 W. Main St., Urbana, IL 61801, USA, {sartori2@illinois.edu;rakeshk@illinois.edu}

Abstract

As device sizes shrink, manufacturing challenges at the device level are resulting in increased variability in physical circuit characteristics. Exponentially increasing circuit density has not only brought about concerns in the reliable manufacturing of circuits but also has exaggerated variations in dynamic circuit behavior. The resulting uncertainty in performance, power, and reliability imposed by compounding static and dynamic nondeterminism threatens the continuation of Moore's law, which has been arguably the primary driving force behind technology and innovation for decades. This situation is exacerbated by emerging computing applications, which exert considerable power and performance pressure on processors. Paradoxically, the problem is not nondeterminism, *per se*, but rather the approaches that designers have used to deal with it. The traditional response to variability has been to enforce determinism on an increasingly nondeterministic substrate through guardbands. As variability in circuit behavior increases, achieving deterministic behavior becomes increasingly expensive, as performance and energy penalties must be paid to ensure that all devices work correctly under all possible conditions. As such, the benefits of technology scaling are vanishing, due to the overheads of dealing with hardware variations through traditional means. Clearly, *status quo* cannot continue.

Despite the above trends, the contract between hardware and software has, for the most part, remained unchanged. Software expects flawless results from hardware under all possible operating conditions. This rigid contract leaves potential performance gains and energy savings on the table, sacrificing efficiency in the common case in exchange for guaranteed correctness in all cases. However, as the marginal benefits of technology scaling continue to languish, a new vision for computing has begun to emerge. Rather than hiding variations under expensive guardbands, designers have begun to relax traditional correctness constraints and deliberately expose hardware variability to higher levels of the compute stack, thus tapping into potentially significant performance and energy benefits and also opening the potential for errors. Rather than paying the increasing price of hiding the true, stochastic nature of hardware, emerging *stochastic computing* techniques account for the inevitable variability and exploit it to increase efficiency. Stochastic computing techniques have been proposed at nearly all levels of the computing stack, including stochastic design optimizations, architecture frameworks, compiler optimizations, application transformations, programming language support, and testing techniques. In this monograph, we review work in the area of stochastic computing and discuss the promise and challenges of the field.

Contents

1

Introduction

The primary driver for innovations in computer systems has been the phenomenal scalability of the semiconductor manufacturing process, governed by Moore's law, that has allowed us to literally print circuits and systems growing at exponential capacities for the last three decades. The resulting exponentially reducing cost per function has resulted in an unprecedented penetration of technology in homes and beyond, leading to profound impacts on society and quality of life.

Moore's law has come under threat, however, due to the resulting exponentially deteriorating effects of material properties on chip reliability and power. As transistors become smaller (the oxide in a 22 nm process is only five atomic layers thick, and gate length is only 42 atoms across), it is becoming increasingly expensive for the current design and manufacturing technology to keep transistors functioning deterministically, even under normal operating conditions. There are three primary sources of nondeterminism [23]. First, decreasing transistor sizes lead to different transistors being doped differently during the manufacturing process, causing them to have nondeterministic electrical characteristics [24]. Second, transistors have become smaller than the wavelength of the light used to pattern them (by 6×) [2]. This causes nondeterminism in the dimensions and characteristics of the manufactured transistors. Finally, the unprecedented increase in the power density of chips, coupled with time and context-dependent variation in temperature and utilization across the chip, cause voltage and timing variations in circuits [7]. These variations are dynamic and largely nondeterministic. The most immediate impact of such nondeterminism is decreased chip yields. A growing number of parts are thrown away since they do not meet timing and power-related specifications. A 5% yield loss on a 90 nm process today directly translates into a cost to the manufacturer that exceeds 2× the design cost for a typical cell-phone manufacturer [19], arguably one of the highest volume parts. Clearly the *status quo* cannot continue. Left unaddressed, the entire computing and information technology industry will soon face the prospect of parts that neither scale in capability nor cost. We must find a solution to the nondeterminism problem if semiconductor technology and industry are to remain a viable driver of science innovation and technology capabilities for the future.

Paradoxically, the problem is not nondeterminism, *per se*, but how computer system designers approach it. Chip components no longer behave like the precisely chiseled machines of the past; yet, the basic approach to designing and operating computing machines has remained unchanged. While there have been many swings in computing platform paradigms, such as from general-purpose to specialized, and from single-core to multi-core, the contract between hardware and software has remained unchanged. This contract guarantees that hardware will return correct values for every computation, under all conditions. In other words, we demand hardware to be overdesigned to meet the mindsets in computer systems and software design of the past. Guardbands imposed upon hardware result in increased cost [12], because getting the last bit of performance incurs too much area and power overhead, especially if performance is to be optimized for all possible computations. Conservative guardbands also leave enormous performance and energy potential untapped, since the software assumes lower performance than what a majority of instances of that platform may be capable of attaining most of the time.

As the marginal benefits of technology scaling continue to languish, a new vision for computing has begun to emerge. Rather than hiding variations under expensive guardbands, designers have begun to relax traditional correctness constraints and deliberately expose hardware variability to higher levels of the compute stack, thus tapping into potentially significant energy benefits, but also opening up the potential for errors. Rather than paying the increasing price of hiding the true, stochastic nature of hardware, emerging *stochastic computing* techniques exploit error resilience by exposing hardware errors and allowing hardware reliability to be traded for increased energy efficiency.

Truly, designers have begun to embrace stochasticity at many layers of the compute stack [37]. Resulting *stochastic architecture frameworks* [20, 29, 38, 42] are structured around stochastic computing techniques, which they exploit to enable energy-reliability tradeoffs and increase efficiency in the face of nondeterministic hardware. For example, the Variation-Adaptive Stochastic Computer Organization (VASCO) [38] dynamically adapts its own hardware reliability based on workload characteristics and environmental conditions to maximize the energy and performance benefits of exploiting error resilience.

Architecture frameworks such as VASCO rely on *microarchitecture and design-level techniques that manipulate the error distribution* of hardware to enable energy-reliability tradeoffs and increase the efficiency of operating in the stochastic domain. Design-level stochastic computing techniques [11, 16, 18, 25, 26, 27, 33, 43, 44] aim to make stochastic computing itself more efficient. For example, recovery-driven design [26] assumes a stochastic operating condition and the availability of hardware or software error resilience and asks how circuits can be designed to be more efficient when errors are allowed even in the nominal case. Microarchitectural techniques [35] attempt to reduce the number of errors that a processor produces for a given operating condition. Compiler optimizations [36, 37] can further improve the extent of benefits on programmable stochastic architectures by manipulating the activity distribution of a stochastic processor to reduce the error rate or cost of error recovery.

Programming language [13, 32] and application [37, 39] support for stochastic processors facilitates programming for stochastic processors and enables more programs to be executed on stochastic processors. Testing techniques [4] are also being re-evaluated so as to not require all parts of the chip to function flawlessly. The focus is especially on increasing chip yields in spite of permanent faults.

Going forward, as variability continues to grow and designers continue to abandon traditional variability-mitigation practices and embrace stochastic computing, stochastic design optimizations, architecture frameworks, compiler optimizations, application transformations, programming language support, and testing techniques will be essential to maximize the potential of stochastic computing to increase performance and energy efficiency.

In the first section of this chapter, we discuss design-level optimization techniques that aim to enable energy-reliability tradeoffs and improve the efficiency of error resilient designs.

References

[1] T. Austin, V. Bertacco, D. Blaauw, and T. Mudge, "Opportunities and challenges for better than worst-case design," in *Proceedings of ASPDAC*, pp. 2–7, 2005.

[2] K. Bernstein, D. Frank, A. Gattiker, W. Haensch, B. Ji, S. Nassif, E. Nowak, D. Pearson, and N. Rohrer, "High-performance CMOS variability in the 65-nm regime and beyond," *IBM Journal of Research and Development*, vol. 50, no. 4, pp. 433–449, 2006.

[3] K. Bowman, J. Tschanz, C. Wilkerson, S. Lu, T. Karnik, V. De, and S. Borkar, "Circuit techniques for dynamic variation tolerance," in *DAC*, pp. 4–7, 2009.

[4] M. Breuer, "Intelligible test techniques to support error-tolerance," in *ATS*, pp. 386–393, 2004.

[5] M. Breuer and S. Gupta, "Intelligible testing," in *MTAS*, 1999.

[6] T. Burd, S. Member, T. Pering, A. Stratakos, and R. Brodersen, "A dynamic voltage scaled microprocessor system," *IEEE Journal of Solid-State Circuits*, vol. 11, no. 35, pp. 1571–1580, 2000.

[7] Y. Cao, P. Gupta, A. Kahng, D. Sylvester, and J. Yang, "Design sensitivities to variability: Extrapolations and assessments in nanometer VLSI," in *Proceedings of IEEE ASIC/SOC Conference*, pp. 411–415, 2002.

[8] L. Chakrapani, B. Akgul, S. Cheemalavagu, P. Korkmaz, K. Palem, and B. Seshasayee, "Ultra-efficient (Embedded) SOC architectures based on probabilistic CMOS (PCMOS) technology," in *Proceedings of DATE*, pp. 1110–1115, 2006.

[9] V. Chippa, D. Mohapatra, A. Raghunathan, K. Roy, and S. Chakradhar, "Scalable effort hardware design: Exploiting algorithmic resilience for energy efficiency," in *DAC*, pp. 555–560, 2010.

[10] N. Choudhary, S. Wadhavkar, T. Shah, S. Navada, H. Najaf-abadi, and E. Rotenberg, "FabScalar," in *WARP*, 2009.

[11] J. Cong and K. Minkovich, "Logic synthesis for better than worst-case designs," in *VLSI-DAT*, pp. 166–169, 2009.

[12] S. Das, C. Tokunaga, S. Pant, W. Ma, S. Kalaiselvan, K. Lai, D. Bull, and D. Blaauw, "Razor II: In situ error detection and correction for PVT and SER tolerance," in *Proceedings of ISSCC*, pp. 400–622, 2008.

[13] M. de Kruijf, S. Nomura, and K. Sankaralingam, "Relax: An architectural framework for software recovery of hardware faults," in *ISCA*, 2010.

[14] S. Dhar, D. Maksimovic, and B. Kranzen, "Closed-loop adaptive voltage scaling controller for standard-cell ASICS," in *IEEE/ACM ISLPED*, pp. 103–107, 2002.

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

[16] S. Ghosh, S. Bhunia, and K. Roy, "CRISTA: A new paradigm for low-power, variation-tolerant, and adaptive circuit synthesis using critical path isolation," *IEEE TCAD*, vol. 26, no. 11, pp. 1947–1956, 2007.

[17] B. Greskamp and J. Torrellas, "Paceline: Improving single-thread performance in nanoscale CMPs through core overclocking," in *PACT*, 2007.

[18] B. Greskamp, L. Wan, W. R. Karpuzcu, J. J. Cook, J. Torrellas, D. Chen, and C. Zilles, "BlueShift: Designing processors for timing speculation from the ground up," in *Proceedings of IEEE HPCA*, pp. 213–224, 2009.

[19] P. Gupta, "Design for ultra-low-k1 patterning and manufacturing," in *Tutorial at ICMTS*, 2009.

[20] R. Hegde and N. Shanbhag, "Energy-efficient signal processing via algorithmic noise-tolerance," in *ISLPED*, pp. 30–35, 1999.

[21] H. Hoffmann, S. Sidiroglou, M. Carbin, S. Misailovic, A. Agarwal, and M. Rinard, "Dynamic knobs for responsive power-aware computing," in *ASPLOS*, pp. 199–212, 2011.

[22] K. Huang and J. Abraham, "Algorithm-based fault tolerance for matrix operations," *IEEE Transactions on Computing*, vol. 33, no. 6, pp. 518–528, 1984.

[23] ITRS, "International technology roadmap for semiconductors," http://www.itrs.net/reports.html, 2010.

[24] S. Jones, "Exponential trends in the integrated circuit industry," http://www.icknowledge.com, 2008.

[25] A. Kahng, S. Kang, R. Kumar, and J. Sartori, "Designing processors from the ground up to allow voltage/reliability tradeoffs," in *IEEE HPCA*, 2010.

[26] A. Kahng, S. Kang, R. Kumar, and J. Sartori, "Recovery-driven design: A methodology for power minimization for error tolerant processor modules," in *ACM/IEEE DAC*, 2010.

[27] A. Kahng, S. Kang, R. Kumar, and J. Sartori, "Slack redistribution for graceful degradation under voltage overscaling," in *IEEE/SIGDA ASPDAC*, 2010.

[28] T. Kehl, "Hardware self-tuning and circuit performance monitoring," *IEEE International Conference on Computer Deisgn*, pp. 188–192, 1993.

[29] L. Leem, H. Cho, J. Bau, Q. Jacobson, and S. Mitra, "ERSA: Error resilient system architecture for probabilistic applications," 2010.

[30] S. Palacharla, N. Jouppi, and J. Smith, "Complexity-effective superscalar processors," in *ISCA*, 1997.

[31] Y. Pan, J. Kong, S. Ozdemir, G. Memik, and S. Chung, "Selective wordline voltage boosting for caches to manage yield under process variations," in *DAC*, pp. 57–62, 2009.

[32] 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 *PLDI*, 2011.

[33] S. Sarangi, B. Greskamp, A. Tiwari, and J. Torrellas, "EVAL: Utilizing processors with variation-induced timing errors," in *IEEE/ACM MICRO*, pp. 423–434, 2008.

[34] J. Sartori and R. Kumar, "Overscaling-friendly timing speculation architectures," in *ACM/IEEE GLSVLSI*, 2010.

[35] J. Sartori and R. Kumar, "Architecting processors to allow voltage/reliability tradeoffs," in *CASES*, 2011.

[36] J. Sartori and R. Kumar, "Compiling for timing error resilient processors," in *TECHCON*, 2011.

[37] J. Sartori, J. Sloan, and R. Kumar, "Stochastic computing: Embracing errors in architecture and design of processors and applications," in *CASES*, 2011.

[38] N. Shanbhag, R. Abdallah, R. Kumar, and D. Jones, "Stochastic computation," in *Proceedings of DAC*, pp. 859–864, 2010.

[39] J. Sloan, D. Kesler, R. Kumar, and A. Rahimi, "A numerical optimization-based methodology for application robustification: Transforming applications for error tolerance," in *DSN*, 2010.

[40] Sun, "Sun OpenSPARC project," http://www.sun.com/processors/opensparc/, 2010.

[41] University of Michigan, "Bug Underground," http://bug.eecs.umich.edu/, 2007.

[42] G. Varatkar, S. Narayanan, N. Shanbhag, and D. Jones, "Variation-tolerant, low-power PN-code acquisition using stochastic Sensor NOC," 2008.

[43] L. Wan and D. Chen, "DynaTune: Circuit-level optimization for timing speculation considering dynamic path behavior," in *ICCAD*, 2009.

[44] Y. Yetim, W. Jia, S. Malik, M. Martonosi, and K. Shaw, "A system-level ISA and its applications to energy-performance-reliability scheduling and scratch-pad allocation," http://www.gigascale.org/pubs/2341.html, 2010.