Mixed-Signal Stochastic Computation Demonstrated in an Image Sensor with Integrated 2D Edge Detection and Noise Filtering

David Fick, Gyouho Kim, Allan Wang, David Blaauw, Dennis Sylvester

University of Michigan, Ann Arbor, MI 48109, USA

Abstract — **In this work we describe mixed-signal stochastic computing (MSSC) and demonstrate how it can be used to efficiently integrate computation into a signal path before data conversion. MSSC performs computation directly on the analog values output by sensors, which enables MSSC to combine the area efficiency of traditional stochastic computing with the information density and performance of analog computation. To demonstrate this technology we integrated MSSC between pixel bitlines and the ADC in an image sensor, enabling *in situ* latency-free edge detection and noise filtering. The MSSC implementation is found to be 2.75× lower power than a traditional digital synthesis implementation while simultaneously requiring 5× lower area.**

Index Terms — **Stochastic computing, image sensor, mixed-signal, integrated computation, sensors, edge detection, noise filtering.**

I. INTRODUCTION

Digital stochastic computing (DSC) originated in the 1960s when transistors were bulky and expensive [1-2]. DSC aimed to reduce the number of transistors needed to perform an operation by representing numbers as one-bit probabilistic streams (*e.g.*, a value of 0.75 is a random stream with 75% ones and 25% zeros), thereby allowing complex operations to be performed by small gates. As shown in Fig. 1, multiplication can be performed by a single AND gate and a weighted average can be performed with a multiplexer. Complex functions such as division or square root can be computed using ~12 gates, including multiplexers and registers [2]. Entire data processors were implemented using DSC [3], but interest in the technology diminished as transistors became cheaper. DSC has significant limitations including the relatively costly conversion of data from binary format to stochastic streams, which requires an independent randomness source for each interacting bit of data, and a linear increase of work needed to achieve increased accuracy (2× accuracy requires 2× as much work).

Mixed-signal stochastic computing (MSSC) mitigates some of these issues by operating on analog values from sensor data rather than stochastic ones/zeros. Working with analog values eliminates the need to convert to a stochastic stream, and by working on sensor data the required accuracy is limited to feasible levels (*e.g.*, 8-16 bits instead of 32 or 64 bits). As shown in Fig. 2, each element of the stochastic stream is a full analog value, which can be stored, multiplexed, or operated on with resistors or capacitors. The sensor data is first sampled and held, so that the sample can be stochastically used multiple times throughout a calculation. Over the calculation, the same samples are stochastically selected and the MSSC ADC aggregates the result. Once the calculation is complete, the S&H circuits obtain new samples and the calculation starts again. By using an analog samples, rather than binary, it is possible for the operation to converge more quickly, depending on the computation being performed.

Since MSSC performs stochastic computation directly on the analog values output by sensors, MSSC has several key advantages over traditional DSC: 1) it has more information to work with, 2) it avoids costly stochastic data conversion, and 3) it provides a more natural coupling of noisy-data with noisy-computation. Additionally, in contrast with analog computation, MSSC operates on discrete voltage samples and is therefore able to reduce or eliminate tail currents. In this technique, the voltage samples are stochastically mixed, transformed, and sampled to perform calculations on sensor data in ways similar to DSC, but with greater efficiency and performance.

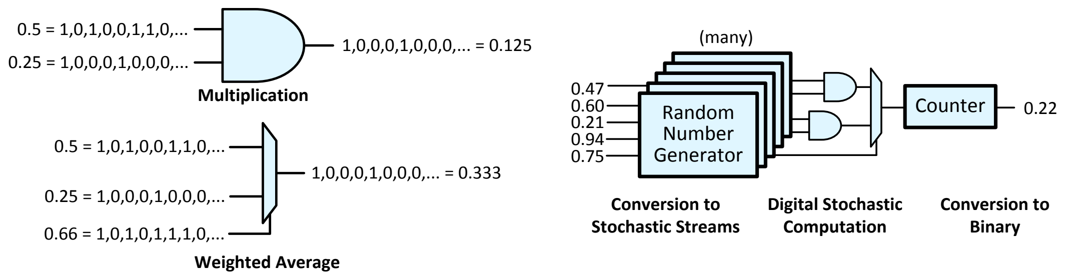


Figure 1: Digital Stochastic Computing (DSC)

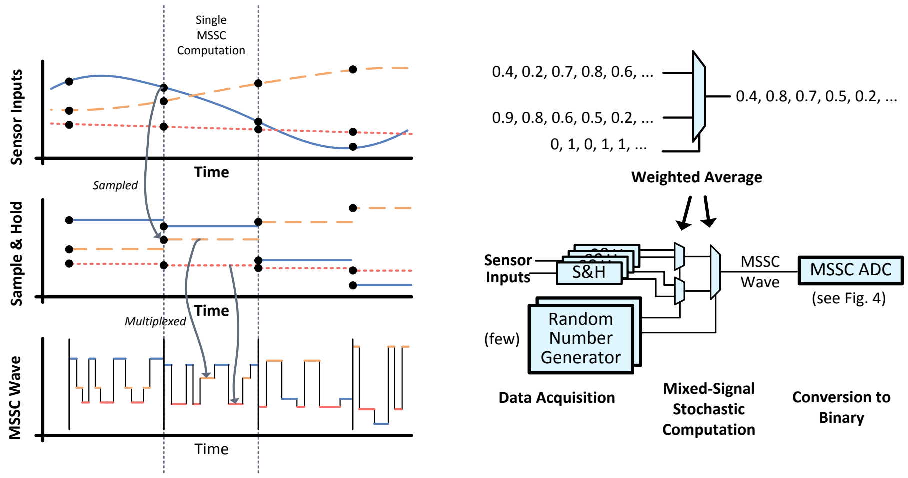


Figure 2: Mixed-Signal Stochastic Computing (MSSC)

II. System Description

To demonstrate MSSC, an image sensor is implemented with MSSC integrated into the pixel readout circuitry (Fig. 3). Each time a row of pixels is read from the imager array, a row of calculated pixel values will be produced. Edge detection and noise filtering are performed by applying pixel-level windowing operations, where each output value is the weighted sum of the original pixel data and its 8 neighbors (the weights to these window operations are shown in Fig. 4). The weighted-sum is performed through multiplexing.

The data flow is shown in Fig. 3. First, the row of pixel voltages is stored into a bank of sample and hold (S&H) circuits. The bank of S&H circuits holds the voltages for the 5 most recently read rows, providing sufficient data to perform both noise filtering and edge detection. Second, the noise filtering operation is performed, creating 3 rows of filtered pixel data. Each output pixel in the noise filtering operation is created by stochastically selecting 1-of-9 pixels from the S&H bank, with probabilities from the currently selected setting). Third, one row of edge detection pixels are created by stochastically selecting 1-of-9 pixels from the filtered pixel data. The negative values in the edge detection operation are created in the next step. Fourth, a comparator compares the pixel value to a uniform random signal, which stochastically converts the pixel value to a 1/0 with a probability proportional to the analog value. The negative sign from the third step is created by inverting this result. Finally, the stream of 1s/0s is aggregated with a counter, which will have a count proportional to the average of the probabilities entering the fourth step. The last two steps are similar to a strategy proposed in [4] for converting a static analog voltage to a DSC stream and then reading it again.

For ease of design, the MSSC component was designed and implemented as a single column bit-slice that was tiled 96 times. As shown in Fig. 3, each slice contains 5 S&H elements, 3 noise filter voltage switch networks (VSNs), 1 edge detection VSN, a comparator, a ripple counter, and a scan chain for data readout. The control signals pass through each bit-slice while samples are passed to/from neighboring bit-slices. The uniform-random signal for comparing with the MSSC samples was implemented as a

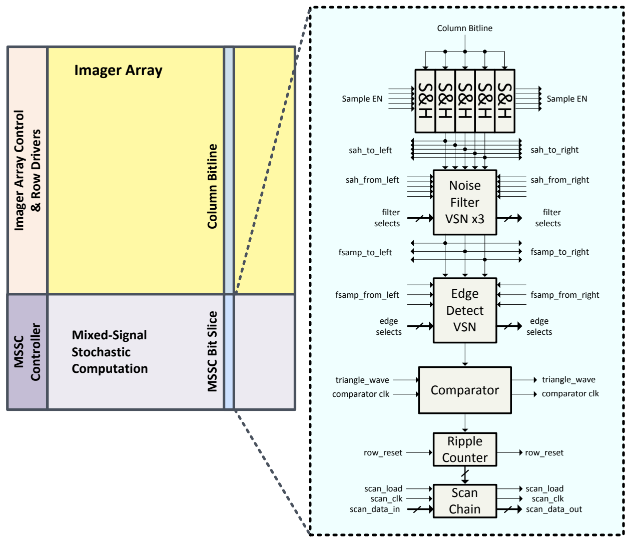


Figure 3: System Block Diagram

1	2	1
2	4	2
1	2	1

σ = 0.85

1	4	1
4	16	4
1	4	1

σ = 0.60

1	8	1
8	64	8
1	8	1

σ = 0.49

-1	2	-1
-1	2	-1
-1	2	-1

Vertical

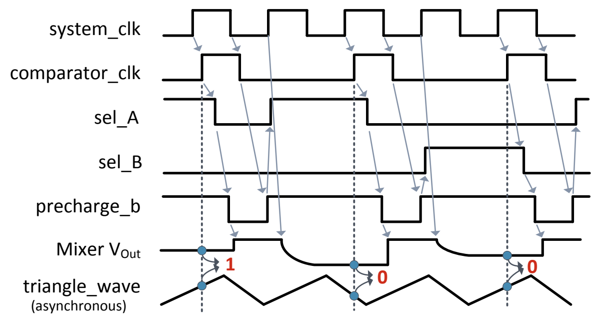
-1	-1	-1
2	2	2
-1	-1	-1

Horizontal

-2	1	-2
1	4	1
-2	1	-2

Vert. + Hor.

Figure 4: Window Operation Weights for Noise Filtering (Left) and Edge Detection (Right)



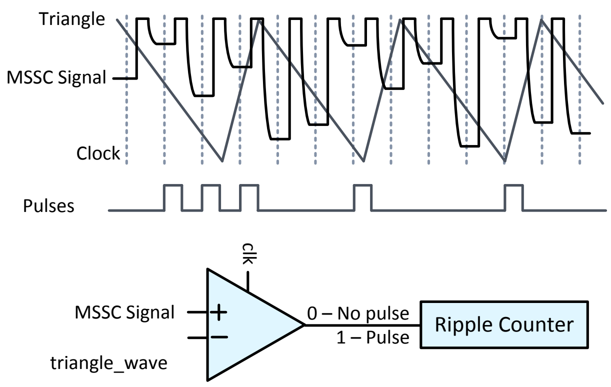


Figure 5: Waveforms for Voltage-Switch Network and ADC (Left), Waveforms for MSSC-ADC (Top-Right), and Mixed-Signal Stochastic ADC (Bottom-Right)

triangle wave. Since pixel calculation for each bit-slice is independent, the same pseudo-random select signals and the same uniform-random waveform can be shared among all bit-slices.

Fig. 6 shows the S&H circuit, VSN circuit, and triangle wave generator. The S&H cell contains a source follower with feedback to the access transistor. The feedback limits access transistor V_{DS} and allows sufficient hold time for five rows of MSSC computation (> 5ms at FF/80°C based on Monte-Carlo simulations). The input from the column bitline is 0.0−0.5V (1.2V process), which is shifted higher by the source follower. A source follower header is not required since mixer inputs are precharged to V_{DD} each cycle. This strategy enables the use of a source follower without any tail current. Similarly, the source follower can be sized for speed/variation without regard for power since it is effectively power-gated when not in use.

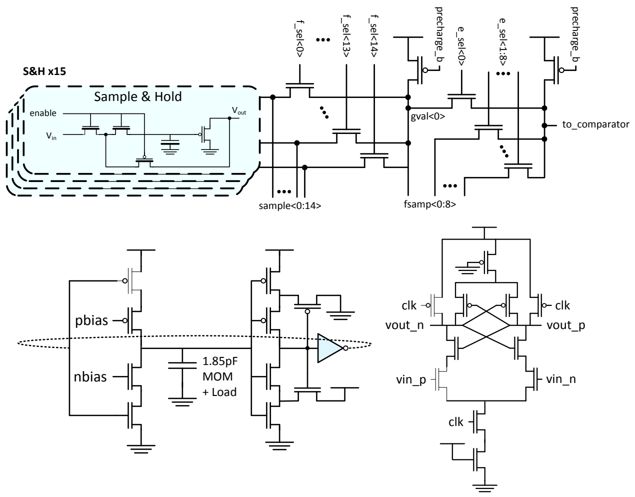


Figure 6: Sample and Hold (Top-Left), Voltage Switch Network (Top-Right), Triangle Wave Generator (Bottom-Left), and Comparator (Bot-Right)

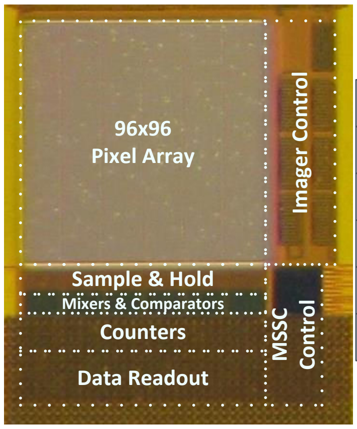
Fig. 5 depicts the internal operation of the voltage switch network and the comparator. The VSN consists of a pass transistor multiplexer and has an output V_{out} that is precharged to V_{DD}. After precharge a particular S&H unit is selected, which then discharges V_{out} to its sampled analog value. Care must be taken to ensure the source follower limits the data swing instead of the pass transistors. After sufficient settling time, the comparator is triggered and the selected value is compared to the uniform-random waveform, outputting a pulse or non-pulse to the ripple counter.

The MSSC control unit generates non-overlapping pulses for the enable signals of the S&H units, the precharge and select bits for the mixers, and the comparator clocks. The control unit can select the noise filter sigma, type of edge detection, or bypass one or both of these calculations. It also controls how many cycles are run for each row of computation, which dictates accuracy and energy consumption. LFSRs are used to generate pseudo randomness for selection, which is shared across all of the bit slices so that only two LFSRs are needed in total.

III. Measurement Results

The image sensor was fabricated in 130 nm CMOS (Fig. 6) with a total MSSC circuit area of 0.064 mm^2. A digital synthesis implementation of the same functionality (without ADC capability) was created for comparison, and occuped 0.33 mm^2 assuming 100% area utilization, or 5× larger than the MSSC implementation.

Power was measured at a frame rate of 30 fps and sample count of 1000, translating to 2.88M samples/sec/column, which can be achieved with near-threshold operation. The MSSC controller and counters operate at 0.6 V, the mixers and comparators run at 1.2 V, and the S&H enables operate at 1.5 V. Level converters are included between the controller and other units to facilitate mixed-V_{DD} operation.



Process	130nm CMOS
Array Size	96×96
MSSC Area	0.064mm^2
Image Sensor Area	0.24mm^2
MSSC Power (30FPS)	20.7μW
Imager Power (30FPS)	1.0μW

Figure 6: Die Micrograph

The measured power of the MSSC circuitry is 20.7 μW. The digital synthesis implementation is estimated to consume a comparable 48.4 μW, and a traditional single-slope ADC in a similar design [5] consumes 8.6 μW for the same frame rate. Combined, the traditional ADC with digital-synthesis approach consumes 57 μW, or 2.75× that of MSSC.

Recorded images from the system are shown in Figure 7. On the left, the effect of increasing the number of samples from 250 to 4000 is shown, as is the effect of increasing the noise filtering on an image taken of a shirt and tie. On the right, sixteen mode combinations are demonstrated on a control pattern: the combinations of four noise filtering levels and four edge detection settings.

IV. Conclusion

In this work we described mixed-signal stochastic computing (MSSC) and a demonstration system which included MSSC within an image sensor with integrated edge detection and noise filtering. The MSSC implementation was shown to be energy efficient, with power consumption 2.75× less than a digital synthesis implementation, and area efficient, with an area 5× less than a digital synthesis implementation.

Acknowledgement

This project was supported by the STARnet C-SPIN and SONIC centers and the U.S. Army Research Laboratory.

References

[1] B. Gaines. Advances in Information Systems Science, 1969, pp. 37-172.

[2] P. Mars et al., "Implementation of linear programming with a digital stochastic computer," Electronics Letters, Sep. 1976, pp. 516-517.

[3] P.J . Gawthrop, "Stochastic and Deterministic Averaging Processors," Control Theory and Applications, IEEE Proceedings D , vol.129, no.5, pp.212, Sept 1982

[4] John Esch, Rascel: a Programmable Analog Computer Based on a Regular Array of Stochastic Computing Element Logic. Ph.D. Dissertation. 1969, UIUC, Champaign, IL.

[5] Y. Lee et al., "A Modular 1mm^3 Die-Stacked Sensing Platform with Optical Communication and Multi-Modal Energy Harvesting," ISSCC, 2012.

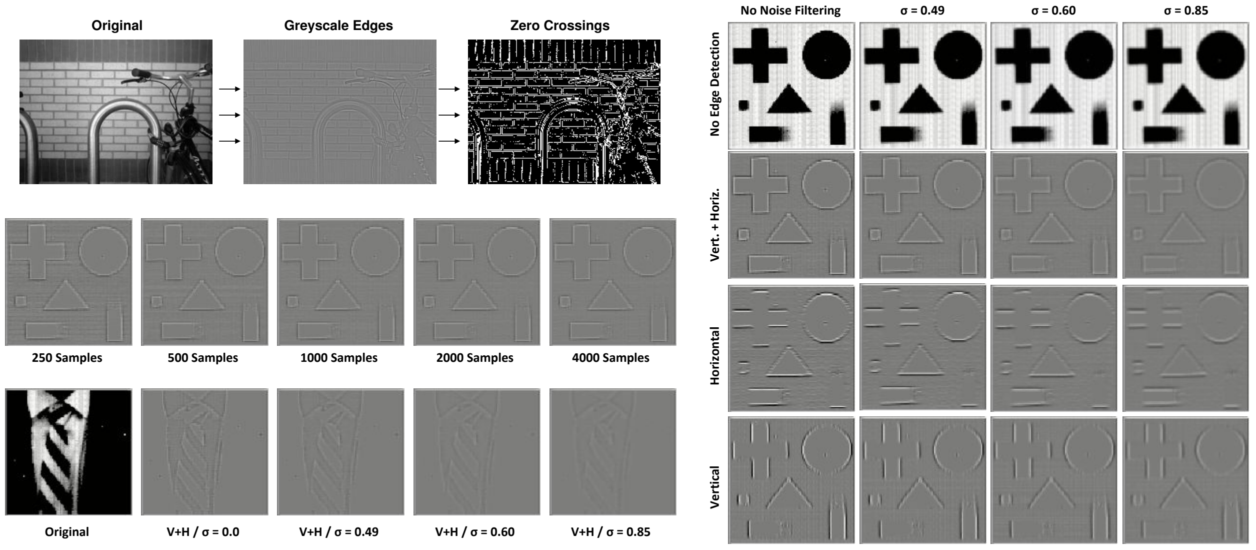


Figure 7: Example computation flow for the edge detection flow (left-top). The effects of using more/fewer samples per calculation (left-middle). Example edge detection photographs with changing noise filtering (left-bottom). Example outputs for fixed-pattern input under various settings (right).