Efficient Hardware Architecture for Sparse Coding

Jung Kuk Kim, *Student Member, IEEE*, Phil Knag, *Student Member, IEEE*, Thomas Chen, and Zhengya Zhang, *Member, IEEE*

***Abstract*—Sparse coding encodes natural stimuli using a small number of basis functions known as receptive fields. In this work, we design custom hardware architectures for efficient and high-performance implementations of a sparse coding algorithm called the sparse and independent local network (SAILnet). A study of the neuron spiking dynamics uncovers important design considerations involving the neural network size, target firing rate, and neuron update step size. Optimal tuning of these parameters keeps the neuron spikes sparse and random to achieve the best image fidelity. We investigate practical hardware architectures for SAILnet: a bus architecture that provides efficient neuron communications, but results in spike collisions; and a ring architecture that is more scalable, but causes neuron misfires. We show that the spike collision rate is reduced with a sparse spiking neural network, so an arbitration-free bus architecture can be designed to tolerate collisions without the need of arbitration. To reduce neuron misfires, we design a latent ring architecture to damp the neuron responses for an improved image fidelity. The bus and the ring architecture can be combined in a hybrid architecture to achieve both high throughput and scalability. The three architectures are synthesized and place-and-routed in a 65 nm CMOS technology. The proof-of-concept designs demonstrate a high sparse coding throughput up to 952 M pixels per second at an energy consumption of 0.486 nJ per pixel.**

***Index Terms*—Algorithm and architecture co-optimization, hardware acceleration, neural network architecture, sparse and independent local network, sparse coding.**

I. Introduction

BETTER understanding of the mammalian primary visual cortex has led to advances in computer vision [1], [2]. The visual cortical neurons respond to visual stimuli with spikes. The visual feature or region that stimulates a cortical neuron in the visual cortex is known as the receptive field of the neuron [3]–[5]. (In this work, we consider receptive field to be excitatory [3]–[5], but in general, the receptive field of a neuron can also be inhibitory.) The receptive fields of the visual cortical neurons can be compared to the basis functions that form the natural images we see, and are closely related to the intrinsic structures of natural images. Learning the receptive fields and neuron activities allows us to carry out many complex vision processing tasks, including efficient encoding of images and detecting features and objects [6], [7].

Much progress has been made in training unsupervised machine learning algorithms using natural images to build receptive fields that resemble the receptive fields of the primary visual cortex. Among the most promising candidates are the sparse coding algorithms [8]–[13] that learn to represent natural images using a small number of receptive fields. The Sparsenet algorithm by Olshausen and Field [8] attempts to minimize the mean neuron activity in learning the representation of natural images, and it is shown to reproduce the receptive fields that match the key qualitative features of the receptive fields of the primary visual cortex. The sparse-set coding (SSC) network by Rehn and Sommer [11] tries to minimize the number of active neurons, and it successfully predicted the distribution of receptive field shapes found in the primary visual cortex of cat and monkey.

Sparse coding algorithms can be mapped to a biologically inspired network of computing nodes, or "neurons". Foldiak proposed a network of model neurons with feed-forward connections between neurons and stimulus, and inhibitory feedback connections between neurons [9]. The feed-forward connection weights are updated by the Hebbian rule [14] to strengthen a feed-forward connection when an input pattern matches the receptive field; and the feedback connection weights are updated by the anti-Hebbian rule to suppress correlated neuron activities and enforce sparse activation. The locally competitive algorithm (LCA) by Rozell *et al.* [12] is naturally mapped to a network similar to what Foldiak proposed. In Rozell's network, a model neuron's membrane potential charges up in response to input stimulus at a rate depending how well the input pattern matches the neuron's receptive field, and when the potential exceeds a threshold, the neuron emits an action potential to inhibit neighboring neurons. LCA was shown to perform the optimal sparse approximation that minimizes the mean squared error (MSE) of image reconstruction and a sparsity cost function. The parallel network of neurons implementation is appealing, but the model neurons in Foldiak's network were designed to communicate with analog signals. Also, the weight updates in Rozell's network are performed offline by costly global computations.

Recently, Zylberberg *et al.* proposed sparse and independent local network (SAILnet) algorithm to perform sparse coding using spiking neurons and local update rules [13]. The SAILnet algorithm was demonstrated to learn the full diversity of the primary visual cortex simple cell receptive field shapes when

Manuscript received February 18, 2014; accepted June 10, 2014. Date of publication June 27, 2014; date of current version July 18, 2014. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. John McAllister. This work was supported by Defense Advanced Research Projects Agency (DARPA) under cooperative agreement HR0011-13-2-0015. The views expressed are those of the author(s) and do not reflect the official policy or position of the Department of Defense or the U.S. Government. Approved for public release, distribution unlimited.

The authors are with the Department of Electrical Engineering and Computer Science, University of Michigan, Ann Arbor, MI 48109-2122 USA (e-mail: jungkook@umich.edu; knagphil@umich.edu; tcchen@umich.edu; zhengya@eecs.umich.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TSP.2014.2333556

trained on natural images. The SAILnet algorithm enables a fundamentally more efficient mapping to a network of spiking neurons that uses only local computation in weight updates. The spiking neural network consists of a set of interconnected simple computing neurons that communicate using binary spikes. In contrast to modern multi-core von Neumann processors that are optimized for sequential tasks and often limited by interconnect and memory bandwidth, the biologically inspired spiking neural network is an inherently parallel array of self-adapting computational units that are ideally suited for computer vision processing. This work aims at building a dedicated sparse coding hardware to be used as the front-end vision processor to achieve both high throughput and energy efficiency, while maintaining a high image fidelity. The hardware will be fully integrated with both inference and on-chip learning capability with low power and silicon area, so that it can be applied to embedded vision processing.

The implementation of a sparse coding hardware can leverage a large body of prior work on neural network implementations. The direct mapping of neural network onto VLSI hardware [15] is not scalable due to the overwhelming interconnect and memory bandwidth necessary to support the full connectivity between neurons and the access to synaptic weights. More scalable architectures including bus [16], ring [17] and array [18] solved the interconnect bottleneck, and the invention of address-event representation (AER) [19], [20] enables the efficient time-multiplexing of sparse neuron spikes on a shared bus. Since then, much progress has been made in the key challenging areas of compact synaptic weight storage [21]–[23], efficient neuron and synapse circuits [21]–[25], and scalable synaptic connections [22], [23], [25]–[28]. The latest wave of new hardware designs have demonstrated increasing capabilities, from simulating real-time spike-timing-dependent plasticity [24] and cortical circuits [21] to digit recognition [22] and pattern recognition [23], from multilayer vision sensing and actuation [26] to performing arbitrary mathematical computations [25] and simulating neuroscience experiments [27]. At the same time, the integration scale has gone from tens of neurons [24] to over 10 K neurons [26] and over 10 M synaptic connections [25], and the power consumption has been lowered to mW level [23] and the energy reduced to tens of pJ/spike [22]. However, some of the latest works are not directly applicable to sparse coding due to the mismatch of learning rules and our requirement of entirely on-chip learning capability. The general-purpose solutions are applicable, but they are not tailored to sparse coding algorithm, thus the energy and area efficiency will be sacrificed.

In this paper, we design custom hardware for the SAILnet sparse coding algorithm [13], and focus on a synchronous digital implementation which exhibits robust deterministic logical behavior at nominal operating conditions [23]. Alternative designs including analog and asynchronous approaches offer unique advantages and are also expected to affect algorithm dynamics. They remain our future work and will not be discussed in this paper.

The SAILnet algorithm is unique as it constrains neuron spikes to be very sparse and uncorrelated [9], [13]. The unique characteristic enables more efficient hardware architectures. On the other hand, the hardware architecture also regulates the dynamics of the sparse coding algorithm, thereby affecting the fidelity of sparse coding. This paper presents a hardware-algorithm co-design for sparse coding, and its technical contributions are two folds: (1) a better understanding of the sparse coding algorithm, including algorithm dynamics and tuning parameters, which translate to hardware performance and complexity; and (2) custom hardware design that is tailored to the sparse coding algorithm to achieve the highest throughput and energy efficiency.

We consider bus and ring architectures for sparse coding based on prior works, but our emphasis on hardware-algorithm co-design goes beyond prior works. It is known that a bus architecture [16], [29] provides an efficient neuron communication and a high throughput, but it results in spike collisions. Though collisions negatively impact the fidelity of sparse coding, we find that their occurrences are vastly reduced with a sparse spiking network and a sufficiently small neuron update step, so the collisions can be tolerated without requiring arbitration. The arbitration-free bus operates at the same clock speed as the neurons, saving significant area and power compared to the conventional synchronous AER bus [21], [24]–[26] that requires arbitration and a higher bus speed to serve neuron requests in a timely manner. It is also known that a ring architecture [17] eliminates spike collisions altogether, but neurons communicate in a serial fashion along the ring, delaying inhibitions and causing neurons to misfire. By damping neuron responses to reduce neuron misfires, we show that the fidelity of sparse coding can be optimally tuned. The resulting latent ring architecture is more scalable than the bus architecture. Combining the bus and ring into a hybrid architecture of multiple small buses connected in a short ring reduces the collision rate and communication latency for the optimal sparse coding performance and throughput.

We demonstrate through synthesis and physical place-and-route three 512-neuron networks in a 65 nm CMOS technology, one implemented in an arbitration-free bus architecture, one implemented in a latent ring architecture, and the other implemented in a hybrid bus-ring architecture. The bus architecture is more area efficient, but the ring architecture runs at a higher frequency. The hybrid bus-ring architecture takes advantage of both the bus and the ring architecture for a high throughput of 952 M pixels per second (Mpx/s) at 0.486 nJ/px. These proof-of-concept designs demonstrate the high throughput and efficiency that can be achieved in practical implementations of the SAILnet sparse coding algorithm.

II. Background

We first review the two operation phases, inference and learning, of a sparse coding algorithm implemented in a neural network. In the learning phase, a spiking neural network is trained using images to extract a library of features. During learning, the neural network updates the connection weights to optimally adapt to the operating environment. Training image pixels excite the neurons, which then generate binary (1 or 0) spikes [9], [10], [13] that are propagated through the neural network. When a neuron spikes, it updates its feed-forward

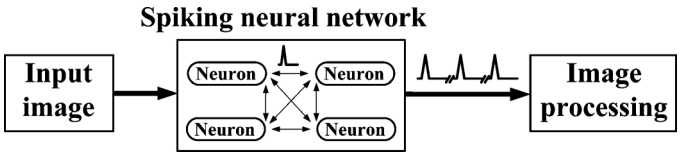


Fig. 1. A spiking neural network for inference.

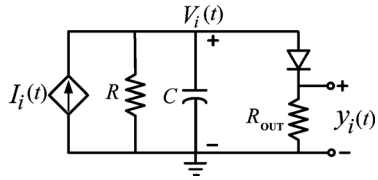


Fig. 2. Integrate-and-fire neuron model.

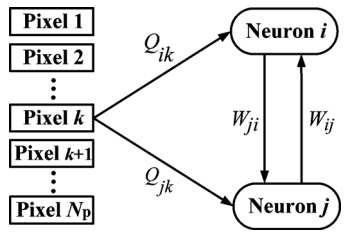


Fig. 3. Feed-forward connection between neuron and pixel, and feedback connection between neurons.

connection weight. When a neuron sees a spike from a neighboring neuron, it updates the feedback connection weight. The updates are based on a learning rule. Upon convergence, the neural network will have finalized a library of feed-forward connection weights, which resemble the receptive fields, as well as the feedback connection weights that regulate the interactions between neurons.

Learning is very compute-intensive as it involves weight updates, but learning is done infrequently – in the beginning for setting up the weights and occasionally to update the weights to accommodate changes in the operating environment. The decision to switch learning on and off is made by a controller that is external to the sparse coding processor, e.g., if the sparse coding processor is integrated as part of a vision system, the system decides whether to switch learning on or off. More specifically, when the sparse coding processor is moved to a new operating environment, e.g., a different terrain, it needs to update the receptive fields to reflect the new operating environment. There is no real-time constraint on learning and its power budget is not the most critical due to its infrequent activation.

In the inference phase, the neural network receives an input image and responds by neuron spikes that correspond to the receptive fields that are activated, as illustrated in Fig. 1. Using a sparse coding algorithm, such as SAILnet [13], spikes will be kept very sparse, thus the neural network will be capable of encoding an image using a sparse set of receptive fields. Tasks including image reconstruction, target extraction and tracking can be performed based on the neuron firing and the receptive fields.

Inference is also compute-intensive, but to a lesser extent compared to learning, because it does not perform weight updates. However, inference needs to be done in real time. Furthermore, inference is always on and its power consumption needs to be minimized. In this paper, we develop optimized algorithm and hardware architecture to reduce the implementation cost and power consumption of a sparse coding hardware for inference. As inference shares the same hardware as learning, the efficiency of learning is also improved.

A. Spiking Neuron Model

A biological neuron in the visual cortex receives stimuli from visual inputs and other neurons in the network in the form of electrical signals. The received stimuli will increase or decrease the neuron's membrane potential. The neuron fires an action potential, or spike, when its membrane potential reaches a threshold value [30]. After the firing, the neuron resets its membrane potential for the next firing.

Fig. 2 describes a simple passive or leaky integrate-and-fire (IF) neuron model, including a current source $I_i(t)$ and a parallel RC circuit [30]–[32]. The current source $I_i(t)$ in a neuron is determined by the inputs and the activities of other neurons in the network along with feed-forward and feedback connection weights. The current $I_i(t)$ is mathematically formulated as a continuous-time function.

$$I_i(t) = \frac{1}{R}\left(\sum_{k=1}^{N_p} Q_{ik}X_k - \sum_{j\neq i} W_{ij}s_j(t)\right), \tag{1}$$

where X_k denotes an input pixel value, $s_j(t)$ represents the spike train generated by neuron j ($s_j(t) = 1$ if neuron j fires at time t, and $s_j(t) = 0$ otherwise). Q_{ik} is the weight of the feed-forward connection between input pixel k and neuron i, and W_{ij} is the weight of the feedback connection from neuron j to neuron i, as labeled in Fig. 3. N_p is the number of pixels. Equation (1) can be interpreted as that the input stimuli increase the current (an excitatory effect) and the neighboring neuron spikes decrease the current (an inhibitory effect).

The voltage $V_i(t)$ across the capacitor C represents a neuron's membrane potential. The resistor R in parallel with the capacitor models the membrane resistance. While the current source $I_i(t)$ charges up the capacitor and increases the membrane potential, some current leaks through R. The following equation describes the leaky integration of the membrane potential.

$$C\frac{dV_i(t)}{dt} = I_i(t) - \frac{V_i(t)}{R}. \tag{2}$$

When $V_i(t)$ exceeds a threshold voltage θ_i, set by the diode, the neuron output $y_i(t)$ emits a spike, or a spike train $s_i(t)$ over time. After firing, the capacitor is discharged through a small R_{out}, i.e., $R_{\text{out}} \ll R$, to reset $V_i(t)$. Note that the spiking neural network described above uses binary spikes to communicate between neurons, different from a non-spiking neural network [11], [12] or a spiking neural network that relies on analog voltage or current as the way to communicate between neurons [9], [10].

For simulation and digital implementation of the neuron, it is customary to discretize the continuous-time voltage and current equations to

$$V_i[n+1] = V_i[n] + \eta\left(\sum_{k=1}^{N_p} Q_{ik}X_k - \sum_{j\neq i} W_{ij}s_j[n] - V_i[n]\right), \tag{3}$$

where $\eta = \Delta t/\tau$ is the neuron update step size, and $\tau = RC$ is the neuron time constant. η is Δt normalized by the time constant τ.

B. Sailnet Learning Rule

In the learning phase, the firing threshold θ_i and the weights W_{ij} and Q_{ik} are constantly updated according to a learning rule. The SAILnet learning rule [13] provides a good performance and is also shown to successfully produce key features of receptive fields in mammalian visual cortex. The SAILnet learning rule enforces sparse and independent neuron spiking, and it involves only local computations, both of which are implementation-friendly features. In the following, we briefly introduce the SAILnet learning rule. In response to an input image X, neurons in the network generate spikes, and the spikes can be used to reconstruct the input image based on the learned dictionary of receptive fields. The dictionary of receptive fields Q is simply the set of feed-forward connection weights: $Q = [Q_1, Q_2, \ldots, Q_N]$ where N is the total number of neurons in the network and $Q_i = [Q_{i1}, Q_{i2}, \ldots, Q_{iN_p}]^{\mathrm{T}}$ are feed-forward connection weights associated with the connections between neuron i and each pixel of the input image patch. The reconstructed image $\hat{X}$ is obtained by the linear combination of the dictionary elements [13], known as the linear generative model.

$$\hat{X} = \sum_{i=1}^{N} s_i Q_i, \tag{4}$$

where s_i denotes the number of spikes generated by neuron i in response to an input image, collected over an inference window w. Given a neuron update step size of η, the number of update steps $n_s = w/(\eta\tau)$. With these, $s_i = \sum_{n=1}^{n_s} s_i[n]$, where $s_i[n]$ is either 1 or 0, indicating whether neuron i has spiked at time step n. Using the SAILnet learning rule [13], the spikes are kept sparse, and the majority of the terms in the summation of (4) are zero.

The SAILnet algorithm minimizes the mean squared error (MSE) between the input image X and the reconstructed image $\hat{X}$ while satisfying the constraints of sparse and decorrelated neural activities across the network [13]. The two constraints are justified by the experiments in [33], [34]. The solution to the constrained optimization problem is the SAILnet learning rule that governs the iterative updates of the firing thresholds, feedback connection weights, and feed-forward connection weights:

$$\begin{aligned}
\theta_i^{(m+1)} &= \theta_i^{(m)} + \alpha(s_i - p),\\
W_{ij}^{(m+1)} &= W_{ij}^{(m)} + \beta(s_i s_j - p^2),\\
Q_{ik}^{(m+1)} &= Q_{ik}^{(m)} + \gamma s_i(X_k - s_i Q_{ik}^{(m)}).
\end{aligned} \tag{5}$$

where α, β, γ are tuning parameters, and p is the target firing rate, or the number of spikes per image, which is set to a low value, and m and $m+1$ indicate the current and the next update iteration. Note that the above learning rule is local: neuron i updates the firing threshold θ_i and the feed-forward connection weight Q_{ik} based on its own spike count, and it updates the feedback connection weight W_{ij} based solely on the activity of the pair of neurons that are associated with the connection. For a detailed derivation of these learning rules, see [13].

In the learning phase, input stimuli and neuron spikes trigger updates of firing thresholds, feedback connection weights and feed-forward connection weights following (5). In the inference phase, the thresholds and weights are all fixed.

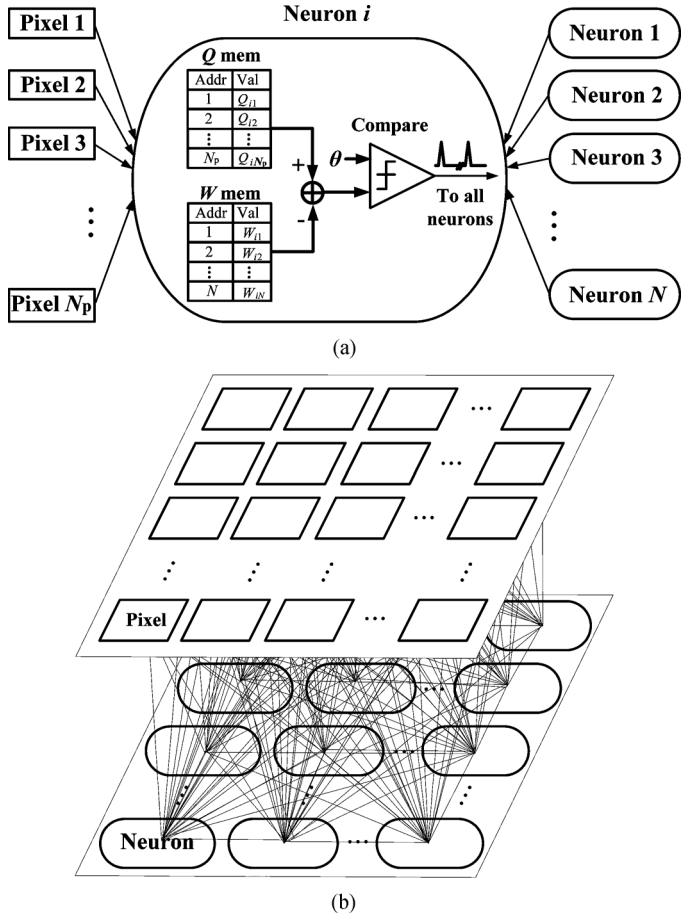


Fig. 4. (a) A digital neuron design, and (b) a fully connected network for sparse coding.

III. Hardware Implementation Challenges

A binary spiking and discrete-time neuron model described above makes it possible to design a simple digital neuron, as illustrated in Fig. 4(a). The neuron is connected to N_p input pixels and $N-1$ neighboring neurons through point-to-point links. The neuron contains two memories: Q memory to store feed-forward connection weights (N_p entries in total, one per each pixel) and W memory to store feedback connection weights ($N-1$ entries in total, one per each neighboring neuron). The neuron performs leaky integrate-and-fire described in (3) in response to pixel inputs and neuron spikes.

The size of the neural network depends on the size of the input image patch. For example, to detect features in a 16×16

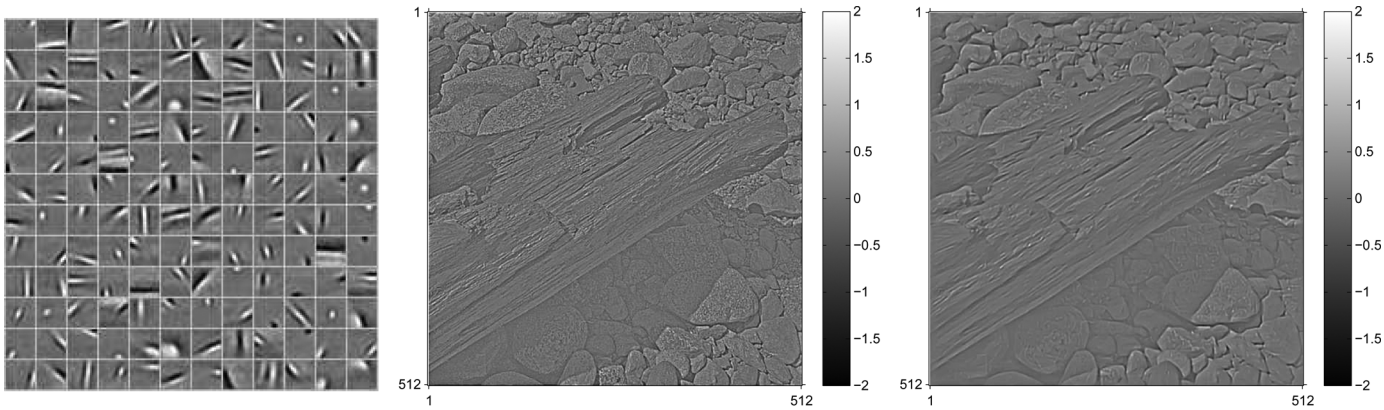


Fig. 5. (a) 144 randomly selected receptive fields (each square in the grid represents a 16×16 receptive field), (b) whitened input image, and (c) reconstructed image using sparse code.

image patch of 256 pixels, at least a $2\times$ overcomplete network of 512 neurons is needed. Each neuron in this network requires a 256-entry Q memory and a 511-entry W memory, which easily dominate the size of the digital neuron. To implement the fully connected network [15], each neuron needs to be connected to 256 pixel inputs and 511 neighboring neurons. The fully connected network provides the highest throughput of f_{clk}/n_s 16×16 image patches per second, or $256f_{clk}/n_s$ pixels/s (px/s), where f_{clk} is the clock frequency, n_s is the number of neuron update steps for each inference. The fully connected network as illustrated in Fig. 4(b) is however impractical due to the high interconnection overhead that grows at N^2 for a network of N neurons.

Regardless of the practicality, the fully connected network is often the underlying assumption of neural network algorithms. The performance of the algorithms using the fully connected network can be simulated in software to provide the baseline reference. In Fig. 5, we show the performance of a 512-neuron network in feature detection and image reconstruction after it has been trained using the SAILnet learning rule. In the beginning of each training, the W weights are set to zero, and the Q weights are initialized with Gaussian white noise. Fig. 5(a) shows 144 sample neurons' receptive fields (Q weights) that are obtained after learning. Fig. 5(b) is the input 512×512 whitened natural image from [35]. Whitening of natural images is an operation that flattens the amplitude spectrum. Whitening is done by computing the 2D FFT of the original images, passing through a 2D ramp filter (since the amplitude spectrum of natural images tends to be $1/f$, where f is the frequency), followed by 2D IFFT [6], [36]. The whitened image is divided to 16×16 patches as the inputs to the neural network for inference. Fig. 5(c) is the reconstruction of the image using the linear generative model (4) with the neuron spikes obtained from the fully connected network and the neurons' receptive fields (Q weights). The fidelity of sparse coding is measured by the error in the reconstructed image by the linear generative model. In the following we will use root-mean-square error (RMSE) as the image fidelity metric.

IV. Dynamics of Sparse Coding Algorithm

Sparse coding implemented in a neural network is a dynamic system, and its learning and inference are dependent on the neuron update step size η and the target neuron firing rate p. η controls the step size of the neuron potential update. The smaller the η, the more the number of time steps and the closer the discrete-time system mimics a continuous-time system for a higher accuracy, but the longer it takes for learning to converge, so intuitively η determines the tradeoff between accuracy and throughput. p controls the target firing rate, and is set to a low value to maintain sparse firing. A low p is also appealing as the sparse firing results in sparse communication and low power consumption. However, the sparseness is relative to the neural network size. A small network is not likely to support a low firing rate.

We analyze the influence of η and p and relate them to the neuron spike rate pattern that underpin the performance of the SAILnet algorithm. In the following experiments, we first train the network based on given η and p values, and then perform inference using the trained network. A fully connected network is assumed. The focus here is on the dynamics of the network when it is performing inference, consistent with the motivation outlined previously.

A. Spike Rate Pattern

The neuron spike rate pattern is depicted in Fig. 6 as the average spike rate p_s at each time step across a network of 512 neurons when performing inference with a target firing rate of $p = 0.04$. η is varied between 2^{-3} to 2^{-6}, and the inference window w is set to 3τ. η determines the number of time steps in the window w, i.e., $n_s = w/(\eta\tau)$. For example, if $\eta = 2^{-5}$, $n_s = 96$. Suppose a fully connected digital neural network operating at a clock frequency of f_{clk}, the sparse coding throughput is f_{clk}/n_s image patches per second. The throughput of the network is inversely proportional to n_s, and thus proportional to η.

To measure the spike rate, the neuron potential is initialized to 0 before performing each inference. When presented with an input image patch, some neurons will start to charge up and fire,

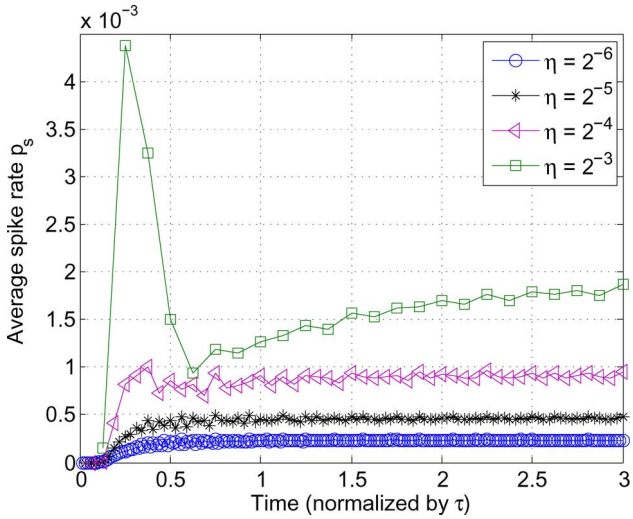


Fig. 6. Average spike rate at each time step across a network of 512 neurons when performing inference with a target firing rate of $p = 0.04$.

leading to a rise of the spike rate until it starts to settle to a steady state. The target firing rate p determines the average spike rate summed over the inference window w, i.e., $p \approx \sum_{i=0}^{n_s} p_s[i]$, where $p_s[i]$ is the spike rate at time step i. A larger step size η means fewer time steps n_s, and a higher spike rate in each time step to meet the a given target firing rate p.

Varying η has a pronounced effect on the spike rate pattern, as seen in Fig. 6. A large η results in a large peaking of the spike rate, which is attributed to the quick rise of neuron potentials: many neurons fire together after a few initial steps and then inhibitions take effect to silence most of the neurons. The bursts of neuron spikes are not desirable for implementation, because they lead to competitions for hardware resources and communication bandwidth. A small η results in a sparse, random, and more evenly distributed spike rate over time, and a more efficient utilization of hardware resources.

The influence of η can be understood by the distribution of the neuron firing threshold and the distribution of the feedback connection weights (inhibitory weights). The neuron firing threshold controls the sparseness of neuron spikes [13]: the higher the threshold, the more difficult it is to reach the threshold, thus fewer spikes are expected. The feedback connection weight controls the correlation between neuron spikes [13]. A higher positive feedback connection weight indicates a strong inhibition – when one neuron spikes, the other neurons will be strongly inhibited to de-correlate spike activities. Fig. 7(a) shows the cumulative distribution function (CDF) of the neuron firing threshold, and Fig. 7(b) shows the CDF of the feedback connection weights for different η values. A smaller η results in higher neuron firing thresholds and feedback connection weights, which confirm the observation that a smaller η produces more sparse and random spikes that are amenable for an efficient implementation.

The smaller update step size improves the fidelity of sparse coding, as evidenced in the lower RMSE in image reconstruction by the linear generative model, as shown in Fig. 8. Therefore it is advantageous to choose the smallest η that meets the throughput requirement.

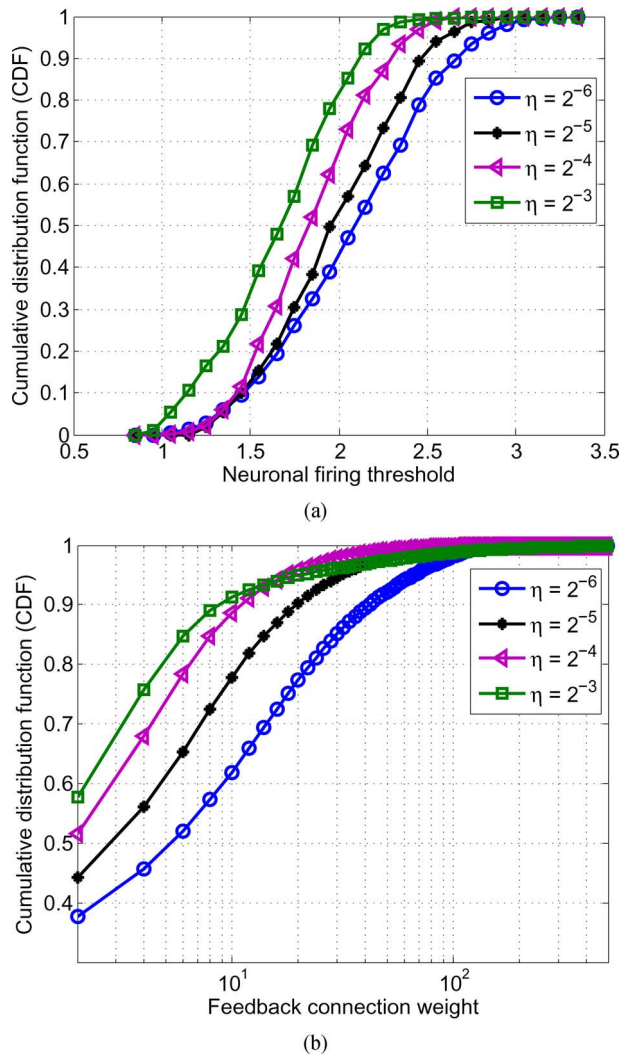


Fig. 7. (a) CDF of neuron firing thresholds, and (b) CDF of feedback connection weights for different η values.

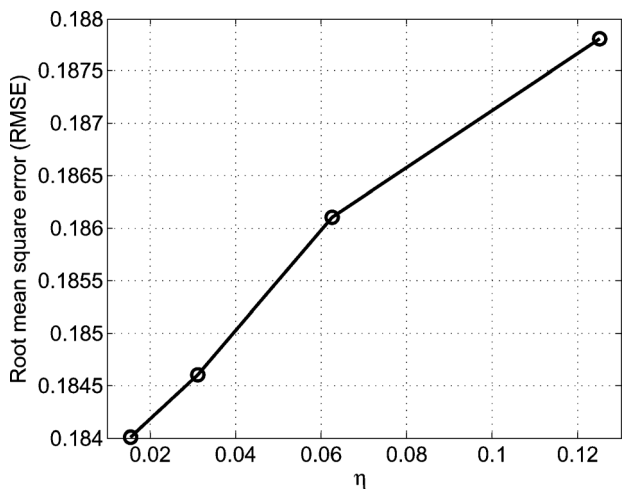


Fig. 8. RMSE of the image reconstruction by sparse code using different step size η.

B. Target Firing Rate

The target firing rate p determines the average spike rate. A low p results in a low average spike rate, as seen in Fig. 9 for a network of 512 neurons with $\eta = 2^{-5}$ and $w = 3\tau$. Reducing p

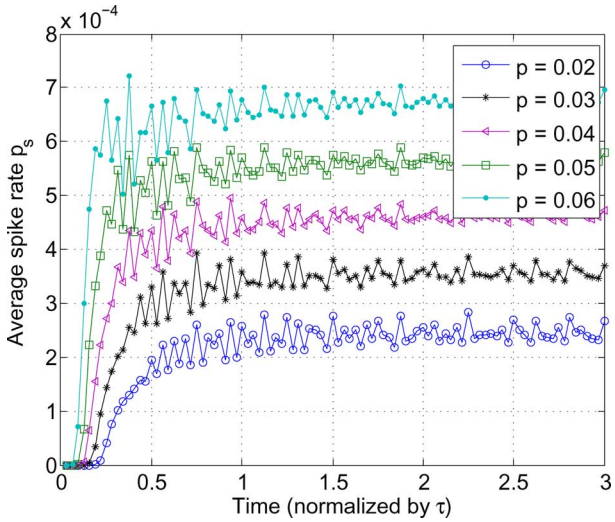
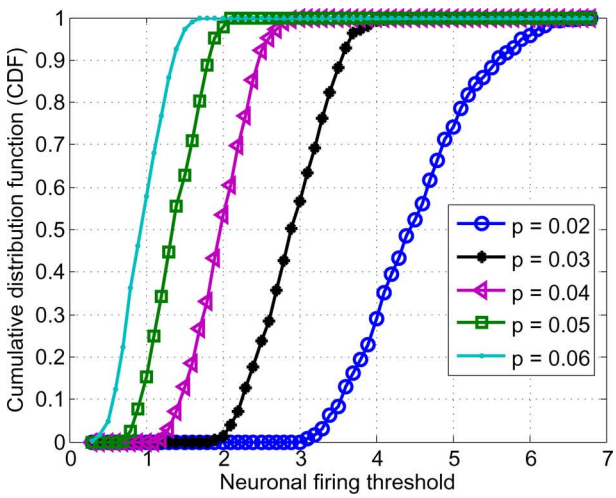
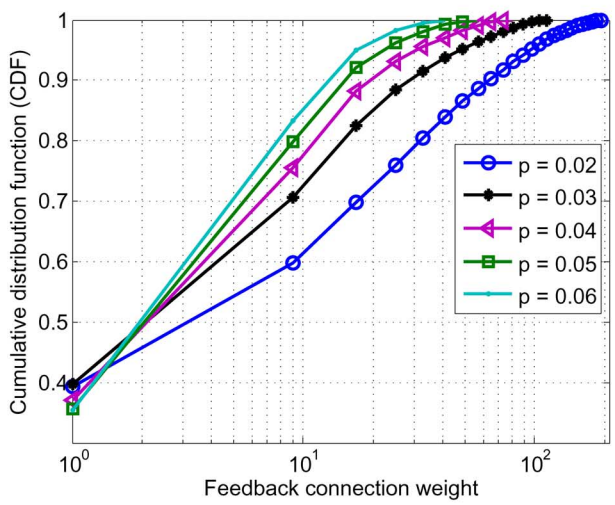


Fig. 9. Average spike rate at each time step across a network of 512 neurons when performing inference with $\eta = 2^{-5}$.



(a)



(b)

Fig. 10. (a) CDF of neuron firing thresholds, and (b) CDF of feedback connection weights for different p values.

raises the firing thresholds and the feedback connection weights, as shown in Fig. 10(a) and (b), and creates a more sparse and random spiking network.

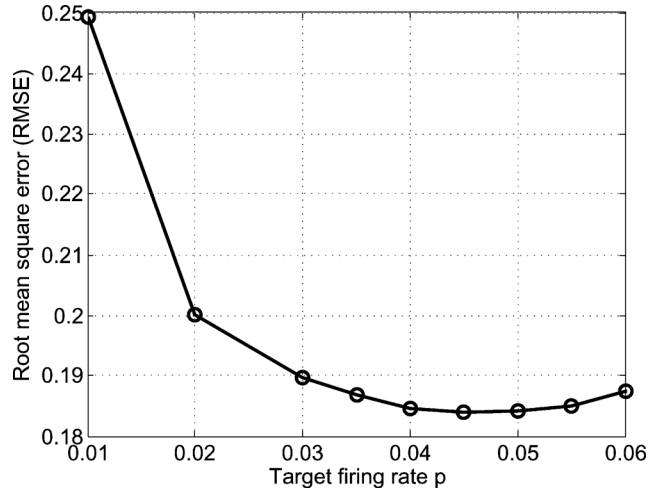


Fig. 11. RMSE of the image reconstruction by sparse code using different target firing rate p.

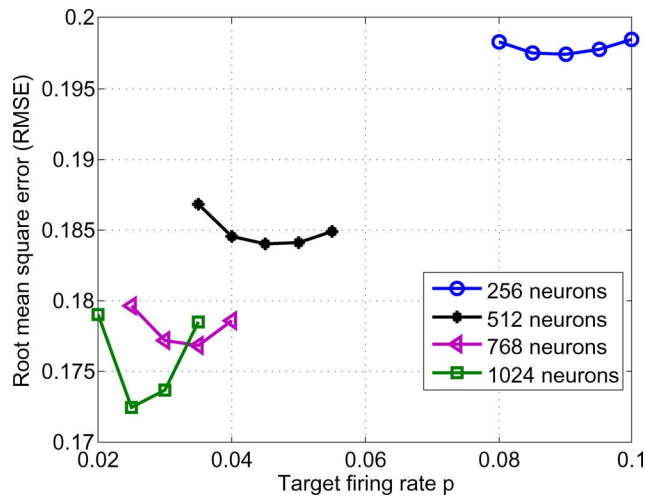


Fig. 12. RMSE of the image reconstruction by sparse code using different target firing rate p and network size.

A low target firing rate p is attractive for an efficient implementation, but a very low p raises the RMSE, as seen in Fig. 11. In a network of 512 neurons, $p = 0.01$ results in an average of only $512 \times 0.01 = 5.12$ spikes over the inference window. The linear generative model (4) contains less than a handful of terms, which are insufficient for sparse coding. Raising p to 0.02 doubles the number of spikes and reduces the RMSE. Continued increase of p improves the RMSE until p reaches about 0.045, or about 23 spikes over the inference window. The RMSE then worsens as the spikes start to crowd. Therefore, while we optimize the SAILnet algorithm for sparse and random spikes, a minimum number of spikes are needed for a good RMSE.

To verify the minimum number of spikes necessary, we designed networks of 256, 512, 768, and 1024 neurons, and analyzed the choice of p, as illustrated in Fig. 12. A lower RMSE is attainable in a larger network, but only with a good choice of p. The optimal p is lower in a larger network: about 0.09 in the 256 network, 0.045 in the 512 network, 0.03 in the 768 network, and

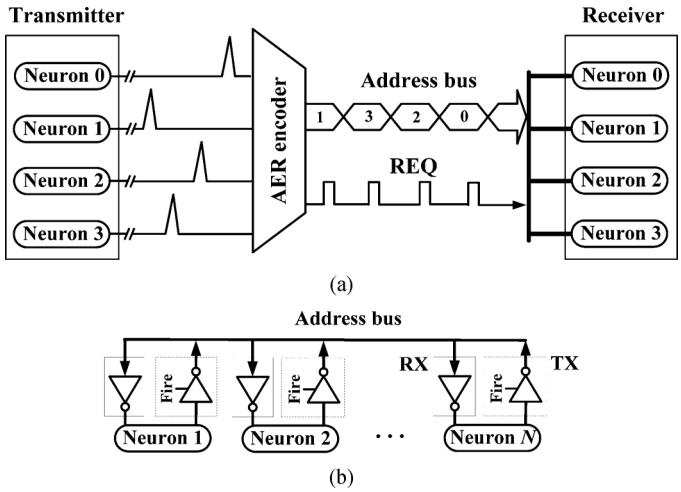


Fig. 13. (a) Neuron communication via AER protocol, and (b) neuron communication via arbitration-free bus.

0.025 in the 1024 network, which all point to the almost constant optimal number of spikes to be around 20 to 25 regardless of the size of the network.

The above analysis yields two important insights: (1) a minimum number of spikes are needed for a good RMSE, and more spikes beyond the minimum are not necessary, and (2) sparse and random spikes result in better RMSE. The two insights guide the selection of the neural network size along with the target firing rate for the best tradeoff between RMSE and implementation cost, and the selection of the update step size for the best tradeoff between RMSE and throughput.

V. Arbitration-Free Bus Architecture

A bus allows many neurons to be connected [16], [29], replacing individual point-to-point links altogether. A bus can be used to connect pixel inputs to the neurons using a pixel bus, and connect neurons together using a neuron bus.

Once the inputs are complete, neurons will be ready to perform integrate and fire and generate spikes. The spikes are exchanged on the neuron bus. Since the neuron spikes are random, arbitration is required to resolve the conflicts when multiple neurons try to access the bus at the same time. An arbiter decides the order that the access is granted depending on a pre- or a dynamically determined priority. The design of an arbiter is complicated by a large neural network, as the arbiter needs to serve many neurons and handle large fan-in and fan-out connections, and the service time is critical as excessive delays alter the algorithm dynamics and degrade the performance. Solutions have been proposed to structure the arbiter design to reduce its fan-in and fan-out connections and improve its service time, but increasing the hardware cost [21], [24]–[26].

AER is a popular time-multiplexing communication protocol for the neuron bus. Fig. 13(a) explains the protocol [19], [20]. When a neuron fires, the AER encoder puts the address of the neuron on the AER bus and asserts a request REQ. All neurons are attached to this bus. Upon hearing REQ, neurons read the address from the bus and perform integrate and fire. If neuron spikes are very sparse, AER enables an efficient sharing of the bus. When there are more than one spike at the same time, it results in a collision, and arbitration is required to resolve the conflict. To be able to handle multiple requests in a timely manner, a synchronous AER bus needs to operate at a higher clock speed. If spikes happen in bursts, the bus speed has to be increased further. An asynchronous AER bus is potentially beneficial, but in this work we focus on a synchronous bus.

A. Arbitration-Free Bus

The power-consuming arbitration and higher bus speed are used to resolve spike collisions. If the collisions can be tolerated, arbitration is removed and the bus will run at the same clock speed as neurons, leading to a more efficient bus architecture. The sparse and independent neuron firing backed by the SAILnet algorithm [13] is promising, as the spike rate is kept low and the spikes are random, making the collision rate much lower than a conventional neural network. It is then plausible to adopt an arbitration-free bus that tolerates spike collisions.

An arbitration-free bus architecture can be designed as in Fig. 13(b). Each neuron is equipped with a tri-state transmitter (TX) and an inverter as receiver (RX). The TX and RX are assumed to be symmetrical, i.e., the pull-up and pull-down strength are balanced. (The assumption is used to simplify the analysis.) When a neuron fires, it puts its address (composed of multiple bits) on the shared neuron address bus. All neurons are attached to the address bus. Upon detecting an address-event, neurons will read the address and perform integrate and fire. The bus runs at the same clock frequency as the neurons. A collision occurs when multiple neurons spike at the same time, as they will all attempt to drive their addresses onto the bus. We assume a collision resolution scheme to match the implementation in Fig. 13(b): if a bus line is driven by multiple neurons, the pull-up and pull-down strength determine the "winning" bit. For example, if there are x neurons pulling up a line and y neurons pulling down a line, the winning bit will be 1 if $x > y$, 0 if $x < y$, and a random draw if $x = y$ (as the voltage level will be in the undetermined region and the RX output will be determined by noise). The throughput of the bus architecture is f_{clk}/n_s image patches per second, same as the fully connected network.

B. Spike Collisions

A collision on the neuron address bus results in spike corruption. The winning neuron address in a collision may not match any of the competing neurons involved in the collision. Therefore, the fidelity of sparse coding will be sacrificed. To minimize the degradation, the collision rate needs to be kept low.

The spike collision probability can be analytically derived by assuming that the spikes are independent. The independent spikes assumption is backed by the SAILnet algorithm [13]. A collision occurs when two or more spikes occur in the same time step, thus the average collision probability in each time step is given by

$$P_c = 1 - (1 - p_s)^N - Np_s(1 - p_s)^{N-1},$$

where N is the size of the network, which is assumed to be large, and p_s is the spike rate in each time step, which is assumed to be low and close to 0. By Taylor series expansion of $(1 - p_s)^N$

and $(1-p_s)^{N-1}$ at $p_s = 0$, and keeping only the first two terms in each expansion, we get

$$P_c \approx 1 - (1 - Np_s) - Np_s(1 - (N-1)p_s) \approx (Np_s)^2,$$

Assume that p_s is approximately the target firing rate p averaged over the n_s time steps, i.e., $p_s \approx p/n_s$, the above equation can be written as

$$P_c \approx (Np/n_s)^2. \tag{6}$$

Note that Np is the total number of spikes over the inference window, which remains approximately constant if the target firing rate p is optimally set based on the size of the network, as discussed previously. The result (6) suggests the collision probability's quadratic dependence on the total number of spikes within the inference window averaged over the number of time steps. In a network of 512 neurons with a target firing rate of $p = 0.045$, $n_s = 96$, the average collision probability $P_c \approx 5.8\%$.

As an experimental verification of (6), the arbitration-free bus is first trained to learn the feed-forward and feedback connection weights and firing thresholds, and then used to perform inference. Fig. 14(a) shows the average collision probability of the four networks of size $N = 256, 512, 768, \text{ and } 1024$ with a target firing rate $p = 0.09, 0.045, 0.03, \text{ and } 0.0225$, respectively. The neuron update step size $\eta = 2^{-5}$, and the number of time steps $n_s = 96$. The average collision probability of each of the four networks is approximately 5%, which agrees with the analytical result. Collisions result in a small increase of the RMSE in image reconstruction using the linear generative model, as shown in Fig. 14(b).

The dependency of the collision probability on η is plotted in Fig. 15(a) for a network of 512 neurons that perform inference with a target firing rate $p = 0.04$. Note that the neuron update step size η is inversely proportional to the number of time steps n_s. The neuron update step size η is varied from 2^{-4} to 2^{-6}, and the collision probability decreases quadratically, confirming the analytical result in (6). A small η helps spread spikes over more steps, resulting in a much lower collision probability and RMSE, as shown in Fig. 15(b).

The efficient arbitration-free bus architecture provides the same throughput as a fully connected network. It also avoids using AER that requires a bus arbiter and a higher bus speed. To implement the arbitration-free bus architecture, the neuron update step size η needs to be kept sufficiently low to maintain a low spike collision rate for a good RMSE.

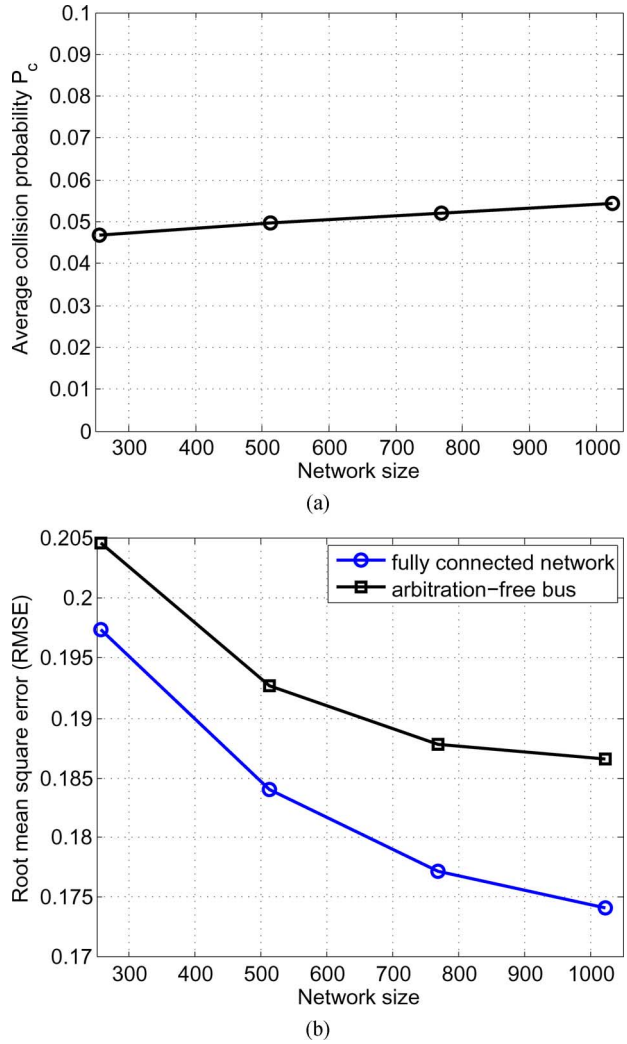


Fig. 14. (a) Collision probability in 256, 512, 768, 1024-neuron arbitration-free bus with $p = 0.09, 0.045, 0.03, \text{ and } 0.0225$, respectively, and (b) RMSE of the image reconstruction by sparse code obtained from the arbitration-free bus compared to a fully connected network.

VI. Latent Ring Architecture

Systolic ring [17] is a serial relay architecture as shown in Fig. 16. In the simplest setup, the communication in a ring is unidirectional, i.e., a neuron can only talk to one neighboring neuron. When a neuron spikes, it passes the spike as an address-event to the next neuron, who then passes the address-event one step further, and so on.

A. Spike Generation and Propagation

In a digital implementation, a spike address-event steps through one neuron every clock cycle. For a spike to be heard by all neurons, it will have to travel along the entire ring, which takes $N-1$ clock cycles, where N is the size of the network. Once an address-event reaches its originating neuron, the event is deleted. One advantage of the ring over the bus is that it is more scalable because communications are local, and it eliminates all collisions. The throughput of the ring architecture is f_{clk}/n_s image patches per second. The short connections between neighboring neurons allow the ring to run at a higher clock frequency than a fully connected network or a bus.

Even though the throughput of the ring architecture can be as high as the fully connected network or the bus, a choice of $n_s < N-1$ will result in the incomplete propagation of spikes, since it takes a minimum of $N-1$ clock cycles for a spike to be propagated through the ring. As a result, some of the spikes are lost. Another consequence of the serial spike propagation along the ring is that the inhibitions due to neuron spikes do not take effect immediately, which allows neuron potentials to grow without inhibitions and mistakenly fire. The delayed inhibitions eventually take effect to suppress the spike rate later in the propagation. The two factors, spike losses due to incomplete

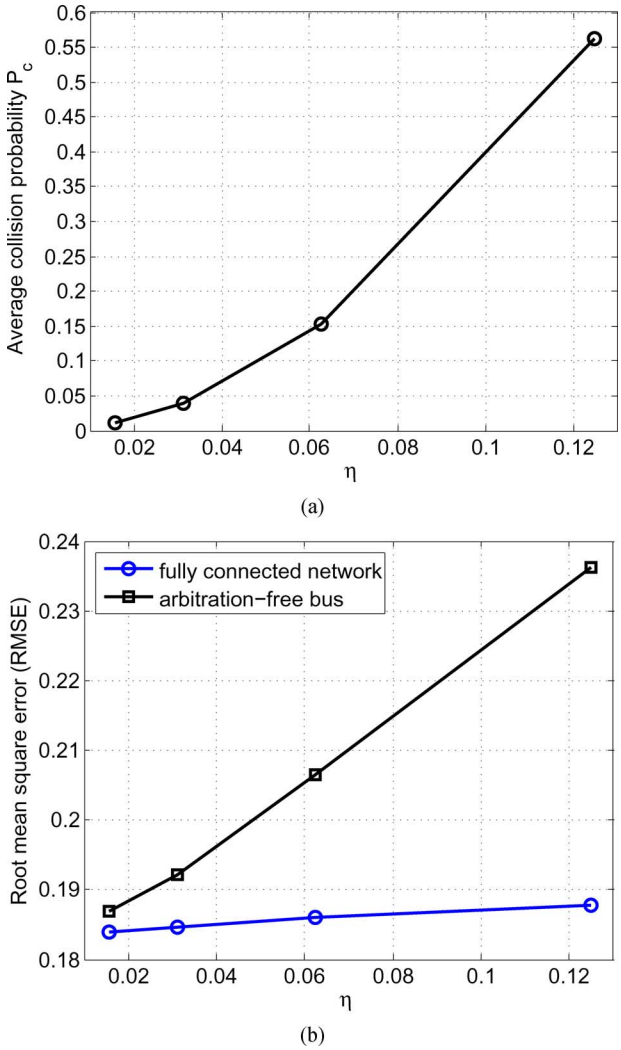


Fig. 15. (a) Collision probability in a 512-neuron arbitration-free bus with $p = 0.04$, and (b) RMSE of the image reconstruction by sparse code obtained from the arbitration-free bus compared to a fully connected network.

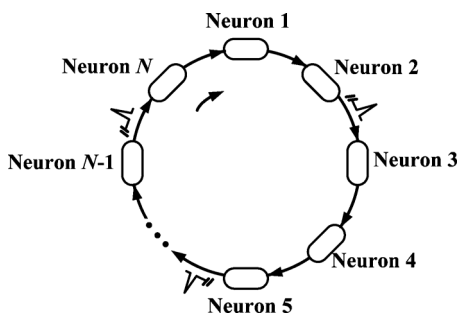


Fig. 16. A ring architecture.

spike propagation, and neuron misfires due to delayed inhibitions, worsen the fidelity of sparse coding.

B. Damping Neuron Responses

Unlike in the bus where the spikes reach all neurons within one clock cycle, the serial spike passing in the ring takes much longer to reach all neurons. Neuron potentials will grow, and without seeing spikes immediately, the potential will grow to high levels, resulting in many more spikes. The majority of the spikes are actually misfires, which contribute to errors. The spike rate pattern of a network of $N = 512$ neurons in a ring with update step size of $\eta = 2^{-5}$ and number of update steps $n_s = 96$ is shown in Fig. 17. The distinctive spike rate pattern of the ring is compared with the fully connected network, showing the spike rate grows up to an order of magnitude higher due to neuron misfires, followed by a depression as the inhibitions take effect to suppress the spikes. Note that since $n_s < N-1$, spikes do not reach all neurons. The neuron misfires and spike losses result in a high RMSE.

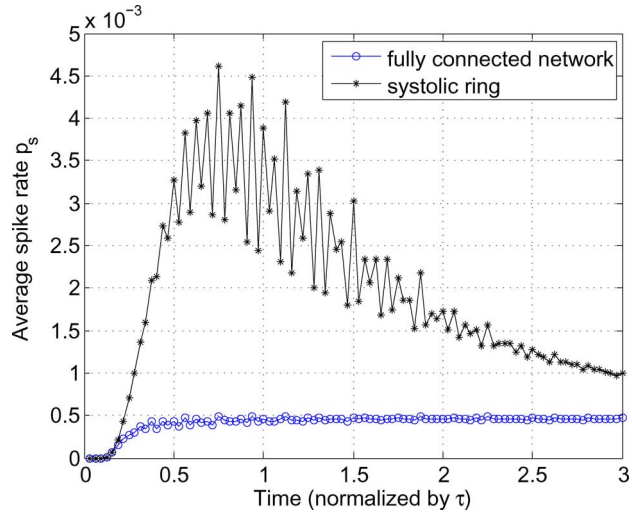


Fig. 17. Average spike rate at each time step across a 512-neuron ring when performing inference with a target firing rate of $p = 0.04$.

The serial spike passing along the ring slows down the inhibitions, whereas the neurons are active in every update step and ready to fire. The mismatch between the inhibitions and excitations cause many neurons to misfire. To improve the fidelity of sparse coding, the inhibitory and excitatory effects need to be balanced. A simple scheme is to implement a holding policy: each neuron is allowed to propagate one spike every cycle, but perform inference update only once every d cycles. For example, setting $d = 2$ will allow each neuron to update once every two cycles. The number of inference steps is still n_s, so the spikes will propagate to dn_s neurons. The holding policy reduces the excessive excitatory strength. Longer spike propagation also reduces spike losses. As Fig. 18(a) shows, after implementing holding, the misfire rate is lower and the RMSE improves until d reaches about 6, as shown in Fig. 18(b). Note that in the example of $N = 512$ and $n_s = 96$, when $d = 6$, $dn_s \approx N$, i.e., spikes are allowed to propagate around the entire ring, eliminating spike losses.

The holding policy is one way of implementing a "latent" ring that damps neuron responses to adapt to the slow spike propagation. During holding, each neuron disables firing. To implement holding of d cycles, a d-entry memory needs to be added to each neuron to store up to d address-events. The latent ring decreases the throughput to $f_{clk}/(dn_s)$ image patches per second, where dn_s should be set to close to N for the best RMSE.

An alternative approach to damp the neuron response is to use a smaller neuron update step size η. A smaller η increases

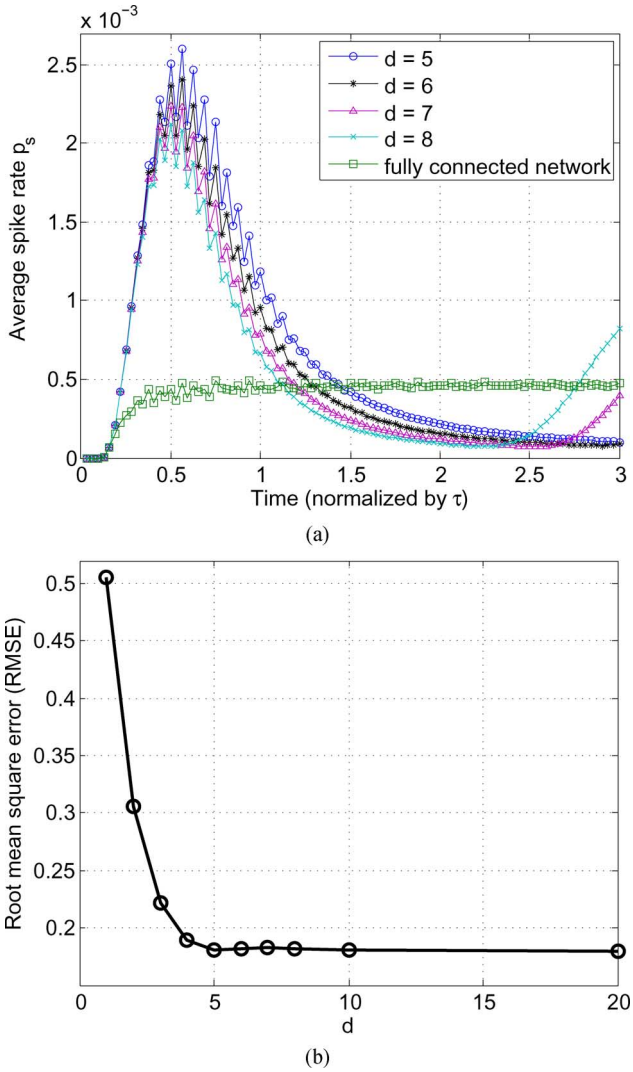


Fig. 18. (a) Average spike rate of a 512-neuron ring with holding, and (b) RMSE of the image reconstruction by sparse code obtained from the ring that implements holding.

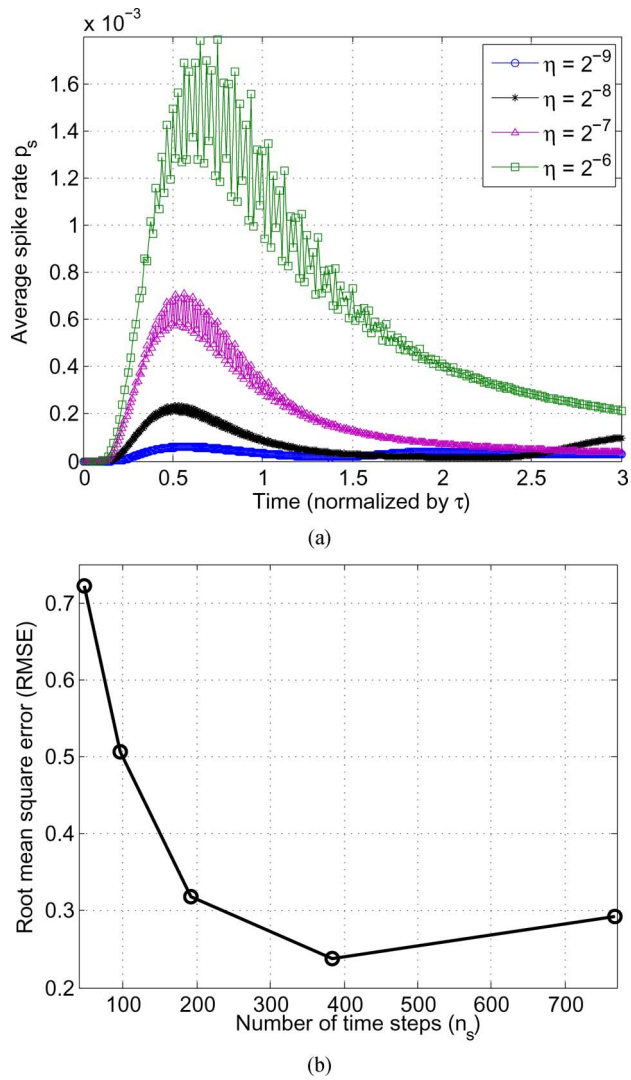


Fig. 19. (a) Average spike rate of a 512-neuron ring by changing update step size η, and (b) RMSE of the image reconstruction by sparse code obtained from the ring.

the number of update steps n_s for a fixed window w. Fig. 19(a) shows the spike rate pattern as η is reduced from 2^{-6} to 2^{-9}, the misfire rate is reduced significantly. The RMSE improves with a lower η, as indicated in Fig. 19(b), and the best RMSE is reached between $\eta = 2^{-7}$ and 2^{-8}, or when $n_s \approx N$.

A latent ring is a scalable network. The serial spike propagation slows down the inhibitions and neuron responses are damped to match the slow spike propagation for a good RMSE. The damping decreases the throughput, but a higher clock frequency can be achieved with the ring architecture than the bus architecture.

VII. Hybrid Bus-Ring Architecture

The arbitration-free bus architecture is efficient and provides high throughput but its clock speed is limited by capacitive loading, so the number of neurons that can be connected to a bus is limited. The latent ring architecture is scalable, but the extra delay introduced hurts the throughput. Therefore, we propose to combine bus and ring in a hybrid architecture.

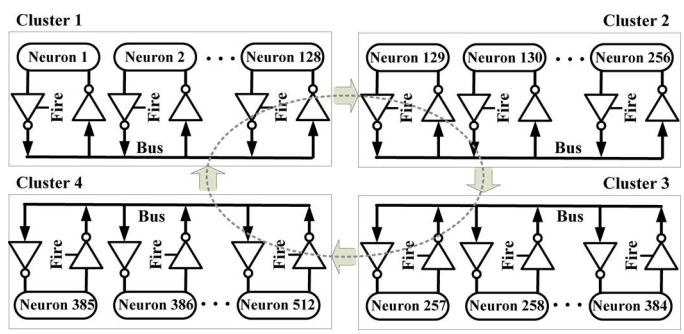
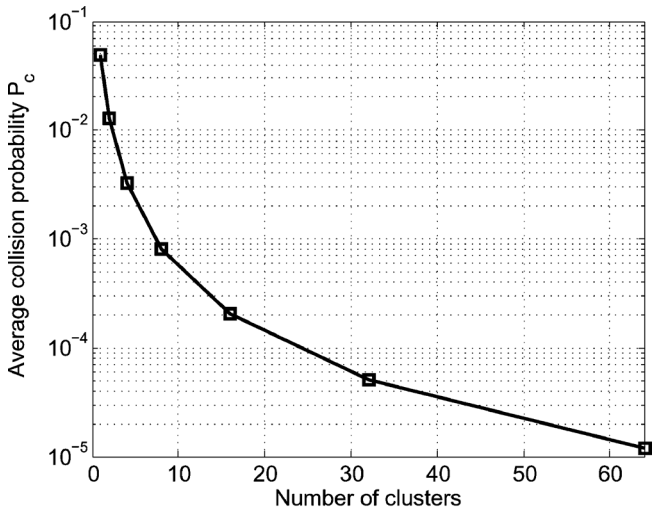
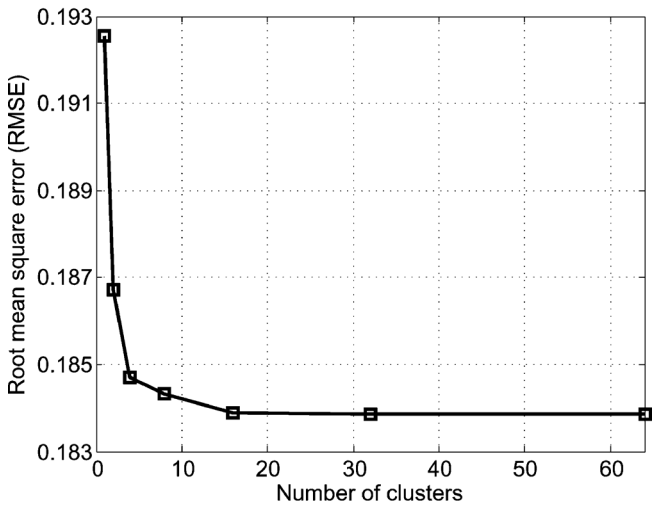


Fig. 20. A 512-neuron 2-layer bus-ring architecture, consisting of 4 neuron clusters.

The hierarchical bus-ring architecture is implemented in a two-level hierarchy, as illustrated in Fig. 20. At the first level, neurons are grouped into local clusters, and the neurons in a cluster are connected in a bus. The cluster size is limited to control the capacitive loading of the bus to maintain a high clock



(a)



(b)

Fig. 21. (a) Collision probability in a 512-neuron hybrid network with $p = 0.045$ and $n_s = 96$, and (b) RMSE of the image reconstruction by sparse code obtained from the hybrid network.

speed. At the second level, a small number of clusters are connected in a ring. The length of the ring is kept short to minimize the communication latency. The spike address-events are generated in local clusters and then propagated through the ring to broadcast to all other clusters. The throughput of the hybrid architecture is f_{clk}/n_s image patches per second. The hybrid architecture is scalable as buses are kept short, and the communication latency is reduced compared to a global bus or a long ring.

With short local buses in the hybrid architecture, the probability of spike collision is reduced. Using (6) and assuming that the spikes are independent, the collision probability is reduced quadratically with smaller clusters, and more clusters are needed for a given neural network size, as shown in Fig. 21(a). With a reduced collision probability and faster spike propagation through a short ring, the hybrid architecture allows for an efficient implementation of the SAILnet sparse coding algorithm to achieve an excellent RMSE with as few as 4 clusters, as shown in Fig. 21(b). In the next section, we compare hardware implementation results to demonstrate the advantages of the hybrid architecture.

TABLE I
65 NM CMOS CHIP SYNTHESIS RESULTS.

Architecture	Bus	Ring	Hybrid
Network size	512	512	512
Memory size	1.5Mb	1.5Mb	1.5Mb
Core area	3.47mm^2	4.08mm^2	3.47mm^2
Frequency	357MHz	357MHz	357MHz
Standard cells	461,691	502,821	482,862
Power	497mW	538mW	463mW
Throughput	952Mpx/s	238Mpx/s	952Mpx/s
Energy	0.522nJ/px	2.261nJ/px	0.486nJ/px

VIII. CHIP DESIGN RESULTS

As a proof of concept, we synthesized a 512-neuron network for sparse coding in a TSMC 65 nm CMOS technology. One design was implemented in an arbitration-free bus architecture, one was implemented in a latent ring architecture with a holding factor $d = 4$, and a third design was implemented in a hybrid bus-ring architecture with a 4-stage ring connecting 4 128-neuron buses.

Learning and inference are both soft operations that are intrinsically noise tolerant. The feed-forward Q weight and the feedback W are quantized to 4 bits with minimal impact on RMSE, saving significant memory and complexity. To improve the area utilization, the W and Q memories of 32 neurons are grouped together to increase the word size and amortize the addressing overhead. Memory grouping is feasible in a bus architecture or a hybrid architecture because all neurons in a bus or a cluster (in the hybrid architecture) will be accessing the same address in the Q memories in response to a pixel input, and the same address in the W memories in response to a neuron spike. However, memory grouping is not possible in a ring architecture, because each neuron receives spikes at different times, and the memory accesses are not synchronized. With a holding policy, neurons perform updates every d cycles, so the memories of d neurons are grouped together to save area. The three designs are synthesized using Synopses Design Compiler, and place-and-routed using Cadence Encounter. The results are compared in Table I, where core area refers to the area of the design excluding peripheral circuits, such as clock generation, testing circuits, and input and output pads. The designs are built using standard cells and SRAMs, and the memory capacity and standard cell count are also reported.

The area of the bus and the hybrid architectures are smaller than the ring because memories can be grouped together into larger arrays to save area. The bus architecture runs at a maximum clock frequency of 357 MHz (2.8 ns clock period), limited by the capacitive loading of the bus. The ring and the hybrid architecture can run faster at up to 385 MHz and 370 MHz, receptively. Note that the 512-neuron network is still relatively small to see a notable difference in clock frequency. For a larger network, we expect the difference will be more pronounced.

At 357 MHz, the power consumption of the three architectures can be compared. The ring architecture consumes the highest power, due to the smaller memory arrays that incur a higher overhead. The bus architecture consumes less power with the help of larger memory arrays that amortize the overhead. The hybrid architecture consumes the least power due to larger memory arrays and a reduction in bus loading. At 357 MHz, the throughput of the hybrid architecture is 952 Mpx/s, which is fast enough to process 3980×3980 image frames at 60 frames per second at a low energy consumption of 0.486 nJ/px.

The ring architecture is more scalable than the bus architecture. To support an even larger network, the area and power of the ring architecture are expected to scale up linearly. However, the ring delays spike propagation, which lowers the throughput. The hybrid bus-ring architecture divides the bus into smaller clusters, which is more scalable than a flat bus architecture, and it improves the throughput and RMSE over the latent ring architecture.

IX. Conclusion

In this work, we design efficient neural network architectures for the mapping of the SAILnet sparse coding algorithm. By exploring the dynamics of the algorithm in a digital neural network, we show that a low target firing rate and a small neuron update step size are necessary to maintain sparse and random neuron spikes for an efficient use of hardware resources. The optimal target firing rate is dependent on the network size, and the minimum neuron update step size is determined by the throughput requirement.

For a practical implementation of the SAILnet sparse coding algorithm, three network architectures are considered: a bus architecture that provides a shared medium for neuron communications, and a ring architecture that serializes neuron communications. The bus architecture results in spike collisions and requires access arbitration. We show that the collision rate is quadratically dependent on the number of spikes averaged over the number of neuron update steps. Keeping the spikes sparse and random by a small neuron update step reduces the collision rate to about 5%, small enough that the errors due to collisions are tolerated by the SAILnet algorithm with only a small impact on the RMSE. We design an efficient arbitration-free bus architecture that tolerates spike collisions and removes bus access arbitration.

A conventional ring architecture propagates spikes serially, delaying inhibitions and causing neurons to misfire. The misfires are fundamentally due to the mismatch between slow inhibitions and fast neuron responses. To reduce the neuron misfires, the neuron responses are damped by a holding policy in a latent ring architecture, where each neuron is allowed to propagate one spike every cycle, but only allowed to perform inference update once every d cycles ($d > 1$). Alternatively, the neuron responses can be damped by a small update step size.

The arbitration-free bus architecture and the latent ring architecture are combined in a hybrid bus-ring architecture to achieve a better scalability than the bus architecture, and a higher throughput than the ring architecture. Synthesis, place-and-route in 65 nm CMOS show that the hybrid architecture occupies the same area as the bus architecture, and it consumes the lowest power. At 357 MHz, the hybrid architecture achieves a throughput of 952 Mpx/s at 0.486 nJ/px. The proof-of-concept designs demonstrate the high throughput and energy efficiency of practical implementations of sparse coding.

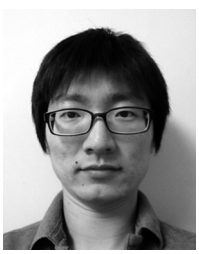
Acknowledgment

The authors would like to thank G. Kenyon, W. Lu, and M. Flynn for helpful discussions.

References

[1] F. Attneave, "Some informational aspects of visual perception," *Psychol. Rev.*, vol. 61, no. 3, p. 183, 1954.
[2] J. J. Atick and A. N. Redlich, "What does the retina know about natural scenes?," *Neural Comput.*, vol. 4, no. 2, pp. 196–210, 1992.
[3] D. H. Hubel and T. N. Wiesel, "Receptive fields and functional architecture of monkey striate cortex," *J. Physiol.*, vol. 195, no. 1, pp. 215–243, 1968.
[4] R. L. De Valois, D. G. Albrecht, and L. G. Thorell, "Spatial frequency selectivity of cells in macaque visual cortex," *Vis. Res.*, vol. 22, no. 5, pp. 545–559, 1982.
[5] J. P. Jones and L. A. Palmer, "An evaluation of the two-dimensional Gabor filter model of simple receptive fields in cat striate cortex," *J. Neurophysiol.*, vol. 58, no. 6, pp. 1233–1258, 1987.
[6] D. Field, "What is the goal of sensory coding?," *Neural Comput.*, vol. 6, no. 4, pp. 559–601, 1994.
[7] B. A. Olshausen, "Principles of image representation in visual cortex," *Visual Neurosci.*, pp. 1603–1615, 2003.
[8] B. A. Olshausen and D. J. Field, "Emergence of simple-cell receptive field properties by learning a sparse code for natural images," *Nature*, vol. 381, no. 6583, pp. 607–609, 1996.
[9] P. Földiak, "Forming sparse representations by local anti-Hebbian learning," *Biol. Cybernet.*, vol. 64, no. 2, pp. 165–170, 1990.
[10] M. S. Falconbridge, R. L. Stamps, and D. R. Badcock, "A simple Hebbian/anti-Hebbian network learns the sparse, independent components of natural images," *Neural Comput.*, vol. 18, no. 2, pp. 415–429, 2006.
[11] M. Rehn and F. T. Sommer, "A network that uses few active neurons to code visual input predicts the diverse shapes of cortical receptive fields," *J. Computat. Neurosci.*, vol. 22, no. 2, pp. 135–146, 2007.
[12] C. J. Rozell, D. H. Johnson, R. G. Baraniuk, and B. A. Olshausen, "Sparse coding via thresholding and local competition in neural circuits," *Neural Computat.*, vol. 20, no. 10, pp. 2526–2563, 2008.
[13] J. Zylberberg, J. T. Murphy, and M. R. DeWeese, "A sparse coding model with synaptically local plasticity and spiking neurons can account for the diverse shapes of V1 simple cell receptive fields," *PLoS Comput. Biol.*, vol. 7, no. 10, p. e1002250, 2011.
[14] D. O. Hebb, *The Organization of Behavior: A Neuropsychological Theory*. Mahwah, NJ, USA: Erlbaum, 2002.
[15] M. Holler, S. Tam, H. Castro, and R. Benson, "An electrically trainable artificial neural network (ETANN) with 10240 'floating gate' synapses," in *Proc. Int. Joint Conf. Neural Netw.*, 1989, pp. 191–196.
[16] M. Yasunaga, N. Masuda, M. Yagyu, M. Asai, M. Yamada, and A. Masaki, "Design, fabrication and evaluation of a 5-inch wafer scale neural network LSI composed on 576 digital neurons," in *Proc. Int. Joint Conf. Neural Netw.*, 1990, pp. 527–535.
[17] U. Ramacher, "SYNAPSE-A neurocomputer that synthesizes neural algorithms on a parallel systolic engine," *J. Parallel Distrib. Comput.*, vol. 14, no. 3, pp. 306–318, 1992.
[18] P. Ienne and M. A. Viredaz, "GENES IV: A bit-serial processing element for a multi-model neural-network accelerator," *J. VLSI Signal Process. Syst. Signal, Image, Video Technol.*, vol. 9, no. 3, pp. 257–273, 1995.
[19] M. Mahowald, *An Analog VLSI System for Stereoscopic Vision*. Berlin, Germany: Springer-Verlag, 1994.
[20] M. A. Sivilotti, "Wiring Considerations in Analog VLSI Systems, with Application to Field-Programmable Networks," Ph.D. dissertation, California Inst. Technol., Pasadena, CA, USA, 1990.
[21] R. J. Vogelstein, U. Mallik, J. T. Vogelstein, and G. Cauwenberghs, "Dynamically reconfigurable silicon array of spiking neurons with conductance-based synapses," *IEEE Trans. Neural Netw.*, vol. 18, no. 1, pp. 253–265, Jan. 2007.
[22] P. Merolla, J. Arthur, F. Akopyan, N. Imam, R. Manohar, and D. S. Modha, "A digital neurosynaptic core using embedded crossbar memory with 45 pJ per spike in 45 nm," in *Proc. IEEE Custom Integr. Circuits Conf.*, 2011, pp. 1–4.

[23] J.-S. Seo, B. Brezzo, Y. Liu, B. D. Parker, S. K. Esser, R. K. Montoye, B. Rajendran, J. A. Tierno, L. Chang, and D. S. Modha, "A 45 nm CMOS neuromorphic chip with a scalable architecture for learning in networks of spiking neurons," in *Proc. IEEE Custom Integr. Circuits Conf.*, 2011, pp. 1–4.
[24] G. Indiveri, E. Chicca, and R. Douglas, "A VLSI array of low-power spiking neurons and bistable synapses with spike-timing dependent plasticity," *IEEE Trans. Neural Netw.*, vol. 17, no. 1, pp. 211–221, 2006.
[25] S. Choudhary, S. Sloan, S. Fok, A. Neckar, E. Trautmann, P. Gao, T. Stewart, C. Eliasmith, and K. Boahen, "Silicon neurons that compute," in *Artif. Neural Netw. Mach. Learn.*. Berlin, Germany: Springer-Verlag, 2012, pp. 121–128.
[26] Serrano-Gotarredona *et al.*, "CAVIAR: A 45 k neuron, 5 M synapse, 12 G connects/s AER hardware sensory-processing–learning-actuating system for high-speed visual object recognition and tracking," *IEEE Trans. Neural Netw.*, vol. 20, no. 9, pp. 1417–1438, Sep. 2009.
[27] E. Painkras, L. A. Plana, J. Garside, S. Temple, F. Galluppi, C. Patterson, D. R. Lester, A. D. Brown, and S. B. Furber, "SpiNNaker: A 1-W 18-core system-on-chip for massively-parallel neural network simulation," *IEEE J. Solid-State Circuits*, vol. 48, no. 8, pp. 1943–1953, Aug. 2013.
[28] C. Zamarreño-Ramos, A. Linares-Barranco, T. Serrano-Gotarredona, and B. Linares-Barranco, "Multicasting mesh AER: A scalable assembly approach for reconfigurable neuromorphic structured AER systems. Application to ConvNets," *IEEE Biomed. Circuits Syst.*, vol. 7, no. 1, pp. 82–102, Feb. 2013.
[29] D. Hammerstrom, "A VLSI architecture for high-performance, low-cost, on-chip learning," in *Proc. Int. Joint Conf. Neural Netw.*, 1990, pp. 537–544.
[30] P. Dayan and L. Abbott, "Theoretical neuroscience: Computational and mathematical modeling of neural systems," *J. Cogn. Neurosci.*, vol. 15, no. 1, pp. 154–155, 2003.
[31] C. Mead and M. Ismail, *Analog VLSI Implementation of Neural Systems*. Berlin, Germany: Springer-Verlag, 1989.
[32] W. Gerstner and W. M. Kistler, *Spiking Neuron Models: Single Neurons, Populations, Plasticity*. Cambridge, U.K.: Cambridge Univ. Press, 2002.
[33] W. E. Vinje and J. L. Gallant, "Sparse coding and decorrelation in primary visual cortex during natural vision," *Science*, vol. 287, no. 5456, pp. 1273–1276, 2000.
[34] A. S. Ecker, P. Berens, G. A. Keliris, M. Bethge, N. K. Logothetis, and A. S. Tolias, "Decorrelated neuronal firing in cortical microcircuits," *Science*, vol. 327, no. 5965, pp. 584–587, 2010.
[35] Advanced Topics in Systems Neuroscience Redwood Center for Theoretical Neuroscience [Online]. Available: http://redwood.berkeley.edu/wiki/
[36] E. P. Simoncelli and B. A. Olshausen, "Natural image statistics and neural representation," *Annu. Rev. Neurosci.*, vol. 24, no. 1, pp. 1193–1216, 2001.

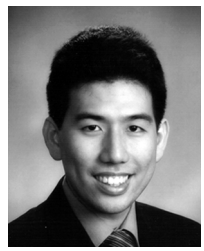


Jung Kuk Kim (S'10) received the B.S. degree in electrical engineering from the Korea Advanced Institute of Science and Technology, Daejeon, South Korea, in 2009, and the M.S. degree in electrical engineering from the University of Michigan, Ann Arbor, MI, USA, in 2011. He is currently working toward the Ph.D. degree in electrical engineering at the University of Michigan.

He received the 2009 Doctoral Fellowship from the Korea Foundation for Advanced Studies, and the 2012 Riethmiller Fellowship. His research interests are in low-power and high-performance VLSI systems with an emphasis on algorithm and architecture co-design for signal and image processing.

Phil Knag (S'11) received the B.S. degree in computer engineering and the M.S. degree in electrical engineering from the University of Michigan, Ann Arbor, MI, USA, in 2010 and 2012, respectively, where he is currently working toward the Ph.D. degree in electrical engineering.

He received a GAANN fellowship in 2010 from the U.S. Department of Education for academic excellence. He was with Medtronic, Inc. as a research intern in 2010. His current research interests include nanoscale and neuromorphic computing systems.



Thomas Chen received the B.S. degree in electrical engineering from the University of Michigan, Ann Arbor, in 2013, where he is currently pursuing the Ph.D. degree in electrical engineering.

He received a Rackham Merit Fellowship in 2013 from the University of Michigan. His research interests are in high-speed and low-power VLSI circuits and systems for computing and communications.

Zhengya Zhang (S'02–M'09) received the B.A.Sc. degree in computer engineering from the University of Waterloo, Waterloo, ON, Canada, in 2003, and the M.S. and Ph.D. degrees in electrical engineering from the University of California, Berkeley, CA, USA, in 2005 and 2009, respectively.

Since 2009, he has been on the faculty of the University of Michigan, Ann Arbor, MI, USA, as an Assistant Professor in the Department of Electrical Engineering and Computer Science. His current research interests include low-power and high-performance VLSI circuits and systems for computing, communications and signal processing.

Dr. Zhang was a recipient of the National Science Foundation CAREER Award in 2011, the Intel Early Career Faculty Honor Program Award in 2013, the David J. Sakrison Memorial Prize for outstanding doctoral research in EECS at UC Berkeley, and the Best Student Paper Award at the Symposium on VLSI Circuits. He is an Associate Editor of the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS-I: REGULAR PAPERS, and an Associate Editor of the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS-II: EXPRESS BRIEFS.