This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/LCA.2018.2804926, IEEE Computer Architecture Letters

On Memory System Design for Stochastic Computing

S. Karen Khatamifard, M. Hassan Najafi, Ali Ghoreyshi, Ulya R. Karpuzcu, David J. Lilja
University of Minnesota, Twin Cities

Abstract—Growing uncertainty in design parameters (and therefore, in design functionality) renders stochastic computing particularly promising, which represents and processes data as quantized probabilities. However, due to the difference in data representation, integrating conventional memory (designed and optimized for non-stochastic computing) in stochastic computing systems inevitably incurs a significant data conversion overhead. Barely any stochastic computing proposal to-date covers the memory impact. In this paper, as the first study of its kind to the best of our knowledge, we rethink the memory system design for stochastic computing. The result is a seamless stochastic system, StochMem, which features analog memory to trade the energy and area overhead of data conversion for computation accuracy. In this manner StochMem can reduce the energy (area) overhead by up-to 52.8% (93.7%) at the cost of at most 0.7% loss in computation accuracy.

1 Motivation

Stochastic Computing (SC) has received renewed attention in recent [2], [3], [4], [6]. This is due to the growing uncertainty in design parameters, and therefore, in design functionality, as induced by imbalances in modern technology scaling. Representing and processing data as quantized probabilities, SC becomes a natural fit. Data operands in SC take the form of bitstreams which encode probabilities: independent of the length (and interleaving of 0s and 1s), the ratio of the number of 1s to the length of the bitstream determines the operand value. Computation accuracy increases with the length of the bitstream at the cost of higher-latency stochastic operations [4]. Still, computing with probabilities can reduce arithmetic complexity significantly, such that the hardware resource cost and the power consumption become orders of magnitude less than their conventional (i.e., non-stochastic) counterparts [9], [14]. At the same time, computing with probabilities results in better tolerance to inaccuracy in input data operands [4].

The common focus of SC proposals from 1960s onwards has been stochastic logic (arithmetic), neglecting memory, which represents a crucial system component. Memory mainly serves as a repository for data collected from external resources (e.g., sensors) or data generated by previous steps of computation, to be used at later stages of computation. Algorithmic characteristics dictate both, the memory capacity requirement and the memory access pattern (particularly for data re-use). Most SC proposals deploy conventional digital memories (designed and optimized for non-stochastic computing) to address such algorithmic needs. Unfortunately, this practice increases hardware design complexity due to the discrepancy in conventional digital (i.e., non-stochastic) and stochastic data representations. Digital to/from stochastic data conversion can reach 80% or more of the overall energy consumption and hardware cost, which can easily diminish any benefit from stochastic computing [3], [16]. *In this study, we rethink the memory system design for stochastic computing.*

Practically *seamless* conversion options between analog and stochastic data representations [5], [15] makes analog memory stand out as a particularly promising point in the memory design space for SC. The downside is potential loss in data accuracy, where a divergence between the written/stored and the read values (at the same memory address) often becomes inevitable, however, which stochastic logic can mask due to its implicit tolerance to inaccuracy in input data operands.

This paper quantitatively characterizes the potential of analog memory for seamless SC, using a representative near-sensor stochastic image processing system as a case study[1]. We will refer to the resulting (practically) seamless stochastic system as StochMem. Cameras have already become ubiquitous sensors. There is a demand for near-sensor image processing both to reduce costly communication with the cloud and to enhance security and privacy. Real-time image processing algorithms often track differences between a stream of frames. It is not uncommon that the processing of the instantaneous frame requires comparison to a history of previously processed frames, which has to be stored in and retrieved from some form of memory. In the following, we will cover five representative image processing applications which span diverse compute and memory access characteristics.

Manuscript submitted: 02-Aug-2017. Manuscript accepted: 20-Dec-2017. Final manuscript received: 26-Dec-2017. This work was supported in part by National Science Foundation grant no. CCF-1408123 and XPS-CCA-1438286. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the NSF.

2 Toward Seamless SC

We will first compare and contrast StochMem featuring analog memory with the corresponding stochastic near-sensor image processor featuring conventional digital memory as a representative baseline.

2.1 Baseline: Stochastic Logic + Conventional Memory

Fig. 1a provides an overview for the baseline stochastic near-sensor image processor featuring conventional digital memory (designed and optimized for non-stochastic computing). The input data operands may represent the result bitstreams of previous steps of (stochastic) computation, or may directly come from analog image sensors. To be able to store such input data in conventional digital memory, a *Stochastic to Digital Converter*, SDC (for stochastic input bitstreams) or an *Analog to Digital Converter*, ADC (for analog inputs coming from sensors) become necessary. Moreover, further (stochastic) processing of the stored data necessitates a *Digital to Stochastic Converter*, DSC, upon data retrieval from digital memory. In the following we briefly describe key system components.

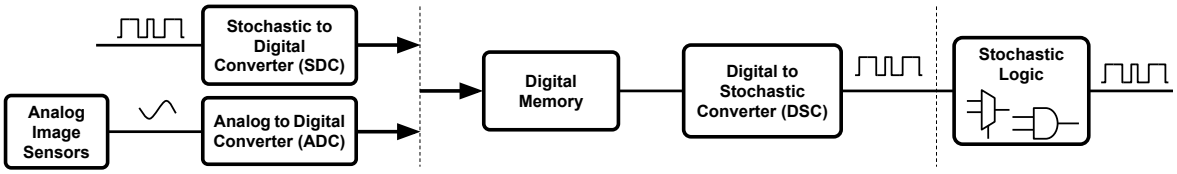
Stochastic Logic incorporates a circuit of basic Boolean gates to carry out the application-specific stochastic computation (Section 3.2). The inputs and outputs are both stochastic bitstreams.

Stochastic to Digital Converter (SDC) can generate the conventional binary representation for any stochastic bitstream. A digital counter usually serves the purpose, by keeping track of the number of 1s in the input bitstream to be converted. An SDC carries out data conversion if the inputs to the stochastic system represent result bitstreams from previous steps of (stochastic) computation.

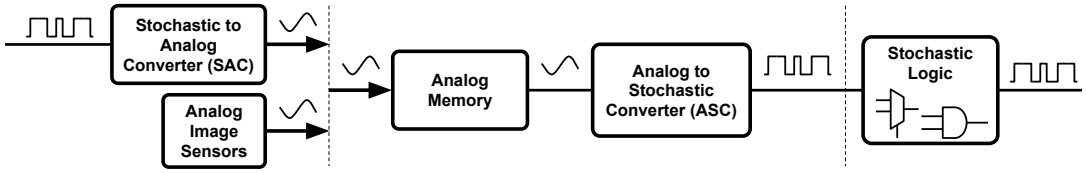
Analog to Digital Converter (ADC) becomes necessary if the inputs to the stochastic system directly come from analog image sensors. Conventional ADCs can serve the purpose. For most applications of SC (including the case study in this paper) an 8 to 10-bit ADC is sufficient [9].

1. Non-stochastic, analog near-sensor image processing accelerators such as [10] exist. The focus of this paper is not design and exploration of image processing accelerators. The scope rather is memory system design for stochastic computing where we use a representative stochastic system to characterize the impact of memory.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/LCA.2018.2804926, IEEE Computer Architecture Letters



(a) Baseline Near-Sensor Stochastic Image Processor featuring Digital Memory



(b) StochMem featuring Analog Memory

Fig. 1: Baseline Near-Sensor Stochastic Image Processor vs. StochMem.

Digital to Stochastic Converter (DSC) transforms conventional binary data retrieved from digital memory (for further stochastic processing) to stochastic bitstreams. Commonly, DSC achieves this by comparing an unbiased random number (obtained from a random number generator) to the binary value to be converted. A one is attached to the output (stochastic) bitstream if the random number is less than the binary value (to be converted); zero, otherwise. The random number generator can rely on physical random sources or pseudo-random constructs such as Linear Feedback Shift Registers (LFSRs).

2.2 StochMem: Stochastic Logic + Analog Memory

The data converters (SDC or ADC and DSC) incorporated into the baseline stochastic system from Fig. 1a each has a significant energy and area footprint [3], which can easily nullify potential benefits from SC. In order to reduce this overhead, StochMem replaces the conventional digital memory with its analog counterpart. Fig. 1b provides the overview for the resulting SC system. In the following we briefly describe key StochMem components:

Stochastic Logic is the same as under the baseline system.

Stochastic to Analog Converter (SAC) replaces the SDC of the conventional system. SAC can generate the analog representation for any stochastic bitstream. A conventional analog integrator can serve the purpose, by measuring the fraction of time a stochastic input bitstream stays at logic 1. Such an integrator usually has a smaller energy and area footprint than the SDC of the baseline system (Section 3.3). A SAC carries out data conversion if the inputs to StochMem represent result bitstreams from previous steps of (stochastic) computation.

Analog to Stochastic Converter (ASC) transforms data from analog memory (for further stochastic processing) to stochastic bitstreams, similar to the DSC of the conventional system. As representative examples, [5], [15] both cover energy-efficient ways for generating stochastic bitstreams from analog inputs.

3 Evaluation Setup

3.1 System Design

We evaluate three stochastic near-sensor image-processing designs: two different implementations of the baseline from Fig. 1a ($Conv_{LFSR}$ and $Conv_{MTJ}$) and StochMem. The two baseline designs differ in the implementation of data converters as follows:

$Conv_{LFSR}$: The baseline SC system featuring a 10-bit LFSR and a comparator as the DSC unit.

$Conv_{MTJ}$: The baseline featuring a DAC followed by an MTJ-based ASC as a more energy-efficient DSC. The rest of the system is identical to $Conv_{LFSR}$.

All systems first store the input in the memory. Then, they convert it to stochastic, and feed it to the stochastic logic.

3.2 Stochastic Applications

To evaluate *Stochastic Logic* from Fig. 1, we use stochastic circuits of five representative image processing applications: *Robert* (Robert's cross edge detection), *Median* (median filter noise reduction), *Frame* (frame difference-based image segmentation) from [9]; *Gamma* (gamma correction) from [16]; and *KDE* (kernel density estimation-based image segmentation) from [8].

(a) *Robert* (b) *Median* (c) *Frame* (d) *Gamma* (e) *KDE*

Fig. 4: Input (expected output) per application on top (bottom).

TABLE 1: Area and energy breakdown.

Stochastic Logic		
Circuit	Area (um^2)	Energy (pJ)(@1GHz)
Robert	339	0.440
Median	5382	4.090
Frame	457	0.413
Gamma	76	0.042
KDE	8691	7.094
Baseline System Parameters		
Unit	Area (um^2)	Energy (pJ)(@1GHz)
ADC 10-bit [13], [7]	50,000	20
SRAM cell	0.35	10
DSC: 10-bit LFSR	194	0.355
DSC: 10-bit Comparator	96	0.041
DSC: DAC 8-bit [11]	16,000	64
SDC: 10-bit Counter	254	0.179
StochMem System Parameters		
Unit	Area (um^2)	Energy (pJ)(@1GHz)
Analog memory cell [12]	58.7	10 (RD) / 100 (WR)
ASC [15]	15	0.030
SAC (integrator)	110	0.010

As input, we use 128×128 gray-scale images for *Robert*, *Median*, *Frame*, and *Gamma*; and 33 recent frames of a video, for *KDE*. Fig. 4 shows the input (expected output) images used for each application on the top (bottom) row. Expected output captures the maximum-possible accuracy. To calculate the accuracy of the end results, we calculate the average pixel-by-pixel difference between the output image of each stochastic circuit and the corresponding maximum-possible-accuracy output.

3.3 Hardware Parameters

Table 1 summarizes the area and energy consumption of different units of the evaluated stochastic systems. We synthesize logic units (including the stochastic circuit implementations of the five benchmark applications from Section 3.2), LFSR, digital comparator, and counter units using Synopsys Design Compiler vH2013.12 with a 45nm gate library. The Floating-Gate (FG) analog memory implementation follows [12]. To model inaccuracy of FG memory, we add Gaussian noise (with standard deviation from measured data in [12]) to the stored data.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/LCA.2018.2804926, IEEE Computer Architecture Letters

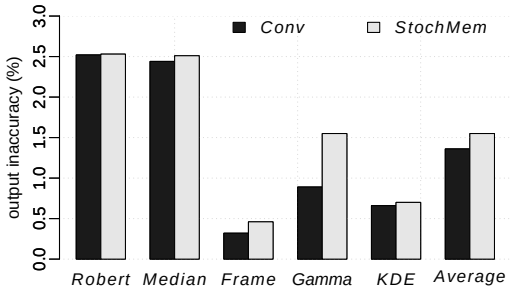
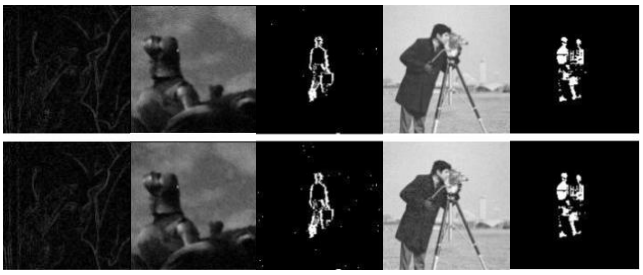


Fig. 2: Output inaccuracy of the baseline vs. StochMem.



(a) *Robert* (b) *Median* (c) *Frame* (d) *Gamma* (e) *KDE*

Fig. 3: Output images: Baseline (StochMem) on top (bottom).

For a fair evaluation, we assume that the input to both the baseline designs and StochMem directly comes from analog image sensors. All designs output a stochastic bitstream. Therefore, the evaluated systems do not feature an SDC or SAC on the feedback path from memory (Fig. 1). However, we include these units in Table 1 for the sake of completeness. SAC area (energy) cost is 2.3× (17.9×) less than SDC. Accordingly, if the evaluated systems deployed these units (as explained in Section 2), StochMem would have shown even larger gains when compared to the baseline.

4 Evaluation

Since all three alternative designs operate at the same frequency, they have similar throughput. So, we start the evaluation with a quantitative characterization of the accuracy loss in the outputs due to the potential read-write discrepancy of the analog memory incorporated in StochMem. We continue with energy consumption and conclude with area cost.

4.1 Output Accuracy of StochMem

A known downside of analog memory technologies is the potential discrepancy between values read and written/stored. We model the impact of this discrepancy after the accuracy measurements of a representative analog memory implementation [12]. All evaluated benchmark applications produce images as output. Therefore, we capture the accuracy loss in the output by the average per-pixel deviation (and SSIM [1]) from the "expected" output for each application as shown in the bottom row of Fig. 4.

Fig. 2 demonstrates the % output inaccuracy (in terms of average per-pixel deviation) of StochMem and the baseline designs for all applications under a stochastic bitstream length of 1024. The y-axis is normalized to the expected accuracy values corresponding to the images in the bottom row of Fig. 4. The two baseline designs evaluated, $Conv_{LFSR}$ and $Conv_{MTJ}$ (Section 3.1), feature the very same output inaccuracy, as given by the *Conv* bar in Fig. 2. We observe that, overall, the degradation (with respect to *Conv*) in the output accuracy of StochMem remains negligible. Only for *Gamma*, the inaccuracy becomes around 0.7% worse than *Conv*. For all other applications, the inaccuracy worsens by less than 0.15%. On average, the % output inaccuracy of StochMem is 1.55%; of *Conv*, 1.36%, with respect to the expected outputs. Besides, on average SSIM gets 3.2% worse for different applications. For the worst-case application, *Gamma*, SSIM gets 7.3% worse than *Conv*.

Fig. 3 tabulates the output images for all benchmark applications under StochMem and *Conv*. In accordance with the comparison results from Fig. 2, the difference in output accuracy is barely perceivable.

We repeat these experiments for 3 different bitstream lengths: 128, 256, and 512 bits. The average output inaccuracy of StochMem with respect to *Conv* increases from 4.08% to 4.21%, from 2.63% to 2.77%, and from 1.87% to 2.03%, as the bitstream length increases from 128 to 512, respectively. The relatively small degradation in the output inaccuracy is in line with the experimental outcomes summarized in Fig.s 2 and 3.

4.2 Reduction in Energy Consumption

We next compare and contrast the energy consumption of the evaluated stochastic designs. In the following, we report the experimental results for a bitstream length of 1024 without loss of generality. As Fig. 5 depicts, due to its more energy-efficient DSC implementation, $Conv_{MTJ}$ can decrease the energy consumption with respect to $Conv_{LFSR}$ significantly, by 45.7% on average. Introducing analog memory – i.e., StochMem – can reduce the energy consumption further, by 11.1% on average over $Conv_{MTJ}$.

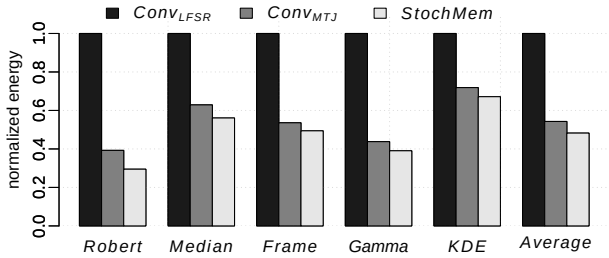
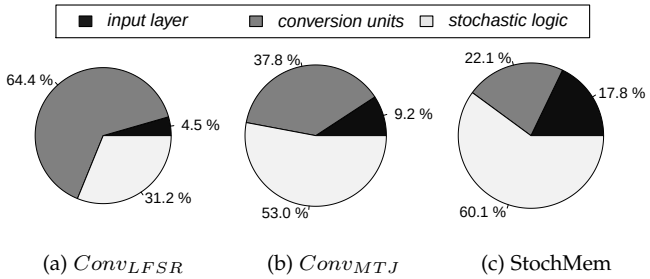


Fig. 5: Energy consumption normalized to $Conv_{LFSR}$.



(a) $Conv_{LFSR}$ (b) $Conv_{MTJ}$ (c) StochMem

Fig. 6: Share of energy consumed by different units.

To demonstrate the sources of these energy gains, we quantify the share of energy spent in different units. We expect an energy-efficient stochastic system to spend most of its energy budget on computation, rather than on data conversion and input operand retrieval. Pie charts from Fig. 6 differentiate between the shares of energy spent in the *input layer* (which covers the input operand retrieval and hence constitutes the ADC, if applicable, and memory units); in the *conversion units* (which constitute the ASC or DSC); and in the *stochastic logic* (which captures the actual computation). Figures 6a, 6b, and 6c, show the shares for $Conv_{LFSR}$, $Conv_{MTJ}$, and StochMem separately (Section 3.1). As the charts reveal, share of *stochastic logic* (*conversion units*) increases (decreases) from 31.2% (64.4%) to 53.0% (37.8%) and to 60.1% (22.1%), as we move from $Conv_{LFSR}$ to $Conv_{MTJ}$ and to StochMem respectively. StochMem represents the most energy efficient design, featuring the lowest (highest) energy share for data conversion (computation), when compared to $Conv_{LFSR}$ and $Conv_{MTJ}$.

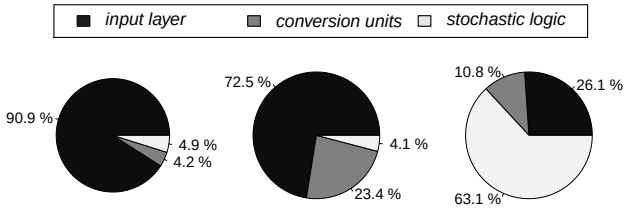
4.3 Reduction in Area

In this section, we evaluate the area cost of each alternative. Since tailoring ADC and DAC units to each application was out

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/LCA.2018.2804926, IEEE Computer Architecture Letters

TABLE 2: Area in μm^2.

Apps	Logic	$Conv_{LFSR}$				$Conv_{MTJ}$					*StochMem*		
		Memory	ADC	DSC	Total	Memory	ADC	DAC	ASC	Total	Memory	ASC	Total
Robert	339	21	50000	1450	51810	21	50000	16000	75	66435	183	75	597
Median	5382	38		2900	58320	38			150	71570	336	150	5868
Frame	457	17		772	51246	17			60	66534	153	60	670
Gamma	76	35		1156	51267	35			120	66231	306	120	502
KDE	8691	122		6166	64979	122			630	75443	1071	630	10392



(a) $Conv_{LFSR}$ (b) $Conv_{MTJ}$ (c) StochMem

Fig. 7: Pie-charts demonstrating share of hardware cost (in terms of area) across different units.

of the scope of this study, for the baselines (i.e., $Conv_{LFSR}$ and $Conv_{MTJ}$) we deploy an ADC and a DAC unit of minimal area (which represents the hypothetical best-case in terms of area cost), even if these units fail short of providing the required precision. Accordingly, if we were to incorporate realistic ADC or DAC units (which would likely incur a much higher area overhead), StochMem (which does not employ any ADC or DAC) would have shown even larger area savings in comparison to the baseline.

Table 2 summarizes the area cost for the evaluated stochastic designs (columns) for the stochastic benchmark applications (rows). While $Conv_{MTJ}$ consumes notably less energy than $Conv_{LFSR}$ (Section 4.2), it requires an extra DAC which increases the area overhead (with respect to $Conv_{LFSR}$) by 20.0% on average. On the other hand, StochMem can cut the area cost significantly, by about 93.7% (with respect to $Conv_{LFSR}$) on average, by eliminating the need for costly conversion units. Only StochMem can deliver area and energy benefits at the same time.

Fig. 7 depicts a detailed break-down of area consumption among different units. Similar to Fig. 6, pie charts from Fig. 7 differentiate between the shares of area in the *input layer*, *conversion units*, and *stochastic logic*, respectively. Only 4.9% of the area in $Conv_{LFSR}$ goes to the *stochastic logic*, while the *input layer* consumes 90.9%. *Stochastic logic* in $Conv_{MTJ}$ has even a smaller share of area (4.1%) when compared to $Conv_{LFSR}$. On the other hand, in StochMem, 63.1% of the area goes to *stochastic logic*; only 10.8%, to *conversion units*.

Data conversion in conventional SC systems necessitates high-overhead units such as LFSRs+comparators, ADCs, or DACs. StochMem-like SC systems, on the other hand, can eliminate or replace these units with lighter-weight counterparts leading to substantial energy and area savings.

5 Conclusion

A challenging artifact of modern technology scaling is growing uncertainty in design parameters, and therefore, in design functionality. This renders stochastic computing (SC) a particularly promising paradigm, which represents and processes information as quantized probabilities. Numerous stochastic computing proposals from 1960s onwards, however, focus on stochastic logic (mainly arithmetic), neglecting memory. Unfortunately, deploying conventional (digital) memory in a stochastic system is particularly inefficient due to the difference in data representations, which can easily incur a significant data conversion overhead.

In this study, we rethink the memory system design for stochastic computing to minimize the data conversion overhead, which can reach 80% of overall hardware cost, considering image processing as a case study. Analog memory is particularly promising due to seamless conversion options between analog and stochastic data representations, despite the potential loss in data accuracy which stochastic logic can easily mask due to its implicit fault tolerance. We thus evaluate analog memory for seamless SC, using a representative stochastic near-sensor image processing system as a case study. We demonstrate how such a system can reduce energy consumption and area cost by up to 52.8% and 93.7%, while keeping the accuracy loss as incurred by analog memory below 0.7%.

References

[1] I. Akturk, K. Khatamifard, and U. R. Karpuzcu. On Quantification of Accuracy Loss in Approximate Computing. In *12th Annual Workshop on Duplicating, Deconstructing and Debunking (WDDD)*, June 2015.

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

[3] A. Alaghi, C. Li, and J. Hayes. Stochastic circuits for real-time image-processing applications. In *Design Automation Conference (DAC), 2013 50th ACM / EDAC / IEEE*, pages 1–6, May 2013.

[4] A. Alaghi, W. Qian, and J. P. Hayes. The promise and challenge of stochastic computing. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2017.

[5] D. Fick, G. Kim, A. Wang, D. Blaauw, and D. Sylvester. Mixed-signal stochastic computation demonstrated in an image sensor with integrated 2d edge detection and noise filtering. In *Proceedings of the IEEE 2014 Custom Integrated Circuits Conference*, pages 1–4, Sept 2014.

[6] K. Kim, J. Kim, J. Yu, J. Seo, J. Lee, and K. Choi. Dynamic energy-accuracy trade-off using stochastic computing in deep neural networks. In *Proceedings of the 53rd Annual Design Automation Conference*, DAC '16, pages 124:1–124:6, New York, NY, USA, 2016. ACM.

[7] L. Kull, T. Toifl, M. Schmatz, P. A. Francese, C. Menolfi, M. Braendli, M. Kossel, T. Morf, T. Meyer Anderson, and Y. Leblebici. A 90GS/s 8b 667mW 64x Interleaved SAR ADC in 32nm Digital SOI CMOS. In *Proceedings of the 2014 ISSCC*, 2014.

[8] P. Li and D. Lilja. A low power fault-tolerance architecture for the kernel density estimation based image segmentation algorithm. In *Application-Specific Systems, Architectures and Processors (ASAP), 2011 IEEE International Conference on*, pages 161–168, Sept 2011.

[9] P. Li, D. Lilja, W. Qian, K. Bazargan, and M. Riedel. Computation on stochastic bit streams digital image processing case studies. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 22(3):449–462, March 2014.

[10] R. LiKamWa, Y. Hou, J. Gao, M. Polansky, and L. Zhong. Redeye: Analog convnet image sensor architecture for continuous mobile vision. In *Proceedings of the 43rd International Symposium on Computer Architecture*, ISCA '16, pages 255–266, Piscataway, NJ, USA, 2016. IEEE Press.

[11] W. T. Lin and T. H. Kuo. A 12b 1.6gs/s 40mw dac in 40nm cmos with 70db sfdr over entire nyquist bandwidth. In *2013 IEEE International Solid-State Circuits Conference Digest of Technical Papers*, pages 474–475, Feb 2013.

[12] J. Lu, S. Young, I. Arel, and J. Holleman. A 1 tops/w analog deep machine-learning engine with floating-gate storage in 0.13 μm cmos. *IEEE Journal of Solid-State Circuits*, 50(1):270–281, 2015.

[13] B. Murmann. "ADC Performance Survey 1997-2016," [online]. Available: http://web.stanford.edu/ murmann/adcsurvey.html, 2016.

[14] M. H. Najafi, P. Li, D. J. Lilja, W. Qian, K. Bazargan, and M. Riedel. A reconfigurable architecture with sequential logic-based stochastic computing. *J. Emerg. Technol. Comput. Syst.*, 13(4):57:1–57:28, June 2017.

[15] N. Onizawa, D. Katagiri, W. J. Gross, and T. Hanyu. Analog-to-stochastic converter using magnetic tunnel junction devices for vision chips. *IEEE Transactions on Nanotechnology*, PP(99):1–1, 2015.

[16] W. Qian, X. Li, M. Riedel, K. Bazargan, and D. Lilja. An architecture for fault-tolerant computation with stochastic logic. *Computers, IEEE Transactions on*, 60(1):93–105, Jan 2011.