*Preprint. To appear, IEEE Computer Architecture Letters (2015)*

# Comparing Stochastic and Deterministic Computing

Rajit Manohar
Cornell Tech, New York, NY 10011, USA

**Abstract**—Technology scaling has raised the specter of myriads of cheap, but unreliable and/or stochastic devices that must be creatively combined to create a reliable computing system. This has renewed the interest in computing that exploits stochasticity—embracing, not combating the device physics. If a stochastic representation is used to implement a programmable general-purpose architecture akin to CPUs, GPUs, or FPGAs, the preponderance of evidence indicates that most of the system energy will be expended in communication and storage as opposed to computation. This paper presents an analytical treatment of the benefits and drawbacks of adopting a stochastic approach by examining the cost of representing a value. We show both scaling laws and costs for low precision representations. We also analyze the cost of multiplication implemented using stochastic versus deterministic approaches, since multiplication is the prototypical inexpensive stochastic operation. We show that the deterministic approach compares favorably to the stochastic approach when holding precision and reliability constant.

## 1 Introduction

Modern CMOS technology can produce reliable chips with many billions of devices. With power being a major consideration in any modern VLSI system, researchers are always looking for approaches to reducing the energy cost of computation. Examples of this include very low voltage CMOS circuits, using new devices for storage and computation, self-timed computation, molecular substrates, and new models of computation such as brain-inspired computing. Reducing the precision at which a computation is performed can also provide significant benefits (e.g. [2], [9]).

One approach to low-precision computation that was explored historically (for example, in [10], [18]) and that has gained some traction in recent years is the notion of *stochastic* computing. In a stochastic computing approach, the designer of a system embraces the notion of probabilistic computing in a fundamental way. A real-valued variable in the range $[0, 1]$ is represented as a sequential stream of bits that can be either $0$ or $1$, where the probability of being a $1$ represents the value being communicated on the wire. Recent work has used stochastic computing for image processing [1], low-density parity check codes [11], factor graphs [16], digital signal processing [14], and recognition/data mining [6]. The goal of this paper is to take an analytical approach to comparing stochastic computing with the more conventional deterministic (non-stochastic) approach to computing.

In any modern highly programmable VLSI system, the energy cost of communication and storage (including moving data to/from memory) dominates the cost of computation. Existing chips that fall into this category include microprocessors, graphics processing units, field-programmable gate arrays and neuromorphic architectures. Due to the importance of the cost of communication, this paper examines a basic question: what is the representation cost (in bits) of a real-valued variable $v$ in the range $[0, 1]$? This directly impacts both storage and data movement costs. We examine a number of different representations including conventional digital, non-standard digital, and stochastic. We also present an analysis of prototypical examples of stochastic computing—multiplication of two numbers, and edge detection—as they are often used to justify the efficiency of stochastic computing architectures. Our analysis shows that deterministic computing compares quite favorably to stochastic computing when one controls for the true precision at which the computation is performed.

*Manuscript submitted: 02-Feb-2015. Manuscript accepted: 02-Mar-2015. Final manuscript received: 06-Mar-2015.*

## 2 Representation cost

Consider the problem of representing a real-valued variable $v$, where $v \in [0, 1]$. $v$ might take on any value in this range. To ensure that we are making a fair comparison between different approaches, we require that $v$ be represented with a fixed precision specified by an error tolerance $\epsilon$. In other words, given the value $v$, the representation can use any $\hat{v}$ as long as

$$|v - \hat{v}| \leq \epsilon \tag{1}$$

What metric is suitable for comparing different approaches to this question? We posit that a useful metric for this purpose is the number of bits that must be communicated from a sender to receiver in order to satisfy Equation 1. In what follows, we examine a variety of approaches to this problem, and quantify their cost in terms of the number of bits that have to be transmitted from the sender to the receiver for correct communication. Note that this is also the storage cost. For most of this section, we assume that the actual process of transporting bits from sender to receiver is error-free. We address the issue of errors in the final part of this section.

### 2.1 Standard Digital Representation

The conventional approach to solving the communication problem is to simply uniformly sub-divide the range $[0, 1]$ into intervals of size $2\epsilon$. Hence, the interval $[0, 1]$ is broken down into $1/2\epsilon$ buckets, where the $i$th bucket is

$$B_i : [i/2\epsilon, (i + 1)/2\epsilon] \qquad i = 0, 1, \ldots, \lceil 1/2\epsilon \rceil - 1 \tag{2}$$

(For simplicity we assume that $1/2\epsilon$ is an integer.) The problem of communicating $v$ is converted into identifying the bucket $i$ that contains $v$, and then representing the bucket identifier using $\lg 1/2\epsilon$ bits. This corresponds to using a fixed-point representation for the value $v$, something that is routinely done in modern computer systems especially in the embedded computing space where power places a more severe constraint.[1] Often the value $\lg 1/2\epsilon$ is used to specify the precision of the computation in the units of *bits* rather than error; that is, $b$ bits of precision corresponds to $\epsilon = 2^{-(b+1)}$.

1. Floating-point representations place a constraint on *relative* error rather than *absolute* error as we have done in Equation 1.

## 2.2 Stochastic Representation

In stochastic data representations, a value is represented as the *average* of a number of independent, identically distributed (i.i.d.) random variables. In the classic approach (see [17] for a summary of work prior to the early 1980s), a value $v$ is interpreted as a probability. Instead of sending the value $v$, a sender sends a stochastic sequence of zeros and ones, where the probability of transmitting a 1 is given by $v$. Since the expected value of the $0/1$ sequence is $v$ (the probability), this sequence is another way to transmit the value $v$.

What is the cost of communicating $v$ in this manner? This is a slightly more complicated question than the original problem that we posed. To understand why, consider the problem of sending the value $v = 0.5$ using a $0/1$ sequence of length $N$ with error that is less than $\epsilon = 0.1$. Since the sequence is i.i.d., with probability $0.5^N$ we transmit $N$ zeros. For finite $N$, although this probability may be very small, it is never zero. Hence, we cannot *guarantee* that we send $v$ with at most $\epsilon$ error. Instead, we must use a slightly weaker guarantee—namely, that we transmit the required value *with high probability*. Let $\delta$ be the probability that Equation 1 is not satisfied. In other words, our problem statement for the stochastic case is modified by assuming that the receiver is permitted to receive $\hat{v}$ as long as

$$\max_{v\in[0,1]} \{\Pr_{\hat{v}}[|v-\hat{v}| > \epsilon]\} \le \delta \tag{3}$$

We can use concentration inequalities to derive an analytical expression that relates the number of bits used, $N$, with the quality of the received value $\hat{v}$. In our scenario, the receiver has $v_1, v_2, \ldots, v_N$ as $N$ i.i.d. Bernoulli random variables, and computes $\sum_{i=1}^{N} v_i/N$ as the estimate $\hat{v}$. Hence, by using Hoeffding-Chernoff bounds [5], [13], we know that

$$\Pr[|v-\hat{v}| > \epsilon] \le 2e^{-2N\epsilon^2} \tag{4}$$

Note the similarity between Equations 3 and 4. This means that to have a good estimate of the value $\hat{v}$ at the receiver, the number of samples relates to $\delta$ by:

$$\delta = 2e^{-2N\epsilon^2} \quad \Rightarrow \quad N = \frac{1}{2\epsilon^2}\ln\frac{2}{\delta}$$

In other words, the number of bits required to communicate $v$ with tolerance $\epsilon$ (with high probability $(1-\delta)$) scales as $O(1/\epsilon^2)$ using a stochastic approach to communication. While Hoeffding-Chernoff bounds are not tight for small $N$, they do capture the correct scaling law. For small $N$, we can numerically compute the minimum value of $N$ such that Equation 3 is satisfied given $\delta$ and $\epsilon$. This can be done through a counting argument, since the number of ones follows a binomial distribution and we can numerically compute the probability that the number of ones is too large or too small to accurately represent the value $v$; we do so in Section 2.5. Other work has noted that $\epsilon$ must scale as $1/\sqrt{N}$ (e.g. see [17]); however, the role of $\delta$, which is critically important (Section 2.5), has been ignored.

## 2.3 Deterministic 0/1 Encoding

The stochastic approach assumed that the bits transmitted were in fact i.i.d. random variables. What if we remove this constraint? In our example of transmitting 0.5, for any even value of $N$ we can *guarantee* that we send precisely $N/2$ ones and $N/2$ zeros, ensuring that the constraint of Equation 1 is always satisfied. This is a well-known idea, and pulse-width modulation is a special case where the transmitted zeros and ones are clustered.

Assuming that we transmit $N$ bits, the deterministic $0/1$ scheme sets a *deterministic* fraction of bits to 1 based on the value $v$. Switching one bit from a 0 to a 1 or vice versa changes the value being transmitted by $1/N$. Hence, to ensure that Equation 1 is satisfied, we must have $\epsilon \ge \frac{1}{2N}$.

TABLE 1
Summary of the complexity of representation and communication cost to bound receiver error by $\epsilon$ (with high probability $(1-\delta)$ for the stochastic case). The "Adaptive?" column indicates if the representation can dynamically adapt to different precision requirements.

| Technique | Bits | Complexity | Adaptive? |
|---|---|---|---|
| Standard digital | $\lg 1/2\epsilon$ | $O(\log 1/\epsilon)$ | N |
| Stochastic | $1/2\epsilon^2 \lg 2/\delta$ | $O(1/\epsilon^2)$ | Y |
| Deterministic $0/1$ | $1/2\epsilon$ | $O(1/\epsilon)$ | Y |
| Adaptive digital | $\lg 3 \lg(1/\epsilon)$ | $O(\log 1/\epsilon)$ | Y |

In this approach, the number of bits required to attain a particular error rate is given by $1/2\epsilon$—clearly superior to a purely stochastic approach. The cost of doing this is that two input streams represented in this manner cannot be combined using the simple probability rules that assume independence, because the value streams are not random. For some special cases, deterministic $0/1$ sequences have been developed with simple combination rules [12].

## 2.4 Adaptive Digital Representation

One of the benefits of using deterministic $0/1$ or stochastic encoding over conventional digital circuits is that they permit *adaptive* precision. Since a circuit capable of computing on serial bit-streams tends to be the same for arbitrary bit-stream lengths, this automatically permits a trade-off between delay (as $N$ gets larger, the delay is larger assuming a fixed bandwidth for communication) and precision. This is appealing for applications such as wireless receivers, where the precision requirements vary based on input signal strength.

However, there is an adaptive *logarithmic* encoding possible for digital computation. Instead of encoding numbers with zeros and ones, the encoding can include a *third* digit that indicates the "end" of the number [15]. The energy usage of such a design *adapts* automatically to the number of significant digits in the number. In the simplest incarnation, this means each digit is one of three values, requiring $\lg 3$ bits per digit. Also, we need an extra digit to specify that the number ends. Hence we require $\lg \frac{1}{2\epsilon}$ digits, plus one more to indicate that the number ends. Each digit requires $\lg 3$ bits. Hence, the cost is $\lg 3 \lg(1/\epsilon)$ bits to represent a number to $\epsilon$ precision. Combining this approach with a bit-serial architecture provides the benefits of logarithmic encoding with adaptive precision. This approach was implemented in a complete microcontroller [8], demonstrating the practicality of this approach. Many variants of this approach are possible (see [4], [15]) trading off complexity with the granularity of adaptation.

TABLE 2
The communication and representation cost in bits for various approaches; lower is better. For the stochastic case, $\delta$ is the probability that we fail to achieve the desired precision (Eq. 3); lower is better. $\delta$ for the deterministic cases is 0. $b$ bit precision is $\epsilon = 2^{-(b+1)}$.

| Prec. | Deterministic | | | Stochastic | | |
|---|---|---|---|---|---|---|
| (bits) | Std. | 0/1 | Adapt. | $\delta = 0.05$ | $\delta = 0.10$ | $\delta = 0.25$ |
| 1 | 1 | 2 | 3.16 | 12 | 8 | 3 |
| 2 | 2 | 4 | 4.75 | 56 | 37 | 16 |
| 3 | 3 | 8 | 6.34 | 233 | 160 | 80 |
| 4 | 4 | 16 | 7.92 | 960 | 672 | 320 |
| 5 | 5 | 32 | 9.51 | 3904 | 2752 | 1344 |

TABLE 3
Same as Table 2, except the stochastic case has to satisfy a weaker constraint $\mathrm{E}_v[\Pr_{\hat{v}}[|v-\hat{v}|>\epsilon]] \le \overline{\delta}$, where $v$ is uniformly distributed.

| **Prec.** | **Deterministic** | | | **Stochastic (avg. error)** | | |
|---|---|---|---|---|---|---|
| (bits) | Std. | 0/1 | Adapt. | $\overline{\delta}=0.05$ | $\overline{\delta}=0.10$ | $\overline{\delta}=0.25$ |
| 1 | 1 | 2 | 3.16 | 8 | 4 | 2 |
| 2 | 2 | 4 | 4.75 | 40 | 24 | 8 |
| 3 | 3 | 8 | 6.34 | 176 | 112 | 48 |
| 4 | 4 | 16 | 7.92 | 704 | 448 | 192 |
| 5 | 5 | 32 | 9.51 | 2816 | 1856 | 832 |

## 2.5 The Low Precision Scenario

From the standpoint of communication and number representation (which can be viewed as a proxy for storage), the asymptotic results from Table 1 show that the conventional wisdom of using fixed-point representation for any reasonable precision is actually well-founded. The scaling laws we have derived show that the cost for stochastic (and even deterministic 0/1) communication will quickly become prohibitive when compared to the deterministic fixed-point (both adaptive and non-adaptive) approach. Two natural questions that arise about Table 1 are: (i) what happens if the precision required is very low? That is, after all, when the stochastic representation should have benefits. (ii) The bounds we derived on $\delta$ in Equation 4 are only bounds and may be misleading.

We address both of these issues by computing the number of bits required for each representation for low values of precision $\epsilon$ in Table 2. For standard digital (labeled "Std."), deterministic 0/1 (labeled "0/1"), and adaptive digital (labeled "Adapt."), our expressions from Table 1 are exact and so we simply use the formula. For the stochastic case, we compute the value of $N$ exactly as follows. If the stochastic sequence is of length $N$ and we receive $k$ ones ($k = 0, 1, \ldots, N$), the value received $\hat{v}$ is $k/N$. The probability of $k$ ones in the sequence when the value $v$ is being represented is given by $\binom{N}{k} v^k (1-v)^{N-k}$. We use this expression to compute the smallest number of bits required in the stochastic case to satisfy Equation 3. The values $v$ that we use are of the form $i/\lceil 1/2\epsilon \rceil$ for $i = 1, \ldots, \lceil 1/2\epsilon \rceil$.

Even if we allow the stochastic case to be incorrect with 25% probability, a one-bit precise representation requires more than 2 bits (Table 2). This may seem surprising, but one can actually see why this is the case. If we use two bits, then we cannot represent say the value 0.5 with a maximum error $\epsilon = 0.25$ (the definition of one-bit precision). This is because there is a 0.25 probability that the received sequence would be 00 (value 0), and a 0.25 probability that the received sequence would be 11 (value 1), resulting in an overall 0.5 probability that the precision requirements are not met. Increasing the reliability requirement to $\delta = 0.10$ requires eight bits—*even though the result is only guaranteed to be correct to one-bit precision*. Most of the literature on this topic ignores the critical role played by $\delta$—the probability that the result no longer satisfies the precision constraint. Finally, one could argue that Equation 3 is too strong of a constraint; we should relax it and use the expected value rather than the maximum on the left hand side of (3). Table 3 is the result of this exercise, showing that the stochastic representation remains costly.

## 2.6 Communication Errors

So far we have not addressed the issue of communication errors. We could have avoided the entire discussion so far and stated that using a fixed-point representation is optimal. The reason follows from information theory. If we assume that the input $v \in [0,1]$ is uniformly distributed over the interval, and we must represent this with error at most $\epsilon$, then the optimal scheme is to divide the interval into uniform bins each of width $2\epsilon$. The entropy of the symbols used to represent $v$ is given by $\ln 1/2\epsilon$ nats, which corresponds to $\lg 1/2\epsilon$ bits. This is achieved by the fixed-point representation. Hence, fixed-point representation is optimal assuming that the input $v$ is uniformly distributed on the $[0,1]$ interval.

What happens if there are errors possible in the communication link from the sender to the receiver? While at first this appears to be a complex question, information theory provides a concrete answer since we have a single sender and single receiver scenario with stationary statistics. Shannon's source and channel coding separation theorem says that encoding the value to be transmitted *separately* from error control coding for the communication link is optimal [7]. Hence, treating the problem of encoding for errors in the channel separately from the representation question is sufficiently general.

Suppose we have a communication channel that randomly flips bits with probability $f$. The effect on stochastic computing is to add a bias and scale to the value communicated—value $v$ changes to $v(1-f) + (1-v)f = f + (1-2f)v$. In the deterministic case of using $b$ bits, as long as the error probability $\hat{f} = 1 - (1-f)^b$ is *below* the value of $\delta$, *no error correction* is necessary to satisfy constraint (3). If $\hat{f} > \delta$, then even the inefficient repetition code that repeats each bit $2k-1$ times (decoding is a simple majority circuit) can reduce $f$ to $\sum_{i=k}^{2k-1} \binom{2k-1}{i} f^i (1-f)^{2k-1-i}$. As a concrete example, consider a communication channel with $f = 0.10$ where we must transmit 4-bit precise values with $\delta = 0.1$. $\hat{f} = 0.344$, and the value of $f$ low enough to make $\hat{f} \le \delta$ is $f = 0.025$. This reduction can be achieved with $k = 3$, i.e. a repetition count of 5. Hence, the deterministic scenario transmits 20 bits to send value $v$ v/s the stochastic case that uses 672 bits to send value $0.1 + 0.8v$.

If we know the distribution of $v$ in the $[0,1]$ interval, we can use it to *compress $v$ before* encoding the value to be transmitted over the channel. Once the data is compressed, $v \in [0,1]$ will be uniformly distributed and our earlier analysis holds.

## 3 Analysis of Simple Computation

While the cost of computation today is small relative to communication and storage, it is still instructive to examine the difference in cost between simple fixed-point and stochastic computations when precision is held constant.

Multiplication: The classic example of why stochastic computing is worth examining in detail is the problem of multiplication. Fixed-point multiplication is a relatively expensive arithmetic operation. The three most commonly used ways to implement fixed-point $n$-bit multiplication are: (i) an iterative scheme, with an $n$-bit add/shift block and extra registers—$O(n)$ hardware—that takes $O(n)$ iterations; (ii) a standard array multiplier, with $O(n^2)$ hardware ($n$ stages of $n$-bit carry save addition, followed by a fast $n$-bit adder) that takes latency $O(n)$, but can have $O(1)$ throughput; (iii) a carry save tree like a Wallace tree, that uses $O(n^2)$ hardware and has $O(\log n)$ latency and $O(1)$ throughput.

Stochastic multiplication, on the other hand, is trivial. We assume that the statistical properties of the two bit-streams are independent of each other. Recall that the value represented by a stochastic bit-stream is the probability that a bit is 1. If the two probabilities are $a$ and $b$, and they are independent, then the probability that both bits are 1 is $a \cdot b$. Hence, a single AND gate implements stochastic multiplication—for *any* precision. On the surface this seems extremely efficient compared to traditional multiplication of fixed-point values.

However, consider the multiplication problem where we impose a fixed error constraint on the result. If the two values

TABLE 4
The gate invocation cost for fixed-point multiplication versus stochastic multiplication; lower is better. For the stochastic case, $\delta$ is the probability that we fail to achieve the desired precision (Equation 3); lower is better. The equivalent $\delta$ for the fixed point case is 0. The bits refer to the precision of the inputs to the multiplier.

| **Bits** | **Fixed-point** | **Stochastic** | | |
|---|---|---|---|---|
| | | $\delta = 0.05$ | $\delta = 0.10$ | $\delta = 0.25$ |
| 1 | 2 | 24 | 16 | 6 |
| 2 | 6 | 112 | 74 | 32 |
| 3 | 36 | 466 | 320 | 160 |
| 4 | 111 | 1920 | 1344 | 640 |
| 5 | 198 | 7808 | 5504 | 2688 |

being multiplied are $a$ and $b$, and we require the result $ab$ to be represented with at most $\epsilon$ error, we require the inputs $a$ and $b$ to be represented with $\epsilon/2 - O(\epsilon^2)$ error [3]. From Tables 1 and 2, this means that the stochastic case requires many more invocations of the AND gate—i.e. $O(1/\epsilon^2)$ gate invocations. The conventional fixed-point arithmetic scenario requires $O(n^2)$ operations, where $n = \lg 1/\epsilon$ (Table 1)—in other words, $O(\log^2 1/\epsilon)$ gate invocations. Hence, the complexity of using fixed-point representations is *lower* than the stochastic case when we control for precision.

From the asymptotic analysis, it is clear that a fixed-point multiplier will eventually be significantly superior to a stochastic multiplier. What about for low precision? To provide a first-order analysis of this, we use "gate invocations" to compare fixed-point versus stochastic multipliers for various precisions (measured in bits) to capture the fact the stochastic circuit must be invoked many more times than the fixed-point multiplier. A detailed comparison would require technology-specific information; to avoid this, we simply use a weighted count (dynamic area as a proxy for switching energy) of the gates with a very simple area model—for example, inverters have cost 1, 2-input ANDs and ORs have cost 2, a 4-input AND/OR has cost 3, etc.

For the fixed-point case, we used Synopsys design compiler to synthesize fixed-point combinational logic multipliers of various bit-widths. Note that this requires the high-order $n$ bits of the output of an $n$-bit integer multiplier. For the stochastic case, we multiply the cost of a two-input AND by the number of times it must be invoked to attain the same precision as the fixed-point case. Since the analytical expression for the precision attained in the stochastic case is based on a bound, we use an exact expression for the error probability $\delta$ (Section 2.5).

The results of this exercise are summarized in Table 4, which assumes one input is correct[2] with probability $(1 - \delta)$. A one-bit fixed-point multiplier is *cheaper* than the stochastic case. The gap between the two diverges for higher precision. Table 4 should not be interpreted too literally; the take-away is that even at low precision a fixed-point representation is not expensive when compared to a stochastic multiplier due to the number of bits that must be processed in the stochastic case to attain equal precision with high probability.

Edge detection: Another example used to show the efficiency of stochastic computing is simple edge detection using the Roberts cross operator which computes $z_{i,j} = 0.5(|x_{i,j} - x_{i+1,j+1}| + |x_{i,j+1} - x_{i+1,j}|)$ where $x_{i,j}$ is the pixel at location $(i, j)$ and $z_{i,j}$ is the corresponding output. Recent work uses a stochastic sequence of $N = 1024$ bits to represent 8-bit pixel values [14]. We implemented the same edge-detection algorithm, except instead of using standard images we simply used an image that is 0.5 at each pixel. The correct output is an image that is all zeros. Our simulations found that an idealized error-free computation with a stochastic pixel representation would produce an output image that is correct with at most 2-bit precision when $N = 256$, and at most 4-bit precision even when $N = 1024$. This is more pessimistic than Table 2 because, for example, subtracting two values that are imprecise—a part of edge detection—can result in a doubling of the error.

2. In the general case of multiple outputs, *all* outputs must be correct with confidence $\delta$, not just one.

## 4 Conclusions

We compared the cost of stochastic number representations with more conventional deterministic approaches. We control for the fact that stochastic representations have a non-zero overall error probability, beyond precision limitations. Our comparison provides both asymptotic complexity comparisons as well as calculations for finite precisions, in both cases showing that stochastic representations are more expensive than one might expect. We also present a similar analysis of the cost of multiplication, because it is often used as an argument in favor of stochastic computing. Our results show that even for simple computation, stochastic computing has disadvantages both in asymptotic complexity as well as in the low precision scenario given our criterion for correct information representation.

## References

[1] A. Alaghi, C. Li, and J. P. Hayes, "Stochastic circuits for real-time image-processing applications," in *Proc. of DAC*, 2013.

[2] J. Bornholt, T. Mytkowicz, and K. S. McKinley, "Uncertain<t>: A first-order type for uncertain data," in *Proc. ASPLOS*, 2014, pp. 51–66.

[3] R. Bulirsch and J. Stoer, *Introduction to numerical analysis*. Springer Heidelberg, 2002.

[4] R. Canal, A. González, and J. E. Smith, "Very low power pipelines using significance compression," in *Proceedings of the ACM/IEEE International Symposium on Microarchitecture*, 2000, pp. 181–190.

[5] H. Chernoff, "A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations," *The Annals of Mathematical Statistics*, pp. 493–507, 1952.

[6] V. K. Chippa, S. Venkataramani, K. Roy, and A. Raghunathan, "Storm: A stochastic recognition and mining processor," in *Proc. Int. Symp. on Low Power Electronics and Design*, 2014, pp. 39–44.

[7] T. M. Cover and J. A. Thomas, *Elements of information theory*. Wiley-interscience, 2006.

[8] V. N. Ekanayake, C. Kelly IV, and R. Manohar, "Bitsnap: Dynamic significance compression for a low-energy sensor network asynchronous processor," in *Proceedings of the IEEE International Symposium on Asynchronous Circuits and Systems*, 2005, pp. 144–154.

[9] H. Esmaeilzadeh, A. Sampson, L. Ceze, and D. Burger, "Architecture support for disciplined approximate programming," in *Proc. ASPLOS*, 2012, pp. 301–312.

[10] B. R. Gaines, "Stochastic computing," in *Proceedings of the April 18-20, 1967, Spring Joint Computer Conference*, 1967, pp. 149–156.

[11] V. C. Gaudet and A. C. Rapley, "Iterative decoding using stochastic computation," *Electronics Letters*, vol. 39, no. 3, pp. 299–301, 2003.

[12] P. Gupta *et al.*, "Binary multiplication with PN sequences," *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 36, no. 4, pp. 603–606, 1988.

[13] W. Hoeffding, "Probability inequalities for sums of bounded random variables," *Journal of the American statistical association*, vol. 58, no. 301, pp. 13–30, 1963.

[14] P. Li and D. J. Lilja, "Using stochastic computing to implement digital image processing algorithms," in *IEEE International Conference on Computer Design*, 2011, pp. 154–161.

[15] R. Manohar, "Width-adaptive data word architectures," in *Proc. 2001 Conference on Advanced Research in VLSI*, 2001, pp. 112–129.

[16] V. K. Mansinghka, "Natively probabilistic computation," Ph.D. dissertation, Massachusetts Institute of Technology, 2009.

[17] P. Mars and W. J. Poppelbaum, *Stochastic and deterministic averaging processors*. P. Peregrinus, 1981.

[18] W. J. Poppelbaum, C. Afuso, and J. W. Esch, "Stochastic computing elements and systems," in *Proceedings of the November 14-16, 1967, Fall Joint Computer Conference*, 1967, pp. 635–644.