End-to-End Stochastic Computing

Mikko Lipasti
Department of Electrical and
Computer Engineering
University of Wisconsin-Madison
Madison, WI
Email: mikko@engr.wisc.edu

Carly Schulz
Department of Electrical and
Computer Engineering
University of Wisconsin-Madison
Madison, WI
Email: cschulz8@wisc.edu

Abstract—**Embedded systems that require high performance hardware for signal processing, optimization, and control have historically relied on optimized fixed-point algorithms deployed on low-power microprocessors and/or digital signal processors. However, the sensing and actuation interfaces in such systems increasingly rely on oversampled, single-bit sigma-delta-modulated (SDM) representations of data inputs and control outputs. Conventional computing substrates requires conversion both on the input side and on the output side in order to effectively interface with physical systems that utilize SDM representations. This position paper argues that embedded systems should be designed to sense, process, compute, and actuate using an approach called stochastic computing, which operates directly on the oversampled SDM representation that is a natural fit for interfacing with physical systems. Stochastic computing offers numerous advantages in terms of datapath simplicity, robustness, error tolerance, and support for variable precision, all of which make it an appealing approach for these embedded applications. On the other hand, stochastic computing has a number of limitations that make design of entire systems based on this approach quite challenging. This paper outlines some of those challenges and details our experiences with building such a prototype system.**

I. Introduction

Presently, digital binary representation is the well-established standard for computation. As technology continues to shrink in size, the circuits become more error-prone and unpredictable [1]. With the limits of the deterministic binary computing, these unpredictable elements have to be guarded against and often end up having a high hardware cost [2].

An alternative paradigm that was proposed in the 1960s is stochastic computing. Stochastic computing is based on probabilities and inherent uncertainties and works well with the same uncertainties that make deterministic computing more difficult [1]. Interest in stochastic computing was driven by contemporary concerns about device and circuit reliability, and was in fact rooted in Von Neumann's even earlier work on probabilistic calculus. As device technology matured, these concerns were successfully addressed by improvements in the reliability of the underlying devices and circuits used as building blocks in computing systems, and so interest in stochastic computing waned. More recently, similar reliability concerns have been raised regarding emerging, post-CMOS device technologies, and there has been a resurgence of interest in stochastic computing, including a number of proposals for using it to replace conventional, deterministic computing models for a broad array of applications.

Stochastic computing has a number of advantages as well as drawbacks, and it is unlikely that it will serve as a wholesale replacement for conventional computing. However, there are domains where it is a compelling and very natural fit. This paper focuses on a class of systems where the most natural representation for both input and output data is one that fits well within the stochastic computing paradigm. Specifically, we are interested in applying stochastic computing to replace a conventional data path in systems that required high performance hardware for tasks including signal processing, task optimization, and control, but that also interface to the physical world using data representations that appear very similar to the probabilistic bit streams utilized by stochastic computing.

Systems that interact with the physical world must both sense their environment and control actuators that provide them with the ability to manipulate physical elements. On the sensing side, oversampling, single-bit sigma-delta-modulated (SDM) representations have become increasingly popular due to their robust performance and ease of implementation. While they require higher data rates for comparable accuracy, the physical design simplicity of both the ADC (single-bit SDM) over more complex multi-bit ADCs, as well as the ease of system design (single-bit, serial interconnects like SPI) has made them the standard choice for applications like digital audio processing. Similarly, on the control side, pulse-width modulation (PWM) has largely replaced earlier, more complex multi-bit or level-based digital-to-analog (DAC) approaches. In the PWM approach, the control output also has only one bit of DAC precision (it is either on or off), but the width of the pulses is modulated to provide a wider dynamic range. Both SDM and PWM represenations are a natural fit for stochastic computing, since signals in these formats can be easily consumed as inputs or generated as outputs by SC datapaths.

This paper argues that the datapath simplicity, robustness, error tolerance, and support for variable datapath precision provided by the stochastic computing approach makes it a natural fit for such applications. We first discuss the history of stochastic computing and the tradeoffs associated with using it, and then describe a prototype system—a digital audio equalizer—that we have implemented, and conclude the paper with an assessment of the advantages of this paper along with some thoughts on future work.

II. Background

A stochastic representation of a number is a bit stream of ones and zeros where the desired transmission value is the probability of whether a given bit is a one [3]. This method of representation is inherently fault tolerant and provides an opportunity as we reach the limits of Moores law and explore new technologies [4]. The field of stochastic computing has gained attention recently for its applications in LDPC decoding, image processing, filters, and fault tree analysis [1].

A. Probabilistic Calculus

Although he is credited with the traditional computing model and computer architecture as we know them today, Von Neumann also began the gestation of stochastic computing with his lecture on performing operations on probabilistic values. He detailed his probabilistic calculus methods using stochastic bit streams and developed automata to illustrate their function. Using probabilistic methods, multiplication becomes trivial to implement at the gate level and is simply calculated by feeding both inputs to an AND gate. As long as the inputs are not correlated with each other which is handled by their random nature, then the result of the gate will be, with high probability, the result of the two values multiplied together [5]. The development and proof of such calculus shows that exploiting stochastic numbers can result in drastically simpler logic to perform operations such as multiplication, division, etc.

Stochastic computing computes Bernstein polynomials with a block of adders and a block of multiplexers. Other functions beyond polynomial functions can be mapped as approximations of a Bernstein polynomial. Stochastic functions can implement any finite interval function that maps to another finite interval [4].

B. Benefits Over Deterministic Computing

1) Error Tolerant: Stochastic computing is quite error tolerant and is well adapted to neural networks. It has several applications in neural networks and image processing due to its parallel nature [3]. Since the bits are randomly set or not set within a predetermined stream of bits, stochastic computing is rather tolerant of soft errors, a flip of a bit, than traditional computing where specific bits have much larger importance [4]. A more error tolerant method of computing requires less hardware resources to be placed in error correcting or guarding techniques. Stochastic bit encodings have been shown to generate correct outputs in applications such as image processing even with soft error rates as high as 30% [6]. In stochastic computing, the hardware is designed so as to allow errors or without the over-designing mechanisms in use today and pushes the handling of errors up to the software level [4].

2) Circuit Size: Using Stochastic numbers allows the implementation of more complicated computations with very simple logic and low gate cost as a result [7]. In addition to the low gate cost multiplication and division, complex computations such as Taylor expansions of the exponential function and square-root function can be implemented with a small number of gates under this computing model . Traditional deterministic binary encoding requires much more complex logic elements or relies on software to compute these functions[4].

3) Timing: Along with the decreased hardware cost, the size often results in a shorter critical path which allows a stochastic circuit to operate at higher clock frequencies [6]. This could result in a faster overall computation time, depending on the length of the bit stream that is being operated on.

C. Traditional Challenges

There are some obstacles that prevented stochastic computing from proliferating. Some issues

1) Representation Cost: A large one is that there is a higher cost in terms of bits required to represent the number with a certain degree of precision when compared to traditional digital representations. That issue creates another, the cost of multiplication in terms of gate invocations can be quite high for stochastic representations as the number of bits increases [8]. Longer streams of bits that are required for stochastic computing may end up consuming more power than expected from a small and simple circuit. When using stochastic computing for issues with low precision requirements, these representation costs can be somewhat mitigated by the decrease in the length of the bit stream [7]

2) Energy Consumption: Generating stochastic numbers from the deterministic binary numbers consumes more energy than leaving it in the deterministic form [7]. This is not an issue when using SDM signals in the end-to-end stochastic computing method. What does become an issue is the potentially higher energy cost from computing with long bit streams of stochastic numbers [7].This is slightly alleviated by pushing the burden of error correcting beyond hardware [4]. Energy efficiency of the system is improved when the design can be optimized for a non-zero error rate [2].

3) Timing: In serial encoding, computation time is a significant factor. (If a parallel implementation is being used, then area is the challenge here.) The length of the bit streams needed to preserve precision leads to longer computation times, even if the clock cycles can be shorter than in traditional processors [4]. Stochastic computing cannot be pipelined which makes high throughput difficult to attain [1].

4) Errors: There is error introduced through mapping the function to a Bernstein approximation in addition to the normal quantization errors that one would expect in hardware [4]. Noise also contributes to the errors in this method of computing. Since stochastic numbers are restricted to the [0,1] range, as the computation progresses the power of the scaled stochastic number decreases which leads to the SNR decreasing as well. It should be noted that in the case of

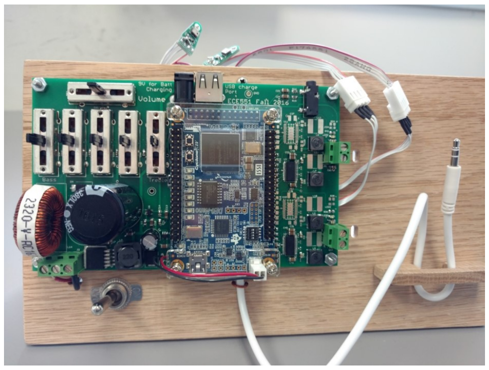


Fig. 1: Prototype PDM Equalizer Board.

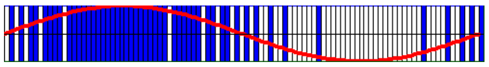
transient noise, stochastic computing tolerates the noise better than binary computing [1].

These issues try to imply that stochastic representations may be more costly than they are worth. We argue that the benefit of eliminating conversion circuits makes stochastic computing a competitive option in systems where both the input and output are encoded in a stochastic way, such as SDM signals.

III. Implementation

This section describes a system prototype that we have built that utilizes principles from stochastic computing to implement a system that processes audio signals end to end using a pulse-density-modulated representation that is a natural fit for stochastic arithmetic. The prototype system board is illustrated in Figure 1. The digital logic that utilizes stochastic computing principles is implemented using a DE0-nano development board from Terasic[9]. This board contains an Altera Cyclone IV FPGA, as well as interface circuitry to the custom daughter-card, which contains a stereo class-D amplifier, connections for two PDM digital microphones (shown attached to ribbon cables behind the board), a 3.5mm audio input with PDM ADC, as well as a set of potentiometer sliders used for controlling volume as well as five bands of frequency components to realize an audio equalizer.

Pulse-density modulation is an approach for encoding time-varying signals that utilizes an oversampled, single bit to represent complex waveforms[10]. Figure 2 shows two examples of sine waves at different frequencies encoded in PDM format. PDM format has become very popular as a representation in the audio domain, and is very commonly used in modern cellular phones, where the on-board PDM microphones are attached to the rest of the system with a low-cost, single-wire PDM bus that it typically over-sampled at around 3MHz. The PDM microphones contain a low-cost but high precision sigma-delta modulator (SDM) [11] that converts the analog audio signal captured by the MEMS microphone into a series of PDM pulses. Our prototype includes two such PDM microphones connected to the system, but also provides a 3.5mm audio jack connected to a low-cost PDM ADC that converts standard analog audio from any sound device (e.g. a portable MP3 player) to an oversampled PDM stream. The main advantage of using PDM input is the simple interface circuitry and relatively low cost and ease of design of sigma-delta modulators that nevertheless provide very high audio fidelity[11].


(a) An example of PDM of 100 samples of one period of a sine wave. 1s represented by blue, 0s represented by white, overlaid with the sine wave.

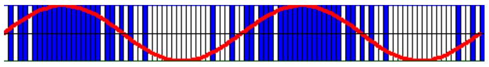

(b) A second example of PDM of 100 samples of two periods of a sine wave of twice the frequency

Fig. 2: Examples of pulse-density modulation (aka SDM). From [10]

Class-D amplifiers are switching amplifiers that do not attempt to utilize the linear region of a transistor for gain, as in conventional amplifiers, but instead rapidly switch the gain device (a MOSFET) based on a train of pulses which represents an audio signal encoded in the above PDM format[12]. Class-D amplifiers are extremely efficient and relatively easy to design, since they do not suffer from the design complexities and vulnerabilities of classic amplifiers that were extremely sensitive to the transfer characteristics of the amplifying MOSFET.

In summary, we have built a protyping platform that takes as input audio signals in PDM format, operates on them while maintaining that representation, and drives output circuitry (class D amplifiers) that then drive actuators (speakers) that create a physical audio signal. Within this context, the advantages of designing circuitry that maintains the original input and output data representation are clear.

Figure 3 shows a block diagram of the digital circuitry we have implemented on the Altera FPGA in our prototype system. As shown, it supports two identical audio channels (left and right). Each channel provides volume control followed by five bands of equalization targeted at frequencies ranging from very low (bass) to very high (treble), and allowing the user to attenuate or amplify these frequency bands to shape the final sound.

In a conventional digital signal processing-based design, the audio inputs would first be converted a pulse-code modulated PCM format with multiple bits of precisions for each sample (e.g. 16-bit samples at 44KHz per channel for CD-quality audio). This conversion task requires a fairly heavy-weight cascaded integrator comb (CIC) filter to reconstruct a lower-sample-rate multi-bit signal from the oversampled PDM representation[13]. Following this conversion, standard digital filtering techniques would be used to implement each of the filtering stages shown, typically using cascaded biquad filters [14]. Such an implementation would require high-performance digital signal processor cores capable of executing high-precision (at least 16x16=32, but preferably higher) multiply-accumulate operations at a fast rate. Finally, after the final

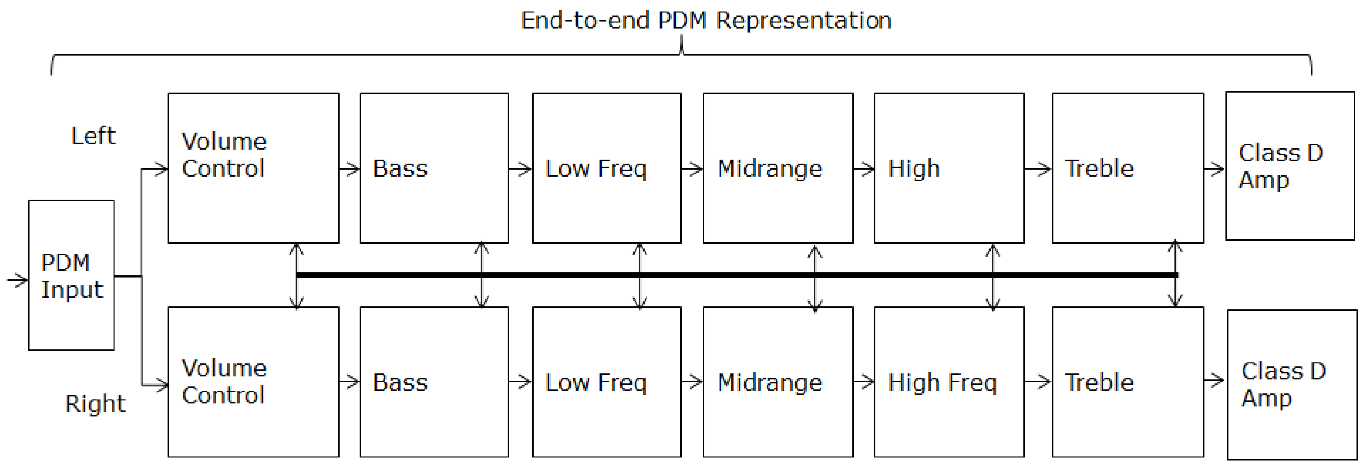


Fig. 3: PDM Equalizer Block Diagram.

equalizer filter stage, the conventional design would require a multi-bit digital-to-analog converter (DAC) in order to supply an analog output to a conventional amplifier design. In our prototype, this analog output would then be converted back to a PDM bitstream to drive the gain MOSFET inside the class D amplifier, leading to an additional level of inefficiency and added noise from the redundant DAC-ADC stages.

Instead, we have implemented each filtering stage with a design that is functionally equivalent to a biquad filter, but utilizes principles of stochastic computing to dramatically simplify the arithmetic datapath. For example, since the PDM input stream represents values with a stream of 0/1 bits that are numerically weighted as -1/+1, all digital multipliers in the design have been replaced with simple, reduced-precision adders. Furthermore, since even the oversampled data rate of 3MHz is still very low compared to the speed of modern digital circuitry, even on FPGAs, these adders need not use high-speed and power-hungry circuit topologies. In the end, the datapath for realizing these filters is substantially simpler and more energy-efficient than a conventional digital baseline.

Finally, since the filtering circuitry operates directly on PDM format data, no input conversion or output conversion is needed. Input from a PDM microphone is consumed directly, while the final filter stage outputs can be used to directly drive the gain MOSFETs in the class D amplifier.

IV. Evaluation

We have implemented the design from the preceding section and synthesized it to target the Altera Cyclone IV FPGA on our DE0-nano development board [9]. To provide a reasonable baseline of comparison, we have also designed a conventional PCM-based design that can be synthesized for the same FPGA. There are three key differences with respect to the design shown in Figure 3:

- The PDM digital input is downsampled (decimated) and converted to 16-bit PCM format.
- The signal processing components (biquad filters) operate on the PCM data.
- The outputs from the rightmost filters are converted from PCM back to PDM via interpolation and sigma-delta modulation.

These changes introduce additional overhead for the format conversion, but also reduce the operating frequency of the signal processing core substantially. We elaborate on these changes in the following paragraphs.

A. *PDM to PCM Conversion*

In our design, the PDM input arrives at a rate of 3.125MHz per channel. All the filters in the system (both PDM-based and PCM-based) are designed with a sampling frequency of 24.414KHz, corresponding to the input PDM sample rate divided by 128 [1]. In order to maintain the same filter coefficients for our biquad filters, we need to downsample (by 128x) the 3.125MHz PDM input to a 24.414KHz rate. We used a low-cost, 3-stage cascaded integrator-comb (CIC) Hogenauer filter [15] modeled after the scheme described in a blog post from Cheshire Engineering [16], but implemented in Verilog rather than C.

B. *Biquad filters*

The biquad filters in our design assume a sampling frequency of 24.414KHz, and are parameterized based on the equations found here [17]. The bass and treble filters are configured as low shelf and high shelf, while the three mid-range filters are configured as peak filters. Each filter can be parameterized to provide from -8dB to +8dB gain for each band. In the PDM design, each biquad simply integrates the five biquad coefficients, stored as 16-bit fixed–point values, based on the five binary PDM inputs (the current input and the last two inputs, as well as the last two outputs), and then quantizes the integrated sum into a single PDM bit. The delayed inputs and outputs are buffered for 128 time steps to

[1]This sampling frequency is clearly too low for high-fidelity audio applications, but we found it acceptable for our *boombox* scenario.

realize the 24.414KHz sampling rate; this delay memory is mapped to block RAMs on our FPGA prototype.

In the PCM baseline, we simply implement these biquad filters in the conventional fashion, using fixed-point multiplication and addition. Since the performance requirement is very low at our 24.414KHz sample rate, we implement the multipliers using a multicycle accumulator-based design. To further reduce hardware cost in the baseline, we share that single multiplier to integrate all five biquad terms (i.e. each biquad evaluation consists of 5x16=80 accumulate operations). Finally, we quantize the output to 16 bits and use 16-bit registers to implement the biquad's delay buffers.

C. PCM to PDM Conversion

The biquad filter cascade generates a 16-bit filtered PCM result at a rate of 24.414KHz. The final stage in our PCM baseline must convert this to a PDM representation at 3.125MHz to drive our class-D audio amplifier. This requires interpolation by a factor of 128x, which we accomplish with a 3-stage filter cascade, where each stage uses 3rd-order CIC filters that interpolate at rates of 2x, 8x, and 8x, respectively, for a total upsampling of 128x. The final interpolated results are then fed to a second-order sigma-delta modulator.[2], which generates a PDM stream at 3.125MHz. The cascaded CIC filters in this conversion step are surprisingly expensive, as the integer datapath in each must be sized to avoid overflow, requiring an additional bit of precision for each order and each stage.

D. Area results

In order to assess the area cost of the proposed PDM design, we synthesized it in the Altera Quartus 16.1 environment. Table I reports the number of logic elements (LEs) required by the biquad filters, by the conversion logic (in the baseline design), and everything else (Misc, which includes control logic for reading equalizer settings and parameterizing the biquads accordingly, etc). The relative LE breakdown was obtained from the synthesis report (pre-place and route) and the LE counts shown are normalized to the final post-PR LE count.

As can be seen from Table I, the overhead of format conversions in this environment is substantial (39% of the total area required for the baseline PCM design). This is particularly true of the PCM to PDM conversion block, which requires wide arithmetic and delay elements to implement the multi-stage CIC filter (progressively wider in the later stages). The PDM biquad filters are about 20% larger than the PCM baseline. Since both designs are based on accumulator-style multipliers using just a single adder, this overhead is mostly caused by the PDM quantizer, which is a first-order sigma-delta modulator. This requires an additional adder/subtractor and register for carrying over quantization error, whereas the PCM quantizer simply truncates the low order bits. However, despite the modest area increase per filter, the PDM design is attractive from an efficiency perspective: the vast majority of the FPGA resources are consumed for useful work (filtering), rather than format conversion. The majority of the RAM in both designs is used by the biquad parameter lookup RAM (16kb); this is used to store the biquad parameters that are chosen for each filter based on the equalizer settings. The PDM biquads also utilize RAM blocks for their delay memories, which requires 3kb of storage across all ten biquads. This leads to an increase of 19% in overall RAM consumption, and highlights the storage inefficiency of unary representations, which is one of the major shortcomings of PDM-based processing.

It is also worth pointing out that both the baseline PCM and proposed PDM designs are first-pass efforts at building a working prototype, and there are numerous hardware optimizations that could significantly affect the area and performance of the designs. Exploration of such optimizations is left to future work.

V. Conclusions and Future Work

This paper argues that stochastic computing can form a compelling computational substrate for systems where inputs and outputs are encoded as a single-bit stream. We have built a prototype system that operates directly on such a bit stream to implement an audio equalizer. This system avoids the overheads of input and output conversion that are mandatory in conventional digital signal processing datapaths that operate on pulse-code-modulated (PCM) data, and also simplifies the datapath hardware, since arithmetic operations like multiplication can be implemented at much lower cost on serial PDM bitstreams. Overall, with respect to a conventional PCM baseline equalizer, the PDM design reduces logic area by 32%, but does increase RAM usage by 19%.

Evaluating the energy efficiency of the PDM design presents a challenge; the approximate power modeling provided by the Quartus tool relies on a single activity rate parameter and returns power results for the two designs that are 1) dominated by the static power of the chip, and 2) differ only marginally in terms of dynamic power. In future work, we will synthesize these designs with a standard-cell design flow and evaluate energy efficiency with sufficient accuracy to determine any relative benefit provided by the PDM design.

As described and implemented, the cascaded biquad approach for implementing an equalizer suffers from additive noise; that is, any noise introduced by the first filter is amplified by subsequent stages. When multiple filters are active, this can lead to very poor audio quality. To avoid this problem, an alternative architecture that operates the filters in parallel and mixes the output of each based on the equalizer settings should lead to a better result. Design of such an alternative is also left to future work.

Generalizing from this experience with a specific system, we would argue that future designs for embedded systems that match these characteristics—systems where the input and output are most naturally represented as a serial oversampled bitstream—stochastic computing forms a natural and very appealing foundation for dramatically reducing cost and complexity. We are currently working on prototyping two

[2]Again, inadequate for a high-fidelity audio solution, but adequate for a *boombox*

Category	PCM LE	PDM LE	PCM RAM	PDM RAM
PDMtoPCM	196 (5%)	0 (0%)	0 (0%)	0 (0%)
Biquads	2052 (50%)	2431 (60%)	0 (0%)	3kb (19%)
PCMtoPDM	1386 (34%)	0 (0%)	0 (0%)	0 (0%)
Misc	444 (11%)	327 (08%)	16kb (100%)	16kb (100%)
Total	4078	2758 (68%)	16kb (0%)	19kb (119%)

TABLE I: Area Cost of PDM vs. PCM Equalizer Design

such additional example systems as further evidence for this assertion. First, we are building a drone system that will rely on computer vision algorithms for object localization and optical flow to provide a guidance system for the drone. In this scenario, the visual input to the drone will be represented as serial bitstreams that represent features extracted from the drone's camera. Our current-generation drone hardware still relies on conventional pixel-based input from the camera, but in the future we expect to transition to a camera system similar to the Inilabs Digital Vision Sensor [18], which reports visual input using spike trains that strongly resemble single-bit bitstreams. Similarly, the control output from the drone's onboard guidance system is a pulse-width modulated (PWM) stream that controls the speed of each of the drone's four fan motors. Hence, this is a system where the inputs and outputs are most naturally represented as streams of single-bit binary data, so we are investigating how best to implement the entire datapath of the drone's flight controller using stochastic computing-like circuitry that operates directly on this data type. Second, we are prototyping a robotic control arm system that relies on solving the inverse kinematics problem to position the robotic arm at a target location[19]. Here again, the visual input to the robotic controller will come from a camera that ideally utilizes the Inilabs sensor[18], while the output is once again a PWM stream that drives rotational actuation in each of the robot arm's joints. Solving the inverse kinematics to determine the correct angular velocity for each joint requires matrix division, which will present a series of interesting challenges for deployment on stochastic circuits.

In summary, we argue that stochastic computing can offer numerous advantages in terms of datapath simplicity, robustness, error tolerance, and support for variable precision, all of which make it an appealing approach for these embedded applications where the most natural representations for inputs, outputs, and computation are serial oversampled bistreams.

VI. Acknowledgments

The demonstration board for this project was designed and built by Eric Hoffman. Chenxiao Guan, the teaching assistant for the Fall 2016 section of ECE 551 (Digital System Design and Synthesis), as well as all of the 56 enrolled students, waited patiently for the final specification for this effort and faithfully implemented it as their course project (thank you!). This work was supported in part by NSF grant CCF-1615014.

References

[1] B. Moons and M. Verhelst, "Energy-efficiency and accuracy of stochastic computing circuits in emerging technologies," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 4, no. 4, pp. 475–486, Dec 2014.

[2] J. Sartori, J. Sloan, and R. Kumar, "Stochastic computing: Embracing errors in architecture and design of processors and applications," in *2011 Proceedings of the 14th International Conference on Compilers, Architectures and Synthesis for Embedded Systems (CASES)*, Oct 2011, pp. 135–144.

[3] A. Alaghi and J. P. Hayes, "Survey of stochastic computing," *ACM Trans. Embed. Comput. Syst.*, vol. 12, no. 2s, pp. 92:1–92:19, May 2013. [Online]. Available: http://doi.acm.org/10.1145/2465787.2465794

[4] W. Qian, X. Li, M. D. Riedel, K. Bazargan, and D. J. Lilja, "An architecture for fault-tolerant computation with stochastic logic," *IEEE Trans. Comput.*, vol. 60, no. 1, pp. 93–105, Jan. 2011. [Online]. Available: http://dx.doi.org/10.1109/TC.2010.202

[5] J. von Neumann, "Probablistic logics and the synthesis of reliable organisms from unreliable components," 1952.

[6] P. Li, D. J. Lilja, W. Qian, K. Bazargan, and M. Riedel, "The synthesis of complex arithmetic computation on stochastic bit streams using sequential logic," in *Proceedings of the International Conference on Computer-Aided Design*, ser. ICCAD '12. New York, NY, USA: ACM, 2012, pp. 480–487. [Online]. Available: http://doi.acm.org/10.1145/2429384.2429483

[7] J. P. Hayes, "Introduction to stochastic computing and its challenges," in *Proceedings of the 52Nd Annual Design Automation Conference*, ser. DAC '15. New York, NY, USA: ACM, 2015, pp. 59:1–59:3. [Online]. Available: http://doi.acm.org/10.1145/2744769.2747932

[8] R. Manohar, "Comparing stochastic and deterministic computing," *IEEE Computer Architecture Letters*, vol. 14, no. 2, pp. 119–122, July 2015.

[9] "De0-nano development and education board," http://www.terasic.com.tw/cgi-bin/page/archive.pl?CategoryNo=139&No=593.

[10] "Pulse-density modulation," https://en.wikipedia.org/wiki/Pulse-density_modulation.

[11] "Delta-sigma modulation," https://en.wikipedia.org/wiki/Delta-sigma_modulation.

[12] "Class-d amplifier," https://en.wikipedia.org/wiki/Class-D_amplifier.

[13] "Cascaded integrator comb filter," https://en.wikipedia.org/wiki/Cascaded_integrator-comb_filter.

[14] "Digital biquad filter," https://en.wikipedia.org/wiki/Digital_biquad_filter.

[15] E. Hogenauer, "An economical class of digital filters for decimation and interpolation," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 29, no. 2, pp. 155–162, Apr 1981.

[16] "Pdm in a tiny cpu," https://curiouser.cheshireeng.com/2015/01/16/pdm-in-a-tiny-cpu/.

[17] R. Bristow-Johnson, "Cookbook formulae for audio eq biquad filter coefficients," http://www.musicdsp.org/files/Audio-EQ-Cookbook.txt.

[18] "Dynamic vision sensor," http://inilabs.com/products/dynamic-vision-sensors/.

[19] "Inverse kinematics," https://en.wikipedia.org/wiki/Inverse_kinematics.