A 4.4-75 TOPS/W 14nm Programmable, Performance- and Precision-Tunable All-Digital Stochastic Computing Neural Network Inference Accelerator

Wojciech Romaszkan *Student Member, IEEE*, Tianmu Li *Student Member, IEEE*, Rahul Garg, Jiyue Yang, *Student Member, IEEE*, Sudhakar Pamarti, *Senior Member, IEEE*, Puneet Gupta, *Fellow, IEEE*

***Abstract*—We present the first programmable and precision-tunable Stochastic Computing (SC) neural network (NN) inference accelerator. The use of SC makes it possible to achieve multiply-accumulate (MAC) density of 38.4k MAC/mm^2, enabling a level of spatial data reuse unachievable to conventional, fixed-point architectures. This extensive reuse amortizes the cost of SC conversion and reduces the number of memory accesses, which can otherwise consume significant energy and latency. Our accelerator is a stand-alone architecture, with a custom instruction set architecture (ISA), and support for end-to-end model inference with convolutional and fully-connected layers of variable input and filter sizes. Further, it demonstrates extensive accuracy-latency trade-offs by varying the stream length. The 14nm demonstration chip achieves 2.4 TOPS and 75 TOPS/W peak throughput and energy efficiency, outperforming comparable fixed-point accelerators.**

***Index Terms*—Stochastic Computing, Neural Networks, Accelerator, CMOS.**

I. Introduction

Growing demand for edge NN inference requires domain-specific accelerators able to satisfy accuracy, latency, and energy constraints. Multi-precision [1], as well as bit-serial [2], designs have been proposed to enable such levels of configurability. Unfortunately, the former incur area penalties for separate arithmetic units for each precision, while the latter provide only fractional improvements with minor precision changes. Further, they often cannot fully capitalize on extensive data reuse available in NNs due to the limited number of relatively large processing elements (PEs). This results in additional costly memory accesses, which can dominate accelerators' energy consumption and latency. Stochastic Computing (SC) [3], which represents numbers using binary streams, can alleviate the above issues. First, manipulating the stream length used enables extensive latency-accuracy trade-offs, on a much finer granularity than available in bit-serial architectures. Second, its use of extremely compact, single-gate arithmetic units makes spatial parallelism possible on a scale unachievable to fixed-point designs. While individual SC components have previously been taped out [4], no system-level designs have been shown to date. We present the first programmable SC neural network accelerator, with a custom ISA supporting a wide range of layer types and sizes, which demonstrates SC precision-tunability. It achieves compute density of 38.4k MAC/mm^2 enabling extensive data reuse, amortizing the cost of conversion and memory accesses.

The authors are with the Electrical and Computer Engineering Department, University of California, Los Angeles, Los Angeles, CA, 90095, US (e-mail: wromaszkan@ucla.edu).

II. Architecture

The block diagram of our SC accelerator is shown in Figure 1. To emphasize the benefits of using stochastic computing, we highlight components that support variable precision in blue and reuse-oriented design choices in red. Numbers associated with each technique refer to the order used in Figure 4. The accelerator consists of control logic, an activation scratchpad coupled with buffers and stochastic number generators (SNGs), and six multiply-accumulate (MAC) block rows, each with its weight memory (total 131KB), and output counters, pooling, and activation logic. Control consists of instruction memory (4KB), a dispatcher, and distributed control units. The activation scratchpad is organized as a ping-pong buffer (2 banks of 16KB). SNG buffers are organized as shift registers that emulate a vertically sliding convolutional window for improved *activation and weight temporal reuse* [3]. After conversion into stochastic streams, *activation broadcast* is used across all six MAC rows, enabling a high level of spatial reuse. All MAC rows use the same activations, but each can compute a single output channel in a convolutional layer, or 3 rows can be coupled to compute one output channel in a fully-connected layer. Values are stored in memory using a fixed-point format and converted into stochastic streams only for computation, meaning that variable-precision support is transparent to memory and control logic. Our architecture can be easily scaled up to a larger area to increase performance. For example, the number of MAC rows, or the size and number of MAC columns can be increased to handle more concurrent filters or larger inputs. Likewise, the on-chip memory capacity can be increased to enable larger models. Because of the extensive memory data reuse of our accelerator, we expect the efficiency to be maintained when the design is scaled up, within the constraints of edge-class devices. Server-class inference and training architectures are beyond the scope of this work.

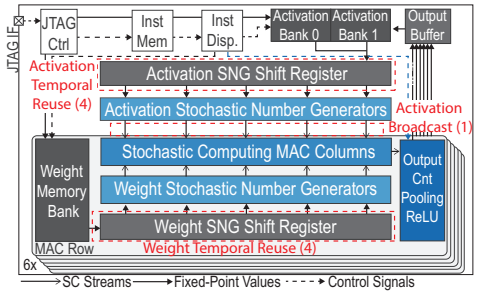


Fig. 1. Overall accelerator architecture.

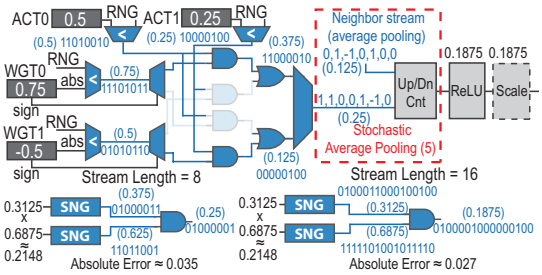


Fig. 2. SC split-unipolar MAC and stochastic average pooling (top). Precision-latency trade-off using different stream lengths (bottom).

III. SC Computation

Figure 2 explains how the underlying SC computation is performed, through an example of a 2-way dot product using 8-bit long streams. We use comparator-based SNGs, with 7-bit linear feedback shift registers [3]. Multipliers and adders are implemented using AND and OR gates, respectively. Their small size allows us to pack extremely wide dot products, up to 200-wide, without timing or congestion issues present in fixed-point designs. To support bipolar weights, split-unipolar computation is used, which separates positive and negative multiplication results, and adds them in separate, duplicated accumulation trees [3]. After accumulation, positive and negative streams are subtracted and fed into a counter for conversion to fixed-point. Each up/down counter accepts outputs of two neighboring accumulation trees to implement X-dimension (*horizontal*, e.g. 1x2) part of *stochastic average pooling* by concatenating the streams corresponding to two adjacent outputs in the same output row. Y-dimension (*vertical*, e.g. 2x1) part of pooling is implemented sequentially, using the previously described sliding window. The choice of average pooling is dictated by its efficient SC implementation, which can reduce computation latency by up to 4x without any loss in accuracy, by skipping the computation of bits that would have otherwise not been used [3] Alternative dimensionality reduction methods such as max pooling do not have such efficient SC implementations. Pooling is only used after convolutional layers. Counters are followed by ReLU activation and a configurable power-of-2 scaling unit to handle different stream lengths. Scaling factors range from $1/4$ to 32, is programmable, and can be set on a layer basis. Figure 2 also shows how by changing the stream length, the computation precision can be changed at runtime. Since values are stored in a fixed-point format, and only converted to streams for computation, our accelerator can support arbitrary precision with negligible area overheads.

IV. Computation Mapping

Convolution mapping is shown in Figure 3. We omit the fully-connected layer explanation due to a lack of space. Each MAC block row is organized into five columns, each corresponding to one input and filter row across 4 or 8 input channels. Fixed-point input and weight rows are loaded into the SNG buffers of its respective column. This way, a complete 5x5 filter can be computed in parallel, generating one complete row of outputs per MAC block row. Both sets of SNG buffers are organized as shift registers, meaning that after one iteration, a new input row is shifted in, and the subsequent output row can be computed. This *sliding activation reuse* behavior emulates a vertically-sliding convolutional window, reducing the number of required memory accesses.

Each column is organized into eight SC MAC blocks, each corresponding to one input channel, operating in a fully-streaming manner. A block takes 16 inputs and 5 weights and implements a *sliding MAC window*, generating up to 16 partial output streams depending on padding. *Activation stream multicast* with overlap is used between successive dot-products, and *weight stream broadcast* is used for all dot products in a block, amortizing stream generation cost. Every two neighboring blocks can be coupled to support wider input rows. Corresponding output streams from all 8 blocks are then reduced (*Z-dimension stream reduction*), after which corresponding outputs from all 5 columns are reduced (*Y-dimension stream reduction*). This hierarchical, SC reduction enables wide dot products (up to 200-wide) to be unrolled spatially, amortizing the cost of converting stochastic streams to fixed-point outputs. Our reuse-oriented design choices make it possible to perform over 130 MACs per memory access, and over 40 MACs per stream generation, as shown in Figure 4, lowering the relative energy cost of those operations. This high level of data reuse was only possible by using very dense SC computation – fixed-point designs have an order of magnitude lower memory access reuse [3].

Since computation, highlighted in blue, operates purely on stochastic streams, it can support arbitrary precision by adjusting SC stream length, only limited by the width of the output counters. Precision selection is possible on sub-layer granularity, e.g., groups of filters. Our custom instruction set enables selective gating of MAC block rows, columns, and blocks to support smaller activation or filter sizes. Filters or inputs that cannot be fully unrolled using existing resources are computed sequentially. A set of hardware loops with multi-stride memory accesses enables a variety of layer shapes and dataflow choices. Our accelerator can therefore support end-to-end inference with different types and shapes of layers.

V. Evaluation & Results

Figure 5 shows the SC training setup. To optimize the accuracy of SC models, stream generation and SC computation need to be accurately modeled during training, which can be expensive if streams for each computation need to be generated

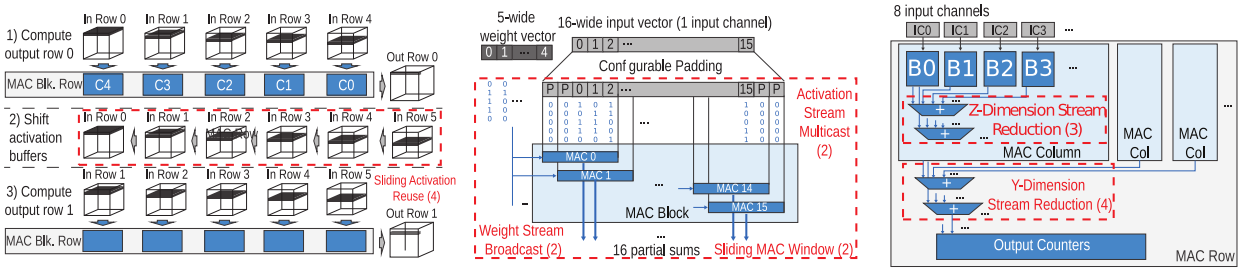


Fig. 3. Mapping of convolutional layers in MAC rows, and memory/stream generation amortization through data reuse. Shift-register organization emulating sliding window (left), MAC block with input/weight reuse and padding support (center), block organization implementing different levels of reduction (right).

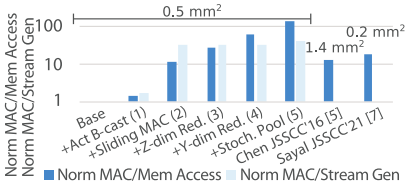


Fig. 4. Normalized ratio of MAC to memory accesses and stream generations compared to fixed-point designs. Accelerator area scaled to 14nm is included.

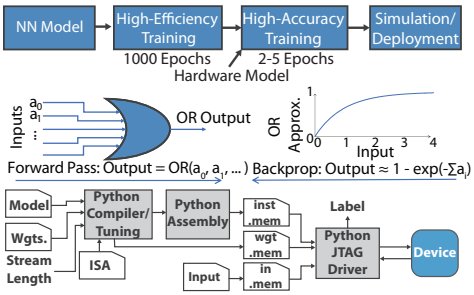
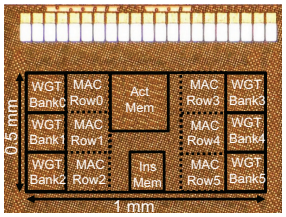


Fig. 5. SC-aware training using accurate stream modeling (top), OR-addition approximation to recover accuracy enables the use of shorter SC streams without sacrificing accuracy (middle), and model deployment pipeline (bottom).

TABLE I
DATASETS AND MODELS USED IN EVALUATION. MODEL SIZES ARE LIMITED BY AVAILABLE ON-CHIP MEMORY.

Dataset	Model	Model Architecture	Size [kB]
CIFAR-10,	tinyConv	CN5x32-CN5x32-CN5x64-FC10-BN*	89
SVHN	tinyConv-L*	CN5x32-CN5x32-CN5x32-CN5x32-FC10-BN*	81
MNIST	LeNet-5	CN5x6-CN5x16-FC120-FC84-FC10	62
	LeNet-3	CN5x6-CN5x16-FC10	7



	Specifications
Technology	14nm LPP
Die Area	5 mm²
Core Area	0.5 mm²
Supply Voltage	0.6-0.9V
Frequency	250-500 MHz
Precision	8-64 bitstreams
Memory	167 KB
Power	16mW @ 250 MHz, 0.6V
	68mW @ 500 MHz, 0.9V

Fig. 6. Die shot and specifications.

separately. To speed up the training procedure, models are first trained using a fast generation scheme that maximizes stream sharing and then fine-tuned using an accurate model of the hardware. The effect of OR accumulation is modeled using an additional activation function shown in Figure 5. Models are trained on MNIST, CIFAR-10, and SVHN datasets, using models shown in Table I. An output counter overflow issue in the taped-out design causes the deployed models to use aggressive scaling factors, which result in a 0.2-7.4 p.p. accuracy drop compared to simulated results without the overflow. The scaling factor is set to be $2^{\lceil log_2(max(|a|))\rceil}$, where $max(|a|)$ is the largest magnitude observed in the output activations of a layer. This scaling factor ensures that output activations do not overflow after scaling, and that the scaling function can be achieved using simple shifts. Accuracy could also be improved through larger networks, or more recent SC accuracy improving techniques, such as the ones proposed in [5]. Once trained, models are translated into a sequence of accelerator instructions using a compiler under active development, which is then programmed onto the device, as shown in Figure 5.

The chip, shown in Figure 6, was fabricated in 14nm LPP technology. The core of our accelerator occupies a 0.5 mm^2 area and operates at 0.6-0.9V voltage and a maximum frequency of 500MHz. Figure 7 shows the accuracy, latency, and energy measurements on different networks and datasets. We highlight that SC exposes a precision-performance tuning knob, by adjusting the stream length. For example, on the MNIST dataset, changing the stream length from 32 to 16 reduces the accuracy by only up to 0.3 p.p., while reducing energy and latency up to 31%. On SVHN, accuracy can be improved by up to 4.7 p.p. with a 50% increase in latency and energy. Non-ideal performance scaling is caused by control logic overhead and will be improved in the future. Thanks to highly compact SC compute we achieve higher data reuse than

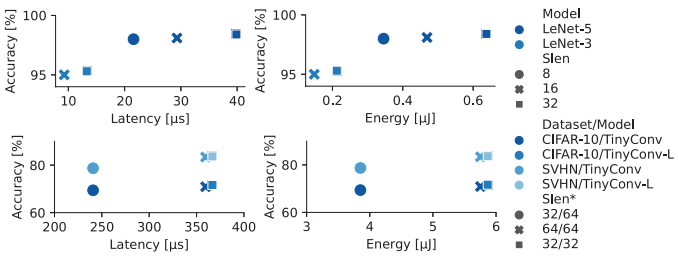


Fig. 7. Accuracy, latency (left), and energy (right) on the MNIST (top), SVHN and CIFAR-10 (bottom) datasets.

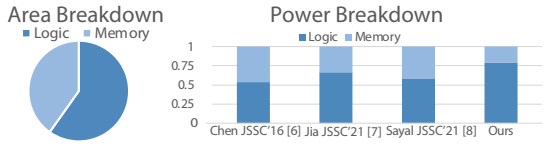


Fig. 8. Area (left) and power (right) breakdown, compared to [6]–[8].

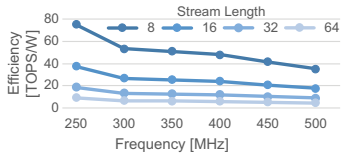


Fig. 9. Peak energy efficiency at different stream lengths.

other accelerators, resulting in lower relative memory power contribution, as shown in Figure 8.

Peak energy efficiency ranges between 9.4 and 75 TOPS/W at 250MHz/0.6V, as shown in Figure 9. Table II shows a comparison with prior work. Overall, SC offers unparalleled computational density in terms of MAC units per mm^2. Our chip outperforms fixed-point accelerators [1] in energy efficiency while enabling configurable precision. While bit-serial architectures [2] can have high peak energy efficiency, it is only achievable at single-bit precision. Mixed-signal, binary design in [9], achieves extremely high energy efficiency, but it comes at a cost of little configurability, supporting only one convolutuional filter size. It requires using much larger, slower models to improve accuracy. We also outperform, or approach the efficiency of neuromorphic and analog designs [10], [11], without suffering from their scalability and programmability issues, on account of being purely digital.

TABLE II
COMPARISON TABLE.

	ISSCC '21 [1]	ISSCC '18 [2]	JSSC '18 [9]	ISSCC '19 [10]	ISSCC '19 [11]	This Work
Type	Digital	Digital, bit-serial	Mix.-sig., binary	Neuro-morphic	Time-Domain	Digital, SC
Purpose	Train./ Infer.	Infer.	Infer.[4]	Train./ Infer.[3]	Infer.[4]	Infer.
Node	7nm	65nm	28nm	65nm	40nm	14nm
Area $[mm^2]$	19.6	16	4.6	17.6	0.124	0.5
Voltage $[V]$	0.55-0.75	0.63-1.1	0.53-0.8	0.8	0.375-1.1	0.6-0.9
Clock	1-1.6	200	1.5-10	20	0.2-3.1	250-500
Power $[mW]$		3.2-297	0.09--0.9	23.1-23.6	0.03	16 68
#MAC	32k[1]	3.5k[1]13.8k[2]	65k			19.2k
MAC /mm2	1.6k[1]	0.2k[1] 0.9k[2]	14.2k			38.4k
Mem.	8MB	256KB	328KB			167KB
Precision	FP8-32 INT4/2	1-16	1			4-8[5]
TOPS/W	8.9-16.5[1]	11.6[1]50.6[2]	532-772[2]	3.42	12.08	4.4-75
TOPS/ mm2	3.27-5.22[1]	0.086[1] 0.46[2]	0.015-0.1[2]			0.3-4.8 (0.75-12)[6]
GOPS	62K-102.4K[1]	1.4K[1]7.3[2]	72-532[2]		0.365	150-2.4k
MNIST Perf.	-	-		97.8%, 3.4 TOPS/W, 100k Fr/s	97%, 4.65-12.08 TOPS/W[4]	95.1-98.7%, 1.1-9.5, TOPS/W, 25k--215k Fr/s
CIFAR-10 Perf.	-	-	85.7-86.1% 532-772 TOPS/W, 0.04k-0.24k Fr/s	-	-	69.4-71.7%, 2-6.3 TOPS/W, 2.8k--8.3k Fr/s

[1] Int4. [2] Binary. [3] Fixed-function, MNIST only. [4] Convolutional layers only. [5] Effective precision with 8-64 bit SC streams and SC average pooling. [6] Without on-chip memory.

VI. Conclusion

We presented, to the best of our knowledge, the first stand-alone, configurable, and programmable SC NN accelerator in silicon. The chip, taped out in 14nm technology, achieves 2.4 TOPS and 75 TOPS/W throughput and energy efficiency. It has a custom ISA and supports end-to-end inference with convolutional and fully-connected layers of variable input and filter sizes. The use of SC makes it possible to pack 19,200 MAC units in a small area, enabling a high degree of spatial data reuse, amortizing the cost of SC conversion, and reducing the number of memory accesses. By varying the stream length, it enables extensive accuracy-latency trade-offs.

VII. Acknowledgments

This material is based on research sponsored by AFRL and DARPA under agreement number FA8650-18-27867.

References

[1] A. Agrawal, S. K. Lee, J. Silberman, M. Ziegler *et al.*, "A 7nm 4-Core AI Chip with 25.6TFLOPS Hybrid FP8 Training, 102.4TOPS INT4 Inference and Workload-Aware Throttling," in *2021 IEEE International Solid-State Circuits Conference-(ISSCC)*, 2021, pp. 144–145.

[2] J. Lee, C. Kim, S. Kang, D. Shin *et al.*, "UNPU: A 50.6TOPS/W Unified Deep Neural Network Accelerator with 1b-to-16b Fully-Variable Weight Bit-Precision," in *2018 IEEE International Solid - State Circuits Conference - (ISSCC)*. IEEE, 2018, pp. 218–220.

[3] W. Romaszkan, T. Li, T. Melton, S. Pamarti *et al.*, "ACOUSTIC : Accelerating Convolutional Neural Networks through Or-Unipolar Skipped Stochastic Computing," in *2020 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2020, pp. 768–773.

[4] Z. Li, J. Li, A. Ren, R. Cai *et al.*, "HEIF: Highly Efficient Stochastic Computing based Inference Framework for Deep Neural Networks," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 38, no. 8, pp. 1543–1556, 2018.

[5] D. Wu, J. Li, R. Yin, H. Hsiao *et al.*, "uGEMM : Unary Computing Architecture for GEMM Applications," *2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA)*, pp. 377–390, 2020, iSBN: 9781728146614.

[6] Y. H. Chen, T. Krishna, J. Emer, and V. Sze, "Eyeriss: An Energy-Efficient Reconfigurable Accelerator for Deep Convolutional Neural Networks," *IEEE journal of solid-state circuits*, vol. 52, no. 1, pp. 127–138, 2016, arXiv: 1512.04295 ISBN: 978-1-4673-9466-6.

[7] T. Jia, Y. Ju, and J. Gu, "A Dynamic Timing Enhanced DNN Accelerator With Compute-Adaptive Elastic Clock Chain Technique," *IEEE Journal of Solid-State Circuits*, vol. 56, no. 1, pp. 55–65, Jan. 2021. [Online]. Available: https://ieeexplore.ieee.org/document/9222216/

[8] A. Sayal, S. Fathima, S. T. Nibhanupudi, and J. P. Kulkarni, "COMPAC: Compressed Time-Domain, Pooling-Aware Convolution CNN Engine With Reduced Data Movement for Energy-Efficient AI Computing," *IEEE Journal of Solid-State Circuits*, vol. 56, no. 7, pp. 2205–2220, Jul. 2021. [Online]. Available: https://ieeexplore.ieee.org/document/9310226/

[9] D. Bankman, L. Yang, B. Moons, M. Verhelst *et al.*, "An Always-On 3.8 μ J/86% CIFAR-10 Mixed-Signal Binary CNN Processor With All Memory on Chip in 28-nm CMOS," *IEEE Journal of Solid-State Circuits*, vol. 54, no. 1, pp. 158–172, Jan. 2019. [Online]. Available: https://ieeexplore.ieee.org/document/8480105/

[10] J. Park, J. Lee, and D. Jeon, "7.6 A 65nm 236.5nJ/Classification Neuromorphic Processor with 7.5% Energy Overhead On-Chip Learning Using Direct Spike-Only Feedback," in *2019 IEEE International Solid- State Circuits Conference - (ISSCC)*. San Francisco, CA, USA: IEEE, Feb. 2019, pp. 140–142. [Online]. Available: https://ieeexplore.ieee.org/document/8662398/

[11] A. Sayal, S. Fathima, S. S. T. Nibhanupudi, and J. P. Kulkarni, "All-Digital Time-Domain CNN Engine Using Bidirectional Memory Delay Lines for Energy-Efficient Edge Computing," in *2019 IEEE International Solid- State Circuits Conference - (ISSCC)*, vol. 49. IEEE, 2019, pp. 228–230, issue: 4.