CANONICAL NEURAL COMPUTATIONS IN ASYNCHRONOUS NEUROMORPHIC CIRCUITS

A Dissertation

Presented to the Faculty of the Graduate School

of Cornell University

in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

by

Nabil Imam

May 2014

© 2014 Nabil Imam

ALL RIGHTS RESERVED

CANONICAL NEURAL COMPUTATIONS IN ASYNCHRONOUS NEUROMORPHIC CIRCUITS

Nabil Imam, Ph.D.

Cornell University 2014

This thesis describes the architecture, implementation and application of a parallel computing system modeled after the structure and function of neural circuits in animal brains. The system is constructed with custom digital CMOS circuits and consists of distributed cores of model neurons and synapses that are embedded in a spike-based communication network. Energy consumption associated with high neural fanout is kept at a minimum in the system with the use of distributed on-chip memory arrays that tightly couple synaptic information with the neural datapaths, and with the use of event-driven asynchronous communication circuits that efficiently transfer spikes within and between cores. A compact crossbar memory arrangement along with compact and low-power neuron datapaths add to the energy efficiency of the system and keep the silicon footprint of each core small. Synchronization circuitry between the computing and communication elements ensure that all cores run in lockstep with a software simulator enabling rapid development of algorithms. In a $28nm$ CMOS process 4096 of these cores are packed in a single chip measuring $4.3cm^2$. The chip supports 1 million neurons and 256 million synapses, operates in real time and consumes only $70mW$ during typical network operation. Because of its compact size, high energy efficiency and real-time operating speed, the chip can be used to implement brain-inspired algorithms in a wide range of mobile and embedded systems. For demonstration, the implementation of four canonical neural computations – intensity invariance, pattern decorrelation, associative recall and attentional modulation – is developed in this thesis.

ACKNOWLEDGEMENTS

A wonderful and talented group of individuals have made the past few years very enjoyable for me. I have the following people to thank for their advice in some of the most exciting areas of research and for their company in some of the most memorable times.

Thank you Professor Rajit Manohar for helping me develop an exciting research agenda through your insightful guidance. Thank you for the freedom you have given me and for the opportunities that have come my way because of our work. It was great to have an advisor with so many interesting ideas about so many different fields.

Several other professors at Cornell have influenced my work in very productive ways. Thank you Professor Thom Cleland for helping me develop intuitions about the workings of sensory systems and for your exciting ideas about the applications of our work. Thank you Professor Al Molnar for initially helping me come to grips with neuroscience and for your continued advice. Thank you Professor Kevin Tang for insightful discussions about brain networks and for pointing me towards productive ways to learn dynamical systems theory. Thank you Professor Barbara Finlay for introducing me to some exciting work in brain network development and evolution. Thank you Professor David Field for your insights on sensory coding and for your support and encouragement.

Several researchers at the IBM Almaden Research Center have also had a strong impact on my work. Thank you Dharmendra Modha, Bryan Jackson, Paul Merolla, John Arthur, Rodrigo Alvarez and Andrew Cassidy for your support and advice. I am looking forward to some exciting and productive years of collaboration ahead.

Thank you, Professors David Ahlgren, Mark Silverman, John Mertens and Harvey Picker at Trinity College. Your teaching, guidance and inspiration have been instrumental in my achievements so far.

My experience wouldn't have been complete without some wonderful friends. Kyle, you were a fantastic partner and it's unfortunate our collaboration was cut short. Carlos and Rob, thank you for those amazing tools. James and Kedar, thank you for some very

useful conversations and seminars. Ben, Dan, Raymond and gang, thank you for the hustle on the courts. Jon and Stephen, thank you for being contrasting but entertaining office mates. Filipp and Saber, thank you for a productive few months. The CSL crew, thank you for creating a fun community. Zac, Ali, Kirk, Basit and gang, and the SLC crew, thank you for those poker nights. Ruchira, thank you for being the most amazing person. Andrew, Pranav and gang, Ashesh and gang, Nikolay and gang, Asheaque, Farahin, Lisa and gang, and numerous other friends and acquaintances over the years, thank you for those crazy nights and great conversations.

Finally, thank you Mom, Dad and Tuhina for your support and encouragement, for the inspiration that you provide, and for your endless affection.

TABLE OF CONTENTS

Chapter 1

Introduction

Biological neural circuits are capable of an incredible range of energy-efficient real-time computations. Tasks such as sensory perception, motor pattern generation, autonomous learning and cognitive decision making are carried out by these circuits[1] in the brain much more effectively and efficiently than today's most advanced computer algorithms. Information is processed in these circuits through the spatiotemporal spiking activity of neurons, with different connectivity configurations between neurons with specific dynamical properties implementing distinct algorithms. The human brain consists of approximately 10^{11} neurons and 10^{14} synapses, all contained within a volume of $2L$, weighing less than $4lbs$, and running on a $20W$ power budget [1].

Simulation of brain circuits in general-purpose processors are common and specialized software environments such as NEURON [2] have been developed to facilitate them. General-purpose processors are based on a von Neumann computing architecture, wherein processing and memory are physically separated and instructions are largely executed following a sequential compute model. This architecture is not a natural fit to the parallel and event-driven nature of neural computations. The storage and retrieval of a large number of neuron and synapse parameters from off-chip memory arrays lead to high energy consumption and increased latency, limiting efficiency and scalability. In addition, the computing and communication circuitry of these processors are not customized for modeling the behavior of neurons and their networks. To achieve large-scale implementations using these processors requires supercomputer-levels of computational power and associated costs, for example [3] and [4]. More specialized off-the-shelf hardware such as GPUs [5] or FPGAs [6] can manage some parallel applications more efficiently, but the fine-grained parallelism of brain networks and their memory intensive simulations also

[1]The basic computational units of the brain are nerve cells called neurons.The junction between the output of one neuron and the input of another is called a synapse. A configuration of neurons and synapses is referred to as a neural circuit or a neural network.

render these processors inefficient.

Custom-designed VLSI chips provide a means with which these inefficiencies can be overcome. Neural-inspired Application-Specific Integrated Circuits (ASICs), commonly refered to as "neuromorphic" circuits, are intended to mimic the parallel and distributed function of biological neural systems in real time and approach their compact size/weight and low power consumption. Implementing large numbers of model neurons and synapses in a scalable and reliable platform and within aggressive area and power constraints requires careful design and implementation choices to be made. Efficient design can lead to orders of magnitude better speed, energy consumption and compactness compared to off-the-shelf processors.

Traditionally, neuromorphic designs have used continuous-time analog circuits to model biological components, and digital asynchronous circuits for spike communication [7]. Analog circuits have been popular in the past, since they are compact, and reduce power consumption by directly using the transconductance properties of transistors to mimic the dynamics of neurons. Dense analog circuits however are sensitive to fabrication process variations, ambient temperatures and noisy environments, making it difficult to configure circuits that operate reliably under a wide range of external parameters. This limited correspondence between what the software (the neural algorithm) has been configured to do and how the hardware (the analog implementation) functions adversely affects the usability of such circuits in real-world applications and simulation-based research. In addition, analog circuits do not scale well to deep-submicron CMOS processes [8], in part due to a lack of high-density capacitors and increasing sub-threshold currents..

This thesis describes the architecture, implementation and application of a digital neuromorphic system that was designed to overcome the problems associated with implementing brain-like computations in off-the-shelf digital chips and neuromorphic analog hardware. The following features of the chip described here make it a highly-efficient brain-like processor –

- Parallel computing datapaths, custom-designed to model neuron-like dynamics in compact and low-power circuits, are distributed across the chip.
- Distributed memory arrays storing neuron and synapse configuration parameters are tightly coupled to the computing datapaths minimizing data movement across the chip. Each memory array has a crossbar architecture to efficiently implement a high neural fanout.
- Energy-efficient event-driven routing circuits, customized for spike-based communications, manage distributed communication between the computing and memory elements.
- Synchronization circuits maintain fidelity of freely-configurable algorithms.
- Deep submicron transistor feature sizes allow compact size and high energy-efficiency.

The resulting compact and low-power hardware can mimic large-scale brain-like networks in real-time, enabling new classes of algorithms in a variety of mobile and embedded systems. Example applications include brain-based robotics, neural-inspired pattern recognition systems, neural implants and brain-simulation platforms.

The thesis is organized as follows. Chapter 2 describes the architecture and circuit-level description of a scalable neuromorphic processor that is inspired from the computing and communication features of biological brains. Chapter 3 describes spike-based information coding mechanisms on the chip that reflect the operational principles of biological neural circuits. Chapter 4 describes the configuration of four commonly observed neural computations on the chip. The implementation of these computations are inspired from the hypothesized mechanisms at play in several biological neural circuits, but are not necessarily meant to be an exact replica. Rather, through necessary innovations and simplifications, it is shown that the configured computations are effective means of solving pattern recognition tasks.

Together, these chapters demonstrate how the application of systems neuroscience principles to computer architecture and artificial intelligence can lead to new classes of powerful and efficient devices.

Chapter 2

Neuromorphic Systems Architecture

2.1 Design Factors

Digital ASICs can be customized from the architecture level down to the transistor level to create compact, energy-efficient and real time neuromorphic systems. This section describes the design considerations associated with building such systems.

2.1.1 Computing

The response properties of neurons to synaptic inputs can be captured through various mathematical models. These range from low-level multi-compartment models to high-level phenomenological models [9]. Lower-level models precisely account for the morphological and electrochemical properties of neurons and synapses and capture their dynamical characteristics in detail. In contrast, higher-level models reduce the number of free parameters and capture the neuron's essential properties without accounting for all biophysical details.

These models, expressed as systems of differential equations, are implemented in a digital ASIC through customized circuitry that numerically solve the equations in discrete time. Because of this customization the solution of the differential equations are carried out with substantially higher speed, energy efficiency and compactness compared to that in general purpose processors. Any reduction in model complexity translates directly to simplifications in ASIC circuits and correspondingly reduces computing time, energy consumption and silicon footprint.

Transistors in modern digital circuits have switching speeds in the picosecond scale. Therefore these circuits easily solve the model differential equations within the millisecond timescales of biological neurons. For example, one iteration of the Izhikevich equations [9] with a 16-bit word length can be carried out within $86ns$ in a 65nm ASIC implementation

[10]. Thus a neuron model circuit can be run much faster than biological real time or alternatively, one instantiation of the circuit can be used to solve for multiple neurons (see Sec. 2.3.2).

The energy per neuron update, summed across all neurons, contributes to the overall system energy consumption and therefore should be kept low particularly for neuron models that require a large number of operations per update. For the $65nm$ ASIC neuron (point model) mentioned above, the energy consumption per neuron update is $0.5nJ$. This is a relatively small number compared to the communication energy of the system (see Sec. 2.1.2).

The area footprint is the most significant factor in the design of a neuron circuit. Not only does it directly affect the number of neurons that can be implemented in a given silicon real estate it also affects the energy consumption associated with neural communications. To implement synapses, data storing connectivity information has to travel (often large distances) across the neuron circuits in the system. The large number of synapses in typical neural networks result in high energy consumption associated with this data movement (see Sec. 2.1.2). Smaller neuron circuits shorten the distance that this data moves and therefore increases the energy efficiency of neural communications. The $65nm$ ASIC neuron mentioned above occupies an area of $0.03mm^2$.

2.1.2 Communication

Typical neurons have 10^3 to 10^4 synapses, which make point-to-point connections among the neuron circuits with dedicated wires on an ASIC intractable for any significant number of neurons. However, neuron activity is measured in the Hz range, whereas ASIC wire bandwidth can reach hundreds of MHz to several GHz. Thus, neuromorphic systems [7, 11] time multiplex wire usage between groups of neurons to implement dense interconnectivity. These groups are encapsulated in *neural cores* (Fig. 2.1) and are tiled within a packet routing network (Fig. 2.2).

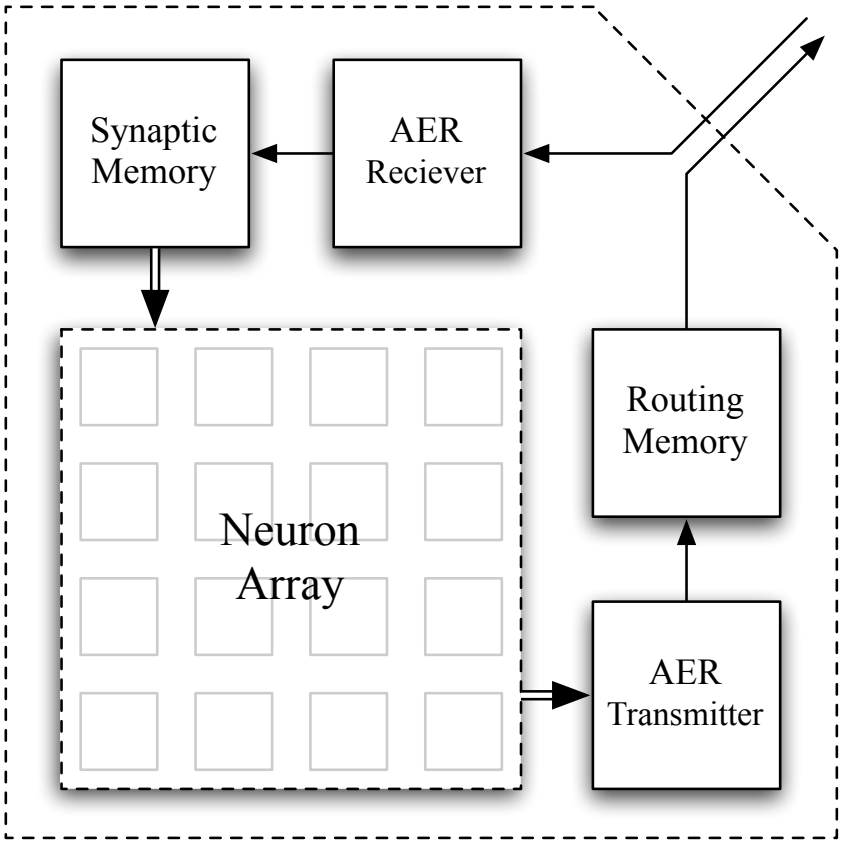


Figure 2.1: A typical neural core with on-core memory for storing connectivity configurations.

Communication circuits within a core time multiplex the connections of all neurons inside the core through a protocol termed Address-Event Representation (AER) [12]. In this protocol, each neuron in the core is associated with a unique address, and a shared I/O bus local to each core communicates all the spikes to and from the core via discretized AER packets. A packet usually takes the form of a (*source, destination*) tuple indicating the addresses of the source neuron and the destination neural core respectively. Some systems use other packet designs, for example packets with multiple destination fields or packets without destination fields where the direction of routing is determined via a memory look up at each routing stage.

The AER packet associated with each neuron is usually stored in on-core memory (illustrated as "Routing Memory" in Fig. 2.1). Packets are accessed from this memory when neurons spike and they are sent out of the core via an AER-packet router [13, 14]. These routers are distributed across the cores and they direct inter-core traffic using the

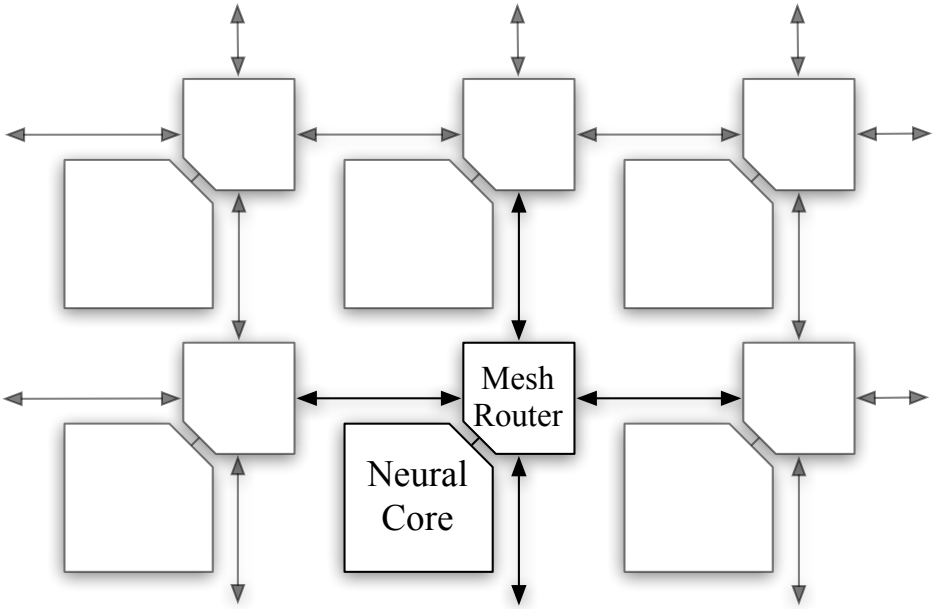


Figure 2.2: Array of neural cores, connected by a 2D Manhattan mesh network.

source or destination fields of the packets. When a packet reaches its destination core its postsynaptic targets (i.e. the neurons inside the destination core that connect to the source neuron) along with the associated parameters (e.g. the synaptic weights) are accessed from off-chip or on-chip (illustrated as "Synaptic Memory" in Fig. 2.1) memory to update the state of the network.

For a target temporal precision (usually 1 millisecond), the speed of this AER communication system (usually 100s of MHz of packet bandwidth) determines the degree of multiplexing that can be implemented as well as the total network size and network activity that can be supported. The area footprint of the communication circuits is relatively small compared to the area occupied by the neurons in the core, but nevertheless should be kept at a minimum to reduce area overhead.

The most critical factor in the design of the communication system is the active energy consumption of data movement. Because the number of synapses exceed the number of neurons by three to four orders of magnitude in typical neural networks, the energy consumption associated with fetching synaptic parameters from off-chip memory

as well as that associated with AER packet traversal is a large determinant of the energy efficiency of the system. For example, in a system of ARM968 processors [15] that communicate across a custom-designed AER routing network and that store synaptic information in off-chip memory, the energy per neural update (Izhikevich equations [9]) is $27nJ$ while the energy per synaptic event is $8nJ$. This means that a neuron with a $1ms$ timestep, firing at $40Hz$ and making 1000 synaptic connections will have over an order of magnitude more communication energy consumption than computing energy consumption. This effect is even more pronounced for custom neuron circuits where computing energy goes down by an order of magnitude (see Sec. 2.1.1).

A variety of ASIC design optimizations can significantly lower the communication energy consumption (see Sec. 2.2). These include for example, tightly coupling large memory blocks to the neural cores to reduce the distances of data travel associated with fetching synaptic configurations, arranging these memory blocks in structures that minimize storage and access requirements, restricting the number of cores that an AER packet can communicate to, customizing routing topologies to reflect the properties of typical neural networks, and using event-driven asynchronous communication circuits that naturally minimize energy consumption during idle periods while maintaining high throughput during bursty periods[1].

2.1.3 Memory

Typical general-purpose multiprocessors have some amount of on-chip cache memory that compliment the computing cores of the processor. In the implementation of neural networks on these processors the on-chip memory is usually used up for the storage of neuron variables, parameters, and update instructions. For example, in the ARM968 implementation [15] mentioned above, all of of the 96 KBytes of tightly coupled memory

[1]In addition, in both ASICs and general purpose multiprocessors, mapping highly communicating neural groups close to each other in the hardware can significantly reduce energy consumption.

associated with each core is used for the storage and update of 1000 neurons. The synaptic parameters in this system are stored in a 1 Gbits off-chip memory table.

As mentioned in the previous section, the energy consumption associated with fetching synaptic configuration parameters from off-chip memory tables is a large factor in the total energy consumption of the system. The energy consumption and latency of accessing off-chip memory in a medium-size chip (in the order of pJ/bit and 10s of ns respectively) are about 2 orders of magnitude higher compared to accessing on-chip memory that is tightly localized with the computing datapaths (in the order of fJ/bit and a 100s of ps). Therefore as mentioned earlier it is highly desirable to incorporate memory blocks storing synaptic parameters close to the neural cores in an ASIC implementation.

With hundreds or thousands of synaptic connections per neuron, the size of the on-chip memory block for synapse storage becomes a critical design factor. For example, in a $65nm$ technology the size of a SRAM bit cell is approximately $0.6um^2$. Assuming 1000 connections per neuron for 256 neurons in a single core and 8-bit synaptic weights, this corresponds to approximately $1.2mm^2$. Assuming an Izhikevich neuron datapath ($0.3mm^2$ in $65nm$ [10]) shared for all the neurons in the core, the synaptic memory area is $4\times$ the neuron datapath area. Thus, reduction of the synaptic memory area, for example by using new memory technologies [16] and compact memory designs (see Sec. 2.2), is a crucial component of the ASIC implementation.

2.1.4 Synchronization

Digital neuromorphic systems can be designed to run discrete-time simulations of neural networks like those that run through software environments on general purpose processors. In such instances a fidelity between hardware operation and an equivalent software simulation is desirable. A one-to-one equivalency between the two allows users to develop and test algorithms in a software simulator of the neuromorphic system on any platform with the guarantee that it will run exactly the same way once mapped on the hardware.

For such equivalency between hardware and software with respect to a global simulation timestep, extra circuitry is necessary inside a neural core to synchronize the integration of presynaptic spikes (coming from a shared non-deterministic routing network) with the generation of postsynaptic spikes in the neurons. Such synchronization circuits, along with necessary bounds to the computation and communication time in the system, can keep hardware operation in lock step with a software simulator (see Sec. 2.2.1).

Alternatively, a system can be designed to run freely without synchronization to a global update signal of a discrete-time simulation. Neural updates can be carried out as AER packets come in, or updates can be continuously made and incoming packets checked for between updates. Without the need to lock to a global timestep, the speed of the system will only be bound by the speed of the circuits. Hardware-software correspondence will no longer be guaranteed due to indeterministic communication sequences.

2.2 Design of a Neuromorphic Core with Integrated Synapses

This section presents the design and implementation of a scalable asynchronous neuromorphic computing core [11] named Golden Gate, specifically describing (i) the asynchronous circuits that mimic central elements of biological neural systems; (ii) an architecture that integrates computation, communication and memory; (iii) the asynchronous communication infrastructure required to accommodate the architecture; and (iv) the synchronization mechanisms required to maintain a one-to-one correspondence with software (this is the first neuromorphic system to demonstrate such an equivalence). A prototype chip consisting of a single core with 256 digital leaky-integrate-and-fire neurons, 1024 inputs, and 1024×256 programmable binary synapses implemented with a SRAM crossbar array is described. The entire core fits in a 4.2mm^2 footprint in IBM's 45 nm SOI process and consumes 45pJ per spike in active power.

Two primary factors are behind the efficiency of the core. First, an on-chip memory array that stores synaptic parameters is integrated in close proximity to the neurons

minimizing data movement across the system. A crossbar architecture in the memory array is used to implement high neural fanout with low storage requirements. Second, an asynchronous design style results in circuits that naturally mimic the distributed event-driven processing of neural networks and ensures that power dissipation of inactive parts of the system are kept at a minimum.

2.2.1 Architecture and Operation

Neurons and Synapses

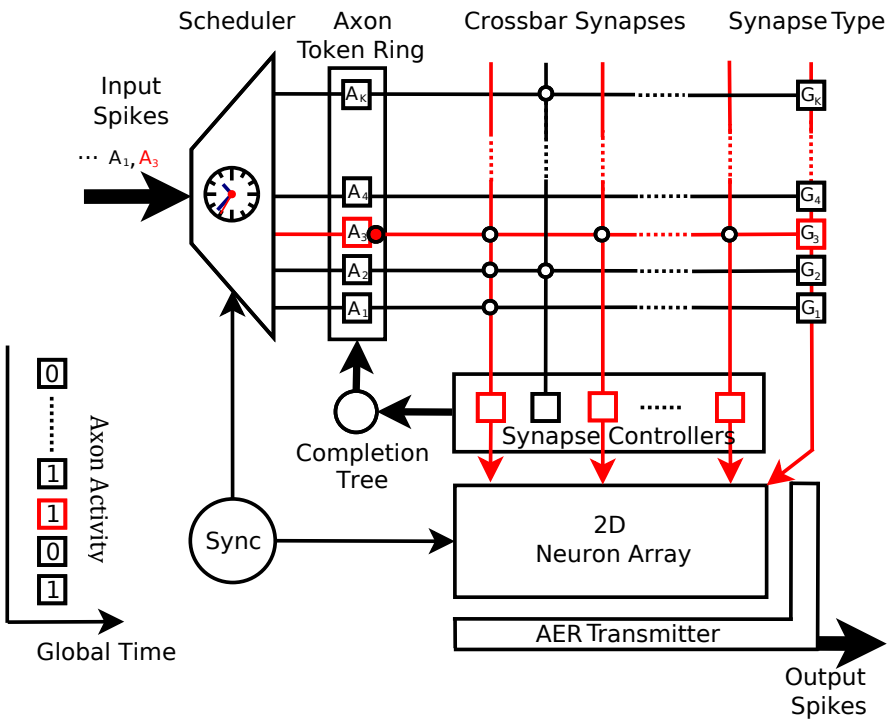
The leaky integrate-and-fire model of a neuron captures the behavior of real neurons in a range of situations and offers an efficient implementation. This neuron model is used as the basic computational unit of the core.

The neurons in the chip are interconnected through axons and synapses. Each axon may correspond to the output of a neuron in the same core or somewhere else in a large system of many cores. Some axons may also be driven by embedded sensors or some external driver. The connection between axon j and neuron i is represented as S_{ji}. Each axon is parameterized by a type G_j that can take one of three different values indicating the type of synapse (e.g. strong excitatory, weak excitatory or inhibitory) that the axon forms with neurons it connects to. Each neuron is parameterized by a leakage current L, a spike threshold θ and three synapse weights W^0, W^1, W^2 that correspond to the different axon types. All these parameters are configurable during start-up.

The core implements a discrete-event simulation where the neuron states are updated at each timestep according to external input and interconnectivity. The state of neuron i at some time t, is represented by its voltage $V_i[t]$, while the state of axon j is represented by its activity bit $A_j[t]$.

The parameters and state variables of the system are tabulated in Fig. 2.3. Neuron i receives the following input from axon j:

$$A_j[t] \times S_{ji} \times W_i^{G_j}.$$



NAME	DESCRIPTION	SIZE AND TYPE
V_i	Voltage of Neuron i	10 bit signed variable
$W_i^{0..2}$	3 Synaptic Weights of Neuron i	9 bit signed constant
L_i	Leak of Neuron i	9 bit signed constant
θ_i	Threshold of Neuron i	8 bit unsigned constant
S_{ji}	Connection between axon j and neuron i	Binary constant
G_j	Type of Axon j	3 distinct constants
A_j	State of Axon j	Binary Variable

Figure 2.3: *Top:* Architecture of the neuromorphic core with K axons and N neurons. Each junction in the crossbar represents a synapse between an axon (row) and dendrite (column). Each neuron has a dedicated column in the crossbar. Active synapses are represented by an open circle in the diagram. An example sequence of events in the core is illustrated. The scheduler accepts an incoming address event and communicates with the axon token-ring. The token-ring activates axon 3 (A3) by asserting the third wordline of the SRAM crossbar array. As a result, a synaptic event of type G3 is delivered to neurons N_1, N_3, and N_M. The AER transmitter sends out the addresses of these neurons if they consequently spike. *Bottom:* State variables and parameters of the system. All values are represented as integers, and all constants are configurable at start-up.

The neuron's voltage is updated at each time step by subtracting a leakage from its voltage and integrating the synaptic input from all the axons:

$$V_i[t+1] = V_i[t] - L_i + \sum_{j=1}^{1024}(A_j[t] \times S_{ji} \times W_i^{G_j}).$$

When $V[t]$ exceeds a threshold θ, the neuron produces a spike (represented by a digital '1' in its output) and its voltage is reset to 0. Negative voltages are clipped back to 0 at the end of each time step.

Communication Infrastructure

At the heart of the neurosynaptic core is a crossbar memory that forms the synapses between the axons and the neurons. This embedded memory allows large synaptic fanout without resort to off-chip memory, drastically reducing energy consumption and latency associated with accessing synaptic parameters (See Sec. 2.1.3). The crossbar array is user configurable and arbitrary networks can be set up in the system (e.g. Axons 1, 2, and 3 are connected to the first neuron in the 2D neuron array in Fig. 2.3). Each row of the crossbar corresponds to an axon, each column corresponds to the input of a neuron (the dendrite), and the junctions are binary synapses implemented by a two terminal memory cell (e.g., SRAM). Thus each of the N neurons may get up to K synaptic inputs depending on the activity in the axons and the configuration of the crossbar. K was chosen to be 1024 and N was chosen to be 256 resulting in 1024×256 crossbar synapses and an enormous configuration space. Each row of the crossbar memory has two additional bits that indentify the type of the axon. The postsynaptic neurons weigh a synaptic event on an axon with one of three distinct 8-bit weights based on this type information.

Spikes events are sent to and from the core using *address-event representation* (AER) packets (See Sec. 2.1.2 and Sec. 2.3.1). On the output side, an AER transmitter [12] encodes spiking activity by sending the locations of active neurons through a multiplexed channel, leveraging the fact that the bandwidth of wires (easily larger than 100s of MHz) is orders of magnitude larger than the bandwidth of biological axons (in the 10s of Hz

range). The spikes can be sent off chip, or routed to an axon of another core via a memory table. On the input side, an AER receiver delivers incoming spikes to the appropriate axon at a time determined by a configured field in the AER packet. As spikes are serviced sequentially, their addresses are decoded to the crossbar where all 256 synaptic connections of an active axon are read out in parallel.

Because the crossbar structure implements the synaptic connections of an axon using a 256-bit (+2 bits for the type information) vector, the need to store and access 256 multi-bit synaptic weights for each synapse is eliminated. As a result, the energy consumption and area associated with implementing synapses is reduced. These are achieved however at the cost of reduced flexibility since the type and delay of an axon in the crossbar structure has to be the same for all postsynaptic targets.

Discrete-time Operation

An example sequence of operation in the core is illustrated in Fig. 2.3. The operation has two phases during each time step.

The positive edge of a global synchronization clock (Sync) initiates the first phase of operation. In this phase, address-events along with their time stamps are sent to the core and are received by the scheduler. The scheduler evaluates the time stamps and asserts the appropriate axons that go into a token-ring. The units in the token-ring that receive active axons assert the rows of the crossbar in a mutually exclusive manner. Once a wordline in the crossbar is activated all the neurons that are connected to the axon (corresponding to the 1's in the row) receive an input spike along with information about the type of the axon. The neurons update their voltages as axon events come in. The first phase needs to complete within the first half of the global synchronization clock (that usually has a period of 1 millisecond) – a precise margin in which neural updates need to complete for potentially all 1024 axon inputs.

In the second phase of operation, the negative edge of the synchronization clock is detected by all the neurons. On receiving this event, neurons whose voltages have

exceeded their respective thresholds produce spikes in their output ports. The spiking addresses are encoded by the AER transmitter and sent out of the core sequentially. This phase needs to complete within the other half of the global clock – i.e. the AER transmitter has to service 256 potential spikes within the global timestep.

A 1 millisecond global clock period (typical temporal precision in biological neural networks) means that the performance requirements of the circuits in the two phases of operation are easily met. Breaking neural updates into two phases ensures that the hardware is always in sync with an equivalent software simulation at the end of each time step. Specifically, the order in which address-events arrive to the core or exit from the core can be variable due to resource arbitration, especially when events are sent through a non-deterministic routing network. To preserve one-to-one correspondence, the different orderings must not influence the spiking dynamics, and this is achieved in the system by first accounting for all the synaptic events and then checking for spikes.

2.2.2 Circuit Design

The neuromorphic architecture described above was implemented using asynchronous quasi-delay-insensitive (QDI) circuits [17] that are synchronized with the global timestep. A QDI design style leads to extremely robust circuits that remain operational under a wide range of process, voltage and temperature variations, making them ideally suited for mobile, embedded applications.

In this section the concurrent processes of the architecture is described using Communicating Hardware Processes (CHP - see Appendix) that can be synthesized into QDI asynchronous circuits using Martin's synthesis method [17].

Scheduler

The scheduler receives packets from outside the core and delivers spikes on the axons at specific times. The packets may come from spiking neurons in the core or from outside the core. In addition to these packets, the scheduler also receives a clock and a global

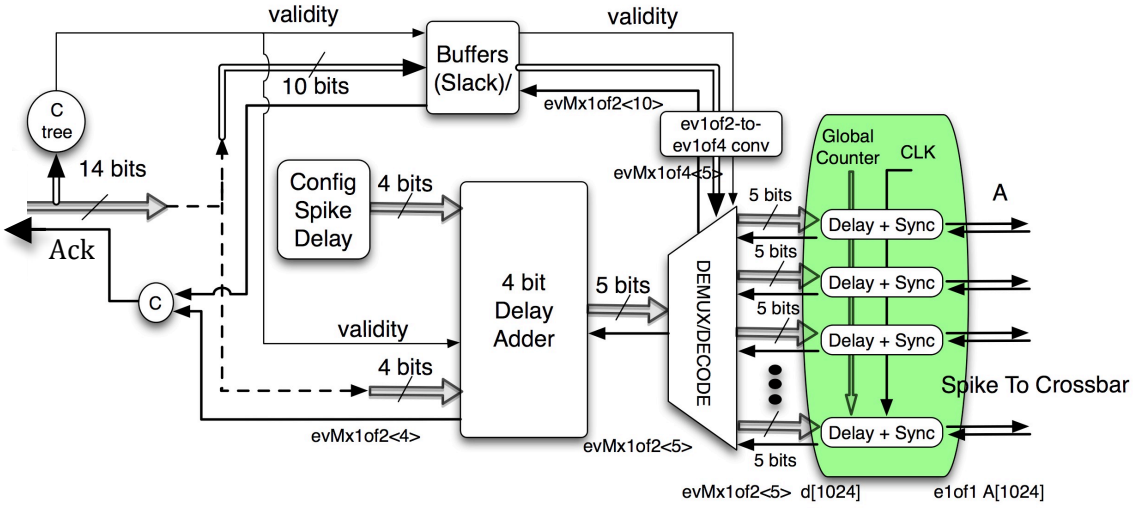


Figure 2.4: Internal structure of the Scheduler. The overall function is: (1) Receive input packets; (2) Add the stored axonal delay to the time field of the packet to get the spike delivery time; (3) Demultiplex the address field of the packet; and (4) Deliver the spike to the appropriate axons at the edge of a global clock when the global timestep equals the computed delivery time of the spikes.

counter time. The block also stores a configurable axonal delay (inside an SRAM array) for each of the axons. Each packet (y) coming in contains an axon address ($y.dest\_addr$) and a time of spike ($y.dt$ - in units of clock ticks). The scheduler decodes the packet to determine where the spike should be delivered (the axon number). The time in the packet is added to the axonal delay for the specified axon, and this value is compared against the current global counter time on every tick of a clock that the scheduler receives. When the time matches, a spike is delivered to the crossbar. This makes the spike delivery to the crossbar synchronous with the global system clock. The CHP of the processs is:

$Scheduler \equiv$

```
*[[IN?y;
    axon_delay := SRAM[y.dest_addr];
    delay_y := y.dt + axon_delay;
    [delay_y = timestamp];
    A[y.dest_addr]!spike
 ]]
```

The scheduler implements the axonal delay stored in a 1024 (number of axons) by 4 (range of delays) SRAM array. It receives the packet in the channel IN, adds the axonal delay to the time in the packet, waits for the global time to reach this value and then delivers the spike to the axon corresponding to the address in the packet. Besides implementing the axonal delay, this procedure synchronizes input spikes to the core with the clock edge, implementing the first of the two phases of operation (Section 2.2.1) that allow the system to produce 1-1 correspondence with a software simulator.

The internal blocks of the scheduler are illustrated in Fig. 2.4. For the prototype chip, a common delay block was used instead of a full 1024×4 SRAM array to implement the axonal delay. This fixed delay is a 4 bit number that can be configured at the chip's startup, and is the delay value for all axons. The Adder adds this delay to the time value in the packet and passes it to the DEMUX/DECODE unit. The control for the DEMUX/DECODE unit is the destination address in the input packet, slack-matched to the SRAM and the adder. The DEMUX then outputs 5 bits indicating the final timestamp for spike release to the axon. This value goes to 1 of 1024 [Delay + Sync] blocks. These blocks must contain full buffers at the input because all the hardware used prior to this stage is time-shared and the DEMUX output channel has to be "released" as soon as possible to allow assertion of the next spike. Each [Delay + Sync] unit compares the timestamp of the global timer with the obtained final timestamp and initiates a spikes to an axon when the value matches.

The clock synchronization is contained in the [Delay + Sync] unit. The synchronization part is not trivial and requires some design effort. The data obtained from the DEMUX/DECODE must be aligned to the clock without any knowledge of the data arrival time in relationship to the clock edge. Precautions have to be taken in order to avoid potential metastability during synchronization with the global clock. Such metastability may occur since there is no timing correlation between the edge of the global clock and the packets arriving from the router. A modified two flip-flop synchronization scheme is used in the design [18].

Axon Token-Ring

At an edge of the synchronizing clock, the scheduler pulls up the axon lines that have spikes. Each axon has a corresponding row (a wordline) in the crossbar memory array. The dendrites (inputs) of a neuron correspond to a column (bit line) of the crossbar array. Since a dendrite can potentially connect to multiple axons, the rows of the crossbar have to be asserted in a mutually exclusive manner. This function is carried out by an array of axon servers that implement a token-ring mutual exclusion algorithm [19]. Each server has an axon as its input and the wordline of the crossbar as its output. A token circulates among the servers to give them mutually exclusive access to the crossbar.

When an axon is asserted, its server requests its neighbor to pass the token. The request propagates through the token-ring, and the token is passed along to the requesting server. Upon the arrival of the token, the server asserts the corresponding row of the crossbar. The CHP of an individual server is given below. Channel A communicates with the axon from the scheduler and channel WL with the crossbar, while channels U and D communicate with the neighbor above or below. The local variable b represents the token. The D port of the last server is connected to the U port of the first server. The channel C communicates with a completion tree (see Fig. 2.3) that indicates when all the neurons have completed processing events for a particular axon.

$Axon\_Server \equiv$

$$
\begin{array}{l}
*[[\overline{A} \longrightarrow [b \longrightarrow skip \;\; [] \neg b \longrightarrow D!; b\uparrow \;\;]; WL!; \\
\qquad\qquad [\overline{C} \;\; \longrightarrow \;\; C; \;\; A?] \\
\quad | \, \overline{U} \longrightarrow [b \longrightarrow skip \;\; [] \neg b \;\; \longrightarrow \;\; D! \;\;]; b\downarrow; U? \\
\;\;]]
\end{array}
$$

Crossbar Memory

The states of the memory cells of the crossbar array represent the synaptic configuration of a network (i.e. which axons connect to which neurons) and in part determine the unique properties of that particular network. Organizing the synapses in this crossbar structure allows an active axon to fan out to potentially 256 neurons in parallel through a single control operation. For large connectivity, this reduces the dynamic power consumption and accelerates the speed at which the network is updated.

The configuration of the crossbar has to be set up prior to the operation time of the chip. Shift register scanners were included to configure the bit cells of the array. Standard 6T SRAM cells were used as the bit cells. The axon token-ring controls the wordlines of the bit cells while a set of synapse controllers interfaces the bitlines with the neurons. The CHP of a synapse controller unit is given below.

$Synapse\_Controller\_Unit \equiv$

$$
\begin{array}{l}
*[[BL.t \longrightarrow N!; \;\; C! \\
\quad [] \;\; BL.f \longrightarrow \;\; C! \\
\;\;]]
\end{array}
$$

When the axon token-ring drives one of the wordlines of the crossbar, one of the two bitlines of each cell in the corresponding row will discharge asserting either the $BL.t$ or the $BL.f$ wires of the synapse controller. If $BL.t$ is asserted the controller communicates with the neuron corresponding to the column to update the neuron's state. Once this communication is over, or if $BL.f$ was the wire originally asserted, the controller

communicates with a completion tree. The right-most controller receives the "type" information for each axon (they are stored in the last 2 bit cells of each crossbar row) and communicates this information to the neurons. Once all synapse controllers have completed their operation, the completion tree communicates with the C channel of the axon token-ring, after which the token-ring unit with the token releases the wordline of the crossbar and token becomes free to travel. Another token-ring unit with an active axon will then get the token and assert its corresponding wordline.

Neuron

The neurons are the basic computational units in the system. They were implemented using dedicated circuits (non-multiplexed), allowing all the neural updates to happen in parallel. This parallelism comes at the cost of relative density inefficiency that is in part mitigated by the use of a dense technology.

The design of the neuron circuits needs to accomodate two conflicting requirements. On the one hand, a neuron must service events quickly to avoid holding up the crossbar. On the other hand, a neuron receives very few events in typical scenarios, so it must not burn dynamic power when it is idle. A purely event-driven design is therefore ideal: it has a fast service time (100MHz-GHz range), but only burns power during the rare occurrence of an event.

During the first phase of operation (Section. 2.2.1) each neuron receives event-driven synaptic input from the crossbar memory synapse controller. The neurons update their state (represented by an internal voltage V) by integrating the incoming synapses (Section. 2.2.1). During the second phase of operation, the neurons synchronize with the global clock edge, at which point they output a spike (represented by a 1 bit output) if their voltage is above threshold or leak out an amount of the voltage if it is below threshold. The synchronization of the neuron during the second phase of operation, along with the synchronization of the scheduler during the first phase (Section. 2.2.2) ensures that the operation in the core is in 1-1 correspondence with a software simulator.

The internal parameters of the neurons (the threshold, the leak, and the three synaptic strengths) are configured at start-up. They are all represented with 8-bit integers in 2's compliment form. The internal voltage of a neuron is represented by a 10-bit integer also in 2's compliment form.

The operation of the neuron is decomposed into control and datapath blocks. A block diagram of the processes involved is illustrated in Fig. 2.5.

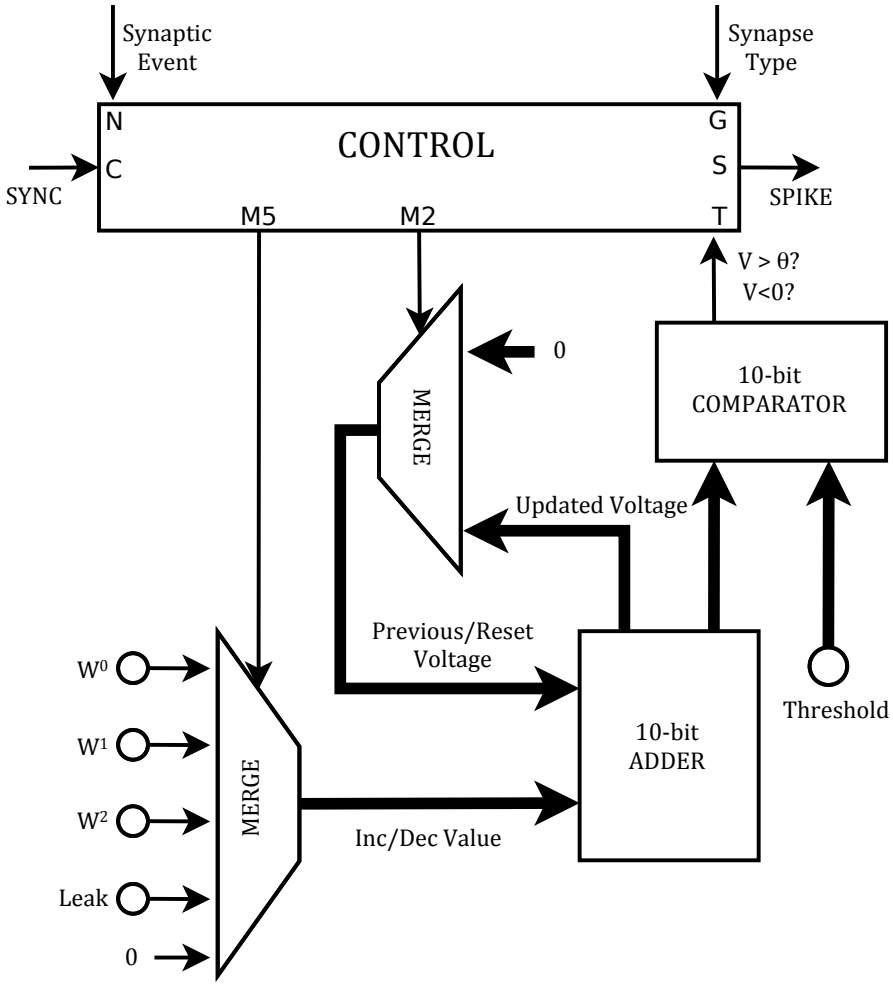


Figure 2.5: Block diagram of the neuron. The control block interfaces with the crossbar, directs information flow in the datapath, synchronizes with the global time step and outputs a spike when the neuron voltage exceeds its threshold. The datapath elements update the voltage after each synaptic event and check for a spike when directed by the control.

The CHP of the control block is:

$Neuron\_Control \equiv$

$$
\begin{aligned}
*[[\overline{N} \longrightarrow\ & G?x; \quad [x = Type0 \longrightarrow M5!0 \\
& \quad\ [] x = Type1 \longrightarrow M5!1 \\
& \quad\ [] x = Type2 \longrightarrow M5!2 \\
& \quad\ ], M2!0; N \\
[]\overline{C} \longrightarrow\ & T?y; \quad [y = spike \longrightarrow S, \quad M5!3, \\
& \qquad\qquad\qquad\ M2!1 \\
& \qquad\ [] y! = spike \wedge (y <= 0) \longrightarrow \\
& \qquad\qquad\qquad\ M5!4, M2!1 \\
& \qquad\ [] y! = spike \wedge (y > 0) \longrightarrow \\
& \qquad\qquad\qquad\ M5!3, M2!0 \\
& \qquad\ ]; C \\
]]&
\end{aligned}
$$

The control block interfaces with the synaptic input coming from the neuron's dedicated crossbar column through the channel N. Upon a communication request on this channel, the control reads in the "type" information of the axon through the channel G that connects to the synapse controller representing the last two columns of the crossbar. Before completing the handshake in N the control communicates with the datapath of the neuron through channels $M5$ (that relays synaptic type information) and $M2$ (for voltage control in the datapath). Once these communication actions have completed N and G are acknowledged and the next cycle of synaptic inputs may come in.

In the second half of the global timestep, the neurons receive a synchronization event in the channel C (that is driven by a process that uses one of the edges of the global clock to initiate a handshake). When this event comes in, the control initiates a communication with the datapath through the T channel. The datapath sends back one of three distinct values indicating whether the voltage for the current timestep is above threshold, below threshold but above zero; or below zero. If the voltage is above threshold the control communicates with the AER transmitter through the channel S to send out a spike.

Through $M2$ the control also instructs the datapath to reset the voltage to 0 if there is a spike or if the voltage is below 0. If the voltage has not been reset to 0, the control instructs the datapath to apply the leak to the current voltage.

The CHP of the datapath units are:

$MERGE\_5 \equiv$

$$
\begin{aligned}
&*[[\overline{M5} \longrightarrow M5?d; \\
&\qquad [d = 0 \longrightarrow AI1!W^0 \\
&\qquad []d = 1 \longrightarrow AI1!W^1 \\
&\qquad []d = 2 \longrightarrow AI1!W^2 \\
&\qquad []d = 3 \longrightarrow AI1!L \\
&\qquad []d = 4 \longrightarrow AI1!0 \\
&\qquad]; \\
&]]
\end{aligned}
$$

$MERGE\_2 \equiv$

$$
\begin{aligned}
&*[[\overline{M2} \longrightarrow M2?d; \\
&\qquad [d = 0 \longrightarrow AO0?x;\ \ AI0!x \\
&\qquad []d = 1 \longrightarrow AI0!0 \\
&\qquad]; \\
&]]
\end{aligned}
$$

$ADDER \equiv$

$$
\begin{aligned}
&*[\overline{AI0} \wedge \overline{AI1} \longrightarrow AI0?x,\ \ AI1?y;\ \ z := x + y \\
&\ []\overline{AO0} \longrightarrow AO0!z \\
&\ []\overline{AO1} \longrightarrow AO1!z \\
&\ ]]
\end{aligned}
$$

$COMPARATOR \equiv$

$*[\overline{T} \longrightarrow AO1?V;$

$[V > Threshold \longrightarrow T!0$

$[](V < Threshold) \wedge (V <= 0) \longrightarrow T!1$

$[](V < Threshold) \wedge (V > 0) \longrightarrow T!2$

$];T$

$]$

When the control drives $MERGE\_5$, the process forwards either a synaptic strength (during the integration phase), a leak (during the resetting phase if $V > 0$) or the value 0 (during the resetting phase if $V <= 0$) to one of the inputs of the $ADDER$ process. When the control drives $MERGE\_2$, the process forwards either the previous voltage (during integration) or the value 0 (if $V > threshold$ or $V < 0$ during the firing phase) to the other input of $ADDER$. The $ADDER$ process is a 10 bit adder that sends out the sum of its inputs to the $COMPARATOR$ and the $MERGE\_2$ processes when they request it. The control drives $COMPARATOR$ when it needs to evaluate the state of the neuron voltage.

AER transmitter

Spikes from the 2-dimensional array of neurons are sent out of the core through token-ring AER transmitter circuits [12] that allow all the neurons to share one output channel. In this scheme, a set of row servers and column servers circulate tokens in each dimension of the neuron array and give spiking neurons mutually exclusive access to the shared communication channel. A counter keeps track of the location of the tokens, and sends out neuron addresses upon request. This methodology leads to compact transmitter circuits capable of efficiently servicing clusters of spiking activity.

The design of the transmitter is illustrated in Fig 2.6. The sequence of operation is: (1) A spiking neuron asserts a row request line; (2) The corresponding row server requests for the row token from its neighbors; (3) As the row token moves, the counter

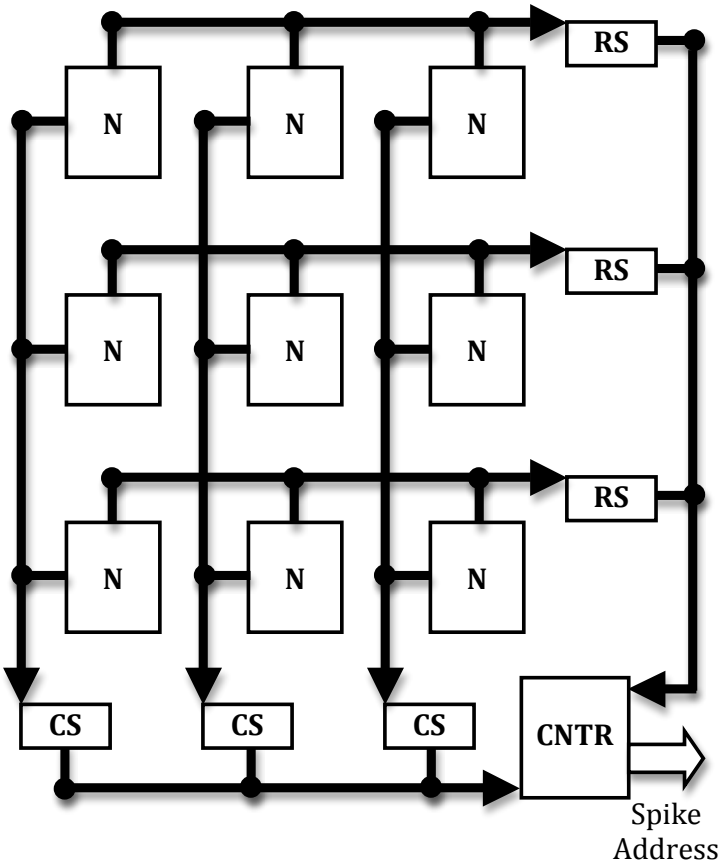


Figure 2.6: Illustration of the AER transmitter architecture using a 3×3 sample neuron array. When a neuron (N) has a spike, it communicates with its corresponding row server (RS) and column server (CS). A counter (CNTR) keeps track of circulating row and column tokens and sends out the spike address through the shared output bus.

keeps track of its position; (4) Upon receipt of the token, the row server acknowledges the row request; (5) The neuron then asserts a column request line; (6) The corresponding column server requests the column token from its neighbors; (7) As the column token moves, the counter keeps track of its position; (8) Upon receipt of the token the column server communicates with the counter to send out the row and column token addresses and then acknowledges the column request; (9) The neuron does a second communication with the row server to indicate the completion of service.

The neurons interface with their respective servers via open-drain shared row request lines and shared column request lines. The servers also communicate with the counter via shared wires. These wires need to be carefully implemented since transitions are shared across processes and span an entire dimension of the neuron array.

2.2.3 Results

As demonstration of the architecture, one core (minus the scheduler) was implemented in a 45 nm SOI process. The fabricated chip occupies $4.2mm^2$ of silicon (Fig. 2.7, left). Each of the 256 neurons in the core occupies 35μm $\times$ 95μm. Each SRAM bitcell in the 1024×256 synaptic crossbar array occupies $1.3\mu m^2$ (plus another 1.9 μm^2 associated with conservatively-designed periphery). A custom printed circuit board allows the chip to interface with a PC through a USB link (Fig. 2.7, right).

Active Power

The primary focus during the design was the reduction of active power since passive power can be addressed through fabrication options and active leakage reduction circuitry. The purely event-driven nature of the core results in very low power consumption since active power in such a design style is dependent only on the typically low activity rates of the neurons. The QDI methodology allows aggressive reduction of the operating voltage in the core without affecting chip functionality. As shown in Fig. 2.8, the core consumes only 45pJ of active power per spike at $V_{dd} = 0.85$.

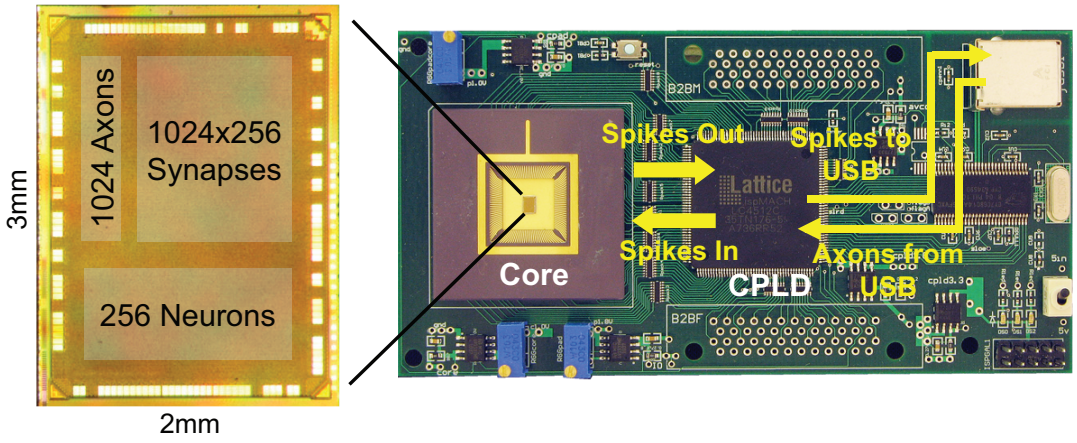


Figure 2.7: *Left:* Die measures 2mm × 3 mm including the I-O pads. *Right:* Test board that interfaces with the chip via a USB 2.0 link. Spike events are sent to a PC for data collection and can also be routed back to the chip via the CPLD.

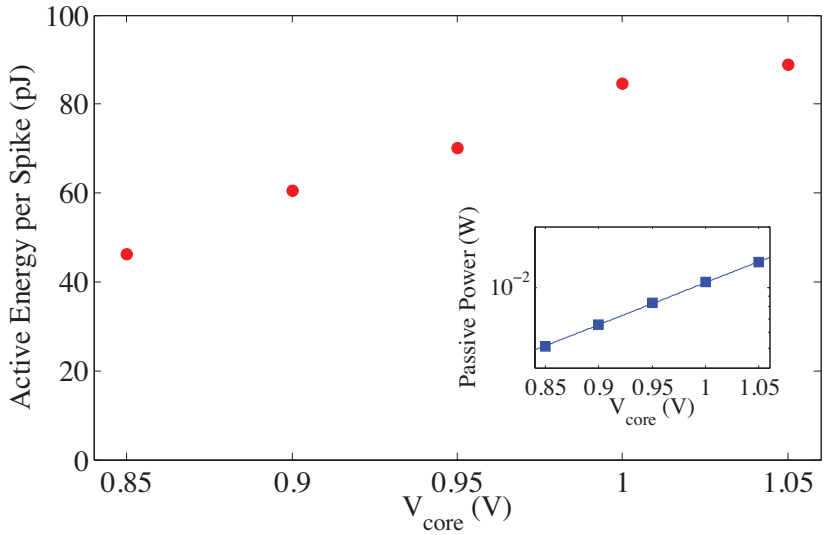


Figure 2.8: Active and passive energy consumption in the core.

One-to-One Equivalence

To test that the chip satisfied one-to-one equivalence, the synaptic strength and leak values of each neuron were set to +1 and the thresholds were set to to +100. Random connectivity was generated where each synapse had a 20% probability of being active. The CPLD was set to route all neuron spikes back into the core (neuron 0, 1, 2 drove axon 0, 1, 2, respectively), creating a complex recurrent network. After 100 time steps in the chip, all the neurons spike in unison due to their identical positive leaks. This first barrage of spikes is routed back around to the axonal inputs, activating the random pattern of excitatory recurrent connections. These inputs cause neurons to spike earlier in the next cycle, thereby having a de-synchronizing effect. Within a few cycles, the recurrently-driven activity dominates the dynamics leading to a complex spatiotemporal pattern. Results from a software simulation of a network with an identical configuration confirmed that the software and hardware have identical behavior (Fig. 2.9).

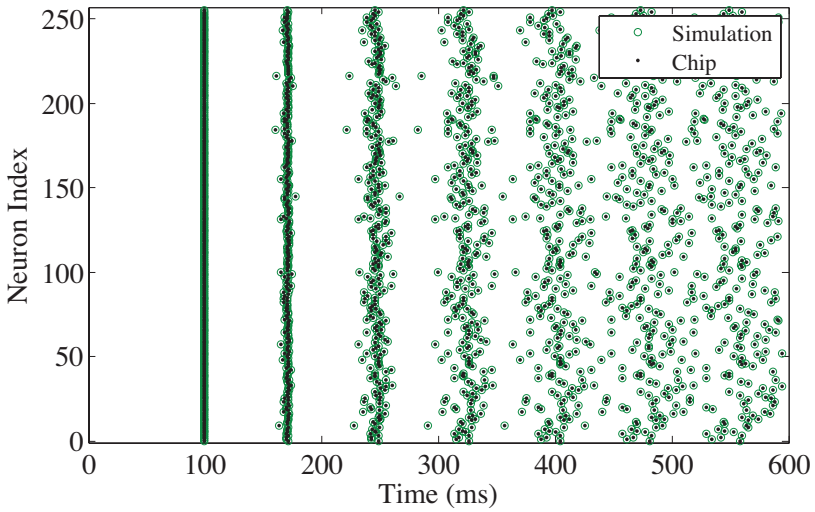


Figure 2.9: The spiking dynamics of the chip exactly matches software simulation when configured with the same parameters and recurrent connectivity. Spikes are plotted as dots for measured chip data and circles for simulation.

2.3 Design of a Multi-Core System

A multi-core chip consisting of multiple instantiations of Golden Gate-like (Sec. 2.2) cores was fabricated in a 28 nm CMOS process [20]. The chip, named True North, consisted of 64×64 cores and a total of 1 million neurons and 256 million programmable synapses. For effective multi-core implementation, the design of the core was slightly modified from the design of Golden Gate. A router block was added to each core for the communication of spikes between cores. The number of axons in a core was reduced from 1024 to 256 to reduce the footprint of the core. The availability of many neurons and axons across numerous cores meant that this reduction did not have a significant effect on the types of networks that can be accomodated. Further reduction in the footprint of the core was achieved by multiplexing, via a control block, the datapath circuits that are used to update the states of the neurons in a core. The scheduler block (Sec. 2.2.2), that was not included in the 45 nm implementation (Sec. 2.2.3), was added to each core.

This section describes the design of True North, focusing on the communication and control circuitry required for efficient large-scale spike-based computing.

2.3.1 Spike Routing

The cores were tiled in a 2-dimensional routing mesh and the router blocks of each core formed the backbone of the mesh. When a neuron of a core spikes, an associated AER (see Sec. 2.2.1) routing packet (stored in the local memory of the core) is sent to the local input terminal of the router. The router uses the information in the packet to direct it in one of five directions – east, west, north, south or local. Each router was also responsible for communicating traffic coming from the neighboring cores in each direction of the 2-dimensional mesh. A block diagram of the router is shown in Fig. 2.10.

The AER routing packet produced by a spiking neuron is shown in Fig. 2.11. The packet encodes the address of one axon anywhere in the system. 5-bit dx and a dy fields in the packet relatively address the destination core by encoding the number of routing

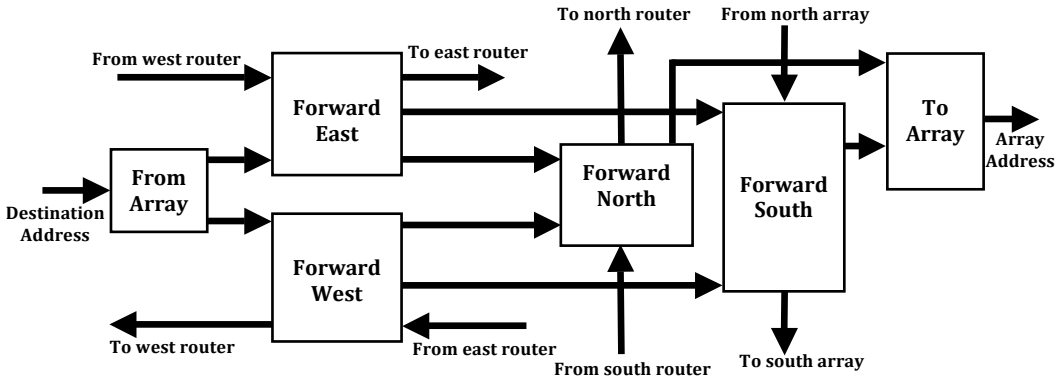


Figure 2.10: Block diagram of the router associated with each core. The "To Array" and "From Array" blocks communicate packets to and from the local core. The "Forward East" and "Forward West" blocks communicate packets coming from each of the horizontal directions in the mesh. The "Forward North" and "Forward South" blocks do the same for the vertical directions in the mesh.

hops in the horizontal and vertical directions of the mesh respectively. At each hop, the router decrements the value in the *dx* field (for a horizontal hop) or the *dy* field (for a vertical hop). A 4-bit *dt* field in each packet encodes a delivery time that allows users to configure axonal delays. A 8-bit *axon* field in each packet encodes the address of the axon in the crossbar memory (see Sec. 2.2.1) of the destination core. The *dt* and *axon* fields are decoded by the scheduler block of the destination core to assert the destination axon at some time of the simulation. A necessary condition to maintain 1-1 correspondence between hardware and software (see Sec. 2.2.1) is that the routing network routes the spike to the destination core within *dt* ticks of the global timestep.

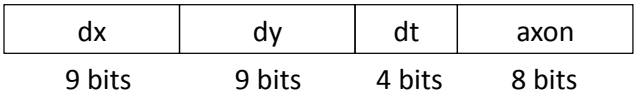


Figure 2.11: The AER routing packet associated with each neuron.

The AER packet design along with the on-chip crossbar memory structure (Sec. 2.2.1) results in efficient spike-based communication. When a neuron spikes it sends out one

packet addressing only one axon of the whole system instead of addressing its postsynaptic neurons individually. The drawback of this is that the postsynatpic targets of a neuron has to be restricted to one core (and 256 neurons) of the system[2]. As noted in Sec. 2.2.1, the integrated crossbar synaptic memory inside the core results in a drastic reduction of distances that data has to travel across the system compared to an implementation that uses off-chip memory.

The router was constructed with asynchronous QDI circuits that naturally mimic the event-driven nature of neural communications. In a typical application, neurons in a core will stay inactive for relatively long periods of time with intermittent bursts of spikes caused for example by some change in the external environment that the neurons represent. Spikes therefore are not expected to create regular traffic patterns in the router and therefore a synchronous design would result in energy overheads or extra circuit complexity.

The circuit-level description of the router and its deadlock free operation in the mesh are described in [14].

Circuitry for chip-to-chip spike communication at the edges of the chip enable the scalability of True North to multi-chip systems. These circuits implement serialization and deserialization processes that multiplex AER packets crossing chip boundaries through limited I/O pins. Before exiting a row (or column) of a mesh, each packet is tagged with a 6-bit value to identify the row (or column) of the destination mesh that the packet is to be delivered to.

The serialization and deserialization processes were implemented with asynchronous merges and asynchronous controlled splits as shown in Fig. 2.12. The splits at the destination use the 6-bit tag to direct the packet towards its destination in the receiving chip. With these processes at all four edges of a chip, multiple chips can seamlessly tile together without any modification to the routing processes inside a chip.

[2]This restriction can be overcome using multi-stage synaptic communications using multiple neuron instantiations to simulate one neuron.

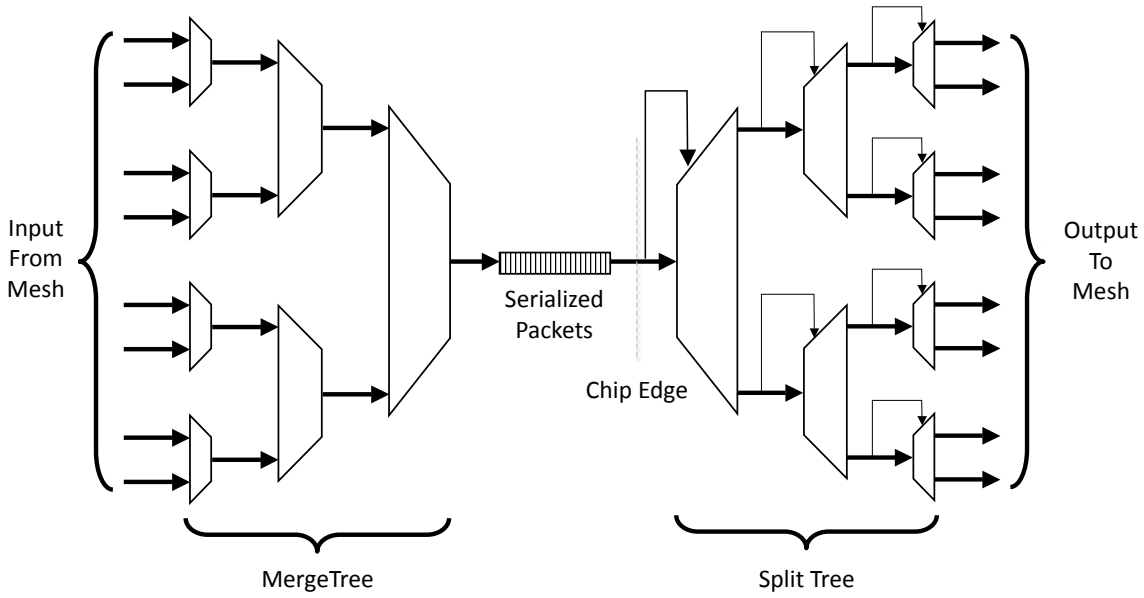


Figure 2.12: Chip-to-chip serialization/deserialization process illustrating the communication of AER packets across chip boundaries. The merge tree adds a tag to the AER packet and the split tree uses the tag to direct the packet towards its destination. The padded bits are stripped off before the packets enter the receiving mesh.

2.3.2 Core Control

Whereas Golden Gate (Sec. 2.2) has 256 parallel neuron datapaths per core, True North uses one multiplexed datapath block to sequentially update the state of all 256 neurons in a core. This design choice lowers the area footprint of the neurons and consequently enables the implementation of a more complex neural update equation [21] within aggressive area constraints. The latency overhead associated with multiplexing is not significant because neural updates are in the millisecond scale (orders of magnitude slower compared to transistor switching speeds) and all 256 updates are carried out well within 1 millisecond unless held up by the spike communication system (see below). The variables and parameters of each neuron in a core reside in a memory array that is integrated with the synaptic memory array (Sec. 2.2.1) of the core. The memory block is tightly localized with the datapath block, and with the small feature sizes of a 28 nm process, the energy overhead associated with data movement from the memory onto the multiplexed neural

datapaths is not significant.

Parallelism is maintained at the multi-core level with 4096 cores operating in parallel and communicating through the mesh routing fabric.

A control block inside each core coordinates the activity of the other blocks to implement the multiplexing. This is carried out with the following steps and is illustrated in Fig. 2.13.

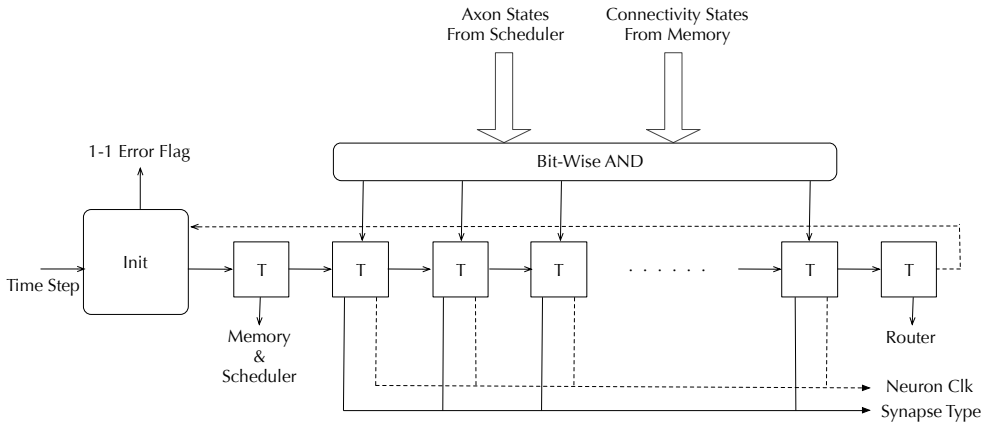
1. At the positive edge of the global synchronization signal ("Time Step" in Fig. 2.13a) the control communicates with the memory to assert the wordline of one memory row. Consequently, the variables and parameters of a neuron are read by the neuron datapath, the state of the configured connections of the neuron are read by the control, and the value of the output AER packet of a neuron is buffered on the local input terminal of the router. The control also accesses the state of the axons in the given timestep from the scheduler. A sub-block of the control carries out a bit-wise AND between the state of the connections and the state of the axons.

2. Through a series of blocks (represented by T in Fig. 2.13a) that implement token-ring mutual exclusion [19], synaptic integration is carried out in the neuron datapath for each active synapse (the output of the bit-wise AND). Each of these blocks store the type information of its corresponding axon. An axon can be one of four types representing different strengths of excitatory or inhibitory inputs to the neuron (similar to the synaptic integration in Golden Gate described in Sec. 2.2). At the arrival of the token in a token block, a clock signal is generated and the synapse type communicated to the synchronous neuron datapath (see below) if the token block has an active synapse. At the end of the period of the generated clock, the token block passes the token to its neighbor.

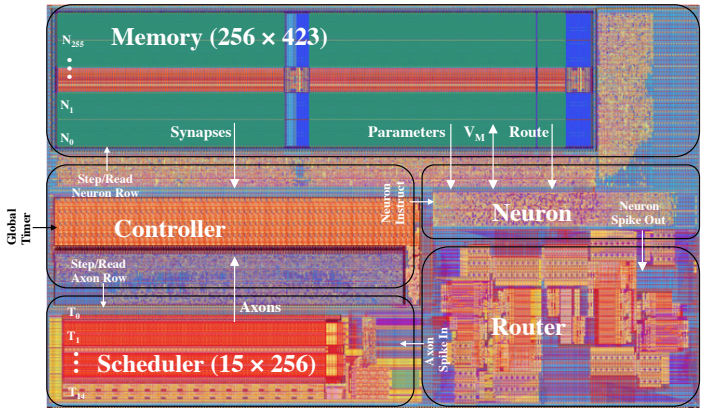
3. After all axons have been accounted for, a final token block communicates with the router. The router in turn communicates with the neuron, and in the event of a spike the buffered AER packet is sent out from the core.

4. The updated state of the neuron is stored back into memory.

5. The steps are repeated for 256 neurons in sequence. Note that in the event of high traffic congestion in the local routing network, the token can circulate back to the block that communicates with the router before the router is able to send out the packet of the previous spiking neuron. In such a scenario the system will stall until the communication network frees up.



(a) Controller Sub-Blocks



(b) Core Layout

Figure 2.13: (a) The sub-blocks of the controller consist of an initialization block (Init) that initializes update at the arrival of the global timestep, a set of token blocks (T) that communicate with other elements of the core, and a bit-wise AND block that determines the state of the synapses. (b) Layout of the core illustrating its operation.

The control sub-blocks (Fig. 2.13a) are designed with asynchronous event-driven circuits that naturally fit with the other event-driven communication blocks of the system (namely the router and the scheduler). This design style is especially effective in the token blocks of the control since the state of the synaptic inputs are not known apriori – they are dependent on the network activity and the connectivity configuration. Instead of having to wait for predefined clock periods, the asynchronous token blocks swiftly skip inactive synapses thereby completing the neural updates at a faster pace. The extra speed is especially critical at times of spike traffic congestion in the local routing network that can hold up progress of sequential neural updates, as described above, and endanger the 1-1 correspondence feature of the chip.

In the event that traffic congestion slows down the operation of the control to a point where it is unable to carry out all 256 updates within the global timestep (1 ms), an error signal is sent out of the control to indicate a breakdown of 1-1 correspondence between hardware and software. In this scenario the user would be required to reconfigure the network (e.g. by mapping the network more efficiently or reducing the network activity) if 1-1 correspondence is to be maintained.

In contrast to the control, router and scheduler blocks, the neuron block was constructed with synchronous circuits. This allowed the use of commercial design tools that are more streamlined than asynchronous design tools. In addition, the reduced number of wires in a synchronous design compared to a QDI asynchronous one (where multiple wires are required to represent one bit) resulted in a smaller footprint of the datapath elements.

2.3.3 Results

True North was fabricated in Samsung's $28nm$ process (Fig. 2.14 a and b) [20]. Each chip consists of 64×64 cores tiled in two dimensions and occupies an area of $4.3cm^2$. Each core measures $390um \times 240um$ (Fig. 2.13b), 18 times smaller than the core of Sec. 2.2.

A probabilistically-connected recurrent network with neurons firing at $20Hz$ and 50% utilization of the crossbar synapse array was configured onto each of the cores. At $0.775V$ the chip consumed $70mW$, corresponding to $26pJ$ per synaptic event and $70uJ$ per millisecond timestep [20]. This is four orders of magnitude lower than a general-purpose processor running the same model [22].

With each chip supporting a million neurons and 256 million synapses, multi-chip systems (Fig. 2.14c) can be constructed to implement millions of neurons and billions of synapses. The system thus provides a platform for implementing scalable brain-like computations in compact, energy-efficient and real-time systems.

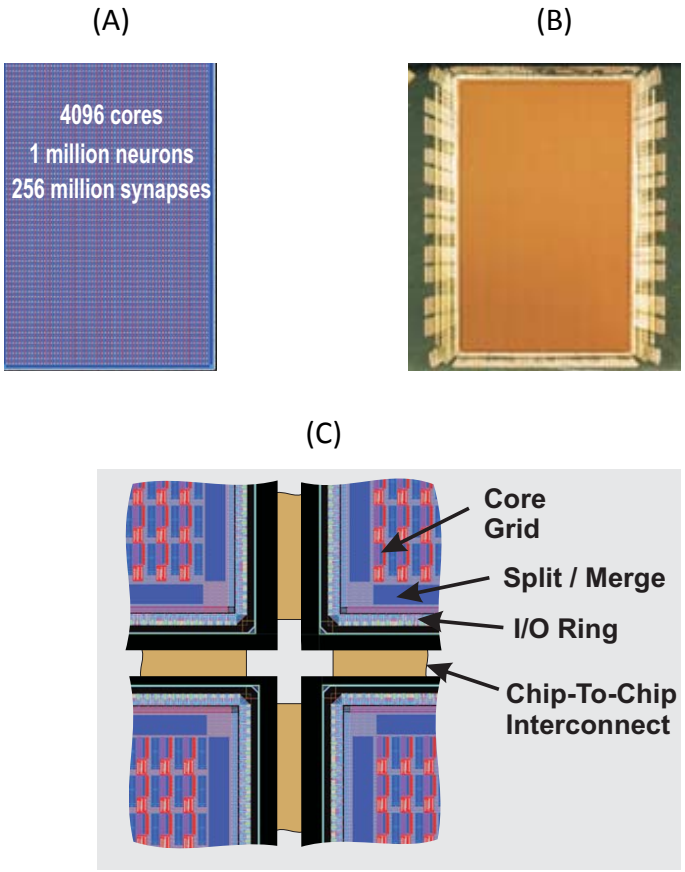


Figure 2.14: (a) The layout of the chip consists of cores tiled in a 64×64 mesh. (b) Picture of chip die. (c) Illustration of chip tiling for a multi-chip system.

Chapter 3

Spike-Based Information Coding

Trains of spikes between neurons communicate information across a neural circuit. These spike trains are encoded as either rate codes or temporal codes in a single neuron or in a population of neurons. Codes are distinguished by their time scale of operation[1], their tolerence to noise, and their energy efficiency. This section describes the implementation of these spike codes in Golden Gate (the neuromorphic core of Section 2.2). The neural computations described in Section 4 use these codes.

3.1 Rate Codes

Neurons may change their average firing rate over some interval of time as the strength of their inputs are varied. Such firing rate modulations are ubiquitous across the brain [23]. Examples of neurons that use firing rate codes include retinal ganglion cells, olfactory sensory neurons, and motor neurons at neuromuscular junctions.

The neurons in Golden Gate support a rate coding mechanism. As the rate of AER packet receipt (firing rate stimulus) of a neuron increases, the neuron integrates faster towards its spike threshold and consequently increases its rate of AER packet transmission (firing rate response).

Three parameters (freely configurable by the user) in the neuron affect the output transmission rate given a receipt rate. These are the periodic leak (L), the firing threshold (θ), and the weight of the synaptic input (W). They determine the number of excitation cycles required to reach threshold. This value is ceil$[\theta/W]$ with $L = 0$, where ceil$[x]$ refers to the rounding of x to the next highest integer.

For a neuron with $L = 0$, the time between two consecutive AER output packets p_o

[1]Spike codes in biological neural circuits typically have precisions of a few milliseconds to hundreds of milliseconds. The precision of spike codes in Golden Gate is bound by the global time step of operation.

(the period of spiking) can be expressed in relation to the time between two consecutive AER input packets p_i as

$$p_o = ceil[\frac{\theta}{W}]p_i \tag{3.1}$$

The output firing rate f_o (equals $1/p_o$) of the neuron is thus related to the input firing rate f_i (equals $1/p_i$) as

$$f_o = ceil[\frac{W}{\theta}] \times f_i. \tag{3.2}$$

A non-zero transmission rate of a neuron with zero leak is thus linearly related to the receiver rate with a slope determined by the value of W/θ as shown in Fig. 3.1.

The amount of traffic in the local AER routing network of a neuron affects how large its AER transmission/receipt rates can be. The maximum rate is bounded by the temporal precision (the period of the global synchronization signal of Fig. 2.3 – usually set to 1 ms) of the system.

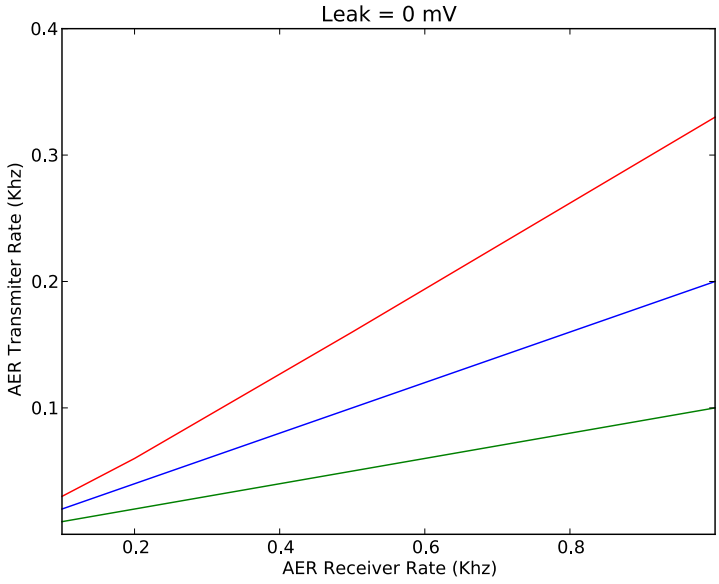


Figure 3.1: Firing rate code with zero leak of a Golden Gate neuron. Different colors represent different values of W/θ. Higher slopes correspond to higher ratios.

An input-independent temporal component to the neural voltage integration is added by setting a non-zero leak. In between AER inputs, there is now a decrease of the neuron voltage. The net excitation of the neuron between two consecutive excitatory synapses is $W - (L \times p_i)$. The number of excitatory synapses required to excite the neuron to a voltage of θ is

$$\frac{\theta}{W - (L \times p_i)}. \tag{3.3}$$

The number of excitatory synapses required to excite the neuron to threshold is

$$ceil[\frac{\theta}{W - (L \times p_i)}]. \tag{3.4}$$

The relationship between p_i and p_o is thus

$$p_o = ceil[\frac{\theta}{W - (L \times p_i)}] \times p_i, \tag{3.5}$$

and I/O firing rate relationship is

$$f_o = ceil[\frac{\theta}{W - (\frac{L}{f_i})}] \times f_i. \tag{3.6}$$

As f_i changes, the slope of the $f_o - f_i$ curve encounters discrete jumps because of the ceil[] term. For example, consider a neuron with $\theta = 30mV$, $L = 1mV$ and $W = 10mV$ that has one input. As the period of the input changes from $p_i = 4ms$ to $p_i = 3ms$ the number of presynaptic spikes required to excite the neuron to threshold stays at 5. However as p_i reduces further to $2ms$, the number of presynaptic spikes required for a postsynaptic spike changes to 4. These abrupt changes correspond to changes in the slope of the $p_o - p_i$ (and $f_o - f_i$) curve.

Rate codes are robust to interspike-interval noise when spikes are averaged over wide time windows. Encodings over narrower windows are less tolerant because of erroneous inclusion or exclusion of spikes that may result from spike-time jitters.[2]

[2]Because of the high signal-to-noise ratio of digital circuits, spike-time jitters in digital

A population of neurons can represent a vector with a firing rate code, with individual neurons representing individual components of the vector. A population rate code in Golden Gate is represented in Fig. 3.2, where a group of 8 neurons is driven by a 8-dimensional input signal.

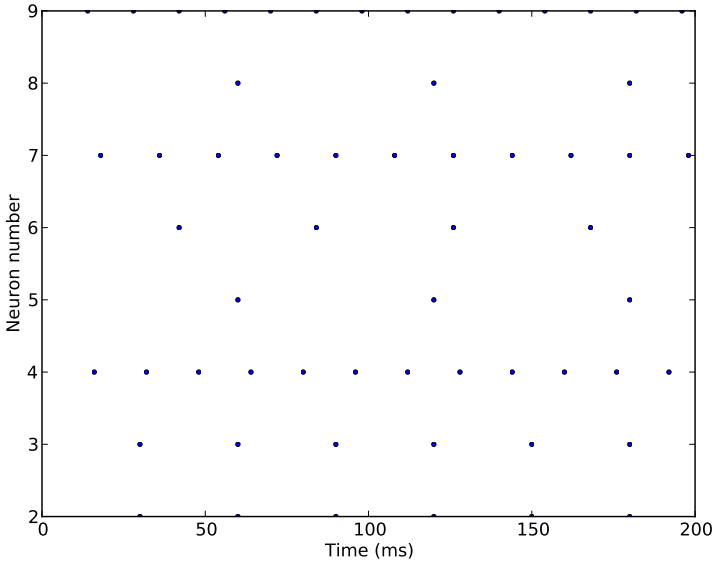


Figure 3.2: Peri-stimulus time histogram illustrating population rate coding in Golden Gate. The presynaptic spike rate of each neuron is chosen from a Gaussian distribution with mean = 100 Hz, standard deviation = 50 Hz.

3.2 Temporal Codes

A *temporal code* enables faster and more energy-efficient processing compared to a rate code by using the precise spatiotemporal pattern of individual spikes in a population of neurons to encode information. This type of code however may have a lower tolerance to interspike interval noise especially when defined over fine timescales. Temporal codes have been observed in numerous neural circuits, such as those in the hippocampus, the olfactory bulb, and the primary visual cortex [23].

---

neuromorphic systems are solely from the noise associated with external signals or internal random number generators.

In Golden Gate, the timings of neuronal spikes are maintained across the system within a precision determined by the global synchronization clock (usually 1 ms). This feature, along with the neuronal leak and the axonal time delay, can be exploited for neural processing using temporal codes.

One form of a temporal code is a *latency code*. Since the strength of excitation of a Golden Gate neuron translates to how fast it integrates its interval voltage, the time t_s between the start of a stimulus and the generation of a spike can code for input strength. For a neuron with a threshold of θ, a leak of L and one synaptic input with weight W and period p_i, t_s (related to Eq. 3.3) can be expressed as

$$t_s = \frac{\theta}{W - (L \times p_i)} \times p_i. \tag{3.7}$$

A common temporal reference frame is necessary in a population of neurons coding for a stimulus vector using latency codes. In-vivo such a reference frame may be provided for example by oscillatory potentials arising from shared interactions across a population [23]. Neurons lock their spike times to a constant phase of the oscillation that reflect their activation levels and therefore this coding scheme is also referred to as a *phase code*. One way of mimicking this mechanism in Golden Gate is by periodically driving a common inhibitory rhythm that connects to all neurons in a population and resets their voltages to zero. This implementation is illustrated in Fig. 3.3 where a rate-coded input vector is transformed into a latency code. The timing of a neuron's spike after inhibition is released is dependent on the strength of its input. Multiple spikes within one oscillation cycle from the same neuron is prevented by "handshaking" with a spiking neuron's input that results in a shut down of the input until the start of the next cycle.

Another type of temporal code that is related to a phase code but does not necessitate an explicit temporal reference frame is the interspike interval code. Here, information is encoded by the time between two or more consecutive presynaptic spikes. In its simplest form, a neuron may code for the timing difference between two excitatory synapses. If

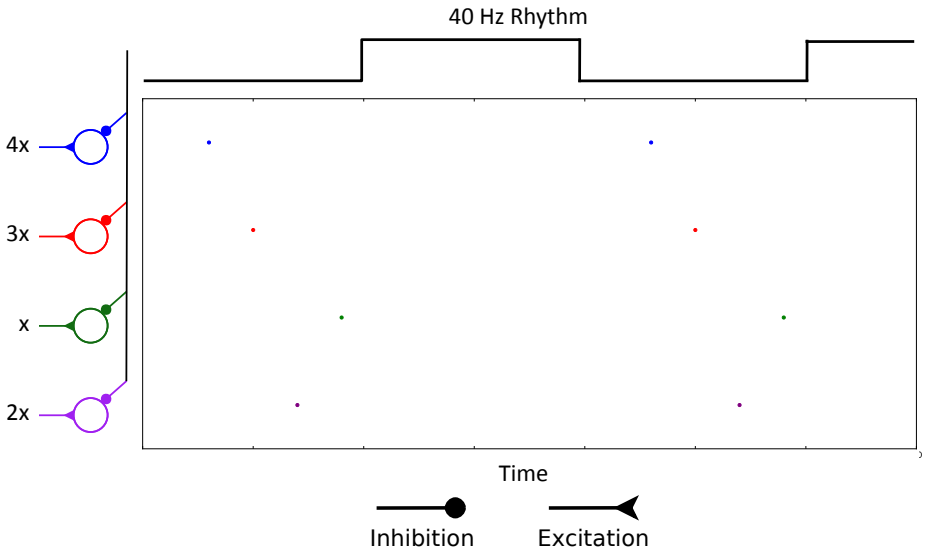


Figure 3.3: Latency code of Golden Gate neurons under a common inhibitory rhythm.

the presynaptic spikes come within some time window, the neuron spikes. Otherwise, the reduction in voltage from the leak of the neuron prevents it from crossing its spike threshold voltage. Such a neuron is called a coincidence detector, and the boundaries of the time window it detects is set by its internal parameters. The time to spike t_s of a coincidence detector with two inputs of same weight can be expressed by Eq. 3.7, with p_i representing the period between presynaptic spikes in the two input terminals.

Coincidence detectors along with axonal delays in Golden Gate can be utilized to process time varying features in a neuronal network. An example is provided in Fig. 3.4 where the movement of a sound source is represented by a population of neurons. Sound may arrive at different times to two sensors located at opposite sides of Golden Gate. A set of coincidence-detecting neurons (Fig. 3.4(a)) arranged systemically along axonal delay lines inside the chip can detect this difference (each input of each neuron is represented by a distinct axonal line in the chip). For example, the top-most neuron has a short delay line from the left sensor but a long delay line from the right sensor. When the sound source is exactly adjacent to the right sensor, the axonal delay in the right sensor path compensates for the time lag between the arrival of the sound in the two sensors. The neuron would thus receive coincident inputs and respond maximally, decoding the

location of the sound source. Neural circuits in the auditory pathways of birds use this mechanism for sound localization [24]. As the sound source moves from the left of the core all the way to the right in a semi-circular trajectory and then back to its original position (Fig. 3.4(b)), distinct neurons in Golden Gate spike representing the source in neural space (Fig. 3.4(c)). A software simulation running the same algorithm confirms that our chip is in 1-1 correspondence.

Interspike interval coding is not restricted to coincident detectors in Golden Gate. For example, the postsynaptic spike rate of a neuron can be modulated based on the temporal proximity of its presynaptic spikes.

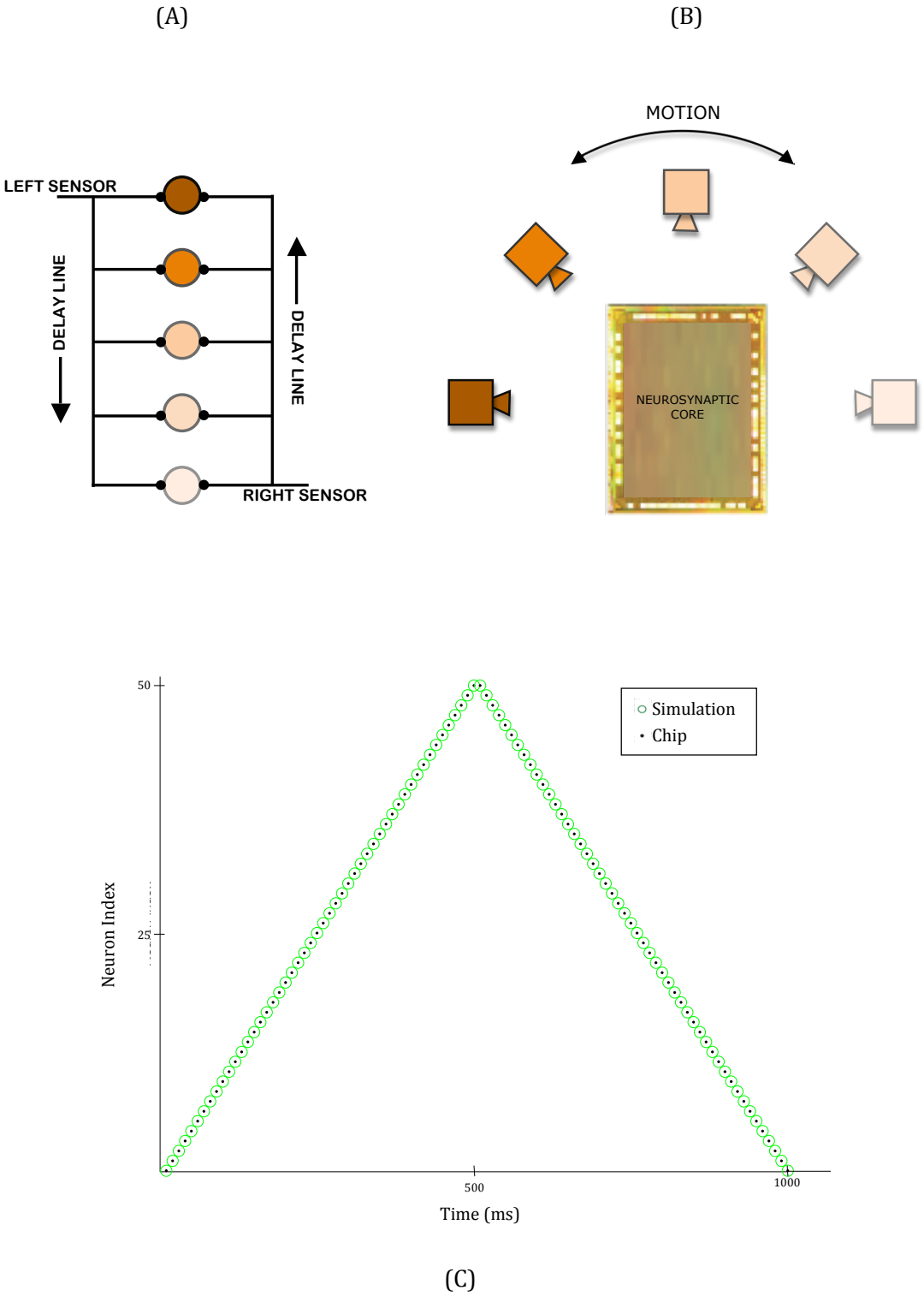


Figure 3.4: Sound localization using axonal delays and coincidence detection. (a) Systemic arrangement of axonal delay lines. (b) Motion of sound source across the chip. (c) Spatiotemporal spike representation of the sound source in neural space.

Chapter 4

Canonical Neural Computations

The brain uses a set of canonical neural computations to implement common operations for solving different problems. These computations are observed in a wide range of neural systems across multiple species and they form modules with which larger systems are constructed. Implementing them in Golden Gate (the neuromorphic core of Section 2.2) is a step towards using the system for brain-like perception, control and cognitive tasks.

The mechanisms with which these computations are implemented vary between different regions of the brain and between the brains of different species. Here we use the mechanisms available in Golden Gate to mimic those computations. Four canonical neural computations are illustrated. Each section describes the what, where and why of a computation and its implementation in the chip.

4.1 Intensity Invariance

In the brain, the stimuli of a population of neurons can vary across a large range of intensities. For example, across visual environments light intensities range over 10 factors of 10 [25]. The visual system recognizes the identity of objects despite this variation in illumination levels. Similar examples abound across the brain [25], such as odor recognition across a wide range of concentrations in the olfactory system and texture recognition across a wide range of pressures in the somatosensory system. Neurons in the brain and in Golden Gate therefore need to recognize patterns of activity across a large dynamic range.

4.1.1 Relational Representations

The firing rates of neurons in both systems have an upper limit. In the brain, the time constants of the molecular processes underlying spike generation bound the maximum

firing rate of typical neurons to approximately 1 kHz. In Golden Gate the maximum firing rate of a neuron is also limited to 1 kHz by the synchronization clock (the global timestep) that needs to be slow enough to allow the multiplexed communication circuits to reliably transfer spikes[1]. The available dynamic range of firing rate responses of neurons in both systems is equal to the encoding time window of the firing rates. For example with an encoding window of 10 ms, a neuron can fire with one of 10 distinct rates between 0.1 kHz and 1 kHz. Therefore, encoding large dynamic ranges necessitates large encoding time windows that reduce the information processing speed of the system. Alternatively, large neural populations can be employed to distribute the dynamic range in space (i.e. use lots of neurons to represent a wide range of stimuli) with a corresponding increase in area and energy requirements of the system.

To represent and recognize patterns across large dynamic ranges without incurring these overheads, neural circuits use *relational representations* of stimulus values. That is, instead of representing the absolute level of input, a neuron's activity reflects the amount of input relative to the input received by other neurons in a population. The operating points of neurons (the region of high slope) in their rate-coded input-output graph will shift based on the global level of activity in such a representational scheme. A wide dynamic range can be represented by a neuron with a narrow dynamic range by such modulations of the operating point. Stimulus identity is thereby encoded in the relative population activity that remain largely invariant to absolute changes in intensity.

4.1.2 Global Inhibition

A possible mechanism with which operating points can be modulated and relational representations created is via a global inhibitory network that inhibits the activity of

[1]In the prototype Golden Gate chip, the 1 ms period of the global synchronization clock is much larger than what is necessary to communicate all spikes between 256 neurons. However, for the larger multi-core/multi-chip system described in Sec. 2.3, the full period of the clock will often be necessary to communicate all the spikes across the system within a timestep.

every neuron based on the mean level of input on the population. Such a mechanism, illustrated in Fig. 4.1, reflects properties of neural circuits in the olfactory bulb [26] and primary visual cortex [25] where principal neurons are inhibited by interneurons that increase their activity in proportion to the global level of afferent input.

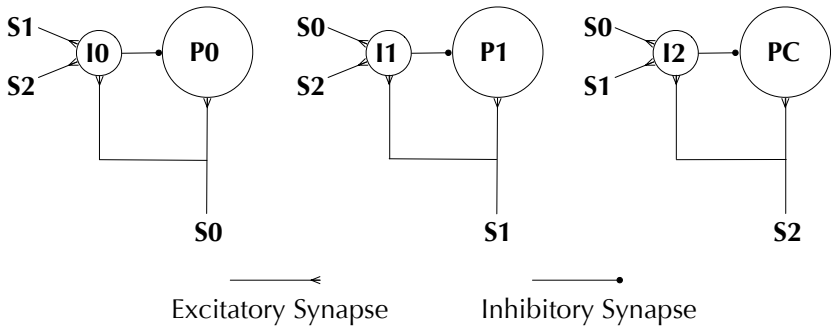


Figure 4.1: Illustration of a global inhibitory network for creating relational representations between three principal neurons. Inhibitory interneurons ($I0$, $I1$, $I2$) deliver inhibition proportional to the global amount of input activity ($S0$, $S1$, $S2$) on the principal neurons ($P0$, $P1$, $P2$). As a result principal neurons reduce their activity in proportion to the total input.

Golden Gate can be configured to implement inhibitory networks like Fig. 4.1. Fig. 4.2 shows the I/O curve of a neuron in the chip when it is part of such a network of 10 principal neurons (P cells), each inhibited by an interneuron (I cell) and excited by a firing rate stimulus[2] (S). The I cell network is fully connected as in Fig. 4.1. When the mean level of input in the population is 0.1 kHz, the high-slope region of the curve falls around the input value of 0.1 kHz. As the mean activity increases, the curve shifts its operating point to higher input levels. Relational representations of the input across wide dynamic ranges are created in the neural population in this way. The parameters used in the network are presented in Table. 4.1.

A drawback of networks such as Fig. 4.1 is that they do not scale efficiently with the dimensionality of the stimulus because of all-to-all connections among the I cells. Better scalability is achieved by implementing dense *local* connections among I cells as in Fig. 4.1

[2]S is represented by 10 inputs with the same average firing rate.

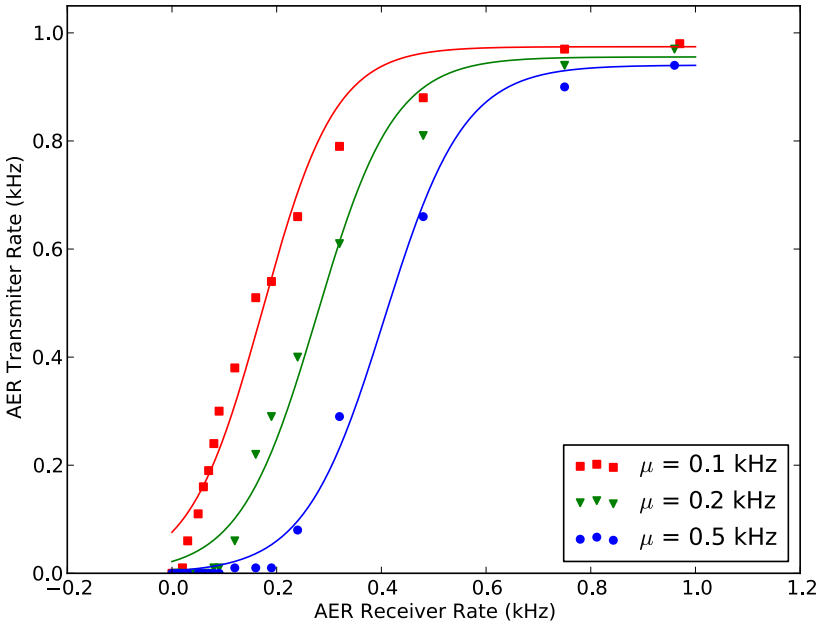


Figure 4.2: Operating Point Modulation. The high-slope region of a neuron's rate-coded I/O curve shifts based on the mean level of activity of a neural population. As a result a relatively narrow dynamic range of an individual neuron is used to represent input with a large dynamic range via relational population activity.

Table 4.1: Parameter Values for Fig. 4.1

Neuron	Threshold (mV)	Leak (mV)	W_{exc} (mV)	$W_{inh}(mV)$
Principle Neuron	21	1	20	-50
Interneuron	25	16	20	-

along with a few longer range projections via additional shared interneurons (S Cells). Each S cell is driven by I cells in its local neighborhood and delivers excitation onto other I cells at various locations in the network. High clustering of the I cells and the long-range projections via the S cells result in a short average path length in the inhibitory network and consequently a uniform level of I cell activation across the network. Consequently a uniform level of inhibition (proportional to the total amount of stimulus drive) is delivered onto each P cell – a small-world network effect. This structure is depicted in Fig. 4.3, where the S cells and the I cells are shown.

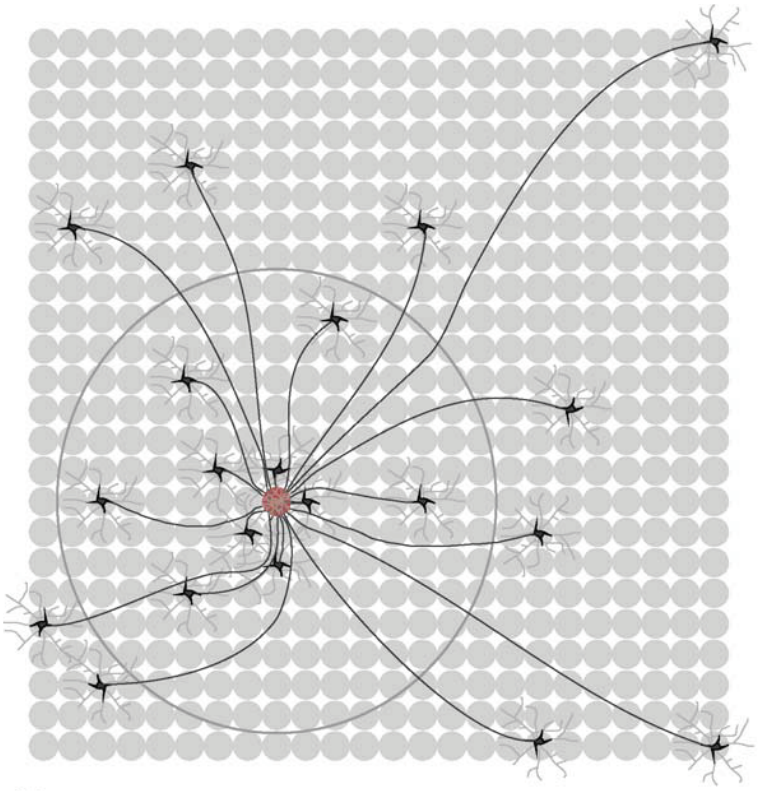


Figure 4.3: An efficient arrangement for providing global inhibition. S cells (black) are driven by I cells (grey) in its local neighborhood. The axons of the S cells branch to various distances, although for clarity this is not shown. Each I cell receives inhibition from S cells various distances away. This is depicted in the I cell shown in red. The short average path length between the I cells as a result of the S cell projections result in their activity being approximately uniform across the network. As a result, each P cell in the network receives a uniform level of inhibition without the need for all-to-all connectivity among the I-cells.

Biological neural circuits may use this solution [26] to reduce the metabolic costs associated with providing global inhibition. The inhibitory effects of such a network structure configured in Golden Gate is depicted in Fig. 4.4 [27]. The results shown are for a network of 48 P cells, 48 S cells, and 96 I cells divided into two classes – one that is driven by the same stimuli as a corresponding P cell and that drives local S cells, and the other that is driven by S cells from various distances and that delivers inhibition onto a corresponding P cell.

The reduction in the spiking activity of the network due to global inhibition is quantified in Fig. 4.5. The global inhibition provided by different degrees of connectivity among the inhibitory interneurons is depicted in Fig. 4.6. As expected, a small-world connectivity results in better energy efficiency compared to an all-to-all network while still delivering uniform global inhibition across the network.

Other mechanisms besides global inhibition are also at play to aid the construction of intensity invariant representations over large dynamic ranges in the brain. For example, the retina consists of photoreceptors with preferential sensitivities to different illumination levels; and many retinal ganglion cells adapt to background light intensities, responding only to temporal changes in intensity levels. The configurability of Golden Gate can be used to mimic these mechanisms in addition to the global inhibition described above.

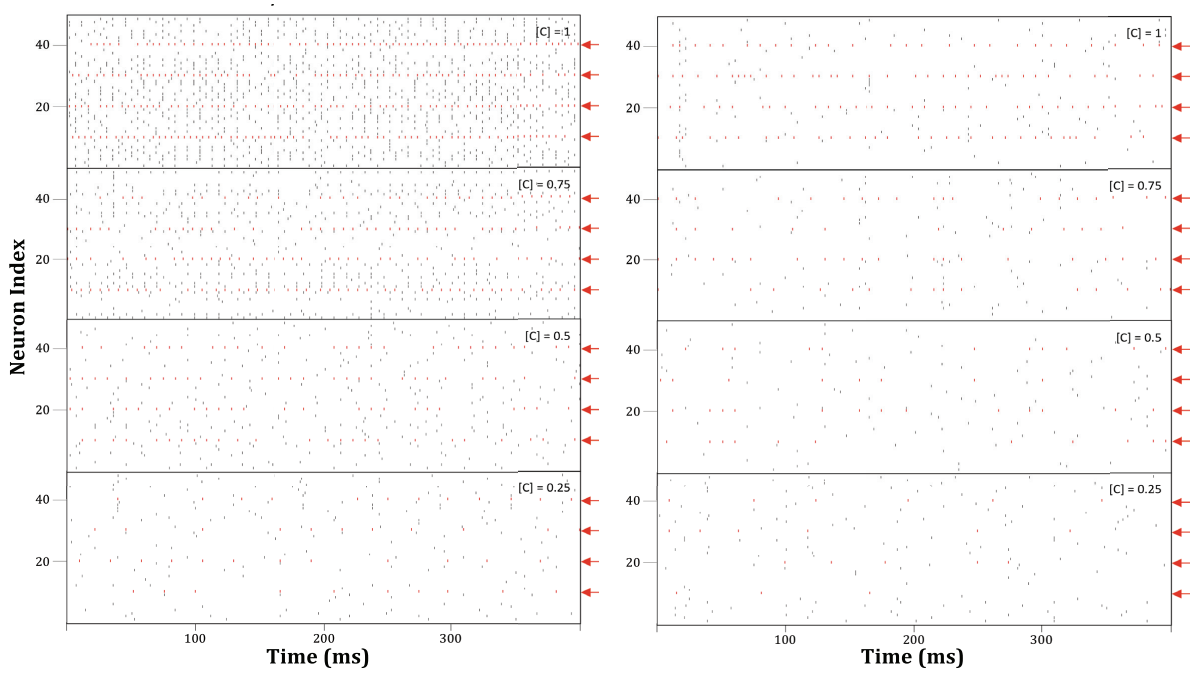


Figure 4.4: Principle neuron spiking activity with and without the effects of global inhibition. With the inhibitory network off, principal neuron population activity increases monotonically in proportion to input intensity (concentration, [C]). The spikes of those neurons that are most sharply tuned to the presented stimulus are depicted in red and indicated by red arrows; additional, weakly-activated neurons also are recruited into the active ensemble as intensity increases. With the inhibitory network on, aggregate principal neuron population activity remains largely independent of intensity. Stimulus identity is represented by the activity profile of the most sharply tuned principal neurons since their relative activities remain essentially stable when global inhibition is delivered.

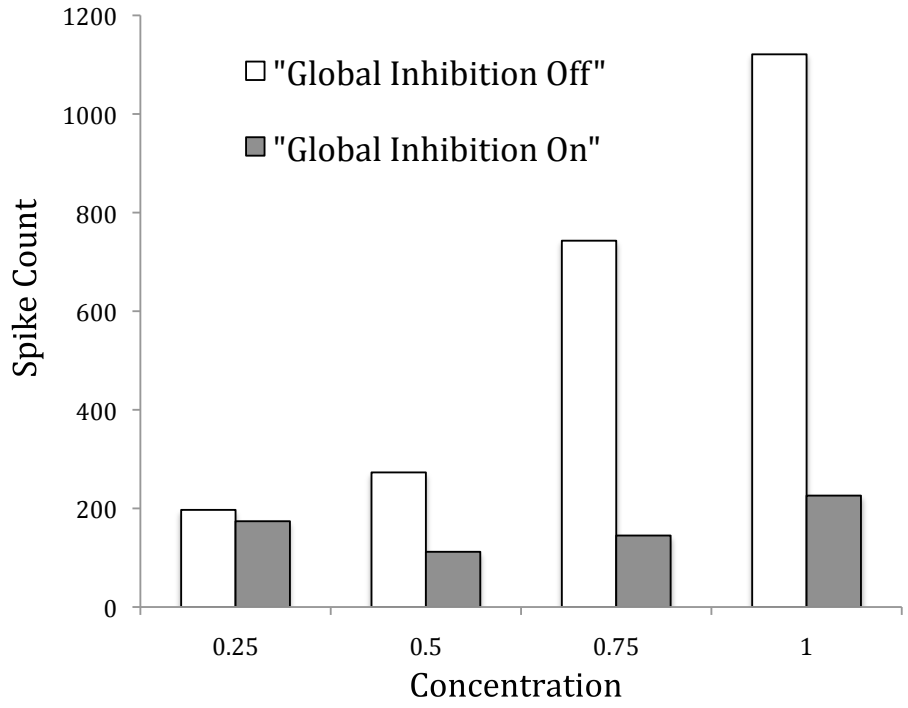


Figure 4.5: Quantification of the inhibitory effects in Fig. 4.4. Spiking activity of neurons that are moderately or poorly tuned to the stimulus (i.e., the count of all black marks in the raster of Fig. 4.4, excluding the spikes from sharply-tuned neurons as denoted in red) is plotted against stimulus concentration (intensity). With global inhibition, the spike count is approximately equal across concentrations, consistent with unchanged relative population activity.

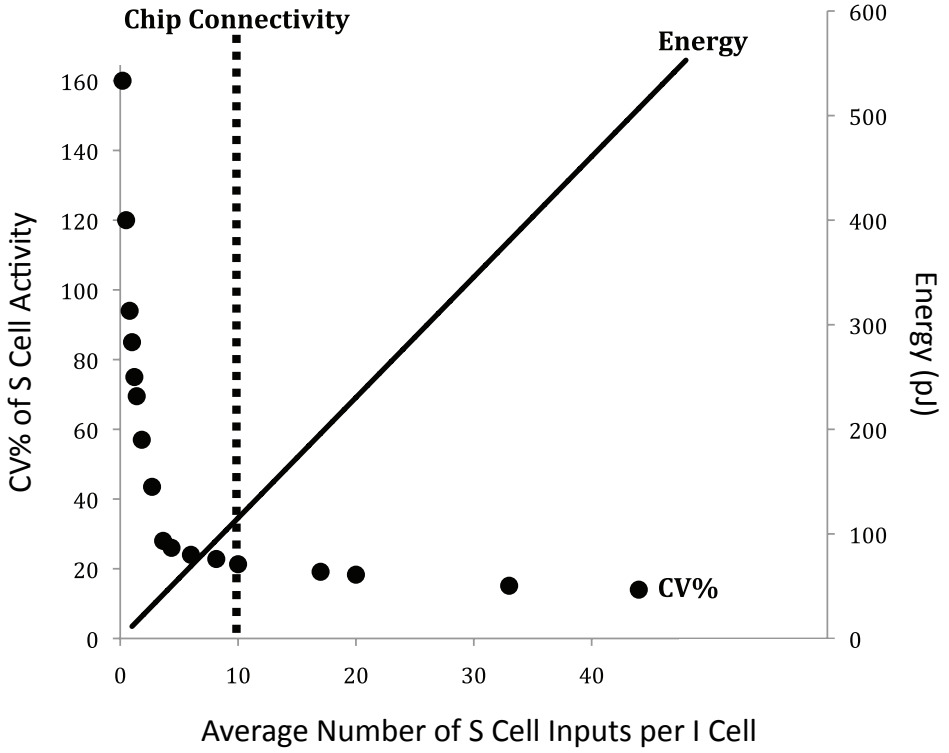


Figure 4.6: A sparsely connected inhibitory network on Golden Gate is functionally equivalent to a fully connected network, but delivers global inhibition at a reduced energy cost. The horizontal axis is a measure of the density of interconnections among the inhibitory network, bounded by the extremes of a fully isolated network at x = 0 and an all-to-all connected network at x = 48. The primary vertical axis (left) denotes the coefficient of variation (CV%– the standard deviation as a percentage of the mean) of inhibitory neuron activity across the population. Low CVs indicate high uniformity in inhibitory activity, i.e. a uniform level of inhibition on all principal neurons. The secondary vertical axis (right) denotes the energy consumed in the inhibitory population. Denser interconnectivity requires more energy. The dotted vertical line denotes the density chosen for a near-optimal chip implementation (results in Fig. 4.4 and Fig. 4.5).

4.2 Pattern Decorrelation

Interfering backgrounds from the environment as well as imprecision and noise in the sensory transduction process[3] can induce undesirable correlations between distinct neural representations (an overlap in receptive fields), making it harder to differentiate between them. Reduction of this correlation is another common operation that a wide range of neural circuits in the brain implement. This operation, commonly referred to as pattern decorrelation or pattern seperation, makes neural representations more distinct from one another thereby facilitating subsequent decoding (readout) by downstream circuits.

Decorrelation in the brain is implemented between primary sensory neurons, for example between retinal ganglion cells representing neighboring points in the visual field; as well as between neurons that have higher-order receptive fields, for example between orientation-selective neurons in the primary visual cortex. The computation can operate within neural maps that are precisely organized in space such as the visual retinotopic maps and auditory tonotopic maps; or it can operate within neural populations that do not have a precise topographical organization, such as those in the olfactory system and the hippocampus.

A variety of mechanisms implemented in neural circuits aid the construction of decorrelated representations. One mechanism is the sparse projection from a neural population of N neurons to a neural population of K neurons [28]. Sparse denotes a projection where $K >> N$ and the number of afferent synapses S onto each of the K neurons is small. Because of the increase in the dimensionality of the representation through such a projection, any overlap between representations in the N neuron population will be reduced in the K neuron population by an amount determined by the values of N, K and S. Examples include axonal projections from retinal ganglion cells to pyramidal cells of the primary visual cortex; and from mitral cells of the olfactory bulb to pyramidal cells of the piriform cortex. Arbitrary networks can be set up in Golden Gate via the

[3]In addition to external sources of noise, biological neural circuits also have intrinsic noise related to molecular processes.

freely configurable crossbar, and therefore sparse projections can easily be implemeted. With successive versions of the chip incorporating orders of magnitude more neurons and synapses (Sec. 2.3) sparse projections can be used as a useful mechanism for decorrelating patterns in the Golden Gate architecture.

Another mechanism that aids decorrelation is the inhibition, and subsequent shut down, of neurons that are weakly activated by a particular stimulus. Overlap of weakly-tuned neurons (that are often particularly sensitive to noise) between the representations of different stimuli can be reduced in this way. Examples include the center-surround organization of retinal ganglion cells that decorrelate neighboring neurons [29], and the PGo cell-mediated non-topographical decorrelation of mitral cells in the olfactory bulb [30]. The high degree of configurability in the synaptic parameters of Golden Gate allow such a mechanism to be readily implemented [27]. The degree of decorrelation (dictated by the amount of inhibition) is often controlled by modulatory circuits such as those underlying attention (see Sec. 4.4).

Yet another mechanism used by neural circuits to decorrelate is competitive inhibitory interactions among representations of similar patterns [30, 31, 32]. Here, a group of neurons representing a frequently-encountered pattern inhibits other neurons in a population that are not part of the group but are involved in the representation of other distinct (but possibly similar) patterns. Correlations between distinct groups induced by positive noise[4] can be removed in this way. In this section, such a decorrelation procedure is illustrated in Golden Gate. No spatial organization is assumed in the configured network and the pattern of inhibitory weights is learned from the input statistics of the stimuli. The procedure leads to a powerful engine to decorrelate arbitrary high-dimensional patterns expressed as neural spike trains.

[4]Here, positive noise refers to noise that increases the activation levels of neurons.

4.2.1 Inhibition Via Higher-Order Receptive Fields

A group of 16 principal neurons (PCs) receive a 16-dimensional input vector. Each PC is associated with a group of 100 interneurons (ICs) that form inhibitory connections onto them. The axons of the PCs extend across the entire PC population. Each IC randomly connects to a subset of 3 PC axons and, when excited beyond its threshold voltage, delivers inhibition to its corresponding PC. The ICs thus possess higher-order receptive fields, i.e. they spike in response to combinatorial patterns of activity in the PC population (essentially a logical AND of spikes from three PCs) driven by specific features of the input. The strength of inhibition provided by the ICs onto the PCs is synapse specific, with PCs that are not part of learned ensembles of activity being strongly inhibited (see Network Plasticity below). Groups of neurons involved in representing different stimuli thereby inhibit each other resulting in a reduction in overlap (decorrelation) between stimulus representations in the PC population. This network arrangement is shown is Fig. 4.7.

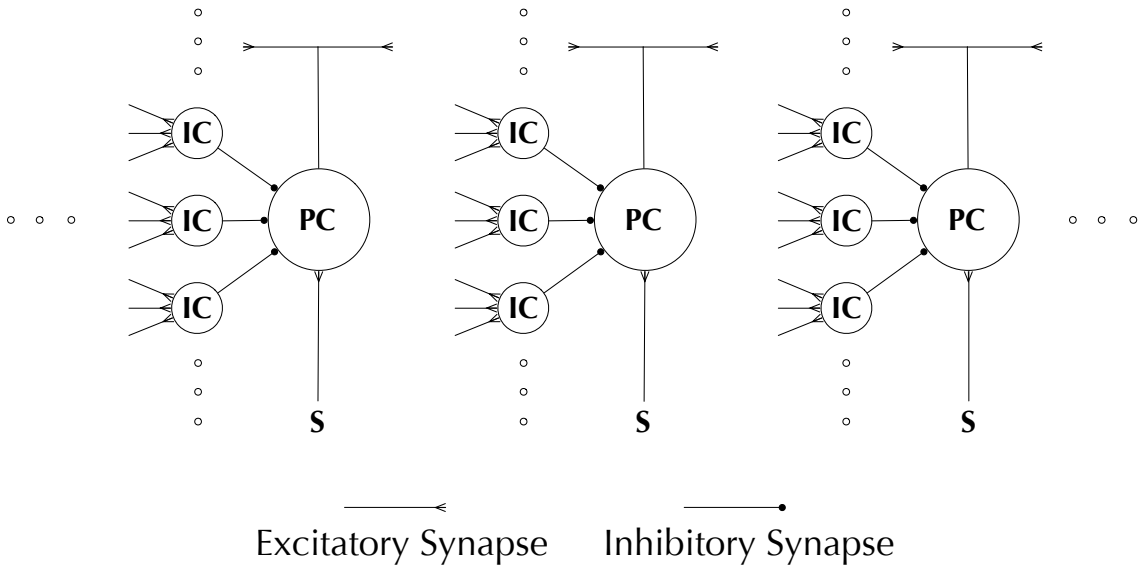


Figure 4.7: Pattern Decorrelation via inhibitory interneurons (ICs) that learn higher-order receptive fields and deliver synapse specific inhibition on principal neurons (PCs)

These kinds of networks are seen in neural circuits that perform decorrelation of high-

dimensional spiking patterns, such as the network formed by mitral cells and granule cells in the olfactory bulb [30] or that formed by CA3 pyramidal cells and DG granule cells in the hippocampus [31]. This network structure can also support decorrelation between neural populations that are far apart in a topographic map. For example, long-range inhibitory interactions can explain some of the extra-classical receptive field properties of neurons in the primary visual cortex [32].

To make efficient use of the crossbar synaptic array in Golden Gate, ICs are shared across PCs as shown in Fig. 4.8. Since the role of ICs is to learn common activity patterns and deliver PC specific inhibition, the modified structure is effective since the IC $\rightarrow$ PC synapses remain individually tunable. By "factoring out" ICs in such a fashion the total number of ICs required to learn a given number of spatiotemporal patterns is reduced.

Recall that each axon in the crossbar synapse memory (see Sec. 2.2) can be one of three distinct "types". A neuron uses the "type" value from the crossbar to assign one of three distinct weights to a synapse. For a PC, one of these weights are excitatory (for the S input), one is weakly inhibitory and one is strongly inhibitory[5]. The drawback of sharing ICs as described in the previous paragraph is that the axon of an IC would be required to have the same "type" value for all the PCs. To overcome this and enable an IC to weakly inhibit some PCs while strongly inhibiting other PCs, two separate neurons (along with their two separate crossbar axons) with the same receptive field is used to represent one IC.

Another restriction that arises due to the sharing of the ICs is that the neuronal parameters and the presynaptic connectivity of an IC is no longer PC specific, but these are inconsequential to the decorrelation computations of this section.

[5]The 8-bit weight of a weak or strong inhibition is determined via learning and is neuron specific.

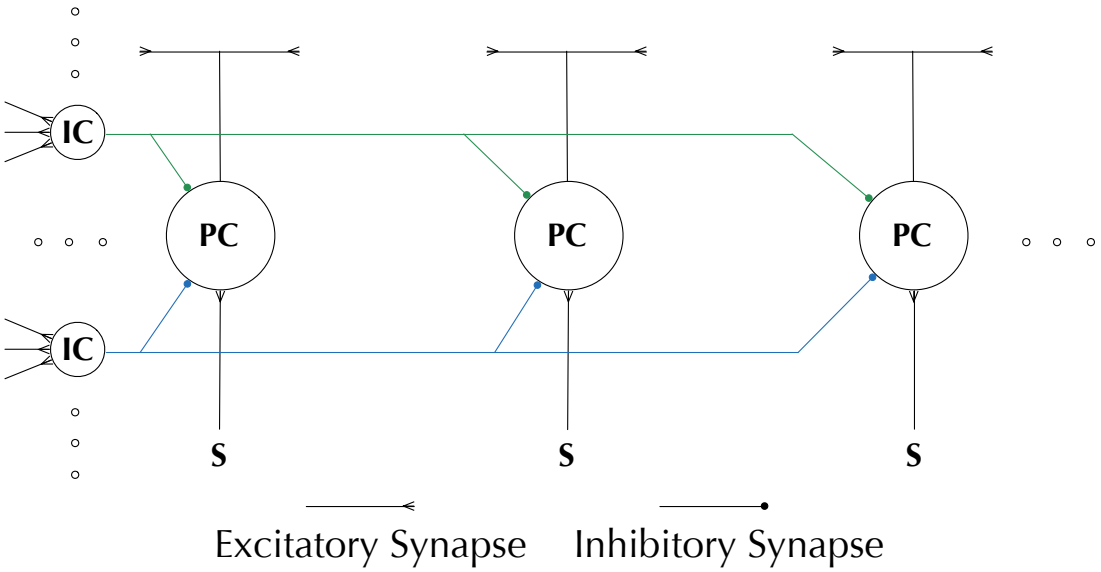


Figure 4.8: Network arrangement for pattern decorrelation in Golden Gate. Parameter values are listed in Table. 4.2

4.2.2 Rhythm Generation and Phase Codes

Stimulus patterns across a neural population is often represented as a latency code with respect to a shared oscillatory potential as described in Sec. 3.2. The PCs in the decorrelation network implemented in Golden Gate use this kind of coding. A common inhibitory signal (driven from outside the chip) represented by a step function (see Fig. 3.3) oscillates at 25 Hz and is delivered to all PCs. The PCs spike during the positive phase of this "gamma" rhythm (in analogy to the gamma frequency local field potentials measured in-vivo) at a time that reflects the strength of input on the PCs, with higher strengths causing earlier spikes. If a PC spikes, its S inputs are set to zero for the rest of the gamma cycle (i.e. until the inhibitory rhythm is asserted) to prevent a PC from spiking multiple times in one cycle.

The input on the PCs are sampled at a rate of 5 Hz – a "theta" rhythm. Thus a given input is sampled by the network 5 times, allowing dynamical interactions in the network across multiple cycles of the inhibitory rhythm. Interactions across multiple cycles is important for several reasons. First, late spiking PCs can influence earlier spiking PCs

only across cycle boundaries. While in the biological system this may be due to IC inhibition lasting across cycle boundaries, in Golden Gate it is implemented via delays in some of the PC $\rightarrow$ IC projections. Second, reciprocal interactions with downstream neural circuits, such as a pattern completer (Sec. 4.3) can evolve over multiple cycles. Third, differential modulation of these circuits, for example by different concentrations of neuromodulators (Sec. 4.4), can act over multiple cycles to extract salient information in the network.

Table. 4.2 lists the parameter values used in Fig. 4.8. Each PC has one of two distinct inhibitory weights whose value is determined through learning (see below) and capped at $-20mV$.

Table 4.2: Parameter Values for Fig. 4.8

Neuron	Threshold (mV)	Leak (mV)	W_{exc} (mV)	$W_{inh\_max}(mV)$
PC	30	1	10	-20
IC	31	2	15	–

Fig. 4.9 shows PC population activity when Golden Gate is configured with the structure of Fig. 4.8 and the dynamics described above. The figure shows the 40 Hz gamma rhythm embedded into the 5 Hz theta rhythm.

4.2.3 Learning Inhibitory Weights

The IC-PC inhibitory synapses are learned via a synaptic plasticity rule in a software model of the network. The learned weights are then downloaded onto Golden Gate, and the one-to-one hardware-software correspondence feature of the chip guarantees desired operation in the hardware circuits. A drawback here is that whereas biological neural circuits continuously learn and adapt to the statistics of their inputs, Golden Gate requires fixed weights and therefore reconfiguration of the chip if changes in input statistics are expected. Synaptic plasticity for online learning will be incorporated in future versions

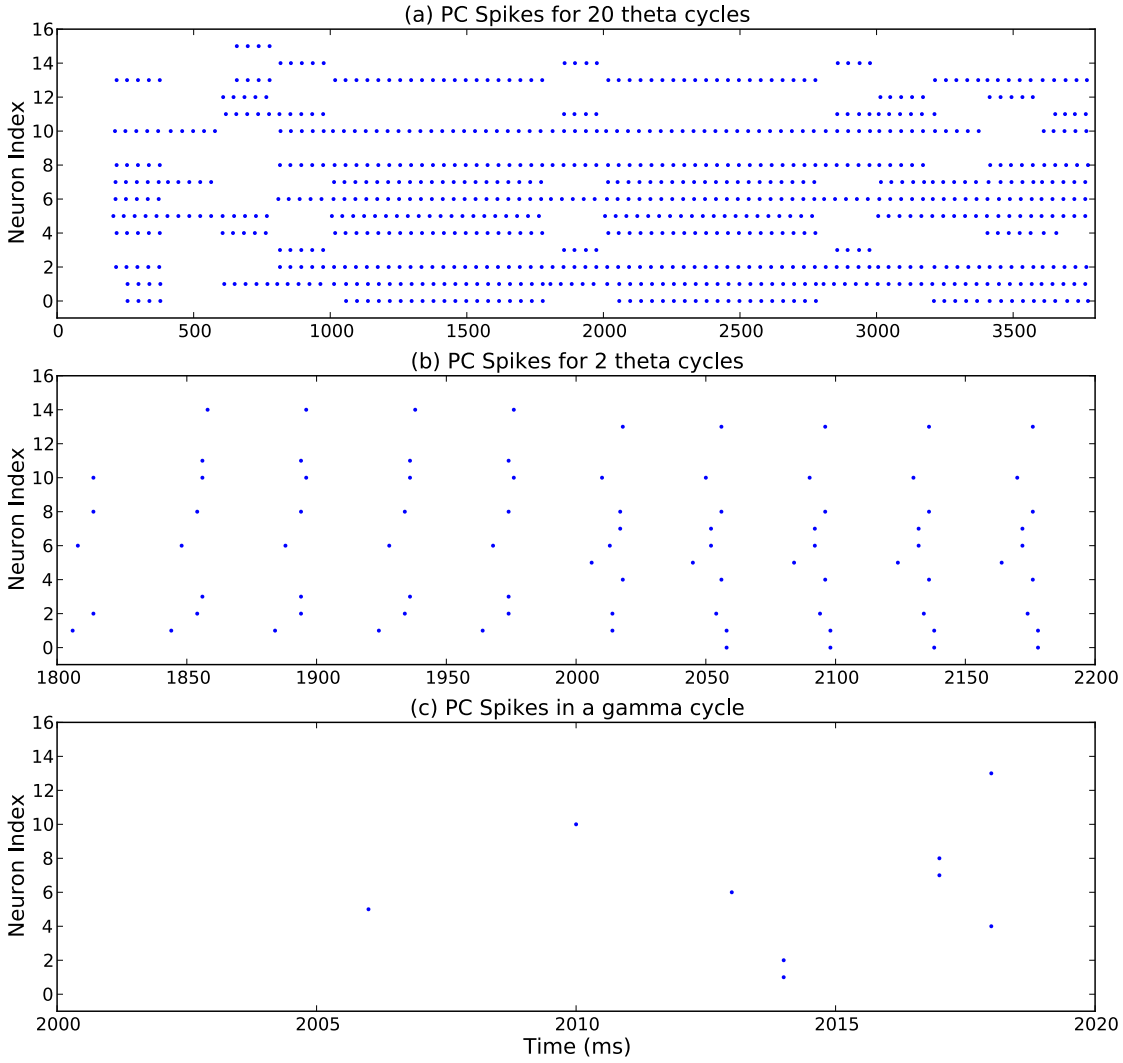


Figure 4.9: Raster plot showing PC activity when the network in Fig. 4.8 is configured in Golden Gate. (a) 20 periods of the 5 Hz theta cycle. The network takes a new sample of the input data at the start of each theta cycle. (b) Part a expanded between 1800 ms and 2200 ms showing 2 theta cycles with different data samples. Periodic inhibition break each theta cycle into 5 gamma cycles. (c) Part b expanded between 2000 ms and 2020 ms showing one half period of a gamma cycle. The times of the spikes relative to the start of the cycle at 2000 ms (a latency code) represent the stimulus vector.

of the chip (see [33] for a prototype).

The learning rule to update the IC→PC inhibitory synapse is expressed in Eq. 4.1. When an IC is excited beyond threshold it grows inhibitory weights on all PCs that are not part of the active ensemble in that gamma cycle. This rule ensures that ICs activated by a unique stimulus identity only grows inhibition on those PCs that are not consistently involved in the representation of that identity. The value of δ used was $0.2mV$. The weights were bound between $0mV$ and $-20mV$. Before downloading to the chip, the weights of the IC inputs to a PC were quantized to three values between $0mV$ (corresponding to a 0 in the state of the synapse in the crossbar) and the most negative learned weight.

$$w = \begin{cases} w + \delta & \text{if PC has a spike within current gamma cycle} \\ w - \delta & \text{otherwise} \end{cases} \tag{4.1}$$

The result of the network structure of Fig. 4.8 and the learning rule of Eq. 4.1 is that a statistically dominant pattern of activity in the PC population recruit unique ensembles of ICs that inhibit all PCs not part of the pattern, thereby reducing any representational overlap with a distinct pattern that may be caused by noise from sensory transduction and/or interfering backgrounds. Activity patterns representing different stimuli are therefore decorrelated from one another.

Plasticity in the PC→IC synapse, although not implemented here, can also benefit the decorrelation process. As described above, an IC is driven by three PCs chosen at random. If the three PCs chosen are not combinatorially recruited for any one of the frequent inputs that the network encounters then the IC will not contribute to the decorrelation process. The chances of productively using an IC can be improved by increasing the number of PC inputs to an IC and pruning those synapses via Hebbian plasticity mechanisms to *learn* the higher order receptive fields [34]. Furthermore, if none of the PC synapses on an IC show significant growth after some period of time (i.e. if no subset of the initial PC group connected to the IC show consistent activity) the input of

the ICs can be rewired to sample a different combination of PCs, reflecting the process of adult neurogenesis in neural circuits of the olfactory bulb and the hippocampus [35].

4.2.4 Evaluation

Fig. 4.10 illustrates decorrelation performed by the network of 16 PCs and 160 ICs described above once configured in Golden Gate. When a learned pattern is corrupted by noise, ICs inhibit out the PCs that are not part of the learned pattern, thereby decorrelating its representation from other possible patterns.

To further test the network, 16-dimensional vectors, representing the activity of 16 sensors with 8 distinct activation levels, were constructed from a normal distribution (truncated at 0 and 8) with mean 1 and standard deviation 5. 10 such vectors, labelled A through J were presented to the network. At each presentation, a vector drove the PCs for a full theta cycle and the population activity of PCs were recorded. This was termed the *labeling period*. A small amount of noise (see Fig. 4.11) was added to each vector before presenting it to the network during the labeling period. Next, the vectors were presented to a software model of the network for a period of time, termed the *learning period*, during which the IC $\rightarrow$ PC inhibitory weights were learned according to Eq. 4.1. During the learning period, one of the 10 vectors were presented to the network at random with a probability distribution shown in Table. 4.3. The learned weights were downloaded onto Golden Gate, and the vectors were again presented to the chip but this time with random Gaussian noise added to each sensor activation level for each of the vectors. The noise distribution was truncated at 0 and 8, and had mean 0 and varying standard deviations described below. The resulting pattern of PC activity was again recorded from the hardware, and a nearest-neighbor classifier (operating outside the chip in a software environment) classified the pattern according to its euclidean distances to the patterns in the labeling period. This period was termed the *noise period*.

The protocol for labeling, learning and testing is illustrated in Fig. 4.11.

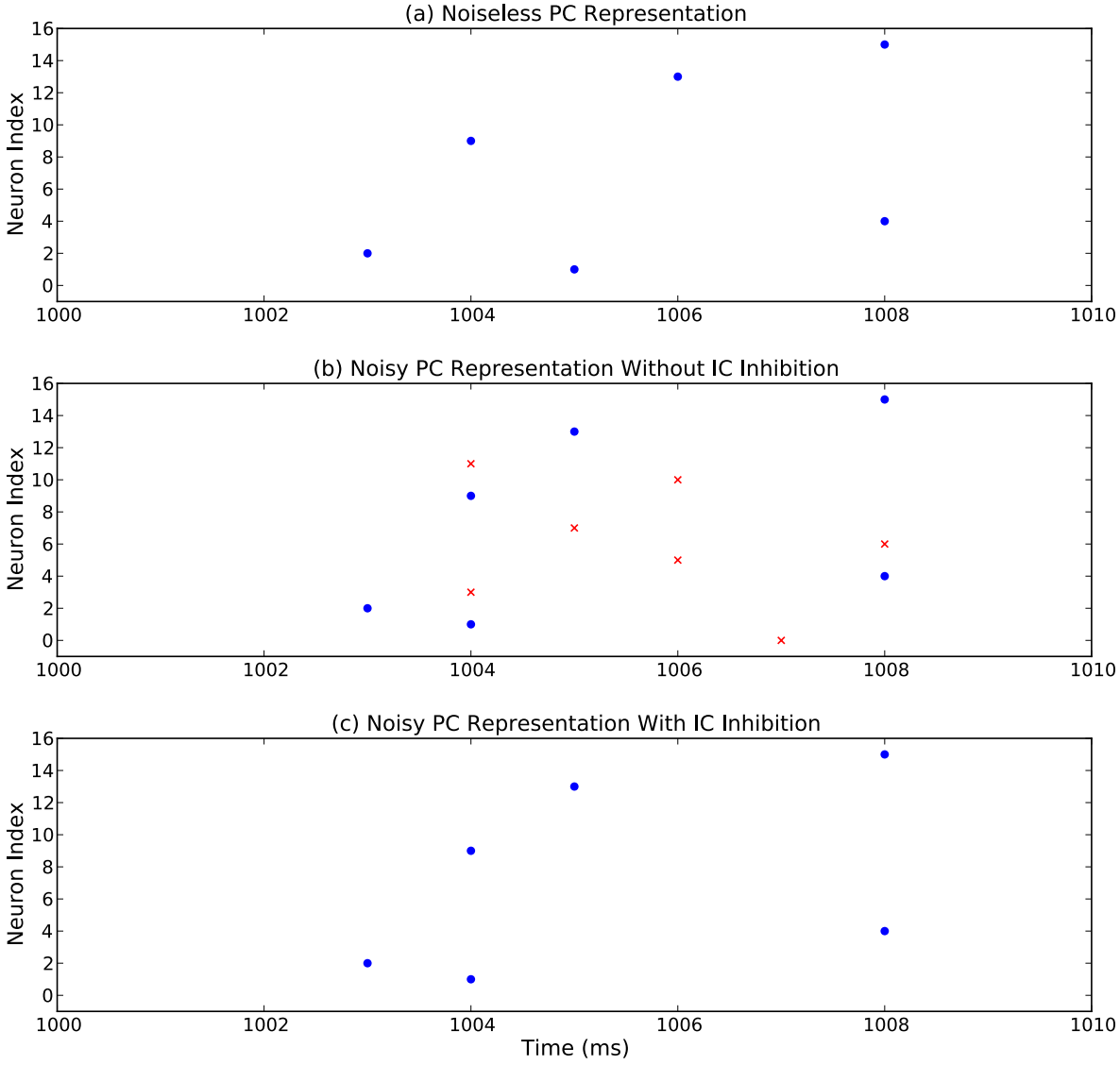


Figure 4.10: Raster plot illustrating decorrelation of PC activity.

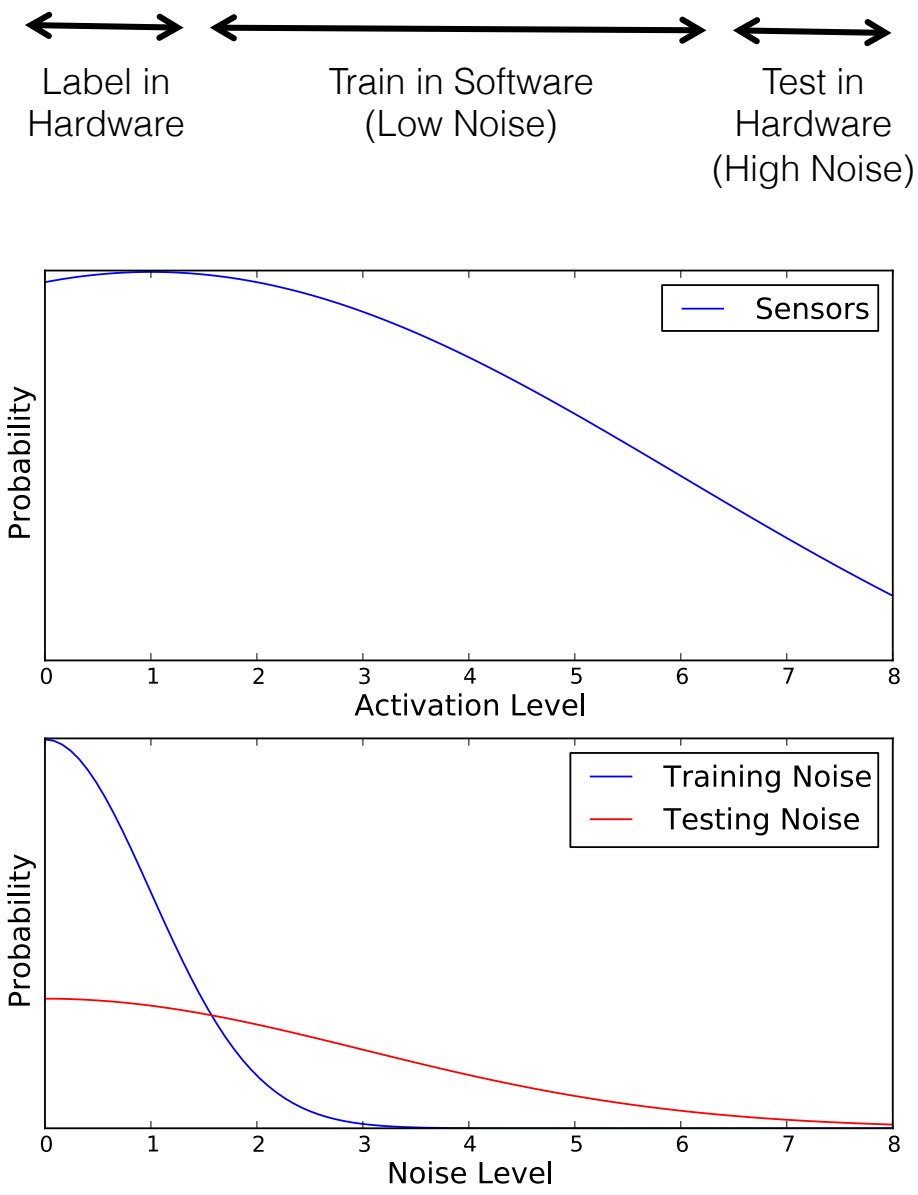


Figure 4.11: Labeling, training and testing protocols of the 16-dimensional sensor vectors. The sensor activation levels and additive noise on each sensor in the training and testing periods were drawn from a truncated Gaussian distribution as shown.

Table 4.3: Probability distribution of data labels during the learning period

Data Label	A	B	C–J
Probability	60 %	30 %	10 %

The inhibitory network shuts down noisy activity to extract learned patterns in the data. In a Euclidean space spanned by PC activity levels, noisy data points will come closer to the learned data after inhibition of the PCs by the ICs. To quantify this effect, a self-similarity index was computed for each of the labelled data

$$Self\ Similarity = \frac{1}{1 + ED(label, noise)}\ , \tag{4.2}$$

where $ED(label, noise)$ is the Euclidean distance between a vector in the labeling period and the same vector in the noise period. This index is a measure of how close a vector in the noise period resembles the same vector in the labeling period. An index of 1 represents perfect resemblance.

For data label A (see Table. 4.3), Fig. 4.12 shows the effects on the self-similarity index as the amount of noise in the noise period is increased. Here, the learning period is fixed at 2000 ms (10 theta cycles). For increasing standard deviation of the zero-mean additive noise, the self-similarity index computed from the PC population activity decreases as expected. However, due to the effects of the inhibitory network, this index always remains higher than the self-similarity index computed directly from the sensor array. As a result, the performance of the nearest-neighbor classifier is more robust using the PC patterns compared to using the raw sensors patterns (Fig. 4.13).

Fig. 4.14 and Fig. 4.15 show how the network performs for different lengths of the learning period for a standard deviation of noise set at 5. The most frequently-encountered data (label A) shows the most marked increases in self-similarity and thus can be classified most robustly. Data that are not frequently observed during the learning period do not result in corresponding IC→PC inhibition growth and thus are often misclassified. The

number of misclassifications for these less frequently-observed data may actually grow as the learning period increases as the network biases towards detecting the frequently-observed data labels.

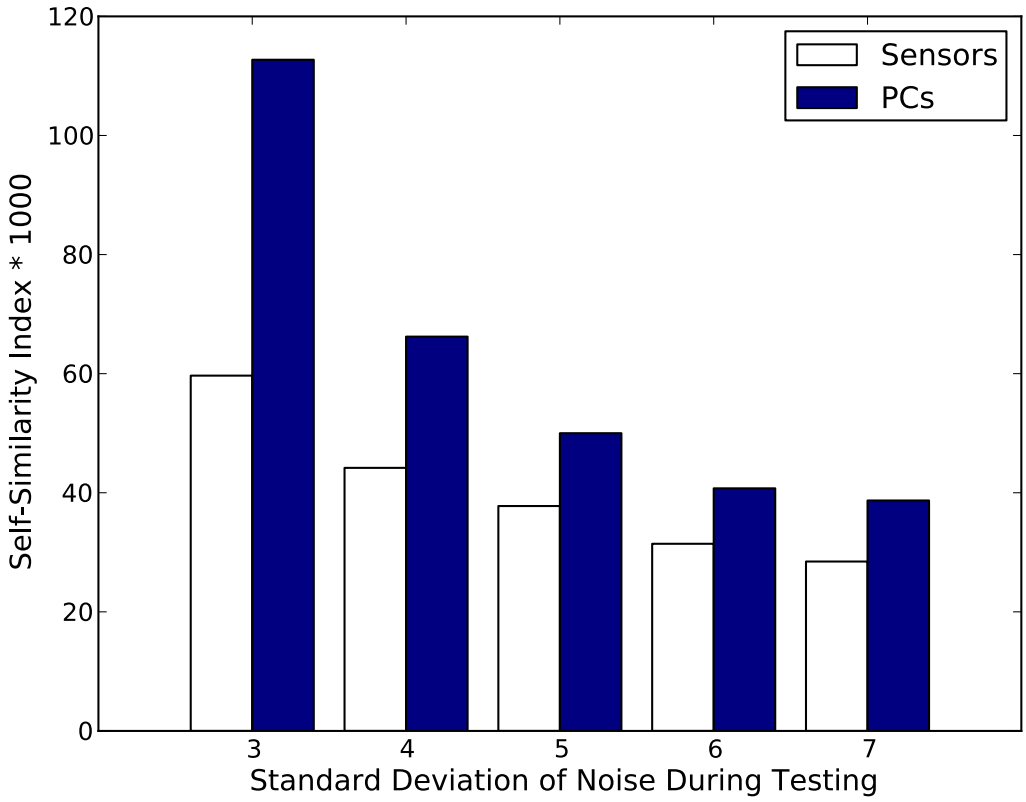


Figure 4.12: Self-Similarity index for different noise levels for data label A (Table. 4.3). As the standard deviation of the noise increases, the data becomes more dissimilar to itself. However, as a result of the decorrelation performed by the network of PCs and ICs, the self-similarity index computed from the PC population is always higher than that computed from the sensors directly. The representation created by the PCs are therefore more tolerant to noise than the sensor representation increasing the robustness of a nearest-neighbor classifier that decodes the patterns (Fig. 4.13).

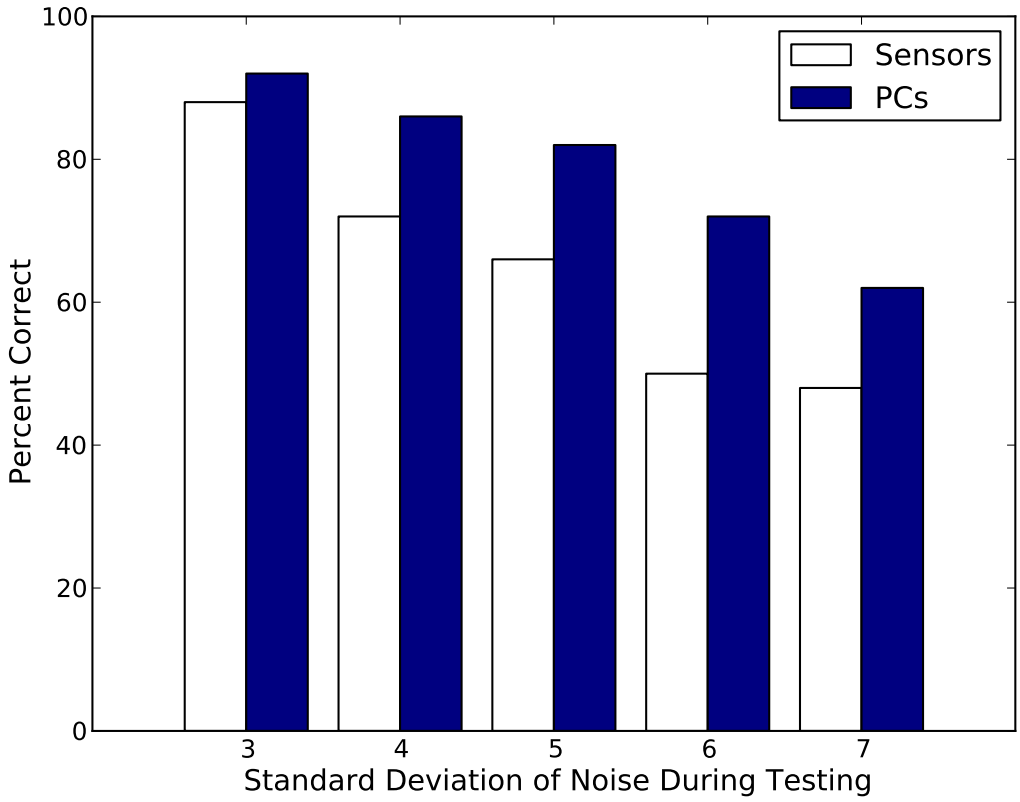


Figure 4.13: Classification performance of the nearest-neighbor classifier for different noise levels for data label A (Table. 4.3). The increase in the self-similarity index (Fig. 4.12) as a result of decorrelation performed by the network of PCs and ICs creates greater noise tolerance in the classification of the PC patterns compared to the classification of the sensor patterns.

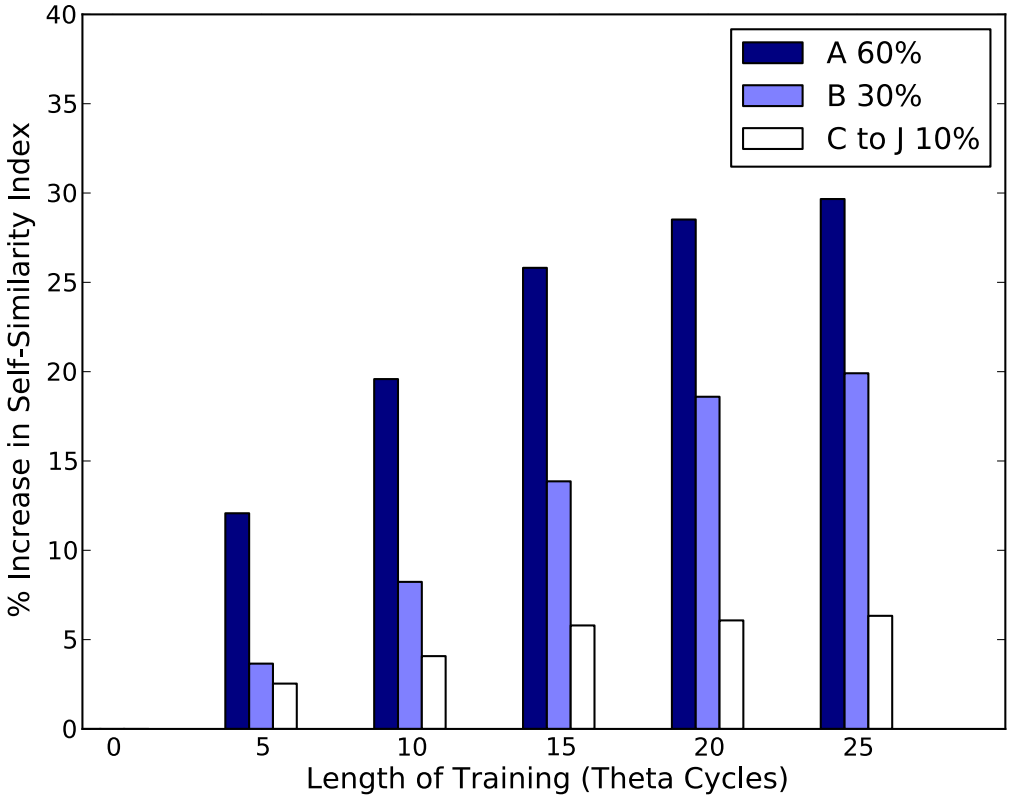


Figure 4.14: Increase in self-similarity index for different learning periods. Longer learning periods bring the noisy data closer to the labeled data and thus increases the self-similarity index. The increase is higher for more frequently-encountered data. Increased self-similarity leads to better classification performance (Fig. 4.15).

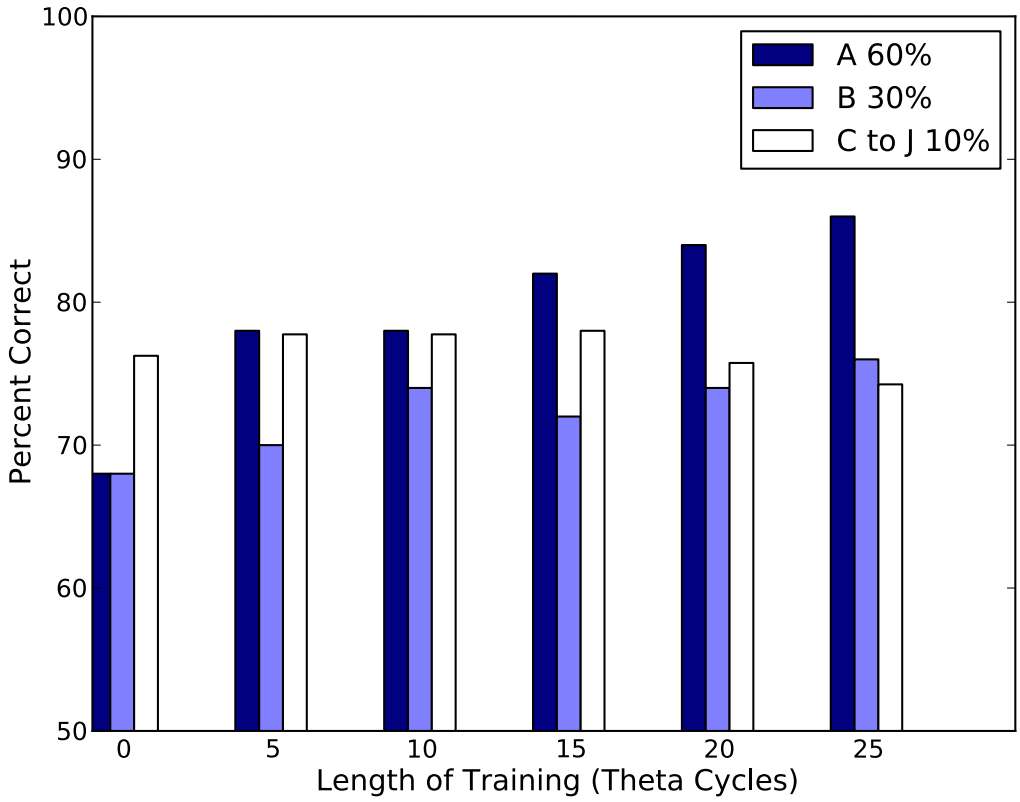


Figure 4.15: Classification performance of the nearest-neighbor classifier for different learning periods. The increase in the self-similarity index (Fig. 4.14) leads to an increase in the classification performance. The increase is higher for longer learning periods and more frequently-encountered data.

4.3 Associative Recall

Associative recall, or *pattern completion*, in neural circuits work in tandem with pattern decorrelation (Section. 4.2) to recover common spiking patterns that are degraded by noise (e.g. from interfering signals). Whereas pattern decorrelation seperates overlapping representations, pattern completion recovers representations that are partially present by matching degraded population activity to previously encountered spiking patterns. This process promotes perceptual stability, allowing the system to maintain previously acquired associations and meaning despite stimulus fluctuations induced by noise.

Associative recall has been most thoroughly studied in the hippocampus, where recurrent connectivity within subregion CA3 "fill in" incomplete afferent patterns based on previously encountered input [36]. By doing so, neural circuits in this region maintain stability of complex memories (such as the spatial map of the animal's environment or the details of episodic memories) in the face of variations. Pattern completion is also critical in maintaining stable percepts in the olfactory system since most natural odors are rarely encountered with exactly the same components in the same proportions. Recurrent neural circuits in the piriform (olfactory) cortex have the capacity to associate some molecular features (represented by afferent spiking activity) with others, thereby recognizing degraded odorant signals among complex mixtures [37].

In this section, a recurrent network that performs associative recall of learned patterns is illustrated in Golden Gate. Excitatory synapses in the recurrent network group neurons together thereby completing patterns that have been degraded by negative noise[6]. The basic association network presented here can be further enhanced by adding inhibitory interneurons that modulate network activity, for example by sparsifying the number of neurons participating in the storage of each pattern to reduce memory interference.

Patterns of interest from complex mixtures of activity from sensor arrays feeding into Golden Gate can be extracted by passing them through the decorrelator of Section. 4.2

[6]Here, negative noise refers to noise that reduces the activation levels of neurons.

followed by the associator described in this section. Such a structure can counter both positive noise (where pattern decorrelation is required) and negative noise (where pattern association is required), providing a powerful substrate for high-dimensional signal processing.

4.3.1 Recurrent Associative Connections

A group of 16 association neurons (ANs), each driven by 4 values from a 16-dimensional stimulus vector (S) was configured in Golden Gate. The ANs have recurrent excitatory connections among them as illustrated in Fig. 4.16. The connectivity profile among the ANs are learned via a spike-time-dependent plasticity rule as described below. This structure resembles the recurrent connections among pyramidal cells of the piriform cortex and of the CA3 subregion of the hippocampus.

Similar to what was done for the ICs in the decorrelation network (Sec. 4.2.1), each AN is represented by two neurons driven by identical S values. This allows each AN to have two distinct "types" of axons in the crossbar. Thus an AN could strongly excite some ANs while weakly exciting other ANs[7].

4.3.2 Rhythm Generation and Phase Codes

The stimulus vector S driving the ANs in the network of Fig. 4.16 have a latency code (see Sec. 3.2) under a common "gamma" rhythm oscillating at 25 Hz. The stimulus identity is encoded in the spatiotemporal spiking pattern of S inside one gamma cycle. The ANs naturally inherit the gamma-band oscillation of their inputs.

The parameters used for the ANs are shown in Table. 4.4. For an AN to be driven to threshold solely through afferent synapses, 3 out of its 4 S inputs need to spike within a relatively narrow time window. If the presynaptic activity is less than this, the AN will require excitation from the recurrent connections to be driven to threshold. The recur-

[7]The 8-bit weight of a strong or weak excitation is determined via learning and is neuron specific.

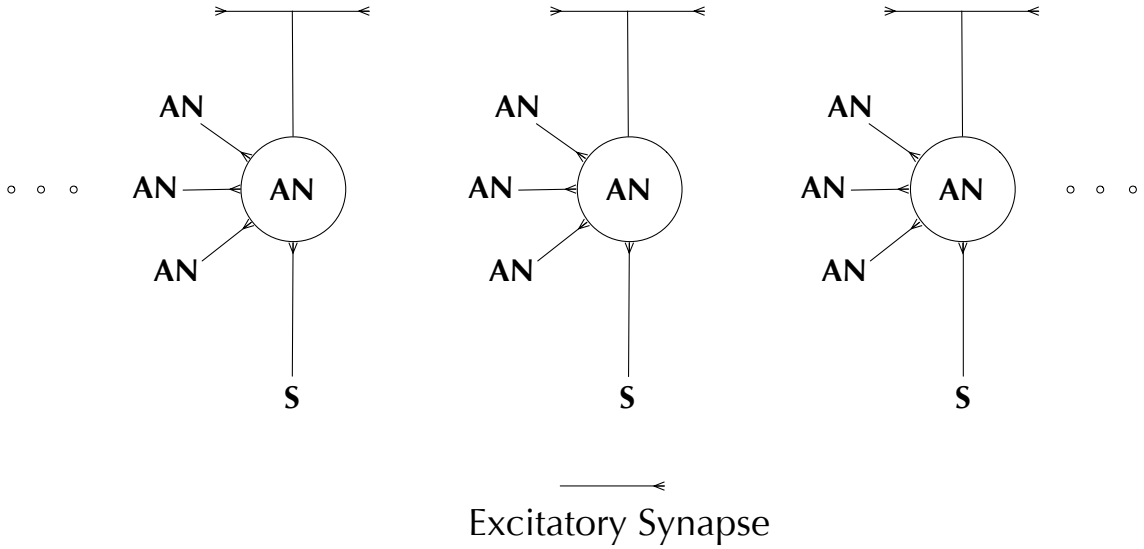


Figure 4.16: Structure of the associative network. Three association neurons (ANs) are shown. Each AN is driven by a stimulus vector S and the axons of other ANs. The strength of excitation is between the ANs is learned through a spike-time-dependent plasticity rule (Fig. 4.17).

rent connections therefore enable the recovery (or completion) of learned AN population activity patterns in the face of incomplete input that has been degraded by noise.

Table 4.4: Parameter Values for the Association Neurons in Fig. 4.16

Threshold (mV)	Leak (mV)	W_S (mV)	$W_{AN\_Max}(mV)$
70	1	30	10

4.3.3 Learning Associations

As was done in Sec. 4.2, a software model of the network was used to learn the weights of the recurrent connections between the ANs. These association weights were initialized to zero and the ANs were driven by a set of stimulus vectors. As learning progressed, these weights grew according to the spike-time-dependent plasticity rule shown in Fig. 4.17. A recurrent synapse developed strong excitation if a postsynaptic spike consistently followed a presynaptic spike on that synapse. The weights were bounded between $0mV$ and $10mV$

(Table. 4.4). After a period of learning, the learned association weights of an AN were quantized to three different values between $0mV$ and the maximum learned weight and configured on the chip. As explained earlier each AN (represented by two different neurons on the chip) has two associated axons in the crossbar representing two distinct excitatory "types". An AN weighs another AN's spike by a weight of 0 (no connection in the crossbar), or one of its two positive association weights.

The result of this learning procedure is that excitatory connections develop among ANs enabling them to perform associative recall when negative noise degrades the S inputs.

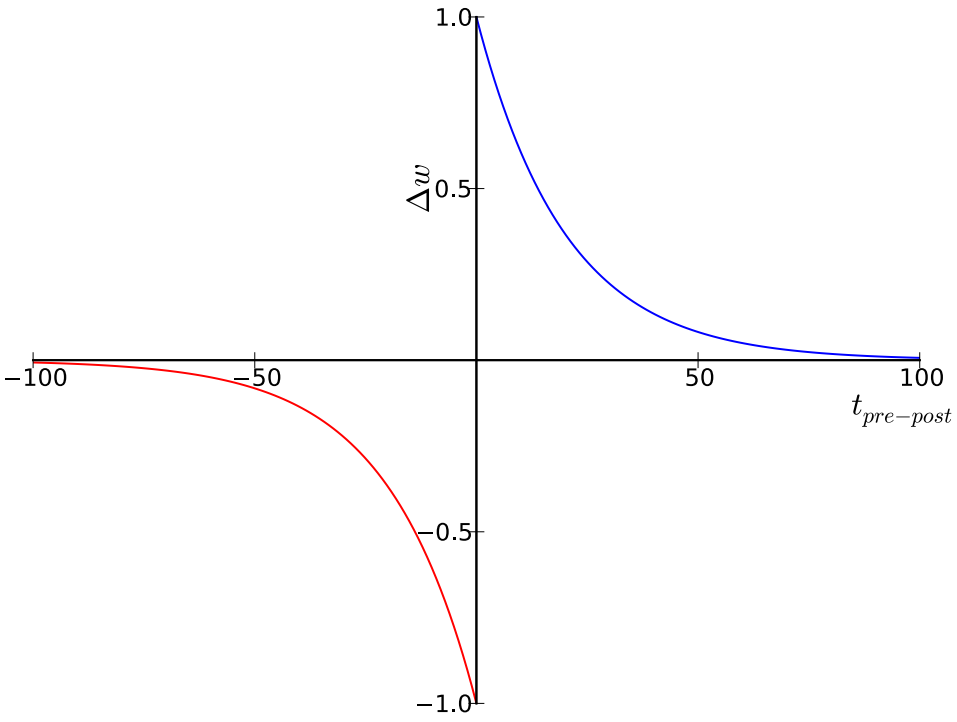


Figure 4.17: The learning rule used to modify the weights of the associative connections between the ANs. The x-axis is the time difference between the arrival of a pre-synaptic spike at a synapse of an AN and the generation of a postsynaptic spike in that AN. The y axis is the resulting change in the weight of that synapse.

4.3.4 Evaluation

Fig. 4.19 illustrates associative recall performed by the recurrent network of ANs described above once it is configured in Golden Gate. When a learned pattern is corrupted by negative noise, the recurrent excitatory connections among the PCs delivers extra excitation to the suppressed PCs to complete the degraded pattern.

To further test the network, 16-dimensional stimulus (S) vectors, representing the activity of 16 sensors with 8 distinct activation levels were constructed from a uniform distribution. NP (initially set to 5) such vectors were presented to the network. At each presentation, a vector drove the ANs for a gamma cycle and the population activity of the ANs were recorded. This was termed the *labeling period*. A small amount of noise (see Fig. 4.19) was subtracted from each vector before presenting it to the network during the labeling period. Next, the vectors were presented to a software model of the network for a period of time, termed the *learning period*, during which the recurrent associative connections among the ANs were learned according to the plasticity rule of Fig. 4.17. During each gamma cycle of the learning period, one of the NP vectors was presented to the network at random with equal probability. After the network was driven by every vector for 15 gamma cycles the learned weights were downloaded onto Golden Gate, and the vectors were again presented to the chip but this time with large random negative Gaussian noise (see Fig. 4.19) added to each sensor activation level for each of the vectors. The noise distribution was truncated at 0 and -8, and had a standard deviation of SD (initially set to 1). The resulting pattern of AN activity was again recorded from the hardware, and a nearest neighbor classifier (operating outside the chip in a software environment) classified the pattern according to its euclidean distances to the each of the patterns in the labeling period. This period was termed the *noise period*. For comparison, the performance of the nearest neighbor classifier was also tested for a network (with the same configuration) which did not undergo the learning process (i.e. all associative connections were zero).

The protocol for labeling, learning and testing is illustrated in Fig. 4.19.

Fig. 4.20 shows the performance of the network in the noise period as the standard deviation of noise is varied from 1 to 5 with the number of different input patterns fixed at 5. For increasing standard deviation of the zero mean negative noise, the performance of the classification drops because the inputs become progressively more corrupted. However, because of pattern completion, the performance of the nearest-neighbor classifier after learning the recurrent associative weights is always better compared to the performance without the learned weights.

Fig. 4.21 shows the performance of the network during the noise period for different numbers of pattern stored with the standard deviation of noise fixed at 4. For increasing number of patterns stored, the performance of the classification drops because of interference caused by the overlap of pattern representations. However, because of pattern completion, the performance of the nearest-neighbor classifier after learning the recurrent associative weights is always better compared to the performance without the learned weights.

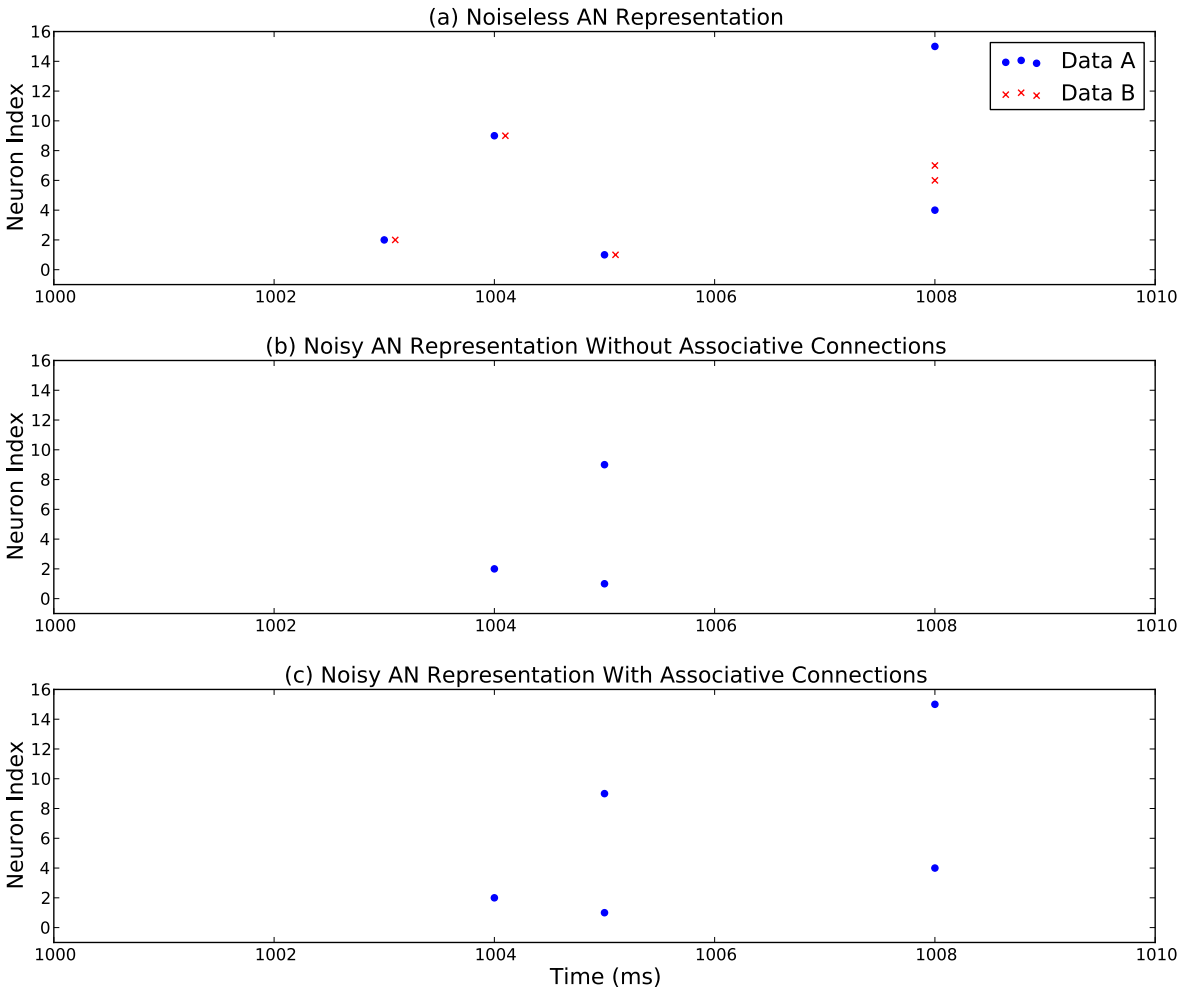


Figure 4.18: Raster plot showing pattern completion via the recurrent associative connection of the ANs. (a) The representations of two similar data labels. (b) The representation of data label A corrupted by noise. Noise reduces afferent excitation on Neuron 4 and Neuron 15 (required to differentiate between labels A and B) and prevents them from reaching their spike thresholds. (c) The associative connections provide extra excitation on Neuron 4 and Neuron 15, causing them to spike and complete the pattern for label A.

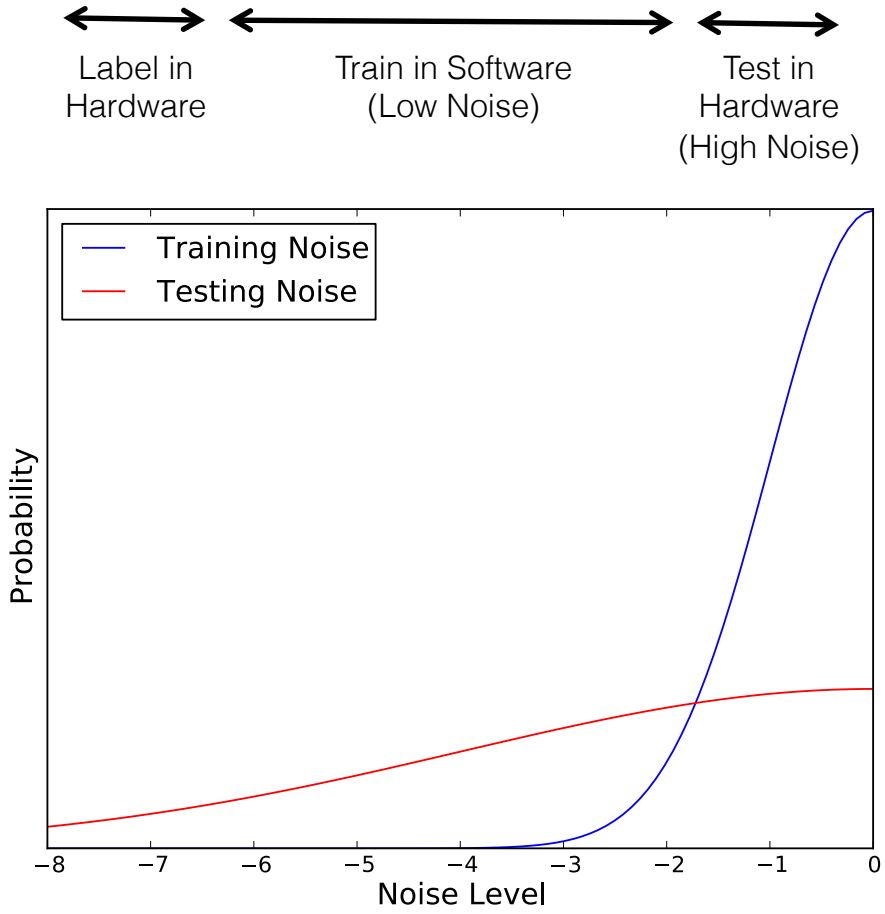


Figure 4.19: Labeling, training and testing protocols of the 16-dimensional sensor vectors. The sensor activation levels are drawn from a uniform distribution (not shown). Negative noise on each sensor in the training and testing periods were drawn from a truncated Gaussian distribution as shown.

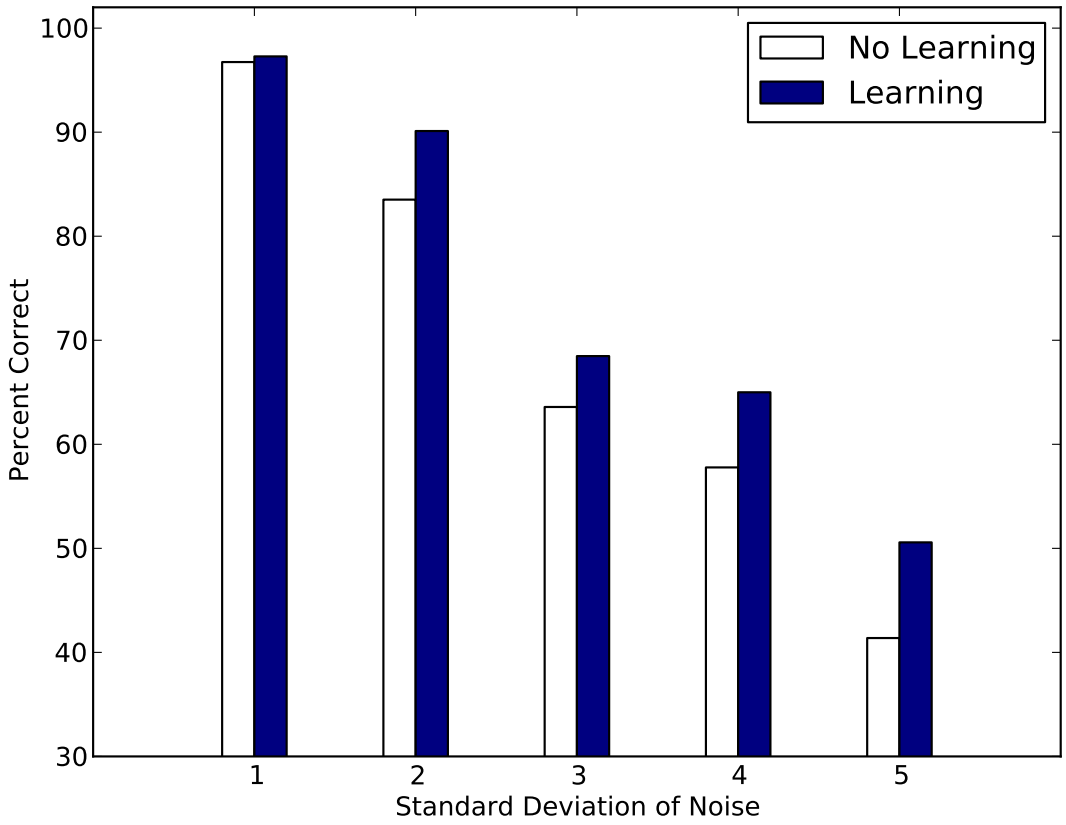


Figure 4.20: Average classification performance (for all data labels) of the nearest-neighbor classifier (number of patterns = 5) for different noise levels. The pattern completion resulting from learned associative connections result in greater noise tolerance.

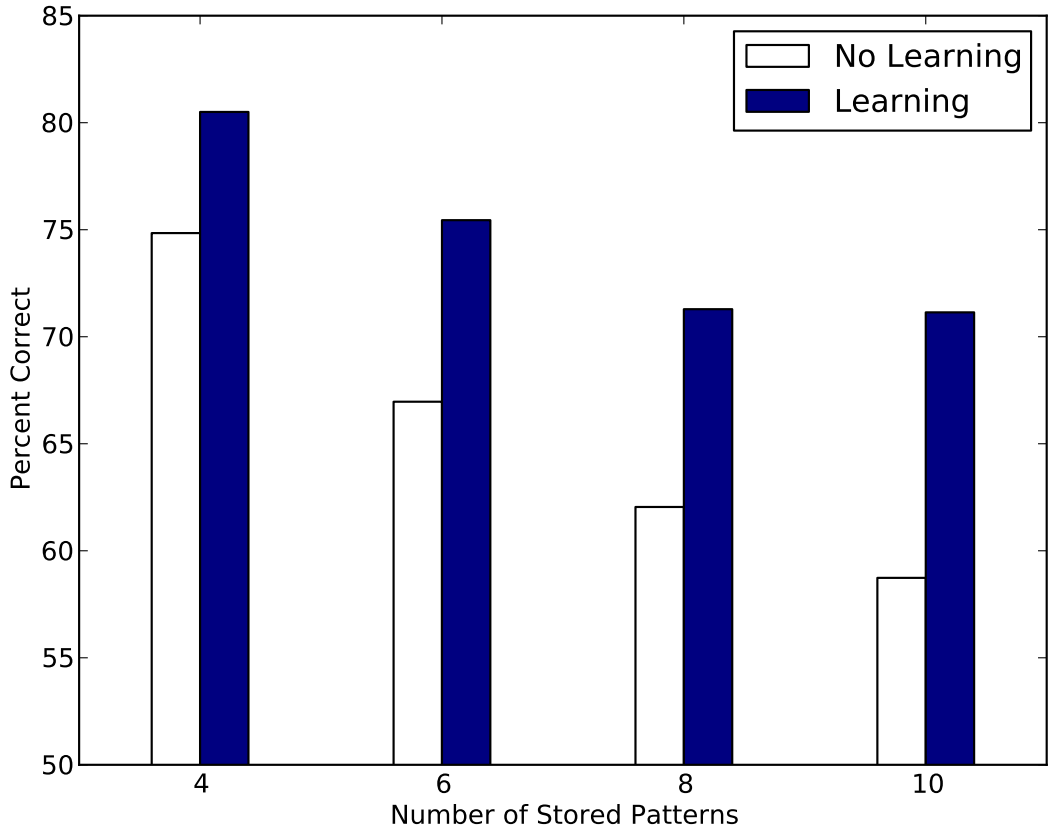


Figure 4.21: Average classification performance (for all data labels) of the nearest-neighbor classifier (standard deviation of noise = 4) for different number of patterns stored in the associative network. The pattern completion resulting from learned associative connections result in greater noise tolerance.

4.4 Attentional Modulation

Neuromodulation refers to the change in the state of a neuron or a population of neurons that subsequently alters its response properties [38]. The release of neurotransmitters associated with the modulation process in biological neural circuits can be either through individual synapses or through diffused transmitter release at the vicinity of the target population. A number of neuromodulatory systems are used in the brain such as the dopaminergic system that modulates reward-driven behavior and the acetylcholine system that modulates attention and memory.

4.4.1 Acetylcholine and Attention

Attention is the process of selectively enhancing the effects of certain stimuli at the expense of others in order to meet task demands and to resolve ambiguities. The acetylcholine (ACh) neuromodulatory system has been shown to be critically involved in attentional processes across different sensory and memory systems of the brain [38]. Lesions of ACh inputs have been shown to impair performance in tasks related to sustained attention and memory whereas infusions of ACh have been shown to have the opposite affect in the same tasks.

At the cellular and circuit levels, ACh release increases the firing rates of some neurons while decreasing that of others, thereby selectively enhancing the effects of certain stimuli. This affect can be used for several processes required for attention such as the potentiation of feedforward environmental stimuli along with the suppression of feedback cortical projections [39], the sharpening of neuronal tuning curves [40, 41], or the selective gating of top-down biases [42].

This section describes the implementation of a network in Golden Gate that mimics ACh-driven neuromodulation in the brain. Specifically, the sharpening of neuronal receptive fields in response to a neurmodulatory input, similar to ACh effects in the primary visual cortex [40] and the olfactory bulb [41], is demonstrated.

4.4.2 Saliency Enhancement

The configured network, illustrated in Fig. 4.22, consists of a group of 18 principal cells (PCs) each associated with an inhibitory cell (IC) and a modulatory cell (MC). Each PC and IC are driven by one value of a stimulus vector (S), represented by a rate code. Each IC inhibits its corresponding MC. Inputs simulating the activation of neuromodulators (ACh), representing by a rate code, provides a uniform level of excitation to all PCs and all MCs. The parameters of the cells, given in Table .4.5, are set such that when the ACh input is high, PCs that have low levels of S excitation will lower their activity while PCs that have high levels of S excitation will increase their activity. This is because for low S_i, IC_i is not strongly driven and as a result MC_i strongly inhibits the weakly activated PC_i. In contrast for high S_i, IC_i strongly inhibits MC_i, thereby reducing its inhibition on the strongly activated PC_i.

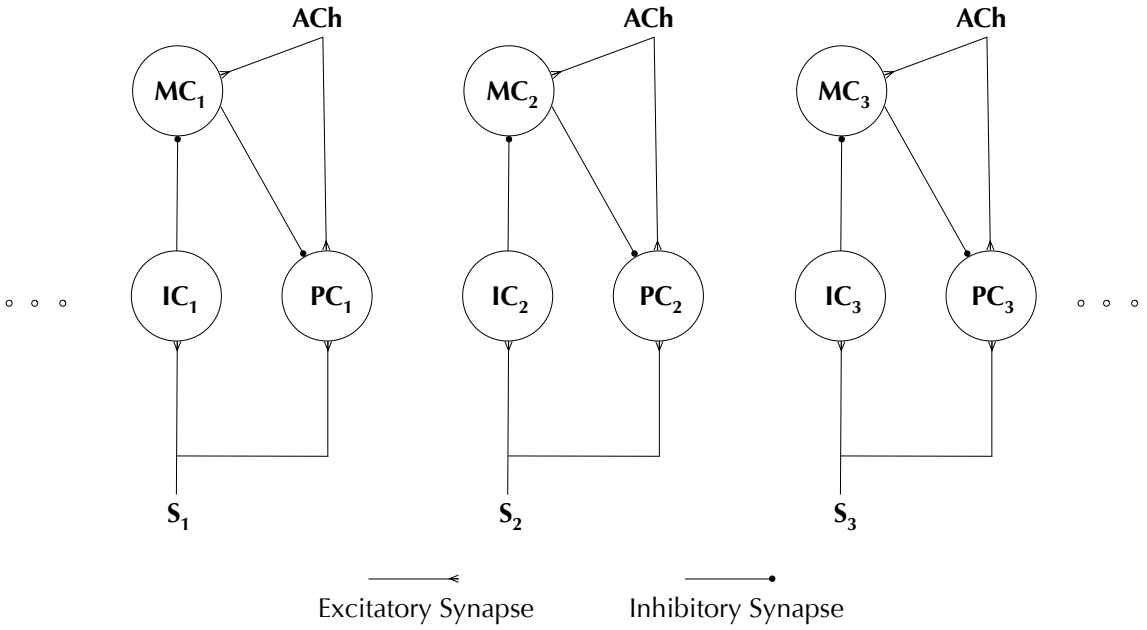


Figure 4.22: Structure of the network showing the principal cells (PCs), the inhibitory cells (ICs), and the modulatory cells (MCs). The network is driven by a stimulus vector (S) and by inputs simulating acetylcholine (ACh) activation. Each input S_i is represented by 10 axons with the same spike rate. The ACh input, also represented by 10 axons with the same spike rate, is common to all PCs and MCs.

Table 4.5: Parameter Values for Fig. 4.22

Neuron	Threshold (mV)	Leak (mV)	W_{exc_S} (mV)	W_{exc_Ach} (mV)	$W_{inh}(mV)$
PC	40	5	8	5	-40
IC	40	1	8	–	–
MC	80	0	–	40	-200

The spike rates recorded from the configured network in Golden Gate is shown in Fig. 4.23. Activation of the ACh inputs reduce the activity of weakly-tuned PCs and increase the activity of strongly-tuned PCs in the population, thereby sharpening the population tuning curve. The result of this "attentional modulation" is that the representation of the salient features of the stimulus vector, corresponding for example to the strongest signal from the sensory environment or the strongest internally-generated memory pattern, are enhanced while the representation of the other features, corresponding for example to background environment or interfering memory patterns, are suppressed. The corresponding increase in the signal-to-noise ratio of the salient features is an important property for a pattern recognition system required to detect signals in the face of uncertainty, interference and noise.

The level of attention is represented by the activation level of the ACh inputs. As ACh activation rises the population tuning curves become sharper. A raster plot of the network as the ACh level is swept from $0.05kHz$ to $0.2kHz$ at $100ms$ intervals is shown in Fig. 4.24. A pattern recognition system using this network can modulate the ACh input depending on what activity levels need to be included/discarded to resolve ambiguities in the recognition process.

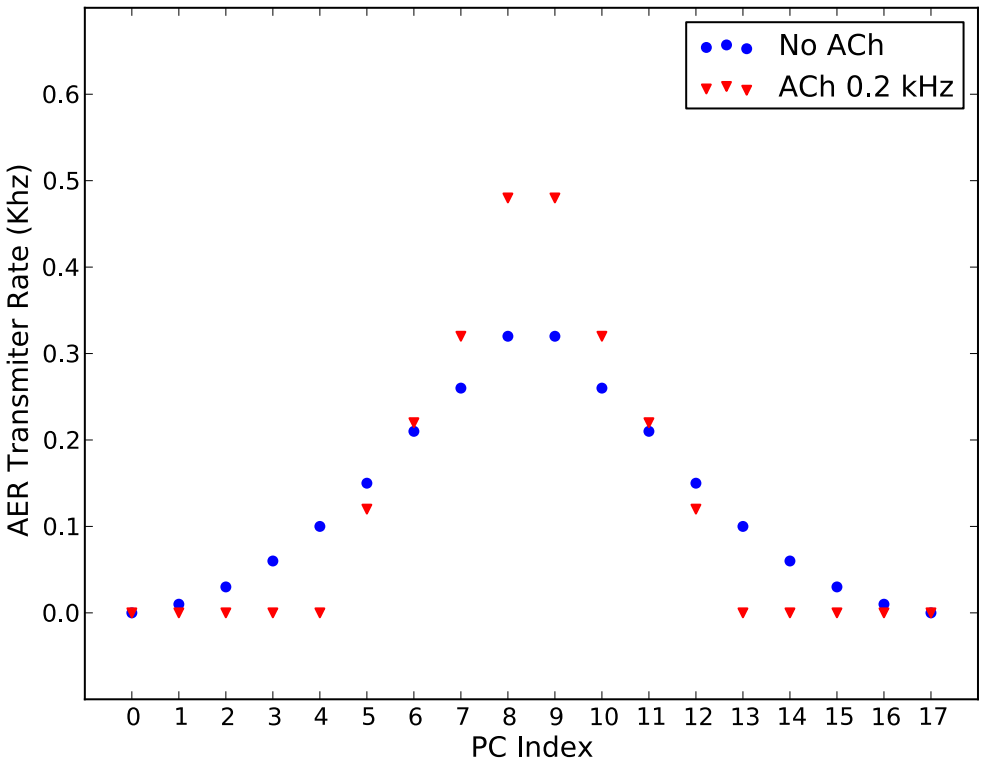


Figure 4.23: Effects of ACh Modulation in the output firing rate of the PCs. ACh input sharpens the population activity. For illustrative purposes, the PCs are indexed such that the most highly active cells are symmetrically grouped in the center.

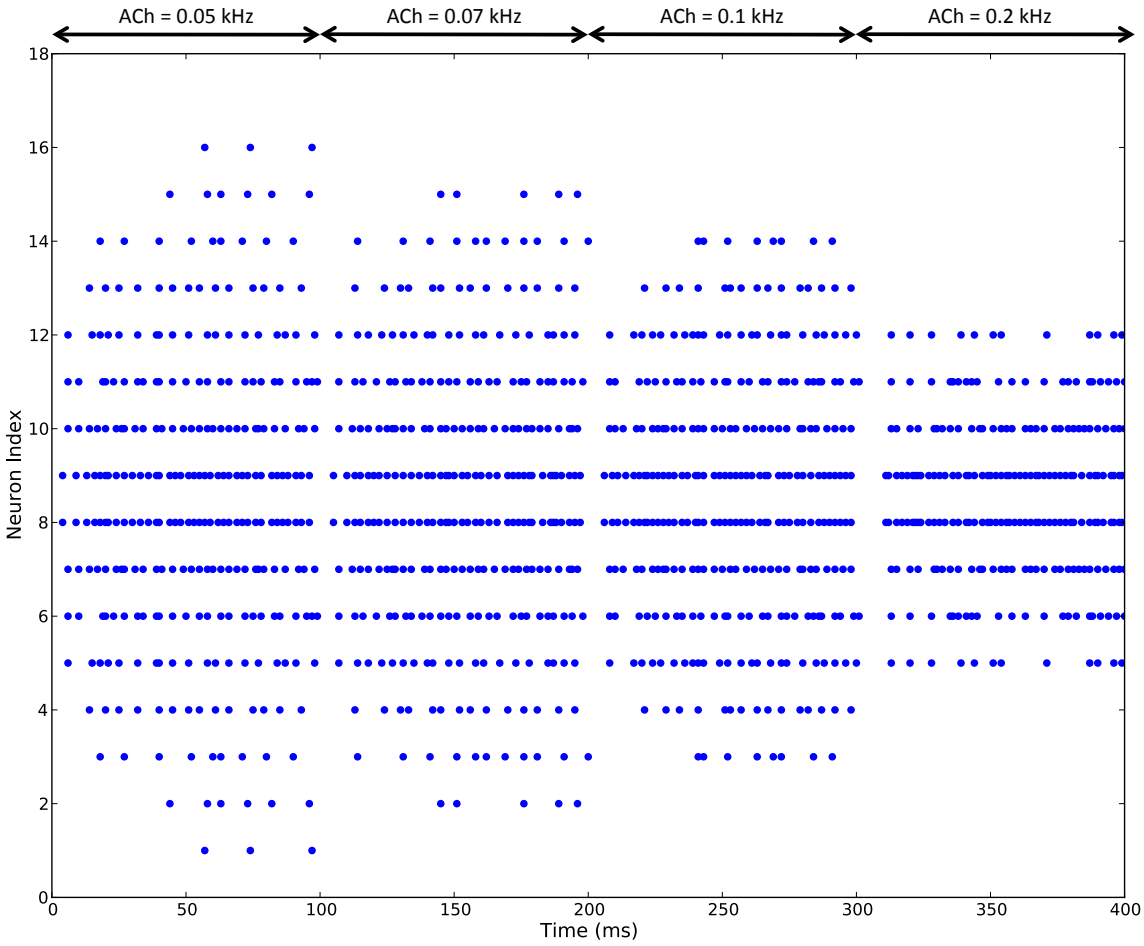


Figure 4.24: PC population activity for different levels of ACh. As the ACh input increases the weaker PCs (e.g. PC #4) lower their activity while the stronger PCs (e.g. PC #8) increase their activity, representing a progressive sharpening of the population tuning curve.

Chapter 5

Example Applications

The four neural computations described in this thesis can be combined for the creation of robust sensory representations despite many sources of ambiguities such as interfering stimuli and variable environmental conditions. As demonstration, a simple visual object representation task is illustrated below.

The visual object and its pre-chip representation is shown in Fig. 5.1. 50×50 patches from the original 200×200 greyscale image is passed through an edge filter to create a 16-dimensional vector of edges. Each element in the vector is a value in the interval [0,7] representing the fraction of pixels in the patch that constitute an edge (0 for low path edginess, 7 for high path edginess). Uniform positive noise and random Gaussian noise are then added to the vector to model effects of contrast variance, visual clutter, visual occlusion, etc. In a space spanned by the elements of the edge vector, the Euclidean distance between the original image vector and the"noisy" image vector is 12.49. The task of the neural computations is to bring these two points closer together in the 16-dimensional space of edges so that they can be recognized to be the same visual object, amidst the memory of other visual objects, by downstream neural circuits.

An unambiguous neural representation of the "noisy" image vector is created when the vector is passed through the configured computations on the chip. This is illustrated in Fig. 5.2. Each of the computations brings the noisy image vector closer to the original vector as noted by the shorter euclidean distance (ED) between the two. The global inhibition stage reduces the activation levels of the elements in the edge vector in proportion to the summed activation level, thereby reducing the effects of increased image contrast that can potentially increase activation levels beyond the available dynamic range. In the pattern decorrelation stage, the synaptic strengths of the inhibitory interactions are learned (see Sec. 4.2.3) based on the features of the noise-free vector before the noisy vector is presented. As a result, features added to the edge vector, for example due to

visual clutter, are suppressed. A similar learning stage in the pattern completion network (see Sec. 4.3.3) results in the recovery of features of the primary visual object (i.e. the clock tower) that are missing from the noisy vector, for example due to visual occlusion. Finally, the attentional modulation stage increases the activity of the strong elements of the edge vector while suppressing the weak elements, thereby preferentially enhancing the representation of the most salient features of the visual scene.

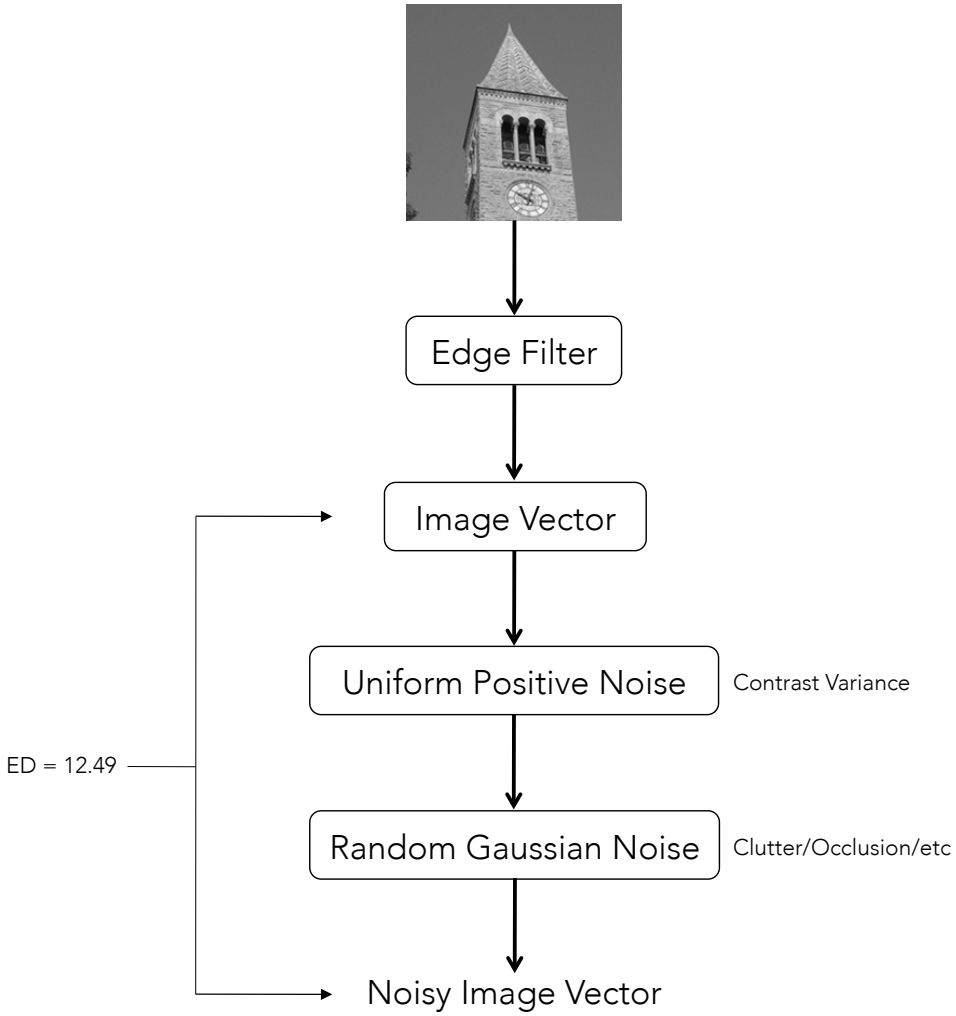


Figure 5.1: Creation of a "noisy" image vector to model various sources of ambiguities such as contrast variance, clutter, occlusion, etc.

The implementation of larger networks to process higher dimensional stimuli (for example, higher resolution images), the explicit account of many sources of variance across multiple learned objects, and the use of these neural computations in other sensory

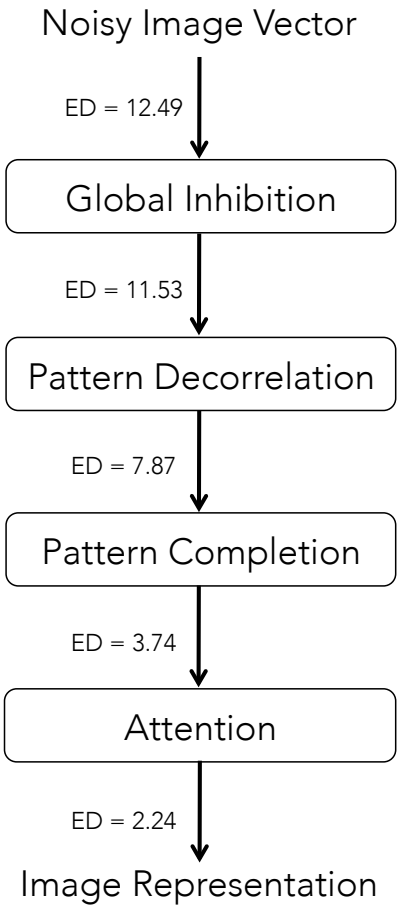
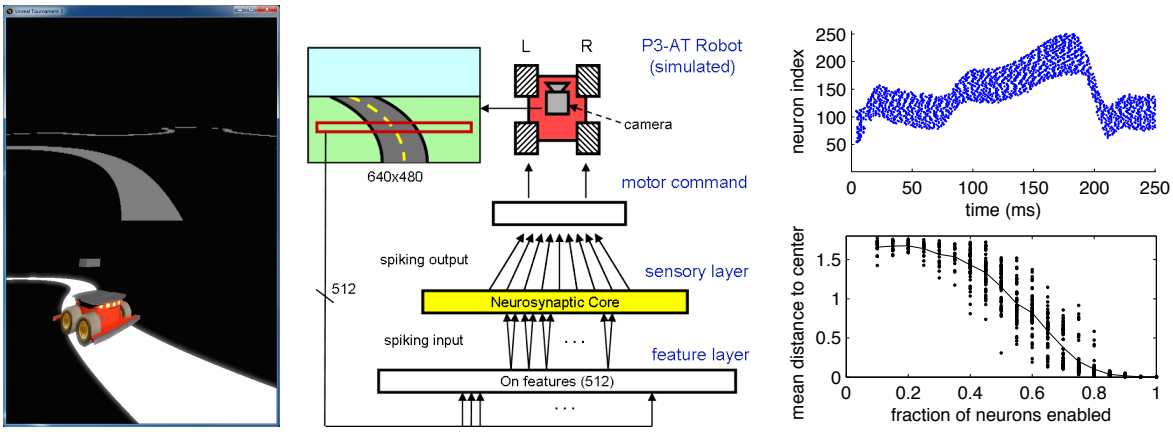


Figure 5.2: A series of neural computations on the "noisy" vector progressively reduces its euclidean distance (ED) to the noise-free representation of the vector, thereby creating a robust representation despite various sources of ambiguities.

(particularly olfaction) and non-sensory applications are left for future work.

Several other applications have been previous demonstrated [43] on the neuromorphic system described in Sec. 2.2. Some of these are illustrated in Figs. 5.3–5.5.



(a) Virtual Environment (b) System Block Diagram (c) Results

Figure 5.3: The chip was configured to drive a physics-based emulation of the MobileRobots Pioneer 3-AT (P3AT) – a four wheel drive robotic platform with a vision sensor – around a racetrack defined in the Unreal Tournament 3 virtual environment [43]. (a) Screenshots of the robot-eye view in the virtual environment at top and the simulated P3AT robot on the racetrack at bottom. (b) Block diagram of the autonomous closed-loop control system that receives visual input from a camera and issues differential wheel speed commands to the robot. (c) *top* Spike activity of network during driving task. *bottom* Average distance from center of driving track as a function of the number of neurons used.

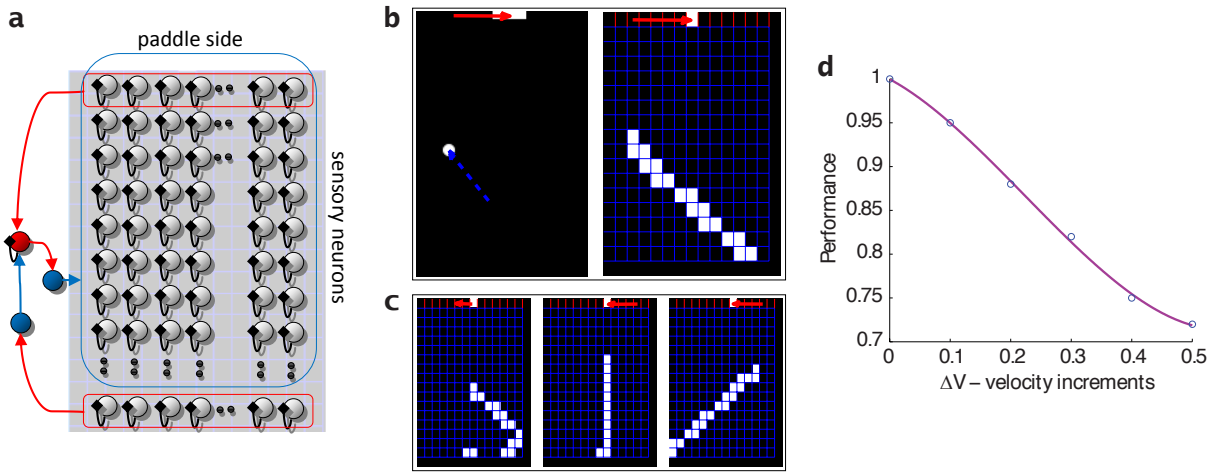


Figure 5.4: The chip configured to control the paddle in the classic game of pong [43]. The ball moves faster than the paddle and the chip is required to predict the position of the ball far enough in advance based on visual information of the ball's position. (a) Direction-selective neurons spike when the ball moves towards the paddle and remain spiking via self-excitatory synapses (black). The trajectory of the ball is associated with a position of the paddle and the latter is continuously updated. Interneurons driven by the sensor neurons at the top-most row and the bottom-most row shut down the persistent activity when the ball reaches the paddle and prevent sensor neuron activation until the ball reaches the bottom-row of the playing area. Excitatory connections are red and inhibitory connections are blue. Areas of projecting or projected connections are red/blue circled. (b) *left* Representation of the game area. *right* Example traces of active neurons left by the ball trajectory (white squares) and the motor neuron corresponding to the position of the paddle center (upper grid in red). The red arrow indicates the paddles initial and final positions. (c) Three example traces. (d) Performance of the system measured as the percentage of hits as a function of increasing ball velocity.

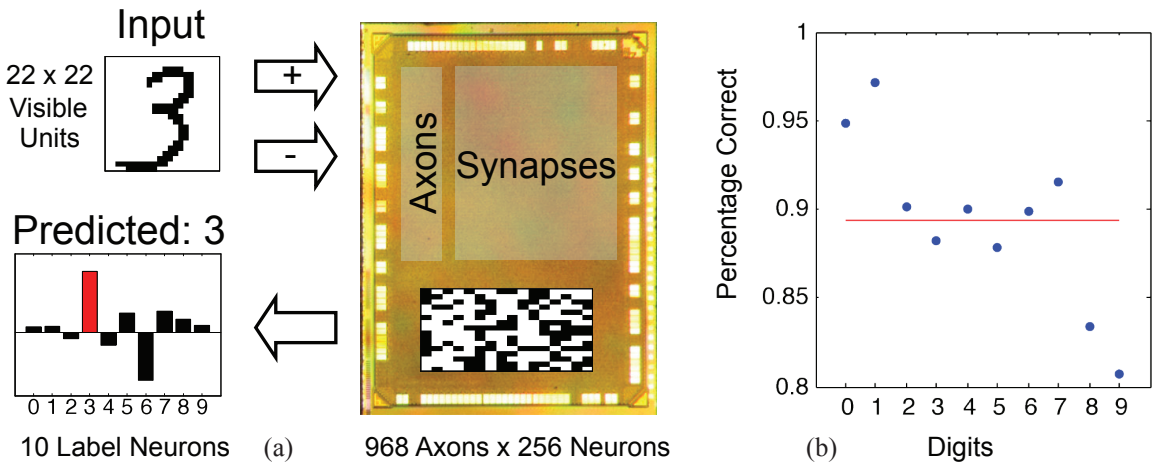


Figure 5.5: Classification of handwritten numeric digits from the MNIST dataset through a Restricted Boltzmann Machine configured on the chip. The network captures the probability distribution of pixel correlations in the digits and its activity is used to classify the digits via an off-chip linear classifier. (a) Pixels from a handwritten digit drive the activity in the chip via excitatory (+) and inhibitory (-) connections on the 16×16 grid of neurons. Spikes on the chip (indicated as black squares) create a representation of the digit based on the features that are present in the pixel correlations. This representation is sent to an off-chip linear classifier that predicts the identity of the digit. (b) Measured performance for each digit for out-of-sample data (points), and the average performance (red line).

Chapter 6

Conclusion

This thesis has described the design and implementation of a compact, energy-efficient and scalable neuromorphic architecture for mimicking neural computations in real-time artificial systems. The unique features of the architecture are the distributed on-chip synaptic memory arrays that implement high neural interconnectivity with minimal data movement, the event-driven asynchronous circuits that multiplex hardware with low active energy consumption, the compact and low-power custom-designed digital circuits that model neural operation, and the synchronization circuitries that create a 1-1 equivalence between software and hardware. The architecture is modular and its scalability and efficiency is demonstrated using a $28nm$ CMOS process with which 1 million neurons and 256 million synapses are packed within $4.2mm^2$ of silicon that dissipates only $70mW$ of power during typical network operation. Future work will include the scaling of this architecture into large multi-chip systems, incorporation of features such as synaptic plasticity and non-linear neural voltage integration, and exploitation of future advances in memory, logic and sensor technologies.

The configuration of four canonical neural computations in the chip was also demonstrated in this thesis. These computations, inspired from various regions of the biological brain, can be used in a variety of artificial pattern recognition tasks. First, a global inhibitory network that creates relational neural activity was demonstrated. Such relational representations are capable of encoding information across wide ranges of stimulus intensities using neural codes with small dynamic ranges. Second, the decorrelation of distinct neural representations was shown using reciprocal interactions between a population of principle neuron and inhibitory interneurons. This process enables the detection of distinct patterns despite large interference with background signals and noise. Next, associative interactions among neurons were demonstrated through a recurrently connected excitatory network. These interactions implement the process of pattern completion

whereby missing elements of a pattern are recovered to enable the recognition of degraded signals. Finally, a process of attentional modulation was demonstrated whereby the representation of the most salient signals of an environment are enhanced over a noisy background. Work in the immediate future will include scaled up implementations of these computations and illustration of their effectiveness in a wide variety of datasets.

Several applications await the use of neural-inspired technologies. First, conventional machine learning algorithms have fallen short in many sensory-motor problems such as visual, auditory and chemical signal recognition, and locomotor central pattern generation. The hardware and computations developed in this thesis offer novel, powerful and efficient ways of tackling many of these problems [27, 44, 45]. Second, brain-based robots [46] that explicitly model the brain's interaction with the body and the environment in real time are made more effective with systems that closely mimic neural computations such as the chips described here. These robots are promising tools for studying the brain and are novel approaches for designing intelligent physical systems. Third, brain-machine interfaces [47] intended for neural prosthetics and implants can be made more natural and effective with compact and energy-efficient spike-based computations in silicon, such as the ones demonstrated here. Finally, the hardware architecture developed here can be exploited in custom-designed brain simulation platforms for high-speed and energy-efficient simulation-based research [3, 48].

Appendix A

CHP Notation

The CHP notation we use is based on Hoare's CSP [49]. A full description of CHP and its semantics can be found in [17]. What follows is a short and informal description.

- Assignment: $a := b$. This statement means "assign the value of b to a." We also write $a\uparrow$ for $a := \mathit{true}$, and $a\downarrow$ for $a := \mathit{false}$.

- Selection: $[G1 \rightarrow S1 \,[]\, ... \,[]\, Gn \rightarrow Sn]$, where G_i's are boolean expressions (guards) and S_i's are program parts. The execution of this command corresponds to waiting until one of the guards is *true*, and then executing one of the statements with a *true* guard. The notation $[G]$ is short-hand for $[G \rightarrow \mathit{skip}]$, and denotes waiting for the predicate G to become true. If the guards are not mutually exclusive, we use the vertical bar "|" instead of "[]."

- Repetition: $*[G1 \rightarrow S1 \,[]\, ... \,[]\, Gn \rightarrow Sn]$. The execution of this command corresponds to choosing one of the *true* guards and executing the corresponding statement, repeating this until all guards evaluate to *false*. The notation $*[S]$ is short-hand for $*[\mathit{true} \rightarrow S]$.

- Send: $X!e$ means send the value of e over channel X.

- Receive: $Y?v$ means receive a value over channel Y and store it in variable v.

- Probe: The boolean expression $\overline{X}$ is *true* iff a communication over channel X can complete without suspending.

- Sequential Composition: $S; T$

- Parallel Composition: $S \parallel T$ or S, T.

BIBLIOGRAPHY

[1] S. Herculano-Houzel, "The human brain in numbers: a linearly scaled-up primate brain," *Frontiers in human neuroscience*, vol. 3, p. 31, 2009.

[2] M. L. Hines and N. T. Carnevale, "The neuron simulation environment," *Neural computation*, vol. 9, no. 6, pp. 1179–1209, 1997.

[3] R. Ananthanarayanan, S. K. Esser, H. D. Simon, and D. S. Modha, "The cat is out of the bag: cortical simulations with 10^9 neurons, 10^13 synapses," in *High Performance Computing Networking, Storage and Analysis, Proceedings of the Conference on*, pp. 1–12, IEEE, 2009.

[4] T. Sharp, F. Galluppi, A. Rast, and S. Furber, "Power-efficient simulation of detailed cortical microcircuits on spinnaker," *Journal of neuroscience methods*, vol. 210, no. 1, pp. 110–118, 2012.

[5] J. M. Nageswaran, N. Dutt, J. L. Krichmar, A. Nicolau, and A. Veidenbaum, "Efficient simulation of large-scale spiking neural networks using cuda graphics processors," in *Neural Networks, 2009. IJCNN 2009. International Joint Conference on*, pp. 2145–2152, IEEE, 2009.

[6] A. Cassidy and A. G. Andreou, "Dynamical digital silicon neurons," in *Biomedical Circuits and Systems Conference, 2008. BioCAS 2008. IEEE*, pp. 289–292, IEEE, 2008.

[7] K. Boahen, "Neurogrid: emulating a million neurons in the cortex," in *IEEE international conference of the engineering in medicine and biology society*, 2006.

[8] A.-J. Annema, B. Nauta, R. van Langevelde, and H. Tuinhout, "Analog circuits in ultra-deep-submicron CMOS," *Solid-State Circuits, IEEE Journal of*, vol. 40, no. 1, pp. 132–143, 2005.

[9] E. M. Izhikevich, "Which model to use for cortical spiking neurons?," *IEEE Transactions on Neural Networks*, vol. 15, pp. 1063–1070, 2004.

[10] N. Imam, K. Wecker, J. Tse, R. Karmazin, and R. Manohar, "Neural spiking dynamics in asynchronous digital circuits," in *Neural Networks (IJCNN), The 2013 International Joint Conference on*, pp. 1–8, IEEE, 2013.

[11] N. Imam, F. Akopyan, J. Arthur, P. Merolla, R. Manohar, and D. S. Modha, "A digital neurosynaptic core using event-driven QDI circuits," in *Asynchronous Circuits and Systems (ASYNC), 2012 18th IEEE International Symposium on*, pp. 25–32, IEEE, 2012.

[12] N. Imam and R. Manohar, "Address-event communication using token-ring mutual exclusion," in *ASYNC*, pp. 99–108, IEEE Computer Society, 2011.

[13] P. A. Merolla, J. V. Arthur, B. E. Shi, and K. A. Boahen, "Expandable networks for neuromorphic chips," *Circuits and Systems I: Regular Papers, IEEE Transactions on*, vol. 54, pp. 301 –311, feb. 2007.

[14] N. Imam, "A communications infrastructure for multi-chip neuromorphic systems," *Masters Thesis. Cornell University*, 2011.

[15] E. Stromatias, F. Galluppi, C. Patterson, and S. Furber, "Power analysis of large-scale, real-time neural networks on spinnaker," in *Neural Networks (IJCNN), The 2013 International Joint Conference on*, pp. 1–8, IEEE, 2013.

[16] S. H. Jo, T. Chang, I. Ebong, B. B. Bhadviya, P. Mazumder, and W. Lu, "Nanoscale memristor device as synapse in neuromorphic systems," *Nano letters*, vol. 10, no. 4, pp. 1297–1301, 2010.

[17] A. J. Martin, "Programming in VLSI: From communicating processes to delay-insensitive circuits," in *Developments in Concurrency and Communication, UT Year of Programming Series* (C. A. R. Hoare, ed.), pp. 1–64, Addison-Wesley, 1990.

[18] F. Akopyan, *Hybrid Synchronous/Asynchronous Design*. Ph.D. thesis, Cornell University, April 2011.

[19] A. J. Martin, "Distributed mutual exclusion on a ring of processes," *Sci. Comput. Program.*, vol. 5, pp. 265–276, October 1985.

[20] P. Merolla *et al.*, "Inorganic brain-like machine," *In Review*, 2014.

[21] A. S. Cassidy, P. Merolla, J. V. Arthur, S. K. Esser, B. Jackson, R. Alvarez-icaza, P. Datta, J. Sawada, T. M. Wong, V. Feldman, A. Amir, D. B. dayan Rubin, E. Mcquinn, W. P. Risk, and D. S. Modha, "Cognitive computing building block: A versatile and efficient digital neuron model for neurosynaptic cores," in *International Joint Conference on Neural Networks (IJCNN). IEEE*, 2013.

[22] R. Preissl, T. M. Wong, P. Datta, M. Flickner, R. Singh, S. K. Esser, W. P. Risk, H. D. Simon, and D. S. Modha, "Compass: A scalable simulator for an architecture for cognitive computing," in *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, p. 54, IEEE Computer Society Press, 2012.

[23] S. Panzeri, N. Brunel, N. K. Logothetis, and C. Kayser, "Sensory neural codes using multiplexed temporal scales," *Trends in neurosciences*, vol. 33, no. 3, pp. 111–120, 2010.

[24] B. Grothe, "New roles for synaptic inhibition in sound localization," *Nature Reviews Neuroscience*, vol. 4, no. 7, pp. 540–550, 2003.

[25] M. Carandini and D. J. Heeger, "Normalization as a canonical neural computation," *Nature Reviews Neuroscience*, vol. 13, no. 1, pp. 51–62, 2012.

[26] T. A. Cleland, B. A. Johnson, M. Leon, and C. Linster, "Relational representation in the olfactory system," *Proceedings of the National Academy of Sciences*, vol. 104, no. 6, pp. 1953–1958, 2007.

[27] N. Imam, T. A. Cleland, R. Manohar, P. A. Merolla, J. V. Arthur, F. Akopyan, and D. S. Modha, "Implementation of olfactory bulb glomerular-layer computations in a digital neurosynaptic core," *Frontiers in neuroscience*, vol. 6, 2012.

[28] B. A. Olshausen and D. J. Field, "Sparse coding of sensory inputs," *Current opinion in neurobiology*, vol. 14, no. 4, pp. 481–487, 2004.

[29] X. Pitkow and M. Meister, "Decorrelation and efficient coding by retinal ganglion cells," *Nature neuroscience*, vol. 15, no. 4, pp. 628–635, 2012.

[30] T. A. Cleland, "Early transformations in odor representation," *Trends in neurosciences*, vol. 33, no. 3, pp. 130–139, 2010.

[31] E. Rolls, "The mechanisms for pattern completion and pattern separation in the hippocampus," *Frontiers in Systems Neuroscience*, vol. 7, no. 74, 2013.

[32] F. Sharifian, L. Nurminen, and S. Vanni, "Visual interactions conform to pattern decorrelation in multiple cortical areas," *PLoS ONE*, vol. 8, p. e68046, 07 2013.

[33] F. Sharifian, L. Nurminen, and S. Vanni, "Visual interactions conform to pattern decorrelation in multiple cortical areas," *PLoS ONE*, vol. 8, p. e68046, 07 2013.

[34] C. Linster and T. A. Cleland, "Decorrelation of odor representations via spike timing-dependent plasticity," *Frontiers in computational neuroscience*, vol. 4, 2010.

[35] A. Sahay, D. A. Wilson, and R. Hen, "Pattern separation: a common function for new neurons in hippocampus and olfactory bulb," *Neuron*, vol. 70, no. 4, pp. 582–588, 2011.

[36] V. Cutsuridis and T. Wennekers, "Hippocampus, microcircuits and associative memory," *Neural Networks*, vol. 22, no. 8, pp. 1120–1128, 2009.

[37] L. B. Haberly, "Parallel-distributed processing in olfactory cortex: new insights from morphological and physiological analysis of neuronal circuitry," *Chemical senses*, vol. 26, no. 5, pp. 551–576, 2001.

[38] M. R. Picciotto, M. J. Higley, and Y. S. Mineur, "Acetylcholine as a neuromodulator: cholinergic signaling shapes nervous system function and behavior," *Neuron*, vol. 76, no. 1, pp. 116–129, 2012.

[39] M. E. Hasselmo and M. Sarter, "Modes and models of forebrain cholinergic neuromodulation of cognition," *Neuropsychopharmacology*, vol. 36, no. 1, pp. 52–73, 2011.

[40] M. Roberts, W. Zinke, K. Guo, R. Robertson, J. McDonald, and A. Thiele, "Acetylcholine dynamically controls spatial integration in marmoset primary visual cortex," *Journal of neurophysiology*, vol. 93, no. 4, p. 2062, 2005.

[41] G. Li and T. A. Cleland, "A two-layer biophysical model of cholinergic neuromodulation in olfactory bulb," *The Journal of Neuroscience*, vol. 33, no. 7, pp. 3037–3058, 2013.

[42] A. J. Yu and P. Dayan, "Uncertainty, neuromodulation, and attention," *Neuron*, vol. 46, no. 4, pp. 681–692, 2005.

[43] J. V. Arthur, P. A. Merolla, F. Akopyan, R. Alvarez, A. Cassidy, S. Chandra, S. K. Esser, N. Imam, W. Risk, D. B. D. Rubin, *et al.*, "Building block of a programmable neuromorphic substrate: A digital neurosynaptic core," in *Neural Networks (IJCNN), The 2012 International Joint Conference on*, pp. 1–8, IEEE, 2012.

[44] S.-C. Liu and T. Delbruck, "Neuromorphic sensory systems," *Current opinion in neurobiology*, vol. 20, no. 3, pp. 288–295, 2010.

[45] A. J. Ijspeert, "Central pattern generators for locomotion control in animals and robots: a review," *Neural Networks*, vol. 21, no. 4, pp. 642–653, 2008.

[46] G. M. Edelman, "Learning in and from brain-based devices," *Science*, vol. 318, no. 5853, pp. 1103–1105, 2007.

[47] M. A. Lebedev and M. A. Nicolelis, "Brain–machine interfaces: past, present and future," *TRENDS in Neurosciences*, vol. 29, no. 9, pp. 536–546, 2006.

[48] E. M. Izhikevich and G. M. Edelman, "Large-scale model of mammalian thalamocortical systems," *Proceedings of the national academy of sciences*, vol. 105, no. 9, pp. 3593–3598, 2008.

[49] C. A. R. Hoare, "Communicating sequential processes," *Communications of the ACM*, vol. 21, no. 8, pp. 666–677, 1978.