University of California
Santa Barbara

Digital Non-binary Spiking Communication and Computation Channel

A dissertation submitted in partial satisfaction
of the requirements for the degree

Doctor of Philosophy
in
Electrical and Computer Engineering

by

Carrie Hartley Segal

Committee in charge:

Professor Forrest Brewer , Chair
Professor Li-C Wang
Professor Timothy Sherwood
Professor Yuan Xie

March 2022

The Dissertation of Carrie Hartley Segal is approved.

Professor Li-C Wang

Professor Timothy Sherwood

Professor Yuan Xie

Professor Forrest Brewer , Committee Chair

March 2022

Digital Non-binary Spiking Communication and Computation Channel

by

Carrie Hartley Segal

Dedicated to my children

Alexander Andreas Segal
Andre Max Segal

Acknowledgements

I appreciate the support and knowledge contributions from my advising committee, Tim Sherwood, Li C Wang, Yuan Xie and my advisor, Forrest Brewer.

This dissertation is inspired by the dedicated undergraduate teachers and advisors during my time working towards my Bachelor of Science at State University of New York at Stony Brook with a major in Physics. Without the support and encouragement I received from my undergraduate institution I would not have had the resolve to pursue graduate studies, let alone the necessary foundation in mathematics and science needed to become an outstanding engineer. In particular the support of Dr. Hal Metcalf and Dr. John Noe motivated my decision to keep working at experimental methods to study the limits of computation.

My early college career at the State University of New York at Fashion Institute of Technology was also instructive in how to become a designer. Even though design seems much different than computer engineering, the best practices for design extend across disciplines. My time spent living in New York, New York will always be a memorable experience where I was first able to appreciate the diversity of the world and the constant pressure to innovate.

Upon arrival at UCSB, the Computer Engineering department, especially Val De Veyra, welcomed me and provided plenty of opportunities to talk to very wise engineers and teachers, including Umesh Mishra, Volkan Rodoplu and Michael Melliar-Smith, all who have left lasting impressions on my perspectives of what constitutes worthwhile teaching and engineering.

The constant support and kindness from my husband Ivgeni Segal and his 'start-up team' at FindTheBest/Graphiq made my time in Santa Barbara, CA, with my dog, Logan, and my cats, Mr. Molly and Cody, memorable. I spent several years as a teaching

assistant and enjoyed my time working with the computer engineering and computer science undergraduates in the Software/Hardware interfacing courses and Computer Engineering Capstone Projects with Yogananda Isukapalli.

When I arrived at UCSB I already knew the direction my graduate research was intended to take, and thankfully I was able to find Forrest Brewer's research lab where independent pursuit of experimental and practical engineering is supported and encouraged. My labmates over the years provided formative support for early ideas, and working with Kunal Arya, Merritt Miller, Aditya Dalakoti, Prashansa Mukim and David Mc Carthy are my best memories of UCSB.

I thank my parents, John Hartley and Nancy Ray Hartley for their unending support and assistance in completing graduate studies. Without their constant help and advice I doubt I will accomplish much of anything. My brothers John Jason Hartley and Finley Hartley and their partners Sabine Hartley and Charles Taylor also provided support and help when it seemed like it would be impossible to continue at UCSB.

After working as a teaching assistant, while continuing to pursue my graduate research goals, I was lucky enough to work as a part of the amazing engineering team at Senseeker Engineering as an ASIC design engineer. The team at Senseeker really knows how to take apart complex problems and build solutions. Their support and encouragement reminds me to persevere and build any time I find myself thinking how I will ever accomplish a useful contribution to science and engineering.

Most importantly my two children born during my time at UCSB, Alexander Andreas Segal and Andre Max Segal have directed my attention to considering why it is necessary to engineer in the first place. Without their hopeful voices asking why and how, I am certain there would be no good reason to investigate why and how.

Curriculum Vitæ

Carrie Hartley Segal

Education

2022	Ph.D. Computer Engineering (Expected), University of California, Santa Barbara.
2017	M.S. Computer Engineering, University of California, Santa Barbara.
2013	B.S. Physics, Stonybrook University, State University of New York.

Publications

2022	Low Energy Response of Spike Train Encoded Data. ICICT 2022 ***Carrie Hartley Segal***
	Lower bounds on neuromorphic communication by representation as stationary computation (To be sent for review at Journal of Neuromorphic Computing and Engineering) ***Carrie Hartley Segal***
2019	Distributed Pulse Rotary Traveling Wave VCO: Architecture and Design. ISVLSI 2019. *Prashansa Mukim, Aditya Dalakoti, David McCarthy, Brandon Pon,* ***Carrie H. Segal****, Merritt Miller, James F. Buckwalter and Forrest Brewer*
2018	Impolite High Speed Interfaces with Asynchronous Pulse Logic. GLSVLSI 2018. *Merritt Miller,* ***Carrie H. Segal****, David McCarthy, Forrest Brewer*
	Low Power Circuits for Satellite Object Geolocation and Detection, Poster, spatial@ucsb.local2018 Improving Information Accuracy for Extreme Events ***Carrie H. Segal****, Aditya Dalakoti, Prashansa Mukim, Merritt Miller and Forrest Brewer*
2016	Minimal Connectivity Graph for Hardware Artificial Neural Networks. WiML (Co-located NIPS 2016) ***Carrie H. Segal***
	5 Gbps Interconnection Links for Particle Accelerator Environments. Journal of Radiation Effects *Forrest Brewer, Merritt Miller,* ***Carrie H. Segal****, Di Wang, and G. Magazzu.*
	Connectivity Effects of Energy and Area on Asynchronous Neuromorphic Serial Links. SLIP 2016. ***Carrie H. Segal****, Aditya Dalakoti, Merritt Miller, Forrest Brewer*
	Asynchronous High Speed Serial Links Analysis using Integrated Charge for Event Detection *Aditya Dalakoti,* ***Carrie H. Segal****, Merritt Miller, Forrest Brewer*

Abstract

Digital Non-binary Spiking Communication and Computation Channel

by

Carrie Hartley Segal

In the search for intelligent silicon, the energy costs of traditional sensing and computation are barriers to progress and are forcing new modalities for communication, computation and storage. Conventional signaling (discrete binary digital, analog level) suffers from non-idealities due to physical noise on large wires between miniature transistors. The errors found in the communication channels between the devices cause increased power demands because a classical computation must use enough energy to compute and transmit the answer across wires. This work combines recent advances in computation and communication, to simultaneously sense and transmit information acquired while sending the data through a spiking communication channel with additional computation capabilities.

Spiking or pulse-based asynchronous computation and communication schemes indicate additional energy bounds useful for understanding noisy answers. The use of pulse signals provide behaviorally robust and scalable system architectures for novel encoders. The encoders take advantage of hierarchical uneven fractional connectivity to transmit data during a space-time computation for the purposes of neuromorphic communication. These encoders enable semi-intelligent sensors capable of efficient data transfer from practical CMOS mixed-signal race logic integrated circuits.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

"Software is eating the world." [1], so spoke a prominent venture capitalist in the early 2010's while watching the technology stocks begin to dominate the economic output of Silicon Valley, the home of innovation and technological development which pioneered the age of the personal computer. In hindsight, this prediction has panned out, as the success of the surviving dot-com's lead to a plethora of technological innovation and economic growth. The driving force propelling the economic success of the software rested upon the incredible advances in computing power made possible by silicon valley hardware houses which reliably delivered improvements in the computational abilities of their central processing units (CPU) and increasingly larger memories with continual improvements in the processor-memory bandwidth.

Computer peripherals in the form of printers, monitors, speakers, microphones and cameras continued to propagate in their form and function. Peripherals integrated together through the CPU lead to the development of computers as recognized by the general populace; machines with colorful screens, touch sensitive inputs and speakers to interact with their users sense of sound.[2] Eventually it became commonplace to expect

[1] https://future.a16z.com/software-is-eating-the-world/

[2] The senses of taste and smell never caught on for widespread use.

a small, portable computer to contain accelerometers, gyroscopes, radios, temperature, touch and camera sensors. Perhaps most importantly, wireless communications to access to the globally connected internet.

The dynamic interplay between hardware and software is a history stretching back to the first questions about the theoretical development of a universal computing engine and the first programs written for the nonexistent physical computing machines. The boundary between hardware and software is called the software-hardware interface. A simplistic definition of the boundary recognizes software as a *program* and the hardware as the *machine that spends energy to make the program state change over time* (execute). The interface is the set of core operations that the hardware builder writes down and makes available for a software author to access. The interplay involved between the hardware builder and the software author helps to define the software-hardware interface. Sometimes the hardware builder and the software author are the same person, who will build the machine and decide how to define the programs that the machine will be able to execute.

As the software industry grew to encompass more and greater functionality the hardware instruction set architecture (ISA) developed at a slower rate. The complexity of software, however, continued to grow, with development of programming languages and operating systems, built upon the large ecosystem of already developed software libraries. When the CPU was unable to handle the increasing computational load for some tasks, additional computer hardware, such as encryption and graphics processing units (GPU), were added. Once software unleashed its rapid pace of innovation upon GPU's, new types of programs forming the basis of modern artificial intelligence began to propagate throughout the software ecosystem, rapidly leading to demand for specialized hardware enabling fast and less energy intensive implementations of specialized pattern recognition tools. Neuromorphic co-processors are now becoming prevalent. The role of the CPU

as the coordinator of the many different hardware integrated circuits continues based on the ease for programmable, reliable behavior. Each additional hardware peripheral includes additional ISA expansions, and the software is able to support expansions to the dominant programming languages as well as special purpose compilers for offloading specialist compute to these neuromorphic cores.

Recognizing the dominance of the existing software ecosystem is critical for successful development and deployment of these new types of neuromorphic hardware. The software-hardware interface has never been more critical for the widespread adoption of new neuromorphic hardware, because from an economic perspective, "software has eaten the world". The following work presents potential variant of a neuromorphic co-processor, where the overarching goal is a hardware sensor that transmits a non-binary spiking data stream into a hardware decoder for further processing by software. We believe this abstraction definition places another tool into the hands of the computer engineer for how to best define the software-hardware interface when interacting with sensed data.

1.1 Unsupervised Communications Channel

A communication channel sends a transmission, described herein, as a probabilistic model of a circuit for sensing a signal and transmitting it into a software friendly description. It is an example of neuromorphic engineering because it is a biologically inspired design for a computer, intended to persist in challenging environments where the signal to noise ratio is unknown or poorly defined.

For most of the history of computation there was little need to be concerned about building a machine that is sensitive to the analog signals of the environment. Circuits are loops made from interconnected electrical signals. They range from the simplest single loop circuit to modern system-on-chip circuit's (SOC), which are designed to be

incorporated onto electrical printed circuit boards (PCB) as parts of even larger systems connected to sensors for many different situations and environments. Most computers are digital systems purposely designed with binary signals. The binary nature of the system was designed into it, from the first decisions made to build a computer using only the 16 Boolean functions it is possible to construct with two inputs and one output and 4 Boolean functions with only 1 input and 1 output.[7]

Binary signal encoding is necessary to protect the computation signals from the noisy analog world. Binary digital signaling provides stable margins and is designed to operate with known manufacturing and temperature variations given two voltage levels assigned meaning as 0 and 1. However, the computer is only able to function within the specifications of its design. If the power is unstable or the fabrication variations are too high the computation performance will degrade into the low-energy, high-noise regime we have sought to avoid.[4] As the power fails and the physical environment heats and cools the output of the computer no longer is as expected, eventually the formerly understandable output becomes only noise.

Noise is the fundamental problem for sensing systems, from the earliest days of telegraph and telephone operators struggling to understand the message sent from far away, to modern issues with image classification systems presented with slightly distorted imagery, classifying cherries as flamingos. Increasingly, noise is present as a problem for computation systems as well, given the drive for improving computer system's performance while seeking lower operating power. It is further exacerbated by fabricating at smaller technology nodes which are more prone to variation and timing margin errors.

Space-time computing is when information is embedded in the timing of the computation signal.[8] That unconventional practice means the slope of the signal edge becomes a matter of practical concern, since the abstraction to binary digital signals is no longer valid.

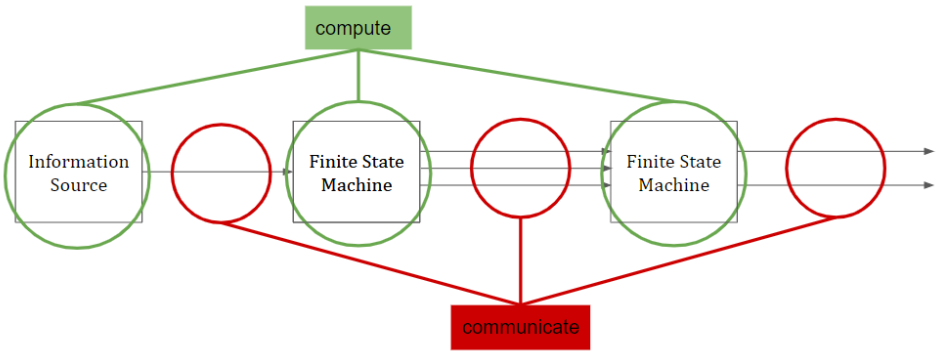


Figure 1.1: Computation and Communication transform the dimension of observable data

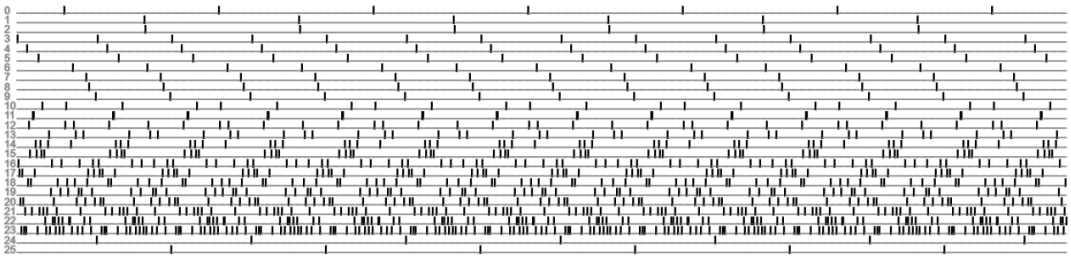


Figure 1.2: A known non-binary digital stationary broadcast on a spiking channel.

Figure 1.1 illustrates how a *noiseless* computation and communication system operates to seamlessly transform the dimension of observable data. In Figure 1.1 the information source for a computer comes from the resources available in the memory hierarchy of the machine, which over long enough time scales and repeated access will begin to degrade and introduce small errors into the data. The finite state machines used for computation are expected to begin operation only when the inputs are constant, and to output again only once their internal states are constant. The timing is handled by global synchronous clocking, which, when successful, prevents noise from uneven Boolean gate timing.

In this work, additional practices for space-time computing are developed to enhance the computation capabilities when information is embedded in the *noisy* timing of the computation signal. This is done by pairing a known non-binary digital communication

signal as a stationary broadcast[3], shown in Figure 1.2, with the noise from the sensing array. The conversion from sensor response to a data description is handled through a series of taps into the sensing circuit to form a communication from the large number of sensors at the input of the spiking channel. The communication can travel a long distance through wires eventually transmitting to a paired decoder. The communication experiences the same effects as the computation because both are electronic signals moving across the same media. The output of the known stationary broadcast communication becomes distorted as the errors caused by power and temperature change the known expected output. Neuromorphic asynchronous pulse logic systems when implemented in mixed signal, mixed process fabrication technologies demonstrate promise for exceptionally low power, failure resilient, semi-intelligent silicon circuits and push the design space limits of VLSI.

1.2 Thesis Statement

The use of pulse signals provide behaviorally robust and scalable system architectures for novel encoders that take advantage of hierarchical uneven fractional connectivity to transmit data during a space-time computation for the purposes of neuromorphic communication.

This thesis statement developed from investigations into the fan-in and fan-out of a type of communication circuit used for asynchronous serial links and the application of that circuit to race logic.[9], [10] The support for the statement is presented as follows: Chapter 2 provides detailed information about mathematical tools used to justify later experimental results. Chapter 3 details how neuromorphic inspired circuit design allows novel designs with high information, low energy performance. Chapter 4 reviews

[3]A stationary broadcast is a periodic (only in the case of noiseless operation) signal that is expected to endlessly repeat. While it may become interrupted due to noise, it will resume eventually.

communication theory as applied to a spiking channel. Chapter 5 understands minimally acceptable communications sufficient to claim semi-intelligent operation. It also explores biological computation and restraints on what could be classified as intelligent energy use. Chapter 6 is about the development of special purpose computers that rather quickly pass through all of their physically possible states before developing into a stationary operation mode. Chapter 7 establishes a lower bound for neuromorphic communication and demonstrates the principles to apply for increasing the system size. Chapter 8 connects the findings and studies how to build a software-hardware interface for compressed spike-encoded high capacity non-binary digital transmission.

1.2.1 Motivation for Thesis Statement

The motivation to develop the thesis statement originated from the topic "Noise and Process Scalability in Space-Time Computing with Race Logic". Space-Time Computing is an alternative type of computation, that is currently (circa 2020) simulated for general purpose computation.[11] An **a**pplication **s**pecific **i**ntegrated **c**ircuit (ASIC) designed to carry out this type of computation already existed, with the development of an ASIC for the custom purpose of DNA sequencing and pair matching.[12] The DNA sequencing engine requires a reset signal before each new computation can be performed, and when the program is held constant the engineered input reset sequence and generated output sequence create a periodic waveform when sampled at a frequency lower than than the maximum computation time for the DNA sequence engine. This periodic race logic waveform is reminiscent of a stationary computation, and drove investigation presented in the dissertation about the nature and utility of a stationary computer.

A stationary computation process is a term that is practical with an understanding of temporal race logic, where computation is embedded in the timing of signals as they

propagate across the computer. Further explanations of stationary computation and computing are in chapter 6 and chapter 7.

The connectivity of a physical race logic structure was at first taken to be local and uniform.[12],[13] The similarities between temporal computation and neural networks was noticed and developed into a computational paradigm, to be implemented in both software and hardware.[8] Connectivity is a topic that influences both software and hardware designs of neural networks, because, while the 'gold standard' for a software programmer is a fully connected network; in practice even the most generous computer will exhibit unexplained slow-downs if a fully connected network is programmed.[14] Once a software programmer understands that their request for full connectivity places an undue burden on the memory to CPU interface, alternatives to full connectivity become desired. Connectivity models are inspected in chapter 3.

Space-time computing and connectivity models both mix together the concept of communication with the intent to compute an output.[15] Communication in electronic digital Boolean circuits is a clearly defined methodology which requires the assumptions of the physical system to be clearly stated and understood by end users of the communication system.[16],[17],[18] Neuromorphic systems which intermingle computation and communication require stating of the set of assumptions used to handle the mixed nature of the two systems and meeting the assumptions of both the computer and communications leads to system designs classified as either analog or digital.[19]

A computer is modeled as a finite state machine (FSM). When a CPU is outputting a pattern of Boolean symbols over time its operation should match a specification fully describable by a FSM.[20] The only time this is not that case is when there is a failure of the hardware to maintain the desired level of abstraction.[21] The failure to have clear layers of abstraction caused Edsger W. Dijkstra, an early computer programmer, to write lengthy justifiably angry reports on how frustrating a failure of hardware not accounted

for in the specification can be. However, since then, alternative computation models have later discovered that there are advantages to be found in using the physical failure modes for purposeful computation, as in the case of 'Abusing Hardware Race Conditions for High Throughput Energy Efficient Computation'.[22]

A communication channel can have many purposes. There are communication channels designed to send signals long distances (*long* as in half-way around the earth), designed to send colorful images back from outer space (Images of Pluto), and designed to send a file from a computer across a cable to a printer. Voyager 1 and Voyager 2 have achieved interstellar distance, and continue to send data back to earth. Small scale communication channels are prevalent in modern technology.

A communication channel is used to output the result of the FSM back into the digital memories of the computer. In Chapter 4 how to build a spiking neural communication channel is studied. Biological systems have evidence of a communication channel in the form of sensing nerves and coordinating brain, but these systems are not quite cleanly defined in the assumptions necessary for their operation.[23],[24] Models of biological cells often refer to *channels*, meaning a chemical sensor port enabling communication to and from a cell.[25] In fact, the evaluation of biological systems rely on the scientist defining some measurable response from the biological system.[26],[27],[28] Combining the mathematical theory of communication, the finite state machine nature of a computer, and the biological ability to *instantaneously* respond to the environment has resulted in the thesis statement which probes how a biological system is able to compute a response in time across its communications system which is seemingly faster than the speed of data propagation down a wire.[29],[30],[31]

1.3 Permission and Attributes

Chapter 3 is an updated presentation of "Connectivity Effects on Energy and Area for Neuromorphic System with High Speed Asynchronous Pulse Mode Links" which previously appeared in SLIP '16: Proceedings of the 18th System Level Interconnect Prediction Workshop June 2016 Article No.: 3 Pages 17 https://doi.org/10.1145/2947357.2947365, C. Segal et al. . Chapter 6.3 has elements from "Impolite High Speed Interfaces with Asynchronous Pulse Logic", GLSVLSI 2018 https://doi.org/10.1145/3194554.3194592 M. Miller et al. . Chapter 6 and section 2.5.2 contains elements from "Low Energy Response of Spike Train Encoded Data" C.H. Segal. Previously published in Proceedings of Seventh International Congress on Information and Communication Technology: ICICT 2022, London, Volume 2 *Reproduced with permission from Springer Nature.* Chapter 7 is a copy of "Lower bound on neuromorphic communication by representation as stationary computation" C.H. Segal to be sent to review at Journal of Neuromorphic Computing and Engineering. The author C.H. Segal bears all responsibility for errors and mistakes yet to be found in this work.

Chapter 2

Mathematical Tools

This dissertation combines concepts from computer engineering, information theory, computer science, neuroscience, biology and physics. It is an interdisciplinary dissertation for a computer engineer, and to make sure all of the notation used is commonly understood despite different backgrounds, the preliminary concepts about entropy, relation of entropy and energy, entropy estimation, information channel capacity, Markov chains and ergodic hierarchy are discussed, with references.

Entropy, Markov chains and application of ergodic hierarchy to finite known Markov chains are used throughout. Channel capacity is applied in communication. Mixing times are used to establish lower bounds. Hierarchical Markov chains are used in connectivity of neuromorphic systems.

2.1 Claude Shannon's Entropy

The probability metric used for information, Entropy H, shown in Equation 2.1, has a coefficient K, that "is a matter of convenience" and it amounts to the choice of a unit of measure, per Appendix 2 in "A Mathematical Theory of Communication". K is a

positive real valued number.

K is chosen to be 1 since it represents a unit utility function for code families in a binary channel. In this dissertation, K is not normalized but is scaled to depend on the coding set cardinality $|A_\chi|$ as is used in "Information Theory, Inference, and Learning Algorithms" by David Mackay.[32] This matches modern entropy estimation tools [33].

This work presents non-binary digital codes which are discrete and do not need to approximate to continuous functions, thus K is permitted to be different positive integer numbers, dependent on the dataset used to create the probabilities, $\sum_{i=1}^{n} p_i \log_n p_i$.

$$H = -K \sum_{i=1}^{n} p_i \log_n p_i \tag{2.1}$$

A short proof for this is found from Jensen's inequality (Mackay page 35), which states, in the special case of a concave $\frown$ function, and for a random variable x:

$$\mathcal{E}[f(x)] \leq f(\mathcal{E}[x]) \tag{2.2}$$

Where $\mathcal{E}$ denotes expectation. Jensen's inequality can be used to demonstrate

$$H(X) \leq f(K) \tag{2.3}$$

where f can be selected to be any concave function including log for positive values of x. Choosing $f = C \log_b$ concludes:

$$H(X) \leq K \log K \tag{2.4}$$

Jensen's inequality only holds for a measurement in which x is constant, which is specified by K, the *unit of measure* as first denoted by Shannon. Thus, entropy as a measure of

information content can take on the unit of binary digital bit, or any other metric as long as the unit of measure is understood. For example, binary digital computers can describe the entropy of the program state using the unit of 'bits'.

Since a spiking channel can be a non-binary digital channel, by adding additional $N_t > 1$ traces, entropy for alphabets of size greater than 2 is given by H_n, where n is the number of symbols, $p(i, j, ..., s)$ is probability of the sequence of symbols $i, j, ..., s$ and the summation is over all of the symbols.[34] In this case the sum is over all of the monogram symbols, that is, $|A_\chi|$, for the dataset used to build the information source.

$$H_n = -\frac{1}{n} \sum_{i,j,...,s}^{n} p(i, j, ..., s) \log_2 p(i, j, ..., s) \tag{2.5}$$

Then, the entropy H is equal to:

$$H = \lim_{n \to \infty} H_n \tag{2.6}$$

2.2 Entropy Estimation

Entropy is a generally useful concept, with many different approaches to calculating or measuring it. When the knowledge of the number of unique symbols, $|A_\chi|$, in an information source is unknown entropy must be estimated instead of directly calculated.

An estimation is made by gathering samples of the symbols output from the information source. When the number of samples are being gathered from the information source it is not known if the probability sums to one. That is $p = 1$ cannot be assumed

for the gathered data representing the state space of the function under consideration.

$$H_\infty = \begin{cases} |A_\chi|, & \text{for } p = 1 \\ E(x), & \text{for } 0 < p < 1 \\ 0, & \text{for } p = 0 \end{cases} \tag{2.7}$$

When the probability $p = 1$ is completely known, entropy H_∞ is equivalent to $|A_\chi|$. When the probability is partially known $0 < p < 1$, entropy H_∞ must be estimated using samples x and an estimator $E(x)$. When the probability is completely unknown $p = 0$, then entropy H_∞ is 0, and anyone who wants to know about it should go get some samples of whatever function they intend to calculate an entropy for.

Maximum Likelihood Entropy Estimation

The gathering of data samples has a count of the samples n, and begins with $n = 0$ samples and progresses in time, adding a sample for each timestep. A standard approach to estimating entropy builds a probability distribution function (PDF) as samples of the data are gathered. The known PDF is called the likelihood and it is used to estimate the entropy by assuming new samples are intended to maximize the likelihood. Once the number of samples n greatly exceeds $S/ln(S)$ of the support size S of the unknown function, maximum likelihood estimation represents the best possible estimate for the entropy.

JVHW Entropy Estimation

In circumstances when the number of samples n is close to the same size as S there are other ways to estimate the entropy. The method for estimating entropy that gives the best results when n is smaller than $S/ln(S)$ is the JVHW minimax estimation. JVHW

[33] minimax estimation of functionals uses a 'surprised' estimator for data it evaluates as unusual or a non-surprised estimator for the normal smooth world. It depends on how many assumptions the estimation is able to make about the input vectors.

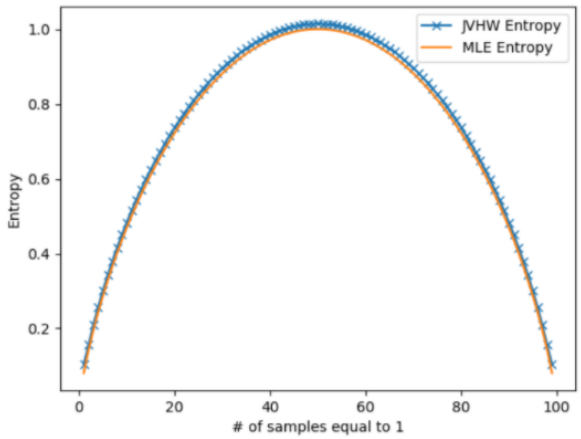


Figure 2.1: The difference between a maximum likelihood estimator (MLE) and minimax Estimator (JVHW) is apparent when 100 observations are made, and the population of 0's to 1's is varied from 100 0's, 0 1's to 0 0's, 100 1's. At the 50/50 split the JVHW estimator is slightly higher than MLE. That is because the JVHW estimator is uncertain if it is studying a random chance system (B-system) or if it is observing a simple system with only two state variables. With the benefit of the doubt it gives a slightly higher estimate of entropy, slightly beyond 1.

The estimation of entropy for a 'toy' problem which uses two sacks of balls, one sack containing only white balls and the other sack containing only black balls demonstrates the difference between the MLE estimator and the JVHW Estimator. Figure 2.1 shows a plot of the entropy when it is estimated using 100 samples and is repeated 100 times. One estimation of entropy, with $n = 100$, draws zero balls from the sack containing black balls and results in an estimation of entropy where every one of the 100 samples is identical to the other samples. On the second repetition, the draw takes 99 balls from white ball sack and 1 ball from black ball sack. These draws repeat again with 98 and 2, 97 and 3 continuing on until it reaches the point where 1/2 of the sample belongs to black and 1/2 of the sample belongs to white. Once the samples pass beyond the 50/50 split, the

estimate mirrors the results from the experiments already completed. That is because the entropy estimator does not value one color differently and that makes the sample of 43 black and 47 white equivalent to 47 black and 43 white balls.

2.2.1 Mutual Information

Mutual information $M_I(X, Y)$ is a value related to entropy. It is calculated between two vectors X and Y, and it is the sum of the entropy of X and Y minus the joint entropy of XY. X is a vector set of samples from the process that an entropy estimate is needed from and Y is also a second vector set of samples gathered from the same process. The joint entropy of XY is the estimate of entropy made when the two vectors are joined. Mutual information between two vectors indicates similarity between their underlying probability distribution function (PDF).

$$M_I(X|Y) = H(X) + H(Y) - H(X|Y) \tag{2.8}$$

Mutual information is a useful metric to compare two spike trains from the same spiking channel. The reason for this is that since there is a difference between entropy's from the same spiking channel it is not necessary to have knowledge about K, the unit of measure, for the spike train.

2.3 Relation of Entropy and Energy

The utility of a metric depends on the ability of the scientist using the metric to understand the numeric summary of the physical entity the metric is intended to represent. For physical properties, for example, the density of a mass in physics, the innate understanding of what a number is intended to convey is intuitive. Density, as conven-

tionally used in an ideal physics world model with constant temperature, is about how much volume it takes for a mass to displace in a bucket of water before the group of mass and water comes to an equilibrium and the mass is floating in the water, either resting on top of the water, partially submerged, or sunken to the bottom of the bucket.

Entropy, as it comes from information theory, does not have quite a clear intuitive understanding, and it was independently discovered/invented by Claude Shannon (named H, Equation 2.1) and Boltzmann (named S shown in Equation 2.9)[1].[35]

$$S = -k_B \ln(2) \sum_{i=0}^{n} p_i \log_n p_i \tag{2.9}$$

Boltzmann was a physicist who specialized in statistical mechanics, and calculating the entropy of a two state system S_0, S_1 with no extra knowledge of the system (so equal probability of being found in either state), results in Equation 2.10 with an entropy of $k_b \ln(2)$.

$$S = -k_B \ln(2)[\frac{1}{2}\log_2 \frac{1}{2} + \frac{1}{2}\log_2 \frac{1}{2}] = -k_B \ln(2)[-\frac{1}{2} + -\frac{1}{2}] = k_b \ln(2) \tag{2.10}$$

However, the same calculation of entropy according to Shannon, the information theorist, results in entropy K, the unit of measure given to the system, and often permitted to be 1.

$$H = -K[\frac{1}{2}\log_2 \frac{1}{2} + \frac{1}{2}\log_2 \frac{1}{2}] = -K[-\frac{1}{2} + -\frac{1}{2}] = K \tag{2.11}$$

If the entire system is small enough to represent the knowledge with 1-bit then it is

[1] I added the base to the log in equation 2.9, because the bases are inconsistent between all of the different research areas (Information theory, computer science, physics, engineering). Sometimes log means base 2, other times log means base 10, occasionally it's base e but often then written ln. I put the subscript b for k because Boltzmann must have referenced their own constant Boltzmann constant $= k_B$

a 1-bit computer. Repeating the same calculation again for a 1-bit computer that tells us we are in 1 state or the other, the results between the two entropy's are equivalent, Equation 2.12 and 2.13 below.

$$S = -k_B \ln(2)[1 \log_2 1 + 0 \log_2 0] = -k_B \ln(2)[0 + 0] = 0 \tag{2.12}$$

$$H = -K[1 \log_2 1 + 0 \log_2 0] = -K[0 + 0] = 0 \tag{2.13}$$

The resolution of the two different results of entropy (Equation 2.12 and Equation 2.10 are $k_B \ln(2)$ and also 0 depending on knowledge of probabilities) is to use a generalized definition of entropy $\mathfrak{I}$ (in bits) shown in Equation 2.14 from [35].

$$\mathfrak{I} = \Delta S - I \tag{2.14}$$

Where ΔS is the difference of the thermodynamic entropy of the system and I is the information about the system possessed by an external observer. In that case, in the case of the unknown probabilities, the 1-bit of extra information used to encode the known state is given to the external observer, $I = k_B \ln(2)$ or $I = K$, and ΔS is equal to S for probabilities 1/2 and 1/2:

$$\mathfrak{I} = \Delta S - I = k_B \ln(2) - k_B \ln(2) = 0 \tag{2.15}$$

$$\mathfrak{I} = \Delta H - I = K - K = 0 \tag{2.16}$$

In Boltzmann's case, where the statistical mechanics interpretation of entropy helps understand the number of states the probabilities are summed across, the entropy S is

a quantity representing the ignorance of the observer about the complete system. Boltzmann's entropy S is maximized when the probabilities are all equal, $1/n$, and minimized when the state is known, $S_n = 1$, all other states $S_{!n} = 0$.

However, for Shannon's case, information theory permits the assignment of K to any real valued positive constant, and there is no need to have a system with an unknown number of states in the system, because it can be the entropy of a system with a known number of states. In that case $\Delta H = |A_\chi| - H$, where K, the unit of measure is a positive integer, $K = |A_\chi|$, which is the number of elements in the set summed, and the entropy H is taken from an entropy estimator, because the measured entropy has to come from physically storable and recallable data. For the case of a binary alphabet, $|A_\chi| = 2$, when the sampled data is returning $1/2$ of one symbol and $1/2$ of the other sample, $\Delta H = 2 - 1$, and the generalized entropy is equal to 0 as shown in equation 2.17.

$$\mathfrak{I} = (2 - 1) - 1 = 0 \tag{2.17}$$

For the case of a binary alphabet, where the sampled data is returning 100% of one symbol, and 0% of the other symbol, $\Delta H = 2 - 0$, and the generalized entropy is equal to 1.

$$\mathfrak{I} = (2 - 0) - 1 = 1 \tag{2.18}$$

Thus, we can conclude that entropy is a measure of ignorance if the sampler has no information about the alphabet the entropy is intended to be a measure for and $|A_\chi|$ is unknown. However, entropy is a measure of knowledge (in particular of disorder), in the case of prior knowledge of $|A_\chi|$.

2.4 Channel Capacity

The capacity of a spiking channel in the presence of arbitrary noise can be defined as a ratio of the noise entropy power N_1, noise power N (limited to bandwidth B) vs the average signal power P, times a scaling factor of bandwidth B that is stated in cycles per second (cps).[34]

$$B \log_2 \left(\frac{P}{N_1} + \frac{N_1}{N_1} \right) \leq C \leq B \log_2 \left(\frac{P}{N_1} + \frac{N}{N_1} \right) \tag{2.19}$$

In the noiseless, controlled situation, where $N_1 = N$ and the only remaining source of noise is white thermal Gaussian noise Equation 2.19 reduces to:

$$B \log_2 \left(\frac{P}{N} + \frac{N}{N} \right) \leq C \leq B \log_2 \left(\frac{P}{N} + \frac{N}{N} \right) \tag{2.20}$$

Thus the channel capacity of a channel maximally encoded against noise (Equation 2.21) is only a ratio between the power devoted to the the signal P vs the power wasted on white Gaussian thermal noise N.

$$C = B \log_2 \left(\frac{P}{N} + 1 \right) \tag{2.21}$$

However in the case of channel where the power devoted to arbitrary noise is equal to the signal power $N_1 = P$ (Equation 2.22), the capacity C is different.

$$B \log_2 \left(\frac{P}{P} + \frac{P}{P} \right) \leq C \leq B \log_2 \left(\frac{P}{P} + \frac{N}{P} \right) \tag{2.22}$$

$$B \log_2 (1 + 1) \leq C \leq B \log_2 \left(1 + \frac{N}{P} \right) \tag{2.23}$$

The capacity of any coding strategy is limited by the channel it is transmitted through. In the case when the noise power equals signal power, $N_1 = P$, the capacity of a channel is equal or greater than the bandwidth of the channel as shown in equation 2.24. There is still the possibility that the channel will transmit at a bandwidth higher than the capacity, due to the contribution from the noise power N. In an instance of receiving transmissions beyond the bandwidth B, noise is a noticeable problem and the channel is over capacity.

$$B \leq C \leq B \log_2 \left(1 + \frac{N}{P}\right) \tag{2.24}$$

The rate R of a channel is given by the ratio of the capacity C to the entropy H.

$$R = \frac{C}{H} \tag{2.25}$$

The rate R_N of a channel for a spiking channel, with N_t traces, where each trace is enabled to transmit 1 symbol from a larger alphabet symbol, each of the traces also represents two binary symbols, either (1) symbol or (0) no symbol. Here, a spike is equivalent to symbol.

$$R_n = \frac{C}{H_n} \tag{2.26}$$

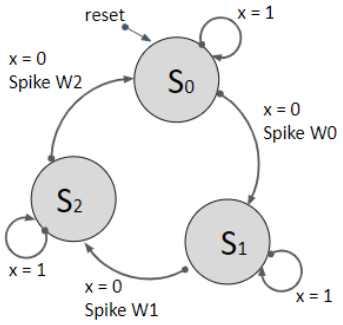
The definition of rate and capacity for a spiking channel with $N_t > 1$ rely on H_n, detailed in section 2.1.

2.5 Markov Chains and Identification by Measurement

This work focuses on 2 types of finite Markov chains, driven Markov chains and stationary Markov chains. There is also a third type of Markov chain, a hierarchical Markov chain, which combines the two.

2.5.1 Driven Markov Chain

Finite state machines (FSM) are design elements used in Boolean logic to describe the behavior of a timed digital system. The states $\{S_0, S_1, S_2, ...S_n\}$ are assigned binary labels and the transitions occur dependent upon the inputs $x_0, x_1, x_2, ...x_{Ninputs}$ and the present state $S_{present}$. When x is switching at a constant rate the FSM can be represented as a Markov chain with a table of transition probabilities as shown in Figure 2.2.



W0 [0. 1. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0.]
W1 [0. 0. 0. 1. 0. 0. 0. 0. 0. 1. 0. 0. 1. 0. 0. 0.]
W2 [0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0.]

Sa [1. 1. 1. 1. 0. 0. 0. 1. 1. 1. 1. 1. 1. 0. 0. 0.]
Sb [1. 1. 0. 0. 1. 1. 1. 1. 1. 1. 0. 0. 0. 1. 1. 1.]

	S0	S1	S2
S0	0.8	0.2	0.0
S1	0.0	0.8	0.2
S2	0.2	0.0	0.8

Figure 2.2: A Finite State Machine can be described as a driven Markov chain if the transitions are moderated with a constant symbol x that toggles at a periodic rate. With three states it has transition probabilities between the different states. The state can be represented as a binary variable ($\{S_a, S_b\}$)or it could be represented as an event on wires ($\{W0, W1, W2\}$) indicating a transition is taking place.

2.5.2 Stationary Markov Chain

A Markov chain is a design element with unique stationary states $\{S_0, S_1, S_2, ...S_n\}$. It is a dynamical system $(\Omega, \mathcal{B}, P, T)$ that is *stationary* if $P(T^{-1}G) = P(G)$. The changing of states is governed by a fixed probability $P(S, \cdot)$ of transition between stationary states taken from a finite set Ω. A finite Markov chain has the property that the transition matrix P with dimensions of $\Omega \times \Omega$ is sufficient to describe it because the sequence of prior transitions leading up to the present state does not change the future states.

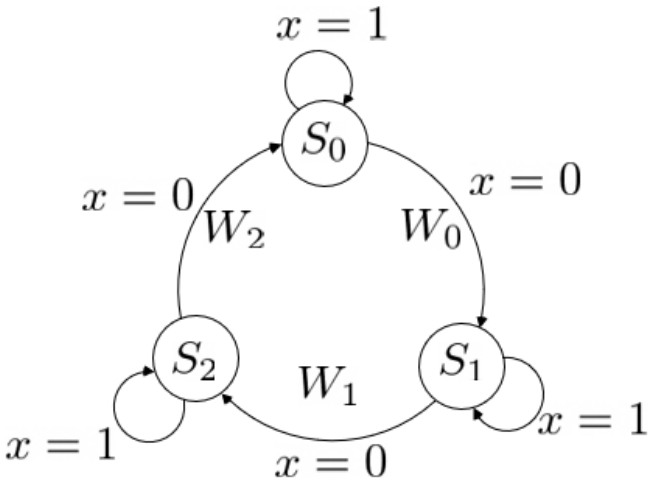


Figure 2.3: A Markov chain transits through stationary states dependent upon the input x.

$$P = \begin{bmatrix} p_{S_0S_0} & p_{S_0S_1} & p_{S_0S_2} \\ p_{S_1S_0} & p_{S_1S_1} & p_{S_1S_2} \\ p_{S_2S_0} & p_{S_2S_1} & p_{S_2S_2} \end{bmatrix} \tag{2.27}$$

Practical implementations of Markov chains are, by necessity of finite time, forced to place limits on the time that the 'fixed' probability is able to remain so. The limit comes from the noise sensitivity of the measurement system. The finite state machine illustrated in Figure 2.3 has noise corresponding to the measurement of x.

Asymptotically mean stationary (AMS) provides $\bar{P}$, a stationary mean of P, which is a probability measure under which the *noise is not a problem* and the Markov chain is stationary.

$$\bar{P}(G) = \lim_{n \longrightarrow \infty} \frac{1}{n} \sum_{n-1}^{k=0} P(T^{-k}G) \tag{2.28}$$

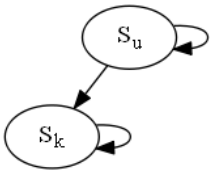
The smallest probability measure that could be achieved, is dependent upon an AMS source $\{\chi_n\}$. $\sigma(\chi_n, \chi_{n+1}, \chi_{n+2}, ...)$ denote the σ-field generated by the random variable $\chi_n, ...$, that is, the smallest σ-field with respect to which all these random variables are measurable. The finite state machine in Figure 2.3 only transitions at a change in x, meaning for that FSM to be stationary, the digital circuit needs to check x at a known periodic sampling rate.

2 State Markov Chain - Known or Unknown

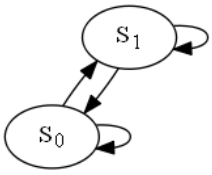
A Markov chain represents a system with measurable states.[36] A 2 state Markov chain has 2 measurable states, S_0 or S_1. The Markov chain state space experiences a reset event when the knowledge about the system transitions from S_u *unknown* to S_k *known*.

The set of states for the Markov chain is always $\Omega = \{S_0, S_1\}$ independent of the knowledge the observer has about the current state. An identifying property of a Markov chain is that the transition from current state to the next is independent of the prior history and only depends upon the known state.[36] The changing of states is governed by a fixed probability $P(S, \cdot)$ of transition between states taken from the finite set Ω.[36] A finite Markov chain has the property that the transition matrix P with dimensions of $\Omega \times \Omega$ is sufficient to describe it because the sequence of prior transitions leading up to the present state does not change the future states.[36]

The *transition matrix* P is assembled from the conditional distribution u_t that S_{t+1}



(a) 2 states and 3 edges.



(b) 2 states and 4 edges.

Figure 2.4: A Markov chain with 2 states and 3 edges transitions once, from S_u unknown to S_k known. The transition occurs instantaneously when the first measurement is complete. A Markov chain with two states and four edges is able to transition multiple times between known states S_0 and S_1.

will equal another state given knowledge of the current state.[36] That is, u_t is the probability that the present state at time t will transition to a new state at time $t+1$.[36] For the graph shown in Figure 2.4a the transition matrix depends on ρ, a variable which represents the long term constant state.[37]

$$P = \begin{pmatrix} P(S_u, S_u) & P(S_u, S_k) \\ P(S_k, S_u) & P(S_k, S_k) \end{pmatrix} = \begin{pmatrix} 1-\rho & \rho \\ 0 & 1 \end{pmatrix} \tag{2.29}$$

For the graph shown in Figure 2.4b the transition matrix depends on e, an independent variable which is a function of the *immediate* environment.

$$P = \begin{pmatrix} P(S_0, S_0) & P(S_0, S_1) \\ P(S_1, S_0) & P(S_1, S_1) \end{pmatrix} = \begin{pmatrix} 1-e & e \\ e & 1-e \end{pmatrix} \tag{2.30}$$

Figure 2.4a, with states $\{S_u, S_k\}$ is **not irreducible** because it has an entry $P(S_k, S_u)$ that is 0.[36] Figure 2.4b with states $\{S_0, S_1\}$ is **irreducible** because all entries in P are greater than 0.[36] Figure 2.4b is a **periodic** chain because from any starting position x there is a set of time measurements $\mathcal{T}(x) := \{t \geq 1 : P^t(x, x) > 0\}$ when it is possible to return to position x.[36]

An **aperiodic** chain is one where the greatest common divisor (GCD) between the

sets of possible periods is equal to 1.[36] Figure 2.4b is not aperiodic, for arbitrary values of e, because for some of the values, the probability of remaining in a state or transiting to another state is unequal, and in those cases the GCD is greater than 1.[36]

The values of the transition matrix P is independent of the state the observation begins from, because as $t \to \infty$ the probability distribution values π will converge to the stationary distribution $\pi = \pi P$.[36] This guarantee of convergence is *only* true for Figure 2.4b with value of $e = 1/2$ because that Markov chain is both **irreducible** and **aperiodic**, however, for any nonzero value of e, the entries of the transition matrix will be non-zero.

2.5.3 Ergodic Hierarchy Applied to Known Finite Markov Chains

The theory of the ergodic hierarchy describes how the state space of physical systems interact.[38] If a system is ergodic (non-ergodic is also a choice), the highest degree of interoperability comes from a Bernoulli system (B-system), where the space mean and time mean are equal almost everywhere except possibly on a set of measure zero.[39] There are multiple levels of ergodicity, with a dynamical system labeled as simply ergodic if it is guaranteed to transit through each state.

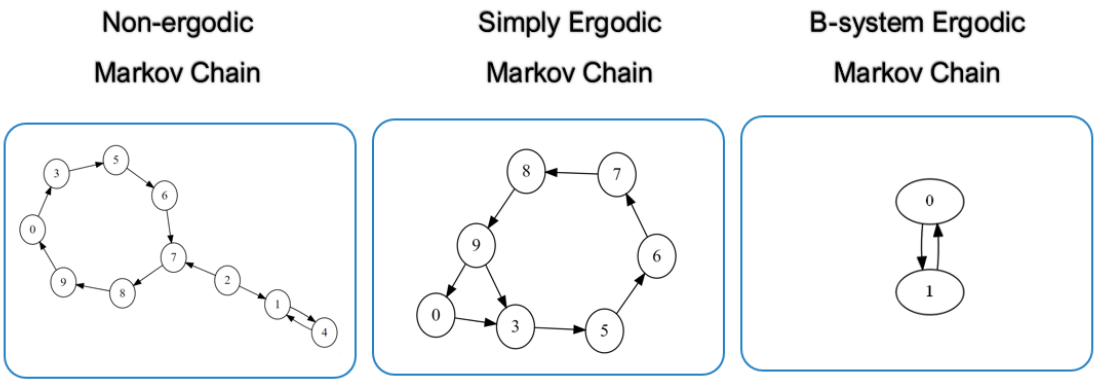


Figure 2.5: A non-ergodic Markov Chain and two Markov Chains with different ergodicity.

Two of the levels of Ergodic Hierarchy are shown in Figure 2.5. The simply ergodic system shown in Figure 2.5 is guaranteed to eventually transit to all other states after enough observations of state transitions are made. The relation between the two levels of ergodic hierarchy are that a B-system is also classifiable as simply ergodic, however, a simply ergodic Markov Chain may not have sufficient states available to be reclassified as a B-system.

$$\text{Simply Ergodic} \supset \text{B-System Ergodic} \tag{2.31}$$

None of the examples of Markov chain's in Figure 2.5 are shown with an indicator of time. Typically that is because the depiction of a Markov Chain is a capture of the transition matrix P. However, when a Markov Chain is irreducible, that action does make an assumption about the ability to maintain a probability measure under which noise is not a problem. The example of a non-ergodic Markov chain is shown without any indicator of time, meaning it has implicit stationary states. That means that for all space and time it would have the same states and they will not change. The ability of an observer to explore those states depends on what the observer's initial state is.

2.6 Markov Chain Mixing Time

A probability distribution π which meets the requirement $\pi = \pi P$ is called a stationary distribution of the Markov chain.[36] If the starting distribution $u_0 = \pi$, the chain is started in a stationary distribution, an all distributions for time t after will be a stationary distribution $u_t = \pi$ for all $t \geq 0$.[36]

The **hitting time** τ_x is the first time at which a chain visits a state x.[36]

$$\tau_x : \ \min\{t \geq 0 : S_t = x\} \tag{2.32}$$

Once the chain has progressed to the second state at a positive time, there is a chance it could have remained at the same state.[36] In that case, the **first return time** τ_x^+ is equal to 1.[36]

$$\tau_x^+ := \min\{t \geq 1 : S_t = x\} \tag{2.33}$$

A stationary distribution has the probability measure $\pi(y)$ for each y in Ω.[36]

$$\pi(y) = \frac{\deg(y)}{2|E|} \tag{2.34}$$

$\pi(y)$ is dependent on the $\deg(y)$ number of degrees emitting from the state and $|E|$ the total number of edges in the graph.[36]

The convergence theorem as explained by Levin, Peres and Wilmer relates the long-term fractions of time the finite irreducible aperiodic Markov chain spends in each state to the chain's stationary distribution.[36] For Figure 2.4b the stationary distribution is:

$$P = \begin{pmatrix} P(S_0, S_0) & P(S_0, S_1) \\ P(S_1, S_0) & P(S_1, S_1) \end{pmatrix} = \begin{pmatrix} 2/4 & 2/4 \\ 2/4 & 2/4 \end{pmatrix} = \begin{pmatrix} 1/2 & 1/2 \\ 1/2 & 1/2 \end{pmatrix} \tag{2.35}$$

The stationary distribution with the non-reduced number of degrees and edges for the Markov chain shows how a complete state exploration would require four measurements to take place to gain complete knowledge of the Markov chain.

Knowing the stationary distribution of a finite Markov chain explains how the 1-bit computer energy use changes depending on its environment. For instance, if the environment variable e is $1/2$ the energy use should be $1/2$ of the total possible energy use for the computer, circuit or neural system. If the energy use is not $1/2$ of total possible energy use, then either e or ρ are not precisely known, and the measurable information is less than the maximum available from the immediate environment. Since

ρ is a variable dependent on the fraction of macrostates above the threshold of the 1-bit computer, it is already known during the construction of the 1-bit computer and the missing information can be attributed to the active environment in the variable e.

Recording the states and transitions of a Markov chain from an initial state A is a way to find the probability distribution of states $\theta(A)$ from that initial state A.[36] Additional probability distributions can be gathered on Ω, and the **mixing time** can be calculated from the total variation distance between the two distributions.[36] The mixing time is the distance, or number of transitions, the initialization into the initial state is from the stationary state.[36]

$$||\theta - \nu||_{TV} = \max_{A \subset \Omega} |\theta(A) - \nu(A)| \tag{2.36}$$

The 1-bit computer has a mixing time of 2 (2 transitions) if $e = 1/2$, because for any of the states it can start in, it will take at most 2 transitions for the measured probability distribution to return to the stationary distribution.

2.7 Hierarchical Markov Chain

When a 2-state Markov chain is stationary, but, the 2 stationary symbols are represented further by additional driven symbols, the Markov chain is only 1st order stationary over a long time period, it has a table of transition probabilities P_{1st}, shown in equation 2.37. The 2nd, 3rd, etc... additional sub-machines the original stationary Markov chain is made from, are driven Markov chains that rely on a the periodicity of the symbols composing the stationary machine.

$$P_{1st} = \begin{bmatrix} p_{00} & p_{01} \\ p_{10} & p_{11} \end{bmatrix} \tag{2.37}$$

For example, the 2nd order Markov chain has a table of transition probabilities P_{2nd}, shown in equation 2.38. The measurement sum of the number of events in P_{1st} quadrant element p_{00} is equal to the sum of the elements in the upper left quadrant in P_{2nd}.

$$P_{2nd} = \begin{bmatrix} p_{0000} & p_{0001} & p_{0100} & p_{0101} \\ p_{0010} & p_{0011} & p_{0110} & p_{0111} \\ p_{1000} & p_{1001} & p_{1100} & p_{1101} \\ p_{1010} & p_{1011} & p_{1110} & p_{1111} \end{bmatrix} \tag{2.38}$$

This is shown in equation 2.39.

$$p_{00} = p_{0000} + p_{0001} + p_{0010} + p_{0011} \tag{2.39}$$

This is beneficial with a spiking channel, because, it means that receiving a spike related to the event with probability p_{00}, is equivalent to receiving a spike on the 4, 2nd order events. This results in the ability to transmit multiple 2nd order events with a single 1st order event.

Chapter 3

Connectivity in Neuromorphic Systems

Neural systems models require computation elements, and mechanisms for moving information between among the elements. Pulse mode communication, either in link form or in neural net form have the potential to create circuits with energy costs per effective bit of communication that is very energy efficient, but is still far from thermodynamic limits. The human brain and human DNA assembly show very high effective efficiency. Uneven fractional connectivity is the only practical CMOS interconnect strategy able to meet demands for widest range energy use. The consequence of uneven fractional connectivity allows the system to trade between energy use and time to communicate as the changing inputs demand. Biological systems like the brain also demonstrate hierarchical uneven fractional connectivity.

Neuromorphic systems are the class of integrated circuits which are inspired by biological models of the brain and central nervous system. Since the original use of the term by Carver Mead, to describe electronic circuits with a direct mimicry of biological nervous system functions, it has grown to include digital implementations as well.[40]

This work focuses on the spiking nature of biological systems, and strives for biological mimicry down to the level of spike based computation and communication, but not to the level of individual neuron channel modeling as exemplified in large scale human brain modeling systems.[41],[42]

Biological systems are formed through two stages. First, the prototype lifeform is assembled from stem cell DNA into primitive constituents, the cells, which will eventually multiply and differentiate into multicellular organisms.[43] The result of this first stage is a variety of rather specialist biological cellular 'computers'. Secondly, the neuron cells, generally held to be responsible for intelligence, continue to change their operation, predominately through modifications of the interconnect weights between the multiple neural cells.[8] The second state of biological system formation is much less broad in the application of its changes.

The first stage of biological intelligence development is concerned with the transformation of a cell into a multicellular species specific intelligence. With a broad enough definition of intelligence, it is already possible to classify the future intelligence of the single cell, by only knowing what the DNA inside of it contains. For instance, no one expects a single embryo cell from an ostrich to turn into an octopus. If the intelligence of the ostrich is only contained to the biological form of an ostrich, then having the biological form of an ostrich is enough to classify the being as intelligent, within range of any other living ostrich.

This dissertation is concerned with providing a silicon alternative to the first stage of biological intelligence development, so that a minimal level of intelligent computation could be expected to be output from the communicating sensor, that is an artificial spiking neural network with preexisting connectivity and fixed size of computation and communication nodes. This work is inspired by software artificial neural networks, which are often modeled with real-valued floating point numbers. The following section 3.1

addresses the disconnect between dominant software based models for neural networks and questions how energy used for a hardware based approach, with pulse mode links, would vary with different types of fixed connectivity between the neurons that compose the 'cells' of the hardware circuit.

3.1 Neuromorphic System Architecture

A neural net is an arrangement of computation nodes (cores) connected through synaptic weights representative of the memory of the net.[44],[45] The computation nodes might correspond to an analog model of a neuron, the biological primitive used to represent a nerve cell, or they might refer to a digital computer capable of running a simulation of the model.[46], [47], [48] Any neuron contains at minimum a **soma** body that is the compute unit, input signals that travel from **dendrites** and outputs signals onto the **axon** which tangles with the dendrites of other neurons giving rise to communication.[47] Any CMOS implementation is a trade-off between compute time, energy and area. [49], [50], [51] The architecture selection determines how closely the design will match to the biological model of a neuron cell and how much area will be dedicated to a single neuron.[41] Routing decides how communication is emulated across the net. Physical connectivity determines the interconnect length necessary for the system and the energy cost to communicate between cores.

The design of a neuromorphic ASIC can generally be classified into one of three different architectures. The selection between these 3 architectures influences the overall area of a chip dedicated to interconnect.

Fully Dedicated implies physical correspondence between the neuroscience model of a neural net and the layout of the system. The interconnect demands are high for this architecture, as each compute node has a dedicated axon used to communicate to

all of the receiving neurons. **Shared Axon** utilizes a digital bus and requires routing to represent multiple axons. The digital bus may be routed in a grid or tree like structure, and possible routing choices are discussed in 3.2. **Shared Synapse** stores all of the presynaptic spikes in a RAM for the neuron to receive and utilizes a digital bus and a scheduler to maintain a concept of time.

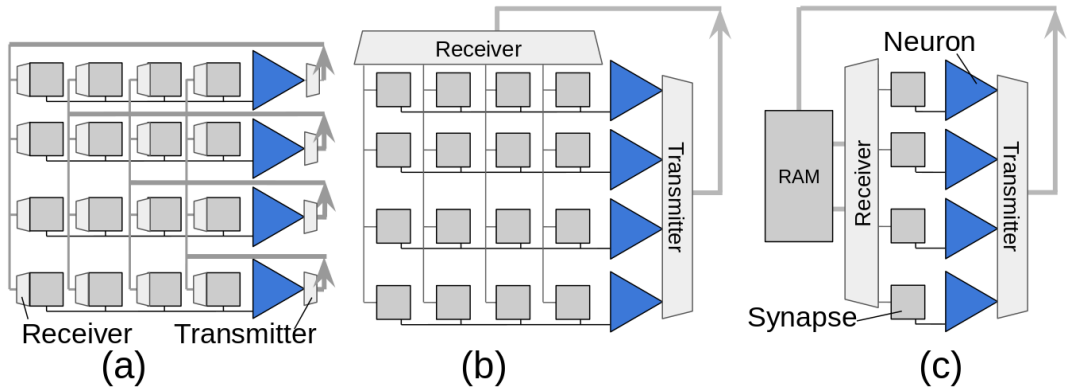


Figure 3.1: (a) Fully Dedicated means each neuron has a dedicated axon. (b) Shared Axon implements a routing scheme to communicate between neurons. (c) Shared synapse trades synapse area for a time multiplexed neuron core.

The architecture choice is dependent to some extent on the selection of digital or analog neurons. A comprehensive treatment of the pro's and con's of digital or analog neurons is found in [19], essentially it becomes a question of tolerable signal to noise. The area used for the neuron changes with the decision to use analog or digital neurons. Mixed-signal design incorporates both analog and digital components. Taking mixed-signal design to a further blended design style, we utilize digital analog neurons following a verification and modeling schema defined by Horowitz.[52] While this section focuses on interconnect choices, the selection of a neuron unit is covered in Section 4 and Section 6.

The A_{FD}, area cost per fully dedicated neuron core, is directly equal to the on-chip implementation of the core dependent on the process choice and type of neuron core

selected.

A_{SA}, area cost per shared axon neuron core, is similar to the area cost for A_{FD}, with an additional overhead for routing blocks within the design.

The shared synapse model permits the area dedicated to a single neuron computation core to be shared between multiple synapses and enables an effective area (A_{SS}) of the neuron core area (A_N) divided by the number of synapses (N_{SS}) stored in a RAM.

3.2 Neuromorphic System Routing

Routing permits variable configuration of the hardware. Many recently developed neuromorphic systems take advantage of Address Event Representation (AER), which enables the connectivity of the net to be reconfigured at will. AER is a digital signaling method, and while it is spike based in that conceptually the neuron ID's and timestamps of events that are sent through the router represent spikes, the underlying signaling is binary. The routing schemes shown in Figure 3.2 are all implemented using AER, and offer different trade-offs for energy, depending on the connectivity model being implemented on the system.

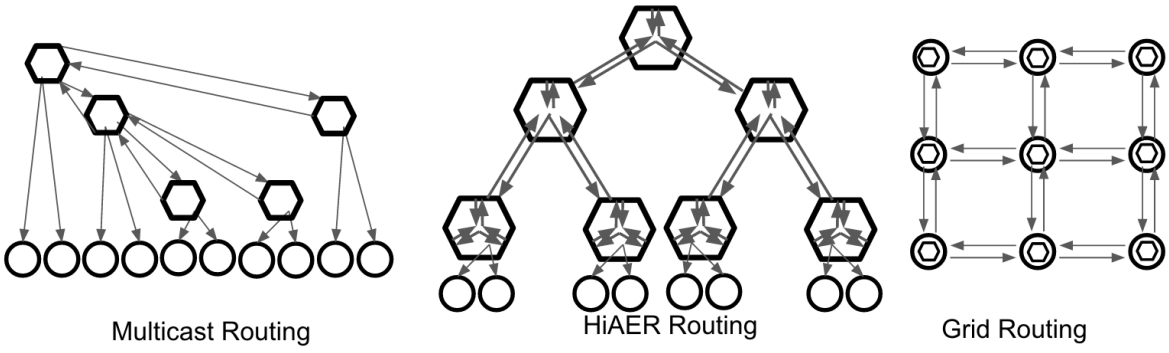


Figure 3.2: Multicast routing[1] has an average energy per spike per link of 14nJ, and prevents deadlock. HiAER routing [2],[3] builds packets destined to arrive before their intended spike time. Grid (Mesh) routing passes spikes between neighborhoods.

Grid routing requires a presynaptic spike to route through neighbors, at times requiring multiple hops from one chip to the next before reaching the chip which contains its postsynaptic neuron connection. The HiAER routing scheme incorporates the intended delay into the routing decisions and builds packets of spikes specially arranged to make sure the spikes arrive at their destination before the intended delay for the spike. Leaf routers, shown as empty hexagons in Figure 3.2, are primarily used for communicating to neuron cores, while the upper levels of the tree, shown as hexagons containing arrows in Figure 3.2, are used to communicate between routers. Multicast routing eliminates the risk of deadlock and reduces traffic through the upper levels of the hierarchy, by sending transmissions down to the lower levels as soon as the configuration is possible. The quasi-delay insensitive asynchronous strategy used by Multicast relies on handshaking to transmit large packets of spike information. The time-of-flight communication between the neurons is not equal for all of the routing choices. It should be separately considered that though the routing options reduce area, they do incur additional time delays.

Another option for routing spikes from one core to the next is to use a synapse and core, to directly forward a spike from one neuron to the next. Forwarding is not typically used, but because the pulse-mode links are able to quickly activate and send short messages, while otherwise not using power, it is used for the connectivity, energy and area trade-offs with fully dedicated architectures. Additionally, since pulse mode links are built from self-resetting domino logic, which is an event based signaling strategy, it is not necessary to maintain an overhead with a digital AER bus, and instead spikes can be transmitted with event signals. The self-resetting domino logic gates are used as the basis for the mixed-signal neuron architecture used later in Section 4 for communication and 6 for computation.

3.3 Connectivity Models

The physical connectivity models are resource constrained by on chip limitations. In the fully dedicated architecture a single neuron core can only support so many physical connections before the effective area dedicated to a core must increase. Likewise, excessive time-multiplexing in the shared synapse model could create locations of intense switching activity at particular compute cores. The shared axon model is also susceptible to long time delays, if routers in the upper levels of the tree structures become congested. This is modeled by constraining capacity for shared axon and shared synapse models to be no greater than $\frac{1}{2}$.

A biological system considers the time-of-flight for a spike to travel from one neuron through the axon delay (10-20ms), the axon branches (1-.5ms), across a synapse and into the dendrite structure of the postsynaptic neuron (100-50μs).[3] The connectivity of biological systems consists of single neurons which directly connect to a limited number of neighbors plus a few long distance cores. A biological system achieves a flexible configuration by employing some synapses as 'pass-through' connections, responsible for forwarding spikes from one neuron to the next.

In hardware based systems it is desirable, from the perspective of the programmer, for the connectivity to be fully programmable. This is achievable using any of the three routing choices presented in section 3.2. Selecting a reduced connectivity model leads to restrictions on the possible implementations available to the programmer. There are restricted models which would be suitable for implementation with reduced connectivity, for instance, the classes of neural nets which rely on small world models and associative memories, would be an ideal fit for reduced connectivity models.[53] These small world models indicate non-symmetric weights and non-symmetric local connectivity are favored for higher capacity memories in comparison to symmetric weights and connectivity.

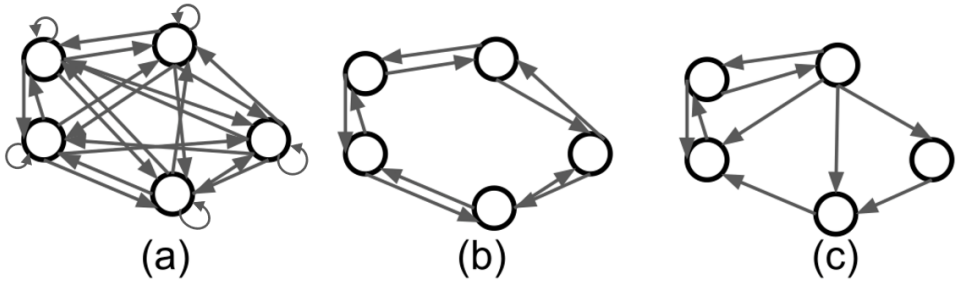


Figure 3.3: A soma (represented by a circle) has an axon dendrite structure with directional connectivity (indicated by lines and arrows). (a) Fully Connected (b) Even Fractional Connectivity (c) Uneven Fractional Connectivity

The connectivity of a net is described using N to represent the total number of neurons and M is the the number of connections per neuron. The **fully connected model** is one in which $M = N$. An **even fractional connectivity** model is a system in which $M < N$ and M is the same value for each neuron in the system. The selection of how many connections to dedicate to neighboring neurons and how many to dedicate to longer range communication is a trade-off, where the fan-in supported by a single link with a hardware implementation is limited by Rent's rule, which is described next.

Rent's rule, when applied to an integrated circuit states $T = tG^{p_i}$, where T is the number of terminals emerging from the area on the chip under question, t is the average number of terminals required by a single gate and p_i is the internal Rent exponent corresponding to the degree of placement optimization exhibited by the functional unit occupying the area under question.[54] For 1-input, 1-output logic gates, the value of $t = 2$, because the directional of the wires is not a factor in the emergence of the wires from the area they occupy. To solve for the value T, the values of G and p_i are referenced for published results from a digital neuron core, which states $G = 1272$, for the number of gates.[55] The values of p_i are based on prior works that for estimates of optimally placed microprocessors state $p_i = 0.21$,[56] up to a nominal value of 0.6 for an FPGA architecture.

p_i	T
0.21	4
0.5	35
0.6	72

Table 3.1: $G = 1272$, Number of Terminals for p_i

In the model used to estimate the connectivity of the pulse-mode links, it is assumed that there is some flexibility as to the number of terminals emerging from a neuron core, and this is used to preferentially select $p_i = 0.6$ with 72 terminals. In fact, when a single self-resetting domino logic gate is selected as a neuron model, increasing the number of terminals to what is declared a single neuron is limited by the desired timing constraints for how long the neuron cell should take to emit an event (Section 7). The number of terminals can increase as long as the timing constraint is met.

An **uneven fractional connectivity** model is a system in which each neuron has a specific connectivity. Both of the fractionally connected systems exhibit a native time-of-flight requirement for the communication of some neurons to other neurons in the system. This is observed in the trade-off by noting that some connections have a higher energy cost, because the energy cost is associated with the interconnect length traveled between two cores.

3.4 Interconnect Energy per Bit

The interconnect energy per bit considers the cost to send a single bit across distance d. The calculation assumes that the bit travels some number of hops at a cost of E_L energy across the long distance links, then as far as possible across medium length links for energy E_M and finally completes the journey across the shortest links at a cost of E_S.[6] The assumption that it is best to go as far as possible with long distance and

medium distance links means it is better to take a modulo of the distance d against the length of the links L_L and L_M.

$$E(d) = E_L \left\lfloor \frac{d}{L_L} \right\rfloor + E_M \left\lfloor \frac{d \ \% \ L_L}{L_M} \right\rfloor + E_S \left\lfloor \frac{(d \ \% \ L_L) \ \% \ L_M}{L_S} \right\rfloor \tag{3.1}$$

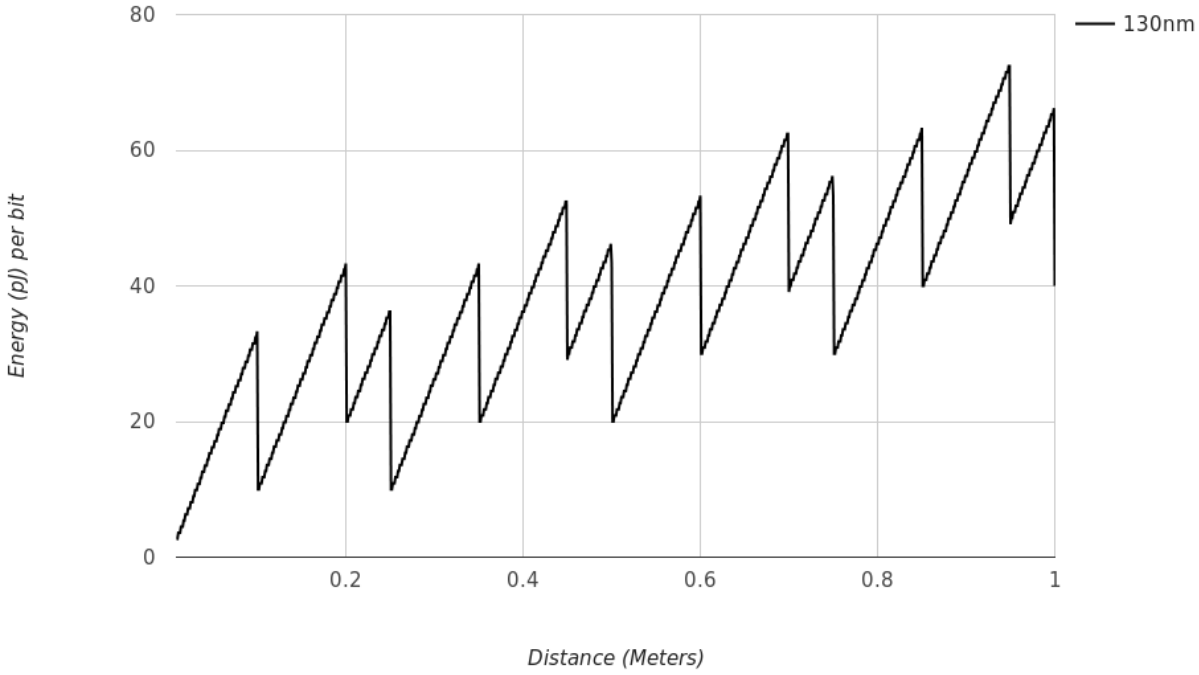


Figure 3.4: Energy to transfer a single bit across a distance d.

The energy figure is dependent on the routing scheme selected. For this hypothetical routing figure, it is assumed the network is fractionally connected network and preferentially selects long distance connections, either on or off chip.

From the plot shown in Figure 3.4, at times the energy to transfer a bit for a long distance is preferable to transferring the bit via many small hops. The additional area overhead for a single long distance transmitter-receiver pair on a chip is $0.02mm^2$ at 130nm, and the potential energy savings for a system like TrueNorth[1], which has an

[1]IBM TrueNorth [57] is capable of representing a million neurons in one chip and is an asynchronous design.

off-chip energy per bit cost of 26pJ to transfer between two adjacent cores is significant. The energy figure for the area and energy per bit performance figures for the pulse-mode link, with a data rate of 5Gbps, would improve beyond 10pJ if implemented at a smaller process node, as studied in [58]. A comparison between Pulse-Mode Links, and the communication schema of existing neuromorphic ASIC's is shown in Table 3.2, which also contains additional energy estimates for biological systems, to provide perspective on alternative mixed communication/computation strategy energy costs.

Name	Energy per Bit	Process	System
Pulse-Mode Link	10pJ	130nm	Mixed Signal
TrueNorth	26pJ	28nm	Digital
HiAER-IFAT*	59pJ*	130nm	Digital
Single Pulse Gate Firing	50fJ	130nm	Mixed Signal
Single Human Neuron Firing	376pJ		Biological
Human Brain Activity Per Neuron	20fJ		Biological
DNA assembly (per step**)	0.2aJ		Biological

Table 3.2: Energy per bit comparison of chip-to-chip interconnects for large scale neuromorphic systems. Additional energy estimates for neuron circuit pulse gate and biological systems included below double line. *HiAER-IFAT Energy is inclusive of computation cost. **DNA Step is replication, transcription and translation.

The existing software implementations of neural networks follow an even fractional connectivity model.[14], [59], [60] The exploration of a hardware model utilizing pulse-mode links indicates that a lower energy-per-bit is achievable with a uneven fractional connectivity model. The pulse mode link has low power figures at longer distances, and because of this, there are unique uneven fractional architectures which provide lower power figures for greater degrees of connectivity.

The power figures for pulse mode links compare favorably with the energy needs of biological systems, shown in Table 3.2, where they are estimated using references focused on the lower energy limits for computation to happen. The limiting demands of a brain-

like circuit are easy to calculate: 10^{11} neurons, 10^4 synapses on average, with an average synaptic activity of 1Hz would require synaptic firing to consume only 20fJ to fit in a 20W emulated brain. This energy cost is still orders-of-magnitude higher than the fundamental limits of computation energy, the Landauer limit, which is $k_bT\ \ ln(2)$ when the environment is a closed system with a 1-bit computer.[4],[61]

Biological neurons have a dissipative energy around $10^{11}k_bT$ per event due to their macroscopic size being quite large compared to the size of molecules and chemicals they are made with.[61] This provides an upper bound for the limited source of energy for biological systems which was referred to in the thesis statement. DNA assembly takes energy closer to Landauer limit, with an estimate between $20 - 100\, k_bT$ per step, which when estimated at environment temperature of $273°K$ is about 0.2aJ. DNA assembly of biological cells is both reversible and irreversible depending on the timescale of observation.

Achieving energy use close to 20fJ becomes within range of uneven fractional connectivity models, particularly at processes from $130nm$ and below, when mixed-signal design is feasible as opposed to purely digital implementations at smaller nodes.[62] Energy estimates taken from pulse gates used to build mixed signal neuron circuits in Section 7 are 50fJ, and since pulse gates continue to operate reliably with lower than nominal power, an energy use close to 20fJ is achievable.

Chapter 4

Communication with Spike Trains

Digital binary communication and computation rely on the notion of the presence of a binary bit (thus 0 or 1) captured at a defined synchronization interval (the clock). The unit of measurement for binary is the symbol, the bit. Alternative computation schemes offer the promise of symbol based communication and computation with options for an expanded unit of measurement, the symbol, which can be larger than a digital bit. The non-binary digital spiking communication and computation channel needs to operate using the same core modules as a classical communication channel, with an information source, a transmitter, a media channel, and a receiver. Here we review how to construct an information source.

Existing Q-ary communication channels are implemented using parallel binary bits, and, since they require multiple measurements of the many bits to achieve Q-ary signaling, they are not equivalent to a spiking channel.[32] The communication principles reviewed here are for digital non-binary signals, that can communicate multi-bit size symbols as single measurement events. Digital non-binary signals are implemented using spikes, which are actively studied in neuromorphic systems. An overview of the existing encoding patterns for spikes is presented, primarily to point out that a single spike

trace cant be immediately converted to a binary pattern of zeros and ones. Digital non-binary communication can use entropy estimation and mutual information to confirm transmission of data.

4.1 Introduction to Spiking Channel

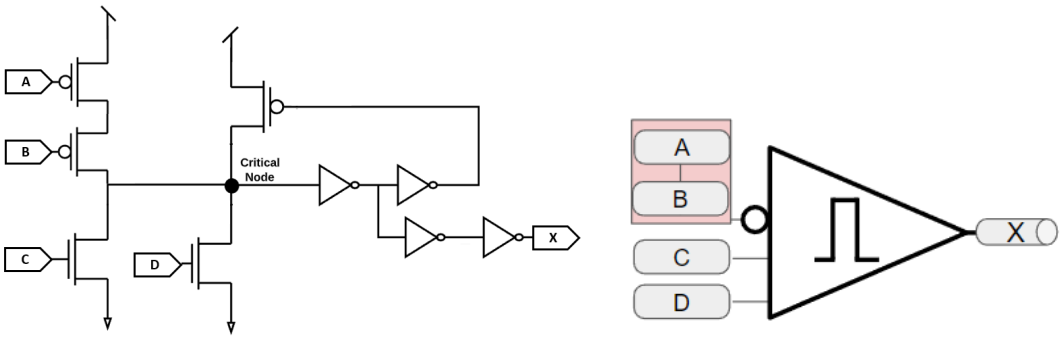


Figure 4.1: A pulse gate is the fundamental cell used for constructing a spiking channel. The transistor schematic of a pulse gate, shown on the **left**, has two series pull-up inputs (A and B) and two parallel pull-down inputs (C and D). The voltage levels on A,B,C and D control the output at X which will emit spikes unless A,B,C and D are able to completely overwhelm the inverter loop and force a constant high voltage at output X. The symbol for the transistor schematic is shown to the **right**, denotes series connections by grouping A and B into a single box, while parallel is a direct line into the gate. Pull-up connections have a circle before the input and pull-down connections are straight lines into the gate.

A spiking channel describes an interacting group of environment sensors and their neurons which transmit a coordinated signal across multiple nerves describing the state of the environment.[24] For a neuromorphic communication channel, the multiple nerve cells with their connected dendrites, somas and axons represent a single spiking channel when they share the same environment sensors and have some degree of interconnectivity. A neuromorphic (spiking) channel is composed of passive resistor capacitor effects moderated through the physical properties of the wires which interconnect the active response units that are made from CMOS transistors, assembled into pulse gates as in

Figure 4.1.

4.2 Leaky Integrate and Fire Neuron Model

A leaky integrate and fire (LIF) is a analog spiking neuron model commonly used in neuromorphic systems as a threshold function.[63] A LIF neuron is a resistor R and capacitor C arranged in parallel, which are combined into a single variable, $\tau_m = RC$, the refractory period.[64] When a current $I(t)$ interacts with the circuit inputs, charge q accumulates on the capacitor. The model has an event threshold θ, a inherent resting membrane potential (voltage) u_{rest} and a time dependent membrane potential $u_i(t)$ which accumulates charge q until enough charge accumulates to cross θ at the firing time $t_i^{(f)}$. After the firing time, a refractory period τ_m occurs.

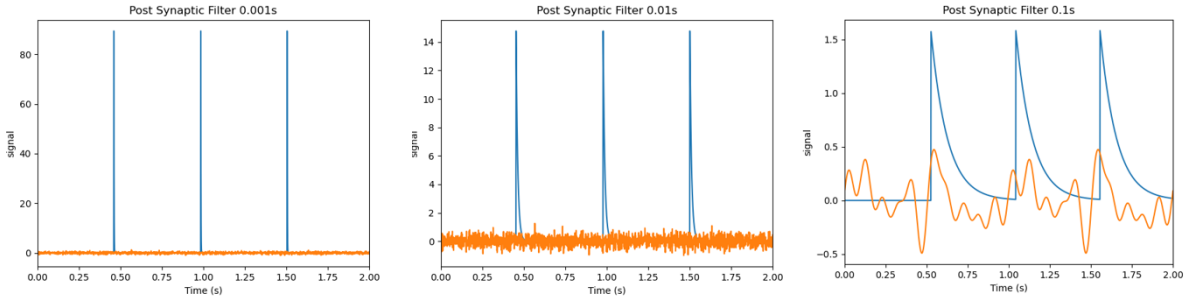


Figure 4.2: Three different postsynaptic filter connections show the same spike events with differing magnitude and slope of falling edge (Refractory period τ_m).

$$\tau_m \frac{du}{dt} = -[u(t) - u_{rest}] + RI(t) \tag{4.1}$$

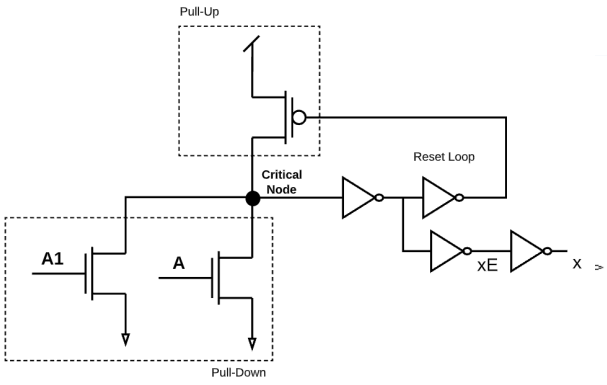
The refractory period, or 'time of inactivity' is controlled by RC.

$$u(t) - u_{rest} = \Delta u \exp\left(-\frac{t - t_0}{\tau_m}\right) \quad \text{for} \quad t > t_0 \tag{4.2}$$

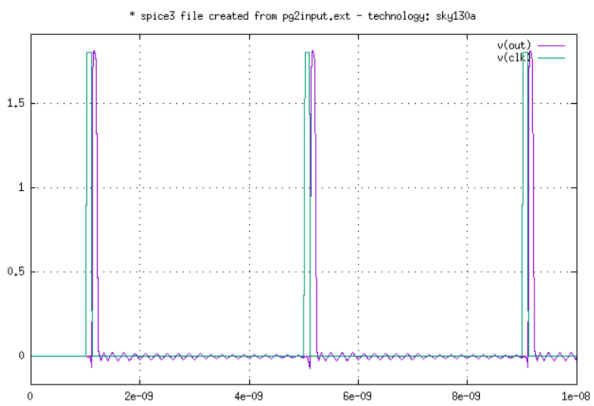
The refractory period only occurs after an event.[64] The refractory period can only occur in locations after the first measurement. A communication channel takes the input it receives and reproduces that input at the output of the channel.[18] A neural communication takes place through a synapse, which acts like a filter on the incoming continuous signal, and outputs onto other postsynaptic connections.[65]

Figure 4.2 shows how an input signal (orange line) is moderated into output spikes through different postsynaptic filters with a LIF model. Biological systems are a collection of different cells which output spikes at order-of-magnitude timing differences.[66] That range of timing differences eventually results in intelligent systems which are quantifiably measurable at human sensing timescales (millisecond >).[27] The synaptic filters are a detail which may be safely ignored when the orders-of-magnitude range of input is able to generate at minimum a single event.[67]

A LIF neuron circuit is shown in Figure 4.3a, where the relationship between the pull-



(a) Self-resetting domino logic circuit for emitting a spike event, oscillation of spike events or no event, dependent upon input A, A1. This logic gate with only input A is represented by the symbol shown in Figure 7.2 Circuit 1.



(b) Input to output

Figure 4.3: Circuit model for digital LIF neuron. (a) CMOS schematic of a neuron with two parallel environment sensors (labeled A1 or A). (b) Spike output (purple) of circuit model when input A or A1 experiences a spike event (green).

up and pull-down network defines the RC value, τ_m, for the critical node. When charge q accumulates at the critical node an event is generated as shown in Figure 4.3b.[68] The characteristics of the digital LIF neuron shown in Figure 6.4 are equivalent to software models, particularly when τ_m in the software simulation is minimally filtered as shown in the left plot in Figure 4.2.

The bundle of wires and pulse gates represent a single spiking channel, where each of the wires produces a temporal stream of spiking events called a spike train because the traveling groups of spikes resemble train cars moving together down a train track. The wires can represent neuron axons or dendrites. The dendrite is a wire when it is connecting between either an analog input from the environment, into the dendrite tree, that is the transistors with their drain connected to the critical node of a pulse gate. The spike trains are a communication because they are generally understood to move information from one physical location to another. When a wire is connected between a pulse gate output and the gate of a transistor with the drain connected to a critical node of another pulse gate, that wire is an axon, because, it is communicating an output between two event gates capable of emitting spikes.

A stationary computation process is defined as the Markov chain of the computer that describes the probable state of the computer as time goes to infinity. It works to communicate based on the influence of the environment on the probability of events occurring. Communication is how intelligence makes itself known. In the end, Shannons definition of entropy is a measure of how unpredictable a bit pattern is — as any easy to predict pattern cannot be information (since an intelligence could predict it). Entropy is based on trying to remove all simply predicable behavior from the bit pattern and minimally encoding it. Thus, to Shannon, random noise is not distinguishable from intelligent data. Entropy is an attempt to quantify this unpredictability.

When a message with systematic probabilities from a ‘dataset’ is used to make a

Markov chain, it is an emulated channel that with likelihood sends messages related to the original 'dataset' extracted probabilities. In a sense, such a machine is communicating, since observation of its output strings allow reconstruction of the probabilities.

Biological systems do not have an inherent unit of measure known to each other. However, there are species specific learning circuits present in mammals and birds, which are tuned to vocal variability of the species.[69] Anytime there is a communication there are two or more participants, communication to self is not communication. Communicating presence relies on common information, the relied upon and shared understanding of the symbols of the common alphabet.[70] To measure the communication using a measure of information, it is necessary to have an outcome agreed upon between the two participants.

4.3 Alphabet in communication

All communications use alphabets to compose a message.[70] In fact, alphabets are present in all communications, with the caveat that the specific alphabet in use might not be evident immediately.[71] Dataset's used for machine learning and artificial intelligence also use alphabets. Figure 4.4 shows a sample entry from a dataset, it is only a sample and does not represent the entire dataset. There is no obvious label for the sample.

```
This is a language and these are the 14 words.
```

Figure 4.4: An entry from a dataset.

If we wanted to know the information content of the sample, we could calculate its entropy using Shannon's entropy which is equal to the sum of the probabilities of receiving

each symbol in the entry multiplied by a constant K, the unit of measure.

$$H = -K \sum_{i=1}^{n} p_i \log_n p_i \tag{4.3}$$

In this dissertation the unit of measure is a positive integer, $|A_\chi|$, which is the number of elements in the set summed, to subscribe to the more recent notation used in information theory for communication channels as described by David Mackay in their early 2000's text book.[32]

With a known information source, a more general form of entropy is where $K = |A_\chi|$.[72] Looking at the sample in Figure 4.4, in a hypothetical situation where the sample contains all of the symbols from the information source, the sum of the probabilities is known to be 1. Now the only unknown is $|A_\chi|$, the number of symbols in the information source. Since it is hypothesized that all the symbols from the information source are observed, we can count the unique symbols in Figure 4.4 by writing the set $A = \{T, h, i, s, \ , a, l, n, g, u, e, d, t, r, 1, 4, w, o, d, .\}$ and discover that $|A_\chi| = 20$. Entropy is useful for calculating the information received over time, but the quick trick of counting the unique observed symbols does not consider time. It is not that important though if all we want is to get a small understanding of what the entropy means.

The entropy range that a sample of data can fall within is the number of unique elements, $|A_\chi|$, used to construct the *information source* for the communication channel the data was acquired from.

Counting the unique symbols from a sample of the data is a quick and practical way to estimate entropy, with a few assumptions.

1 Assume that the frequency of occurrence of each symbol is not equal.

2 Assume that one of the symbols will occur much more than all of the other symbols.

3 Assume that the number of uniquely counted symbols is discovering new symbols at a rate of less than $n \log_2(n)$, where n is the number of unique symbols already counted.

If these three assumptions can be validated for the sample of the dataset, then the entropy guess is reasonable within $\pm|A_\chi|/2$. To fully demonstrate that the guess is acceptable it is necessary to know the full size of the dataset the sample is taken from, but, if that is unknown, checking on the two assumptions will help validate the quick guess.

For example, the sample entry in Figure 4.4 belongs to a dataset with a known value, $|A_{simpleThis}| = 24$. The dataset is used to build an information source 'simpleThis'[1]. Its entropy $H_{maxForSimpleThis} = |A_{simpleThis}|$ is the maximum amount of information that a message from that information source could contain. The known value of 24 is not much different than the estimate of 20. It is only off by the value of 4, which is a comparatively small number given the original estimate was 20. But 4 is twice as much as the maximum quantity a symbol taken from a binary alphabet could contain.

A typical electronic communication happens via signaling conducted through wires with the choice between analog or digital, serial or parallel, synchronous or asynchronous being three of the first decisions necessary to understand a communication.[73] Serial or parallel refers to the number of simultaneous ports available. Synchronous or asynchronous is a choice between the use of a clock signal. Naturally having a large number of parallel wires seems ideal for more information, but, as the number of parallel wires grows physical limitations begin to impose on the system particularly for synchronous systems which need to maintain clock distribution. The introduction of a communication channel exposes the message to the noise of the channel. Serial or parallel signaling is affected by channel noise in different ways. The serial message is experiencing noise on each symbol sequentially and might also have effects caused by the prior symbols in the

[1]Details of the simpleThis dataset are in Section 8.4

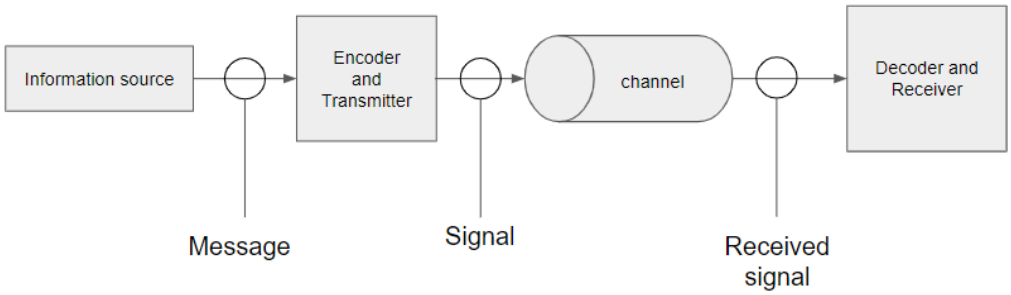


Figure 4.5: A message is composed from the symbols in the information source, then encoded to be resilient to the noise of the channel, transmitted through the channel as a signal, and received at a different place to be handled by the decoder and returned to the original message.

channel. The parallel signals experience noise across the number of parallel symbols at one point in time, and the act of placing multiple symbols into the same channel at once means they can interfere with each other.

A typical human communication happens via signaling of two wave detectors and emitters. (Like sight and sound) Some humans use more or less detectors and emitters, in the case of one wave, we might actually not be directly within contact with the other human. One may use a communication model like the one shown in Figure 4.5 which moves the information through a channel.[74] A communication model contains an information source to provide the alphabet and to act as a source for the statistics to find a 'good' encoding for the channel.

Conventionally, a 'good' communication channel was one that exactly replicated the message. That may not be the case under all circumstances, in particular when the communication channel is operating as an asynchronous link signaling in a general channel. "*Given the prevalent and practical synchronous solutions, why take another look at asynchronous link signaling?*"[2] Neuromorphic computing is asynchronous and draws inspiration for circuit design from practices observed in biological systems. When a bi-

[2]asynchronous means without a clock signal to synchronize.

ological system is studied to understand neural signaling, the inputs and responses are not understood in terms of decodable data. The representation used instead has a metric which labels a 'good' communication as one that elicits the desired response from the living system. In principle, a neural network is able to replicate any other function.[75] The neural connectivity of living systems is best described as uneven fractionally connectivity, with some neurons easily able to communicate with thousands of others nearly immediately as well as a few medium and long distance connections.[76] Uneven fractional connectivity opens up the range of possible solutions to support the boundary of the VLSI design space, where low energy, high information computation is feasible.[9]

The encoders presented in this dissertation take advantage of uneven fractional connectivity to encode data as a space-time communication for the purposes of neuromorphic communication. Small models of intelligent behavior can be described with probabilistic Markov chains or with cellular automata.[77] With a goal of recreating a neuromorphic communication channel, where 'good' is defined as eliciting the desired response, the creation of the channel depends on creation of an information source. To build an information source, it is critical to understand from what data the probabilistic table is gathered and how it was measured.[39] The information source is made by gathering data for the statistics of the intended communications. The gathered data is the *dataset*. For example, the information source for 'simpleThis' contains only two entries. By examining those entries it is possible to create a set of representative statistics and use that to communicate quickly.

4.4 Construction of an Information Source

An information source is made by first gathering a set of representative samples of the data intended to be sent over the channel, that is the dataset. For instance, in the

situation of analog telephone lines, sound samples are recorded and used, so that the designer is able to understand the range of frequencies the information source will need to support. Before you can build an information source it's necessary to go gather the data for the types of messages you want to send through the communication channel. An information source is a set of possible messages that can be sent through a communication channel. The set of messages comes from studying a dataset of possible messages for communication. For instance the telephone operators asked humans to speak into microphones so they could record their voices and discover the range of sounds that human voices need to communicate. The dataset is used to build the information source. The dataset measurements are reproducible from the information source.

The information source is a really interesting question when it comes to knowledge graphs or other large sets of structured data. When it comes to knowledge graphs, the information source is not as relevant because the knowledge graph itself is intended to be a curated set of data, sufficient to be used as an information source itself. Here the promise is of cleanly defined, and factually correct relationships.[78], [79]

The idea of an information source as the alphabet for the message is not really applicable, but, at its core, these large datasets still need the same descriptors, even if the majority of the data is only present in a binary alphabet. (Those binary alphabets could have been strung together to form larger sizes, and those larger alphabets may be useful, but we can't just know that from the start.) Their long term goal is to develop associations and correlations [80], in much the same way as an ideal symbol set should exist for the information source.

For a communication system the information source provides the alphabet; a set of symbols from which the message is composed for encoding. Binary is an example of an

information source which contains two symbols.

$$A_{binary} = \{0, 1\}$$

or

$$A_{binary} = \{n_1, n_2\}$$

An information source is the starting point of a communication system, as depicted in Figure 4.5. An information source is also the starting point of a computation system, shown in Figure 1.1. In a computer the information source is the memory hierarchy the programmer has access to. Typically that means it is a layered arrangement of bit sizes and access times.[81] The fastest access times are provided by a cache sitting physically close to the CPU, possibly sharing the same silicon die area.[82] After a series of caches with decreasing power needs and increasing bit size and latency the computer has a file storage system that contains solid state NAND memory or spinning magnetic disks. Finally, most computers have a internet connection and the *computer information source* might be considered to include data accessible over interfaces, like the high performance computing interface, InfiniBand.[83]

The difference between what makes a good information source for computers and what is good for a communication channel comes down to the statistics of the symbols.

The ideal information source best suited to a communication channel is one that detaches the statistics of the message from the statistics of the symbol.[70],[84] This results in an information source with equal probability of emitting $1/n$ of each symbol over time, an example is shown in the lower pattern of Figure 4.6.

Computers are not given the *ideal* information source of communication systems. In fact, a software engineer would like to have as much different data accessible at once

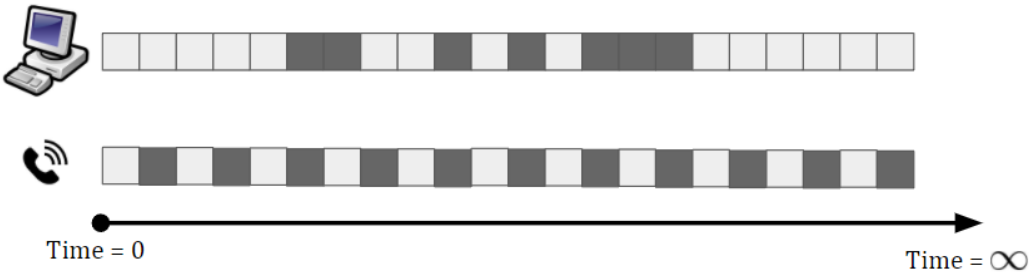


Figure 4.6: A 'best case' output stream of a computer is shown in the upper pattern where the result is not an equal partition of the two possible symbols. An ideal information source for a communication channel is shown in the lower pattern, where for every timestep the output value is the inverse of the preceding symbol.

(Shown in the upper pattern in Figure 4.6), a request that has caused computer engineers to build the memory hierarchy. The ideal information source for a computer programmer (the intelligence responsible for operating the tapes of the Turing machine) is something that contains as many different patterns of the symbols as possible. To understand, consider the subject of 'Turing Complete' which is a model held up by computer engineers of an ideal computer capable of simulating any other program[85], and also consider the preferred alphabet of noiseless binary digital computers is a language that is 2 symbols long. To simulate any other program, the Turing machine outputs on a hypothetical infinite tape for each timestep of the read-write head. The best behavior meeting 'any other program', by necessity means all of the combinations of the 2 binary symbols, not just the combinations that give a probability of an infinite tape with $n_1(1/2)$ and $n_2(1/2)$.

With two symbols it is possible to express anything, although to do so, it is necessary to begin assigning additional meaning to groups of the two binary symbols. Any complicated mathematical function or a look-up table of references and key-value pairs could be written. If there is a long enough block of the two symbols, by inspecting it, eventually we would find extremely unlikely combinations and patterns. An information

source with only two symbols is capable of representing anything representable[3].

It is acceptable to have a larger information source with more symbols. In fact, in some circumstances it is easier to complete a communication if there are more symbols since now the receiver does not waste time looking up meanings of blocks of binary and converting to the operating language. Decimal numbers and mathematical operators are examples of other symbols in other alphabets.

$$A_{decimalAddSub} = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, +, -, =, \}$$

Human spoken languages come with alphabets or groups of spoken symbols and there exists a universal phonetic alphabet, International Phonetic Alphabet (IPA).

It is easy to make an alphabet for a dataset for a language. Gather a number of samples n and as long as n is much greater than the support size S (By 2 or more orders-of-magnitude, as a guideline) the probability distribution function for the dataset will produce a representation of the language. DNA has 4 symbols.

$$A_{DNA} = \{A, C, T, G\}$$

Most languages in use for communication use small sets of symbols.[86] The selection of what property qualifies as a 'good' property for an information source is up to the designer, thus, in our case we choose to select a good information source as one which permits for highest entropy. There are many different alphabets and which one to use is completely the choice of the communication system designer.

An information source cannot contain only a single symbol because it would require infinite (or zero) energy to represent that symbol. The reason for this is that the mea-

[3]Apparently things in mathematics exist that are unrepresentable or at least uncomputable, for example, Kolmogorov Complexity

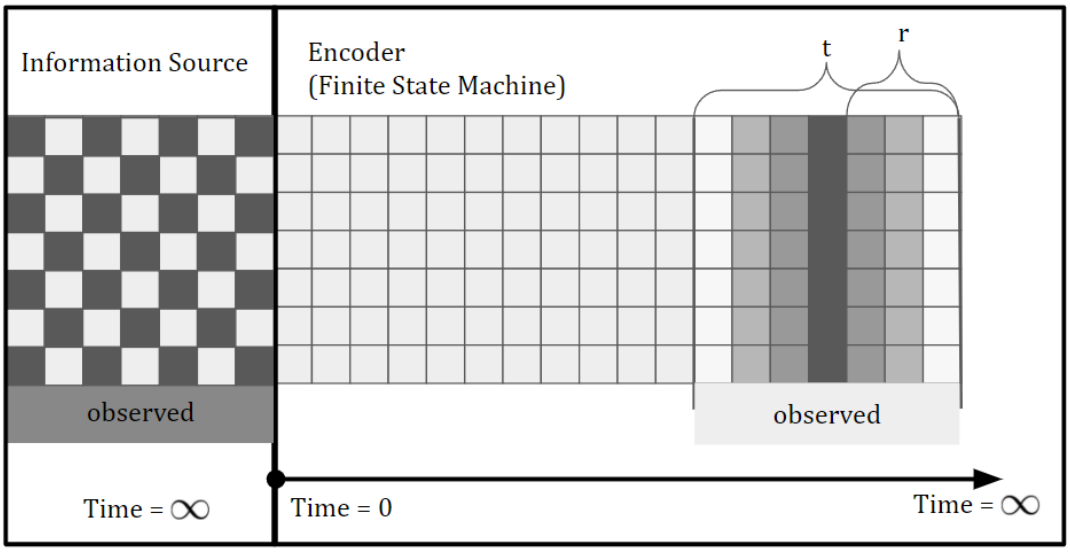


Figure 4.7: An ideal communication information source must emit an equal proportion of each symbol contained in the alphabet as time goes to infinity. The measurement used to build the information source is also the same one used to set a minimum measurement for a symbol. When a block of bits r is used to compose the message with spike or no spike corresponding to a binary code, *if we are able to know we have observed the first observation*, we can decode the message after $r+1$ time steps. Otherwise the spikes must be recorded for t, twice the duration of the message, as the FSM repeats the message an infinite number of times.

surement for the single symbol would need to take place in a finite length of time or in a finite place. Once the first measurement is made, there needs to be a place to store the symbol, but, since it is the only symbol that exists it would already be in the memory and the measurement is stuck taking forever, or, conversely, having never taken place at all since it cannot be communicated.

A good information source contains more than one symbol, beyond that the answer to what is a good information source depends on the dataset. For instance, one might inspect the dataset and conclude that the timescale being used to measure the levels is much smaller than is necessary given the minimum time observed that a level is held constant. In this case a good information source would be one that emits symbols at a rate within the measurement period.

Experimental evidence must be gathered using imperfect measurements, which introduce noise and uncertainty into the message. The information source used for the message has complete information of the source, because it must contain all of the possible symbols in the language. As the message is being encoded and transmitted not all of the content of the message is available at once. The encoder can continue to repeat the message using the same symbols from the information source. As long as a decoder is able to receive a representative block of encoded data it will have enough information to decode that data, despite not knowing precisely where the information begins and ends.

The necessity is that the *time of the information source to emit one symbol*, τ must become the minimum simulation time step, but the *time it takes to complete a message* r can be much longer.

$$\tau < r \tag{4.4}$$

It has a grid of time for each inter-computation channel with changes in the timescale as additional processing and measurement takes place. The time of T, which is the information source's time, should be considered equal to one so that the information content of the source is measured at equilibrium. The measurements of the data used to create the information source are correct and without error.

4.5 The Unit of Measure for Information

Measure refers to the act of using a tool to make a record of observable data. It is an idea from physics, which has spent much time discussing what exactly a measurement means. Generally a explanation of measure is *"coupling the value of a parameter y at one moment with the simultaneous value of a fluctuating parameter x of the system, in such a way, that from the value of y, we can draw conclusions about the value that x has at*

the moment of the measurement.", quoted from a translation of Szilard's 1929 paper "On The Decrease of Entropy in a Thermodynamic System By The Intervention of Intelligent Beings".[87] The unit of measurement is the smallest distinguishable mark on the tool, it is the *moment* Szilard is referring to.

The unit of measurement of a communication system should not matter, because the designers would have made the channel adequate to contain anything which needs to be expressed by the end users. But, in fact this assumption, *the unit of measurement is inconsequential*, is only possible because of a requirement that the states of the communication system be continuous in their transitions. (Appendix 2 of [18]) An analog signal is quantized through the Nyquist theorem which states that it is necessary to sample at twice the desired resolution to preserve the signal with the intended quality.[88] The sampling rate is a periodic signal, and, while it can begin at any point when the coupling parameter y is known to be constant, if the fluctuating parameter x is also a periodic signal whose rate of change is less than half the sampling rate, then instead of measuring x to have an accurate reflection of its true periodic signal, it will instead appear to be a rotating sequence of periodicity. Sampling at multiple rates can produce a distorted view of the data.[89] This becomes apparent when binary data is being sampled at a rate not periodic with the rate the data is switching at.

A measure can be ascribed to a probability distribution as well as a unit of measurement. In that case measures explain how randomly chaotic a system can become as it evolves in time.[38] Making a measurement in error and communicating it makes the meaning wrong. The separation between communication and the physical media of transmission means that defining an error as something which was produced by the measurement to form the message, is a type of error that is not handled by the error correction channel, because the error is part of the intelligence operating the channel. Noise-enhanced associative memories are a type of neural network which are able to

store exponentially more memories for recall than their noiseless counter-parts.[90] They achieve this improvement in error free recall by introducing internal error, up to a threshold, to their rate-coded neurons. The rate-coded neurons used in the noise-enhanced associative memories are a software implementation example of a non-binary digital code. Instead of transmitting the message through a binary alphabet, small groups of spike events are converted to integer's and the formerly analog output of an associative memory, which was originally shown to have linear memory capacity[91], until the addition of internal noise was added to the model.[90]

The solution is to encode the data for transmission in a way that the data measurement is redundant and repeated more than once. However, if the translation is analog to binary digital, there will still be a limit to how many bits are necessary to adequately capture the analog signal and duplicating the measurement will not change that. Expanding to analog to non-binary digital conversion will permit for a higher noise-enhanced associative memory capacity because the internal errors will allow for greater uncertainty in a measurement, as long as the internal error threshold is within tolerances defined by the overall symbol capacity of the integer rate coded network output.[90]

4.6 Interpretations of multi-wire spike trains as bit sequences

The primary goal of this overview is to demonstrate how the symbols communicated through a neural network are representable as distinct integers and are able to scale differently according to interpretation as population, rate or phase codes which are detailed in section 4.7. Binary digital data is represented by voltage levels on a single wire. In this dissertation, most of the inputs are non-binary, but they may be confused as binary,

because often 0's and 1's are used to represent **not spike** (0) or **spike** (1). A spiking channel has multiple (connected through non-linear elements) independent traces, which can be converted to integer symbols which are representable as binary codes. Any integer is representable by a binary code as long as the bit-space of the binary code is large enough to allow an integer of that size. As an example of this, decimal `7` in binary code with a bit-space of length 4 is `4'b0111`. In the bit-space of length 4, the maximum integer size is `15`.

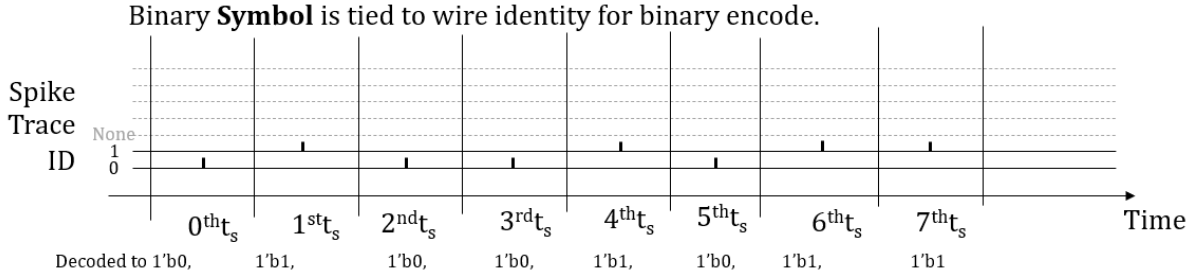


Figure 4.8: Conversion of a spike train to a binary code using the identity of the trace wire as a symbol responsible for emitting the spike.

Spikes are not necessarily only a binary language, and the conversion of binary data to spikes is a common difficulty experienced by those interested in using pulse signaling strategies for logic.[92] In this work, spike based signaling transmits data with identified wires. Figure 4.8 shows a binary encoding used in low latency serial links designed for communicating digital 16-bit words by sending packets of 16 closely spaced (but not overlapping) spikes representing `0` or `1` across two wires.[93] The signaling used for this pulse based asynchronous serial link incorporates additional constraints for asynchronous signaling, described using the Impolite methodology which is detailed in section 6.3. Any of the spike codes utilize a minimum pulse-width, referred to as t_s, which includes the time for the spike to rise from baseline, the spike fall back to baseline, and a short resting period usually assumed to be equal to the rise-fall time of the spike.

Spike codes have a long history of study and were first brought to prominence through

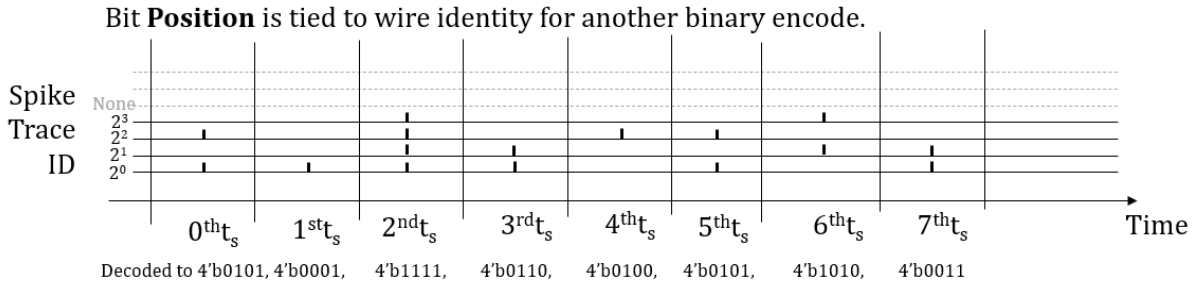


Figure 4.9: Conversion of a spike train to a binary code, using the identity of the trace wire as corresponding to binary bit position and 0 (no spike), 1 (spike).

the work of Thorpe, who made use of a binary bit position code, where the spikes are expected to arrive at regular intervals, and the number of traces corresponds to the number of binary digits an example is shown in Figure 4.9.[94] The binary code was only a starting point for Thorpe to expand into order-of-arrival codes, like those shown later in Figure 4.12. Bit position codes can transmit at most 1 code of bit-space length 2^N per timestep t_s.

The rate of a binary code in the case of $N_t = 1$, and using Figure 4.9 binary encoding, where 1 is a spike and 0 is no spike, and the bandwidth B for a pulse gate in 130nm is on the order of $B = 2\tau_m$, then $C = 10^9$ bits per second, using pulse width and spacing from experiments described in Section 7. When the energy use of the spiking channel is maximally utilized, that is the trace is toggling between 1 and 0, $H = 1$, and the rate $R = 10^9$ bits per second, where the 'symbol' is bits, because there the number of traces is only 1. Then, the rate, using $N_t = 26$ traces instead of 1, is an improvement over the rate of $N_t = 1$, because it uses far from the maximal energy use of the 26 traces to achieve 2.3 Gigasymbol per second.

A single wire can represent a single symbol from the alphabet, or, the single symbol events can be combined into multiple hierarchical symbols which transmit the result along another single wire. An example of this code is shown in Figure 4.10, where the

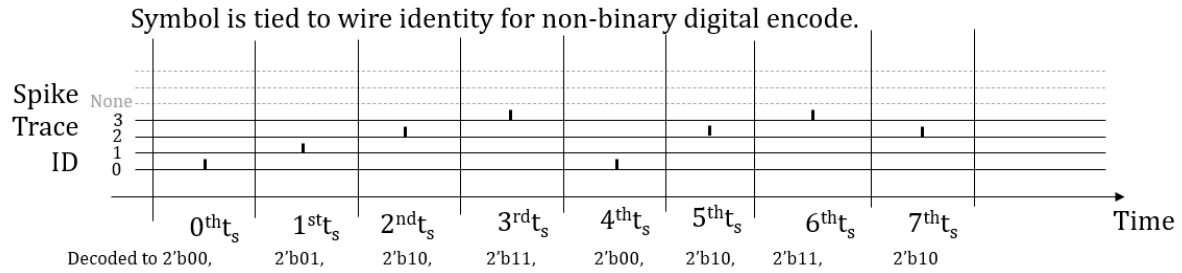


Figure 4.10: Non-binary digital codes assign a symbol to the wire identity, with more than 2 symbols permitted.

traces are labeled $0, 1, 2, 3$. In no case does a wire emit an event meaning symbol `1` and at a later time emit another event meaning symbol `2`. Additionally, in this sort of code, it is possible to add additional traces to the spiking channel, that are allowed to transmit twice the information for a specific code, for instance by adding a channel that only emits an event if an event happens on trace 0 at the 0th t_s followed by an event on trace 1 at the 1st t_s. Then on the 2nd t_s the trace representing 0 happened followed by 1 happened could be emitted and transmit the 4 bit-space code word for that order of occurrence.

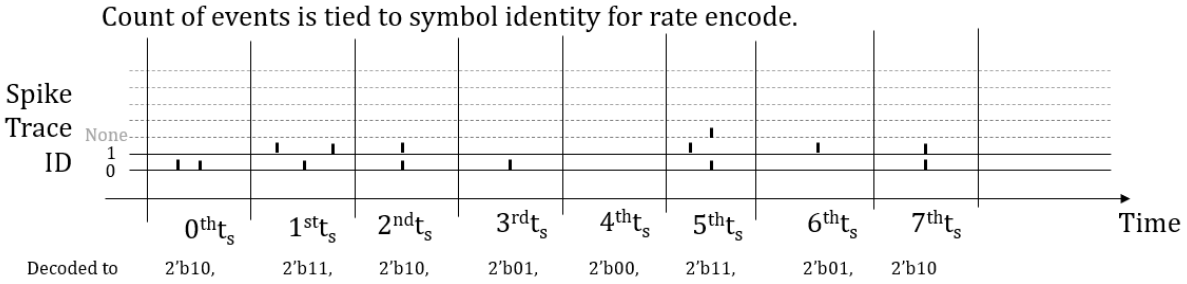


Figure 4.11: Rate codes assign a symbol based on the number of spikes which happen within a timestep.

Spike codes are frequently found in biological systems, where the spikes are translated into codewords using a timestep t_s larger than the minimum spike time, shown in Figure 4.11. Then the number of occurrences of spikes which happen within a t_s is used as a codeword, across multiple traces of a spiking channel.[95] This approach is often used for entropy estimation of biological systems.[96]

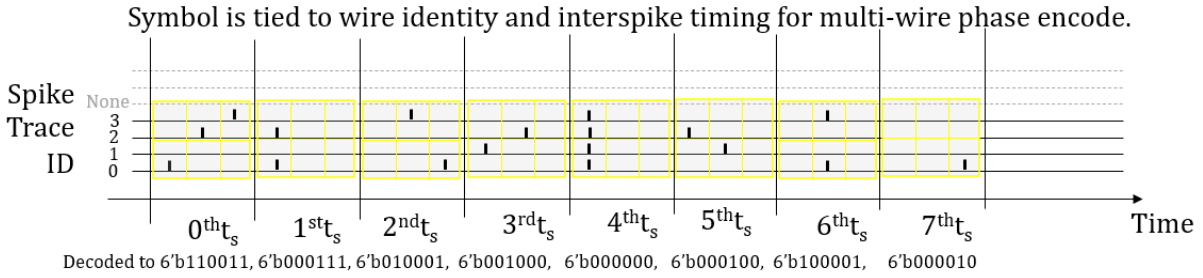


Figure 4.12: Multi-wire phase codes are able to transmit more bits than the number of wires, because they incorporate information about the local timing of each spike in relation to the other spikes in a timestep.

A communication system which builds upon the symbol tied to trace ID system utilizes a multi-wire phase code.[97] In this system, shown in Figure 4.12, only a single event is permitted to occur on a timestep t_s, however, the start and end time of each occurrence is permitted *happened before* and *happened after* timing relations creating inter-spike timing information. This means the bit-space of decoded symbols can be greater than the number of traces, but also the utilized code space may be less than the complete bit-space.

This overview of spike coding methods is quite incomplete compared to the literature which exists on neural spike coding [98], that includes oscillations [99], synchrony[100], persistent activity [101] and multi-firing barrages[102]. Additional capacity metrics are presented in Section 4.7, where the approach to capacity is presented in terms of neural codes, as opposed to the review of existing spike encoding means just covered.

4.6.1 Electronic Signaling with Digital Bits

The Nyquist theorem is used to convert an electronic analog signal into a digital electronic signal. A communication system consists of active electronic components surrounding a passive channel. At the smallest scale the passive channel is just a short and wide wire, meaning that the signal passing through the channel does not have to

travel a distance great enough for the effects of dispersion to matter, and the channel is wide enough that resistance to the flow of electric charge is negligible. The signals are transmitted using electronic signaling. There is a choice between using a level, edge

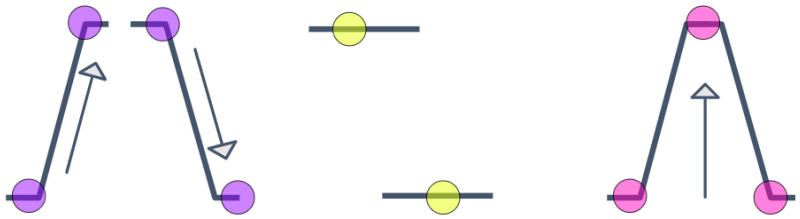


Figure 4.13: Signaling strategies use edge (purple), level (yellow), and pulse (pink) to require 4 (edge), 2 (level) and 3 (pulse) measurements.

or pulse as the unit of measurement for the communication, illustrated in Figure 4.13. Two edges and a level are a pulse, but how we measure to understand that changes the communications. That is because to measure a level we must sense at minimum twice, while an edge and a level is energetically half a pulse but takes four measurements to completely sense. Since we are able to choose levels or events, we can also choose to communicate using discrete events or continuous levels.

Physical measurement determines the possible sources of error, and when we are trying to gather a large amount of independent samples it is the limiting factor in the number of measurements we could make over a period of time.

Error and uncertainty build up in a system of measurements, over time as the measurements are completed, or simultaneously if many measurements take place in parallel. To demonstrate with a simple 'length' measurement, the length ℓ, If we have an object and we want to find out how long it is, get a ruler full of ℓ's, and use it as a known source of measurement. Observe the smallest unit measured on the device, 1ℓ and call it something, say u. Now it is known that minimum the margin of error is $1/2$ of u. Lining the ruler up with the thing, and making a measurement gives one independent measure.

We could go ahead and make that measurement 10 times, and then find the average measurement, to produce one result with some uncertainty to it. The uncertainty there will only diminish or change to $u/2$ regardless of how many times the measurement is made, because uncertainty comes from the ruler and its unit of measurement.

4.6.2 Differential Signaling

A differential signal is one that is transmitted using dual logic gate sets. One set of the gates operates on the idea of $+V = 1, -V = 0$ and the other set on the inverse of that $+V = 0, -V = 1$. Since the two sets are just signaling inverses of each other, they should still give the same answer as a signaling strategy based on $V - 0$ or $0 - V$. Throughout this work, most signaling is assumed to be $V - 0$ except when specifically noted as differential signaling. Using differential signaling at the circuit level is not problematic, although it does increase the size of the circuit area it increases reliability. In software using differential signaling doubles the number of computations with no benefits, so it is not used.

4.6.3 Neural Signaling

Neural signaling refers to the types of signals used by biological nervous systems to send communications between the underlying cells which compose the nervous system. The study of the entropy or 'information content' of nerve system cells can only happen by observing physically measurable signals. There are an assortment of methods for studying neural signaling and they range from nearly imperceptible non-invasive techniques, like recording the concentration of oxygen at spatial locations in the brain, to heavily invasive methods like cutting into the skull of a mammal to insert electronics.[103] All of the gathered neural signals recorded in biology labs share common concerns about how to

estimate entropy when the length of time the recorded information was gathered varies over multiple timescales.[104], [105], [106]

The problem stemming from biology and other experimentally gathered data is that instead of understanding what the smallest unit on the 'ruler' is, it is a guessing game. The starting state of that game is usually discrete or at minimum discretized data is present, because analog signals are recorded using analog-to-digital converters which are sampling the analog data at a known rate.[107] Starting from the neural recordings it's possible to use quasi-Bayesian estimators to quantify the statistical dependency between a pair of random variables.[108] That approach uses special functions to describe what could have happened in the past to result in the recorded data. Approaches to estimating the information in a neural signal rely on estimating the mutual information between two sections of the recorded data, a task which it is necessary to estimate the entropy of the data for, but entropy estimation is a challenging field, particularly when dealing with incomplete, wrong, or too few samples. Neural coding in particular is forced to deal with measured data that is only known in terms of the recording uncertainty, not with respect to the underlying information content.

Referencing Equation 4.3, where $|A_\chi|$ is the term that corresponds to the information content, and the choices of edge, level or pulse cost 4, 2 and 3 in additional uncertainty about each possible symbol. If one is to choose to measure 4 times vs 3 times vs 2 times the uncertainty due to measurement device uncertainty is still the same. The error in entropy from choosing to measure 4 times, 2 times or 3 times is still present as well, except the level measurement and the edge measurement are 2 observations apart from each other, and inadvertently measuring a level as a edge results in a possible extra symbol for A_χ.

4.7 Coding for Neural Communication

Biological systems use neuron cells to send messages in the form of electrical spikes through the life form. The encoding and decoding used by biological networks are an open problem under active study.[109] There are several observed biological neural codes, with multiple theories about how they may operate together.[98],[110]

Neural communications is typically treated as an unbounded problem, where the life form is able to have a population of interacting neurons that are able to support large (10^4) fan-in and fan-out connections.[76] Recordings of biological systems from a first reset event of a system are not able to happen without interfering with the growth of the life form.[103],[111] Lifespan recordings of neuron behavior within *C. elegans* worms are performed by using optical imaging of fluorescent calcium indicator proteins which are transferred to the worms via transgenic expression on the first day of the adulthood, occurring after egg and larvae stages of growth.[112]

We explore how much information the three primary neural coding strategies; population, rate and phase are able to contribute to measurable information states.[94] Neural coding strategies based on synchronicity and oscillations, which are derived from population dynamics, are excluded because those codes arise from the three primary neural coding strategies.[113]

An immediate problem with the translation of a spiking neural recording into a time series of discrete events is the question of what timing parameter defines an atomic event. The Hopfield model of neural connectivity assumes a synchronous global clock distributed to all neurons, which is used to measure an asynchronous 'firing' event as it occurs across the global population.[45] An alternative model proposed by William A. Little (1975)[4] uses continuous time dynamics with discrete time steps that are nevertheless able to

[4]Do not confuse Little model with Long Short-Term Memory (LSTM) by Hochreiter 1997.

maintain the key properties of the continuum for short and long term memories.[44]

The key takeaway from the statistical memory as proposed by William A. Little is that the long term memories are stored only within the connectivity of the entire network, and that the recall of long term memory is independent of time, while the short term memory is represented by a time step on the order of τ_m. Critically, to maintain the key properties of continuous time, the short-term memories need to respond after one timestep, develop into an intermediate multi-timestep code and finally settle into the long-term recalled memory. This approximation to discrete time does not rely on firing codes, but it does depend on intertime-scale decision making for binary spike events.[44] The following population, rate and phase codes are based upon a time slot of time τ_m, from the neuron in the system with the smallest refractory period, a time of which at least one measurable event is possible.

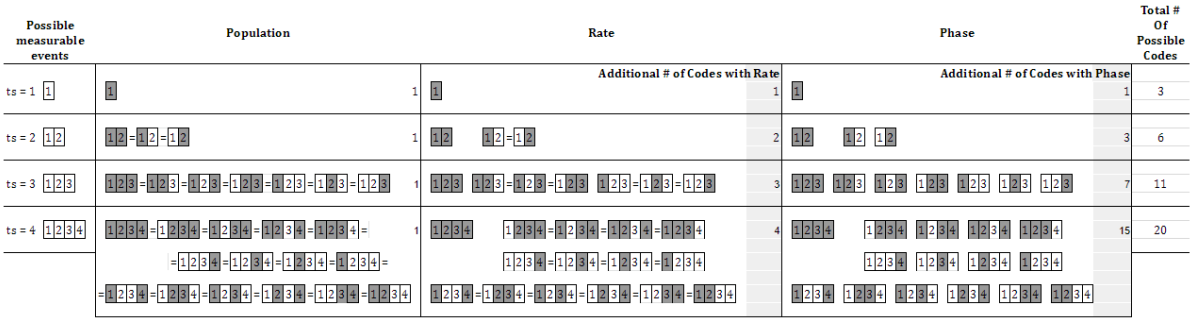


Figure 4.14: Population, rate and phase codes for $N = 1$ as the number of time slots t_s of size τ_m expands. All of the coding choices have access to the same range of measurable events, but the coding choice decides how 'redundant' the encoding is.

Figure 4.14 and Figure 4.15 shows how an observed temporal spike series can generate an encoding of population, rate or phase codes, where a symbol in the finite alphabet Ω is represented by a spike event. While each possible timestep is given a unique integer identifier here, that is only really the case for information maximally encoded as phase. It is unlikely that in practice all of the information would only be encoded as phase. If rate codes are introduced, then the coding can form subgroups of time steps that are

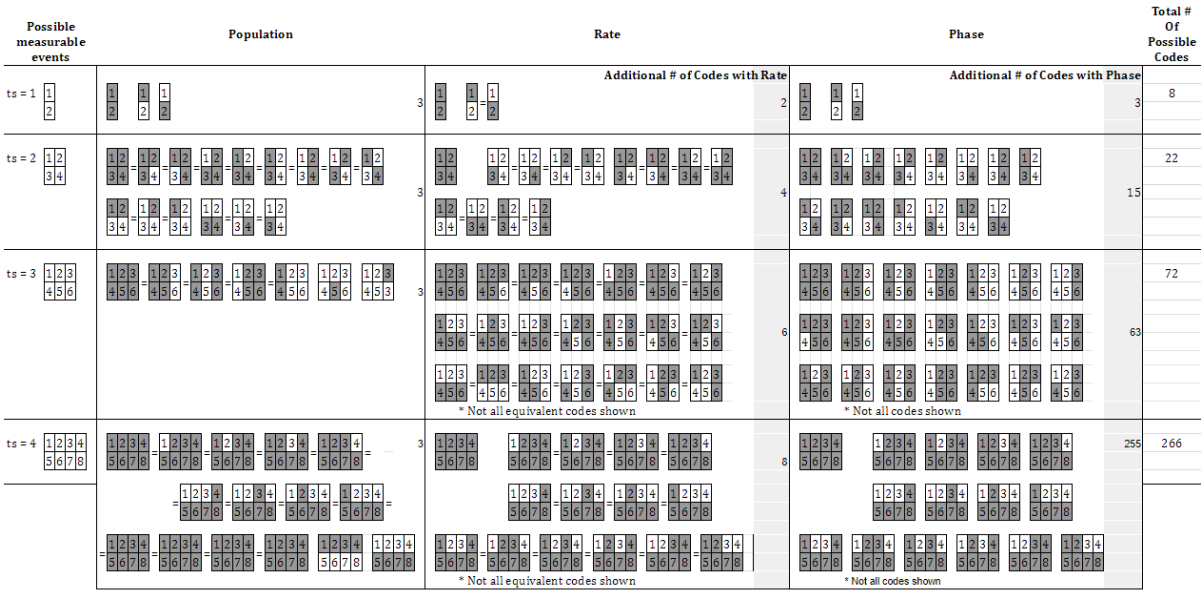


Figure 4.15: Population, rate and phase codes for N = 2 as the number of time slots expands. The number of available phase codes increases rapidly in comparison to the rate codes.

identically labeled. The cost to complete the reset event $S_u \rightarrow S_k$ for is 1 event from each neuron composing the channel.

Population Code

A population code looks at a group of neurons and asks how many neurons spiked in that group over a time frame of at least 1 τ_m. Population codes are found in biological systems, like the visual system of monkeys [114] and neocortex of rats[115]. Living systems have more than 1 neuron, with a general number known for many species under study. When the count of spike events occurring within a time frame for a population of neurons is bounded, the range of information states is limited by N, the total number of neurons.[94]

$$H(N) = \log_2(N + 1) \tag{4.5}$$

The simplest population code only assigns meaning to a neuron output in the form of event/non-event. It does not consider how many times an event might occur from any of the neurons, effectively setting the time frame equal to $t \to \infty$ through that approximation. Using a population of two neurons as an example, population code means:

—x—

—o—

For a population of two neurons over a time window, one the neurons spiked is the only thing that changes the outcome of the decoder.[114] A population code is limited by the measurable number of neurons in the system.

Rate Code

A rate code is for a finite time frame, $\tau_m < t_s < \infty$, after which it must become periodic.[94] This periodic constraint guarantees that the population code from section 4.7 is able to be expanded to count the number of spikes on a single neuron in the channel. Effectively, a rate code is just a population code where at least one of the neurons in the channel has such a large refractory period it can only spike once in the measured timestep. The presence of the 'long refractory period' neuron is not necessary for the rate code to function, as long as the time is finite. A code based on the number of spikes from a single neuron grows with the number of measurable timeslots, t_s. With a population of two neurons and two measurable timeslots, a rate code means:

—o—o—

—x—x—

For a population of two neurons, over two time slots for possible measurements, two events occurred on one neuron, meaning the outcome of the decoder could be decoded after the first event and updated after the second event. The number of spikes on each

channel changes the outcome of the decoder.

$$H(N, t_s) = \log_2(t_s N + 1) \tag{4.6}$$

The code assigns meaning to the measurement of a rate of spikes per unit time frame.

If a rate code is to be represented as a Markov chain with a stationary distribution it is necessary for some of the possible code states to have 2 edges emitting from that node. When the transition matrix is constructed from a node with 2 emitting edges as the starting point, it is able to have $e = 1/2$ for those entries and the spiking channel is able to settle into π.

Phase Code

A phase code is event based, when there is a single neuron that is sensing its environment, it will eventually emit an event (spike) which is encoding the state of the environment in the time it took to spike. When a single neuron is signaling with a phase code it does not transmit more than 1/2 of the possible information until it emits an event. Phase code means:

—o—o—

—x—o—

For a population of 2 neurons, over 2 time slots for possible measurements, the spike occurred in the second measurement slot. A phase code exists in a rich code space, as the time increases the number of possible codes rises very rapidly.

$$H(N, t_s) = \log_2((t_s N + 1)!) \tag{4.7}$$

In fact, for phase codes the information limit is not enforced by the discrete timestep limit of τ_m, because while the maximum information a spiking event can produce is a spike to represent τ_m, there is only an energy limit on how closely the interspike timing between two neurons who at rest beyond the refractory period can occur.[116] Thus, for a phase code to be useful it must happen only after the system is measured for 1 or more τ_m.[95]

Neuromorphic Communication Codes

Population, rate and phase codes are all produced by the same leaky integrate and fire neuron model. Multiple neuron scenarios depend upon a modicum of connectivity, with the illustrations in Figure 4.14 and Figure 4.15 understanding that to output two events as shown in Figure 4.15 requires two spatially different measurement points taken from the same SNN communication channel. With populations that are measured at 1 or 2 spatial locations for 1 or 2 timeslots, the maximum information present in measurable slots has a Markov mixing time less than the number of possible measurement slots. For a small population with many measurable timeslots, the size of the possible code space becomes much greater than the mixing time for the Markov chain. A Markov chain has a finite depth of the mixing chain it is possible to maintain before being unable to discriminate and the code space of a spiking neural net is bounded by the number of output measurements which use phase codes beyond the number of rate codes. The codes are based on sensing in the environment; a solution to limit the extremely quick rise of the phase code space, is to increase the dependence on dendrite tree computation as shown in Experiment 7.2.4.

The combination of all three shown in Figure 4.16, has a unique single measurement

9 combination code means:
x|x|x
x|x|o
x|x|x
Over a time window with 9 possible measurements, the first missing spike is important, or, the eight occurring spikes are important.

Figure 4.16: An example of a triply replicated neuromorphic code. x means spike. | means new time step. o means no spike.

feature, because it requires to either measure 8 (x) successfully or 1 (o) successfully, and measuring only 1 successfully is a great way to capture plenty of information at once. Independent of the selected encoding the observed measurement has a chance to confuse a transition between edges and for a 'good' probabilistic encoder, with the meaning of the message triply encoded across phase, rate and population it is unlikely to be a poor mistake. For instance, if the phase code is the first fully decoded message, there is still a subset of rate and population codes the message can also correctly decode too. If both the rate and population codes also decode to the same message as the phase code, then the likelihood of receiving all three codes incorrectly is low. If the rate and population codes decode to a differing message, then the correct message can be subject to a vote among three possibles, which still removes most of the other candidate messages from consideration and is a favored outcome compared to complete uncertainty about what the message was. Additionally, if there are three candidate messages, the decoder could wait an additional time step and craft a new set of three possible messages, to use as a vote between six candidates.

It is true that phase coding of neural signals is the assumption that the arrival of spiking signals in a particular order contributes to information concerning the structure of the world. i.e. the sensory signals of the hand are able to detect forces and directional changes of the object contained within their grasp.[117] That means that a triply

redundant message with varying decoded meaning need not be incompatible with each other. The environment is a complex scene and a signaling strategy which allows for varying meanings to a single message is able to respond to the changing environment with multiple meanings. Existing neuromorphic codes demonstrate these multiple meaning messages. To compare two known strategies used for spiking communication, consider the Winner-takes-all algorithm and the Winner-shares-all algorithms. In a winner-takes-all approach, the first arriving spike (a phase code) on a particular trace is determined to be the only spike of importance, and the meaning assigned to that trace is decreed the decoded memory to be accessed and returned as the 'answer'.[118] Winner-shares-all algorithm is similar to winner-takes-all, but has additional reasonable bounds on the length of time for a learned pattern to be recalled. It takes a few time steps to recall, instead of the single time step in winner-takes-all. Winner-shares-all is a fixed pattern for remembering how to do things. For the winner-shares-all algorithm the number of output neurons heavily influences the possible solutions the machine is able to output. The architecture of a winner-shares-all algorithm is influenced by the data used for training. The number of outputs is close to the number of classification label.[119]

Any neuromorphic code can be applied to any information source, although, the effectiveness of the codes only become apparent at alphabet sizes $|A_\chi| > 2$, as shown in Section 2.2. An example information source with 24 alphabet symbols is shown in Figure 4.17. The Markov chain shown in Figure 4.17 is a simply ergodic Markov chain because from any of the symbols it is possible to eventually reach one of the other symbols in the set of $|A_\chi|$. It is also a second-order Markov chain, because, the probabilities of reaching the symbols is taken from a dataset which consists of two entries which are equally probable. That means this graph can be represented in two ways. From one coding perspective (the population code) it can be decoded to mean one of two events has happened, because there are three symbols in the graph that are unique between the

two population codes (The symbols `4,1` only occur in one entry of the underlying dataset, while the symbol `f` only occurs in the other entry in the dataset.)

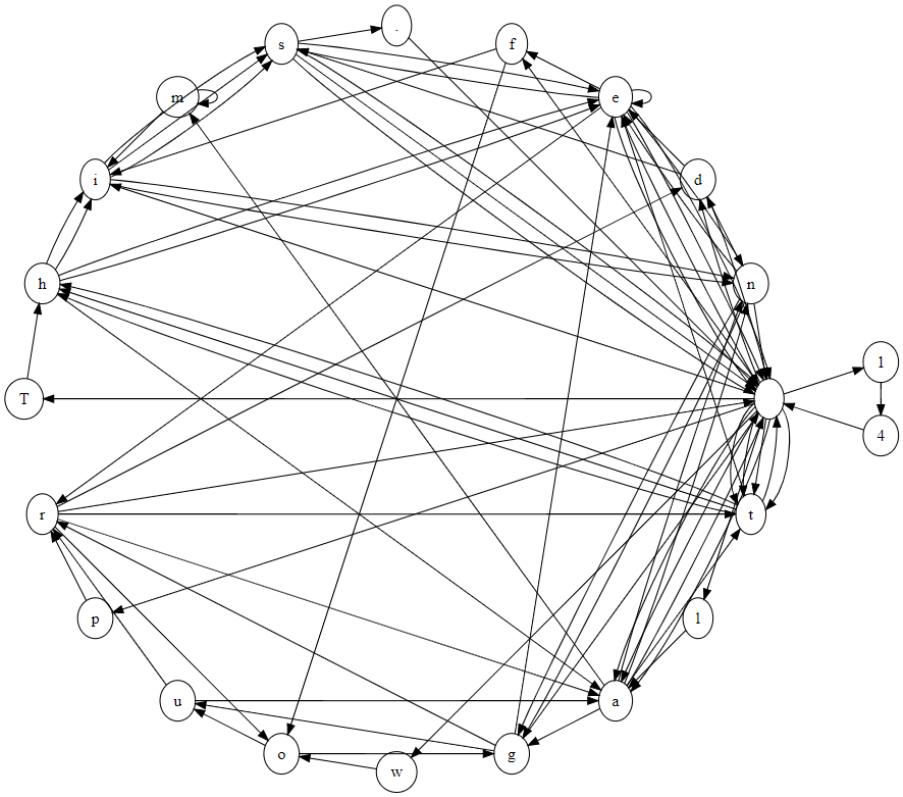


Figure 4.17: Monogram symbols from simpleThis in a connected graph. The monogram is the minimum number of symbols that an information source must contain to enable instantaneous communication of atleast 1 message. Otherwise, a communication cannot happen without sending 2 or more symbols.

The allocation between the symbols can be assigned to population, rate or phase codes. If the allocation is split so that the energy cost to send each symbol is equal, that means the alphabet codes are not assigned to rate codes, then a third order Markov chain can be constructed from the alphabet symbols (shown in Figure 4.17) and made available for later construction into a second order Markov chain that could be represented as an energy cost rate code. That would mean that even without direct access to information concerning which traces are spiking per time step, it would still be able to recover the

information present in the third order Markov chain by only measuring energy use of an ASIC circuit and without measuring trace level activity. An example of a second order Markov chain constructed in this manner is shown in Figure 4.18.

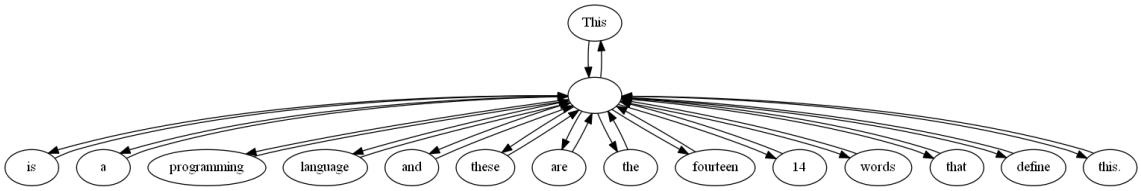


Figure 4.18: Varying length strings from simpleThis using 'space' symbol as a return point.

The varying length strings composed from the simply ergodic third order Markov chain which held the original alphabet symbols can in turn be used to create their own second order Markov chain, as shown in Figure 4.18. Noting the symbols used in the second order Markov chain, there are two alphabet symbols `1` and `4` which are not present in this second order Markov chain. That is because there are actually two separate and distinct second order Markov chains which are constructed from the original alphabet symbols. Each one of the second order Markov chains corresponds to a unique entry in the underlying dataset used to construct the information source used to create the Markov chains from the monogram symbols. Only one of the dataset entries is used to create Figure 4.18, however, an additional figure could be created as well for the other dataset entry. It looks very similar to Figure 4.18, however `fourteen` is replaced with `14`. That means that the energy use between the two data entries is unique and identifiable. The meaning of a spike train observed from the spiking channel corresponding to these two entries could be decoded using phase alone, or using rate alone, or population alone. However, phase would be the fastest way to detect the meaning of the message since it would not rely on transmitting the entire message before decode.

4.8 Spiking Communication Channel

The sending of a message through a communication system involves using a component, the channel, which is the media occupying the space between the communicators. It is easy to visualize it as a tube or pipe connecting two different spots. A message is dropped in one end of the pipe and emerges at the other end of the pipe. That is a simple picture, good for starting to think about a channel. It could be a single wire, or many wires bundled together, or optical light, many different things are useful as channels. What makes the different physical media part of the same digital 'channel' is the agreement of the designer as to how all of the media will operate together to transmit.

The channel describes the transmission media which is paired with the encoder and decoder. Channels are described by whether or not the signal contains noise.[120] Examples of different types of channels include on-chip wires, which once they show signal degradation with transmission line effects, are noisy channels.[74], [121] The binary erasure channel is a wire which is transmitting a binary symbol and has a probability of receiving the symbol p_0 and a probability of not receiving the symbol of $p_1 = 1 - p_0$ While in this work the channel is electronic, there are also optical channels and other more unusual channels. For instance, a pressurized wave sent through a controlled path could be considered as a channel.

A spiking channel is taken from biology, where it is observed that the central nervous system of living organisms signal around the body using combinations of voltage-,ion- and neurotransmitter-sensitive conductance. The neuron cells communicate amongst themselves with nerve impulses, self-regenerating spikes of the membrane voltage.[24]

The channel is used to send symbols from the information source which have been encoded into a suitable representation for the types of noise the channel is afflicted by. The characterization of the channel should be performed to determine what encoding to

use. The encoding should match the effects of the channel and counteract against common errors. A noiseless channel is a theoretical channel that does not exist in practice, it is the ideal channel if there was no noise.

Shannon's noiseless coding theorem states a valid lossless message is permitted if able to transmit one symbol more than the original data contained. Shannon's noiseless coding theorem introduces the concept of a block of symbols used to compose the message. It then states that once one symbol more than the number of symbols in the block is transmitted the message should be fully recoverable, without loss of data. But what does this mean when we are trying to compute while transmitting the data?

As soon as the first symbol has been transmitted and received a lossy representation of the data should be possible, and as additional symbols are received to complete the message a lossless recovery is possible as soon as $|A_\chi|+1$ symbols are transmitted. Figure 4.10 demonstrates this spiking channel feature with 4 traces. This means that a lossless recovery of the trace identities should be possible once each trace has emitted a single event, plus any one of the traces emits a second event. This is because by connecting spike trace ID to a symbol the binary block is represented by sensing a spike event instead of sensing level or edge events corresponding to the block size.

In the case of $|A_{simpleThis}|$, where we choose to assign a trace to each of the monogram symbols (24), and 2 extra traces for each of the dataset entries, $N_t = 26$

$$H_n = -\frac{1}{1}\sum_{i,j,...,s}^{n=24} p(i,j,...,s)\log_2(i,j,...,s) + -\frac{1}{84}\sum_{i,j,...,s}^{n} = 2p(i,j,...,s)\log_2(i,j,...,s) \quad (4.8)$$

The summation of the probabilities for the monograms is detailed in Section 8.4, the

primary objective was to calculate the rate R, which is:

$$R = \frac{C}{H} = \frac{26 * 10^9 \; bits \; per \; symbol}{11.266 \; bits \; per \; sec} = 2.3 \; Gigasymbol \; per \; second \tag{4.9}$$

This rate, using $N_t = 26$ traces instead of 1, is an improvement over the rate of $N_t = 1$, because it uses far from the maximal energy use of the 26 traces to achieve 2.3 Gigasymbol per second.

With channels, the possibility of opening and closing the channel comes up, and for synchronous channels the 'open' state is when a clock is active. For asynchronous channels 'open' can refer to being in a state shortly after a 'handshake' signal is received. An asynchronous communication channel which does not use a handshake is Impolite, and is fully described in section 6.3. For Impolite asynchronous channel, 'open' is anytime unless things are physically disconnected in which case it is no longer physically a circuit. Impolite asynchronous channels are not closed intentionally.

4.9 Encoding and Decoding

Choosing a method of communication that makes sure the operation and intention of the message come across early is very important in the selection of an encoder. The construction of the non-binary digital encoder differ's from conventional binary encoders as shown in Figure 4.19.

After the stationary Markov chain is used to encode the input for transmission through the channel, the receiver absorbs the energy and presents the received message to the decoder. It is up to the decoder to translate back into the original language.[32] A straight forward decoder performs the inverse operation of the encoder.[122] It begins to perform the inverse operation as soon as it is able to absorb any energy, and once it has begun

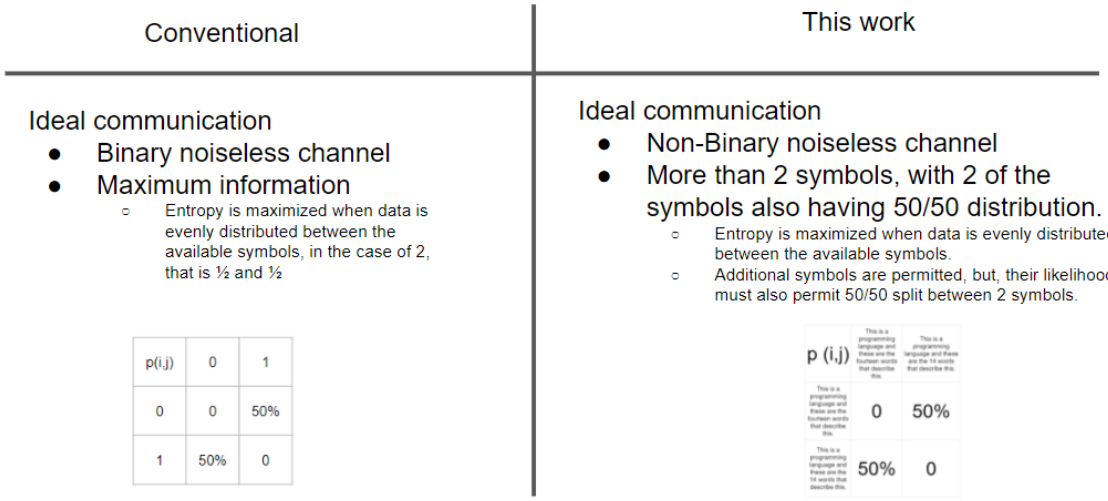


Figure 4.19: Conventional binary encoder compared to non-binary digital encoder.

to decode it is continually decoding. The decoder is also a stationary Markov chain, with the same dataset as the encoder. The decoder stationary Markov chain operates on single symbols, blocks of symbols and sequential symbols. The decoder requires its own energy source, separate from the energy absorbed by the receiver.

At its simplest form, the transmission could be a direct, noiseless symbol which the decoder is able to immediately recognize and add to the received message.[123]

When the decoder is a probabilistic decoder, it means that once it receives a symbol it is necessary to add it to the existing partially decoded message. As the new symbol is incorporated into the message the entire meaning of the message could change, or it may not change at all. The longer the message is stationary the greater the confidence in the decode should be.

For efficient energy use the decoder should have advance knowledge of the bitspace of the information to be decoded into. Or in other words, the size of the communication channel should be the same in the start and end. If this is not the case, or, if the size is unknown, it is acceptable to embed the decoded message int a projection of simply ergodic binary data and b-system data, as shown in Figure 4.20.

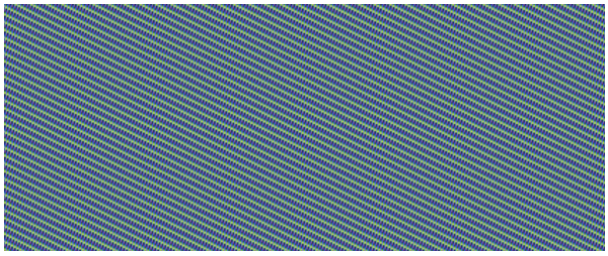


Figure 4.20: 2D rendering of simply ergodic binary data and b-system data.

In practice the size of the information to be decoded is known. Often encoding operates on fixed-width blocks made from binary symbols with blocks of size N representing the number of bits in the block.[32] Generally for a fixed-length block code if a message to be transmitted is less than the size of the block, it is still sent in a packet of the block size, with additional filler to occupy the extra space.[32] When the block is permitted to include multiple measurement binary symbols in groups, they are sometimes classified as q-ary symbol codes, which devote some of the measured binary symbols to classifying which q-ary symbol is being represented. To avoid the problem of an unknown decode size, all of the decoders (in this work) place the message into a simply ergodic encoded base message, that is allowed to expand and contract as necessary to fit the best possible decoded values. An example of how this base message looks for a 2D compressed example is shown in Figure 4.20. The visualization of the base message is composed by repeating the information source binary encoding shown in Figure 4.21.

A decoder can impart 'compute' energy on the received message to repair any errors and restore the message to its original noiseless form.[124],[125],[126] A decoder can perform operations upon the received message apart from error correction, and making the additional operations programmable will increase the generality of the decoder as a computation machine. For example, an adder which receives two binary values encoded for error correction from memory, checks those values and then computes the sum, is a communication channel which spends more energy to produce the computation result.

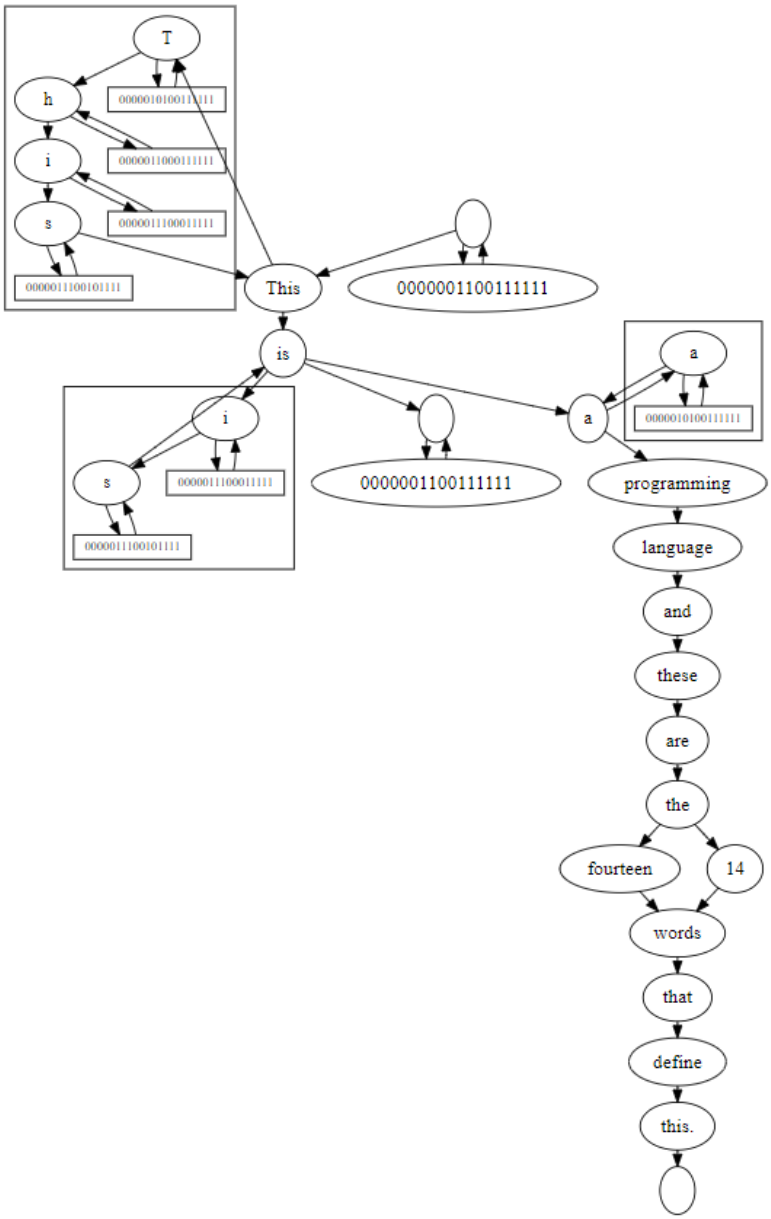


Figure 4.21: The software encoder used to create sparse temporal spike streams from binary data. Built on the dataset from simpleThis. For purposes of display the binary encoding used to drive a third order phase-coded Markov chain is shown only for three of the second order codes (This,is,a) and one of the first order codes ( ) (space is a first order code).

This is because while the transfer of the original values of the addends only costs the communication energy, to shift the output of the decoder from binary b-system data

requires an expenditure of energy equal to the number of bit-flips necessary to place the sum into the binary b-system data space.

An encoder capable of taking an input and transforming it into a phase-coded software generated Markov chain is shown in Figure 4.21.

Chapter 5

Semi-intelligent Computation with Neural Coding

Biological intelligence occurs in two stages of development. First the cell divides into a multicellular organism that has the baseline abilities that already know how to survive as that spealized type of multicellular organism. We call that first stage of development the acquisition of semi-intelligence and propose that its signature is simply ergodic patterns of local behavior, with global b-system patterns, i.e. a beating heart, or some other sort of slow periodic signal.

Neural systems with large feed-forward and feed-backward loops are found in humans, and other types of mammals, birds, amphibians and fish. Vertebrates and invertebrates are arguably all considered intelligent when compared against alternative types of life.

5.1 Semi-intelligence

Languages between specific intelligence's are unique, to observe this, monitor a back and forth communication and construct an n-gram of that conversation. When two

parties claim to have a shared language it means that the declaring party thinks it has spent enough time receiving communication from the other intelligence to form its own opinion on the communication, but not that a language is actually shared.[127] Language starts from symbols, the two parties think they have shared symbols in common and they babble back and forth, learning to express internal states.[128] Experiments which study the neurons of the juvenile songbird have uncovered that species-typical brain circuits contribute to the ability of the songbird to eventually learn songs as sung by adults of the species.[129] Symbols are best represented as population codes, because a communication channel cannot exist without a symbol set, and neural communication would also become impossible if population codes are excluded from any type of neural codes. If artificial intelligence is only observable through communication from a life form, with the most basic understanding of intelligence, that is, semi-intelligence. Semi-intelligence is simply the devoted consumption of energy with the intent to communicate information.

Computational language models are constantly encountering new words that are not in their original model.[130] While human languages continue to introduce new words over a lifetime, the number of symbols is mostly limited to those learned prior to adulthood.[131],[128] Handling the continual arrival of new words is more difficult for computational models, because there is always a chance that a new word might introduce a new symbol. Sampling across a dataset of words to try and gather all of the necessary symbols is a way to reduce computational complexity of a language model.[131] The language model of an intelligence does not have guarantees unless they come from the symbols of the language. For instance, if the language needs to be built up from only binary symbols, then it could have guarantees about how understandable it is, up to the underlying known symbols, by enforcing a bit length to symbol relation for a language. In this case, the parity of the language could be used to implement a selection of minimum height XOR trees to guarantee a verifiable bound of computation down to the

language symbols as represented in binary. This example is shown in Figure 5.1 where combinations of a binary code are taken to guarantee a 50/50 rate, as in the 8b10b coding family.[86]

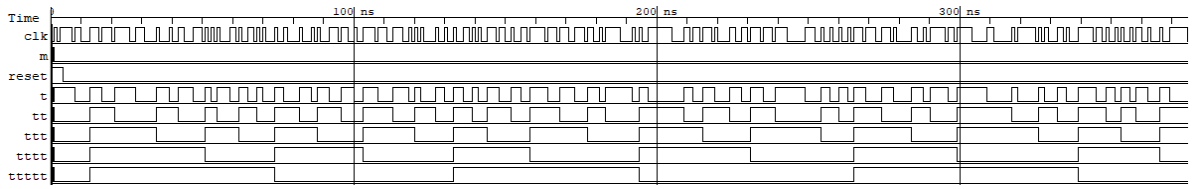


Figure 5.1: The output of an XOR tree (line ttttt) is a periodic slow signal, when the input (clk) is an irregular input of high and low with the average value 50/50 between 0 and 1.

If the guarantee of 50/50 duty cycle is evident from the low code rate (line ttttt) then it should guarantee that it could hear multiple symbols at the high code rate (line t) at a period set by the low code rate. A semi-intelligence has internal signals that operate at multiple rates.

Language models are built up from studying an underlying dataset.[131] Since language is inherently temporal (we only read a few words or characters at a time and the order we do it is important), recurrent neural network architectures are a good match for learning languages with their long memory.[131] First software models demonstrated success at tasks such as sentence generation and prediction of the next word.[131] Hardware implementations of long term memory using a Legendre memory unit are successful solving MNIST image recognition over an observation time period, and they are run on analog hardware (Braindrop[62]) and digital hardware (Loihi[132]). Sentence generation is done using sentences made of 'words' using word2vec. Word2vec describes a process for encoding words with unique identifiers in an encoding that assumes a nearby linguistic distance between the words, if they are typically near each other in sentences.[78] This encoding schema is useful during the training stage because it lets the training operate on vectors.

Languages are only necessary when an intelligence or a computer needs to communicate. An intelligence or computer which does not need to communicate does not need to have a language. However, either of those things without communication are functionally useless. Semi-intelligence needs to use a symbolic language to communicate, and, it should be communicating the result from sensing in its environment.

5.2 Irreversible Biological Computation

Where in the biological system lies the same irreversible operations found in computation? When biological stem cells are forming into an intelligence, the Sonic Hedgehog Signaling system[1] functions as a genetic bistable switch, that is modulated by the local environment the stem cells are experiencing.[133] As the stem cells operate to guide the developing lifeform through development and into adulthood, there become times of two alternate fate behaviors, decided in an all-or-none fashion.[133] These types of decisions include cell survival versus apoptosis, and other choices leading to major consequences in the eventual outcome of the intelligence.[133] Biological systems experience both microscale reversible computation, at the level of RNA assembly,[61] and macroscale irreversible computation through fate deciding assembly behaviors.[133] Irreversible computation is necessary for progress towards a fully functional life form that may be capable of expressing intelligent communication. The overall energy used of developing life takes placed in an energy bounded environment. While it does differ between individuals of a species, overall, the lifetime energy use across species has a species level range, quite different across species.[134]

[1]Sonic Hedgehog Signaling system is from biology, and it was discovered in the late 1990's, during the time period when the video game Sonic the Hedgehog was popular. There were already two existing other hedgehog signaling systems (Indian and dessert hedgehog). The hedgehog name is because in a Drosophila larva, the failure of the hedgehog signals results in a larvae that looks like a hedgehog. Harvard researcher (Cliff Tabin), discovered this one and decided to name is Sonic Hedgehog.

With a non-binary code based on a maximum finite energy use with a finite set of lifetime timesteps, the energy use will move up and down, at times when the environment is not causing much activity the energy use might dip below the expected computation use. The slow periods of time indicate a non-binary XOR operation, taking place with little energy use and corresponding to Table 7.1. The necessary need to operate at both slow and fast speeds is used to establish a lower bound for computation in Chapter 7.

Biological systems demonstrate that there is value in performing irreversible operations and if that operation happens to be the capture of the environment into the memory, the minimum value of energy expenditure that should not be reversible is 1/2 of the enclosed system energy. The time to capture the measurement also needs to take place in a fast enough time that the slow movement of the atom has not changed.

The ratio between the slow and the fast, has limits as to what speed of performance a requested capture must have to take place. For digital signaling circuits using pulses, that limit has to do with the variance and noise margins needed by the rising edge uncertainty.

That uncertainty means that the amount of reliable computation fast could be expected to do, is a ratio to the slow boundary. When an XOR gate is sampled at a periodic rate, eventually it will convert a noisy signal into a periodic wave representing the duty cycle of the original fast signal.

Periods of low energy use are stationary measurements and they last for time greater than the τ_m of the system as the system inputs are constant and it becomes a physical system with a function describing it by repeated measurements to confirm the observed system is stationary.

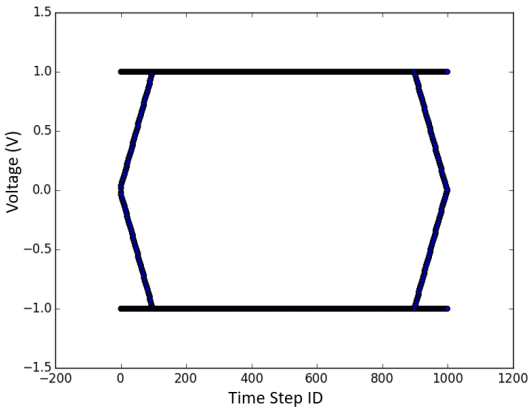
5.3 Biological Computation

Biological computation takes place in the central nervous system.[135] The intelligence is comprehensively assigned to the entire physical structure of the life form be it a bird, bat or fish.[136] The central nervous system is there connecting stimulus from the sensors into a heavily feedback driven mass of neurons which eventually display perception and motivate the intelligence to respond to the environment.[137],[69], [138], [139] Intelligent communication corresponds to the abstract meaning defined by the biological system, not the physical properties of the communication channel.[70]

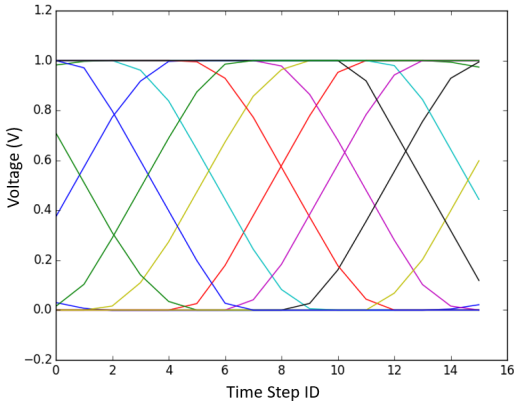
To make testing and verifying an intelligent communication less complex, a bare minimum of acceptable intelligent signaling, that is, semi-intelligent signaling, is defined as a simply ergodic spike pattern across a collection of measurable neurons. For instance, recordings of ferret neurons exhibit two different oscillating patterns when exposed to synchronous and asynchronous tones.[140] That means if the ferret neurons are labeled with traces $0, 1, 2, 3, 4$, when one sound is playing the neurons have events in the pattern $0, 1 \to 2, 3, 4 \to 0, 1 \to ...$ but for another pattern is $0, 4 \to 1, 2, 3 \to 0, 4 \to ...$ where some of the spikes are occurring on traces at nearly the same time, however increasing the precision of the measurement might find that to be a gross error. Rats develop oscillations of spikes in the olfactory bulb when hunting food scents long distance.[141] The spikes come in waves across the spiking channel traces, but a common pattern is that the granularity of the measurement can be changed until the oscillations indicate groups of neurons that transition between activity. In an evolutionary system overall survival of a population is improved by communication instead of fighting.[142] Semi-intelligent signaling formed from a simply ergodic set of primitive symbols enables a greater number of communications for the best outcomes of long life and reproductive success.

Neural signaling includes information coding strategies which are not obvious from

viewing an eye diagram and require the inspection of multiple measurements to discern response.[143] Eye diagrams are useful for clearly defined static timing.[120] Neural sys-



(a) An ideal static timed signal has an open eye.



(b) A temporal phase coded signal has a closed eye.

Figure 5.2: Eye diagrams are used to understand what constitutes an ideal signal for static timed (a) and temporal data (b).

tems are spiking events, and the information is dispersed over time.[24] Phase encoding means one pulse has significant information in comparison to other pulses and if it is used in a conventional 'good' channel that has a clear eye diagram (Figure 5.2a) it may still be mistaken as noise.[144] As sensing occurs over a 2D area or with temporally oriented sound, the internal spike timing of the neuron's improve their jitter as the sensing moves deeper into the CNS.[135] The reduction in jitter leads to a closed eye with multiple neatly separated phases as shown in Figure 5.2b. Observing the propagation from sensed spikes through to behavioral change demonstrates that sub-sensor timing is used to communicate through coordinated actions of small groups of neurons. [98]

Chapter 6

Alternative Computing

Computation is an inherently noisy process when completed with classical computers. Boolean computation is performed with a mechanical or electrical component called a switch.[7] A typical computation is measured by understanding if current is flowing, or not, with the switch. It is also acceptable to measure computation with integrated charge, as demonstrated in Section 6.3. Self-resetting domino logic circuits are used to sense integrated charge in pulse gates.[68] The pulse gate is a dynamic computation circuit which is different from a switch, because it does not have a requirement for flowing current, only for the transition of charge quanta.

Modern computer architecture is dominated by Von Neumann or Harvard data processing styles,[85] but there are also many variants of computers that are less expected in their offerings. Recent proposed computation machines include architectures such as the Neural Turing Machine [145], Nondeterminstic Finite Automata [146] and a fault tolerant Multicore processor [43] designed with principles of biological resilience. The implementation of these new machines all share common principles of digital logic at the silicon level, where Boolean variables are used to build circuits matching their RTL descriptions. These machines are binary computers, and even when their instruction set operates on

instructions longer than 2, what is meant by statements like a 64-bit CPU or 32-bit CPU is an instruction bit width that requires 64 measurements or 32 measurements.[85] This is the form of computation design pioneered in the 1930's by Claude Shannon in their thesis on bi-stable switching theory.[7] Early computers did consider alternative computation bases, including analog computers and systems dependent on a non-2 base (Base 3 is ternary and base 10 is decimal). [147],[148] Ternary computation continues to prompt investigation into new architectures and materials explicitly designed to capture the benefits of base 3 computation.[149] The circuits in these alternative architectures continue to rely on external memory to store the state of the global system, with specifics for each architecture about when possible errors are corrected and how data is transferred through the system. Quantum computation is a growing area of study, with recent results indicating success at building machines capable of performing computation in complex valued Hilbert spaces.[150] While quantum computation offers the promise of very low energy use reversible computation, the practical constraints of classical computation show how a known cost for irreversible computation might also be beneficial.[151] An irreversible computation will have a definite cost of energy that is a permanent cost for the answer.

Modern computer design is non-stationary, during the operation of the machine the probability matrix of finding the computer in any particular state is constantly changing. Alternative computation explores new types of computer design, and in particular, Space-time Computation (Section 6.1) indicates that stationary computation is attainable. The utility of stationary computation is explored by first establishing a lower bound for neuromorphic communication in Chapter 7 and second, by demonstrating software experiments with stationary computation beyond 1-bit space in Chapter 8.

6.1 Space-Time computation

Any computation which relies on the passage of physical time to arrive at an answer is a space-time computation.[8] **Does that mean transmission through a channel is a space-time computation?** Yes, it takes energy to erase information, and it takes energy to transmit information. Most of the time when we have a channel we want to send the 'data' from one end of the media to the other end of the media. Using more energy than necessary to compute during transmission is normally avoided because computation has a cost and the original goal of communication is to move a copy not a computed result.[18] But, if you wanted to care about computation along the way, it would be possible to design it into a channel.[35]

6.1.1 Race Logic

This section shared with conference paper, "Low energy response of spike train encoded data" by C.H. Segal, presented at ICICT London 2022

Space-Time computing, when implemented with race logic, is a form of computation where information is embedded into timing delays.

6.1.2 Circuits for Non-binary Digital Codes

Alternative computing based on temporal 'race' logic enables non-binary encoding.[11] The state identity of a symbol can be encoded based on the transit of a single solitary wire.[12] A system using two-wire ($w = 2$) encoding to indicate '1' with a wire and '0' with the other wire is used for high speed signaling in asynchronous circuits.[152] Systems with $w > 2$ are theoretically proven and physically simulated to solve classification problems.[153]

Race Logic is a CMOS implementation of Boolean logic that embeds the timing and

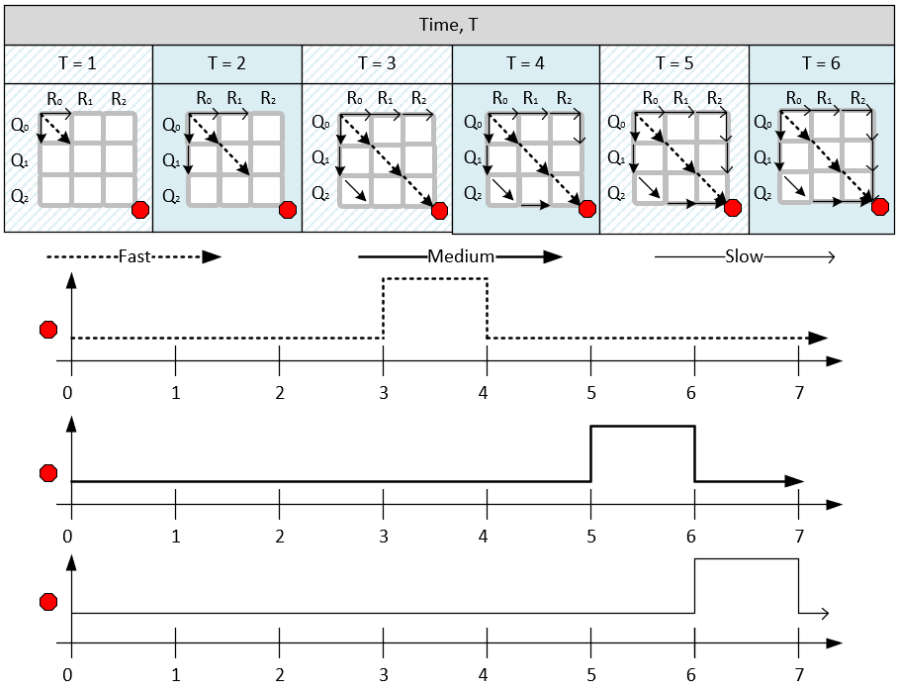


Figure 6.1: Race logic circuit with $A = 3$ symbols, **fast**, **medium**, **slow**, shown as voltage timing waveforms. A 1D compute grid operated on stationary program signals Q and R, to configure the timing path of an incoming spike (upper left corner) which eventually leaves the 'race' (octagon, lower right corner) encoded with data and timing information corresponding to a non-binary digital signal representative of the possible computation outcomes for Q and R.

data information into a single edge or pulse.[12] A compute grid of identical compute elements is configured with a 1D program, that is broadcast, across an array with the unit cells providing alternative timing paths dependent upon the 'program' stored in the stationary computation vectors R and Q.[154][1] Each of the individual compute elements operates on stationary binary input signals and the communication signal which propagates out the end point of the circuit is a non-binary signal which decodes to the result of the program.

CMOS circuits to implement race logic are concerned with the measurement of a transition $\overline{01}$.[8] The physical measurement has transition probability equivalent to a

[1]The references on race logic use P and Q. In this work R is used instead of P to prevent confusion with P probability.

first-order **b**inary **M**arkov **s**ource P_{FOBMS}.

$$P_{FOBMS} = \begin{bmatrix} p_{0|0} & p_{0|1} \\ p_{1|0} & p_{11} \end{bmatrix} \tag{6.1}$$

For the binary source the stationary probability vector is represented $(\mu(0), \mu(1))$, where

$$\mu(0) = \frac{p_{0|1}}{p_{0|1} + p_{1|0}}, \mu(1) = \frac{p_{1|0}}{p_{0|1} + p_{1|0}} \tag{6.2}$$

The reset state is taken from the two binary options. It is the state corresponding to the stationary distribution $p(\chi_1 = 0) = \mu(0)$, or the probability that if the reset state is 0 the probability of observing $\mu(0)$ is 1. From reset, χ^n represents all possible n-bit sequences containing a single transition as a second order Markov chain from the original first order stationary distribution with reset. That sequence $z_1 = 0^{n-\ell}1^{\ell}$ has a sequence of $n - \ell$ 0's followed by ℓ 1's.[155]

An entropy estimation taken by sampling the output wires would need to gather independent samples at multiple times, equal to the number of possible symbols (2 for `binary`), before it would be able to return accurate estimates.[33]

A first-order non-binary Markov source that is *simply ergodic* is one that has at minimum a non-zero probability of transition in each row. Furthermore, to establish a baseline entropy to energy equivalence, a first-order Markov source has a *B-system ergodic* if for each $p \in P_\chi = 1/\chi^2$

$$P_{FONBMS} = \begin{bmatrix} p_{00} & p_{01} & \cdots & \cdots \\ p_{10} & p_{11} & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & p_{A_\chi == H_\chi} \end{bmatrix} \tag{6.3}$$

If the number of symbols is increased from binary to 3 or greater symbols, as demonstrated in Figure 6.1, where 3 symbols, `fast`, `medium`, `slow` are shown as voltage timing waveforms, the theoretical entropy cost to access the reset state increases with the entropy of the order of the Markov chain $\frac{1}{2}\log n$. Where n is the number of possible symbols used by the specific implementation of the non-binary digital encoding.

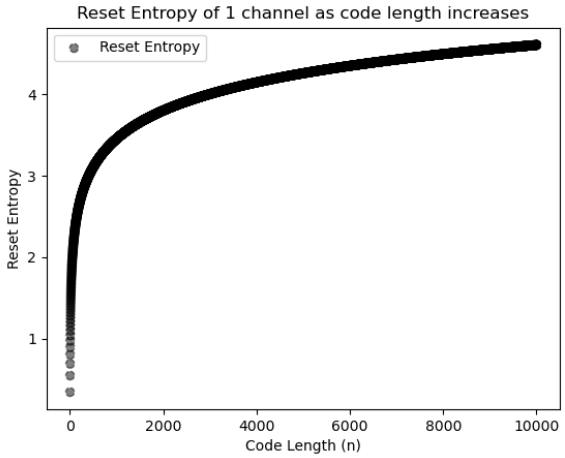
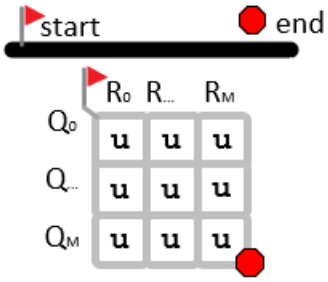


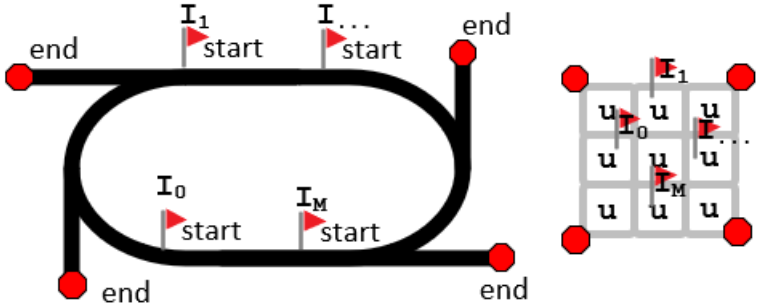
Figure 6.2: The theoretical energy (Reset Entropy) necessary to devote to reset, when n possible codewords can be transmitted on 1 wire.

The increasing entropy penalty with code length n means that for a physical implementation of a unit cell the energy cost to reset should grow quickly at first with the number of unit cells. However, as the number of unit cells increases towards larger magnitudes, the energy cost to reset the unit cells should not differ significantly. Figure 6.2 demonstrates the theoretical cost of reset that is acceptable for a race logic circuit. The solution for our circuit, to achieve closer to theoretical reset, is to move from a 1D Race logic circuit (Figure 6.3a) to a 2D design based upon an original known 1D race.

A 2D race (Figure 6.3b) is only preferable if the compute element u is an atomic, self-resetting unit, as detailed in the following section 6.1.3, as opposed to the 1D grid element originally used to construct the ASIC implementations of a 1D race logic circuit. A 2D



(a) 1D race, has a single start and end point, with stationary computation signals in vectors R and Q.



(b) 2D race, has multiple start and end points with a stationary communication system of a shared racetrack.

Figure 6.3

race needs to encode it's output into a rate, as shown in Figure 4.15, which is detailed in section 4.7. Of course, if the 2D race track is for an $|A_\chi| = 2$, the race will settle into an energetically wasteful rapid rate, as detailed in section 7. To prevent excessive energy use, it is better to use a larger value of $|A_\chi|$ for instance $|A_{simpleThis}| = 24$. However, since the ASIC implementation needs to be able to sufficiently interconnect the symbols in $|A_\chi|$, the physical connectivity concerns, explained in chapter 3, indicate $|A_\chi| < 72$, although the precise number depends on the selected process.

6.1.3 Unit Cell from Self-resetting Domino Logic Circuits

Self-resetting domino logic is a CMOS implementation that was developed with the motivation of 'speed up the slow signals' instead of the conventional approach of latch-based design which has the motivation to 'slow down the fast signals'. [156],[157]

An atomic event gate is a self-resetting domino logic circuit for emitting a spike event, oscillation of spike events or no event, dependent upon input A, A1, or more is shown in Figure 6.4 With one or more input event ports (labeled A and $A1$ in the case of 2)

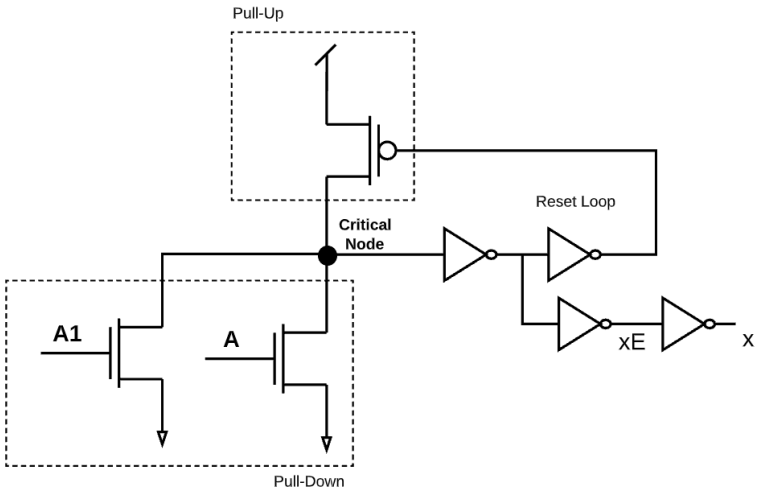


Figure 6.4: Self-resetting domino logic circuit for emitting a spike event, oscillation of spike events or no event, dependent upon input A, A1.

and an output event port x and an additional port xE for the early event, this is a minimum number of inputs for a race logic `unit cell` $\mathtt{u}_2$. When unit cells built with self-resetting domino logic are interconnected into 2D race tracks, the resulting output is a sparse spike train that still maintains entropy close to the original. A self-resetting domino logic looping race track circuit is used as the basis for our experiments with 2D race tracks shown in Table 6.1.

Race logic permits the encoding of large ($|A_\chi| > 2$) alphabet's upon multiple wires (traces) , and, additionally it permits the encoding of additional symbols as phase relations between the different wires.[13],[119] When the communication between unit cells is handled through analog-to-(non-binary-digital) (hardware-to-software or hardware-to-hardware) or (binary-digital)-to-(non-binary-digital) (software-to-hardware or software-to-software) an energy-to-entropy cell equivalence is defined as the energy and time to reset the unit cell. The circuit designer can assume 1 pulse contains 1 'unit' of information (As determined by the information source used to build the unit cell). The layer of abstraction where energy use is a justifiable stand-in for information capacity makes

a stable starting point for building low energy analog-to-(non-binary-digital) signals in hardware.

Table 6.1: Markov chains for high information capacity data transfer.

Markov chain		
Name	Description	A_χ
simpleThis	Simply Ergodic	24
biGraphThis	Simply Ergodic	16
ThisWords	Simply Ergodic	16
This	B-system ergodic	24
binary	B-system ergodic	2

The values for A_χ shown in Table 6.1 allow for generously long code words when compared to reset entropy predicted in Figure 6.2. By modifying the encoding of symbols from our dataset to fit into a binary encoding scheme with equal weight and then to further extend that into a non-binary encoding scheme based on rate, and phase we are able to generate a spike train with a higher estimate of entropy and a higher known figure of entropy when compared to binary encoding.

A high entropy spike train is valuable because it indicates there could be a large amount of information. A non-binary communication channel can seem to accept 'compressed' data, and decode into a binary memory. The data is not compressed binary data though, instead it is just an address event representation of the original data, from a known information source encoder and decoder pair, using non-binary digital coding.

6.1.4 Energy Efficiency Metric

Communications within a space-time computation system, are evaluated using an energy efficiency metric which asks how many bits of information are communicable

using a single spike event. *b bits per n spike*

$$\frac{b}{n} \tag{6.4}$$

Expanding this concept to incorporate non-binary representations, the metric becomes *s symbols per n spike.*

$$\frac{s}{n} \tag{6.5}$$

A one bit race logic cell has a b/n that scales with the grid size. Compared to neural systems, the b/n figures are astonishing. When the race is conducted on a 2D race track, with multiple traces outputting into a phase coded, relatively large symbol $|A_{simpleThis}| = 24$, from a hierarchical Markov chain, s symbols per n, it is possible to capture a single spike within a mixing time that sufficient energy must be used to generate a single spike with larger than expected energy. The result is b bits or s symbols per $n = 1$ spike is a single spike transmitting a message containing all of the symbols.

$$\frac{b}{n} = \frac{b}{1} = b \tag{6.6}$$

$$\frac{s}{n} = \frac{s}{1} = s \tag{6.7}$$

If there is only a single spike, then it must be capable of representing all of the possible symbols at once. If a single event is expected to be able to communicate all of the possible symbols at once, it must have entropy minimally equivalent to the number of symbols in the information source.

$$\frac{s}{n} = \frac{s}{1} = H_{A_\chi} = E_{A_\chi} \tag{6.8}$$

$$H_{simpleThis} = 24$$

$$H_{Thiswords} = 16$$

$H_{Thiswords}$ corresponds to the Markov chain shown in Figure 4.18.

6.2 Irreversible Computation

A theoretical method of computation called Brownian computation, relies on the idea of reversible computation.[61] This theoretical method permits the effects of thermal noise to influence the result.[61] While reversible computation is promising with specialized devices capable of exhibiting quantum effects, classical computation can also benefit from acknowledging that an irreversible cost of computation must be spent in the form of energy, but, it does not need to be spent at every stage of combinational logic. The consequences of the thermal noise transform the calculation into a random walk across the low potential energy region of the truth table. The truth table of a reversible function has an output that can be equally divided into partitions of the alphabet.[4] For example, the Boolean functions XOR, and XNOR (The inverse of XOR) are reversible, as shown in Table 6.2, *if one of the inputs is also known.*[158] The reason XOR and XNOR can be considered reversible in the circumstance when one of the inputs is also known is evident from its truth table, since in the case 2 inputs are equal results in a false output, while the case 2 inputs are inverse results in a true output.

A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

Table 6.2: If 1 of the 2 inputs of XOR is known, it is a reversible function.

With binary digital computers that operate with a state space larger than 2-bit, the question of what does XOR function computes is generally taken to be that XOR computes the parity of the decimal number that is represented by the binary inputs representing a decimal number. However, that is a software abstraction that does not hold with non-binary coding schemes. While it is acceptable to use that abstraction for software models of non-binary coding, in a hardware implementation, a non-binary XOR indicates that the order of past events is no longer information for future events.[2]

C_2	A_1	B_0	Decimal	F = A XOR B	F XOR C	mod 2
0	0	0	0	0	0	0
0	0	1	1	1	1	1
0	1	0	2	1	1	0
0	1	1	3	0	0	1
1	0	0	4	0	1	0
1	0	1	5	1	0	1
1	1	0	6	1	0	0
1	1	1	7	0	1	1

Table 6.3: Logical XOR for binary inputs numbered 2 > is often called the parity bit because it is 1 for binary digits with odd parity (number of 1's counted is an odd number). Additionally, it can be implemented as the mod 2 operator with an alternative output partition still with 50/50.

In the case of the binary alphabet, the Brownian computation is less efficient than larger alphabets because a random walk on a binary alphabet only proceeds for 2 timesteps before mixing into an irreversible state of equally divided partitions.[36] This can be demonstrated experimentally by creating an information source with two symbols and equal probability of emitting either of the symbols per timestep. When symbols are sampled from this information source, they are equivalent to Figure 2.1 where 100 observations are made from an information source with two symbols (0 or 1) and 50 of the samples are equal to 1, and thus the estimated entropy is equal to 1.0.

[2]If necessary an additional proof could be authored on why non-binary XOR indicates that the order of past events is no longer information for future events. Also, see Figure 5.1.

With an alphabet size larger than binary, there is a longer period of time when the Brownian drift is able to traverse a random walk before mixing into an equally partitioned system. During that period of mixing time there is a chance for the maximal efficiency while the thermal noise is driving the measurements from precise to unknown.[159] This is shown in Figure 6.5, where the entropy estimates from an alphabet size of 3 tend to values greater than 1.0, around 1.6 for JVHW estimate and 1.5 for maximum likelihood.

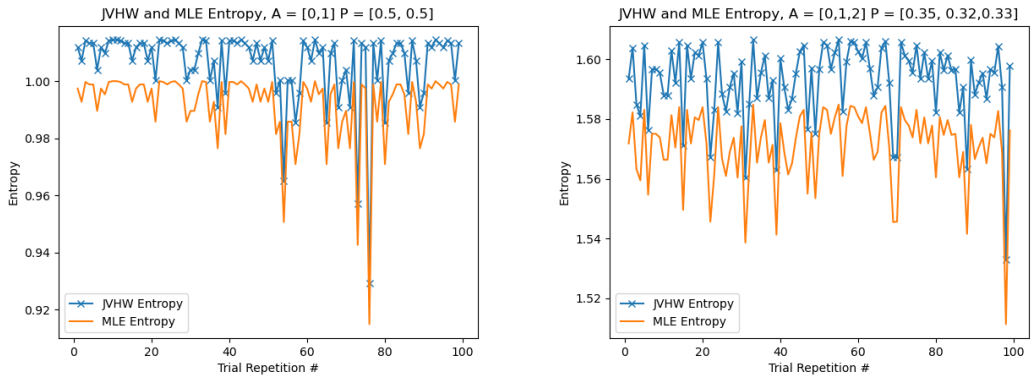


Figure 6.5: Entropy estimates gathered by drawing 100 samples from a B-system distribution of 0 or 1 values remain constant while a B-system distribution of 0,1,2 values is constant and larger.

The Boolean operator XOR is not able to operate on a physical system without timing constraints. Of course the timing constraint of gate delay t_d, is present for any Boolean function, but, it is made worse for XOR because there is not a straightforward static CMOS gate for XOR, but rather it is implemented as $F = A\bar{B} + \bar{A}B$. With a clocked computation cycle, t_d varies dependent upon the input, for instance a temporal transition of $\{A = 0 \longrightarrow A^* = 1,\ B = 1 \longrightarrow A^* = 0\}$ results in $F = 1 \longrightarrow F^* = 1$ which in theory has $t_d = 0$, but in practice is more likely to experience a timing glitch where $F = 0$ for a time less than t_d, that is propagated to the next stage of an XOR tree.[160]

The truth table from 6.2 is only true for constant values of A and B, or minimally values of A and B which are computed simultaneously through a CMOS Boolean logic

gate with a known setup time. The approximation to constant inputs and constant outputs is an assumption of Boolean combinational logic when static timing analysis is used to verify a synthesized CMOS digital design.[161]. High level synthesis of digital designs often tends to create feed-forward (acyclic) Boolean logic designs for the functional blocks.[162] Because most modern digital design tools favor acyclic combinational logic, cyclic combinational logic is frequently considered to impose additional design complexity, and is not the default design choice. [162] The difficulties with physical timing verification of Boolean XOR layout's become more complex when the numbers of bit's input into the XOR function is increased, as is the case for binary decision diagrams created at with large numbers[3] of input bits.[163] Binary digital error correction codes often rely upon low density parity check codes, which contain check bits implemented using XOR as mod-2 addition.[164], [165] When the size of the block used to generate the check bits grows large (greater than 64 bits), it is easy to find examples of parity check bits dependent upon an XOR tree of depth greater than 4 input bits.[166]

Irreversible logic is used to support an alphabet of ternary description which is permitted to contain 3 symbols $0, 1, \perp$, where $\perp$ is used to represent an unknown or undefined symbol equal to either 0 or 1.

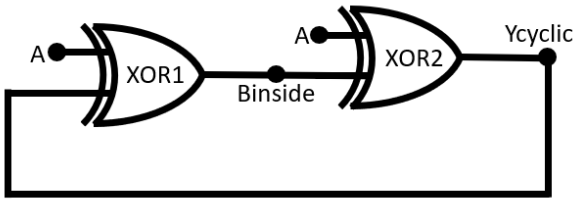


Figure 6.6: Riedel's cyclic XOR, does not demonstrate undefined $\perp$ when implemented in a physical system where noise is present.

Riedel uses the ternary symbol to demonstrate that cyclic combinational logic, where

[3]Compare to how quickly Karnaugh maps grow with size of inputs.

a single wire is able to support at one time one of the three symbols, is able to produce reduced area and reduced complexity circuits using optimal number of gates, because the current time value of the output is an irreversible computation.[162] However, an alphabet with a symbol $\perp$ is not acceptable for neural coding when noise is present in the system. Additionally, since a symbol represented as $\perp$ is not measurable, it cannot be an emitted symbol from a known information source.

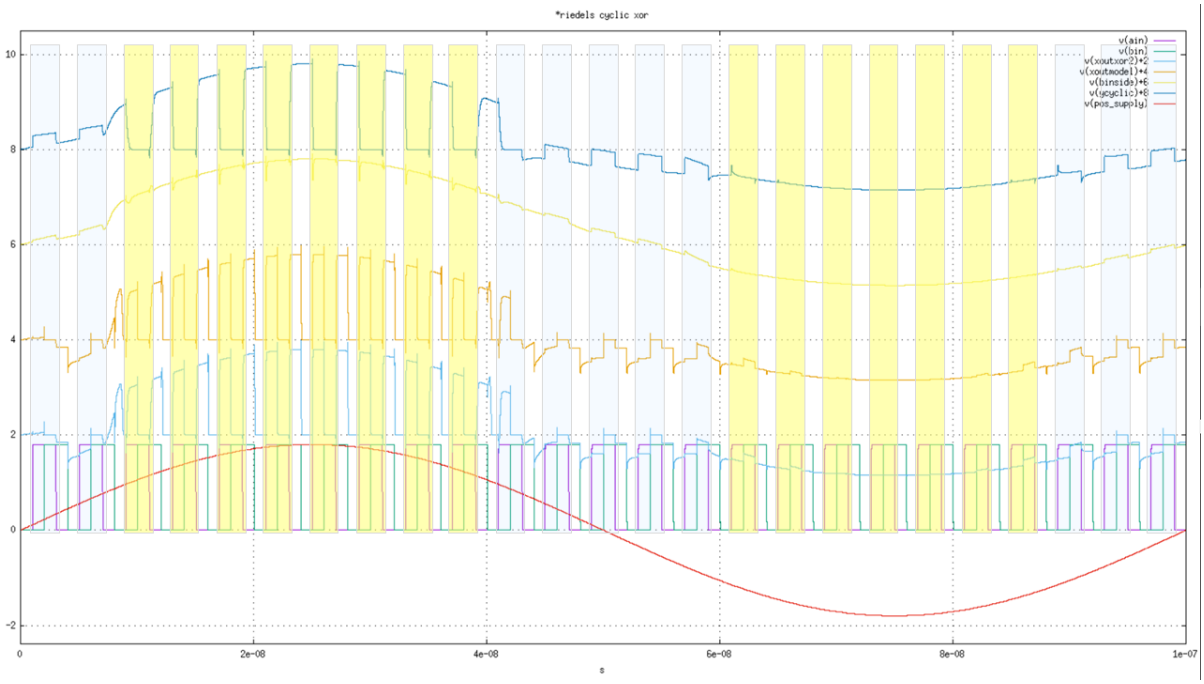


Figure 6.7: A cyclic XOR circuit is reversible, because of noise present in the system. When the power noise is varied, the output *yCyclic* is either a stable $NOT\ A$, or stable A, depending on how much noise is present.

Riedel's proof, in "Cyclic Combinational Circuits", of XOR as an irreversible element in a noiseless operating environment demonstrates '*The order of updates is irrelevant.*'[162] after multiple XOR logic operations have occurred. This argument could instead be dependent upon the existence of Maxwell's Demon, detailed in the following section, which requires erasure of the prior computation value to prevent violation of the second law of thermodynamics. The consequences of Maxwell's demon demonstrate how

if a logic gate is expending large amounts of energy without experiencing changes at the input, the output is unknown ($\perp$).

6.2.1 Maxwell's Demon and the Consequences for Computation

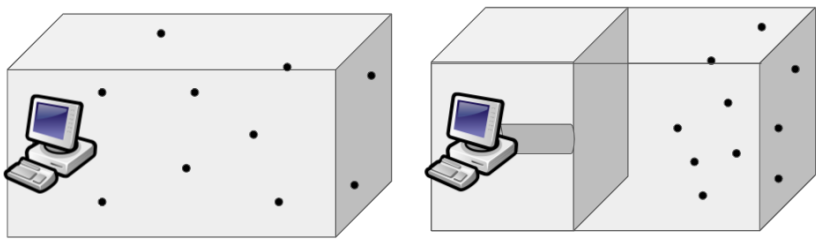


Figure 6.8: In the left box, a 1-bit computer is placed in a thermal bath. The computer is used to sense the state of the environment and when it senses the environment state is acceptable for extra work it will switch the memory. In the right box, the switch indicates it is time to place a piston into the environment with the goal of extracting useful work. The lower bound on this work is proven to be $k_B T\ ln(2)$[4] when the environment is a closed system with a 1-bit computer. For arbitrary computation size U this generalizes to $k_B T\ ln(2)$ times the Kolmogorov complexity(σ) and log(Bernoulli measure of the set of strings that compute σ) and log(halting probability for U).[5] Practical classical computers operate at an energy cost orders-of-magnitude larger than the lower bound.

Maxwell's demon is a thermodynamic thought puzzle, where 'free' energy is extracted from a system based on knowledgeable placement of a divider within an enclosed chamber. [35] Figure 6.8 shows an enclosed system at two different points in time. At both times there is a computer present in the room that is able to sense the state of the environment. The computer represents the fictitious Maxwell's Demon, which is able to extract 'free' energy from the environment, until the energy lost to store the memory of the environment state is accounted for. [35]

Maxwell's demon is present in the design of information systems through the concept of reset.[61] A computer architecture relies on the concept of reset to make sure all of the state variables are in a known state after start-up. Reset could be thought about as

setting the bit in the Demon's memory, which to resolve the additional energy the Demon would be able to extract from the environment, requires an expenditure of energy to erase the Demon's memory.[167]

6.2.2 Noise in Race Logic

An unexpected use of energy in a computation is noise. In an ideal 1D race, with a fixed R and Q, it is expected that the dynamic propagating wavefront of the race, as the formerly reset to 0 1D grid, is completing the rising edge operation $0 \rightarrow 1$ will take place noiselessly. For a very small grid, this is indeed the physical case, because for a small enough circuit power rail noise or process noise are not noticeable concerns. For a static CMOS circuit in 130nm, with a typical voltage on the power rail of 1.8V, the power rail needs to drop down closer to 1.0V, as shown in Figure 6.9, before the switching velocity of the gate is causing a significant enough timing voltage variation to change delay based information signal that a 1D race logic circuit is propagating.

Process noise is the instability in the reliability of the rising edge caused by the fact that a hardware circuit is made of many numbers of physical elements which are all intended to be repeating the same Boolean operation. Again, when the number of physical logic gates is small, in comparison to the number of physical logic gates used in the specification of the process, the overall process variation is not a noticeable contributor to noise. Once the grid size grows large, there tends to be blobs of process variation across the grid. For static CMOS, this type of noise manifests as a portion of the gates switching slightly slower against the rest of the gates, and, in a synchronous system that obeys the dictate that the slowest gate sets the clock, it necessitates and overall slower clock rate to prevent timing errors. Asynchronous Impolite pulse logic (section 6.3) uses pulses, which are more noise resilient to process noise than static

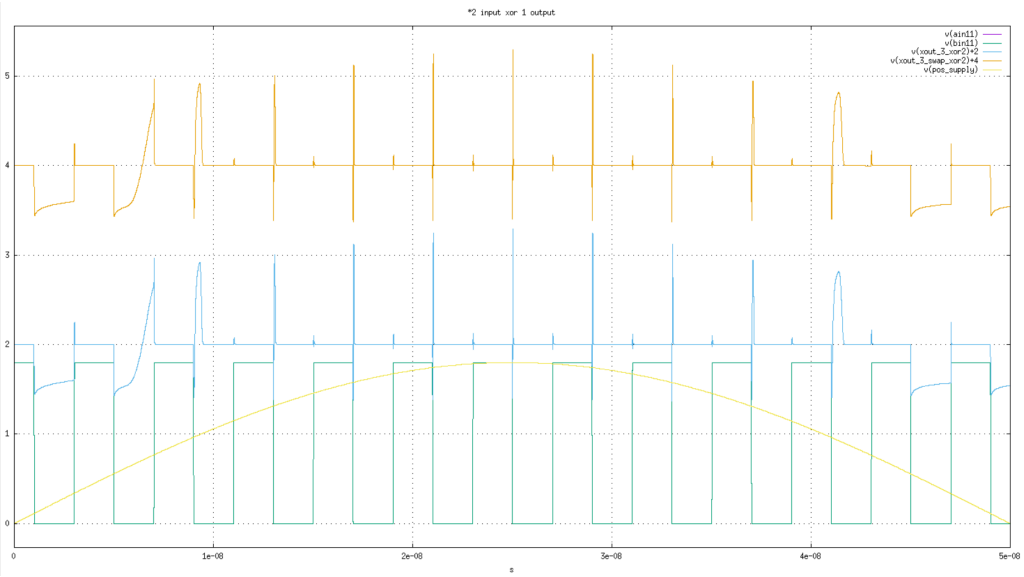


Figure 6.9: The switching velocity of a static CMOS XOR gate in 130nm begins to degrade when the power rails are failing and only operating at close to 1/2 of the original expected voltage.

CMOS, however, there is still a spread in operating speeds as shown in Figure 6.10. This can be managed through replicating the unique structures on chip, using Shannon's coding theorem, that repeating the message more than once is always a possible solution to categorical noise.

Stationary computation begets noiseless communication, because the states the machine is able to transition between are guaranteed to each represent a time allocation equal to the rate of measurement distributed over $1/n$. Since it is possible to represent a computation as a stationary program, when the computer is actually moving information across a tiled grid, then the inverse should also be true and the communication could be the stationary program. The propagating race wavefront is a source of dynamic noise as a row and a column tend to switch simultaneously, as shown in Figure 6.12. The propagating race wave is the source of computation. It is not noise. When the race wave is a changing computation result sustained by the dynamics of the 2D race tracks as shown in Figure 6.3b and Figure 8.4, the multiple entry and exit points cause the

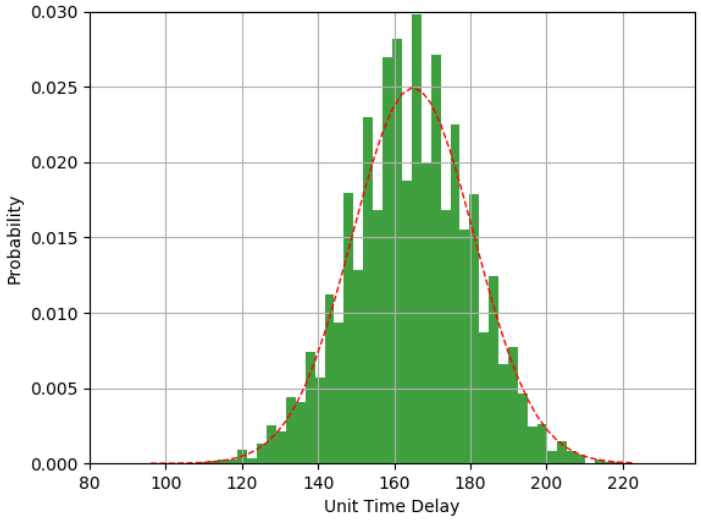


Figure 6.10: GF 130nm Process Noise from 1000 Monte Carlo runs.[6]

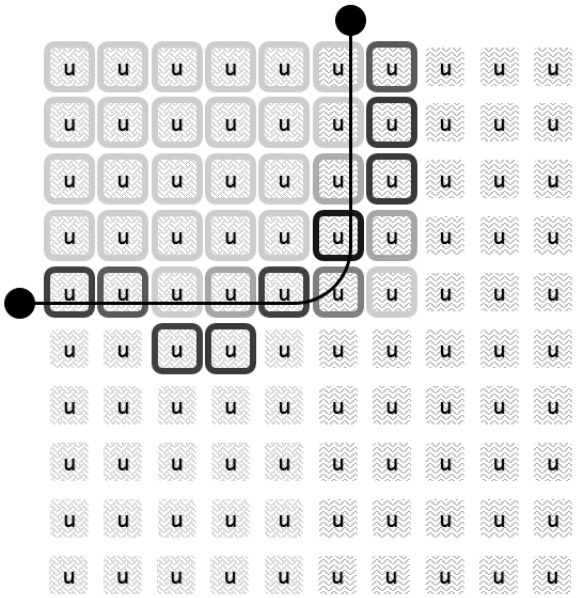


Figure 6.11: Power noise shown by the weave/zigzag texture partition and dynamic wavefront noise shown by the dark/light perimeter, manifest errors in the calculation. The propagating race wave (thin black line) is the source of the computation, and the noise is directly because of the computation signal. If there is no computation or race underway, then there is 0 dynamic wavefront noise.

race to constantly operate, and if the input is fixed and the noise is constant, eventually the race will output a stationary signal repeating the spiking channel communication representative of the input. Even if the stationary computation is not as expected, the deviations are both the signal and noise, when energy use is measured for known times, as shown in Figure 4.7.

A reasonable conclusion is that as the grid size of a 1D race logic circuit grows, power noise is the most significant contribution, followed by process noise, and finally dynamic wave front switching noise.

When it comes to a 2D race logic circuit, where a racetrack is implementing one of the possible gram's from a Markov chain for a dataset, for instance, for *simpleThis*, an information source containing all of the possible monograms would contain 283 grid tiles of unit `u`, where the dendrite tree arrangement specifies which population code is generating from each tile.

Then a monogram information source must contain 28 `u =` (u equals space), 16 `u = e` (u equals lowercase e), ... and so on including 5 `u = o` (u equals lowercase o) ... all the way through to the 3 monograms which are only present through 1 tile, (1,4,f). The monogram racetrack which interconnects for the monogram `o` is shown in Figure 6.12. The number's (28,16,...,5,...,1,1,1) come from the dataset, detailed in section 8.4.

6.3 Asynchronous Impolite Communication Links

6.3.1 Impolite Design Methodology

Impolite asynchronous pulse circuits do not rely on the traditional asynchronous strategy of hand-shakes.[168] The Impolite methodology is a practical approach for energy-efficient, high-performance design. The approach allows interfaces to run faster than core

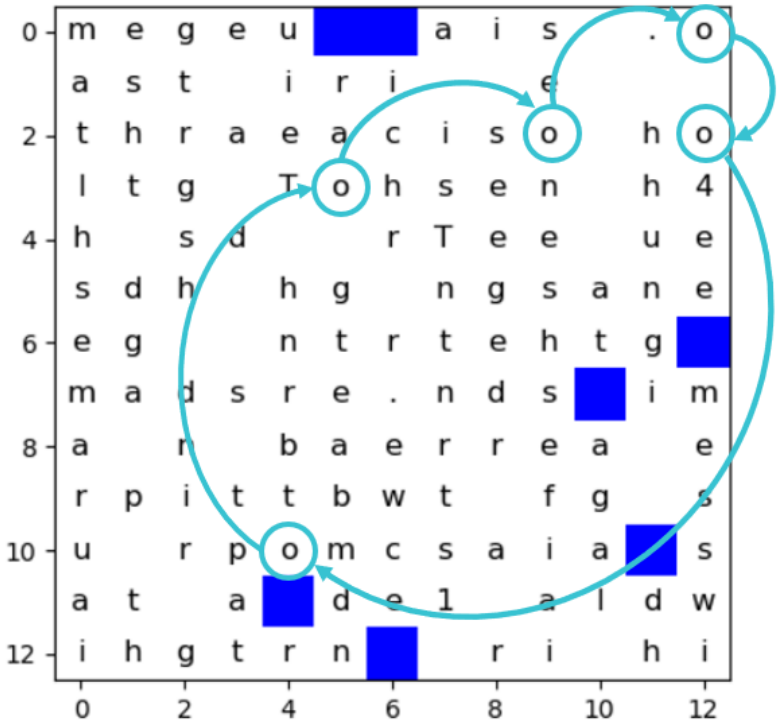


Figure 6.12: A monogram information source for simpleThis, with a 2D racetrack for monogram symbol 'o'. (The lower case letter o.) The noise susceptibility of a 2D race track is reduced, because the energy based monograms are distributed across the entire grid of possible monograms, thus the effects of power noise and process noise are equally likely to contribute to uncertainty in the phase for any of the 24 monogram tiles.

logic and enables interfaces between structurally different finite state machines. Timed asynchronous automata can use high-frequency signals without the overhead of high-performance clock distribution and recovery.

The methodology is an adaptation of existing asynchronous design approaches, specially targeted to the production of high-rate circuits embedded in slower fabrics.

Circuits with such a rate disparity are common in sensor data links, where the slow circuits sense data and the high-speed communication system transmits the results. Fast and slow rates are especially useful in neuromorphic systems, where the activity of the communications is known to take place in bursts.[98] Impolite methodology eases the design and extends the capability of neuromorphic systems.

Interfacing between high and low speed clocking regimes is a common design problem in integrated circuits where serial interface and communication speeds frequently exceed core clock rates. Complex communication link circuits involve operating a metal wire near the rate where substantial amplitude loss and symbol timing jitter are serious issues. Correction of symbol-dependent timing errors are difficult to mitigate.

Synchronous circuits have limited capacity to handle timing issues. Specifically, time domain synchronization is difficult, and is commonly relegated to specialized blocks, such as DLLs, PLLs, and skew compensators, whose behaviors are outside of the synchronous domain.

The presented design implements asynchronous logic blocks with a style to handle the slow-parallel to fast-serial domain interface independent of the behavior a timing circuit.

Timed asynchronous logic increases tolerance of timing variance. Fully unconstrained asynchronous design has high design and verification complexity. A solution is a set of composition rules for pulse gate logic that allows a limited set of classical timing constraints to close both the low-speed and high-speed design behaviors.

The "Impolite" scheme often uses feed-forward construction, meaning performance with picosecond timing resolution is possible. Feed-forward logic does come at a cost: the timing of the system needs to be verified as part of the construction procedure. One of the novel contributions of our methodology to the research community is the means to limit the complexity of the timing verification.

6.3.2 Pulse Signals Instead of Edges

Pulse signals (as opposed to edge based signals) are used to communicate and perform computation in neurmorphic signaling systems. The pulses are communicating timing critical events. The one-at-a-time pulse model makes the event timing check, *pulses must*

not overlap for correct behavior, except in the condition when the selected neural coding strategy permits codes with both variants. Pulse signaling with self resetting buffers permits an event cycle rate as fast or faster than edge event rates.

Jitter and propagation behavior of pulses and edges are nearly identical for practical interconnect cases. Pulsed signals, with a single characteristic width, allow the use of pulse gates, such as in [169, 170, 171, 172] and similar to [173]. These gates are known to maintain stable, narrow pulse widths, whereas logic without feedback would accumulate edge-to-edge uncertainty, widening the minimum pulse width.

In edge-based signaling, a rising edge must be followed by a falling edge, requiring separate types of event detection. Since different devices are involved, systems relying on edge-based signaling have inherently higher sensitivity to process variance but have been used successfully (e.g.) [174].

Since edge-based communication has a theoretical advantage in both power and bandwidth, it is important to demonstrate that pulsed signaling does not come at a high cost relative to edges in practical on-chip design. A case study using a 130nm process node interconnect demonstrates that pulse signals have similar energy cost relative to edges in practical on-chip design. The 130nm process node is used to demonstrate, because it is in this process node and smaller, where wire dimensions became a limiting factor[175] in signaling rate.

Case Study 5mm wire 130nm Process Node

A 5mm wire in a metal layer of a 130nm process is used to compare pulsed and edge encoding for an event signal. The results for propagation time of an edge and a pulse are shown in Table 6.4. The metal layer has typical conductor thickness of $.3\mu m$ and a inter-layer dielectric thickness of $.3\mu m$. The wire width and spacing is selected to optimize the cost function of $delay \times wire\ pitch$. The optimized wire width is $.55\mu$ and the wire

spacing is .38μ. This configuration has fringe capacitance of $100fF/mm$, a side coupling capacitance of $31fF/mm$, and, assuming a copper conductor, a resistance of $133\Omega/mm$.

With an inverter size of 26μ NMOS, minimal worst-case delay, with an even number of stages, occurs with 4 internal repeaters meaning there are five, 1mm long wire segments. The single-stage worst-case delay time constant between two repeaters in this configuration is 68.4ps.

	(100ps rise) Edge		(165ps width) Pulse	
Coupling noise	Average	σ	Average	σ
No coupling	251.3 ± 1.0	8.6 ± 0.7	239.5 ± 0.9	7.7 ± 0.7
Fastest	215.6 ± 0.9	7.5 ± 0.7	208.5 ± 0.8	6.9 ± 0.5
Slowest	285.9 ± 1.1	9.5 ± 0.7	276.7 ± 1.0	8.7 ± 0.6

Table 6.4: Propagation times for edge and pulse shown with 95% confidence interval marked.

Table 6.4 compares the propagation times for edges and pulses. Using 165ps as the full-width, half-maximum measure of the pulse, propagation times are similar to edges. The minimum pulse period is twice the pulse width – 330ps, marginally faster than the non-skew-compensated rate of edges.

Edge-communicated Signal

The arrival jitter is approximated by Monte-Carlo simulation consisting of 1k runs, sufficient to gain 95% confidence value. Process variation is taken from a vendor (IBM) model for the 130nm process. Power variation is estimated to have a global, correlated variation of ±30mV power to ground to model power regulator noise. A local, uncorrelated variation of noise, ±30mV is added to each power and ground node, modeling IR noise internal to the IC.

When there is a single fast edge ($< 100ps$) of a slow signal ($f < 200MHz$) the average (across process and voltage variation) propagation time is projected to be $251.3 \pm 1.0ps$,

close to the value that $\#stages \times stage\ delay \times ln(\frac{1}{2})$ predicts. The sample standard deviation (σ) of the arrival time in this experiment is $8.6 \pm .7ps$. Assuming a Gaussian distribution, a 5σ interval gives a delay between $203.5ps$ to $298.5ps$ for a 5mm wire, considering only process and voltage variation.

Due to the high coupling capacitance (˜38% of the total) the impact of neighbor wires must be considered. The worst-case jitter occurs when both neighbors are correlated in the same or opposing direction as the main signal. Same direction switching has delay $215.6 \pm .9ps$ with a sample σ of $7.5 \pm .7ps$. The 5σ fast arrival time under these circumstances is $173ps$. Opposite direction switching delay is $285.9 \pm 1.1ps$ with $\sigma \approx 9.5 \pm .7ps$, giving a 5σ slow case of $338ps$. This is a range of $165ps$ of environmental jitter that an ideal latching strategy cannot compensate for. A more common single clock phasing strategy would need a flip-flop with $\tau_{su} + \tau_{clk-Q} < 162ps$ to run at 2GHz.

Pulse-communicated Signal

Jitter for the self-timed buffer case is slightly higher than for the inverter buffer case at an estimated total jitter (5σ+pattern dependence) of $188ps$. This extra jitter, as compared to the inverter buffer case, is due to the extra logic required in a self-resetting buffer. This extra logic extends maximum propagation time, in this case to $405ps$. The arrival order uncertainty limit of $188ps$ is the dominant of the two restrictions.

Pulse vs. Edge for Marking an Event

The relative propagation timing of pulses and edges are very similar, even when the width of the pulse is small. Jitter dominates gain/bandwidth in limiting performance, and a pulse because it is atomic and unambiguous in its arrival, need not be correlated to any other signal. Thus pulse-based event detection can operate as fast or faster than edge event detection correlated to another signal or state.

Using self-resetting gates to create a regenerative buffer, improves performance. Self-resetting gates protect pulse widths, and thus jitter cannot destroy a pulse. For systems using self-resetting buffers, two conditions must be met: First, the pulse width and its reset time must be obeyed. Second, the pulses must arrive in order. The pulse width is set locally within the buffer since it is self-timed. Native self-resetting pulse width in 130nm was determined to be $64ps$ with $\sigma \approx 4ps$, giving $84ps$ for a maximum pulse width, and $168ps$ for a safe pulse interval.

Table 6.5: Event detection times for various signaling methods

Method	Period	Notes
Handshake	597p	Interleaving forward & backward wires (minimizes worst-case propagation)
Clocked Edge	338p	No Phase compensation
Edge with DLL or PLL	223p	Requires DLL/PLL of ˜20mW (assumes 15ps RMS jitter ≈ -112dBc/Hz phase noise)
Pulse	*330p*	*Not using self-reseting buffers*
Pulse	*188p*	*With self-reseting buffers*

Table 6.5 compares five methods for event detection and the associated minimum period of operation for a BER of 6×10^{-7} (corresponding to $\pm 5\sigma$ variance). For the purposes of comparison, the minimum latch timing is left out of the presented period. In a clocked system, the latch sample interval adds 50-200ps. In asynchronous systems, both pulsed and handshake, latch sampling time need not add to the minimum period, as both techniques have sampling times built into their respective operating mechanisms.

6.3.3 Composition Rules

The following construction techniques give rise to a class of systems that are delay sensitive, and are easier to verify than a system completely free in timing specification.

The delay sensitivity means timing verification is required for behavioral closure for

timing critical circuits. The construction only admits designs that can be verified with static timing. Static timing analysis is a common part of verifying clocked systems and numerous extensions, such as yield estimation, have been formulated[176]. This method uses a constraint system with the same analytic methodology, similar to clocked SRCMOS[177].

Signals are partitioned into two classes or types: Events and Data. The Event class serves to mark time, and is analogous to a clock. The state of a gate can only change with an event, communicated by a pulse of fixed width. The other class, Data informs state updates in the presence of an event. Data is communicated as traditional digital levels.

Basic Rules

There are a handful of rules to enable simple, closed examination of these event-driven circuits. The circuit can be specified hierarchically, where each level of the hierarchy obeys these rules. For the purposes of discussion a *Block* is a functional component at one level of the hierarchy, while a *Gate* is specifically at the lowest level. The rules for blocks are as follows:

1 There is a strong typing of *Event* and *Data* signals. A signal will be of exactly one of those two classes.

2 Specification, and verification occurs within a time *Frame*. The time frame is marked by a start event and a finish event.

3 Within a *Frame*, Data values will update at most once.

4 For any *Block*, and most *Gates* only a single *Event* can occur at a time.

5 When an event occurs, Data cannot be in transition.

6 The *Consensus* gate is exempt from rule #5. The behavior of this gate is handled as a special case.

The Frame requirement from these rules enforces a data-path check that is similar to traditional clocked logic. The model fits well with a system matching high-rate and low-rate regimes – the low-rate timing can easily provide the Frame.

Timing Check Complexity

For this acyclic-within-frame logic, the complexity of the static timing check is simply bound by the number of delays along the path and thus scales as $\mathcal{O}(n)$ given n delay bearing nodes including latches. For a circuit with m unconstrained events $m!$ constraints are needed in worst case although this is practically limited by gate fan-in. The basic rules are placed on gates to ensure predictable (inertial) delays apply and have complexity $\mathcal{O}(m^2 \cdot n)$. This has the consequence that some kinds of circuits fall outside of the constraints. However, circuits including near-miss arbiters and all circuits necessary for communication links are permitted.

Logical Constraints on Construction

The first logical constraint is that *the value of a Data signal must be stable between the setup and hold time* for each gate relative to the arrival of an updating event.

The second constraint is that a block interface may have only *one active event at a time*. One-at-a-time events prevent complex timing issues from arising within the relative timing check. In order for two events to be processed simultaneously they must act on separable parts of the system, or be passed into a Consensus gate for processing.

For two events to inform interacting parts of the system they must have a fixed timing relation. The relationship can be confirmed by verification, and event re-timing can be

part of the construction. The designer should describe any two-event behavior as a set of one-at-a-time actions. Fair arbitration is not a feature of the methodology, instead the designer must describe the arbitration method. This can be aided by a Separating Arbiter gate, that while it cannot perform ideal arbitration, it can separate vanishingly-close (1ps) pulse interactions.

Because of these interaction rules the timing can be checked in $\mathcal{O}(m^2n)$ time complexity, involving two different timing checks limited by this complexity class. The timing checks are of $Event \rightarrow Event$ relationships, and $Event \rightarrow Data \rightarrow Event$ relationships, since no more complex relation can exist. The Event simultaneity check needs to be performed at each gate to confirm that no two events arrive simultaneously; This allows the event-as-a-clock equivalent model.

Then data timing is checked on $Event \rightarrow Data \rightarrow Event$ sequences, similar to a clocked system where the sequence is $Clock \rightarrow Data \rightarrow Clock$. Since events are one-at-a-time, event-data and event-event checks are orthogonal.

In practice, the timing check is further simplified by the limited number of events used and the limited fan-in and fan-out of practical designs.

Event timing

The one-at-a-time pulse model makes the event timing check the dual of the SRCMOS timing check, *pulses must not overlap* for correct behavior. Pulse Gating is done with a self-resetting gate structure and data latching utilizes a Set-Reset latch style. The pulse arriving at the gate serves as the sampling aperture for the pull-down network; this sets the data hold window to the actual pulse width of an event. The minimum pulse-width is thus set by the need to reliably sample, and is a trade-off with the complexity of the pull-down network.

There is a special case to the non-overlap verification: the Arbiter circuit. An arbiter

circuit is allowed, logically, to have two events occur at its inputs as long as the events are of known order and avoid exact overlap. This non-overlap constraint is of practical importance, as arbitration circuits near meta-stability have ill-defined behavior and can take an infinite time to resolve[178]. In the contexts of these checks, arbitration is reduced to describing near-miss cases.

6.3.4 Gate Construction

The strict classification of signals is crucial for timing analysis and simplifies design of the gates. This structure allows a gate to act on a given event given a set of data guard values since the data is known to be stable on event arrival. Logic functions are incorporated into the front-ends of Latches and pulse Gates, giving fast, small designs.

There are two classes of gate in this construction paradigm, pulse *Gates* and *Latches*. A Pulse gate shown in Figure 6.13a, has a spike output, and is used to create *Events*. Latches, shown in Figure 6.13b, have a level output creating *Data* signals. Pulse gates use a self-resetting logic to recreate the event pulse.

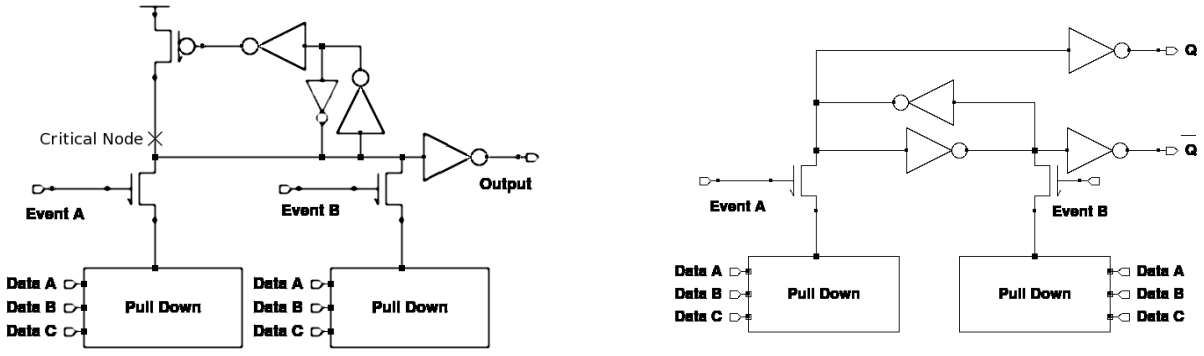


Figure 6.13: (a) The pulse-gate, outputs a pulse given an event-data condition. (b) Pulse-Triggered SR Data-Latch - outputs a data level that changes on a pulse given a condition

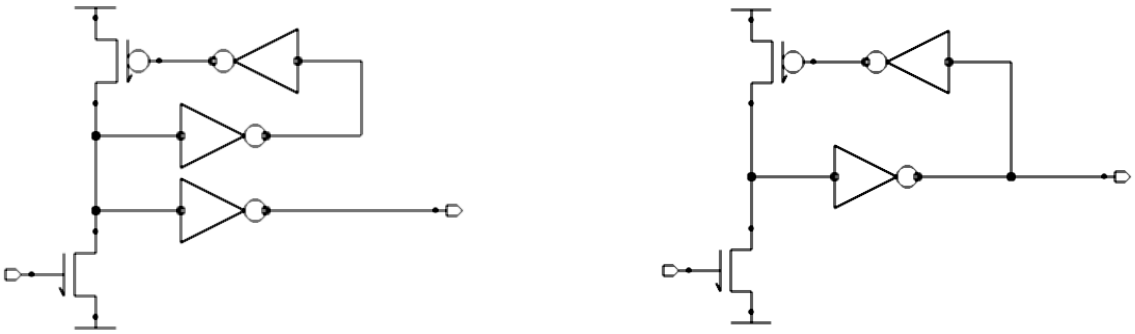
Pull-down network timing

The structure of the pull-down network, combined with the typing rules for Events and Data, create a timing constraint set for each pull-down network. The behavior of these networks are similar for pulse gates and latches, and the analysis holds for both. Correct functioning of the pull-down network determines the values for the timing constraints in the composition rules (Section 6.3.3). The prohibition on event overlap ensures that the action due to any event is not preempted by another event, a critical requirement for timing verification to not be forced to check all event combinations. Consider the timing of as SR latch, like the one shown in Figure 6.13b. Event A triggers setting this latch, while event B triggers reset. In both cases, the action is contingent on the logic of Data A, B, and C encoded into the respective pull-down networks. Electrically, the pull-down network is assumed conducting or non-conducting when the associated event pulse arrives. In the case of the SR latch, the set condition must be stable for the set pulse and the reset condition must be stable for the reset pulse.

Drive and Feed-back network

The characteristic pulse width of a self-resetting gate is set by the propagation delay through the feedback path. The output amplitude is dependent on both the output driver and its load. Pulse detection requires that the delay of the feedback path must match or exceed the amount prescribed by driver and load environment.

The feed-back network for a pulse gate can either be connected or isolated from its load as shown in Figure 6.15. The isolated-load model (Figure 6.14a) is faster than the load sensitive feedback and has more reliable timing. This feed-back network is well suited when the gate load is either constant, so that a correct pulse width can be selected, or for cases with small fan-out where pulse attenuation is not a concern.



(a) Load isolated feed-back gate.

(b) Load sensitive feed-back gate.

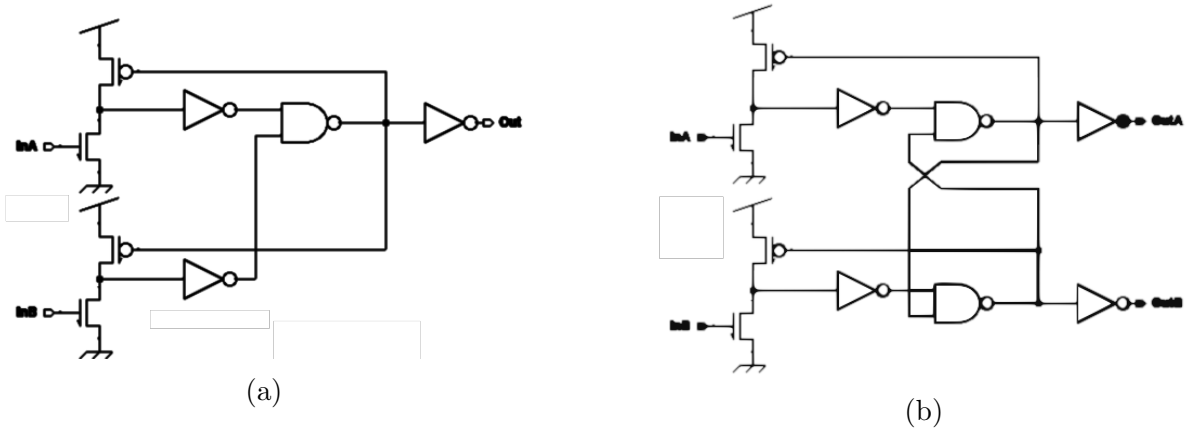
Figure 6.14: Two self-reset feed-back network options.

Load sensitive feedback (Figure 6.14b) is useful in a production environment where more universal cells are desired. The load sensitive gate's pulse-width and timing are derived from an estimate of pulse detectability. For reasonable levels of output loading, the feedback increases driver on-time until the local detectability threshold is met. This model keeps the impolite fire-and-forget model, by approximating detectability, rather than using a handshake. To illustrate the difference, we consider two different cases that arose in real designs executed in the 130nm process node. First with $7fF/\mu_{driver}$ of load capacitance – a typical value for timing-critical, local wires. In this case, the two feedback networks perform similarly. In the second a load of $50fF/\mu_{driver}$ represents a typical drive configuration for moderate-distance interconnect or a heavily loaded reset signal. In this case, load-sensitive feedback produces a slower, longer, pulse. This results in a pulse that is nearly 4x as detectable.

Two-Pulse Gates

Gates where two pulses arrive with the potential of overlap are special cases. There are two types of these gates: The *Consensus* gate, which produces an event after seeing two events is shown in Figure 6.15a and the *Arbiter*, shown in Figure 6.15b can resolve

pulses that are near complete event overlap (<5ps timing difference).



(a) (b)

Figure 6.15: (a) Pulse Consensus gate. Gate fires output `out` once at least one pulse arrives on each input (since the last firing). (b) Separating Arbiter gate. Gate recreates input pulse sequence with restored separation.

The Charlie diagram, a visualization of output timing vs input timing[179, 180], for these two gates is shown in Figure 6.16.

It is important to note the unstable behavior of the arbiter circuit near complete event overlap (<5ps timing difference). This establishes the limiting constraint for event-event timing checks. Two architectural solutions exist for cases when operating in this regime. First, is to insert arbitration up-stream to correct order. Second is to design a system that is tolerant of this hard-to-arbitrate condition, for example creating a parallel data-path dependent on the consensus gate that will time-out an ambiguous arbitration.

Pulse Gate Timing Stability

Pulse gates have a higher timing stability when compared to standard CMOS gates. This phenomenon can be be seen in the high timing stability of ring oscillators made form pulse gates[181]. The stability of the timing may be due to lowered noise in MOS transistors for inputs in some state transitions[182]. In the case of pulse logic decisions are made by NMOS transistors turning on for all decision cases. To confirm the applicability

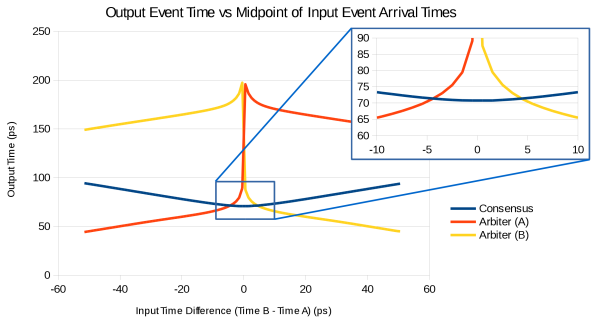


Figure 6.16: Consensus and Arbiter gates

of these results to the case of logic cells, timing is confirmed for the well-characterized 130nm process node, using noise models that take bias-condition into account. Figure 6.17 shows the timing stability of post-layout performance of pulse logic as compared to foundry-provided cells – in this case a buffer cell for driving a long wire.

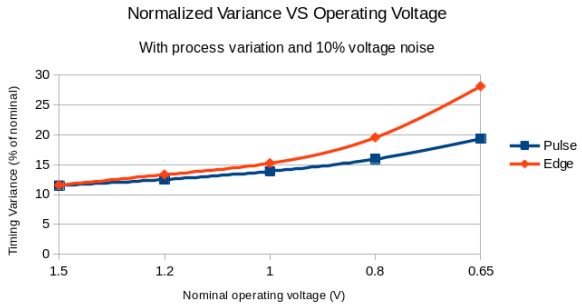


Figure 6.17: Variance of Pulse Gates, CMOS gates. Pulse gates have much lower variance at low operating voltage

Pulse Gate Library

Table 6.6 shows timing from a number of pulse gates. Computed from post-layout extracted designs. The gate set is an equivalent of a standard cell set for pulse-logic, and is used to build the designs of Section 6.3.5.The pull-down networks are sized for equal on current. The gate measured by simulation is triggered by a gate with similar drive (Fanout of 1) and loaded with a 4x pulse-repeater load (Fanout of 4, FO4). The decision

threshold for measurement is set at VDD/3, an approximation of the pulse-gate decision threshold.

Table 6.6: Pulse Gate Timing in 130nm process

Gate	FO4 Delay	Delay Std. Dev.
Pulse Repeater	29.9ps	2.2ps
Or of two pulses	31.9ps	2.7ps
Pulse Single-condition	31.5ps	2.7ps
Pulse 2-Condition AND	29.4ps	2.6ps
Pulse 2-Condition OR	29.1ps	2.3ps
Or of 2 pulses, w/conditions	33.3ps	3.5ps
SR Latch	37.0ps	3.3ps

6.3.5 Data Link Performance Estimation

A hypothetical 4-bit serializer(SER) and deserializer(DES) are analyzed to demonstrate the verification procedure and construction methodology For a high-speed interface. The 2-line encoding of [170, 171] is used. The SER uses 4 delayed copies of the data valid signal, which need not be locked to any other circuit. The DES produces 1 data vector per packet and a single data valid pulse. Given these constraints the timing results computed are shown in Table 6.7.

Table 6.7: Performance Estimates for 4-bit SER/DES system

	130nm	65nm	45nm
Trigger Time (Pulse Width)	93.8ps	52.1ps	23.0ps
Max Bit-Bit Time	187.6ps	104.2ps	46.0ps
Minimum DDR Clock Period	592MHz	960MHz	2.17GHz
Max Data Rate	4.74Gbps	7.68Gbps	17.4Gbps
Comparable Max Data Rate	2Gbps[183]	4.8Gbps[184]	10.5Gbps[185] [4]

Static Timing Analysis Complexity

For the serializer(SER) and deserializer(DES) from Section 6.3.5 there are 9 blocks in the DES and 13 blocks in the SER. For the SER there are 16 events, while the DES has 10 events. The composition rules require checking events at each block. A simplistic algorithm can verify timing with only 4228 path sums. This check not only allows verification with much less computation than a mixed-signal simulation, but also allows designers to use asymmetric or varying components without resorting to an exponential number of cases that need to be checked. Figure 6.18, the timing $\mathcal{O}$ complexity for a number of cases, demonstrates the scaling of the checks. The rules and verification become an enabling technology for systems checked in $\mathcal{O}(m^2n)$ time, compared to free-for-all asynchronous systems bound by $\mathcal{O}(m!n)$.

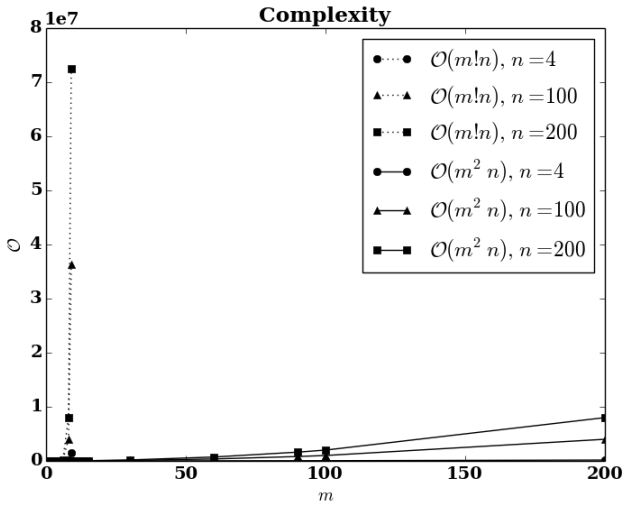


Figure 6.18: The limited complexity of "Impolite" methodology shown by solid lines extends nearly horizontal vs the delay insensitive timing checks growing exponentially.

[4]Design uses even and odd channels, and when combined are limited at 21Gbps

6.3.6 Benefits of Impolite Methodology

Since the "Impolite" methodology permits the exploration of a design space containing a much larger number of events, it is practical to consider what those sorts of systems could be developed to solve. The most obvious use is the embedding of simple mathematical functions in communications.

The Internet of things and the growing influence of practical artificial intelligence offers a platform where localized computational models can be of great benefit. The communications between these many diverse systems will require many channels variable, and burst data rates as well as unique light-weight interfaces. The logic family presented here shows promise for creating unique, and possibly asymmetric solutions that remain easy to verify.

Additionally, this composition style can implement Race Logic, which shows promise as a new computational model[12]. The required temporal functions (MIN, MAX, and COMPARE) are permissible using the composition rules discussed in Section 6.3.3.

Table 6.8: Events per Frame for different fast-slow systems.

Slow Rate	Fast Rate	Events per Frame
200 Mhz	1.2 Ghz	8
200 Mhz	2.4 Ghz	12
80 Mhz	2.4 Ghz	30
50 Mhz	3.2 Ghz	64

A practical method to efficiently construct self-timed feed-forward gate circuits was described. The method consists of two classes of self-resetting and latched gates implementing all logic switching in pull-down networks for performance. Circuits are composed using a few composition rules, and a small number of static timing checks suffice to verify functionality. Since the gates are self-timed and the system allows for safe forward composition, very high performance networks can be designed in small footprints with rather

large tolerance to timing and process variance. The technique is shown to produce simple serial links which scale well for implementation of on-chip networks without the need for conventional low-jitter circuits. Perhaps most importantly, the realizable circuits have behavior that is fully verifiable though conventional timing analysis, and this analysis has simply bounded, low complexity. It is important to note that the composition rules force the designer to practically deal with arbitration issues, but greatly extend the timing verifiable reach in terms of simply achievable performance.

In short, Impolite Asynchronous Pulse Logic enables verifiable construction of high-timing-performance circuits from small cells and simple composition rules. The fire-and-forget model, as well as timing rules enable static-logic style checking at decision rates typically reserved for mixed-signal style design. The resulting circuits maintain precise timing and have good idle behavior.

6.4 Spike Coding for Noisy Digital Computation

Noiseless digital computation is achieved through Boolean logic with binary digital signaling. The 16 fundamental Boolean functions which operate on 2 input bits and output 1 bit are used to compose input output functions with larger numbers of inputs or outputs. For instance, the classic mathematical operation 'ADD' takes as input 2 bits and outputs 2 bits, 1 bit representing the sum and a second bit representing the carry. The description of that 2 input to 2 output function is drawn as a circuit schematic which can be translated into many different physical implementations, that, as long as they are operated within the specifications determined by the physical implementations parameters[5] can be taken to be a faithful reproduction of the original ideal mathematical function.

[5](temperatures near room temp, power supplies that are constant, environment with average levels of radiation, not at extreme pressure etc...)

The advent of digital communication was achieved by building small, error free binary switches for small, easily understood functional blocks of logic design. The Boolean functions are guaranteed to make use of 2 discrete digital symbols, and a 2 input XOR function is used to add additional information about the original inputs's back into the output to allow the incorporation of error checking bits. If more than 1 copy of the 2 input XOR output is included, it becomes possible to both check for errors in the output and to correct for errors in the output. Enough copies of the XOR output is all that is necessary to create a guarantee of noiseless digital binary computation.

6.5 Error Correction

When selecting how to allocate code space between the three base coding strategies available in a neuromorphic system, there are a number of rate codes which, if chosen, are able to produce moderate mistakes without repercussions because they do not cause the outcome to change. Error correction and its utility changes as time passes from the first received message. If the method of neuromorphic encoding is completely unknown then the first amount of time steps to record over must be at least 4 timesteps and across 2 channels. This is shown by the following experiment which asks the viewer to draw a conclusion about a depiction of data, Figure 6.19, which must be viewed with **NO assumptions**.

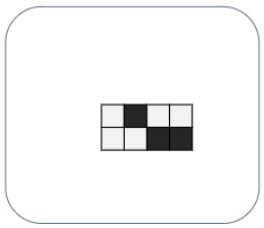


Figure 6.19: At least 8 measured data samples are needed before it's possible to theorize about what sort of information a neural coding strategy unknown data might contain.

If we have totally unknown data, we could still analyze the data against itself, if we split it up into smaller segments.[96], [186] There are multiple ways to partition the data, and if we assume the difference between the two colors are 'spike' or 'no spike', there are some possible patterns where a halfway partition between the data will result in a group of data that is entirely 'no spike'. At that point it becomes difficult to draw any conclusions because what is the difference between a measurement of 'no spike' and the complete lack of measurement? There is no difference, unless there are some underlying assumptions. To avoid adding additional measurements, make the assumption that the information source is emitting a repeating message, and after splitting the data in half, each channel will loop back upon itself.

Figure 6.20: Assuming the information source will repeat the message, it must wrap back around on itself. With 8 measurements, there are 3 ways to wrap a 'rolling window'.

The unknown data can be looked at and an alphabet can be guessed. There seem to be only two fundamental units, grey and black. Then, we could split into chunks of 3 units each, per trace, and make those into symbols. If we split into groups of 4 each, we could have two total groups, but what happens at the 'edges' of the data? Since the data is only 4 timesteps long, we can't just assume there are extra 'no spike' symbols around it. The only possibility is to assume the data comes from an information source that would continue to repeat the original message again, by wrapping around back to the beginning. We could have started with data so large that we know there is a generous 'empty' section around our data, but then what does 'empty' mean in a binary symbol language? The symbols are either grey or black, there is not a third option for 'empty'. Having 3-bit available for each symbol, it could be combined into an 'end' symbol of 3

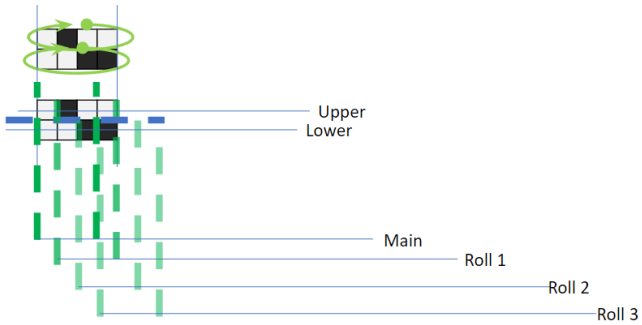


Figure 6.21: Choosing method 2 from Figure 6.20, the data is first divided into an upper trace and a lower trace. Then, 3 bit symbols are gathered, with the upper 3 bit symbol referred to as **main**. A choice to handle the edges of the data is to 'wrap' the symbols around from the beginning to the end, so moving the 3 bit window 1 bit to the right for each roll, results in 4 3-bit symbols from the upper trace, and 4 3-bit symbols from the lower trace.

grey and a 'start' symbol of 3 black. That leaves intermediate symbols of combinations of grey and black available for containing information. But, from an energy perspective, if each of the original alphabet symbols is assigned to equal energy codes, i.e. in the form of a population or a phase code, they cannot be distinguished on the basis of energy alone.

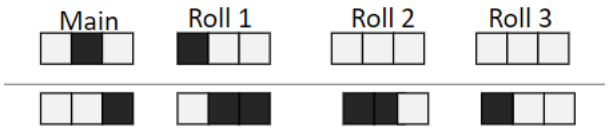


Figure 6.22: The smallest *useful* segment for unknown data is three bits. Each of the traces provides 4 symbols of 3-bits each, and since there are 2 assumed traces there are 8 symbols total that can be made from the unknown data.

If we need to have a start and end symbol from this group of 8 bits or spikes, then we need to have a minimum symbol block size. To solve the edges we can use a boundary condition where the data wraps around on itself both from right to left forming a 2-dimensional torus. [187] Using a torus wrapping gives periodic boundary conditions with

a period of 1. Alternate wrappings give different periodic boundary conditions, with a 'reflected' dimension resulting in a doubled period boundary condition and the 'let us just ignore the boundary' condition resulting in a $\frac{1}{2}$ period. Estimating the periodic boundary

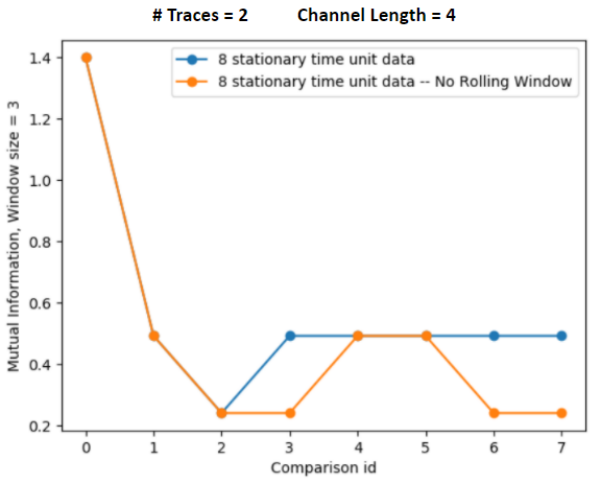
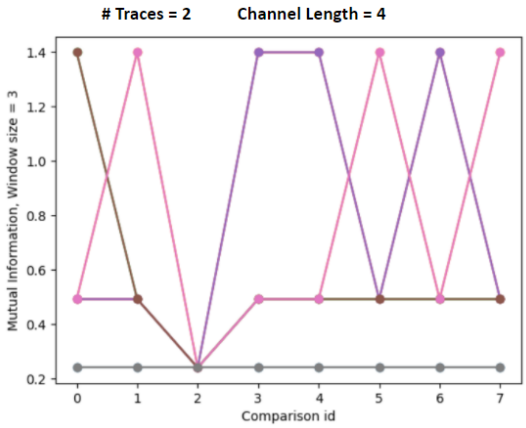


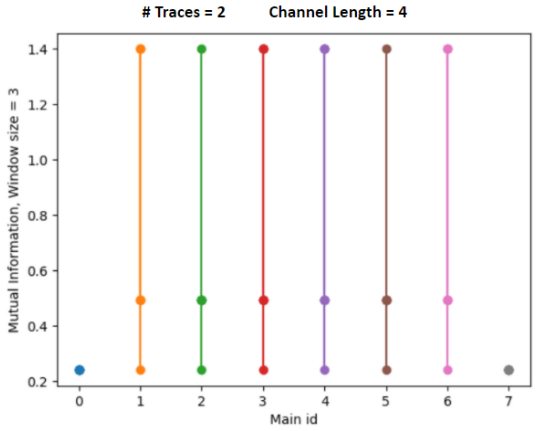
Figure 6.23: The difference between mutual information when a rolling window is used (8 stationary time unit data) vs appending extra 'no spike' symbols at the end of the data (8 stationary time unit data - no rolling window), creates an illusion of an extra periodic symbol.

conditions from mutual information between the main symbols helps distinguish the type of encoding that is possible, because a rate code must have finite time, as detailed in section 4.7. If the data for the selected alphabet size is not able to achieve mutual information maximization for the guessed alphabet size then more data is necessary before an encoding guess should be made. Using a 2-dimensional torus boundary wrapping, and estimating the mutual information between the **main** symbol and the other symbols results in a plot of JVHW mutual information heavily dependent upon the selection of **main**, as shown in Figure 6.24.

The implications of the unknown data experiment indicate that an energy based code, which is to be directly understood to correspond to the information content of a transmitted message in the form of a rate code, is only available to 1/2 of the codes for



(a) If any of the symbols (except **000** or **111**) is treated as **main**, it creates alternating periodic patterns.



(b) The JVHW mutual information either has consistent values for comparisons of 1 grey 2 black vs 1 grey 2 black, depending on match or non-match.

Figure 6.24: When **main** is **000** or **111** the JVHW mutual information is 0, because for any comparison there are 0 symbols in common or only 1 symbol present.

a prospective window size. This is apparent from Figure 6.24b, because only 6 of the 8 possible 3 bit codes are able to return a JVHW mutual information estimate above 0. However, when the alphabet size is allowed to expand, the number of rate codes available for energy to information equivalence expands, as shown in the following Figure 6.25.

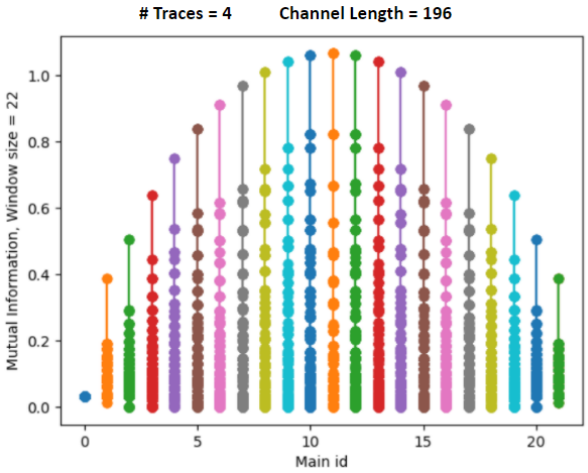


Figure 6.25: The mutual information of Mnist binary data when compared against main symbols of length 22 bits with a ratio of 0's to 1's varying from 22 0's to 0 0's.

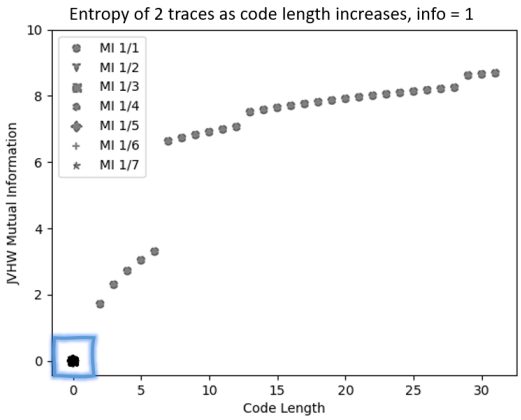
The plot shown in Figure 6.25 shows an entropy estimate higher than 1 for the situations where the partition of 1's to 0's is close to 50/50. That is because there is likely additional information to be found in an ordered symbol assignment right around the bit patterns where 50/50 distributions of 1's and 0's are to be found, and, if the symbol language was switched from binary to a larger integer alphabet, the JVHW entropy estimator would be better able to distinguish between meaningful 50/50 patterns and those that are caused by noise.

6.6 Low Energy Response of Spike Train Encoded Data

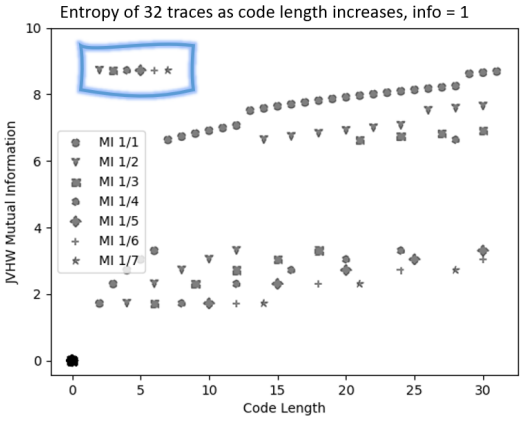
To establish a phase code for the two binary symbols 0 and 1. This can be accomplished by receiving a driving signal on one wire and emitting an outgoing signaling with three wires (channels) at minimum, which then signal a 0 by spiking in a cyclic order from $w_0 \longrightarrow w_1, w_1 \longrightarrow w_2, w_2 \longrightarrow w_0$ as long as the 'driving' symbol is 0. When the driving symbol is 1 the spikes emerge in the opposite order from the three wires $w_0 \longrightarrow w_2, w_2 \longrightarrow w_1, w_1 \longrightarrow w_0$. The relation between the single incoming signal wire and the outgoing 3 wires is that the 3 output wires encode the original signal into a non-binary digital code, which in this case, transmits 1/3 bit per event.

To estimate the spike train entropy for a non-binary digital encoding natural language is used to form the basis of a hierarchy of codes following the form of "phrase", "word", "letter", "binary". Two phrases with equivalent meaning but different letters are selected to start. Then the letter's for each phrase are converted to binary representation using a balanced code with equal numbers of 1 and 0 with different orderings to represent a letter. Each letter represents a unit cell in a 2D Race Logic unit cell shown in Figure 6.3b. The

representation is only there through assignment of an identity by the circuit designer to a trace in a spiking channel. Figure 6.26 shown below with either 2 or 32 traces demonstrate how the estimate 'fails' and defaults to a maximum estimate when the code length is less than the number of channels.



(a) When the alphabet size is 2 there is no difference between a phase coded information vs rate coded information, because a binary alphabet is B-system ergodic.



(b) When a state machine with 32 'traces' is permitted to run for many time steps, and the mutual information is calculated for ratios of the data, the MI follows the expected logarithmic trend, except at the beginning, where the data is both simply ergodic and B-system ergodic. The uncertainty causes the JVHW estimator to use its 'surprised' estimate and significantly overestimate.

Figure 6.26: The relevance of phase encoded data is not visible until the number of traces is increased to represent $|A_\chi| > 2$, as shown in (b)

The over estimate is only found when the driving symbol, labeled **info** is a single character 0 or 1 to indicate the direction traversed by phase code.

Low Energy to Entropy

A low energy spike train response can directly correspond to the entropy of the incoming spike train. The work assumes a spike is represented by a physical pulse. A neural code conveys several types of information simultaneously and the content of the

information is able to continuously evolve to best respond to the incoming stimuli. [98] There are select Markov chains which are both B-system ergodic and simply ergodic, which are capable of outputting neural codes. The notable contribution of the theorem indicating an information is physical relation to reset entropy developed in Figure 6.2 allow for advances in data reduction.

For a coding strategy based on minimum numbers of measurements, entropy is only useful as a measure of information if the energy of the signaling strategy is also used as a measure of information.

To perform a successful entropy estimation for a block of gathered spike event data, but also at the same time to be able to treat it like an energy equivalent, using a larger number of parallel channels (> 200) allows for much better possible entropy values. Increasing the number of parallel channels is akin to adding more elements to the set A_χ and increasing the value for $|A_\chi|$. The estimate of around 200 channels is for the JVHW estimator which in its algorithm uses a predefined table of estimates for its base guess if the input data looks 'surprising' .

The phase encoded data is only able to be present in address event representation form if the spiking channel has sufficient numbers of traces present with at least 1 event occurring on each trace. An acceptable energy use for energy as information equivalence to be used for decoding is only up to the number of traces in the channel which have already experienced an event in the recording time. Most of the time, the energy to information estimation is limited to performing logarithmic information estimates, which are only accurate within 1/2 of the maximum entropy estimate for the alphabet size. However, there are likely to be occasional times when the energy use is able to return an information estimate much closer to the actual information content delivered. This can be seen because The high entropy estimate caused by short code lengths in Figure 6.26 can be reproduced by shifting the codes.

6.7 Data in computation

Existing data is digital and stored as 0's and 1's. However, the binary coding means often it is necessary to convert a binary file into an address event representation (AER), and, the original underlying phase data that formerly existed before the analog to digital recording of the data may no longer be present.[92] Possible methods to translate existing binary data to spike event format include using a saccade pattern to scan a small window through a 2D image, interpreting the integer values of a color as a spike rate and converting a 32-bit floating point to a volley of spikes.[188]

That is why this dissertation proposes to intersect the capture at the point where the analog signals are first converted to 'digital' although that is taken to 'binary'. Building an encoder to receive analog signals and convert them into a noiseless symbol language is also still digital but it is no longer binary. The incoming data is immediately converted to spiking event form, and it is also stored as a spiking event because it is still traveling through the encoder as it waits for transmission off-chip to the decoder.

The decoder needs to be able to interface back into a typical binary machine. In that case, the decoder needs enough memory flip-flops to decode a message corresponding to the largest parallel output of the encoder. Generally in that type of a decoder it is assumed that the encoder will send the signals to the decoder, which will toggle the memory flip-flops and after $|A_\chi|$ spikes have passed, the decoder memory is valid for each successive spike.

The non-binary spiking channel consists of many wires, which form into a link between the analog-to-(non-binary) digital sensor. The link is able to amplify the signal for transmission across longer distances where transmission line effects are present, thus, the phase codes which are selected for particular relevance need to be partitioned across the sustainable rate codes. That means it would be unwise to choose to place most of the

phase codes at an energy use position where they are equally interchangeable and all are using rate 1/2. But, also, it would be a mistake to only focus on phase codes at very low rates, ie, rates with sparsity less than 30% or greater than 70%. That is because the most energetically distinguishable code spaces exist along the entropy curve at the half way point between minimum and maximum entropy for the given non-binary coded alphabet.

The reason to still choose some codes to transmit as phase instead of rate is that occasionally there are messages it is necessary to communicate faster than typical. Asynchronous event driven signaling is a low energy solution for classification problems. A classification machine is inspecting the inputs and sorting the outputs either into space or time. A synchronous approach might be an XOR logic function, which classifies the inputs as 'same' or 'different'. With a synchronous solution the XOR Boolean feature requires a slower 'slow' clock to move the input data past the input ports.[189] The clock only needs to be slow in comparison to the arrival speed of the input data. The data used to compare for 'same' or 'different' is restricted to a 'slow' clock. An asynchronous solution is able to have timing precision at the arrival speed of the input data and uses output energy when the classification is one of the two neighboring rate coded labels.

6.8 Conclusions about Alternative Computation

Classical computation requires irreversible, that is to say, permanent, expenditure of energy, to deliver an answer to a calculation into the memory. When these irreversible actions complete, the logical operation is equivalent to a non-binary digital XOR of the past history of the Boolean values. It means that the order of the operations leading up to that erasure of memory did matter at a point in the past, but, now, in the future the energy is already spent to erase the memory and the result of the computation in

the future no longer need to depend on the past order of the operations. The original formulation of race logic for delay based computation relied upon this classical expenditure of energy to deliver an answer in the form of a delay based rising edge, which required either asynchronous 'handshakes' or a fast synchronous timed clock to decode the answer. Asynchronous Impolite signaling with pulse logic maintains a constant open channel that is not dependent upon a clock or 'handshakes', which makes it suitable for feedback driven race logic circuits that settle into a rate code for a constant input.

Should the noise in the communication channel manage to grow beyond twice the possible energy use, as determined by the population code, through an extremely unlikely scenario of events occurring on the input sensors, the output of the circuit will be unable to overcome the environmental errors in the channel despite the redundancy the of the population, rate and phase channel code.

The shifting energy use of a circuit implemented with Impolite pulse gates is bounded on the upper side of operation to a maximum of twice the possible energy use, before the delivered answer is no longer comprehensible because it has turned into an indistinguishable oscillating blob of meaningless noise. A way to control the possible energy use is to lower the overall power to the sensor, and reducing the power for a cyclic CMOS circuit (with feedback) will result is a less noisy signal. The lower bound on the energy use is explored in the next Chapter, where the lower bounds of energy use are considered for a neuromorphic communication channel.

Chapter 7

Lower Bounds on Neuromorphic Communication by Representation as Stationary Computation

This chapter is shared with a paper titled "*Lower bound on neuromorphic communication by representation as stationary computation*". (To be sent for review at Journal of Neuromorphic Computing and Engineering)

We define the computational lower bound that a channel utilizing neuromorphic communications must meet. This work treats a neuromorphic system as a communication channel which imparts additional information onto the sensed data in the form of stationary computation. Communication theory divides between: (a) the problem of knowledge understanding across intelligence on either end of the communication channel, (b) the physics of how to sustain the message through the physical media propagating the signals. It separates the signaling from the meaning abstracted to form knowledge. Neural dynamics and neural coding strategies are communications with a basis from evolutionary

pressure, where the lifeforms best able to communicate efficiently out compete their counterparts. The results of these experiments present biologically plausible spiking neural networks for instantaneously (within $|A_\chi|\tau_m$) sensing the state of the environment.

To prevent violating the second law of thermodynamics the energy costs of the system must consider the erasure energy $E_{erasure}$ to replace the unknown value originally stored in the memory of the 1-bit computer.[167] For classical computation the energy use needs to include the communication cost to transmit the sensed state of the environment. This value is dependent upon the physical implementation details of the 1-bit computer and represents the minimum process energy quanta E_{PQ} to communicate a sensed measurement and emit an event in the form of a voltage pulse. The state of the environment can be stored in the memory of the 1-bit computer using different methods.

1. A binary level can represent 2 instantaneous measurements
2. A pulse can represent 3 instantaneous measurements
3. An edge transition can represent 4 instantaneous measurements
4. A switching rate can represent *6 or 8+ measurements*
5. An average of binary levels can represent *6 or 8+ measurements*

Focusing on method 1, 2 or 3 at first, the 1-bit computer has only two states, S_0 and S_1, where S_1 only occurs when the computer senses the environment to be in a state where placing a piston will extract useful work. The initial state of the computer is unknown, $S_u == S_0||S_1$ and the initial state of the environment is noisy, because analog signals are noise. The computer requires a finite length of time to observe the environment to make its measurement, and when the measurement time is past the state transition will instantaneously move to S_k, the known state of the computer, which is $S_k == S_0||S_1$.

The first measurement is $S_u \rightarrow S_k$. If there is only 1 likely state of the environment (S_0) then measuring S_1 is likely to be an error that occurs with probability μ.

Table 7.1: The first measurement represents a state transition from unknown to known.

$S_u \rightarrow S_k$	Probability of state transition	$E_{erasure}$
$S_0 \rightarrow S_0$	$(1-\mu^4)/(1-\mu^4+2\mu+\mu^2)$	0
$S_0 \rightarrow S_1$	$\mu/(1-\mu^4+2\mu+\mu^2)$	$1 * E_{PQ} * m$
$S_1 \rightarrow S_0$	$\mu/(1-\mu^4+2\mu+\mu^2)$	$1 * E_{PQ} * m$
$S_1 \rightarrow S_1$	$\mu^2/(1-\mu^4+2\mu+\mu^2)$	0

The initial state transition from unknown to known can take place using level, pulse or edge *instantaneous measurement* which imposes an additional communication cost m, which will change the energy use and error probability. The instantaneous measurement requirement means that the copying of information needs to proceed through a reversible logic gate, such as an inverter, to communicate the result of the computation from sensing the environment. A 1st order approximation of energy use in CMOS circuits states that only switching from $0V \rightarrow 1V$ or $1V \rightarrow 0V$ uses energy use. Thus for a binary level $m = \epsilon$, where ϵ is a small fraction of energy, only observed with 2nd order energy effects. This small value of m, indicates instantly detecting 2 measurements necessary to capture a binary level will be likely to miss an error. indicating it is unsuitable for an encoding with energy-to-information correspondence.

A pulse includes 2 edge transitions, with the second transition guaranteed to happen after the 1st transition occurs. Since the second transition is guaranteed, $m = 1 + \epsilon$ because E_{PQ} is measured by recording the minimum pulse the process is capable of sustaining. If an edge transition is used $m = 0.5 + \epsilon$, because only 1 transition is sufficient to capture the measurement. However, when $m < 1$, the separation between lifetime energy states is harder to distinguish, because they are closer together.

If μ increases, the likelihood of immediately spending energy on the cost of reset

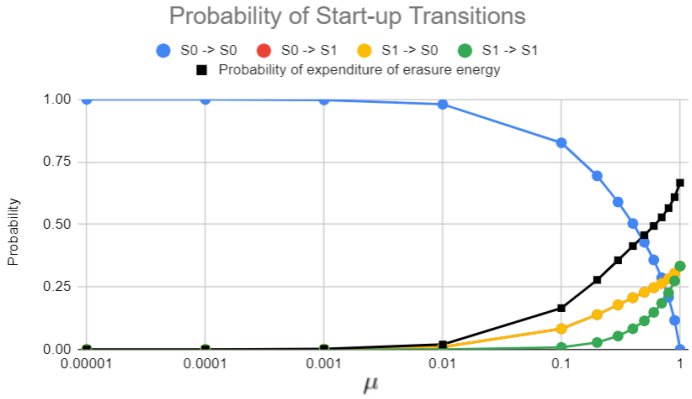


Figure 7.1: The probability of expending energy at reset increases as, μ, the probability of measurement error, increases.

increases, shown in Figure 7.1 where the probability of immediately expending reset energy rises to 66% when the error $\mu = 1$. The value is less than 100% because the state of the 1-bit computer memory is unknown and if it was in S_1 then the error in measurement does not change the 1-bit memory and there is no energy cost for the measurement.

Now that the 1-bit computer has entered a known state the next measurements are limited to transitions from the recently measured state and the transition $S_u \to S_k$ can not repeat again.

Once the 1-bit computer has reached the known state S_k the energy costs that will be incurred in running the machine only depend on the environment. The reason the energy costs of operation only depend on the environment is that the sensed environment controls how often the memory bit will need to transition. The state of the 1-bit computer can be represented as a Markov chain, and the reset energy cost can be calculated by waiting for a mixing time. Using method 4 or 5, the lifetime energy cost of the 1-bit computer will become a rate or average representing 6 or more measurements of the environment, with a Markov chain represented by the energy cost. The rate code can be represented

as a Markov chain with lifetime energy use for the state description.[113] When the state label is dependent upon lifetime energy use the lower bound on computation energy use generalizes to Wolpert's lower bound, $k_B T\, ln(2)$ times the Kolmogorov complexity(σ) and log(Bernoulli measure of the set of strings that compute σ) and log(halting probability for U).[5] This work is focused on the identification of a lower computational bound for specific implementations of a physical 1-bit computer. It is not realistic to try to achieve an operation close to $k_b T\ ln(2)$ with classical computation. Instead the goal is to identify how a physical connectivity structure could be arranged such that the energetically wasteful operations of reset and synchronous clocking can be avoided while continuing to output information suitable for neural codes.

7.1 Experiments

This is a group of experiments to look at the encoding capabilities of a 1-bit computer which emits an event when the state of the environment moves above a threshold. The environment variable e from equation 2.30 is modeled by Gaussian white noise with a mean and standard deviation selected to correspond to the sensitivity of the process parameters from open source process development kit, Skywater 130nm [190]. While the LIF circuit model proposed in Figure 6.4 is an acceptable equivalent to the software defined LIF neuron model, both of the two implementations are not capable of the full range of action potential expression demonstrated in biological neurons.[64] However it is acceptable to use multiple instances of the models to create a more accurate biological response.[191]

These experiments begin by considering a single LIF circuit (Figure 7.2 circuit 1) as acceptable for sensing and asking what is the lower connectivity bound to produce population, rate and phase codes. Next, the number of inputs to the 'dendrite' tree, analogous

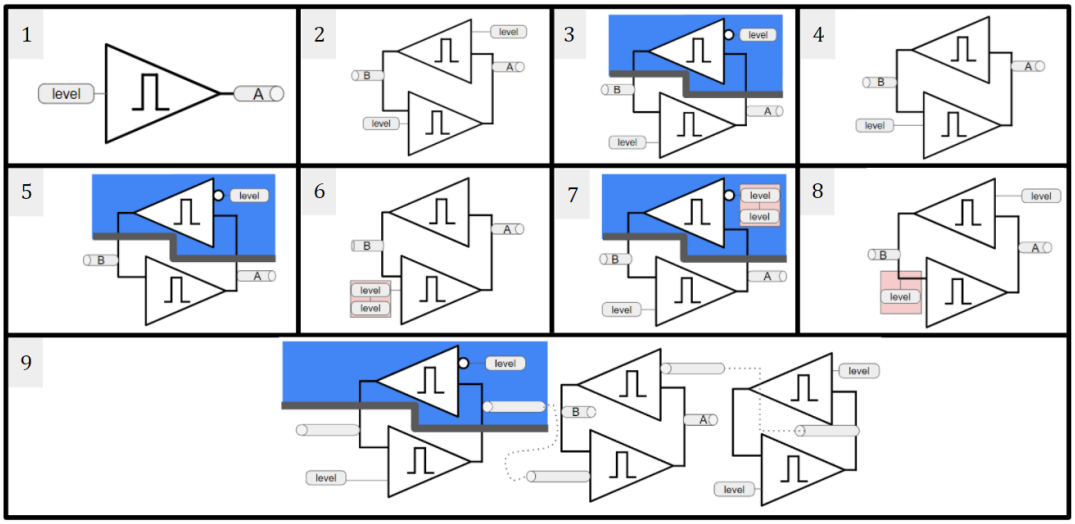


Figure 7.2: Level represents an analog input. Cylinder labeled 'A' or 'B' represents a tap. Tap is a measurement point resulting in a digital signal. The classification of each circuit, labeled 1-9 is to be determined in the following experiments.

to the pull-down and pull-up network of the critical node of the single LIF in Figure 6.4 is explored, to examine additional computational capabilities that may be present in the dendrite tree alone.[192] Finally the number of LIFs used in a circuit (Figure 7.2 circuit 2-9) is expanded to consider what connectivity demonstrates acceptable encoder signals to fully capture the range of codewords permissible under population, rate and phase coding as detailed in subsection 4.7.

7.2 Results and Analysis

7.2.1 Single LIF neuron with 1 input Exposed to Noise

The 1-bit computer is analogous to a population of 1 leaky integrate and fire neuron, in turn analogous to a Nengo model utilizing the neural engineering framework.[63], [193]

Both of these models are exposed to white noise on their single input and the time to first event is measured for a range of noisy conditions. The output of the LIF circuit

is shown in Figure 7.3, where a single neuron with 1 sensing input is able to generate a population code once the standard deviation of the Gaussian noise is 0.1V from a mean of 0.2V. The output is not a rate code because the single neuron has zero feedback meaning $t \to \infty$, thus the LIF is unable to produce a rate code. The output of the Nengo model is shown in Figure 4.2. The software Nengo model is able to simulate for a much wider range of τ_m and supports a spiking neuron model with programmable membrane RC constant separate from the refractory time, which is the time duration the neuron output is held at reset after an event. The Nengo software model requires a maximum and minimum rate to be pre-determined in the parameters to initialize the LIF model. That means that the Nengo software model could output both a population and a rate code for a single LIF neuron, however, those limits are artificially imposed and are intended to represent the physical limits of the LIF. If the Nengo simulation is driven to those extremes it will fail with an error message.

The experiment looked at spike occurrences when a single input, single output LIF neuron (Figure 7.2 circuit 1) is exposed to a noise. Each plot in Figure 7.3 is gathered by exposing the LIF model to a randomly generated Gaussian noisy signal with increasing standard deviation ($+0.1V$) for each consecutive plot. The fractional macrostate variable ρ from Equation 2.29 is inherent in the design of the circuit LIF, by setting a threshold to measure an event crossing, and is constant for all of the plots. In fact, if ρ is not constant between runs, the behavior for the circuit is only altered as much as the performance of the Nengo model is altered when the start-up random number seed of the SNN is changed to an alternative number. The overall behavior at a particular noise level will remain consistent.

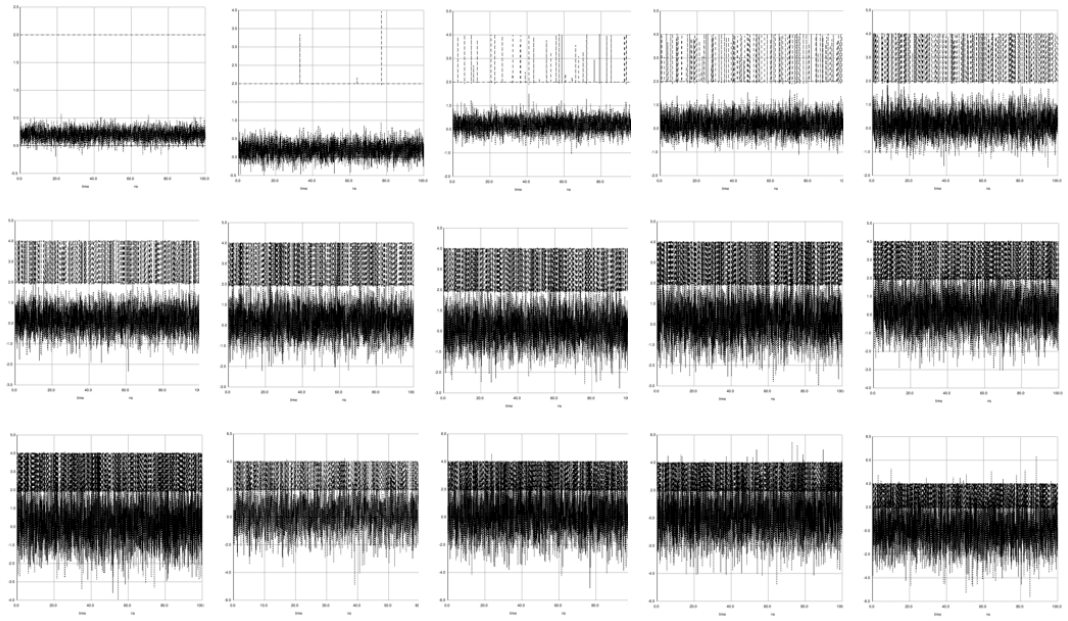


Figure 7.3: Experiment 7.2.1 with a noise source at 0.2V, and standard deviation of 0.1V is unable to generate an output from the LIF within the 10ns simulation period. The LIF fires more often as the standard deviation is increased in 0.1V intervals toward the plots at the bottom right, where the output resembles an oscillator and is spiking at a rate limited by τ_m.

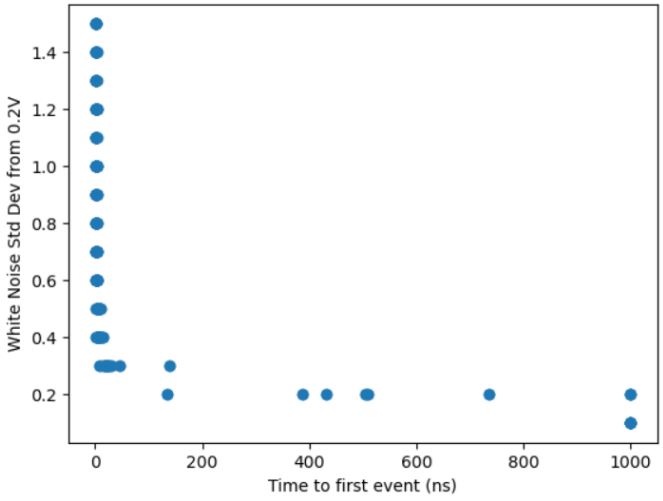


Figure 7.4: The start-up time until the first event decreases when the environment variable e has greater standard deviation.

7.2.2 Start-up time of single LIF neuron with 1 input

When the 1-bit threshold computer (Figure 7.2 circuit 1) is started with an unknown start-up state, the period of time until the 1st event occurs represents the state of the environment variable e from equation 2.30. It represents the noisy environment that the 1-bit threshold computer is placed in. The threshold value for the LIF model is 1.2V which is a value derived from the design of the circuit and is related to ρ from equation 2.29. When the white noise is Gaussian with a mean of 0.2V, varying the standard deviation changes the length of time the 1-bit computer must be exposed to the environment before the state is known. The experiment uses a single environment sensor (A LIF circuit with 1 output) and uses uncorrelated Gaussian noise sources when there are more than 1 input. That is equivalent to the condition of a single particle in the system shown in Figure 6.8. With only a single sensor all of the dynamics of the noisy environment are captured by the timing of the spike occurrence of the first measured spike of the 1-bit threshold computer.

The start-up time experiment is run for a time period until an event occurs even as the standard deviation voltage moves to smaller values. Figure 7.4 shows that an event will occur eventually, with times up to 1us eventually resulting in events. Once enough time has passed to witness two events it is reasonable to estimate e the input environment, but, the time for two events to pass has an order-of-magnitude range from 10^{-9}s to 10^{-7}s for standard deviations from 1.6V to 0.2V. In fact, waiting a duration of time to observe an event at standard deviation 0.2V would produce a code using many timeslots t_s, implying 100's of possible rate codes given equation 4.6. However, this has implications for any coding strategy selected because a single neuron, with a phase code, operating for a very long length of time has an exceptionally large number of codes available to it, with an upper limit set by equation 4.7. Existing research on phase codes indicates

they are primarily used for short and fast communications via phase-of-fire[95] codes or winner-shares-all codes [119]. Research conducted into the phase sensitive nature of the LIF circuit model used for this experiment indicates spike timing at sub-minimum pulse-width ($< \tau_m$) is acceptable if the code utilizes multiple wires with single transitions occurring on different wires.[97]

7.2.3 Minimum τ_m event time measurement

Any LIF model has the value of τ_m as a defining parameter of its function. Since τ_m is used as the basis for codewords used for models of neural communication in Section 4.7, the physical time it takes for τ_m to occur in these experimental circuits is measured. The

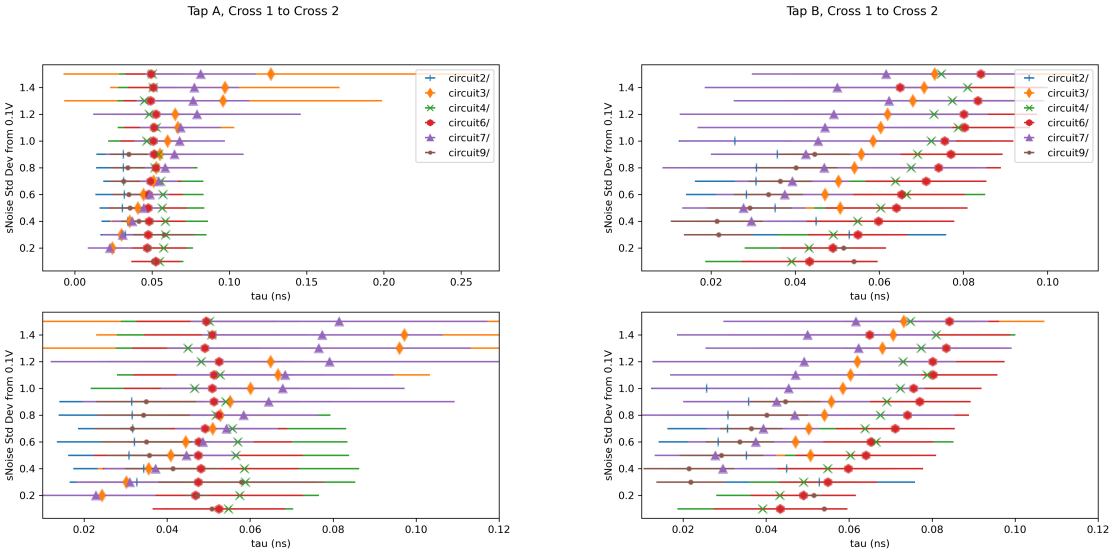


Figure 7.5: τ_m measures the width of a spike event at (Left) Tap A (Right) Tap B.

values of τ_m have a minimum width without concern for the noise level of the environment, but when the standard deviation of the noise in the environment is at or close to the threshold of the LIF, the width of τ_m increases to more than double the width of τ_m at lower noise levels.

7.2.4 Multiple inputs to single neuron

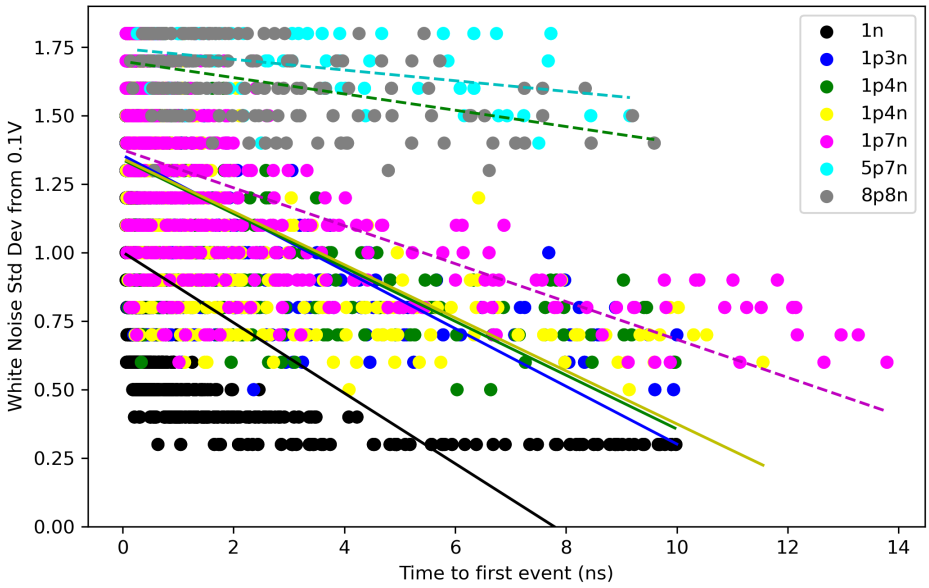


Figure 7.6: White Noise on the input acts as a model of the environment. When the number of inputs is increased the sensitivity of the circuit changes.

When the number of inputs to a single neuron is increased beyond one, the additional inputs can be arranged in series, parallel or a combination of series and parallel. The additional inputs can be added as positive (p) or negative (n) responding inputs. The allocation to p or n is indicated in Figure 7.6 by the labels stating the number of p and the number of n inputs included in the multiple input gate. The time until first event changes with additional inputs to the single sensor, when the additional inputs are added in series the sensitivity to environment noise is reduced as shown in Figure 7.6. An experiment with parallel inputs is not included because it would result in increased sensitivity and phase coding does not benefit from increased sensitivity, there are more available phase codes to sense when the slope of the start-up time is gentle, because there is more space for interspike overlaps.

The additional inputs are added to the circuit model in series to the original input, meaning the circuit is only able to fire in the rare instances when the input noise exceeds multiple standard deviations for all noise sources used as input to series sensors.

With each additional input the curve from very fast event startup time to very slow event startup time becomes smoother and more distributed as shown in Figure 7.6. The additional series inputs serve to decrease the sensitivity of the LIF to the environment e, particularly when the ratio of p to n remains close to 1/2, shown by 8p8n which has the mildest slope.

7.2.5 Two Neurons for 1 τ_m or One Neuron for 2 τ_m

The number of codewords a single neuron can produce is determined by the population code described in Equation 4.5, and there are behaviors that a population code is sufficient to decode with.[115] A population code is able to function when $t \to \infty$, while adding rate codes for a single neuron would not be possible unless it has at least 2 inputs, with 1 of those inputs the result of a feedback loop from the output of the single neuron. The reason 1 neuron requires feedback to produce a rate code is because the feedback loop guarantees $t \neq \infty$. A population code is a lower bound for a spiking neural network because $t \to \infty$ is necessary for a stationary computation $\pi = \pi P$ as explained in Section 2.6, for a stationary computation to develop into a rate code the number of population codes (that is neurons) need to expand.

A system with two neurons measurable for 1 τ_m is shown in Figure 7.2 circuits 2-8, and circuit 1 is only acceptable for a population code. Circuit 1 is a single neuron and when it is measured for 2 τ_m, the number of possible codes is 6: 1 population, 2 rate, 3 phase for a total of 6 available codes. Every **2** time steps we are able to capture up to $log_2(3!)$ codes, but, importantly, since there is no feedback in the population $t \to \infty$ and

the lower bound on the neuromorphic communication due to the stationary computation is 1, from the limit of a population code equation 4.5.

For circuits 2-8, there is feedback present for each of the neurons and because of this $t \neq \infty$. With 2 neurons and 1 τ_m there are 8 possible codes: 3 population, 2 rate, 3 phase. Every **1** τ_m we are able to capture $log_2(3!)$ However, the rate codes measure identical to the phase codes. This suggests that the rate code and the phase code are indistinguishable in the circumstance of measurement for 1 τ_m. The rate of arrival is permitted to be twice as fast with a population of 2 neurons instead of 1 neuron. Though phase codes increase the density of possible temporal codes, when $t_s \leq 2$ a larger population is preferable.

The practical implications of building a sufficiently large SNN and operating it for long lengths of time ($t_s \geq 10^3 \tau_m$), limit the long range effectiveness of phase codes, by introducing an uncertainty to τ_m. However, when phase codes are able to operate at sub-minimum pulse-width ($< \tau_m$) as demonstrated by [97], the lower bound is set by the mixing time of the rate codes for the t_s of the SNN population and not by the pulse-width.

7.2.6 Mixing Times of Neurons

When an SNN has $t_s \neq \infty$ the communication is able to trade a particular rate code for the larger space of the possible codewords that phase enables. To demonstrate the expansion of the rate codes to phase codes consider the mixing time from Section 2.6, where the mixing time is the number of state transitions the 1-bit computer is from the stationary state. Where τ_m describes the length of time that a measurement needs to represent a spike event, it is possible to receive N events in a time less than τ_m, one event from each different neuron. Having explored the relationship between environment and delay until start-up of a single neuron with a single input and for multiple inputs in series, it is time to move on to neuron circuits which are limited to finite rates and

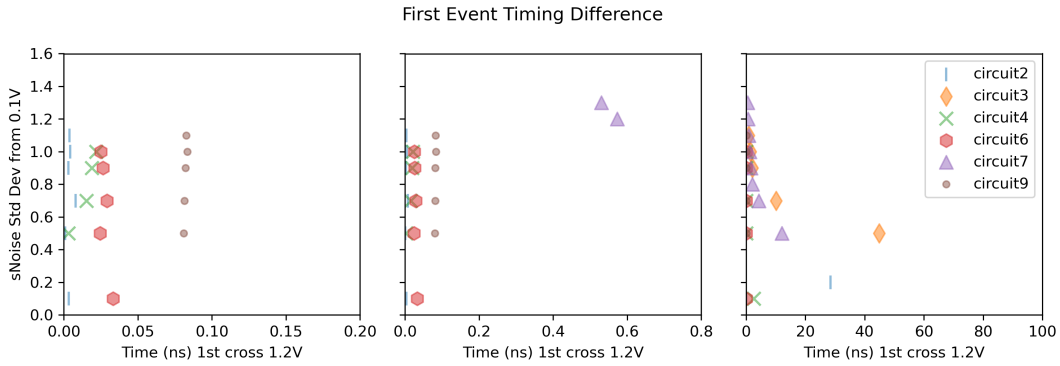


Figure 7.7: First event timing difference compares the time for Tap A and Tap B to fire in each circuit. Figure 7.2 circuits 2-8 demonstrate 2 events firing at sub-minimum pulse-width ($< \tau_m$) for circuits 4 and 6.

correspondingly finite phase. Figure 7.2 circuits 2-8 require an event to happen from each of the 2 neurons in the feedback network before the mixing time is complete.

Circuit 2 has a temporal space between Tap A and Tap B much less than the size of τ_m. Circuit 3 has a temporal space between Tap A and Tap B much greater than the size of τ_m. Circuits 4 and Circuit 6 have a temporal space less than τ_m. Circuit 7 has a temporal space multiples of τ_m, but still much less than Circuit 3. Circuit 9 has a temporal space nearly equal to τ_m.

The lower bound is derived from a code space argument: The population of neurons (N) at the sensor interface of the spiking channel must be large enough that the mixing times permit replacing no more than $N - 2$ rate codes with selected phase codes, for a 1-bit computer.

7.2.7 Energy Use

To maximize the energy use of the circuit for encoding into phase, it is necessary to have more than a 2 neuron system. A 1-bit computer is not large enough to produce useful phase codes. To expand beyond a 1-bit computer it's necessary to increase the number

of neurons. Phase sensitive codes are best implemented by creating rather long looping feedback chains, similar to Figure 7.2, Circuit 9, but with longer chains of neurons. The phase code could be selected by having a large (sufficient to fit an information source around size $|A_{simpleThis}| = 24$) input dendrite tree, as studied in Experiment 7.2.4 at the population code which starts the looping rate codes. Another option is to have multiple population coded neurons in the loop with varying degree of probability of spiking, once again through moderation of the input dendrite tree. Then, a phase code can be constructed based on the positions of taps, placed to interconnect between the rate code loops, which will start at varying degrees of probability, corresponding to $|A_{simpleThis}|$ for its simply ergodic expression, while at the same time, the B-system ergodic probability for $|A_{simplethis}|$ generates as a phase code.

Once $|A_\chi| > 2$ the system is greater than a 1-bit computer. Both of these circuits will eventually reach their mixing time and begin to output the stationary broadcast of the $|A_{simpleThis}|$ information source. At that point, it would be wise to include a reset signal from an inhibit type of neuron circuit, shown in Figure 7.2 Circuit 5, which is the differential signalling opposite of Circuit 4. However, allowing the stationary broadcast signal to come to a mixed state is not necessarily a negative occurrence, because it means from that point forward, the changes observed in the output codes are mostly caused by the environment that is to be sensed.

7.3 Lower Bounds Conclusion

A neuromorphic communication from N neurons, with a maximum feedback chain of 1, has an efficient energy computation limit at mixing time $2\tau_m$. These energy efficient codes are accessible as phase codes beginning at rate 1/2 and are for calculating larger environment energy state changes. Increasing the number of series inputs in the dendrite

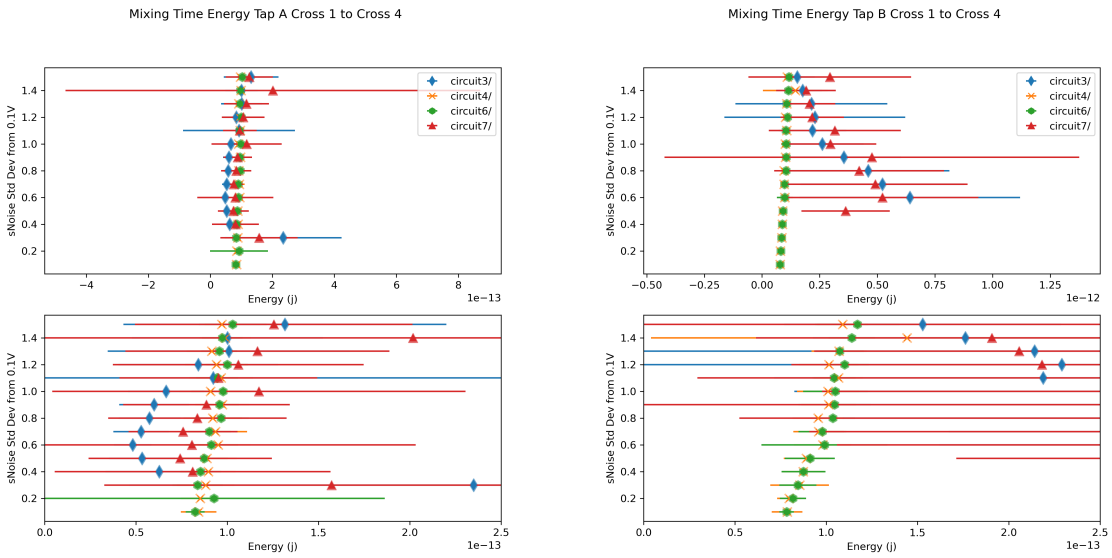


Figure 7.8: Energy necessary to complete mixing time when measured at (Left) Tap A (Right) Tap B.

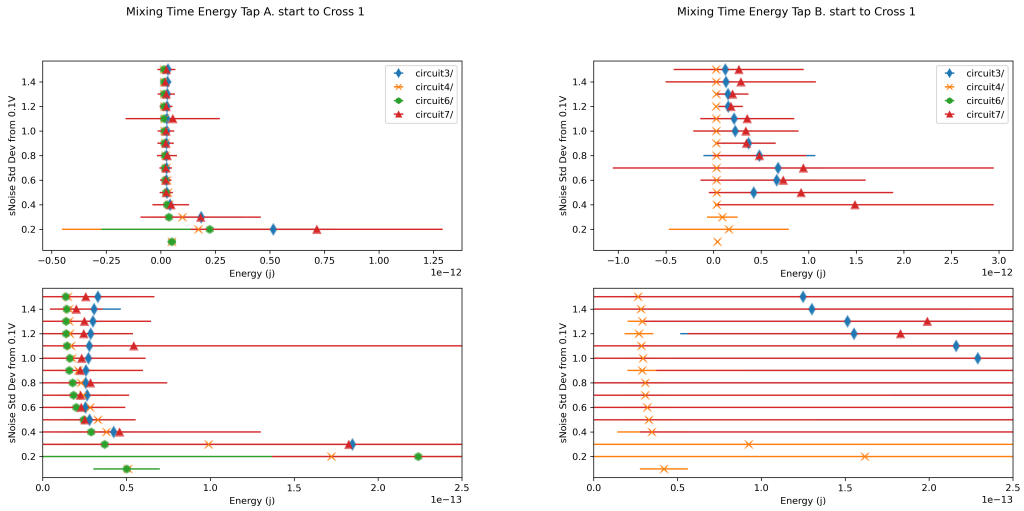


Figure 7.9: Energy necessary to generate first rise at Cross 1, for (Left) Tap A (Right) Tap B.

tree of the neuron creates a larger value of τ_m when the value includes both the RC membrane constant and the time after event that the neuron remains in reset. A direction for future research will explore the relation between phase code space utilization and the

complexity of the dendrite tree.

The circuits described in Figure 7.2 can be classified based on the prior experiments to result in the following code capabilities.

1 Population code with one level input and one tap.

2 Population Rate Code with two parallel inputs and two taps.

3 Population Rate Code with inverse two parallel inputs and two taps.

4 Population rate code with one input and two taps.

5 Population rate code with one inverted input and two taps.

6 Population rate phase code with series input and two taps.

7 Population rate phase code with inverse parallel input plus series code on inverse and two taps.

8 Population rate phase code with two parallel inputs that one include level lock, and two taps.

9 Phase code with two phase inputs from population rate codes and two taps.

Population codes and rate codes require more time to transmit the same information, while phase codes must come from a system capable of using a larger instantaneous amount of energy. The population size forms a base for future code words and a phase code must have sufficient population to support its possible energy use. Populations of only one or two neurons are insufficient for neuromorphic communication, while six (Figure 7.2 Circuit 9) is a particularly useful number of neurons to interconnect because it is the lower bound on the number of neurons necessary to enable communication from

a life form able to utilize population, rate and phase codes where it is energetically favorable to utilize a higher energy cost per code in return for a phase encoded output.

The utilization of phase codes demands a higher energy cost, but, the demands for more energy in exchange for faster performance have a biological imperative. Survival of biological systems depends on many innate different functions, however, critical decision making is an ability completely necessary for survival in emergency situations.[8] Humans have an intrinsic time keeping ability, that develops and changes over the life of the specimen.[29] The sense of time passing is necessary for successful critical interactions with the environment.[29] Time is perceived in multiple ways by humans and, both the outcome and decision making responses change with age or experience.[29] Primates and humans exhibit a tendency to learn from prior events and to adjust their predictive responses accordingly.[194] The survival-level cognitive skills develop from the dynamics of the interconnected neurons.[8] One theory for cognition is based on free-energy minimization, with the causality based result that large feed-forward and feed-backward loops are critical for maintaining the optimal top-down and bottom-up brain states for action response in the environment.[195]

Chapter 8

Unsupervised Channel for Communication and Computation

Unsupervised communication and computation is carried out by known finite Markov chains to mimic the biological ability to instantaneously (within $|A_\chi|\tau_m$) respond to the environment when sensing certain environmental conditions.

It can be modeled as a software implementation of a neural network; or as an ASIC hardware circuit. Furthermore, the two implementations can work together to develop a system with hardware capable of producing fast, low energy neural code response designed for a software neural network to parse while using fewer traditional computation cycles. There is no guarantee of using fewer traditional compute cycles, but since the output code is known to be simply ergodic with a finite alphabet of size $|A_\chi|$ the search space for possible solutions is limited, compared to a search space for codes of unknown ergodicity, which could be non-ergodic as well.

8.1 Hierarchical Markov Chain Probabilities

An information source is only a mathematical model, it does not need to exist as a physical implementation, it could stay as only a software version. To a large extent the output of a digital computer does not depend on the software it is executing. A hardware implementation of the information source makes it possible to bound energy with symbol use. To create a low power information source that is able to equate energy to information, a probabilistic table is defined, where the least frequently used symbol corresponds to the minimum energy quanta E_{PQ} needed to send a population code (Figure 7.2, Circuit 1).

One way to deal with noise in a digital transmission, is to duplicate the same transmission, either by sending the signal twice at the same time, with two parallel transmitters, or, by sending the signal repeatedly from a single transmitter. To improve noise resiliency, it is desirable to repeat the information source multiple times in a tiled spatial layout. However, if the number of symbols in the information is greater than the number of emerging terminals T which could emerge from the area of the integrated circuit, it is infeasible to actually use that information source. Additionally, if all of the symbols in the information source are equally probable, then the defined probabilistic table for the information source is only able to exhibit B-system ergodicity, in terms of energy use measurement, and the noise resiliency for the information source will be reduced, because each of the symbol emitters must use exactly the same as the other symbols.

The only way to make an information source where the output can equate energy to information is to design it according to the statistics of the probabilistic table representing the dataset it was created from.

Once there is a probabilistic table, made from only the monogram symbols in the dataset, there will be a critical message, that dominates within the phase code when

the input is noise. Additionally, when the input is not noise, meaning the input is not split into equal symbol partitions, an alternate message is present and the energy use will change to include the ratio of the difference between the bits needed to send the critical message and the alternate message. Race logic, at least 1D race logic, with a tiled computation grid and stationary inputs Q and R, works on scales when we are able to control the noise within the intergrid timing of the circuit. Typically it is implemented with edge detection, but in this work spiking pulse events are used to implement the race, because the spike event uses quantized energy and has an inherent reset signal.

Biological systems use phase code to orchestrate movements at seemingly unlikely speeds when the biological system is experiencing unusual circumstances. When the probability table is formed into a hierarchical Markov chain for phase, rate and populations codes, just like the 1-bit stationary computation in Chapter 7, but with a larger set of base symbols like $|A_{simpleThis}|$, the result is a hierarchical uneven fractional connectivity model capable of producing a range of spike events when confronted with noise. The sensor is using an energy bounded communication method to construct a phase coded spike event, it is only able to create those spikes by using an energy corresponding to the number of monogram symbols that form the population code necessary to sustain a rate code of corresponding size. The implication of this is that in the unlikely event of all population coded level sensors activating at once, the entire 2D race logic grid needs to simultaneously emit events, even until the point where all of the population code circuits are pushed into an unlikely fixed high voltage state.

The phase coded symbol can transmit because the average space-time state across the near-by circumstances for that system has a mixing state already present where they would be able to move into that fast processing time in less than τ_m, through an unpredicted change in the noise on their input sensors. A temporal phase coded signal is constructed from the underlying population codes of monogram symbols for a specific

dataset.

8.2 Example of software 1D Data Encoding

An example of software 1D data encoding is performed by directly creating a known finite Markov chain (the information source) from a dataset. The software known finite Markov chain is created by connecting the monograms through the n-grams for the dataset. This example is trivial and used a toy dataset: `_0_1_` that is 1 entry. This example is useful for the purposes of explaining how to build a n-gram Markov Chain to create a channel. From studying the dataset a Markov chain of possible grams, monograms through $(n-1)$-gram can be created. The result is a Markov chain, shown in Figure 8.1.

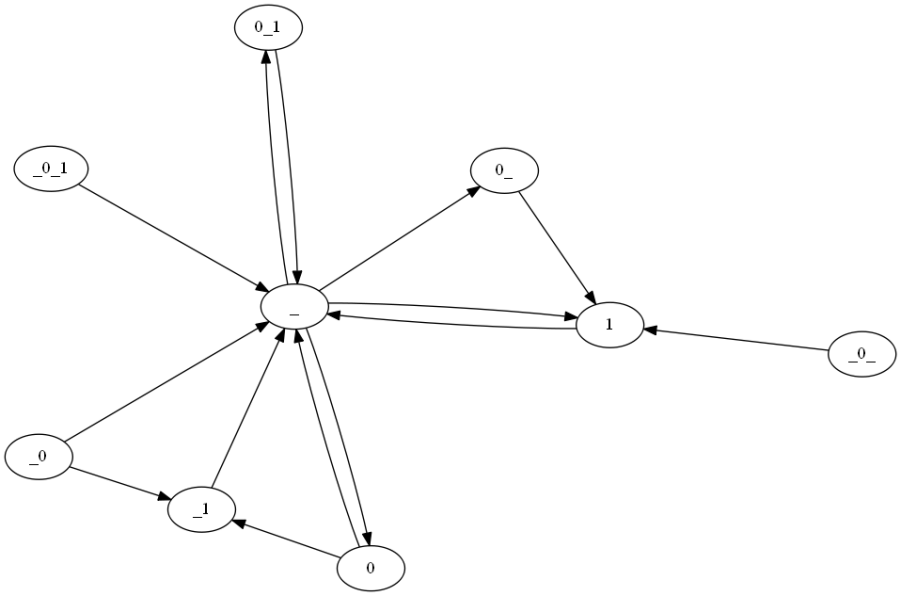


Figure 8.1: Non ergodic Markov chain used to create a 1D data encoding of the original dataset.

The dataset results in an information source containing 3 symbols, meaning $|A_{ternary}| = 3$. The 3 symbols are combined into a total of 10 different mono through n-gram pat-

terns. The dataset shown in Figure 8.1 has 3 mono-grams, 3 bi-grams, 2 tri-grams and 1 $(n-1)$gram. Each of the circled grams represents a message that could be transmitted as a spike on a trace. For instance, a population code, consisting of 9 circuits, with each circuit having a dendrite structure specific to each of the 9 circles in Figure 8.1, could be constructed. Then, the output of each of those nine population code circuits would be assigned a trace ID $0-8$, except, there is nothing specific that states the trace ID has to be an integer, thus, to help make the example easy to follow, the trace ID's are instead given the ID of the gram they represent. The list of grams here is: $(n-1)$gram: _0_1, tri-grams: 0_1, _0_, bi-grams: _0,_1,0_, mono-grams: 0, _, 1. The list of grams, is also the equivalent of the list of trace ID's, which if a shorter notation is desired, can be labeled with integers $0-8$. This section, 'Example of software 1D Data Encoding' is the only time a Markov chain containing all of the grams from mono-gram through $(n-1)$gram is completely written out, because, they tend to grow extremely large rather quickly. [1] Using the completely written out Non-ergodic Markov Chain to reproduce the original dataset is almost trivial. It can be done by randomly navigating around the dataset from any starting state. To do so, print out the Markov Chain from Figure 8.1 and grab a 6 sided dice. Drop the die on the paper and pick the closest gram it lands upon. Write down that gram as the initial state, at time $t = 0$. The gram and time is an address event representation of the 1D data encoding, it is the initial line in Table 8.2.

Initial State:	`0_1`
First Guess:	`0_1 | _`
Second Guess:	`0_1 | _ | 0_1`
Third Guess:	`0_1 | _ | 0_1 | _`

Table 8.1: Randomly traversing the maximum size Markov chain for the dataset _0_1_ results in a recreation of the dataset.

[1] I have a plot of the time it takes the laptop computer to calculate these big grams for larger datasets (simpleThis). It is not included because it did not seem directly relevant. I also have visuals for the larger gram Markov Chains from *simpleThis*. They get hard to read.

Trace ID	Timeseries (τ_m)
0_1	0
_	1
0_1	2
_	3

Table 8.2: An Address Event Representation (AER) containing a 'greater than monogram symbol' (0_1), is able to transmit more information with a single event.

The listing in Table 8.2, is just one of the possible ways to recreate the dataset. In the example in Table 8.2 the initial state is `0_1`, which has only 1 outward bound transition from that gram, to _. Since it is 100% probable that the transition to _ will occur, it is written down as the second event in Table 8.2 or First Guess in Table 8.1. The gram _ has 3 emerging possible paths, `0_1`,`0_`,`0`. That means to choose the next guess, it's necessary to roll the 6-sided die. If the roll is 1 or 2, move to `0_1`, if the roll is 3 or 4, move to `0_`, if the roll is 5 or 6, move to `0`. Based on the die roll, create a new entry in the AER table that is being created from the random traversal of the Markov Chain shown in Figure 8.1. In the case of Table 8.2, the die roll was 2 and the second guess was `0_1`. That gram has only 1 outward bound state again, with 100% transition so it is written down as the third guess and added to the AER table.

A nearly dataset entry can correspond to an phase sensitive hierarchical Markov chain which is only able to emit a spike if all preceding spikes representing the order of the nearly complete dataset entry also take place, as shown in Table 8.2. When a spike is emitted from the hierarchical Markov chain, for the nearly complete dataset entry, as in Table 8.2 Trace ID _0_1, it is known that all preceding events necessary for it to emit have also completed, despite not needing to directly observe each of those spikes.

In summary, a representative temporal code for a complete dataset entry can be assembled by sequencing an address event representation with a timestamp update for each additional monogram, all the way through the maximum n-gram possible for the

dataset. This section demonstrated how an information source produced from a very small dataset, is able to transmit a message using AER in only $2\tau_m$.

8.3 Example of software 2D Data Reduction

A non-binary communication channel can generate data in a spiking channel with symbols that are capable of storing more than 1 bit per event. The encoder used to transform the original data into a non-binary digital spiking communication and computation channel is detailed in section 4.7, however, some of the details of it are repeated here, because it is also used to complete this example of 2D data reduction.

It is an address event representation of the original data, from a known information source encoder. The decoder needs to understand that the spiking channel is a stationary computation and communication from the information source ($|A_{simpleThis}|$)used to populate the channel.

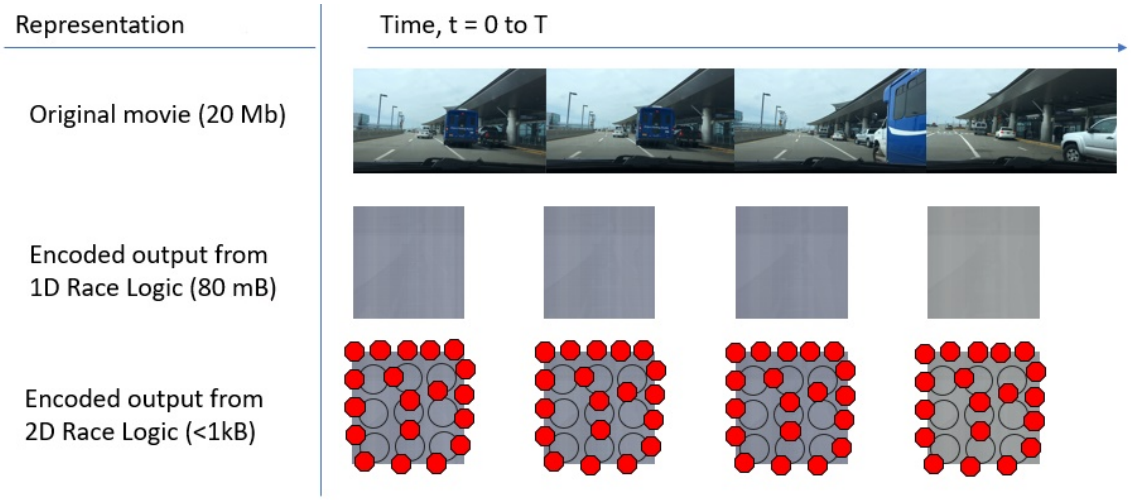


Figure 8.2: The representation, shown to the left of the time series, lists the data formats and the average size of that data in bits. (A) Original movie 20 Mb (B) Encoded output from 1D Race Logic 80 mB (C) Encoded Output from 2D Race Logic $< 1kB$.

To demonstrate, an example, depicted in Figure 8.2 starts with a streaming video

and produces a list of events taken from a 2D encoding race track. This encoder is using $|A_{simpleThis}|$. Comparing an AER representation of an image frame or a series of image frames vs a png or a series of png (i.e. Movie type), the AER representation is order of magnitudes less.

An example of software 2D data reduction is demonstrated through a software model which follows an encoding pattern shown in Figure 4.21 where first data is stored against a B-system and simply ergodic binary input pattern, spatially distributed across the input image, represented by the base message in Figure 8.3.

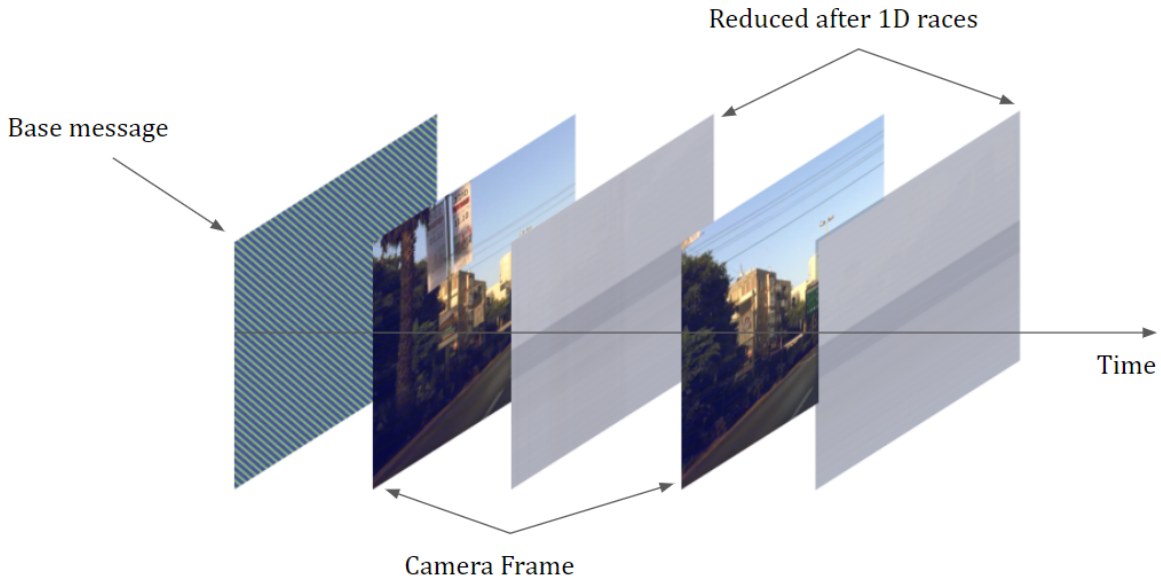


Figure 8.3: The software example of 2D data reduction initializes with a base message (t_0) which expects a b-system ergodic distribution of 0's and 1's which are spatially distributed across the base frame. As time proceeds it produces a data reduced frame representing the 1D output of parallel races happening between the base message + F_{t_s}, which is then used as the next base message for additional camera frames.

In this example, software accesses the video stream and mixes the base message with the frame image through a thresholding operation on each 8-bit color. At each frame the binary representation of the currently most likely monogram is mixed with the base image + frame image, to produce the 1D data reduction. In the example the

$|A_{simpleThis}|$ information source is tiled multiple times across a data reduction starter seed image. Then an updated 1D data reduced image is displayed on the screen. A series of overlapping grids representing 2D racetracks is hidden from the user, but it is used to calculate the parity of the underlying grid cells. A visualization of how the overlapping 2D racetracks calculate across the 1D data reduced image is shown in Figure 8.4. Each frame of the image has an XOR using parity taken across each of the 2D racetrack grid sections. The reason it is acceptable to treat the output of the XOR as the output of a race, is because a race is defined by a single $0 \rightarrow 1$ transition, as detailed in section 6.1.2. Once the output of an XOR event has passed, the logical order of the prior states are no longer relevant, and the temporal race has concluded, as detailed in section 6. The output of the grids produce and download a phase coded spike train file, representing spikes at a time on channel.

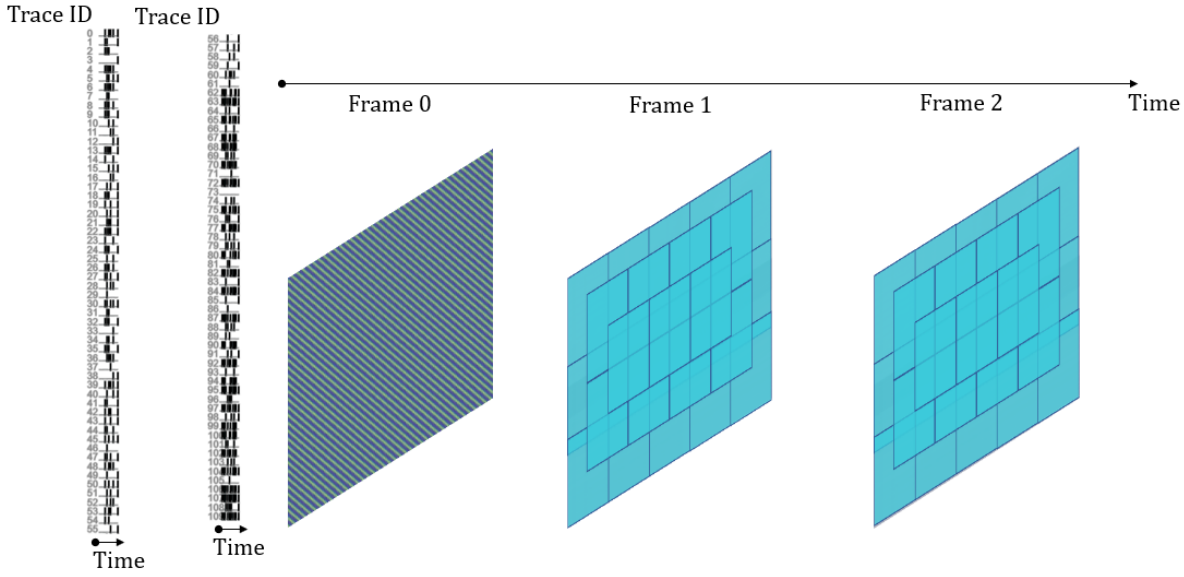


Figure 8.4: The 1D data reduced image has a series of 2D 'Racetrack' overlay's across the image, shown in teal blue for Frame 1 and Frame 2. This visualization shows how the series of overlapping racetrack generates address events for the spiking traces used to create the 110 traces that produce a reduced data stream. The trace ID's are shown to the left, for the first 21 frames.

The software algorithm is not an optimal or high speed solution for data reduction of a video stream, but it does provide information about how a 2D hardware data reduction scheme utilizing phase, rate and population codes would deliver an address event representation stream of data from the dataset $|A_{simpleThis}| = 24$. Since the input video stream is a large frame, the $|A_{simpleThis}|$ is tiled multiple times, until the entire image is covered by the b-system data used to create a population code for the 24 monogram traces. The n-gram traces are created from 110 traces and if the input is constant, will form a stationary broadcast that transmits the the input as a stationary periodic signal. Since the image itself is present, as the source of the arbitrary noise on the sensors, the channel will transmit the disturbances of its original noiseless stationary pattern.

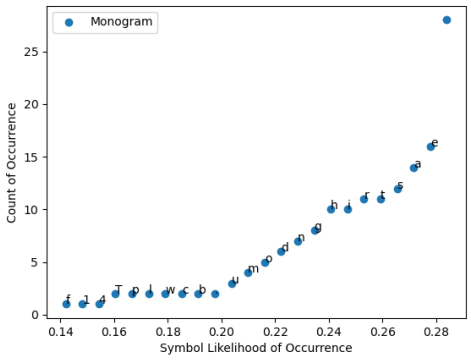
8.4 Dataset: simpleThis

Key	Label	Entry
0	a	This is a programming language and these are the fourteen words that describe this.
1	a	This is a programming language and these are the 14 words that describe this.

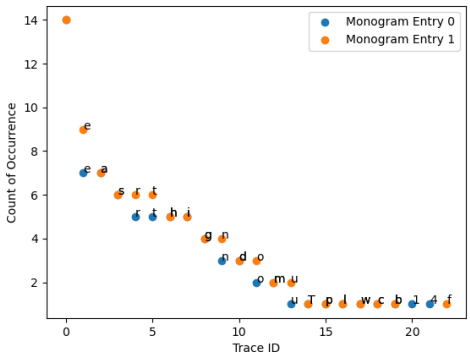
The dataset used to create simpleThis is two strings that have identical label but use different numbers of symbols to convey that label. The implications of the different symbols permits construction of a Markov chain hierarchy, where at the longest time of x, both strings are equivalent.

The original two strings are used to create a key entry pair, with the 'key' an artificially imposed label of **0** corresponding to the string with the word 'fourteen' and **1** corresponding to the string with the number '14'. The entire dataset consisting of the keys $0, 1$ is used to create a set of symbols shown in Figure 8.5a.

A software method to study simpleThis creates a spiking channel using 24 traces,



(a) The monogram symbols available for construction of an information source based on the dataset simpleThis vs the likelihood of receiving a particular symbol.



(b) Each dataset entry used in simpleThis has slightly different probability for occurrence. It is possible to create a 2D phase coded racetrack for either of the two entrys.

labeled 0-23. To create a representative spiking channel for transmitting a single symbol, add a spike at a timestep into the channel. The address event representation of the spiking channel is always limited for all time to only the traces 0-23.

Theoretically, the 24 traces could contain a population, rate, and a phase code, as detailed in Sections 4.9. Additionally, it is up to the author of an encoder to decide how to partition the energy use of the encoder between population, rate and phase assignment of the code.

To transmit one of the monograms as part of a neural code, with a population first encoding, a spike on a single trace is directly equivalent to sending that symbol. In order to use phase coded events, a layout of the information source which is able to emit events in accordance to the likelihood of receiving the symbol is used, as shown in Figure 8.6.

The length of time an encoded message could take is variable, up to the length of the combined dataset entries, and the encoding repeats once it has completed, so there is not a maximum time limit to transmit all messages, there is only the minimum time necessary for a single symbol to transmit.

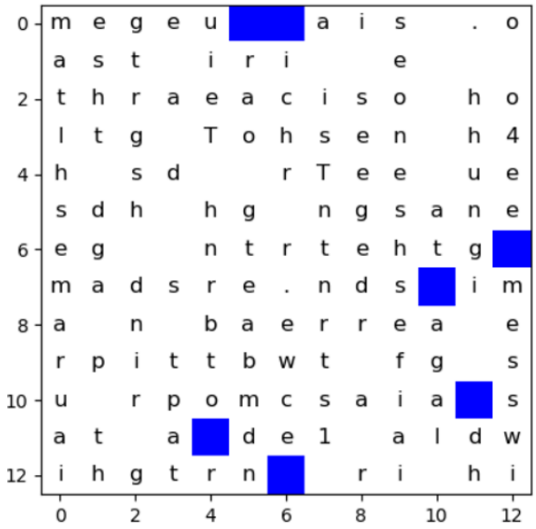


Figure 8.6: An information source has 283 individual circuits (dark blue tiles are empty tiles), each one emits a population code, from Figure 7.2 circuit 1, representing a single symbol in $|A_{simpleThis}|$. Then, an interconnect between the population code circuits connects into a looping circuit, like Figure 7.2 circuit 9, that can sustain rate or phase codes for the underlying symbol in $|A_{simpleThis}|$. This means it is possible to receive output of a symbol in $|A_{simpleThis}|$ as a population, rate, or phase code.

Chapter 9

Conclusion

This dissertation focused on discovering VLSI circuits of utility when arbitrary noise is affecting the communication signal. They make use of 'information is physical' to connect lower and upper bounds of energy use in the communication of semi-intelligent signals, to create a known stationary broadcast signal, that is permitted to be intentionally disturbed by the noise of the environment.

A small dataset, *simpleThis* was created, and used to build an easily learnable, predictable, non-binary digital signal for spiking communication and computation.

The physical connectivity properties, which are only present in hardware, are used to advantageously construct limited connectivity neuromorphic CMOS circuits out of a small dataset.

The small dataset produces a noise resilient data encoding schema that was used for a software demonstration in section 8.3 to show how a reduced communication and computation scheme transmits an address event representation signal, using physical pulse mode spikes, when exposed to 'noise' that is actually the video stream to be encoded.

The encoding schema permits for using a maximum amount of possible energy to transmit a single spike containing the complete dataset. If that single spike must carry

the entire message it must come from an information source capable of encoding the entire message. In that case, the spike capable of carrying the entire message cannot be received until $|A_\chi|$ spikes are received. Once $|A_\chi|$ symbols have arrived at the decoder, additional symbols may cause new information within a range of the size of $|A_\chi|$.

Bibliography

[1] P. Merolla, J. Arthur, R. Alvarez, J. M. Bussat, and K. Boahen, *A multicast tree router for multichip neuromorphic systems*, *IEEE Transactions on Circuits and Systems I: Regular Papers* **61** (2014), no. 3 820–833.

[2] J. Park, T. Yu, C. Maier, S. Joshi, and G. Cauwenberghs, *Live demonstration: Hierarchical Address-Event Routing architecture for reconfigurable large scale neuromorphic systems*, *ISCAS 2012 - 2012 IEEE International Symposium on Circuits and Systems* (2012) 707–711.

[3] S. Joshi, S. Deiss, M. Arnold, J. P. J. Park, T. Yu, and G. Cauwenberghs, *Scalable event routing in hierarchical neural array architecture with global synaptic connectivity*, *Cellular Nanoscale Networks and Their Applications (CNNA), 2010 12th International Workshop on* (2010).

[4] R. Landauer, *Irreversibility and Heat Generation in the Computing Process*, *IBM Journal of Research and Development* **5** (1961), no. 3 183–191.

[5] D. H. Wolpert, *Extending Landauers Bound from Bit Erasure to Arbitrary Computation*, .

[6] M. Miller, *Realization and Formal Analysis of Asynchronous Pulse Communication Circuits*. PhD thesis, UC Santa Barbara, 2015.

[7] C. E. Shannon, *A Symbolic Analysis of Relay and Switching Circuits*. PhD thesis, MIT, 1937.

[8] J. E. Smith, *Space-Time Computing with Temporal Neural Networks*, *Synthesis Lectures on Computer Architecture* **12** (2017), no. 2 i–215.

[9] C. Segal, A. Dalakoti, M. Miller, and F. Brewer, *Connectivity Effects on Energy and Area for Neuromorphic System with High Speed Asynchronous Pulse Mode Links*, *Proceedings of the 18th System Level Interconnect Prediction Workshop SLIP '16* (2016).

[10] C. Segal, A. Dalakoti, P. Mukim, M. Miller, and F. Brewer, *Low Power Circuits for Satellite Object Geolocation and Detection Methods*, 2018.

[11] G. Tzimpragos, G. Michelogiannakis, J. Volk, J. Shalf, and T. Sherwood, *A Computational Temporal Logic for Superconducting Accelerators*, .

[12] A. Madhavan, T. Sherwood, and D. Strukov, *Race Logic: A hardware acceleration for dynamic programming algorithms*, *Proceedings - International Symposium on Computer Architecture* (2014) 517–528.

[13] A. Madhavan, T. Sherwood, and D. Strukov, *A 4-mm 180-nm-CMOS 15-Giga-Cell-Updates-per-Second DNA Sequence Alignment Engine Based on Asynchronous Race Conditions*, in *CICC*, 2017.

[14] A. Krizhevsky, I. Sutskever, and G. E. Hinton, *ImageNet Classification with Deep Convolutional Neural Networks*, *Advances In Neural Information Processing Systems* (2012) 1–9.

[15] C. Eliasmith and C. H. Anderson, *Neural Engineering*, vol. 148.

[16] J. Sparsø, *Asynchronous circuit design - A tutorial*, *Principles of asynchronous circuit design - A systems Perspective* (2006) 1–152.

[17] A. J. Martin and M. Nyström, *Asynchronous techniques for system-on-chip design*, *Proceedings of the IEEE* **94** (2006), no. 6 1089–1120.

[18] C. E. Shannon, *A mathematical theory of communication*, *The Bell System Technical Journal* **27** (1948), no. July 1928 379–423.

[19] R. Sarpeshkar, *Analog versus digital: extrapolating from electronics to neurobiology.*, *Neural computation* **10** (1998), no. 7 1601–1638.

[20] E. W. Dijkstra, *On a methodology of design*, 1971.

[21] E. W. Dijkstra, *On the axiomatic definition of semantics*, 1973.

[22] A. Madhavan, *Abusing Hardware Race Conditions for High Throughput Energy Efficient Computation*. PhD thesis, 2016.

[23] C. Mead, *Analog VLSI and Neural Systems*, .

[24] M. Mahowald and R. Douglas, *A silicon neuron.*, *Nature* **354** (1991), no. 6354 515–8.

[25] R. C. Hardie and K. Franze, *Photomechanical Responses in Drosophila Photoreceptors*, *Science* **338** (2012), no. October 260–263.

[26] S. J. Lederman and R. L. Klatzky, *Hand movement: A window into haptic object recognition*, *Cognitive Psychology* **19** (1987) 342–368.

[27] I. Biederman, *On processing information from a glance at a scene: Some implications for a syntax and semantics of visual processing*, *ACM/SIGGRAPH workshop on User-oriented design of* (1976).

[28] A. Borji and L. Itti, *Human vs. computer in scene and object recognition*, *Cvpr* (2014).

[29] F. Klapproth, *Time and decision making in humans*, *Cognitive, Affective and Behavioral Neuroscience* **8** (2008), no. 4 509–524.

[30] M. Salami, C. Itami, T. Tsumoto, and F. Kimura, *Change of conduction velocity by regional myelination yields constant latency irrespective of distance between thalamus and cortex*, *Proceedings of the National Academy of Sciences of the United States of America* **100** (2003), no. 10 6174–6179.

[31] M. S. Brown and J. L. Goldstein, *Receptor System **, *Cell* **76** (1979), no. 7 3330–3337.

[32] D. J. C. MacKay, *Information Theory, Inference, and Learning Algorithms*. 2005.

[33] J. Jiao, K. Venkat, Y. Han, and T. Weissman, *Minimax Estimation of Functionals of Discrete Distributions*, *IEEE Transactions on Information Theory* **61** (2015), no. 5 2835–2885.

[34] C. E. Shannon, *Communication in the presence of noise*, *Proceedings of the IEEE* **86** (1998), no. 2 447–457.

[35] M. B. Plenio and V. Vitelli, *The physics of forgetting: Landauer's erasure principle and information theory*, *Contemporary physics* **42** (2001), no. 1 36.

[36] D. A. Levin, Y. Peres, and E. L. Wilmer, *Markov Chains and Mixing Times*. 2008.

[37] N. Ganesh and N. G. Anderson, *Irreversibility and dissipation in finite-state automata*, *Physics Letters, Section A: General, Atomic and Solid State Physics* **377** (2013), no. 45-48 3266–3271.

[38] J. Berkovitz, R. Frigg, and F. Kronz, *The ergodic hierarchy, randomness and Hamiltonian chaos*, *Studies in History and Philosophy of Science Part B - Studies in History and Philosophy of Modern Physics* **37** (2006), no. 4 661–691.

[39] R. Gray, *Entropy and Information Theory*. 1990.

[40] G. Indiveri, B. Linares-Barranco, T. J. Hamilton, A. van Schaik, R. Etienne-Cummings, T. Delbruck, S. C. Liu, P. Dudek, P. H??fliger, S. Renaud, J. Schemmel, G. Cauwenberghs, J. Arthur, K. Hynna, F. Folowosele, S. Saighi, T. Serrano-Gotarredona, J. Wijekoon, Y. Wang, and K. Boahen, *Neuromorphic silicon neuron circuits*, *Frontiers in Neuroscience* **5** (2011), no. MAY 1–23.

[41] J. Schemmel, D. Brüderle, A. Grübl, M. Hock, K. Meier, and S. Millner, *A wafer-scale neuromorphic hardware system for large-scale neural modeling*, *ISCAS 2010 - 2010 IEEE International Symposium on Circuits and Systems: Nano-Bio Circuit Fabrics and Systems* (2010) 1947–1950.

[42] S. Scholze, S. Schiefer, J. Partzsch, S. Hartmann, C. G. Mayr, S. Hoppner, H. Eisenreich, S. Henker, B. Vogginger, and R. Schuffny, *VLSI implementation of a 2.8 Gevent/s packet-based AER interface with routing and event sorting functionality*, *Frontiers in Neuroscience* **5** (2011), no. OCT 1–13.

[43] P. Bremner, Y. Liu, M. Samie, G. Dragffy, A. G. Pipe, G. Tempesti, J. Timmis, and A. Tyrrell, *SABRE: A bio-inspired fault tolerant electronic architecture*, *Bioinspiration and Biomimetics* **8** (2013).

[44] W. A. Little and G. L. Shaw, *A statistical theory of short and long term memory*, *Behavioral Biology* **14** (1975), no. 2 115–133.

[45] J. Hopfield, *Neural networks and physical systems with emergent collective computational abilities*, *Proc. Natl. Acad. Sci.* (1982).

[46] W. S. Mcculloch and W. Pitts, *A logical calculus of the ideas immanent in nervous activity*, *Bulletin of Mathematical Biophysics* **5** (1943) 115–133.

[47] J. Hertz, A. Krogh, and R. Palmer, *Introduction To the Theory of Neural Computation*, .

[48] T. Dean, *A Computational Model of Cerebral Cortex*, *AAAI* **5762** (2005), no. 2 458–465.

[49] B. Han, A. Ankit, A. Sengupta, and K. Roy, *Cross-Layer Design Exploration for Energy-Quality Tradeoffs in Spiking and Non-Spiking Deep Artificial Neural Networks*, *IEEE Transactions on Multi-Scale Computing Systems* **7766** (2017), no. 3 1–12.

[50] X. Dong, Y. Xie, and L. Smith, *Cost Analysis and Design Exploration for 3D ICs*, .

[51] Y. Zhang, L. Zhang, A. Deutsch, G. Katopis, D. Dreps, J. Buckwalter, E. Kuh, and C. Cheng, *Design methodology of high performance on-chip global interconnect using terminated transmission-line*, *Quality of Electronic Design, 2009. ISQED 2009. Quality Electronic Design* (2009) 451458.

[52] M. A. Horowitz, M. Jeeradit, F. Lau, S. Liao, B. Lim, and J. Mao, *Digital Analog Design*, .

[53] N. Davey, L. Calcraft, and R. Adams, *High capacity, small world associative memory models*, *Connection Science* **18** (2006), no. 3 247–264.

[54] P. Christie and D. Stroobandt, *The interpretation and application of rent's rule*, *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* **8** (2000), no. 6 639–648.

[55] 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. D. Rubin, F. Akopyan, E. McQuinn, W. P. Risk, and D. S. Modha, *Cognitive computing building block: A versatile and efficient digital neuron model for neurosynaptic cores*, *Proceedings of the International Joint Conference on Neural Networks* (2013).

[56] D. K. Ferry, *Interconnection Lengths and WLSI*, *IEEE Circuits and Devices Magazine* **1** (1985), no. 4 39–42.

[57] P. A. Merolla, J. V. Arthur, R. Alvarez-Icaza, A. S. Cassidy, J. Sawada, F. Akopyan, B. L. Jackson, N. Imam, C. Guo, Y. Nakamura, B. Brezzo, I. Vo, S. K. Esser, R. Appuswamy, B. Taba, A. Amir, M. D. Flickner, W. P. Risk, R. Manohar, and D. S. Modha, *A million spiking-neuron integrated circuit with a scalable communication network and interface*, *Science* **345** (2014), no. 6197 668–673.

[58] M. Miller, G. Hoover, and F. Brewer, *Pulse-mode link for robust, high speed communications*, *Proceedings - IEEE International Symposium on Circuits and Systems* **5** (2008) 3073–3077.

[59] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, *Intriguing Properties of Neural Networks*, *Iclr* (2013) 1–10.

[60] Y. Jia, J. Abbott, J. L. Austerweil, T. L. Griffiths, and T. Darrell, *Visual Concept Learning: Combining Machine Vision and Bayesian Generalization on Concept Hierarchies*, *Advances in Neural Information Processing Systems 27 (NIPS 2013)* **1** (2013), no. 1 1–9.

[61] C. H. Bennet, *The thermodynamics of computation – a review*, 1982.

[62] A. Neckar, S. Fok, B. V. Benjamin, T. C. Stewart, N. N. Oza, A. R. Voelker, C. Eliasmith, R. Manohar, and K. Boahen, *Braindrop: A Mixed-Signal Neuromorphic Architecture with a Dynamical Systems-Based Programming Model*, *Proceedings of the IEEE* **107** (2019), no. 1 144–164.

[63] W. Maass, *Lower Bounds for the Computational Power of Networks of Spiking Neurons*, *Neural Computation* **8** (1996), no. 1 1–40.

[64] W. Gerstner, W. M. Kistler, R. Naud, and L. Paninski, *Neuronal Dynamics*. Cambridge University Press, 2014.

[65] C. Eliasmith and C. H. Anderson, *Neural Engineering: computation, representation, and dynamics in neurobiological systems*, vol. 148. 2003.

[66] M. a. Srinivasan, J. M. Whitehouse, and R. H. LaMotte, *Tactile detection of slip: surface microgeometry and peripheral neural codes.*, 1990.

[67] A. R. Voelker and C. Eliasmith, *Programming Neuromorphics Using the Neural Engineering Framework*. 2021.

[68] A. Dalakoti, C. Segal, M. Miller, and F. Brewer, *Asynchronous High Speed Serial Links Analysis using Integrated Charge for Event Detection*, *Proceedings of the 26th edition on Great Lakes Symposium on VLSI - GLSVLSI '16* (2016) 121–124.

[69] R. Darshan, W. E. Wood, S. Peters, A. Leblois, and D. Hansel, *A canonical neural mechanism for behavioral variability*, *Nature Communications* **8** (2017), no. May 1–13.

[70] R. V. L. Hartley, *Information and transmission*, 1927.

[71] R. M. Gray, *Probability, random processes, and ergodic properties*, *Probability, Random Processes, and Ergodic Properties* (2009) 1–322.

[72] D. J. C. Mackay and R. M. Neal, *Near Shannon Limit Performance of Low Density Parity Check Codes*, *Electronics Letters* (1996) 3–6.

[73] M. Nystrom, *Asynchronous Pulse Logic*. PhD thesis, California Institute of Technology, 2001.

[74] D. MacKay and R. Neal, *Near Shannon limit performance of low density parity check codes*, *Electronics Letters* **33** (1997), no. 6 457.

[75] M. Nielsen, *Chap 4 A visual proof that neural nets can compute any function*, *Neural Networks and Deep Learning* (2015) 1–29.

[76] J. Hasler and B. Marr, *Finding a roadmap to achieve large neuromorphic hardware systems*, *Frontiers in Neuroscience* **7** (2013), no. 7 SEP 1–29.

[77] S. Wolfram, *A New Kind of Science*. 2002.

[78] X. L. Dong, E. Gabrilovich, G. Heitz, W. Horn, N. Lao, K. Murphy, T. Strohmann, S. Sun, and W. Zhang, *Knowledge Vault: A Web-Scale Approach to Probabilistic Knowledge Fusion*, *Kdd* (2014) 601–610.

[79] R. Socher, D. Chen, C. D. Manning, and A. Y. Ng, *Reasoning With Neural Tensor Networks for Knowledge Base Completion.*, *Nips* (2013) 926–934.

[80] D. Harvey and S. Moran, *Matching Words and Pictures .*, *Evaluation* **3** (2009) 1–40.

[81] J. L. Hennessy and D. a. Patterson, *Computer Architecture, Fifth Edition: A Quantitative Approach*, 2011.

[82] J. Zhao, Q. Zou, and Y. Xie, *Overview of 3D Architecture Design Opportunities and Techniques*, *IEEE Design & Test* **PP** (2015), no. 99 1–1.

[83] Mellanox Technologies, *Introduction to InfiniBand*, *Technical Report* (2003) 1–20.

[84] S. R. Kulkarni, *Information, Entropy, and Coding*, in *Lecture Notes for Introduction to Electrical Signals and Systems.* 2002.

[85] J. H. David Patterson, *Computer Organization and Design*, *Computer Organization and Design* (2012).

[86] P. A. Franaszek and A. X. Widmer, *A DC-Balanced, Partitioned-Block, 8B/10B Transmission Code*, *IBM J. RES. DEVELOP.* **27** (1983), no. 5.

[87] L. Szilard, *ON THE DECREASE OF ENTROPY IN A THERMODYNAMIC SYSTEM BY THE INTERVENTION OF INTELLIGENT BEINGS*, *Zeitschrift fur Physik* (0).

[88] H. Nyquist, *Certain Factors Affecting Telegraph Speed*, .

[89] N. Tishby, F. C. Pereira, and W. Bialek, *The information bottleneck method*, *arXiv: Physics* (1999) 1–16, [0004057v1].

[90] A. Karbasi, A. H. Salavati, A. Shokrollahi, and L. R. Varshney, *Noise-Enhanced Associative Memories*, .

[91] E. C. Posner, E. R. Rodemich, and S. S. Venkatesh, *The Capacity of the Hopfield Associative Memory*, *IEEE Transactions on Information Theory* **I** (1987), no. 4.

[92] L. R. Iyer, Y. Chua, and H. Li, *Is Neuromorphic MNIST neuromorphic? Analyzing the discriminative power of neuromorphic datasets in the time domain*, .

[93] M. P. Miller, G. Magazzu, and F. D. Brewer, *5 Gbps Radiation-Hardened Low-Power Pulse Serial Link*, *IEEE Transactions on Nuclear Science* **63** (2016), no. 1 203–212.

[94] S. J. Thorpe, A. Delorme, and R. Vanrullen, *Spike-based strategies for rapid processing Spike-based strategies for rapid processing*, *Neural Networks* **14** (2001), no. 33 715–726.

[95] T. Masquelier, E. Hugues, G. Deco, and S. J. Thorpe, *Oscillations, Phase-of-Firing Coding, and Spike Timing-Dependent Plasticity: An Efficient Learning Scheme*, *Journal of Neuroscience* **29** (2009), no. 43 13484–13493.

[96] W. Li and Y. Li, *Entropy , mutual information , and systematic measures of structured spiking neural networks*, *Journal of Theoretical Biology* **501** (2020) 110310.

[97] P. Mukim, *Analysis and Design of Precision Timing Circuits using Pulse Mode Event Signaling*. PhD thesis, University of California Santa Barbara, 2021.

[98] C. Eurich, *Neural Dynamics and Neural Coding*, .

[99] B. Cramer, D. Stöckel, M. Kreft, M. Wibral, J. Schemmel, K. Meier, and V. Priesemann, *Control of criticality and computation in spiking neuromorphic networks with plasticity*, *Nature Communications* **11** (2020), no. 1 1–11.

[100] J. Arthur and K. Boahen, *Learning in silicon: Timing is everything*, *Advances in neural information processing systems 17* **23** (2006), no. 7 281.

[101] Y. Mi, L. Li, D. Wang, and S. Wu, *A synaptical story of persistent activity with graded lifetime in a neural system*, in *Advances in Neural Information Processing Systems*, vol. 1, pp. 352–360, 2014.

[102] Y. Li and H. Xu, *Stochastic neural field model: multiple firing events and correlations*, *Journal of Mathematical Biology* **79** (2019), no. 4 1169–1204.

[103] C. Ritchie, C. Andres, and P. Anikeeva, *Neural Recording and Modulation Technologies*, *Nature Reviews Materials* (2019).

[104] M. Costa, A. L. Goldberger, and C. K. Peng, *Multiscale entropy analysis of biological signals*, *Physical Review E - Statistical, Nonlinear, and Soft Matter Physics* **71** (2005), no. 2 1–18.

[105] Y. Cao, D. Patnaik, S. Ponce, J. Archuleta, P. Butler, W.-c. Feng, and N. Ramakrishnan, *Towards Chip-on-Chip Neuroscience : Fast Mining of Neuronal Spike Streams Using Graphics Hardware*, *Architecture* (2010) 1–10.

[106] M. Sunnraker, A. G. G. Busetto, E. Numminen, J. Corander, M. Foll, and C. Dessimoz, *Approximate Bayesian computation*, *PLoS computational biology* **9** (2013), no. 1 e1002803+.

[107] J. Sanchez, D. Erdogmus, M. Nicolelis, J. Wessberg, and J. Principe, *Interpreting Spatial and Temporal Neural Activity Through a Recurrent Neural Network BrainMachine Interface*, *IEEE Transactions on Neural Sysyems and Rehabilitation Engineering* **13** (2005), no. June 2005.

[108] E. Archer, I. M. Park, J. W. Pillow, and S. Computation, *Bayesian and Quasi-Bayesian Estimators for Mutual Information from Discrete Data*, .

[109] H. Jaeger, *Towards a generalized theory comprising digital , neuromorphic and unconventional computing*, *Neuromorphic Computing and Engineering* **1** (2021).

[110] T. Sejnowski and T. Delbruck, *THE LANGUAGE OF THE BRAIN*, .

[111] L. Zullo, M. Chiappalone, S. Martinoia, and F. Benfenati, *A "Spike-Based" Grammar Underlies Directional Modification in Network Connectivity: Effect on Bursting Activity and Implications for Bio-Hybrids Systems*, *PLoS ONE* **7** (2012), no. 11 1–10.

[112] B. Zhang, J. Gong, W. Zhang, R. Xiao, J. Liu, and X. Z. Xu, *Braingut communications via distinct neuroendocrine signals bidirectionally regulate longevity in C. elegans*, *Genes and Development* **32** (2018), no. 3-4 258–270.

[113] C. G. Antonopoulos, E. Bianco-Martinez, and M. S. Baptista, *Evaluating performance of neural codes in model neural communication networks*, *Neural Networks* **109** (2019) 90–102.

[114] A. C. Snyder, B. M. Yu, and M. A. Smith, *Distinct population codes for attention in the absence and presence of visual stimulation*, *Nature Communications* **9** (2018), no. 1.

[115] K. Hartmann, E. E. Thomson, I. Zea, R. Yun, P. Mullen, J. Canarick, A. Huh, and M. A. L. Nicolelis, *Embedding a Panoramic Representation of Infrared Light in the Adult Rat Somatosensory Cortex through a Sensory Neuroprosthesis*, *Journal of Neuroscience* **36** (2016), no. 8 2406–2424.

[116] A. Dettner, S. Münzberg, and T. Tchumatchenko, *Temporal pairwise spike correlations fully capture single-neuron information*, *Nature Communications* **7** (2016) 1–11.

[117] R. S. Johansson and I. Birznieks, *First spikes in ensembles of human tactile afferents code complex spatial fingertip events.*, *Nature neuroscience* **7** (2004), no. 2 170–177.

[118] R. Sarpeshkar and M. O'Halloran, *Scalable hybrid computation with spikes.*, *Neural computation* **14** (2002), no. 2002 2003–2038.

[119] M. Payvand, *Area-efficient Neuromorphic Silicon Circuits and Architectures using Spatial and Spatio-Temporal Approaches*. PhD thesis, 2016.

[120] W. W. J. Dally and J. W. J. Poulton, *Digital systems engineering*. 1998.

[121] M. G. Luby, M. Mitzenmacher, M. a. Shokrollahi, and D. a. Spielman, *Improved low-density parity-check codes using irregular graphs*, *IEEE Transactions on Information Theory* **47** (2001), no. 2 585–598.

[122] R. G. Gallager, *Low-density parity-check codes*, .

[123] S. J. Johnson, *Introducing Low-Density Parity-Check Codes*, *Electrical Engineering* (2005).

[124] T. J. Richardson, M. a. Shokrollahi, and R. L. Urbanke, *Design of capacity-approaching irregular low-density parity-check codes*, *IEEE Transactions on Information Theory* **47** (2001), no. 2 619–637.

[125] M. G. Luby, M. Mitzenmacher, M. A. Shokrollahi, and D. A. Spielman, *Efficient erasure correcting codes*, *IEEE Transactions on Information Theory* **47** (2001), no. 2 569–584.

[126] A. Steimer, W. Maass, and R. Douglas, *Belief propagation in networks of spiking neurons.*, *Neural computation* **21** (2009) 2502–2523.

[127] Y. Bengio, *Evolving culture vs local minima*, *arXiv preprint arXiv:1203.2990* **2006** (2012), no. 2009 1–28.

[128] P. K. Kuhl, *Human speech and birdsong: Communication and the social brain*, *Proceedings of the National Academy of Sciences of the United States of America* **100** (2003), no. 17 9645–9646.

[129] D. Aronov, A. S. Andalman, and M. S. Fee, *A Specialized Forebrain Circuit for Vocal Babbling in the Juvenile Songbird*, *Science* **320** (2008), no. May 630–635.

[130] J. Weston, S. Chopra, and A. Bordes, *Memory networks*, *Iclr* (2015) 1–15.

[131] C. Republic and T. Mikolov, *Statistical Language Models Based on Neural Networks*. PhD thesis, 2012.

[132] M. Davies, N. Srinivasa, T. H. Lin, G. Chinya, Y. Cao, S. H. Choday, G. Dimou, P. Joshi, N. Imam, S. Jain, Y. Liao, C. K. Lin, A. Lines, R. Liu, D. Mathaikutty, S. McCoy, A. Paul, J. Tse, G. Venkataramanan, Y. H. Weng, A. Wild, Y. Yang, and H. Wang, *Loihi: A Neuromorphic Manycore Processor with On-Chip Learning*, *IEEE Micro* **38** (2018), no. 1 82–99.

[133] K. Lai, M. J. Robertson, and D. V. Schaffer, *The sonic hedgehog signaling system as a bistable genetic switch.*, *Biophysical journal* **86** (2004), no. 5 2748–57.

[134] T. S. Kraft, V. V. Venkataraman, I. J. Wallace, A. N. Crittenden, N. B. Holowka, J. Stieglitz, J. Harris, D. A. Raichlen, B. Wood, M. Gurven, and H. Pontzer, *The energetics of uniquely human subsistence strategies*, *Science* **374** (2021), no. 6575.

[135] C. E. Carr and S. Amagai, *Processing of temporal information in the brain*, *Advances in Psychology* **115** (1996), no. C 27–52.

[136] R. Z. Shi and T. K. Horiuchi, *A neuromorphic VLSI model of bat interaural level difference processing for azimuthal echolocation*, *IEEE Transactions on Circuits and Systems I: Regular Papers* **54** (2007), no. 1 74–88.

[137] A. Zimmerman, L. Bai, and D. D. Ginty, *The gentle touch receptors of mammalian skin.*, *Science (New York, N.Y.)* **346** (2014), no. 6212 950–4.

[138] I. D. Neveln, A. Tirumalai, and S. Sponberg, *Information-based centralization of locomotion in animals and robots*, *Nature Communications* **10** (2019), no. 1 1–11.

[139] R. J. De Marco, J. M. Gurevitz, and R. Menzel, *Variability in the encoding of spatial information by dancing bees.*, *The Journal of experimental biology* **211** (2008), no. Pt 10 1635–1644.

[140] K. Lu, Y. Xu, P. Yin, A. J. Oxenham, J. B. Fritz, and S. A. Shamma, *Temporal coherence structure rapidly shapes neuronal interactions*, *Nature Communications* **8** (2017).

[141] C. D. Brody and J. J. Hopfield, *Simple networks for spike-timing-based computation, with application to olfactory processing*, *Neuron* **37** (2003), no. 5 843–852.

[142] R. Plutchik, *The Nature of Emotions*, *American Scientist* **89** (2001), no. 4 344–350.

[143] L.-s. Young, *HOW WELL DO REDUCED MODELS CAPTURE*, .

[144] A. Dalakoti, *Collective Pulse Dynamics: A New Timing Circuit Strategy*. PhD thesis, University of California Santa Barbara, 2018.

[145] A. Graves, G. Wayne, and I. Danihelka, *Neural Turing Machines*, *Arxiv* (2014) 1–26.

[146] P. Dlugosch, D. Brown, P. Glendenning, M. Leventhal, and H. Noyes, *An efficient and scalable semiconductor architecture for parallel automata processing*, *IEEE Transactions on Parallel and Distributed Systems* **25** (2014), no. 12 3088–3098.

[147] J. Connelly, *Ternary Computing Testbed 3 ! Trit Computer Architecture*, *Power* (2008).

[148] C. Bissell, *Historical perspectives - The Moniac a Hydromechanical Analog Computer of the 1950s*, *IEEE Control Systems* **27** (2007), no. 1 69–74.

[149] F. Alibart, T. Sherwood, and D. B. Strukov, *Hybrid CMOS/Nanodevice Circuits for High Throughput Pattern Matching Applications Fabien*, *2011 NASA/ESA Conference on Adaptive Hardware and Systems (AHS)* (2011) 279–286.

[150] V. S. Denchev, S. Boixo, S. V. Isakov, N. Ding, R. Babbush, V. Smelyanskiy, J. Martinis, and H. Neven, *What is the Computational Value of Finite Range Tunneling?*, .

[151] M. P. Frank and E. P. Debenedictis, *A Novel Operational Paradigm for Thermodynamically Reversible Logic Adiabatic Transformation of Chaotic Nonlinear Dynamical Circuits*, .

[152] M. Miller and F. Brewer, *Formal Verification of Analog Circuit Parameters across Variation Utilizing SAT*, *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2013* (2013) 1442–1447.

[153] G. Tzimpragos, A. Madhavan, D. Vasudevan, D. Strukov, and T. Sherwood, *Boosted Race Trees for Low Energy Classification*, *International Conference on Architectural Support for Programming Languages and Operating Systems - ASPLOS* (2019) 215–228.

[154] A. Madhavan, T. Sherwood, and D. Strukov, *Race Logic: Abusing Hardware Race Conditions to Perform Useful Computation*, *IEEE Micro* **35** (2015), no. 3 48–57.

[155] M. Falahatgar, A. Orlitsky, V. Pichapati, and A. T. Suresh, *Learning Markov distributions: Does estimation trump compression?*, *IEEE International Symposium on Information Theory - Proceedings* **2016-Augus** (2016) 2689–2693.

[156] B. D. Winters and M. R. Greenstreet, *A Negative-Overhead , Self-Timed Pipeline*, *Proceedings - International Symposium on Asynchronous Circuits and Systems* (2002).

[157] B. D. Winters and M. R. Greenstreet, *Surfing: A robust form of wave pipelining using self-timed circuit techniques*, *Microprocessors and Microsystems* **27** (2003), no. 9 409–419.

[158] S. Y. Chung, G. David Forney, T. J. Richardson, and R. Urbanke, *On the design of low-density parity-check codes within 0.0045 dB of the Shannon limit*, *IEEE Communications Letters* **5** (2001), no. 2 58–60.

[159] G. Paneru, S. Dutta, T. Sagawa, T. Tlusty, and H. K. Pak, *Efficiency fluctuations and noise induced refrigerator-to-heater transition in information engines*, *Nature Communications* **11** (2020), no. 1 1–8.

[160] A. P. Chandrakasan and R. W. Brodersen, *Minimizing power consumption in digital CMOS circuits*, *High-Performance System Design: Circuits and Logic* **83** (1999), no. 4 172–197.

[161] A. P. Chandrakasan, S. Sheng, and R. W. Brodersen, *Low-Power CMOS Digital Design*, *Low-Power CMOS Design* **27** (1998), no. 4 36–46.

[162] M. D. Riedel, *Cyclic Combinational Circuits*, .

[163] C. Y. C. Yang, V. Singhal, and M. Ciesielski, *BDD decomposition for efficient logic synthesis*, *Proceedings 1999 IEEE International Conference on Computer Design: VLSI in Computers and Processors (Cat. No.99CB37040)* (1999) 1–7.

[164] F. R. Kschischang, B. J. Frey, and H. a. Loeliger, *Factor graphs and the sum-product algorithm*, *IEEE Transactions on Information Theory* **47** (2001), no. 2 498–519.

[165] S. Johnson, *An Introduction to Low-Density Parity-Check Codes*, *Electrical Engineering* (2002) 175–197.

[166] D. J. C. MacKay, *Encyclopedia of Sparse Graph Codes*, 1999.

[167] C. H. Bennett, *Notes on Landauer's principle, reversible computation, and Maxwell's Demon*, *Studies in History and Philosophy of Science Part B - Studies in History and Philosophy of Modern Physics* **34** (2003), no. 3 501–510.

[168] M. Miller, C. Segal, D. Mc Carthy, A. Dalakoti, P. Mukim, and F. Brewer, *Impolite High Speed Interfaces with Asynchronous Pulse Logic*, *Proceedings of the 2018 on Great Lakes Symposium on VLSI - GLSVLSI '18* (2018), no. May 99–104.

[169] M. Nyström and A. J. Martin, *Asynchronous pulse logic*. Springer, 2002.

[170] B. D. Winters and M. R. Greenstreet, *A negative-overhead, self-timed pipeline*, in *ASYNC 2002*, pp. 37–46, IEEE, 2002.

[171] M. R. Greenstreet and J. Ren, *Surfing interconnect*, in *ASYNC 2006*.

[172] M. Miller, G. Hoover, and F. Brewer, *Pulse-mode link for robust, high speed communications*, in *ISCAS 2008*.

[173] I. Sutherland and S. Fairbanks, *Gasp: a minimal fifo control*, in *ASYNC 2001*, pp. 46–53, 2001.

[174] I. E. Sutherland, *Micropipelines*, *Communications of the ACM* **32** (1989), no. 6 720–738.

[175] R. Ho, K. W. Mai, and M. A. Horowitz, *The future of wires*, *Proceedings of the IEEE* **89** (2001), no. 4 490–504.

[176] A. Devgan and C. Kashyap, *Block-based static timing analysis with uncertainty*, in *ICCAD 2003*, p. 607, IEEE Computer Society, 2003.

[177] V. Narayanan, B. A. Chappell, and B. M. Fleischer, *Static timing analysis for self resetting circuits*, in *ICCAD 1996*.

[178] T. J. Chaney and C. E. Molnar, *Anomalous behavior of synchronizer and arbiter circuits*, *IEEE Transactions on computers* **22** (1973), no. 4 421–422.

[179] J. C. Ebergen, S. Fairbanks, and I. E. Sutherland, *Predicting performance of micropipelines using charlie diagrams*, in *ASYNC 1998*.

[180] A. Winstanley and M. Greenstreet, *Temporal properties of self-timed rings*, in *Advanced Research Working Conference on Correct Hardware Design and Verification Methods*, pp. 140–154, Springer, 2001.

[181] A. Dalakoti, M. Miller, and F. Brewer, *Pulse ring oscillator tuning via pulse dynamics*, in *2017 IEEE 35th International Conference on Computer Design (ICCD)*, pp. 469–472, IEEE, 2017.

[182] I. Bloom and Y. Nemirovsky, *1/f noise reduction of metal-oxide-semiconductor transistors by cycling from inversion to accumulation*, *Applied Physics Letters* **58** (1991), no. 15 1664–1666.

[183] R. Z. Bhatti, M. Denneau, and J. Draper, *2 gbps serdes design based on ibm cu-11 (130nm) standard cell technology*, in *Proceedings of the 16th ACM Great Lakes symposium on VLSI*, pp. 198–203, ACM, 2006.

[184] P. Wang, Z. Wang, C. Zhang, and Z. Wang, *Data lane design for transmitter of 4.8 gbps serdes in 65nm cmos*, in *Electron Devices and Solid-State Circuits (EDSSC), 2014 IEEE International Conference on*, pp. 1–2, IEEE, 2014.

[185] J. E. Proesel and T. O. Dickson, *A 20-gb/s, 0.66-pj/bit serial receiver with 2-stage continuous-time linear equalizer and 1-tap decision feedback equalizer in 45nm soi cmos*, in *VLSI Circuits (VLSIC), 2011 Symposium on*, pp. 206–207, IEEE, 2011.

[186] A. Oliva and A. Torralba, *Modeling the shape of the scene: A holistic representation of the spatial envelope*, *International Journal of Computer Vision* **42** (2001), no. 3 145–175.

[187] D. P. Landau and K. Binder, *A guide to Monte-Carlo similations in statistical physics*. 2009.

[188] S. Idrees, M. P. Baumann, F. Franke, T. A. Münch, and Z. M. Hafed, *Perceptual saccadic suppression starts in the retina*, *Nature Communications* **11** (2020), no. 1 1–19.

[189] W. Bainbridge and S. Furber, *Delay insensitive system-on-chip interconnect using 1-of-4 data encoding*, in *Proceedings Seventh International Symposium on Asynchronous Circuits and Systems. ASYNC 2001*, pp. 118–126, 2001.

[190] T. Edwards, *Introduction to the SkyWater PDK: The New Age of Open Source Silicon*, tech. rep., 2021.

[191] D. Beniaguev, I. Segev, and M. London, *Single cortical neurons as deep artificial neural networks*, *Neuron* **109** (2021), no. 17 2727–2739.

[192] A. Gidon, T. A. Zolnik, P. Fidzinski, F. Bolduan, A. Papoutsi, P. Poirazi, M. Holtkamp, I. Vida, and M. E. Larkum, *Dendritic action potentials and computation in human layer 2/3 cortical neurons*, *Science* **367** (2020), no. 6473 83–87.

[193] T. Bekolay, J. Bergstra, E. Hunsberger, T. DeWolf, T. C. Stewart, D. Rasmussen, X. Choo, A. R. Voelker, and C. Eliasmith, *Nengo: A Python tool for building large-scale functional brain models*, *Frontiers in Neuroinformatics* **7** (2014), no. JAN 1–13.

[194] P. Janssen and M. N. Shadlen, *A representation of the hazard rate of elapsed time in macaque area LIP*, *Nature Neuroscience* **8** (2005), no. 2 234–241.

[195] K. J. Friston and K. E. Stephan, *Free-energy and the brain*, *Synthese* **159** (2007), no. 3 417–458.