A 2.56mm^2 718GOPS Configurable Spiking Convolutional Sparse Coding Processor in 40nm CMOS

Chester Liu, Sung-Gun Cho, and Zhengya Zhang
University of Michigan, Ann Arbor, MI, USA

***Abstract*—A configurable neuro-inspired inference processor is designed as an array of neurons each operating in an independent clock domain. The processor implements a recurrent network using efficient sparse convolutions with zero-patch skipping for feedforward operations, and sparse spike-driven reconstruction for feedback operations. A globally asynchronous locally synchronous structure enables scalable design and load balancing to achieve 22% reduction in power. Fabricated in 40nm CMOS, the 2.56mm^2 inference processor integrates 48 neurons, a hub and an OpenRISC processor. The chip achieves 718GOPS at 380MHz, and demonstrates applications in feature extraction from images and depth extraction from stereo images.**

I. Introduction

Neuro-inspired sparse coding algorithms have been applied to various types of sensory inputs, including audio, image, and video, for dictionary learning and feature extraction in a wide range of applications including compression, denoising, super-resolution, and classification tasks [1]. Sparse coding implemented as a spiking recurrent neural network can be readily mapped to hardware to achieve high performance. However, as the input dimensionality increases, the number of parameters becomes impractically large, necessitating a convolutional approach to reduce the number of parameters by exploiting translational invariance [2].

In this work, we present a configurable spiking convolutional sparse coding (sCSC) processor. Compared to the popular convolutional neural networks (CNN), sCSC offers several advantages: 1) sCSC produces sparse spikes, presenting opportunities for significant complexity and power reduction; 2) sCSC preserves structural information in dictionary-based encoding, allowing downstream processing to be done directly in the encoded, i.e., compressed, domain; and 3) sCSC uses unsupervised learning, enabling truly autonomous modules that adapt to inputs.

The configurable sCSC processor is made of an array of neurons as the compute units that perform configurable convolutions. The processor implements recurrent networks by iterative feedforward and feedback operations as depicted in Fig. 1. In a feedforward operation, each neuron convolves its input or reconstruction errors, i.e., the differences between the input and its reconstruction, with a kernel. The convolution results are accumulated, and spikes are generated when the accumulated potentials exceed a threshold. In a feedback operation, neuron spikes are convolved with kernels to reconstruct the input. Depending on application, 10 to 50 iterations are required to complete one inference. The inference output, in the form of neuron spikes, are passed to a downstream post-processor to complete various tasks.

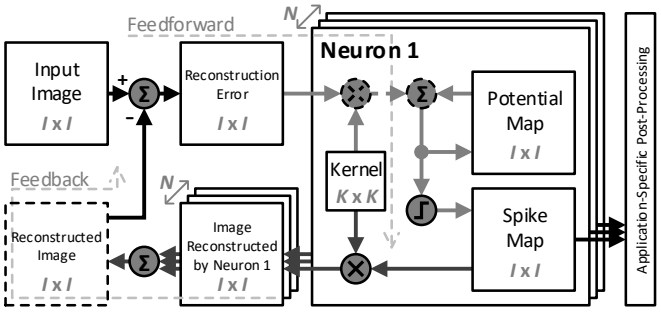

Fig. 1. Hardware mapping of spiking convolutional sparse coding (sCSC) algorithm.

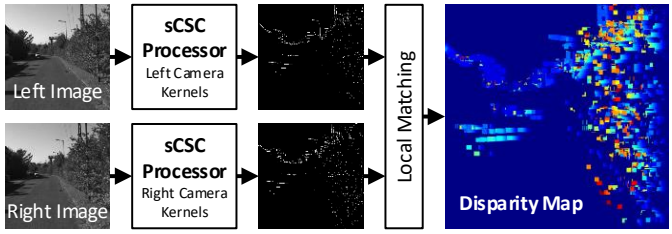

Fig. 2. sCSC processor applied to stereo images to extract depth information.

We demonstrate a configurable sCSC processor chip in 40nm CMOS. The configurable convolution architecture is more versatile than fixed architectures for specialized accelerators [3]. The design optimally exploits the inherent sparsity using zero-patch skipping to make our convolution up to 40% more efficient than the state-of-the-art constant-throughput zero masking convolution [4]. A sparse spike-driven approach is adopted in feedback operations to minimize the cost of implementating recurrence by eliminating multipliers. The sCSC processor contains 48 convolutional neurons with configurable kernel size up to 15x15, which are equivalent to 10,800 non-convolutional neurons in classic implementations [5]. Each neuron operates at an independent clock and communicates using asynchronous interfaces, enabling each neuron to run at the optimal frequency to achieve load balancing. Going beyond conventional feature extraction tasks, we apply the sCSC processor to stereo images to extract depth information as illustrated in Fig. 2. Although we only demonstrate imaging applications in this work, the sCSC processor is input-agnostic and can be applied to any type of input.

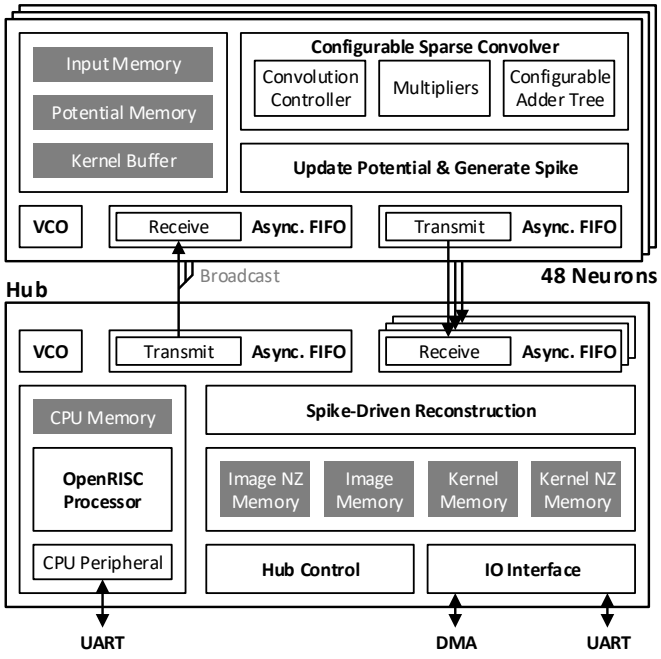

Fig. 3. sCSC processor hardware architecture.

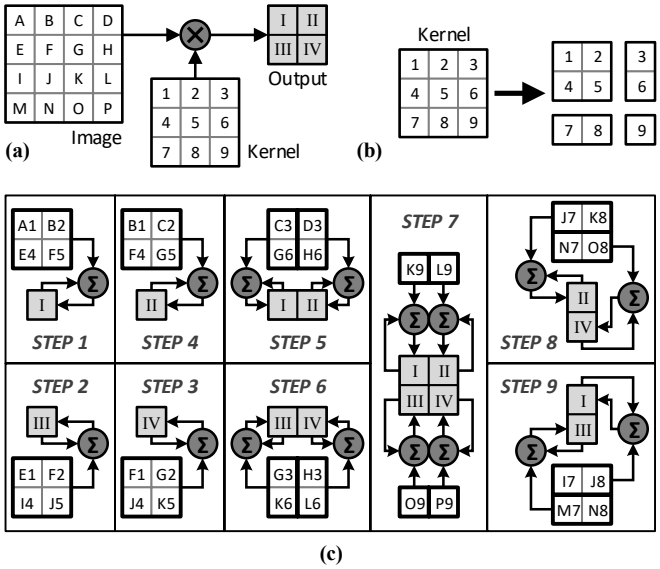

Fig. 4. (a) A 4x4 image is convolved with a 3x3 kernel to produce a 2x2 output; (b) Splitting kernel into sections that fit in the physical convolver (2x2 in this example); (c) In each step, image pixels and kernel are multiplied and accumulated in the output buffer.

II. Hardware Architecture

To implement a recurrent network for sparse coding, a modular hardware architecture is designed as shown in Fig. 3, where the feedforward operations are distributed to neurons, and the neuron spikes are sent to a central hub for feedback operations. The sparse neuron spikes make it possible to deploy efficient asynchronous interfaces and share one hub for feedback operations.

In performing a feedforward operation, a neuron convolves a typically non-sparse input image (in the first iteration) or sparse reconstruction errors (in subsequent iterations) with its kernel (Fig. 1). The feedforward convolution is optimized in three ways: 1) highest throughput for sparse input by exploiting sparsity, 2) highest throughput for non-sparse input by fully utilizing the hardware, and 3) efficient support of variable kernel size. To achieve high throughput and efficiency, we design a sparse convolver supporting zero-patch skipping; and to achieve configurability, we divide variable-sized convolution into smaller fixed-sized sections and design a traverse path for the physical convolver to assemble the complete convolution result. The design of the configurable sparse convolution is detailed in Section III.

Each neuron supports a configurable kernel of size up to 15x15 using a compact latch-based kernel buffer, and variable image patch size up to 32x32. An input image larger than 32x32 is divided into 32x32 sub-images that share overlaps to minimize edge artifacts.

In a feedback operation, neuron spikes are convolved with their kernels to reconstruct the input image (Fig. 1). A direct implementation of this feedback convolution is computationally expensive and could become a performance bottleneck. We take advantage of the binary spikes to replace all multiplications in this convolution by additions, and further make use of the high sparsity of the spikes (typically >90% sparsity) to design a sparsely activated spike-driven reconstruction to save computation and power. The design is detailed in Section IV.

The hub contains a kernel memory, and a multi-banked image memory that provides single-cycle read-accumulate-write capability. An image nonzero (NZ) memory is used to identify NZ entries in the reconstructed image to support sparse convolutions. The hub simultaneously broadcasts reconstructed image and its NZ map and receives spikes from neurons to ensure seamless feedforward and feedback operations without idling the hardware. The design of the asynchronous interfaces between the hub and neurons is detailed in Section V.

The hub uses a 16-bit bi-directional DMA interface for data I/O, and a UART interface for configuration. An OpenRISC processor is integrated on chip, and it can be tasked with on-chip learning and post-processing.

III. Configurable Sparse Convolution

Inside the neuron design is a 4x4 physical convolver. For simplicity, we illustrate in Fig. 4 our configurable convolution using the example of a 2x2 physical convolver computing the convolution of a 4x4 image with a 3x3 kernel (Fig. 4(a)): 1) the 3x3 kernel is systematically divided into sections suitable for the physical convolver, i.e., a 2x2 patch, a 2x1 line, a 1x2 line, and a 1x1 pixel (Fig. 4(b)); 2) the physical convolver scans the image in nine steps to cover all sections of the original convolution (Fig. 4(c)); and 3) intermediate results are accumulated in the output buffer to be the final result. To maximize throughput, multipliers need to be fully utilized, so the two 2x1 sections are processed together by the physical

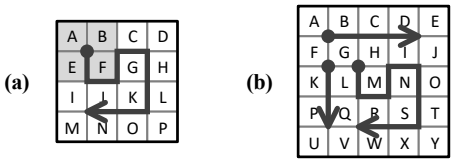

Fig. 5. Maze-walking path for (a) a 3x3 kernel on a 4x4 image and (b) a 4x4 kernel on a 5x5 image.

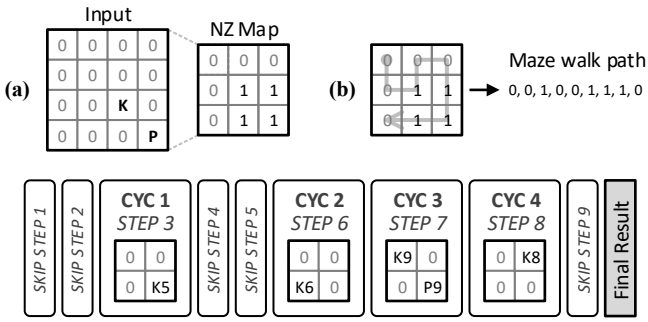

Fig. 6. (a) Entries in a NZ map indicate if at least one nonzero entry exists in a 2x2 block in the input; (b) Walking through the NZ map produces a sequence in which 0 means to skip; (c) 5 steps are skipped in calculating the convolution.

convolver in steps 5 and 6 (Fig. 4(c)). Similarly, multiple sections are processed together in steps 7 to 9 (Fig. 4(c)). The physical convolver is equipped with a configurable adder tree to handle various forms of accumulation in different steps.

To maximize locality of reference, kernel sections are fetched once and reused until done, and image sections are shifted by one row or column between steps. Such a carefully arranged sequence results in a maze-walking path, depicted in Fig. 5(a), that maximizes hardware utilization and data locality. We name the convolution "maze convolution". An optimal path exists for every kernel size; yet, to minimize storage, paths for larger kernels are created with multiple smaller paths. Fig. 5(b) shows a maze-walking path for a 4x4 kernel based on the 3x3 kernel's maze-walking path.

A sparse input enables the use of a sparse convolver to increase throughput and efficiency. In [3], a line convolver was designed to skip lines of zeros in the input. However, we observed that it is more likely to have a patch of zeros than a line of zeros in the input, so skipping zero patches is more effective. Maze convolution readily supports zero-patch skipping with the help of an input NZ map, wherein a NZ bit is 1 if at least one nonzero entry is detected in an area covered by a patch of the same size as the physical convolver. Fig. 6 shows an example in which the NZ map of an image contains two nonzero entries. Maze convolution is guided by the NZ map, skipping steps where the NZ bit is 0 to realize sparsity-proportional throughput increase. Compared to [3], maze convolution with zero-patch skipping increases the throughput by up to 40% at 90% input sparsity. The proposed maze convolution with zero patch skipping is equally applicable to deep neural networks [4], [6].

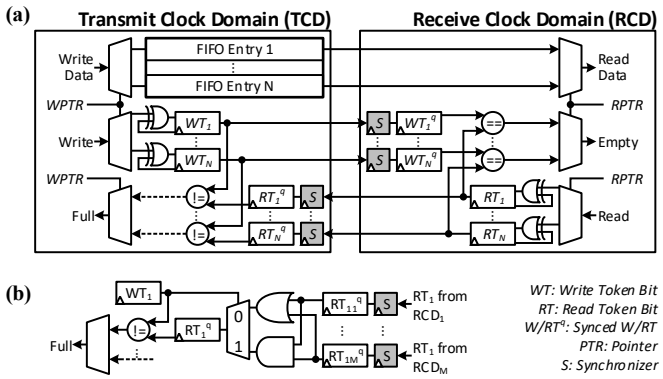

Fig. 7. (a) Token-based asynchronous FIFO; (b) FIFO full condition check for broadcast asynchronous FIFO.

IV. Spike-Driven Reconstruction

Triggered by a neuron's spike, the hub performs reconstruction by retrieving the neuron's kernel from the kernel memory and accumulating the kernel in the image memory, with the kernel's center aligned to the spike location. Like in maze convolution, a kernel is also divided into sections to support variable kernel size in the spike-driven reconstruction. The NZ map of the reconstructed image is computed by OR'ing the NZ map of the retrieved kernels, saving both computation and latency compared to the naive way of scanning the reconstructed image. The spike-driven reconstruction eliminates the need to store spike maps, and a 16-entry FIFO is sufficient for buffering spikes, cutting the storage by 2.5x.

V. Globally Asynchronous Interfaces

The sCSC processor implements globally asynchronous communication between the hub and neurons to achieve scalability by breaking a single clock network with stringent timing constraints into small ones with relaxed constraints. The globally asynchronous scheme further enables load balancing by allowing the hub and individual neurons to run at the optimal clock frequencies based on workload. Following feed-forward operations, neurons send 10-bit messages to identify neuron spikes to the hub via a token-based asynchronous FIFO [7]. Following a feedback operation, the hub sends 128-bit messages that contain reconstructed image and NZ map to the neurons. To avoid routing congestion from the hub to the neurons, we design a broadcast asynchronous FIFO, which is identical to the token-based asynchronous FIFO except for the FIFO full condition check logic.

The asynchronous FIFO design is shown in Fig. 7. The token-based asynchronous FIFO is full when the transmit clock domain (TCD) write token disagrees with the synchronized receive clock domain (RCD) read token. The broadcast asynchronous FIFO has multiple RCDs and it is full when the TCD write token disagrees with any synchronized RCD read token. Synchronizer stage in all asynchronous FIFOs are configurable between 2 and 4 stages to accommodate PVT-induced delay variations.

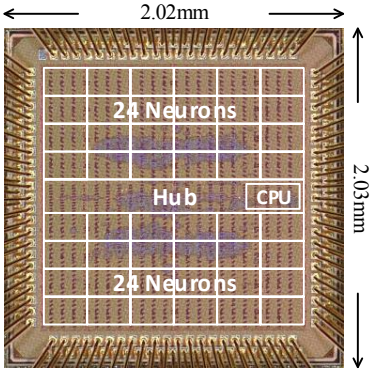

Fig. 8. Chip microphotograph.

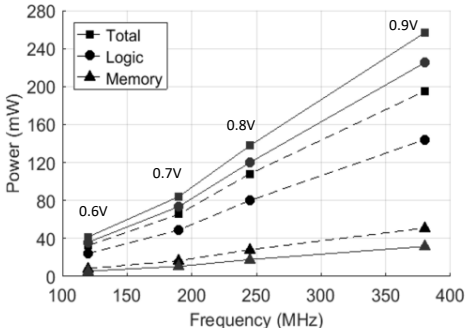

Fig. 9. Chip power measurement result of feature extraction task (dashed lines) and depth extraction task (solid lines).

VI. Chip Implmentation and Measurements

A 4.1mm^2 test chip is implemented in 40nm CMOS, and the sCSC processor occupies 2.56mm^2. The chip microphotograph is shown in Fig. 8. We use a mixture of 80.5% high-V_T and 19.5% low-V_T cells to reduce the chip leakage power by 33%. Dynamic clock gating is applied to reduce the dynamic power by 24%. A balanced clock frequency setting for the hub and neurons further reduces the overall power by an average of 22%. A total of 49 VCOs are instantiated, with each VCO occupying only 250um^2 area. The test chip achieves 718GOPS at 380MHz with a nominal 0.9V supply at room temperature. An OP is defined as an 8-bit multiply or a 16-bit add.

We use two sample applications to demonstrate the sCSC processor: extracting sparse feature representation of images and extracting depth information from stereo images. The feature extraction task is entirely done by the hub and neurons; and the depth extraction task requires an additional local matching post-processing programmed on the on-chip OpenRISC processor. When performing feature extraction using 7x7 kernels, 10 recurrent iterations, and a target sparsity of approximately 90%, the sCSC processor achieves 24.6M pixel/s (equivalent to 375 256x256 frames per second), while consuming 195mW (shown in dashed lines in Fig. 9). In performing depth extraction using 15x15 kernels, 10 recurrent iterations, and a target sparsity of approximately 80%, the sCSC processor achieves 7.68M pixel/s (equivalent to 117 256x256 frames per second) while consuming 257mW (shown in solid lines in Fig. 9). Compared to the optimal baseline designs without exploiting sparsity, the throughputs of the tasks are improved by 7.7x and 9.7x, respectively. Voltage and frequency scaling measurement shows that at 0.6V supply and 120MHz clock frequency, the chip power is reduced to 53.9mW for the feature extraction task and 69.3mW for the depth extraction task.

Table 1. Comparison with prior works

	This Work	JSSC 2016 [4]	ISSCC 2017 [6]
Architecture	Recurrent	Feedforward	Feedforward
Technology	40nm GP CMOS	65nm LP CMOS	28nm UTBB FD-SOI
Core Area	1.6mm x 1.6mm	3.5mm x 3.5mm	1.29mm x 1.45mm
SRAM Size	120KB	181.5KB	144KB
Voltage	0.9V	1V	1V
Frequency	380MHz	200MHz	200MHz
Performance	718GOPS[*1]	33.6GOPS[*2]	204GOPS[*1]
Throughput	256x256 @117fps	227x227 @35fps	227x227 @47fps
Power	257mW	278mW	44mW
Power Efficiency	2.79 TOPS/W	0.12 TOPS/W	4.64 TOPS/W
Area Efficiency[*3]	280 GOPS/mm^2	7.24 GOPS/mm^2	53.4 GOPS/mm^2

[*1]8-bit operation [*2]16-bit operation [*3]Normalized to 40nm

Fig. 10. Comparison with prior works.

Compared to state-of-the-art inference processors based on feedforward only networks, our sCSC processor realizes a recurrent network, supports unsupervised learning, and demonstrates expanded functionalities including depth extraction from stereo images, while still achieving competitive performance and efficiency in power and area as shown in Fig. 10.

Acknowledgment

This work was supported in part by SONIC, DARPA UPSIDE and Intel Corporation. The authors thank Prof. Bruno Olshausen and Dr. Garrett Kenyon for advice.

References

[1] C. J. Rozell, D. H. Johnson, R. G. Baraniuk, and B. A. Olshausen, "Sparse coding via thresholding and local competition in neural circuits," *Neural Compution*, vol. 20, no. 10, pp. 2526–2563, Oct 2008.

[2] F. Heide, W. Heidrich, and G. Wetzstein, "Fast and flexible convolutional sparse coding," in *IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, Jun 2015, pp. 5135–5143.

[3] P. Knag, C. Liu, and Z. Zhang, "A 1.40mm^2 141mW 898GOPS sparse neuromorphic processor in 40nm CMOS," in *IEEE Symp. VLSI Circuits*, Jun 2016, pp. 180–181.

[4] Y.-H. Chen, T. Krishna, J. S. Emer, and V. Sze, "Eyeriss: an energy-efficient reconfigurable accelerator for deep convolutional neural networks," *IEEE J. Solid-State Circuits*, vol. 52, no. 1, pp. 127–138, Nov 2016.

[5] J. K. Kim, P. Knag, T. Chen, and Z. Zhang, "A 640M pixel/s 3.65mW sparse event-driven neuromorphic object recognition processor with on-chip learning," in *IEEE Symp. VLSI Circuits*, Jun 2016, pp. 50–51.

[6] B. Moons, R. Uytterhoeven, W. Dehaene, and M. Verhelst, "Envision: A 0.26-to-10TOPS/W subword-parallel dynamic-voltage-accuracy-frequency-scalable convolutional neural network processor in 28nm FD-SOI," in *IEEE Int. Solid-State Circuits Conf. (ISSCC)*, Mar 2017, pp. 246–247.

[7] I. M. Panades and A. Greiner, "Bi-synchronous FIFO for synchronous circuit communication well suited for network-on-chip in GALS architectures," in *Int. Symp. Networks-on-Chip (NOCS)*, May 2007, pp. 83–94.