Error-Efficient Computing Systems

Phillip Stanley-Marbell
University of Cambridge
phillip.stanley-marbell@eng.cam.ac.uk

Martin Rinard
Massachusetts Institute of Technology
rinard@csail.mit.edu

the essence of knowledge
Boston — Delft

Full text available at: http://dx.doi.org/10.1561/1000000049

Foundations and Trends® in Electronic Design Automation

Published, sold and distributed by:
now Publishers Inc.
PO Box 1024
Hanover, MA 02339
United States
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

P. Stanley-Marbell and M. Rinard. *Error-Efficient Computing Systems*. Foundations and Trends® in Electronic Design Automation, vol. 11, no. 4, pp. 362–461, 2017.

This Foundations and Trends® issue was typeset in LaTeX using a class file designed by Neal Parikh. Printed on acid-free paper.

ISBN: 978-1-68083-358-4
© 2017 P. Stanley-Marbell and M. Rinard

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, mechanical, photocopying, recording or otherwise, without prior written permission of the publishers.

Photocopying. In the USA: This journal is registered at the Copyright Clearance Center, Inc., 222 Rosewood Drive, Danvers, MA 01923. Authorization to photocopy items for internal or personal use, or the internal or personal use of specific clients, is granted by now Publishers Inc for users registered with the Copyright Clearance Center (CCC). The 'services' for users can be found on the internet at: www.copyright.com

For those organizations that have been granted a photocopy license, a separate system of payment has been arranged. Authorization does not extend to other kinds of copying, such as that for general distribution, for advertising or promotional purposes, for creating new collective works, or for resale. In the rest of the world: Permission to photocopy must be obtained from the copyright owner. Please apply to now Publishers Inc., PO Box 1024, Hanover, MA 02339, USA; Tel. +1 781 871 0245; www.nowpublishers.com; sales@nowpublishers.com

now Publishers Inc. has an exclusive license to publish this material worldwide. Permission to use this content must be obtained from the copyright license holder. Please apply to now Publishers, PO Box 179, 2600 AD Delft, The Netherlands, www.nowpublishers.com; e-mail: sales@nowpublishers.com

Full text available at: http://dx.doi.org/10.1561/1000000049

Foundations and Trends® in Electronic Design Automation
Volume 11, Issue 4, 2017

Editorial Board

Editorial Scope

Topics

Foundations and Trends® in Electronic Design Automation publishes 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
- Embedded software and parallel programming
- Multicore, GPU, FPGA, and heterogeneous systems
- Distributed, networked embedded systems
- Real-time and cyberphysical systems

Information for Librarians

Foundations and Trends® in Electronic Design Automation, 2017, Volume 11, 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. 11, No. 4 (2017) 362–461

DOI: 10.1561/1000000049

the essence of knowledge

Error-Efficient Computing Systems

Phillip Stanley-Marbell
University of Cambridge
phillip.stanley-marbell@eng.cam.ac.uk

Martin Rinard
Massachusetts Institute of Technology
rinard@csail.mit.edu

Contents

Abstract

This survey explores the theory and practice of techniques to make computing systems faster or more energy-efficient by allowing them to make controlled errors. In the same way that systems which only use as much energy as necessary are referred to as being *energy-efficient*, you can think of the class of systems addressed by this survey as being *error-efficient*: They only prevent as many errors as they need to. The definition of what constitutes an error varies across the parts of a system. And the errors which are acceptable depend on the application at hand.

In computing systems, making errors, when behaving correctly would be too expensive, can conserve resources. The resources conserved may be time: By making some errors, systems may be faster. The resource may also be energy: A system may use less power from its batteries or from the electrical grid by only avoiding certain errors while tolerating benign errors that are associated with reduced power consumption. The resource in question may be an even more abstract quantity such as consistency of ordering of the outputs of a system.

This survey is for anyone interested in an end-to-end view of one set of techniques that address the theory and practice of making computing systems more efficient by trading errors for improved efficiency.

P. Stanley-Marbell and M. Rinard. *Error-Efficient Computing Systems*. Foundations and Trends® in Electronic Design Automation, vol. 11, no. 4, pp. 362–461, 2017.
DOI: 10.1561/1000000049.

Full text available at: http://dx.doi.org/10.1561/1000000049

1

Introduction

> All software eventually works;
> all hardware eventually fails.
>
> —Clod Berrera.

This review explores the theory and practice of techniques to make computing systems faster or more energy-efficient by allowing them to make controlled errors. In the same way that systems which only use as much energy as necessary are referred to as being *energy-efficient*, you can think of the class of systems addressed by this review as being *error-efficient*: they only prevent as many errors as they need to.

There are numerous related fields relevant to understanding, designing, and evaluating systems which trade controlled errors for improved performance or energy efficiency. These related fields range from sub-areas of computer science, electrical engineering, and materials science, to applied mathematics and psychophysics (the study of perception). There are numerous techniques proposed by researchers in these diverse areas, with a vibrant and growing body of research results. This review focuses on two elements:

- **Fundamental concepts** that underpin any exploration of errors, time-efficiency (i.e., performance), and energy efficiency. These concepts have been developed over many decades in areas ranging from numerical analysis to the physics of semiconductor device behavior.

- **Practical hardware and software implementations** of error-efficient techniques to reduce energy usage in either practical engineering applications or experimental research platforms.

Throughout the review, we will focus specifically on the interplay between errors and the effects of errors as processed by human perception.

1.1 The Cost of Correctness

In computing systems, making errors when behaving correctly would be too expensive can conserve resources. The resources conserved in doing so may be *time*: by making some errors, they may be faster. The resource may also be *energy*: a system may use less power from its batteries or from the electrical grid by only avoiding certain errors while tolerating benign errors that are associated with reduced power consumption. The resource in question may be an even more abstract quantity such as consistency of ordering of the outputs of the system in question.

Which errors are acceptable depends on the application. The degree to which resources such as time or energy can be conserved likewise depends on the design of the computing system. And there are many different kinds of deviations in behavior which can be classified as “errors”. This Chapter provides an overview of the landscape of the applications, computing systems, and techniques that can be used to trade improved efficiency in exchange for occasional errors.

1.2 Historical Context

All hardware eventually fails. Reducing the likelihood of failure and the effects of failure comes at the cost of time, energy, or space. Making computing hardware more reliable was particularly important when the dominant applications of computing systems were in controlling weaponry and in financial applications. Today however, a large fraction of computing systems generate output solely for visual consumption.

Early computing systems based on vacuum tubes provided improvements in switching speed over their predecessors which were based on mechanical relays. They however also failed frequently: Failure rates in early vacuum-tube-based systems were as high as once every eight hours [von Neumann, 1956]. Because the possibility of intermittent and permanent failures has always been present in computing systems, the design of the basic elements of computation has evolved over time to inherently attempt to counteract the effects of failures.

One of the most fundamental techniques for dealing with the most basic source of failures (environmental noise) is to use digital logic, instead of performing computation directly in the analog signal domain. There is a rich body of work studying the tradeoffs between digital and analog computation, as well as on techniques to reduce both manufacture-time defects and runtime faults [Bushnell and Agrawal, 2000].

Redundancy, either in energy, space, or time, is a common approach used in digital logic to overcome the effects of noise. Error-correcting codes [Hamming, 1950] use redundancy in the representation of information to make it possible to detect and correct errors; the particular kinds and numbers of errors that can be detected and corrected depend on the amount of redundancy employed.

At a coarser grain, redundancy is also employed across complete computing systems, such as by replicating entire processors, complete servers, or even by replicating clusters and data centers. The challenges involved in such *fault-tolerant computing systems* are also the subject of a rich area of study [Avižienis et al., 2004].

Unlike traditional applications of computing systems, many modern applications of computation are in situations where the inputs to the system are from sources which are themselves noisy, unlike the inputs to a payroll application. Examples are the computations on sensor values in the many variants of health-tracking wearables. Similarly, the outputs of many applications are primarily for consumption via the human visual channel; an example is the rendering of images for a display. These applications could of course continue to be implemented with the level of redundancy used to guard against errors in traditional applications. Employing redundancy in space, time, and energy, independent of the needs of individual applications would likely have continued to be the way all computing systems are built. However, as the amount of energy used in a single logic operation reduced over time due to semiconductor process technology improvements, the overhead of the redundancy has become significant.

In those applications which do not require the same extremely low levels of errors, it is therefore now interesting to design systems which can trade errors for efficiency. And it is possible to go even further, to induce controlled amounts of errors if doing so would enable simpler, faster, cheaper, or more energy-efficient computing systems.

1.3 Why Precision Matters in Many Numerical Computations

There are many important computations whose implementations require careful attention to numerical stability, however few implementors of large-scale scientific computations have deep knowledge of numerical analysis. In the absence of such expertise, an alternative is to employ greater numerical precision [Bailey, 2005]. Because there are few automated techniques for transforming applications to improve their numerical stability [Panchekha et al., 2015], high-precision computations will continue to be important for a large class of applications. One example of a system where higher precision was used as an expedient solution to numerical instability is illustrated in the work of He and Ding [2001], who showed how problems with the reproducibility of climate-modeling applications could be eliminated by switching to using 128-bit floating-point arithmetic. A central theme throughout this review is that the types and magnitudes of errors permissible in an application must always be considered in the context of the tradeoff between errors and resource usage: a technique should permit only as many errors as an application and context can tolerate. Techniques should weigh permitted errors against the improvement in resource usage obtained from permitting errors. One way to achieve this in numerical simulations is to use multiple levels of precision across the phases of computations.

One cause of numerical instability in the presence of errors is that most general-purpose computations have great *arithmetic depth* [von Neumann and Kurzweil, 2012]. Small errors may therefore get amplified across the steps of a computation.

1.4 Why Some Applications Can Tolerate Errors

Despite the fact that many applications *cannot* tolerate any errors in their computations, there are also many applications which can. Typically, the applications that can tolerate errors are those that either:

1. **Operate on noisy inputs** (e.g., readings from sensors).
2. **Have computation outputs requiring limited precision**, e.g., because they are consumed primarily by human vision.
3. **Employ iterative or self-policing algorithms**. Examples of such algorithms are iterative methods where the computation will still produce the correct output in the presence of errors, provided that the computation makes progress in the right direction (on average) during each iteration.
4. **Do not have data-dependent control-flow**.

1.5 Examples of Improving Efficiency by Permitting Errors

Because displays account for a large fraction of the power dissipation in popular computing platforms such as mobile phones and wearable devices, trading errors for reduced resource usage in displays is an interesting prospect. Organic light-emitting diode (OLED) displays present an interesting opportunity for trading errors for efficiency: Unlike traditional LCD displays, their power dissipation varies significantly as a function of the content displayed. It is therefore possible to purposefully introduce errors into displayed images to reduce the display's power consumption. The earliest examples of such approaches were originated by Dong et al. [2009a] and Dong et al. [2009b], who developed several of the first techniques for trading display power for visual fidelity in OLED displays. Recent research has developed more efficient techniques as well as new approaches that analyze and transform both the color and shape content of the rendered images to save power.

Figure 1.1 shows two variants of the same image, which differ in power dissipation by over 40% when displayed on a representative commercial OLED display panel. The image and corresponding shape and color transformations to reduce power dissipation on displays that behave similar to OLEDs were generated using the Crayon system [Stanley-Marbell et al.,

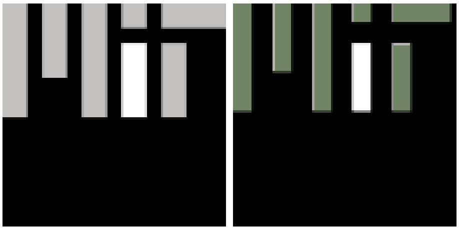


Figure 1.1: The image on the right dissipates more than 40% lower power than the one on the left when shown on OLED displays.

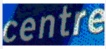
Tolerable Deviation	Image A	OCR Text	Transition Reduction	Image B	OCR Text	Transition Reduction
0%		"centre"	0%↓		"EXIT"	0%↓
10%		"centre"	66%↓		"LTXIT"	61%↓
20%		"centre"	73%↓		""	73%↓

Figure 1.2: Encoding values so that they dissipate less power when transmitted can lead to significant power reductions before they begin to affect optical character recognition algorithms. This is despite the fact that the encoded images look very different to the human eye.

2016]. The difference between the original image and the modified one is that areas of the gray regions in the latter are reduced by 25% and the colors have been modified slightly. Chapter 4 explores techniques for exploiting tolerance in outputs in more depth.

Not all systems have displays however. In the increasingly important domain of embedded sensor-driven systems, because the power dissipated in the digital logic components has continued to drop over the years, a significant fraction of the system's energy usage can result from the activation of sensors and the retrieval of data from them over their electrical communication interfaces.

Figure 1.2 shows how techniques that reduce the energy cost of transmissions by lossy encoding of the data can enable significant reductions in the energy required for transmitting the data. However, when the algorithms consuming the encoded data can tolerate the types of errors introduced by the encoding, they lead to minimal application-level errors, even though the perceived visual distortion may seem significant to the human eye.

Even though tolerating errors in the inputs and output communication of algorithms can be exposed in the syntax of programming languages [Stanley-Marbell and Marculescu, 2006], tolerating errors in the steps of algorithms is much more involved when compared to tolerating errors in the data algorithms process or errors in their outputs. Approaches to tackling this challenge range from annotating individual variables in algorithms as being ones that can tolerate errors (or not) [Sampson et al., 2011], annotating variables corresponding to the outputs of functions to specify which ones are permitted to incur errors [Misailovic et al., 2014], and using program analysis techniques to provide guarantees about the effects of errors as they propagate through the algorithm [Carbin et al., 2013].

An alternative to providing specifications of the tolerable input or output error is to specify how much error is acceptable in the *relation between inputs and outputs*. Figure 1.3 illustrates the formal specification of the computation task of partial sorting, along with an example of an input-output pair that conforms to this computation behavior. This problem of obtaining a partial sort occurs in real applications: Partial sorting accounts for over 24% of the execution time of one popular discrete-event simulator [Jongerius et al., 2014]. One exciting open area of research is to synthesize algorithms (or hardware) that conform to such computation specifications and that permit some degree of error in the relation between their inputs and outputs.

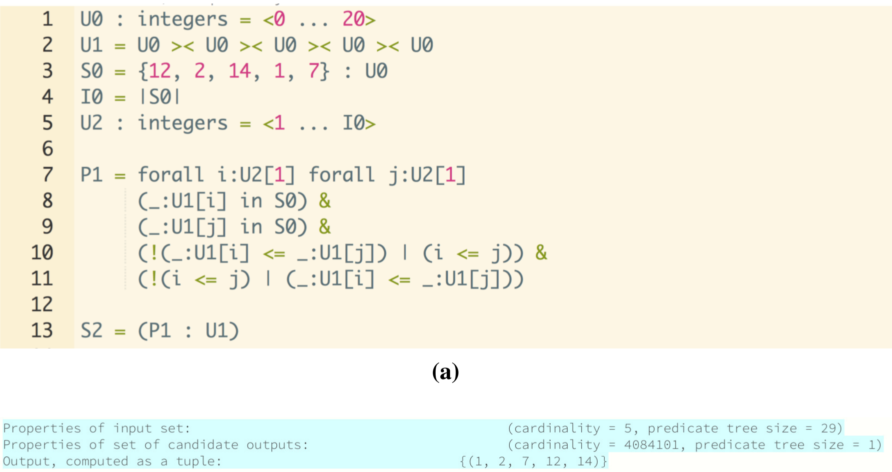
1.6 Fundamental Physical Limits, Energy, and Noise

Computing systems are designed to avoid errors at all levels[1], from copying data from registers to their transmission to other systems or different processors. They prevent errors for all applications and, as a result, require error-correcting coding techniques at all levels; this introduces overheads that are unnecessary in some cases.

Because the traditional mechanisms for improving the density and power consumption of computing systems are reaching fundamental physical limits [Bennett and Landauer, 1985], there has been an increased interest in recent years to develop techniques to explore trading correctness for some tangible improvement in a system, such as improved speed or improved energy efficiency. Figure 1.4(a) shows the reduction in the energy required per bit of

---

[1] Within the limit of economic and performance constraints



```
U0 : integers = <0 ... 20>
U1 = U0 >< U0 >< U0 >< U0 >< U0
S0 = {12, 2, 14, 1, 7} : U0
I0 = |S0|
U2 : integers = <1 ... I0>

P1 = forall i:U2[1] forall j:U2[1]
     (_:U1[i] in S0) &
     (_:U1[j] in S0) &
     (!(_:U1[i] <= _:U1[j]) | (i <= j)) &
     (!(i <= j) | (_:U1[i] <= _:U1[j]))

S2 = (P1 : U1)
```

(a)

```
Properties of input set:                                   (cardinality = 5, predicate tree size = 29)
Properties of set of candidate outputs:                    (cardinality = 4084101, predicate tree size = 1)
Output, computed as a tuple:                          {(1, 2, 7, 12, 14)}
```

(b)

Figure 1.3: Computation specification (a) for the computation that sorts a sequence of integers, expressed in the Sal low-level computation specification language Stanley-Marbell [2010] and its output (b).

information processing, over several decades. Because the diminishing opportunities to reduce power consumption of computing systems is largely due to power delivery and cooling limitations, these challenges are unlikely to be easily resolved in the near future[2], making the exploration of error-efficient systems ever more important in the future.

The underlying physical phenomenon permitting such energy versus correctness tradeoffs is well understood: For a device technology to be useful in constructing computational systems in which logic devices are linked together by non-ideal conductors, it must exhibit the property of *gain* (amplification) [Keyes, 1985]. This amplification requires an input energy source and the extent to which amplification occurs affects the likelihood of errors due to noise. If some amount of noise is tolerable, its presence can be traded for energy efficiency or performance.

---

[2] Supply voltage scaling across technology nodes has ceased, as Figure 1.4(b) shows

Full text available at: http://dx.doi.org/10.1561/1000000049

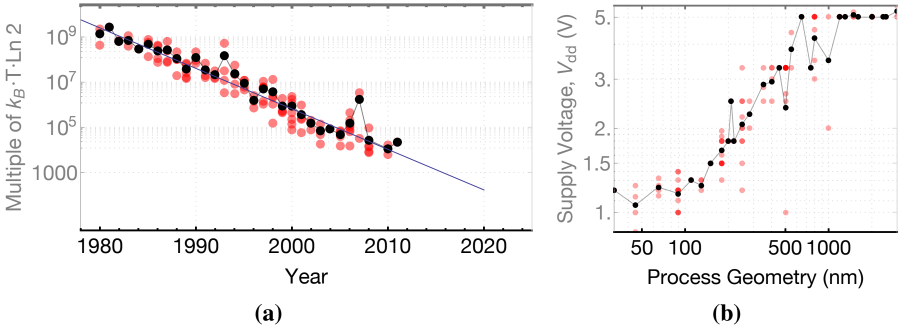


Figure 1.4: (a) The energy per logic transition in traditional circuit techniques is approaching the fundamental thermodynamic limit of $kT \ln 2$ Joules per bit of information (i.e., an ordinate value of 1 in (a) by ~2030). (b) One reason why energy usage in traditional CMOS logic is no longer scaling down, is that it is no longer feasible to decrease supply voltages. In both plots, the red points are published design data and the black points are the averages at a given abscissa [Stanley-Marbell et al., 2011].

1.7 Hardware and Software Systems That Exploit Errors

Techniques to improve system dependability have traditionally taken the approach of hiding (masking) faults in the hardware data-path and control-flow with spatial and temporal redundancy. Such an approach is desirable when there must be no change of system behavior in the presence of faults, except, perhaps, for a change in performance.

Applications of computing systems such as signal processing (in desktops and workstations), and sensor-driven applications (in embedded systems) often drive outputs that are only directly perceived by humans (e.g., the outputs of audio and video processing), or have inputs that are taken from noisy analog sources (e.g., in sensor network applications). In such applications, programs can often tolerate some amount of “going-wrong”. In particular, small deviations in values may be tolerable, and this is already exploited by some lossy compression algorithms for images (e.g., JPEG [Wallace, 1991]), audio, and video.

In many emerging applications of the recent decade, however, computing is moving from the sole purview of commercial business transaction management to more personal and pervasive applications such as embedded sensing and entertainment. In some of these new applications, such as embedded automotive control, there are still stringent requirements on correctness of machine state and computation. However, in many new applications, the need to maintain perfect error-free computation no longer exists.

As a result of these changes in applications of computing, a number of parallel research efforts have begun in recent years to explore ways to reduce the restrictions of perfect machine state. These efforts have ranged across:

- **Reducing the number of bits used to represent data values and datapaths**, either in storing those values or in synthesizing reduced-precision or reduced-accuracy logic in order to save energy (§ 1.7.1).
- **Explicitly exploiting human perception** to reduce resource usage (§ 1.7.2).
- **Circuits that perform logic operations on probability distributions of values**, rather than on unitary instance values (§ 1.7.3).
- **Hardware and software architectures for counteracting the effects of soft errors** (§ 1.7.4).
- **Architectures that assume applications can tolerate errors** in computation or timing, but have no contract with software on the permissible laxity (§ 1.7.5).
- **Programming languages and runtime systems that incorporate annotation of imprecision** in program state or operations, or exploit toleration of errors by applications (§ 1.7.6).
- **Investigation of application domains that can tolerate various forms of computation errors** or imprecision, in computation or state (§ 1.7.7).

These existing efforts have, however, mostly focused either only on adapting hardware independent of applications' requirements, or vice versa.

1.7.1 Reducing representation precision in values and datapaths

The earliest efforts at harnessing potential tolerance of imprecision, at the hardware level, involved reducing the number of bits used in both integer [Stephenson et al., 2000] and floating-point [Tong et al., 2000] representations. These efforts were not based on explicit information exposed by, or extracted from programs, but rather, on the assumption that signal-processing applications inherently deal with values obtained from noisy real-world measurements, and that real-number representations in computers are inherently approximations. Techniques that reduce the bit-level precision of arithmetic, and those that expose notions of incorrectness at the language level must contend with issues of numerical analysis. Kulisch [2008] provides a thorough background on the interaction between numerics of computation and the architectures that facilitate computing. In reducing the number of bits however, while the precision or dynamic range (or both) are reduced, computation proceeds deterministically and independent of the properties (value distributions) due to the applications it executes.

An alternative approach to simply providing reduced precision independent of application properties, is to synthesize logic circuits based on the distributions of values and the tolerance to reduced accuracy of specific applications, as investigated by Lingamneni et al. [2013]

1.7.2 Explicitly exploiting human perception

When the results of computation are consumed by the human aural or visual system, variations in accuracy, precision, or reliability may not always be perceptible. Such variations can be exploited directly in the generation of audio or display of results, for lower-energy, faster, or cheaper output devices (e.g., displays). For example, for displays, a few research efforts have investigated exploiting the variability in human sensitivity across the color spectrum. This phenomenon has been exploited to reduce power dissipation in OLED displays [Dong et al., 2009a, Zhao et al., 2013, Shin et al., 2011, Dong and Zhong, 2011, Harter et al., 2004, Li et al., 2014, Tan et al., 2013] as well as in those traditional LCDs that have coarse-grained controllable backlighting [Chuang et al., 2009]. Even when the results are consumed by non-human entities such as control systems, some amount of tolerance to imprecision, inaccuracy, and unreliability may still exist.

The interfaces for surfacing perceptual signals, such as displays and audio, contribute an increasing fraction of system energy usage in wearable and mobile systems. Because the phenomena underlying their operation (e.g., photon generation, mechanical displacement) are less amenable to improvements in transistor properties than computation is, their relative importance will likely grow in the future. Chapter 4 explores these concepts and implementations in more detail.

1.7.3 Probabilistic computation, probabilistic programming, and computing on probability distributions

In the traditional uses of probability in programming languages, the component which is probabilistic is the *behavior* of a computation, or a composition of concurrent processes [Stark and Smolka, 2000]. These approaches range from the *introduction of randomness into algorithms* [M. O. Rabin, 1976], the analysis of the behavior of randomized algorithms [Pnueli, 1983], and logics for probabilistic programs [Reif, 1980], to probabilistic parallel programs [Rao, 1994].

An alternative to the deterministic behavior of logic in hardware, whether of standard or of reduced precision, is to either employ randomness in the execution of hardware (to perform logic operations probabilistically [Palem, 2005, George et al., 2006]), or to consider the values of machine state due to executing applications, not as fixed instance values, but rather as probability distributions [Shanbhag et al., 2010, Vigoda et al., 2010, Vigoda, 2003]. The latter approach yields architectures that can be considered as forms of analog (as opposed to digital) computers.

1.7.4 Hardware and software architectures for counteracting the effects of soft errors

In the last decade, the observation that different applications (or classes thereof) may have differing tolerance to faults has been investigated [Wong and Horowitz, 2006], as have the possibility of applying different amounts of traditional software-based fault-tolerance techniques to different portions of an application [Reis et al., 2005a], as well as the influence of different hardware structures on the masking versus manifestation of faults as errors. These prior efforts, while recognizing the varying requirements for fault tolerance in applications and in hardware, have not attempted to tradeoff correctness for overheads.

There have been attempts to formalize the effects of soft-errors on the behavior of programs [Walker et al., 2006]. The model addressed in this recent work is one in which the goal is to attempt to nullify the effect of soft-errors (faults), by redundant computation.

The observation that different portions of programs or of hardware may require differing amounts of fault-protection has previously been applied to reduce the implementation overheads of hardware systems. This observation has been extended to phases of programs [Reis et al., 2005c] as well as to the design of error-resilient processor architectures and silicon implementations [Leem et al., 2010, Bau et al., 2007, Borodin et al., 2009, Rhod et al., 2007, Mehrara et al., 2007].

Several research efforts have explored adding architectural support for low-overhead detection and correction of the effects of soft errors, such as the *software anomaly treatment (SWAT)* system and its derivatives [Srinivasan et al., 2004], by determining the effect of soft errors in components of processor microarchitectures on application behavior [Li et al., 2005, 2008]. Purely-software-based approaches can also be used to trade correctness for speed or reduced resource usage. Two examples of such approaches include *loop perforation* [Sidiroglou-Douskos et al., 2011], and relaxing locking requirements in GPU kernels [Samadi et al., 2013].

1.7.5 "Better-than-worst-case" design and approximate hardware architectures

In probabilistic computing architectures (§ 1.7.3), non-determinism is used in a well-defined manner. This is in contrast to so-called better-than-worst-case hardware architectures [Austin et al., 2005, Wagner and Bertacco, 2007, Kahng et al., 2010], which aggressively bias system properties (e.g., power supply voltage) into regimes which may furnish significant energy savings, but increase the chance of failure. These architectures then use a variety of methods (e.g., shadow latches in the Razor system [Austin et al., 2004]) for ensuring infrequently-occurring erroneous state is not committed to final architectural state, or that critical data is not adversely affected (e.g., by reducing DRAM refresh rates, but only for non-critical data, in the Flicker system [Liu et al., 2009]).

Taking the idea of better-than-worst-case design further, are a class of architectures that argue that permitting occasional errors can reduce power consumption. When these platforms rely on applications and system software to deal appropriately with the errors that may result, we will refer to the platforms as *approximate hardware*. Examples of such approximate hardware range from processor architectures (or parts of processors such as ALUs) [Esmaeilzadeh et al., 2012b, Lingamneni et al., 2012], to complete accelerators [Esmaeilzadeh et al., 2012a, George et al., 2006, Sartori and Kumar, 2013], and to portions of the memory hierarchy [Sampson et al., 2013, Liu et al., 2009, Xu et al., 2004]. Techniques for approximation can be applied individually, or can be employed as part of a control system [Hoffmann, 2015] to ensure that a target energy reduction or accuracy constraint is satisfied.

As one example of these architectures, Truffle [Esmaeilzadeh et al., 2012a] defines an architecture in which individual operations (arithmetic instruction, memory accesses, etc.) may individually fail catastrophically with some probability, the rate at which they do so exhibiting a tradeoff with the amount of energy used. The manner in which this tradeoff is obtained is via the ability to set processor state and logic into a voltage-over-scaled (unreliable but energy-saving) state, with cycle-level granularity. Truffle relies on the programming language, compiler, and operating system to ensure that only individual instructions that can tolerate being in error are executed in the unreliable mode, and that unreliable state is appropriately quarantined from reliable state, with flow of data between reliable and unreliable computation obeying a well-defined set of constraints.

1.7.6 Programming languages and runtime systems

Program-level annotation provides an alternative to relegating to hardware all decisions about what machine state's accuracy can be traded for energy efficiency or performance. Language-level specification of tolerable imprecision has ranged from the specification of coarse regions of application code that can, in some broad sense, tolerate errors [Reis et al., 2005c, Walker et al., 2006, Baek and Chilimbi, 2010], memory locations that contain critical data [Pattabiraman et al., 2008], to the elision of loop iterations to tradeoff fidelity of computation results for energy efficiency or performance [Rinard et al., 2010, Rinard, 2006]. Program-level annotations of required precision such as the annotations provided by the EnerJ Java extension [Sampson et al., 2011] as well as tools to infer guarantees on correctness based on static program analysis [Carbin et al., 2013]. Detailed language-level facilities for specifying imprecision at the level of data types [Stanley-Marbell and

Full text available at: http://dx.doi.org/10.1561/1000000049

Marculescu, 2006] have also been developed, and extended to the declarative specification of the computation performed by a given subroutine, incorporating properties of imprecision [Stanley-Marbell, 2010].

1.7.7 Applications of "good-enough" computation in algorithms and software that are naturally resilient to errors

Given the aforementioned techniques for reduced precision arithmetic (§ 1.7.1), probabilistic computation (§ 1.7.3), hardware architectures and software techniques that take license with correctness (§ 1.7.4 and § 1.7.5), and language-level facilities for specifying how much incorrectness applications can tolerate (§ 1.7.6), a natural question is, which applications can best harness the possibilities afforded by these hardware and software innovations? Several proposals for potential application of such "good-enough" computation have been made in the research literature [Chakradhar and Raghunathan, 2010, Chippa et al., 2010, Breuer, 2010, 2005a, Meng et al., 2009, Chong and Ortega, 2007, Li and Yeung, 2007, Mohapatra et al., 2009, Breuer, 2005b, Salesin et al., 1989], however no consensus yet exists on a standard set of applications for evaluating proposed hardware and software techniques. Similarly, no commonly agreed-upon metrics exist for evaluating the degree to which behavior of benchmarks may deviate from correctness. Recent work has however taken an important step in this direction [Akturk et al., 2015].

One class of applications in which errors in computation are often tolerable is signal processing applications. This observation motivated some of the earliest work in trading correctness for performance and power from the work of Shanbhag on ANT [Hegde and Shanbhag, 1999, Shanbhag, 2002, Varatkar et al., 2009, Shanbhag et al., 2010], to silicon implementations of approximate signal processing from Amirtharajah and Chandrakasan [Amirtharajah and Chandrakasan, 2004] and Guo [Guo et al., 2006].

In addition to errors in values and control flow of computations, errors may occur in the timing of actions driven by computation, or in the latencies expected from computation. The term *imprecise computation* was coined in the nineties to denote real-time computing systems in which some deviation from temporal correctness was tolerable [Budin et al., 2004, Hull and Liu, 1993, Liu et al., 1991, Shih and Liu, 1995, Aydın et al., Liu et al., 1994, Kenny and Lin, 1991].

These efforts in computing systems and signal processing are of course predated by a large body of work in numerical analysis, uncertainty quantification (UQ) methods [Klir, 1994], tolerance graphs [Golumbic and Trenk, 2004], interval arithmetic [Hayes, 2003]), fuzzy logic and fuzzy set theory, approximation and randomized algorithms and, of course, existing work on in the broader field of fault-tolerant systems.

1.8 Outline of the Remainder of This Review

The present chapter provides a broad survey of the basic concepts explored in further detail throughout the review. It addresses the question of why error-efficient computing systems matter, and describes the context in which the material of the review is situated. It surveys the general state of the art in this area and positions the material of the review within it. Figure 1.5 summarizes the research referenced in this chapter. Chapter 2 (*Types of Errors and Randomization*) defines terminology, such as precision, accuracy, and reliability, which recur throughout the review and in any discussion of errors and of error efficiency. The definitions in Chapter 2 set the stage for the discussion of how errors affect efficiency in computing systems, in Chapter 3 (*Computation, Energy, and Noise*). Chapter 4 (*Tolerating Errors in Outputs*) addresses how many systems tolerate errors in their outputs. For example, any visual output that must be interpreted by a human may incur some amount of error before being perceptible. Chapter 5 (*Tolerating Errors in Inputs*) discusses the complementary problem of how many systems tolerate errors in their inputs. The review concludes in Chapter 6.

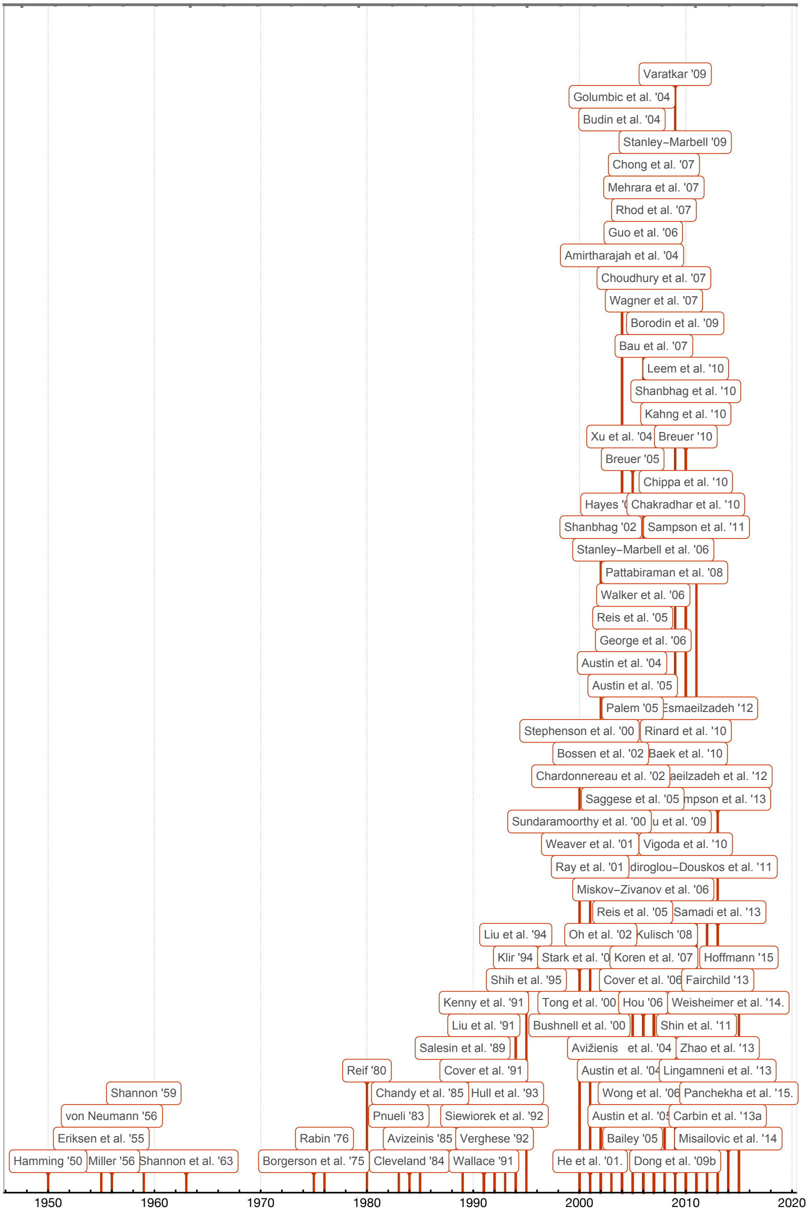


Figure 1.5: Timeline of referenced work in this chapter, listed by author.

References

Ismail Akturk, Karen Khatamifard, and Ulya R Karpuzcu. On quantification of accuracy loss in approximate computing. In *Workshop on Duplicating, Deconstructing and Debunking (WDDD)*, pages 15–, 2015.

Armin Alaghi and John P. Hayes. Survey of stochastic computing. *ACM Trans. Embed. Comput. Syst.*, 12(2s):92:1–92:19, May 2013.

Andres Albanese, Johannes Blömer, Jeff Edmonds, Michael Luby, and Madhu Sudan. Priority encoding transmission. *Information Theory, IEEE Transactions on*, 42(6):1737–1744, 1996.

R. Amirtharajah and A. P. Chandrakasan. A micropower programmable DSP using approximate signal processing based on distributed arithmetic. *IEEE Journal of Solid-State Circuits*, 39(2):337–347, 2004.

Analog Devices. *ADXL362 Micropower, 3-Axis,* ±2 g / ±4 g / ±8 g *Digital Output MEMS Accelerometer*, Data Sheet, 2014.

Bhojan Anand, Karthik Thirugnanam, Jeena Sebastian, Pravein G. Kannan, Akhihebbal L. Ananda, Mun Choon Chan, and Rajesh Krishna Balan. Adaptive display power management for mobile games. In *Proceedings of the 9th International Conference on Mobile Systems, Applications, and Services*, MobiSys '11, pages 57–70, New York, NY, USA, 2011. ACM.

Todd Austin, David Blaauw, Trevor Mudge, and Krisztián Flautner. Making typical silicon matter with razor. *Computer*, 37:57–65, March 2004.

Todd Austin, Valeria Bertacco, David Blaauw, and Trevor Mudge. Opportunities and challenges for better than worst-case design. In *Proceedings of the 2005 Asia and South Pacific Design Automation Conference*, ASP-DAC '05, pages 2–7, New York, NY, USA, 2005. ACM.

A. Avizeinis. The N-Version Approach to Fault-Tolerant Software. *IEEE Transactions of Software Engineering*, SE-11(12):1491–1501, December 1985.

A. Avižienis, J.C. Laprie, B. Randell, and C. Landwehr. Basic concepts and taxonomy of dependable and secure computing. *IEEE transactions on dependable and secure computing*, pages 11–33, 2004.

H. Aydın, R. Melhem, and D. Mossé. Incorporating Error Recovery into the Imprecise Computation Model. In *The Sixth International Conference on Real-Time Computing Systems and Applications (RTCSA '99)*.

Woongki Baek and Trishul M. Chilimbi. Green: a framework for supporting energy-conscious programming using controlled approximation. In *Proceedings of the 2010 ACM SIGPLAN conference on Programming language design and implementation*, PLDI '10, pages 198–209, New York, NY, USA, 2010. ACM.

Yu Bai and Mingjie Lin. Energy-efficient discrete signal processing with field programmable analog arrays (fpaas). In *Proceedings of the 2015 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, FPGA '15, pages 84–93, New York, NY, USA, 2015. ACM.

David H Bailey. High-precision floating-point arithmetic in scientific computation. *Computing in science & engineering*, 7(3):54–61, 2005.

J. Bau, R. Hankins, Q. Jacobson, S. Mitra, B. Saha, and A.R. Adl-Tabatabai. Error resilient system architecture (ERSA) for probabilistic applications. In *IEEE Workshop on Silicon Errors in Logic-System Effects, SELSE*, 2007.

Robert C. Baumann. Radiation-Induced Soft Errors in Advanced Semiconductor Technologies. 5(3):305–316, September 2005.

Charles H Bennett and Rolf Landauer. The fundamental physical limits of computation. *Scientific American*, 253(1):48–56, 1985.

Florian Benz, Andreas Hildebrandt, and Sebastian Hack. A dynamic program analysis to find floating-point accuracy problems. In *Proceedings of the 33rd ACM SIGPLAN Conference on Programming Language Design and Implementation*, PLDI '12, pages 453–462. ACM, 2012.

Toby Berger. Rate-distortion theory. *Encyclopedia of Telecommunications*, 1971.

B. R. Borgerson and R. F. Freitas. A reliability model for gracefully degrading and standby-sparing systems. *IEEE Transactions on Computers*, c-24:517–525, May 1975.

Shekhar Borkar, Tanay Karnik, and Vivek De. Design and reliability challenges in nanometer technologies. In *Proceedings of the 41st annual conference on Design automation*, pages 75–75. ACM Press, 2004.

D. Borodin, B. H. H. B. Juurlink, S. Hamdioui, and S. Vassiliadis. Instruction-Level Fault Tolerance Configurability. *Journal of Signal Processing Systems*, 57(1): 89–105, 2009.

Bosch Sensortec. *BMX055 Small, Versatile 9-axis Sensor Module*, Data Sheet, November 2014.

Douglas C. Bossen, Joel M. Tendler, and Kevin Reick. Power4 system design for high reliability. *IEEE Micro*, 22(2):16–24, 2002.

Melvin Breuer. Multi-media applications and imprecise computation. In *Proceedings of the 8th Euromicro Conference on Digital System Design*, pages 2–7, Washington, DC, USA, 2005a. IEEE Computer Society.

Melvin Breuer. Hardware that produces bounded rather than exact results. In *Proceedings of the 47th Design Automation Conference*, DAC ’10, pages 871–876, New York, NY, USA, 2010. ACM.

Melvin A. Breuer. Multi-media applications and imprecise computation. In *Digital Systems Design, Euromicro Symposium on*, pages 2–7, Los Alamitos, CA, USA, 2005b. IEEE Computer Society.

L. Budin, D. Jakobović, and M. Golub. Genetic algorithms in real-time imprecise computing. *Journal of Computing and Information Technology*, 8(3):249, 2004.

M. L. Bushnell and V. D. Agrawal. *Essentials of electronic testing for digital, memory, and mixed-signal VLSI circuits*. Springer Netherlands, 2000.

George Candea, Shinichi Kawamoto, Yuichi Fujiki, Greg Friedman, and Armando Fox. Microreboot — a technique for cheap recovery. In *Proceedings of the 6th Conference on Symposium on Operating Systems Design & Implementation - Volume 6*, OSDI’04, pages 31–44. USENIX Association, 2004.

Michael Carbin, Sasa Misailovic, and Martin C. Rinard. Verifying quantitative reliability for programs that execute on unreliable hardware. In *Proceedings of the 2013 ACM SIGPLAN International Conference on Object Oriented Programming Systems Languages & Applications*, OOPSLA ’13, pages 33–52, New York, NY, USA, 2013. ACM.

Srimat T. Chakradhar and Anand Raghunathan. Best-effort computing: re-thinking parallel software and hardware. In *Proceedings of the 47th Design Automation Conference*, DAC ’10, pages 865–870, New York, NY, USA, 2010. ACM.

K. Mani Chandy and Leslie Lamport. Distributed snapshots: determining global states of distributed systems. *ACM Trans. Comput. Syst.*, 3(1):63–75, 1985.

Naehyuck Chang, Inseok Choi, and Hojun Shim. Dls: dynamic backlight luminance scaling of liquid crystal display. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 12(8):837–846, Aug 2004.

Chardonnereau, Damien and Keulen, Raijmond and Nicolaidis, Michael and Dupont, Eric and Torki, Kholdoun and Faure, Fabien and Velazco, Raoul. 32-Bit RISC Processor Implementing Transient Fault-Tolerant Mechanisms and its Radiation Test Campaign Results. In *Single-Event Effects Symp.*, NASA, April 2002.

Xiang Chen, Yiran Chen, Zhan Ma, and Felix C. A. Fernandes. How is energy consumed in smartphone display applications? In *Proceedings of the 14th Workshop on Mobile Computing Systems and Applications*, HotMobile '13, pages 3:1–3:6, New York, NY, USA, 2013. ACM.

Xiang Chen, Kent W. Nixon, Hucheng Zhou, Yunxin Liu, and Yiran Chen. Fingershadow: An oled power optimization based on smartphone touch interactions. In *6th Workshop on Power-Aware Computing and Systems (HotPower 14)*, Broomfield, CO, 2014. USENIX Association.

Wei-Chung Cheng, Yu Hou, and Massoud Pedram. Power minimization in a backlit tft-lcd display by concurrent brightness and contrast scaling. In *Proceedings of the Conference on Design, Automation and Test in Europe - Volume 1*, DATE '04, pages 10252–, Washington, DC, USA, 2004. IEEE Computer Society.

Wei-Chung Cheng, Chih-Fu Hsu, and Chain-Fu Chao. Temporal vision-guided energy minimization for portable displays. In *Low Power Electronics and Design, 2006. ISLPED'06. Proceedings of the 2006 International Symposium on*, pages 89–94, Oct 2006.

Vinay K. Chippa, Debabrata Mohapatra, Anand Raghunathan, Kaushik Roy, and Srimat T. Chakradhar. Scalable effort hardware design: exploiting algorithmic resilience for energy efficiency. In *Proceedings of the 47th Design Automation Conference*, DAC '10, pages 555–560, New York, NY, USA, 2010. ACM.

Ching-Te Chiu, Wen-Chih Huang, Chih-Hsing Lin, Wei-Chih Lai, and Ying-Fang Tsao. Embedded transition inversion coding with low switching activity for serial links. *IEEE TVLSI*, 21(10):1797–1810, October 2013.

Inseok Choi, Hojun Shim, and Naehyuck Chang. Low-power color tft lcd display for hand-held embedded systems. In *Proceedings of the 2002 International Symposium on Low Power Electronics and Design*, ISLPED '02, pages 112–117, New York, NY, USA, 2002. ACM.

I. S. Chong and A. Ortega. Power Efficient Motion Estimation using Multiple Imprecise Metric Computations. In *2007 IEEE International Conference on Multimedia and Expo*, pages 2046–2049, 2007.

Johnson Chuang, Daniel Weiskopf, and Torsten Möller. Energy aware color sets. *Computer Graphics Forum*, 28(2):203–211, 2009.

Cristian Constantinescu. Neutron ser characterization of microprocessors. In *DSN '05: Proceedings of the 2005 International Conference on Dependable Systems and Networks (DSN'05)*, pages 754–759, Washington, DC, USA, 2005. IEEE Computer Society.

Thomas M. Cover and Joy A. Thomas. *Elements of information theory*. Wiley series in telecommunications. John Wiley & Sons, New York-Chichester-Brisbane-Toronto-Singapore, 1991.

Pierre-Evariste Dagand, Andrew Baumann, and Timothy Roscoe. Filet-o-fish: Practical and dependable domain-specific languages for os development. In *Proceedings of the Fifth Workshop on Programming Languages and Operating Systems*, PLOS '09, pages 5:1–5:5. ACM, 2009.

Robert H Dennard, Fritz H Gaensslen, V Leo Rideout, Ernest Bassous, and Andre R LeBlanc. Design of ion-implanted mosfet's with very small physical dimensions. *IEEE Journal of Solid-State Circuits*, 9(5):256–268, 1974.

Mian Dong and Lin Zhong. Chameleon: A color-adaptive web browser for mobile oled displays. In *Proceedings of the 9th International Conference on Mobile Systems, Applications, and Services*, MobiSys '11, pages 85–98, New York, NY, USA, 2011. ACM.

Mian Dong, Yung-Seok Kevin Choi, and Lin Zhong. Power-saving color transformation of mobile graphical user interfaces on oled-based displays. In *Proceedings of the 2009 ACM/IEEE International Symposium on Low Power Electronics and Design*, ISLPED '09, pages 339–342, New York, NY, USA, 2009a. ACM.

Mian Dong, Yung-Seok Kevin Choi, and Lin Zhong. Power modeling of graphical user interfaces on oled displays. In *Proceedings of the 46th Annual Design Automation Conference*, DAC '09, pages 652–657, New York, NY, USA, 2009b. ACM.

Hadi Esmaeilzadeh, Adrian Sampson, Luis Ceze, and Doug Burger. Architecture support for disciplined approximate programming. In *Proceedings of the Seventeenth International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS XVII, pages 301–312, New York, NY, USA, 2012a. ACM.

Hadi Esmaeilzadeh, Adrian Sampson, Luis Ceze, and Doug Burger. Neural acceleration for general-purpose approximate programs. In *Proceedings of the 2012 45th Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO-45, pages 449–460, Washington, DC, USA, 2012b. IEEE Computer Society. .

Freescale Semiconductor. *Kinetis KL03 32 KB Flash 48 MHz Cortex-M0+ Based Microcontroller*, Data Sheet, August 2014a.

Freescale Semiconductor. *MMA8451Q 3-Axis, 14-bit/8-bit Digital Accelerometer*, Data Sheet, November 2014b.

J. George, B. Marr, B. E. S. Akgul, and K. V. Palem. Probabilistic arithmetic and energy efficient embedded signal processing. In *Proceedings of the 2006 International Conference on Compilers, Architecture and Synthesis for Embedded Systems*, CASES '06, pages 158–168, New York, NY, USA, 2006. ACM.

Daniel T Gillespie. A general method for numerically simulating the stochastic time evolution of coupled chemical reactions. *Journal of computational physics*, 22(4): 403–434, 1976.

Daniel T Gillespie. Exact stochastic simulation of coupled chemical reactions. *The journal of physical chemistry*, 81(25):2340–2361, 1977.

M. C. Golumbic and A. N. Trenk. *Tolerance graphs*. Cambridge Univ Pr, 2004.

Noah D. Goodman. The principles and practice of probabilistic programming. In *Proceedings of the 40th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, POPL '13, pages 399–402, New York, NY, USA, 2013. ACM.

Noah D. Goodman, Vikash Mansinghka, Daniel M Roy, Keith Bonawitz, and Joshua B Tenenbaum. Church: a language for generative models. *arXiv preprint arXiv:1206.3255*, 2012.

Andrew D. Gordon, Thore Graepel, Nicolas Rolland, Claudio Russo, Johannes Borgstrom, and John Guiver. Tabular: A schema-driven probabilistic programming language. In *Proceedings of the 41st ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, POPL '14, pages 321–334, New York, NY, USA, 2014a. ACM. .

Andrew D. Gordon, Thomas A. Henzinger, Aditya V. Nori, and Sriram K. Rajamani. Probabilistic programming. In *Proceedings of the on Future of Software Engineering*, FOSE 2014, pages 167–181, New York, NY, USA, 2014b. ACM. .

L. Guo, M. Scott, and R. Amirtharajah. An energy scalable computational array for sensor signal processing. In *IEEE Custom Integrated Circuits Conference, 2006. CICC'06*, pages 317–320, 2006.

Richard W Hamming. Error detecting and error correcting codes. *Bell System technical journal*, 29(2):147–160, 1950.

Tim Harter, Sander Vroegindeweij, Erik Geelhoed, Meera Manahan, and Parthasarathy Ranganathan. Energy-aware user interfaces: An evaluation of user acceptance. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '04, pages 199–206, New York, NY, USA, 2004. ACM.

B. Hayes. A lucid interval. *American Scientist*, 91(6):484–488, 2003.

Yun He and Chris H. Q. Ding. Using accurate arithmetics to improve numerical reproducibility and stability in parallel applications. *J. Supercomput.*, 18(3):259–277, March 2001.

Rajamohana Hegde and Naresh R. Shanbhag. Energy-efficient signal processing via algorithmic noise-tolerance. In *Proceedings of the 1999 International Symposium on Low Power Electronics and Design*, ISLPED '99, pages 30–35, New York, NY, USA, 1999. ACM. .

William Heidergott. SEU Tolerant Device, Circuit and Processor Design. In *DAC '05: Proceedings of the 42nd annual conference on Design automation*, pages 5–10, New York, NY, USA, 2005. ACM Press.

C. A. R. Hoare. Algorithm 64: Quicksort. *Commun. ACM*, 4(7):321–, July 1961.

Henry Hoffmann. Jouleguard: Energy guarantees for approximate applications. In *Proceedings of the 25th Symposium on Operating Systems Principles*, SOSP '15, pages 198–214, New York, NY, USA, 2015. ACM.

Robert W. Horst, Richard L. Harris, and Robert L. Jardine. Multiple instruction issue in the NonStop cyclone processor. In *ISCA '90: Proceedings of the 17th annual international symposium on Computer Architecture*, pages 216–226, New York, NY, USA, 1990. ACM Press.

D. Hull and J. Liu. ICS: A system for imprecise computations. In *Proc. AIAA Computing in Aerospace*, volume 9, 1993.

Ali Iranli and Massoud Pedram. Dtm: Dynamic tone mapping for backlight scaling. In *Proceedings of the 42nd Annual Design Automation Conference*, DAC '05, pages 612–617, New York, NY, USA, 2005. ACM.

Francois Jacquet. Design of SRAMs in Scaled CMOS Technologies. Seminar, Center for Silicon System Implementation (CSSI), Carnegie Mellon University, 2006.

R. Jongerius, P. Stanley-Marbell, and H. Corporaal. Quantifying the Common Computational Problems in Contemporary Applications (Extended version), 2014. IBM Research Report RZ3885.

Asim Kadav, Matthew J. Renzelmann, and Michael M. Swift. Tolerating hardware device failures in software. In *Proceedings of the ACM SIGOPS 22Nd Symposium on Operating Systems Principles*, SOSP '09, pages 59–72. ACM, 2009.

Andrew B. Kahng, Seokhyeong Kang, Rakesh Kumar, and John Sartori. Recovery-driven design: a power minimization methodology for error-tolerant processor modules. In *DAC '10: Proceedings of the 47th Design Automation Conference*, pages 825–830, New York, NY, USA, 2010. ACM.

Kevin B. Kenny and Kwei-Jay Lin. Building flexible real-time systems using the flex language. *Computer*, 24(5):70–78, 1991.

Robert W Keyes. What makes a good computer device? *Science*, 230(4722):138–144, 1985.

G.J. Klir. The many faces of uncertainty. *Machine Intelligence and Pattern Recognition*, 17:3–3, 1994.

I. Koren and C. M. Krishna. *Fault Tolerant Systems*. Morgan Kaufmann Publishers Inc. San Francisco, CA, USA, 2007.

U. Kulisch. *Computer arithmetic and validity: theory, implementation, and applications*. de Gruyter, 2008.

Jennifer R. Kwapisz, Gary M. Weiss, and Samuel A. Moore. Activity recognition using cell phone accelerometers. *SIGKDD Explor. Newsl.*, 12(2):74–82, March 2011.

Edwin H Land. Color vision and the natural image. part i. *Proceedings of the National Academy of Sciences*, 45(1):115–129, 1959a.

Edwin H Land. Color vision and the natural image part ii. *Proceedings of the National Academy of Sciences*, 45(4):636–644, 1959b.

Edwin H Land. Experiments in color vision. *Scientific American*, 200(5):84, 1959c.

Edwin H Land. The retinex theory of color vision. *Scientific American*, 237(6):108, 1977.

Edwin H Land. Recent advances in retinex theory and some implications for cortical computations: color vision and the natural image. *Proceedings of the National Academy of Sciences*, 80(16):5163–5169, 1983.

Edwin H Land. An alternative technique for the computation of the designator in the retinex theory of color vision. *Proceedings of the national academy of sciences*, 83(10):3078–3080, 1986.

L. Leem, H. Cho, J. Bau, Q.A. Jacobson, and S. Mitra. ERSA: Error Resilient System Architecture for Probabilistic Applications. In *Proc. Design Automation and Test in Europe*, 2010.

Ding Li, Angelica Huyen Tran, and William G. J. Halfond. Making web applications more energy efficient for oled smartphones. In *Proceedings of the 36th International Conference on Software Engineering*, ICSE 2014, pages 527–538, New York, NY, USA, 2014. ACM.

Ding Li, Angelica Huyen Tran, and William G. J. Halfond. Nyx: A display energy optimizer for mobile web apps. In *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering*, ESEC/FSE 2015, pages 958–961, New York, NY, USA, 2015. ACM.

Man-Lap Li, Pradeep Ramachandran, Swarup Kumar Sahoo, Sarita V Adve, Vikram S Adve, and Yuanyuan Zhou. Understanding the propagation of hard errors to software and implications for resilient system design. In *ACM SIGARCH Computer Architecture News*, volume 36, pages 265–276. ACM, 2008.

Xiaodong Li, Sarita V Adve, Pradip Bose, Jude Rivers, et al. Softarch: an architecture-level tool for modeling and analyzing soft errors. In *International Conference on Dependable Systems and Networks (DSN 2005)*, pages 496–505. IEEE, 2005.

Xuanhua Li and Donald Yeung. Application-level correctness and its impact on fault tolerance. In *Proceedings of the 13th International Symposium on High Performance Computer Architecture*, pages 181–192, 2007.

Michael D. Linderman, Matthew Ho, David L. Dill, Teresa H. Meng, and Garry P. Nolan. Towards program optimization through automated analysis of numerical precision. In *Proceedings of the 8th Annual IEEE/ACM International Symposium on Code Generation and Optimization*, CGO '10, pages 230–237. ACM, 2010.

Avinash Lingamneni, Kirthi Krishna Muntimadugu, Christian Enz, Richard M. Karp, Krishna V. Palem, and Christian Piguet. Algorithmic methodologies for ultra-efficient inexact architectures for sustaining technology scaling. In *Proceedings of the 9th Conference on Computing Frontiers*, CF '12, pages 3–12, New York, NY, USA, 2012. ACM. .

Avinash Lingamneni, Christian Enz, Krishna Palem, and Christian Piguet. Synthesizing parsimonious inexact circuits through probabilistic design techniques. *ACM Transactions on Embedded Computing Systems (TECS)*, 12(2s):93, 2013.

J. W. S. Liu, K. J. Lin, W. K. Shih, A. C. Yu, J. Y. Chung, and W. Zhao. Algorithms for scheduling imprecise computations. *Computer*, 24(5):58–68, 1991.

J. W. S. Liu, Wei-Kuan Shih, Kwei-Jay Lin, R. Bettati, and Jen-Yao Chung. Imprecise Computations. *Proceedings of the IEEE*, 82(1):83–94, January 1994.

S. Liu, K. Pattabiraman, T. Moscibroda, and B. G. Zorn. Flicker: Saving Refresh-Power in Mobile Devices through Critical Data Partitioning. Technical Report MSR-TR-2009-138, Microsoft Research, October 2009.

M. O. Rabin. Probabilistic Algorithms. In *Algorithms and Complexity*, pages 21 – 40, New York, NY, USA, 1976. Academic Press.

S. Mandal and R. Sarpeshkar. Circuit models of stochastic genetic networks. In *Biomedical Circuits and Systems Conference, 2009. BioCAS 2009. IEEE*, pages 109–112, Nov 2009.

H. Bo Marr and Jennifer Hasler. Compiling probabilistic, bio-inspired circuits on a field programmable analog array. *Frontiers in Neuroscience*, 8(86), 2014.

T. C. May and M. H. Woods. Alpha-particle-induced soft errors in dynamic memories. *IEEE Trans. Elect. Dev.*, 26:2, 1979.

M. Mehrara, M. Attariyan, S. Shyam, K. Constantinides, V. Bertacco, and T. Austin. Low-cost protection for SER upsets and silicon defects. In *Design, Automation & Test in Europe Conference & Exhibition, 2007. DATE'07*, pages 1–6, 2007.

Jiayuan Meng, Srimat Chakradhar, and Anand Raghunathan. Best-effort parallel execution framework for recognition and mining applications. In *Parallel and Distributed Processing Symposium, International*, pages 1–12, Los Alamitos, CA, USA, 2009. IEEE Computer Society.

Fabrice Mérillon, Laurent Réveillère, Charles Consel, Renaud Marlet, and Gilles Muller. Devil: An idl for hardware programming. In *Proceedings of the 4th Conference on Symposium on Operating System Design & Implementation - Volume 4*, OSDI'00, pages 2–2. USENIX Association, 2000.

Sasa Misailovic, Michael Carbin, Sara Achour, Zichao Qi, and Martin C Rinard. Chisel: reliability-and accuracy-aware optimization of approximate computational kernels. In *Proceedings of the 2014 ACM International Conference on Object Oriented Programming Systems Languages & Applications*, pages 309–328. ACM, 2014.

Natasa Miskov-Zivanov and Diana Marculescu. Mars-c: modeling and reduction of soft errors in combinational circuits. In *DAC '06: Proceedings of the 43rd annual conference on Design automation*, pages 767–772, New York, NY, USA, 2006. ACM Press.

Radhika Mittal, Aman Kansal, and Ranveer Chandra. Empowering developers to estimate app energy consumption. In *Proceedings of the 18th Annual International Conference on Mobile Computing and Networking*, Mobicom '12, pages 317–328, New York, NY, USA, 2012. ACM.

Michael Mitzenmacher and Eli Upfal. *Probability and computing: Randomized algorithms and probabilistic analysis*. Cambridge University Press, 2005.

Debabrata Mohapatra, Georgios Karakonstantis, and Kaushik Roy. Significance driven computation: a voltage-scalable, variation-aware, quality-tuning motion estimator. In *ISLPED '09: Proceedings of the 14th ACM/IEEE international symposium on Low power electronics and design*, pages 195–200, New York, NY, USA, 2009. ACM.

Rajeev Motwani and Prabhakar Raghavan. *Randomized algorithms*. Chapman & Hall/CRC, 2010.

Michael Nicolaidis and Damien Chardonnereau. Soft-Error Testing: Key Points. *Computer*, 38(2):44 (sidebar), 2005.

NXP Semiconductors. UM10204, *I2C-bus specification and user manual*, April 2014.

Nahmsuk Oh, Subhasish Mitra, and Edward J. McCluskey. ED4I: Error detection by diverse data and duplicated instructions. *IEEE Trans. Computers*, 51(2):180–199, 2002a.

Nahmsuk Oh, Philip Shirvani, and Edward J. McCluskey. Error detection by duplicated instructions in super-scalar processors. *IEEE Transactions on Reliability*, 51(1):63–75, March 2002b.

Krishna V. Palem. Energy aware computing through probabilistic switching: A study of limits. *IEEE Trans. Comput.*, 54:1123–1137, September 2005.

Pavel Panchekha, Alex Sanchez-Stern, James R. Wilcox, and Zachary Tatlock. Automatically improving accuracy for floating point expressions. In *Proceedings of the 36th ACM SIGPLAN Conference on Programming Language Design and Implementation*, PLDI 2015, pages 1–11, New York, NY, USA, 2015. ACM.

Sudeep Pasricha, Manev Luthra, Shivajit Mohapatra, Nikil Dutt, and Nalini Venkatasubramanian. Dynamic backlight adaptation for low-power handheld devices. *IEEE design & test of computers*, (5):398–405, 2004.

Karthik Pattabiraman, Vinod Grover, and Benjamin G. Zorn. Samurai: protecting critical data in unsafe languages. In *Proceedings of the 3rd ACM SIGOPS/EuroSys European Conference on Computer Systems 2008*, Eurosys '08, pages 219–232, New York, NY, USA, 2008. ACM.

Amir Pnueli. On the extremely fair treatment of probabilistic algorithms. In *STOC '83: Proceedings of the fifteenth annual ACM symposium on Theory of computing*, pages 278–290, New York, NY, USA, 1983. ACM Press.

Parthasarathy Ranganathan, Erik Geelhoed, Meera Manahan, and Ken Nicholas. Energy-aware user interfaces and energy-adaptive displays. *Computer*, 39(3):31–38, March 2006.

Josyula R. Rao. Reasoning about probabilistic parallel programs. *ACM Trans. Program. Lang. Syst.*, 16(3):798–842, 1994.

Joydeep Ray, James C. Hoe, and Babak Falsafi. Dual use of superscalar datapath for transient-fault detection and recovery. In *Proceedings of the 34th annual ACM/IEEE international symposium on Microarchitecture*, pages 214–224. IEEE Computer Society, 2001.

John H. Reif. Logics for probabilistic programming (extended abstract). In *STOC '80: Proceedings of the twelfth annual ACM symposium on Theory of computing*, pages 8–13, New York, NY, USA, 1980. ACM Press.

George A. Reis, Jonathan Chang, Neil Vachharajani, Ram Rangan, and David I. August. Swift: Software implemented fault tolerance. In *CGO '05: Proceedings of the international symposium on Code generation and optimization*, pages 243–254, Washington, DC, USA, 2005a. IEEE Computer Society.

George A. Reis, Jonathan Chang, Neil Vachharajani, Ram Rangan, David I. August, and Shubhendu S. Mukherjee. Design and evaluation of hybrid fault-detection systems. In *ISCA '05: Proceedings of the 32nd Annual International Symposium on Computer Architecture*, pages 148–159, Washington, DC, USA, 2005b. IEEE Computer Society.

George A. Reis, Jonathan Chang, Neil Vachharajani, Ram Rangan, David I. August, and Shubhendu S. Mukherjee. Software-controlled fault tolerance. *ACM Trans. Archit. Code Optim.*, 2:366–396, December 2005c.

EL Rhod, CA Lisbôa, and L. Carro. A low-SER efficient core processor architecture for future technologies. In *Design, Automation & Test in Europe Conference & Exhibition, 2007. DATE'07*, pages 1–6, 2007.

S. V. Rice, H. Bunke, and T. A. Nartker. Classes of cost functions for string edit distance. *Algorithmica*, 18(2):271–280, 1997.

Martin Rinard. Probabilistic accuracy bounds for fault-tolerant computations that discard tasks. In *Proceedings of the 20th annual international conference on Supercomputing*, ICS '06, pages 324–334, New York, NY, USA, 2006. ACM.

Martin Rinard, Henry Hoffmann, Sasa Misailovic, and Stelios Sidiroglou. Patterns and statistical analysis for understanding reduced resource computing. In *Proceedings of the ACM international conference on Object oriented programming systems languages and applications*, OOPSLA '10, pages 806–821, New York, NY, USA, 2010. ACM.

Leonid Ryzhyk, Adam Walker, John Keys, Alexander Legg, Arun Raghunath, Michael Stumm, and Mona Vij. User-guided device driver synthesis. In *Proceedings of the 11th USENIX Conference on Operating Systems Design and Implementation*, OSDI'14, pages 661–676. USENIX Association, 2014.

Giacinto Paolo Saggese and Anoop Vetteth. Microprocessor sensitivity to failures: Control vs execution and combinational vs sequential logic. In *DSN '05: Proceedings of the 2005 International Conference on Dependable Systems and Networks (DSN'05)*, pages 760–769, Washington, DC, USA, 2005. IEEE Computer Society.

D. Salesin, J Stolfi, and L. Guibas. Epsilon geometry: building robust algorithms from imprecise computations. In *SCG '89: Proceedings of the fifth annual symposium on Computational geometry*, pages 208–217, New York, NY, USA, 1989. ACM.

Mehrzad Samadi, Janghaeng Lee, D. Anoushe Jamshidi, Amir Hormati, and Scott Mahlke. Sage: Self-tuning approximation for graphics engines. In *Proceedings of the 46th Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO-46, pages 13–24, New York, NY, USA, 2013. ACM.

Adrian Sampson, Werner Dietl, Emily Fortuna, Danushen Gnanapragasam, Luis Ceze, and Dan Grossman. Enerj: Approximate data types for safe and general low-power computation. In *Proceedings of the 32Nd ACM SIGPLAN Conference on Programming Language Design and Implementation*, PLDI '11, pages 164–174, New York, NY, USA, 2011. ACM.

Adrian Sampson, Jacob Nelson, Karin Strauss, and Luis Ceze. Approximate storage in solid-state memories. In *Proceedings of the 46th Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO-46, pages 25–36, New York, NY, USA, 2013. ACM.

John Sartori and Rakesh Kumar. Exploiting timing error resilience in processor architecture. *ACM Trans. Embed. Comput. Syst.*, 12(2s):89:1–89:25, May 2013.

Matthew Schuchhardt, Susmit Jha, Raid Ayoub, Michael Kishinevsky, and Gokhan Memik. Optimizing mobile display brightness by leveraging human visual perception. In *Proceedings of the 2015 International Conference on Compilers, Architecture and Synthesis for Embedded Systems*, CASES '15, pages 11–20, Piscataway, NJ, USA, 2015. IEEE Press.

Naresh R. Shanbhag. Reliable and energy-efficient digital signal processing. In *Proceedings of the 39th Annual Design Automation Conference*, DAC '02, pages 830–835, New York, NY, USA, 2002. ACM.

Naresh R. Shanbhag, Rami A. Abdallah, Rakesh Kumar, and Douglas L. Jones. Stochastic Computation. In *Proceedings of the 47th Design Automation Conference*, pages 859–864. ACM, 2010.

Claude E. Shannon. Coding theorems for a discrete source with a fidelity criterion. *IRE National Convention Record*, 7(4):142–163, 1959.

Claude E. Shannon and Warren Weaver. *The Mathematical Theory of Communication*. University of Illinois Press, Urbana, Illinois, 1963.

W. K. Shih and J. W. S. Liu. Algorithms for scheduling imprecise computations with timing constraints to minimize maximum error. *IEEE Transactions on Computers*, 44(3):466–471, 1995.

Donghwa Shin, Younghyun Kim, Naehyuck Chang, and Massoud Pedram. Dynamic voltage scaling of oled displays. In *Proceedings of the 48th Design Automation Conference*, DAC '11, pages 53–58, New York, NY, USA, 2011. ACM.

Stelios Sidiroglou-Douskos, Sasa Misailovic, Henry Hoffmann, and Martin Rinard. Managing performance vs. accuracy trade-offs with loop perforation. In *Proceedings of the 19th ACM SIGSOFT Symposium and the 13th European Conference on Foundations of Software Engineering*, ESEC/FSE '11, pages 124–134, New York, NY, USA, 2011. ACM.

D. P. Siewiorek and R. S. Swarz. *Reliable Computer Systems, Design and Evaluation*. Digital Press, 2nd edition, 1992.

T. J. Slegel, R. M. Averill III, M. A. Check, B. C. Giamei, B. W. Krumm, C. A. Krygowski, W. H. Li, J. S. Liptay, J. D. MacDougall, T. J. McPherson, J. A. Navarro, E. M. Schwarz, K. Shum, and C. F. Webb. IBM's S/390 G5 Microprocessor design. *IEEE Micro*, 19:12–23, March 1999.

R. Smith. An Overview of the Tesseract OCR Engine. *ICDAR*, 7(1):629–633, 2007.

Jayanth Srinivasan, Sarita V Adve, Pradip Bose, and Jude A Rivers. The case for lifetime reliability-aware microprocessors. In *ACM SIGARCH Computer Architecture News*, volume 32, page 276. IEEE Computer Society, 2004.

ST Microelectronics. *L3G4200D MEMS Motion Sensor: Ultra-stable Three-axis Digital Output Gyroscope*, Data Sheet, December 2010.

ST Microelectronics. *LPS25H MEMS Pressure Sensor: 260–1260 hPa Absolute Digital Output Barometer*, Data Sheet, January 2014.

Phillip Stanley-Marbell. Sal/svm: an assembly language and virtual machine for computing with non-enumerated sets. In *Virtual Machines and Intermediate Languages*, VMIL '10, pages 1:1–1:10, New York, NY, USA, 2010. ACM.

Phillip Stanley-Marbell and Diana Marculescu. A Programming Model and Language Implementation for Concurrent Failure-Prone Hardware. In *Proceedings of the 2nd Workshop on Programming Models for Ubiquitous Parallelism, PMUP '06*, September 2006.

Phillip Stanley-Marbell and Martin Rinard. Lax: Driver interfaces for approximate sensor device access. In *15th Workshop on Hot Topics in Operating Systems (HotOS XV)*, Kartause Ittingen, Switzerland, May 2015a. USENIX Association.

Phillip Stanley-Marbell and Martin Rinard. Value-deviation-bounded serial data encoding for energy-efficient approximate communication. Technical Report MIT-CSAIL-TR-2015-022, MIT Computer Science and Artificial Intelligence Laboratory (CSAIL), June 2015b.

Phillip Stanley-Marbell, Victoria Caparros, and Ronald Luijten. Pinned to the walls: Impact of packaging and application properties on the memory and power walls. In *Proceedings of the 17th IEEE/ACM International Symposium on Low-power Electronics and Design*, ISLPED '11, pages 51–56, 2011.

Phillip Stanley-Marbell, Virginia Estellers, and Martin Rinard. Crayon: Saving power through shape and color approximation on next-generation displays. In *Proceedings of the Eleventh European Conference on Computer Systems*, EuroSys '16, pages 11:1–11:17, New York, NY, USA, 2016. ACM. .

Eugene W. Stark and Scott A. Smolka. A Complete Axiom System for Finite-State Probabilistic Processes. *Proof, Language and interaction — Essays in honour of Robin Milner*, pages 571–595, 2000.

Mark Stephenson, Jonathan Babb, and Saman Amarasinghe. Bitwidth analysis with application to silicon compilation. In *Proceedings of the ACM SIGPLAN 2000 conference on Programming language design and implementation*, PLDI '00, pages 108–120, New York, NY, USA, 2000. ACM.

Jun Sun, Wanghong Yuan, Mahesh Kallahalla, and Nayeem Islam. Hail: A language for easy and correct device access. In *Proceedings of the 5th ACM International Conference on Embedded Software*, EMSOFT '05, pages 1–9. ACM, 2005.

Karthik Sundaramoorthy, Zach Purser, and Eric Rotenburg. Slipstream processors: improving both performance and fault tolerance. In *Proceedings of the ninth international conference on Architectural support for programming languages and operating systems*, pages 257–268. ACM Press, 2000.

Timmy Sundström, Boris Murmann, and Christer Svensson. Power dissipation bounds for high-speed nyquist analog-to-digital converters. *Circuits and Systems I: Regular Papers, IEEE Transactions on*, 56(3):509–518, 2009.

A. Taber and E. Normand. *IEEE Trans. Nucl. Sci.*, 40:120, 1993.

Kiat Wee Tan, Tadashi Okoshi, Archan Misra, and Rajesh Krishna Balan. Focus: A usable & effective approach to oled display power management. In *Proceedings of the 2013 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, UbiComp '13, pages 573–582, New York, NY, USA, 2013. ACM.

Texas Instruments. *CC256x Bluetooth® and Dual-Mode Controller*, Data Sheet, January 2014a.

Texas Instruments. *HDC1000 Low Power, High Accuracy Digital Humidity Sensor with Temperature Sensor*, Data Sheet, Nov 2014b.

Texas Instruments. *TMP006/B Infrared Thermopile Sensor in Chip-Scale Package*, Data Sheet, November 2014c.

Texas Instruments. *TPS8267x 600-mA, High-Efficiency MicroSIP™ Step-Down Converter*, Data Sheet, October 2014d.

Texas Instruments. *TPS82740x 360nA IQ MicroSIP™ Step Down Converter Module for Low Power Applications*, Data Sheet, June 2014e.

T. N. Theis and P. M. Solomon. In quest of the “next switch”: Prospects for greatly reduced power dissipation in a successor to the silicon field-effect transistor. *Proceedings of the IEEE*, 98(12):2005 –2014, Dec 2010.

Jonathan Ying Fai Tong, David Nagle, and Rob. A. Rutenbar. Reducing power by optimizing the necessary precision/range of floating-point arithmetic. *IEEE Trans. Very Large Scale Integr. Syst.*, 8:273–285, June 2000.

G. V. Varatkar, S. Narayanan, N. R. Shanbhag, and D. L. Jones. Stochastic networked computation. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, PP(99):1–1, October 2009.

Benjamin Vigoda. *Analog logic: Continuous-Time analog circuits for statistical signal processing*. PhD thesis, Massachusetts Institute of Technology, 2003.

Benjamin Vigoda, David Reynolds, Jeffrey Bernstein, Theophane Weber, and Bill Bradley. Low power logic for statistical inference. In *Proceedings of the 16th ACM/IEEE international symposium on Low power electronics and design*, ISLPED ’10, pages 349–354, New York, NY, USA, 2010. ACM.

John von Neumann. Probabilistic logics and the synthesis of reliable organisms from unreliable components. *Automata Studies*, pages 43–98, 1956.

John von Neumann and Ray Kurzweil. *The computer and the brain*. Yale University Press, 2012.

I. Wagner and V. Bertacco. Engineering trust with semantic guardians. In *2007 Design, Automation & Test in Europe Conference & Exhibition*, page 140. IEEE, 2007.

David Walker, Lester Mackey, Jay Ligatti, George A. Reis, and David I. August. Static typing for a faulty lambda calculus. In *Proceedings of the eleventh ACM SIGPLAN international conference on Functional programming*, ICFP ’06, pages 38–49, New York, NY, USA, 2006. ACM.

Gregory K Wallace. The jpeg still picture compression standard. *Communications of the ACM*, 34(4):30–44, 1991.

Ji Wang, Xiao Lin, and Chris North. GreenVis : Energy-Saving Color Schemes for Sequential Data Visualization on OLED Displays. 2012.

Chris Weaver and Todd M. Austin. A fault tolerant approach to microprocessor design. In *Proceedings of the 2001 International Conference on Dependable Systems and Networks (formerly: FTCS)*, pages 411–420. IEEE Computer Society, 2001.

Herbert S Wilf and Albert Nijenhuis. *Combinatorial algorithms: an update*. SIAM, 1989.

Niklaus Wirth. What can we do about the unnecessary diversity of notation for syntactic definitions? *Commun. ACM*, 20(11):822–823, November 1977.

Shirley Wong, Simon Lucas, Alex Panaretos, Luis Velazquez, Robert Young, and Anthony Tang. Robust word recognition dataset. In *ICDAR*, 2003.

Vicky Wong and Mark Horowitz. Soft error resilience of probabilistic inference applications. In *In Proceedings of the Workshop on System Effects of Logic Soft Errors*, 2006.

Gunther Wyszecki and WS Stiles. *Color Science: Concepts and Methods, Quantitative Data and Formulae*. Wiley-Interscience, New York, 2000.

YZ Xu, H. Puchner, A. Chatila, O. Pohland, B. Bruggeman, B. Jin, D. Radaelli, and S. Daniel. Process impact on SRAM Alpha-particle SEU performance. In *2004 IEEE International Reliability Physics Symposium Proceedings, 2004. 42nd Annual*, pages 294–299, 2004.

Mengying Zhao, Yiran Chen, Xiang Chen, and Chun Jason Xue. Online oled dynamic voltage scaling for video streaming applications on mobile devices. *SIGBED Rev.*, 10(2):18–18, July 2013.

Neil Zhao. Full-Featured Pedometer Design Realized with 3-Axis Digital Accelerometer. *Analog Dialogue*, 44(06), June 2010.

James F. Ziegler and William A. Lanford. Effect of cosmic rays on computer memories. *Science*, 206(4420):776–788, 1979.

James F. Ziegler and William A. Lanford. The effect of sea level cosmic rays on electronic devices. *Journal of applied physics*, 52(6):4305–4312, 1981.

James F. Ziegler, Huntington W. Curtis, Hans P. Muhlfeld, Charles J. Montrose, B. Chin, Michael Nicewicz, C. A. Russell, Wen Y. Wang, Leo B. Freeman, P. Hosier, et al. Ibm experiments in soft fails in computer electronics (1978–1994). *IBM journal of research and development*, 40(1):3–18, 1996.