Low Cost Hybrid Spin-CMOS Compressor for Stochastic Neural Networks

Bingzhe Li*
University of Minnesota, Twin Cities
Minneapolis, MN, USA
lixx1743@umn.edu

Jiaxi Hu*
University of Minnesota, Twin Cities
Minneapolis, MN, USA
huxxx499@umn.edu

M. Hassan Najafi
University of Louisiana at Lafayette
Lafayette, LA, USA
najafi@louisiana.edu

Steven Koester
University of Minnesota, Twin Cities
Minneapolis, MN, USA
skoester@umn.edu

David J. Lilja
University of Minnesota, Twin Cities
Minneapolis, MN, USA
lilja@umn.edu

ABSTRACT

With expansion of neural network (NN) applications lowering their hardware implementation cost becomes an urgent task especially in back-end applications where the power-supply is limited. Stochastic computing (SC) is a promising solution to realize low-cost hardware designs. Implementation of matrix multiplication has been a bottleneck in previous stochastic neural networks (SC-NNs). In this paper, we introduce spintronic components into the design of SC-NNs. A novel spin-CMOS matrix multiplier is proposed in which the stochastic multiplications are performed by CMOS AND gates while the sum of products is implemented by spintronic compressor gates. The experimental results indicate that compared to the conventional binary implementations the proposed hybrid spin-CMOS architecture can achieve over 125×, 4.5× and 43× reduction in terms of power, energy and area consumptions, respectively. Moreover, compared to previous CMOS-based SC-NNs, our design saves the power by 3.1× - 7.3×, reduces energy consumption by 3.1× - 7.3× and decreases area by 1.4× - 7.6× while maintaining similar recognition rates.

CCS CONCEPTS

• **Hardware** → *Spintronics and magnetic technologies*; *Logic circuits*;

KEYWORDS

Stochastic computing; Neural network; Spintronic; Compressor

ACM Reference Format:
Bingzhe Li, Jiaxi Hu, M. Hassan Najafi, Steven Koester, and David J. Lilja. 2019. Low Cost Hybrid Spin-CMOS Compressor for, Stochastic Neural Networks. In *Great Lakes Symposium on VLSI 2019 (GLSVLSI '19), May 9–11, 2019, Tysons Corner, VA, USA.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3299874.3317989

*Authors contributed equally

GLSVLSI '19, May 9–11, 2019, Tysons Corner, VA, USA

ACM ISBN 978-1-4503-6252-8/19/05…$15.00
https://doi.org/10.1145/3299874.3317989

1 INTRODUCTION

Neural networks (NNs) have become a popular technology in many fields. To solve complex problems designers need to add more and more neurons and layers into NNs which result in higher hardware cost in terms of power, energy and area. However, applications such as the Internet of Things (IoT) cannot tolerate high power or energy consumption due to insufficient power and energy supply. Therefore, reducing the hardware cost becomes a crucial issue for back-end NN-based applications.

Stochastic computing (SC) [1] as a promising technology can implement complex arithmetic operations using simple logic gates. For example, multiplication operations are implemented using simple standard AND gates in SC [2]; Additions can be achieved by OR gates as proposed in [3]. In terms of applications, previous works [4–12] have applied the SC technology to implementation of NNs to achieve energy-efficient and low-power hardware designs. For instance, Li *et al.* [5] proposed a NN implemented using stochastic components only, which successfully reduced the power consumption. Nevertheless, the implementations like [5] led to higher error rates compared to the conventional binary implementations due to the correlation and imprecision of bit-stream-based representation and inaccuracy of OR- and multiplexer-based additions. To solve this issue, hybrid stochastic-binary adders are introduced to sum up the products of matrix elements in the binary domain and so improve the accuracy [6–10]. However, such solutions are sub-optimal regarding the hardware cost due to the use of binary full adders.

Spintronic devices, which utilize electron spin instead of electron charge as the computation variable, provide new design opportunities to conventional VLSI circuits. In this paper, we present a hybrid spin-CMOS architecture to implement cost-efficient SC-based NNs. By taking advantage of the spin diffusion current, we design a novel compressor gate which can efficiently compress N sparse stochastic streams to M dense stochastic streams (N > M). The hierarchical compressor gates together with simple AND gates can realize stochastic matrix multiplication with very low hardware cost. The device functionalities and power/energy consumption are simulated and calculated by using the physics-based compact model [13]. Our experimental results show that, compared to the binary implementations of NNs and the previous SC-NNs implemented by CMOS circuits, the hybrid spin-CMOS architecture can significantly reduce the hardware cost of NNs in terms of power, energy, and area consumptions while achieving slightly higher

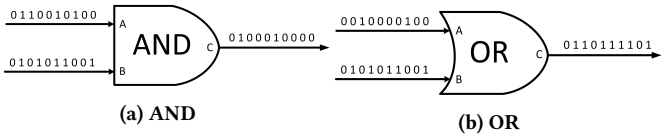


Figure 1: Stochastic multiplication and addition for the unipolar format.

recognition error rates compared to the conventional fixed-point binary implementations.

The remainder of this paper is organized as follows: Section 2 provides the background on SC and the motivation behind this work. Section 3 explains the functionality of the proposed compressor. Section 4 introduces the physical implementation of the spin-CMOS compressor. Section 5 shows the architecture of hybrid spin-CMOS SC-NNs, and the comparisons of recognition error rates and hardware cost. The conclusions are drawn in Section 6.

2 BACKGROUND AND MOTIVATION

2.1 Stochastic Computing

SC can achieve complex arithmetic operations with simple logic gates. A floating-point number in the [0,1] interval is expressed by a sequence of binary bits, called bit-stream, in which all bits have the same weight. The expressed value is based upon the probability of the bits being logic '1'. Unipolar and bipolar are the two common formats [14] to encode a floating-point number into a bit-stream representation. In this work, our proposed SC-NNs are implemented with the unipolar format.

Stochastic number generator (SNG) is used to generate bit-streams. The SNG is conventionally implemented using random number generators (RNGs) and comparators. A random number from an RNG is compared to a constant number (target value) in each cycle and the output of comparison generates one bit of the bit-stream. Linear feedback shifted registers (LFSRs) are widely used as the RNG in the SNGs. By exploiting spintronic devices, prior work [15, 16] reduced the hardware cost of the SNG and achieved promising performance in terms of both power consumption and hardware area cost.

For arithmetic operations, depending on the format of bit-streams, multiplication is implemented using an AND gate (Figure 1a) for the unipolar and an XNOR gate for the bipolar format. The addition can be implemented by OR gate (Figure 1b) for the unipolar format and the MUX gate can be used for both unipolar and bipolar formats.

2.2 Motivation

Matrix multiplication is the most compute-intensive operation in NNs. Previously, researchers used OR or MUX gates to implement the required stochastic additions for matrix multiplications [5]. However, due to the correlation and the non-scalable characteristics, the OR and MUX-based stochastic adders cause large error rates when the number of inputs is large. To decrease this error rate, authors in [6] and [16] proposed parallel counters and approximate parallel counters (APC), respectively. However, these binary adder-based implementations only achieve suboptimal hardware solutions because of the use of costly binary full adders particularly when

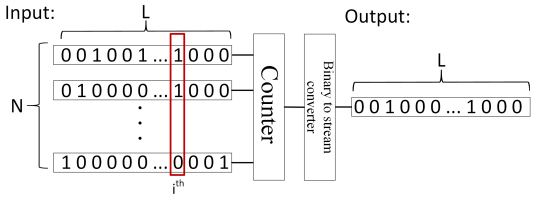


Figure 2: A demo of N-input stochastic addition in neural networks with L bit-length [16].

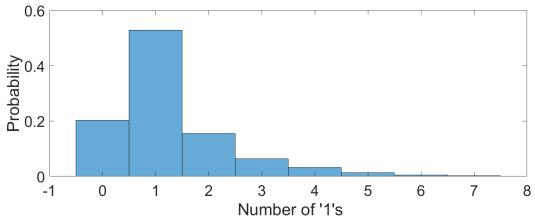


Figure 3: The probability of the numbers of '1's per clock cycle at the inputs of the counter shown in Figure 2

adding a large number of inputs. Efficient reduction of the hardware cost while providing similar recognition rates as the conventional binary designs is still a challenging task.

For matrix multiplications in stochastic NNs, most output values are located in the range of (-1, 1). The output values larger than 1 or smaller than -1 will be approximately assigned to 1 or -1 or be scaled to (-1, 1). As shown in Figure 2, assume that there are N bit-streams with L bit-length at the inputs. If we use a counter to compute the summation of N bit-streams and then convert it back to one bit-stream (the operation is equivalent to the N-input addition), the number of '1's in the input bit-streams should be equal to the number of '1's in the output bit-stream. Since the matrix multiplication results (intermediate results in NNs) are usually very small, it is highly possible that the number of '1's in one cycle (like i^{th} cycle) at the input side is smaller than a small threshold value.

To investigate the above assumption, we studied the statistics of the number '1's in the stochastic matrix multiplication of NNs. Figure 3 plots the probability of the numbers of '1's per clock cycle at the inputs of the counter shown in Figure 2. For example, for the i^{th} cycle in Figure 2 it has about 20% probability that all 500 inputs are '0' and about 55% probability that 500 inputs only have one '1'. Therefore, in the case of Figure 3, the 500 inputs have over 90% probability to contain less than four '1's per clock cycle, which means that the 500 bit-streams can be approximately compressed to three bit-streams. Therefore, if we develop a design that can efficiently compress X bit-streams to Y bit-streams ($X >> Y$), it will tremendously decrease the hardware cost of the parallel counters and hence the total hardware cost of the SC-based NNs.

3 FUNCTIONALITY OF COMPRESSOR

The compressor gates are designed to take N unipolar encoded stochastic bit-streams as input and compress them into M streams (N and M are the number of inputs and outputs of the compressor gate,

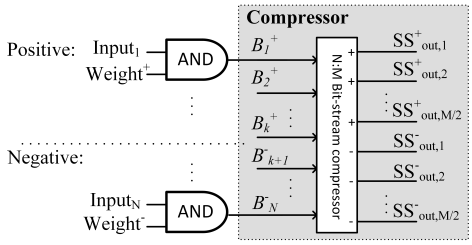


Figure 4: Schematic design of an N-to-M compressor gate.

respectively, and $N > M$). The proposed compressors combining with AND gates is used as the matrix multiplier in the stochastic NN. The inputs of the compressor gate are the products of bit-streams which are operated by using AND gates as shown in Figure 4. In the stochastic matrix multiplication, since we use the unipolar encoded bit-streams, the negative values are first regarded as positive values and later partitioned into positive and negative parts as shown in Figure 4. Since input images and outputs of each layer (assume activation function uses sigmoid function) are always positive, the signs of products depend only on the signs of the weights, which are deterministic after training of NNs. Therefore, the number of positive inputs, K, is deterministic in Figure 4. The function of the N-to-M compressor is to sum up the N input bits (the sum of bits in positive streams subtracts the sum of bits in negative streams) in every cycle. The compressor gate uses M output bits to express the summed value, in which half of the outputs stands for positive values and the other half expresses negative values. Assume S is the summed value in one clock cycle:

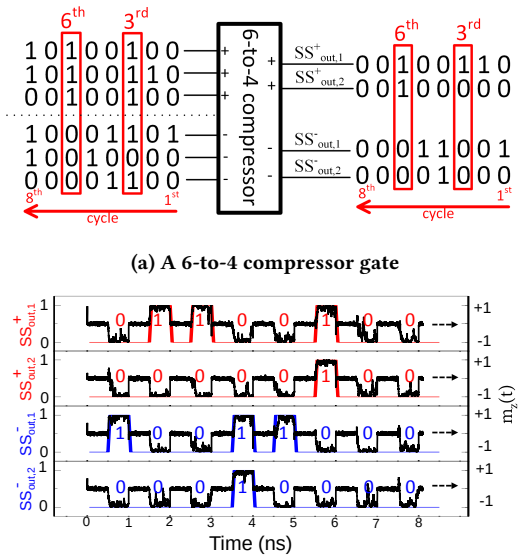
$$S = \sum_{i=1}^{K} B_i^+ - \sum_{i=K+1}^{N} B_i^- \tag{1}$$

where $B_i^{+/-}$ is the i^{th} input bit-stream value for the $+/-$ streams at a clock cycle. The M outputs are designed such that

$$\begin{cases} SS_{out,j}^{+} = 1, j = 1, 2, ..., M/2, & \text{if } M/2 \le S \\ SS_{out,j}^{+} = 1, j = 1, 2, ..., S, & \text{if } 0 \le S < M/2 \\ SS_{out,j}^{-} = 1, j = 1, 2, ..., |S|, & \text{if } -M/2 < S < 0 \\ SS_{out,j}^{-} = 1, j = 1, 2, ..., M/2, & \text{if } S \le -M/2 \\ SS_{out,j}^{\pm} = 0, & \text{otherwise} \end{cases} \tag{2}$$

where $j = 1, 2, ..., M/2$ is the index of the output bit-streams.

Figure 5a shows a structure of a 6-to-4 compressor assuming K=3. In this case, the compressor can compute the range of the summation result, S, from -2 to +2. For example, in the 6^{th} cycle in Figure 5a the positive inputs are '111' while the negative inputs are '000'. As a result, $S = 3$, which is out of the range (-2, +2). The compressor gives the result as +2 rather than +3, of which the outputs have '11' at the positive and '00' at the negative side. As another example, in the 3^{rd} cycle in Figure 5a, the positive and negative inputs have three and two '1's, respectively. Two '1's at the positive and negative inputs cancel each other and only one '1' is left at the positive side. The summation result in this case is a single '1' and thus only the output $SS_{out,1}^{+}$ is set to '1'. For the cases that the total number of '1's of input bit-streams in one cycle is



(a) A 6-to-4 compressor gate

(b) Transient simulation of the 6-to-4 compressor gate shown in (a).

Figure 5: An example of a 6-to-4 compressor gate with six inputs and four outputs (N=6, M=4, K=3 in Figure 4).

zero or one, the outputs will have zero '1' and one '1', respectively, as seen in the 2^{nd} and 7^{th} cycles. As discussed in Section 2, it is very likely to see that the number of '1's in the inputs is less than a threshold value (M), in which the compressor gate accurately represents the summation of the N bit-streams using the M outputs bit-streams. As can be seen in Figure 5b, the proposed spin-CMOS compressor gate can successfully realize the functionality shown in Figure 5a. The implementation details are discussed in Section 4.

4 SPIN-CMOS BASED COMPRESSOR IMPLEMENTATION

In this section, we discuss the implementation of the proposed hybrid spin-CMOS based compressor. Figure 6 shows the structure of the proposed compressor gate which is based on a nonlocal spin valve with multiple inputs and outputs [17]. The nonlocal spin valve provides the ability to operate spin diffusion currents which are efficient in implementing analog summations [18]. The N inputs are connected to the N bit-streams. If an input bit is '1', the charge current injects non-equilibrium spin from the ferromagnetic (FM) spin injector into the nonmagnetic (NM) spin channel. The injected spin diffuses evenly to the M outputs. The polarity of one input stream is defined as the injected spin polarization which is determined by the magnetic states of the FM injector. If multiple '1s' arrives at the inputs, the injected spins will sum up at the merging point ('X' in the middle of Figure 6) of N·M channels in an analog manner. The control transistors at the input terminals cut out the charge current path to achieve low power performance if the input is '0'.

At the injector terminals, tunnel barriers are used to enhance the spin injection polarization, P, and prevent the cross-talk between

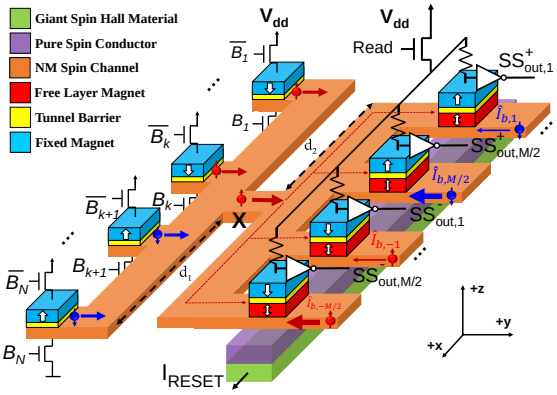


Figure 6: Physical implementation of an N-to-M hybrid spintronic-CMOS compressor gate shown in Figure 4.

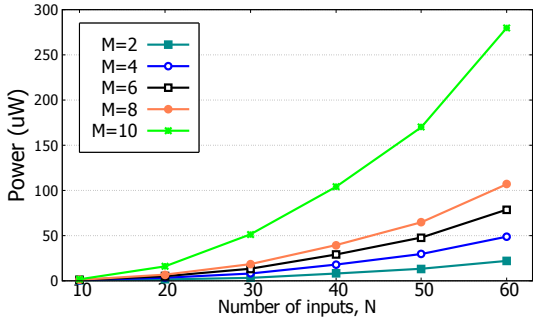


Figure 7: Power consumption of one N-to-M compressor gate with only one '1' appearing at its inputs.

inputs [18]. The analog summation of up/down spin current naturally emulates the arithmetic operation of positive and negative stochastic bits in Eq. 1. At the point 'X', the total output spin current can be written as:

$$\hat{I}_{s,tot} = I_0 \sum_{i=1}^{N} \hat{m}_{z,i} B_i = I_0 (\sum_{i=1}^{K} B_i - \sum_{i=K+1}^{N} B_i) = S \cdot I_0 \hat{z} \quad (3)$$

where B_i is the incoming bit at the i-th input and m_z is the z-magnetization of the corresponding FM injector. I_0 is the unit output current from single input which is defined by the diffusion equation: $I_0 = \eta \cdot PI_c \cdot e^{-d_1/\lambda_s}$. I_c is the applied charge current and $\eta = 0.5$ is the output efficiency of spin current [13]. d_1 is the distance between the injectors to 'X' and λ_s is the spin diffusion length. By programming the magnetic state of the FM injectors, the K value of each compressor gate can be individually programmed from 0 to N [17].

At the output terminals, Eq. 2 is implemented by a series of low-power spin current comparators, which are based on magnetic tunnel junctions (MTJs) built on NM spin channels. The switching of the free layer (FL) magnets are determined by the competition between the output spin currents and the bias spin currents. Beneath the NM layer, a stack of pure spin conductor and Giant Spin Hall Material [19] are introduced as the "reset clock." Because of the "Giant Spin Hall Effect," a charge current in the Giant Spin Hall Material (GSHM) generates $\hat{I}_{s,SHE}$ with in-plane spin polarization on its top surface. $\hat{I}_{s,SHE}$ switches the FL magnets to in-plane, which is the energy maximum state. Once the RESET signal is "OFF", a weak perturbation can determine the switching of FL magnets. Since in SC the operation frequency is high, the FL magnets can be designed with low thermal stabilities, Δ [20]. As shown in Figure 5b, with $\Delta = 4k_BT$, the FL magnet can still well maintain its state at 1 GHz.

Table 1: Important material parameters used in this work

Name	Value
V_{dd}	1V
Sensing MTJ TMR ratio	354% [21]
Spin Injection Polarization	88% [22]
Damping Constant of FL magnets	0.005 [23]
Critical Switching Spin Current I_{sc}	0.18 μA [13]
Spin Hall Angle	18 [24]
Minimum Feature Size (Spintronic Components)	15nm

The MTJ senses the state of FL magnets and transfer it into a voltage signal. To reduce the process-induced variation, in the voltage divider, the reference resistor should also be a neighboring MTJ but with its magnetic state always fixed. The resistance value of the reference MTJ is determined such that when the sensing MTJ is in the parallel state, the corresponding SS_{out} will be '1', otherwise the output will be '0' [18]. The unit output spin current, I_0, and the critical switching current of FL magnets, I_{sc}, are designed as: $I_0 = 2MI_{sc}e^{d_2/\lambda_s}$ such that the spin currents that arrive the output equal to $2I_{sc}$ if there is only a single '1' at the inputs of the compressor gate. d_2 is the distance between 'X' and outputs. Correspondingly, the bias spin currents, which can also be generated by nonlocal spin valves, are designed as $|\hat{I}_{b,j}| = \mp(2 \cdot j - 1)I_{sc}$ for positive (or negative) outputs ($j = 1, ..., M/2$ is the index of output terminals). As a result, by measuring the analog level of the $\hat{I}_{s,tot}$, a compressor gate can count the net amount of the input bits. Figure 5b shows the simulation of a 6-to-4 compressor gate by using the modular approach for spintronic devices. The magnetic dynamics are simulated by solving the Landau-Lifshitz-Gilbert equation with the random thermal fluctuation and the spin torques [13]. In each cycle, the FL magnets are first reseted to in-plane by $\hat{I}_{SHE}$. Then, the RESET signal is "OFF" and the READ signal is "ON" by which the FL magnets complete switching. Based on the parameters in Table 1, the switching delay of FL magnets is shown to be less than 100ps.

A physical limitation on fan-in and fan-out is also considered: the length of NM channels are assumed to increase linearly with N and M: $d_1 = N \cdot 2F$ and $d_2 = M \cdot 2F$, where F is the minimum feature size. As a result, the applied charge current, I_c, exponentially increases with N+M if $d_1 + d_2 > \lambda_s$, which sets the trade-off between the energy consumption per gate and the fan-in and fan-out abilities. Figure 7 shows the power scaling trend of a compressor gate when only a single '1' appears at its inputs. Its strong dependence on the N and M suggests that the energy consumed for nonlocal spin torque switching is still dominating. Thus, increasing λ_s or decreasing I_{sc} can further improve the energy efficiency. In the rest of this

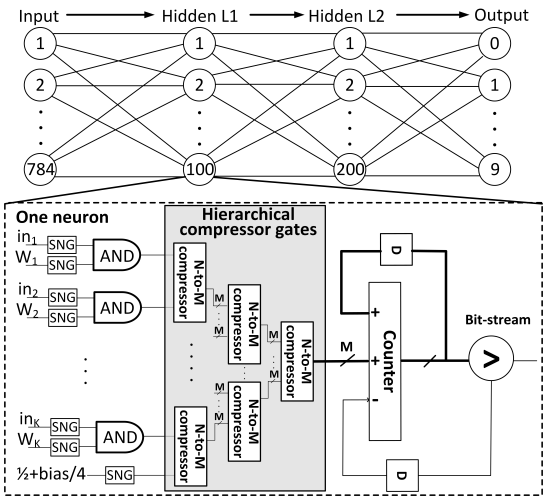


Figure 8: Stochastic neural network implementation with a single neuron architecture

paper, the N value of each compressor gate is kept under 50 and a hierarchical structure is utilized to implement the summation over a large number of inputs (Figure 8).

5 IMPLEMENTATION OF NEURAL NETWORKS

In this section, SC-based implementation of a restricted Boltzmann machine (RBM) classifier [25] with a 784-100-200-10 configuration is investigated. The stochastic implementation of the RBM classifier, the recognition error rates, and the hardware cost for different implementations are discussed in the following sections.

5.1 Implementation

First, we introduce the NN implementation using the hybrid spin-CMOS devices. The basic unit is a neuron formulated as $out = \sigma(X * W + bias)$ where σ is the activation function; X is the input of the neuron; W and $bias$ are constant values in the classification which are obtained from training process.

The MTJ based SNG [26] is used to generate bit-streams in the neuron architecture of Figure 8. The inputs are ANDed with the weighted bit-streams to obtain the intermediate products of matrix multiplications. N-to-M bit-stream compressors are then used in a hierarchical structure to reduce the number of counter inputs to M. Finally, the output of the counter goes through a comparator to convert it back into bit-stream representation and feed the next layer of the NN. The activation function uses the stochastic approximate sigmoid function, $\frac{x+2}{4}$, proposed in [7] which is integrated in the neuron structure of Figure 8.

5.2 Recognition Error Rate

To evaluate the recognition error rates, the MNIST handwritten digit image dataset [27] is used which includes 60,000 training images and 10,000 testing images. For the classification process, the weights and biases of the RBM NN are trained and regarded as constant values for the classifier. Due to a long simulation time, we tested each 10,000 images classification process three times and averaged the error rates.

Table 2: Comparison of recognition error rates of RBMs

Type	N-to-M	Bit-length:			
		32	64	128	256
Binary	-	2.05%			
Stochastic#1 [6][16]	-	>10%	6.06%	5.58%	4.34%
Stochastic#2 [7][8]	-	3.20%	3.00%	2.84%	3.08%
This work	*N-to-2	>15%	>15%	>15%	>15%
	10-to-4	3.99%	3.26%	2.82%	2.79%
	10-to-6	3.79%	3.20%	2.68%	2.88%
	20-to-4	4.00%	3.21%	3.03%	2.77%
	20-to-6	3.62%	2.97%	2.77%	2.69%
	30-to-4	4.02%	3.23%	2.91%	2.94%
	30-to-6	3.81%	2.97%	2.77%	2.71%
	40-to-4	4.15%	3.64%	3.36%	3.22%
	40-to-6	3.85%	3.13%	2.96%	2.89%

Note: *$N \geq 10$ might be 10, 20, 30, etc. and when M=2, the error rates are larger than 15% which are not acceptable.

Four different design methodologies are investigated in this paper. As reported in Table 2, the 8-bit fixed point implementation has the lowest recognition error rate. Both the CMOS stochastic and our hybrid spin-CMOS stochastic implementations obtain similar recognition error rates. By increasing the bit-length, the error rates of stochastic implementations become closer to the error rates of the fixed-point implementation. For our hybrid spin-CMOS implementations, we also investigate the impact of the ratio of inputs to outputs (N/M) on the error rate. As reported in Table 2, when M=2, the recognition error rates are larger than 15%. The reason behind this relatively higher error rates is that in this case the proposed spin-based compressor compresses the inputs to only one '+1' or only one '-1' which results in losing '±1's when one cycle contains more than one '±1'. For example, the case in Figure 3 shows that there is more than 20% probability to generate two or more '1's in each cycle. Consequently, when M=2, the output of the compressor cannot accurately express those 20% '1's resulting in such a high error rate. Therefore, we will not consider the case of M=2 for the implementations. For other cases, when decreasing the ratio N/M, the error rates are slightly decreased. This is because the spin-based compressors with lower ratios of N/M have a lower compression rate and thus introduce smaller errors into the addition operations. In summary, with decreasing the ratio N/M and increasing the bit-length, the recognition error rates are decreased.

5.3 Hardware Cost

To evaluate the hardware cost, the Synopsys Design Compiler is used to synthesize the RBM NN with the FreePDK 45nm library [28] for the CMOS designs including the proposed implementation, conventional binary, and previous stochastic implementations. The power consumption is reported at a constant frequency of 1GHz. The spintronic components are quantitatively simulated using the device and material parameters summarized in Table 1. To provide a fair comparison, we use the MTJ-based SNGs of [26] to generate the required random numbers in all stochastic implementations.

Table 3: Hardware comparisons among different works (32 bits for stochastic works)

Method	N-to-M	Power (@1GHz) (mW)	Area (mm^2)	Energy (nJ)
#1 8-bit Fixed-Point	N/A	39617	55.77	39.62
#2 [6][16]	N/A	1929.77	2.06	61.75
#3 [7][8]	N/A	1004.03	1.39	32.13
#4 This work	10-to-4	258.36	1.01	8.27
	10-to-6	270.00	1.28	8.64
	20-to-4	263.59	0.87	8.43
	20-to-6	273.89	0.93	8.76
	30-to-4	271.47	0.86	8.69
	30-to-6	289.27	0.88	9.26
	40-to-4	292.06	0.85	9.35
	40-to-6	315.97	0.87	10.11

Four implementations, (**#1**) the conventional fixed-point binary implementation, (**#2**) the CMOS stochastic implementation with the bipolar format [16][6], (**#3**) the CMOS implementation with the unipolar format [7][8], and (**#4**) the hybrid spin-CMOS implementation are compared in terms of power, area and energy consumption. As shown in Table 3, compared to the CMOS binary implementation, our designs can achieve over 125× power reduction, over 43× area reduction and 4.5× energy reduction. Compared to the previous CMOS stochastic implementations, our designs reduce about 3.1× - 7.3× power, 1.4× - 7.6× area and 3.1× - 7.3× energy consumption.

Moreover, we investigate the impact of different ratios of N/M on the hardware cost. When increasing the ratio of N/M, the total area of the NN is reduced because a larger ratio means a smaller number of compressor gates. Changing N and M has a direct impact on the power and energy numbers as discussed in Section 3. As shown in Table 3, by increasing N and M, the power and energy consumption are both increased. Thus, the design with the compressor ratio of 10-to-4 has the lowest power and energy consumption.

6 CONCLUSION

In this work, we proposed a hybrid spin-CMOS based stochastic neural network architecture targeting low-cost design. We proposed a spin-CMOS based compressor to compress the input bit-streams into a smaller number of output bit-streams which can significantly reduce the hardware cost of the entire neural network. Experimental results showed that our proposed design obtains similar recognition error rates compared to previous stochastic implementations while achieving slightly higher error rates compared to the conventional binary implementation. In terms of hardware cost, compared to the conventional CMOS-based fixed-point binary implementation, our designs achieve over 125× power reduction, over 43× area reduction and 4.5× energy reduction. Compared to the previous CMOS stochastic implementations our designs reduce about 3.1× - 7.3× power, 1.4× - 7.6× area and 3.1× - 7.3× energy.

7 ACKNOWLEDGMENTS

This work was supported in part by National Science Foundation grant no. CCF-1408123 and Seagate Technology.

REFERENCES

[1] Brian R Gaines et al. Stochastic computing systems. *Advances in information systems science*, 2(2):37–172, 1969.
[2] Peng Li, et al. Logical computation on stochastic bit streams with linear finite state machines. *IEEE Transactions on Computers*, page 1, 2012.
[3] Jeffery A Dickson, Robert D McLeod, and HC Card. Stochastic arithmetic implementations of neural networks with in situ learning. In *Neural Networks, 1993., IEEE International Conference on*, pages 711–716. IEEE, 1993.
[4] Bradley D Brown and Howard C Card. Stochastic neural computation. ii. soft competitive learning. *IEEE Transactions on Computers*, 50(9):906–920, 2001.
[5] Bingzhe Li, M Hassan Najafi, and David J Lilja. Using stochastic computing to reduce the hardware requirements for a restricted boltzmann machine classifier. In *Proceedings of the 2016 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, pages 36–41. ACM, 2016.
[6] Ao Ren, et al. Sc-dcnn: Highly-scalable deep convolutional neural network using stochastic computing. In *Proceedings of the Twenty-Second International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 405–418. ACM, 2017.
[7] Bingzhe Li, et al. Neural network classifiers using stochastic computing with a hardware-oriented approximate activation function. In *2017 IEEE 35th International Conference on Computer Design (ICCD)*, pages 97–104. IEEE, 2017.
[8] Vincent T Lee, et al. Energy-efficient hybrid stochastic-binary neural networks for near-sensor computing. In *2017 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 13–18. IEEE, 2017.
[9] S Rasoul Faraji, et al. Energy-Efficient Convolutional Neural Networks with Deterministic Bit-Stream Processing. In *Design, Automation, and Test in Europe (DATE)*, 2019.
[10] Bingzhe Li, et al. Neural network classifiers using a hardware-based approximate activation function with a hybrid stochastic multiplier. *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, 15(1):12, 2019.
[11] Reza Hojabr, et al. SkippyNN: An Embedded Stochastic-Computing Accelerator for Convolutional Neural Networks. In *Design Automation Conference (DAC)*, 2019.
[12] Bingzhe Li, M Hassan Najafi, and David J Lilja. Low-cost stochastic hybrid multiplier for quantized neural networks. *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, 2019.
[13] Kerem Yunus Camsari, Samiran Ganguly, and Supriyo Datta. Modular approach to spintronics. *Scientific reports*, 5:10571, 2015.
[14] Bradley D Brown and Howard C Card. Stochastic neural computation. i. computational elements. *Computers, IEEE Transactions on*, 50(9):891–905, 2001.
[15] Rangharajan Venkatesan, et al. Spintastic: Spin-based stochastic logic for energy-efficient computing. In *Proceedings of the 2015 Design, Automation & Test in Europe Conference & Exhibition*, pages 1575–1578. EDA Consortium, 2015.
[16] Kyounghoon Kim, et al. Dynamic energy-accuracy trade-off using stochastic computing in deep neural networks. In *Proceedings of the 53rd Annual Design Automation Conference*, page 124. ACM, 2016.
[17] Behtash Behin-Aein, et al. Proposal for an all-spin logic device with built-in memory. *Nature nanotechnology*, 5(4):266, 2010.
[18] Jiaxi Hu, et al. Using programmable graphene channels as weights in spin-diffusive neuromorphic computing. *IEEE Journal on Exploratory Solid-State Computational Devices and Circuits*, 2018.
[19] Shehrin Sayed, et al. Spin funneling for enhanced spin injection into ferromagnets. *Scientific reports*, 6:28868, 2016.
[20] Kerem Yunus Camsari, et al. Stochastic p-bits for invertible logic. *Physical Review X*, 7(3):031014, 2017.
[21] Hong-xi Liu, et al. Giant tunneling magnetoresistance in epitaxial co2mnsi/mgo/co2mnsi magnetic tunnel junctions by half-metallicity of co2mnsi and coherent tunneling. *Applied Physics Letters*, 101(13):132418, 2012.
[22] Benjamin Balke, et al. Mn 3 ga, a compensated ferrimagnet with high curie temperature and low magnetic moment for spin torque transfer applications. *Applied physics letters*, 90(15), 2007.
[23] S Mizukami, et al. Long-lived ultrafast spin precession in manganese alloys films with a large perpendicular magnetic anisotropy. *Physical review letters*, 106(11):117201, 2011.
[24] DC Mahendra, et al. Room-temperature high spin–orbit torque due to quantum confinement in sputtered bi x se (1–x) films. *Nature materials*, page 1, 2018.
[25] Geoffrey E Hinton and Ruslan R Salakhutdinov. A better way to pretrain deep boltzmann machines. In *Advances in Neural Information Processing Systems*, pages 2447–2455, 2012.
[26] Ankit Mondal and Ankur Srivastava. Power optimizations in mtj-based neural networks through stochastic computing. In *Low Power Electronics and Design (ISLPED, 2017 IEEE/ACM International Symposium on*, pages 1–6. IEEE, 2017.
[27] Yann LeCun, Corinna Cortes, and CJ Burges. Mnist handwritten digit database. *AT&T Labs [Online]. Available: http://yann. lecun. com/exdb/mnist*, 2, 2010.
[28] James E Stine, et al. Freepdk: An open-source variation-aware design kit. In *Microelectronic Systems Education, 2007. MSE'07. IEEE International Conference on*, pages 173–174. IEEE, 2007.