COMMUNICATING AND COMPUTING WITH SPIKES IN NEUROMORPHIC SYSTEMS

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF ELECTRICAL ENGINEERING
AND THE COMMITTEE ON GRADUATE STUDIES
OF STANFORD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Sam Fok
June 2018

This dissertation is online at: http://purl.stanford.edu/qk018rf9117

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Kwabena Boahen, Primary Adviser

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Oussama Khatib

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Krishna Shenoy

Approved for the Stanford University Committee on Graduate Studies.

Patricia J. Gumport, Vice Provost for Graduate Education

This signature page was generated electronically upon submission of this dissertation in electronic format. An original signed hard copy of the signature page is on file in University Archives.

Abstract

We provide an overview of neuromorphic engineering and describe two contributions to Braindrop, a state-of-the-art neuromorphic system. First, we describe a method for performing summing and weighting of spike trains by accumulative thinning, a deterministic procedure for merging and dropping spikes. Previous methods relied on probabilistic thinning, which results in Poissonian statistics. As a result, when the thinned spike-train is filtered with a first-order low-pass synapse, the signal-to-noise ratio (SNR) scales as the square-root of its rate. For our accumulative thinning method, the SNR depends on the weight w; it scales linearly in the best-case scenario ($w \to 0$) and as the square-root in the worst-case ($w \to 1$). We find that a three-quarter power scaling minimizes energy consumption.

Second, we present a serial H-tree router for two-dimensional (2D) arrays. Existing routing mechanisms for 2D arrays either use low-overhead grids with one or two shared wires per row or column (e.g., RAM) or high-overhead meshes with many wires connecting neighboring clients (e.g., supercomputers). Neither is suitable for intermediate-complexity clients (e.g., small clusters of silicon neurons). We present a router tailored to 2D arrays of such clients. It uses a tree laid out in a fractal pattern (H-tree), which requires less wiring per signal than a grid, and adopts serial-signaling, which keeps link-width constant, regardless of payload size. To route from the tree's leaves to its root (or vise versa), each node prepends (consumes) a delay-insensitive 1-of-4 code that signals the route's previous (next) branch; additional codes carry payload. We employ this serial H-tree router to service a 16×16 array of silicon-neuron clusters, each with 16 spike-generating analog somas, 4 spike-consuming analog synapses, and one 128-bit SRAM. Fabricated in a 28-nm CMOS process, the router communicates 26.8M soma-generated and 18.3M synapse-targeted spikes per second while occupying 43% of the client's $35.1\times 36.1\mu\text{m}^2$.

Acknowledgments

I thank Chris Eliasmith and Terry Stewart of the Center for Theoretical Neuroscience at the University of Waterloo for the development and assistance with the Neural Engineering Framework that motivated the Braindrop project. I thank Rajit Manohar of the Computer Systems Lab at Yale University for instruction on asynchronous circuit design and for the CAD tools that allow their synthesis and verification in a synchronous world. I thank my colleagues at the Brains in Silicon Lab of Stanford University for countless coffee walks and their stimulating discussions. In particular, I could not have asked for better teammates than Ben Benjamin, Alex Neckar, and Nick Oza on the Braindrop project. And finally I have been immensely fortunate to have Kwabena Boahen as advisor and thank him for his tireless drive for understanding. The works herein were made possible by grants from the Office of Naval Research and the National Science Foundation.

Contents

List of Tables

List of Figures

1

Artificial Intelligence and Neuromorphic Engineering

Humans have long analogized the human brain to their computational tools and sought to imbue their tools with human-like, artificial intelligence (AI). For the industrial age, the brain was like a steam-powered system of pipes and actuators driving the body; for the digital age, the brain is like a digital computer, storing and processing bits of data. However, instead of conceptualizing the brain in terms of the current means of computation, neuromorphic engineers move in the other direction and conceive of computational methods from the structure and function of the brain. The neuromorphic approach comes with good reason: on a power budget of 20W (slightly more than one-half cup of sugar per day), the brain, with 86 billion neurons [18] and thousands of synaptic connections per neuron, still roundly beats existing methods at tasks we take for granted like walking around, having intelligible conversations, and making sense of the world.

That the brain has useful lessons for computation was not as widely accepted in the recent past as it is today in 2018. However, since AlexNet won the ImageNet challenge in 2012 and beat the nearest competitor by an unheard of margin (10.8 percent) [23], brain-inspired artificial neural networks (ANNs) and the hardware to run them have returned the mainstream conscious as a viable approach for AI. ANNs like AlexNet are loosely based on the connectivity of cortex in the brain and contain from thousands of simple units, called ”neurons”, arranged and connected in layers (Fig. 1.1). Each neuron computes a simple nonlinear function of the sum of its inputs, and each connection weights the source neuron’s output, or activation, and delivers it as input to a destination neuron.

Prior to AlexNet’s success, algorithms geared towards AI were largely not brain-inspired and hardware design efforts were concentrated on powerful, general-purpose central processing units (CPUs), which operate synchronously and digitally: a central clock governs the flow of data and

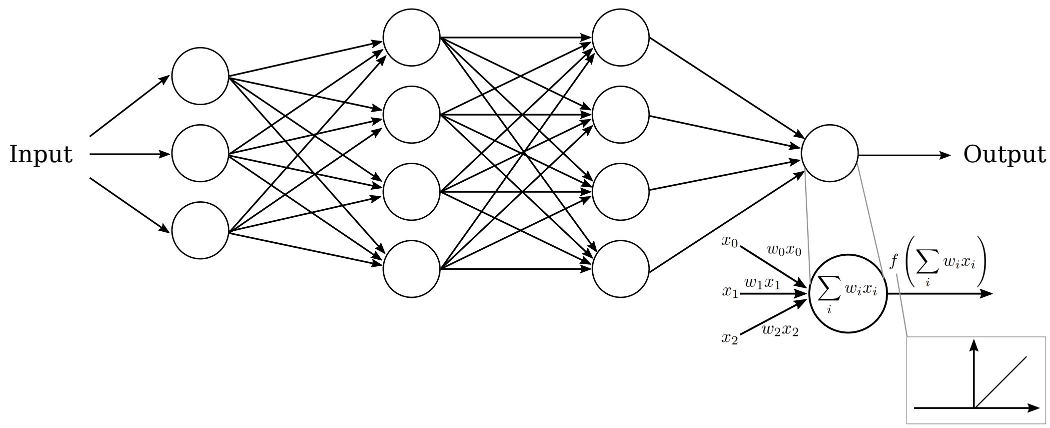


Figure 1.1: Artificial Neural Network
Inputs are passed through layers of neurons and connections. Neurons compute simple nonlinear functions of their summed input while connections weight signals passing through them.

instructions through the computer and numbers are represented in binary. The 0s and 1s of binary correspond to the power supply voltage and ground of digital logic (Fig. 1.2). Further, these CPUs instantiate part of the broader Von Neumann computer architecture, which separates state (the memory) from computation (the CPU) and excel at sequential programs, or programs that require executing steps one at a time. Although Von Neumann (of the Von Neumann architecture) himself noted the differences between a computer's architecture and the brain's as well as the potential of a more brain-like architecture [44], there was little economic incentive to develop alternative computing architectures. The computing industry adopted the silicon semiconductor metal-oxide-semiconductor field-effect transistor (MOSFET) as their physical substrate, which permitted exponentially increasing transistor counts in a chip over time—dubbed Moore's Law. When computer architects could expect to double a designs' performance every two to three years by switching to the latest silicon manufacturing process, alternative designs remained out of consideration. As a result, after decades of Moore's law, the digital computer approach has become synonymous with computation; the information age is synonymous with the digital age.

However, physical devices have physical limits, so developing brain-like architectures makes more economic sense as the semiconductor industry takes longer and spends more to shrink transistors. Specifically, at the smallest transistor sizes, the cost per transistor no longer decreases with transistor size, so it is no longer economically viable to rely on device shrinkage for performance gains. In the vacuum after nearly 50 years of Moore's law, AlexNet succeeded by matching a brain-inspired ANN structure, the convolutional neural network (CNN), with non-CPU hardware, the graphical processing unit (GPU), that was well-suited to the network's architecture. An ANN operates by passing inputs through thousands of neurons and millions of parameters simultaneously, and moving all of that data around is a poor fit for CPUs which are bottlenecked by the separation between state

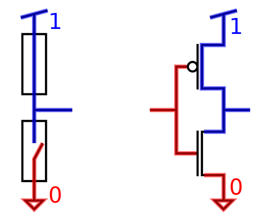


Figure 1.2: Digital Logic
LEFT: Digital computing relies on transistor networks (black boxes) to switch an output node between a connection to the power supply (top, blue), representing the binary value 1, and a connection to ground (bottom, red), representing the binary value 0. RIGHT: In this simple digital circuit, an inverter, the output is the inverted input.

and computation. A GPU better serves ANNs by matching the structure of the ANN dataflow—GPU inputs flow through many simple cores in parallel instead of one complicated core sequentially. AlexNet's success on a GPU convincingly demonstrating the utility of brain-inspired computational approaches and hardware.

Though people now recognize the value of highly-parallel, brain-like computing, there remain significant differences between the brain's architecture and today's hardware. To realize an ANN's thresholding, weighting, and summing operations efficiently, computer architects have developed application-specific integrated circuits (ASICs) [21], but these ASICs still rely on the predominant, synchronous-digital paradigm. However, from Carver Mead's work in the 1980s and 90s [34], neuromorphic engineers recognize that the physics underlying transistors and the physics underlying ion channels in a biological neurons membrane are qualitatively matched. Specifically, electrons or holes passing through a subthreshold MOSFET and ions passing through a neuron's membrane are both diffusion driven phenomena [33]. As a result, both are voltage controlled current sources with an exponential dependence of current on voltage (Fig. 1.3). That is, subthreshold MOSFETs are analogous to ion currents through a neuron's membrane.

With subthreshold MOSFETs as their building blocks, neuromorphic engineers push the brain analogy deeper than network-level structures and build silicon neurons that follow the inherent dynamics of biological neurons—spiking dynamics. In contrast to simple and static artificial neurons, biological neurons are continuous dynamical systems governed by electrochemical gradients and voltage-sensitive ion-channel proteins embedded in their membranes. Their most prominent behavior is the generation and emission of stereotyped, traveling depolarizations (spikes) along their projections (axons) to other neurons.

Neuromorphic engineers construct large scale spiking neural networks (SNNs) to leverage the minimal power consumption of subthreshold MOSFETs for useful computation and to understand the value of dynamics and spiking for brain-like computation with a forward engineering approach. While the predominant ANN approach running on synchronous-digital hardware has led to impressive achievements in image classification, speech classification, game playing, driving autonomy,

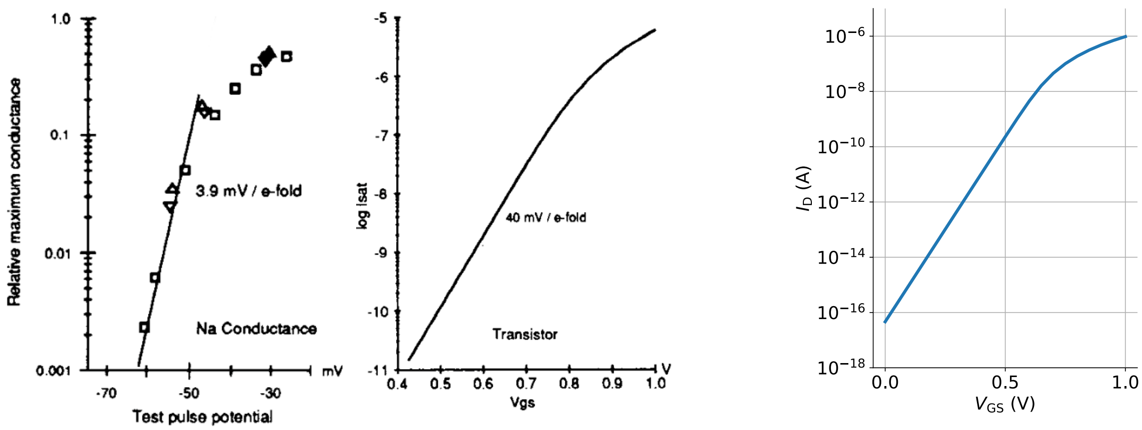


Figure 1.3: Ion Channels and Subthreshold Transistors
LEFT: Adapted from [34]. Comparison between sodium channel conductance in a neuron for a given membrane potential and a transistor's source to drain current for a given gate voltage show similar exponential relationships between voltage and current. RIGHT: the gate voltage to drain current relationship in a more modern, 28nm process transistor.

and many other fields, practitioners are still working to push such AI systems into highly power-constrained environments (and hence limited connectivity) that require real-time operation. The femtowatt power for subthreshold MOSFETs provide an attractive means for computing on minimal power budgets (Fig. 1.3, RIGHT), and by systematically building SNNs from low-power spiking dynamical circuits, neuromorphic engineers also address the critique [20] leveled at Neuroscience that reverse engineering and modeling techniques have yet to yield a mechanistic understanding of the brain. Such a mechanistic understanding is inherent to the systematic construction of SNNs practiced by neuromorphic engineers.

Prior to my joining the field, neuromorphic engineers developed efficient and scalable neuromorphic systems as culminated in Neurogrid with a million neurons and a billion synapses [2]. However, a million neurons and billion synapses are difficult to use without systematic means of mapping higher level computations onto the spiking neurons. By combining Neurogrid with an existing means, the Neural Engineering Framework (NEF) [15], we demonstrated the NEF principles on Neurogrid [12] and demoed a robot arm controller with Neurogrid's spiking silicon neurons [35]. From the lessons of marrying Neurogrid and NEF, we began our next project to build Braindrop, a chip architected specifically for large scale SNNs as systematically arranged with NEF. In this thesis, I present two contributions to the Braindrop chip that push forward our understanding of spiking communication and computation in neuromorphic systems. Chapter 2 describes the statistical consequences of spiking communication and computation in first-order dynamical systems, and Chapter 3 describes a router network for communicating spikes and programming packets.

2

Summing and Weighting Spike Trains

Similar to ANNs like AlexNet, SNNs rely on numerous neurons (providing thresholding) and connections (providing weighting and summing) (Fig. 2.1, ARCHITECTURE & WEIGHT MATRIX). Whereas these synchronous-digital designs operate at high-precision (scales exponentially with the number of bits) thousands of times faster than real-time (processing image or speech data in large batches), the hybrid analog-digital brain operates at vastly lower-precision (1-3 bits per spike [9]) in real-time. Similarly, ANN algorithms perform well at low precisions [13, 19] and real-time operation is appropriate for applications at the edge (processing a single user's speech).

Neuromorphic engineers seek to realize efficient and robust hardware architectures and computational frameworks by emulating the brain's hybrid analog-digital approach, trading precision and speed for energy efficiency [2, 3, 15]. Specifically, spiking-analog somas provide thresholding while consuming femtoamps of current, and all-or-nothing spikes—a unary representation—permit weighting with a random number generator and a comparator as well as summing by merging with a multiplexer (Fig. 2.1, OPERATIONS). Although such probabilistic (p) thinning, is cheap to implement,[1] it may be just as costly in energy as a conventional, binary multiply-accumulate unit when controlling for precision [29] due to quadratic—as opposed to logarithmic—scaling.

In this chapter, we describe an alternative, deterministic (d) thinning method and demonstrate its advantages over probabilistic thinning by analyzing each method's statistical consequences. In Section 2.1, we describe the weighting and summing operations and analyze each operation's output statistics by treating spikes as points in a temporal point-process. In Section 2.2, we analyze the

[1] A spike train with rate λ is thinned by dropping a fraction $(1 - w)$ of the spikes to produce a spike train with rate $w\lambda$, thus weighting the spike train by w. Although this method applies to positive signals, it is readily extended to negative signals by permitting spikes to be signed. Zero is still implicitly represented by an absence of spikes (i.e., zero represents itself).

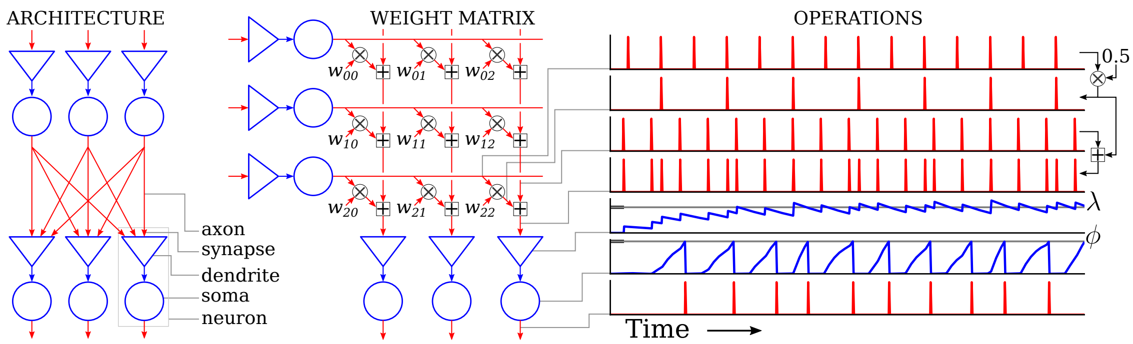


Figure 2.1: Spiking Neural Network
ARCHITECTURE: Spikes flow through axons and synapses (red) while currents flow through dendrites and somas (blue). WEIGHT MATRIX: Spike trains are weighted ($\otimes$) and summed ($\boxplus$) as they travel from one layer to the next. OPERATIONS: Weighting is by thinning and summing is by merging. Conversion from spike trains to currents is by low-pass filtering in synapses, whose currents settle around the input spike-rate λ. Conversion from currents back to spikes is by integration and thresholding in somas, whose spiking threshold is ϕ.

statistical effects of thinning spike-trains on synaptic signals and show that the two methods produce different SNR scaling. In Section 2.3, we analyze and optimize the power requirements for the two methods. In Section 2.4, we discuss our results and conclude the chapter.

2.1 Summing and Weighting

An accumulator implements summing and weighting in an ANN by combining merging with d-thinning. Merging produces a more Poissonian spike-train, while d-thinning produces a more periodic spike-train. It is crucial that merging occur before thinning. That is, instead of merging the outputs of N accumulators together, we merge N spike trains and feed the resulting spike train into a single accumulator (unlike in Figure 2.1, WEIGHT MATRIX). This ordering preserves the accumulators' more periodic-like spike-train statistics.

Summing by merging (Fig. 2.1, MERGING) produces more Poissonian spike-trains (Fig. 2.1, PERIODIC & POISSON INPUT). That is, the interspike-intervals (ISIs) become independent and identically distributed (IID) exponential random variables (i.e. from a renewal process). Consider the case when all N spike-trains have the same rate $\frac{\lambda}{N}$. Pick one as a reference and superpose the remaining $N-1$ on it with random offsets T_j, drawn independently from a uniform distribution on $\{0, \frac{N}{\lambda}\}$ (i.e, within the reference spike-train's first ISI). The resulting, merged spike-train's first ISI is given by $\Delta T_1 = \min(T_1, T_2, \ldots, T_{N-1})$. Its cumulative distribution function (CDF) is

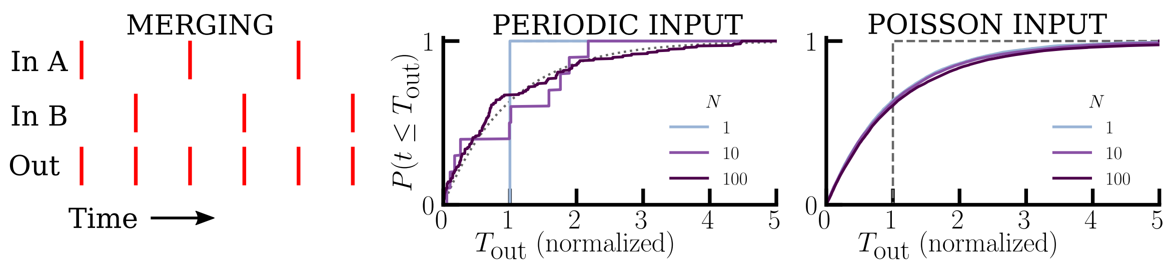


Figure 2.2: Summing by merging.
MERGING: Spike-trains are merged, to producing an output spike-train whose rate is the sum of the input rates. PERIODIC VS POISSON INPUT: As N, the number of (equal-rate) spike trains increases, the cumulative distribution functions (CDFs) of empirical output ISIs ($T_{\rm out}$), transition from degenerate (dashed line) to exponential (dotted line) versus remaining exponential. CDFs include 10,000 samples of $T_{\rm out}$, normalized by their mean.

$$
\begin{aligned}
P(\Delta T_1 \leq t) &= P(\min(T_1, T_2, \ldots, T_{N-1}) \leq t) \\
&= 1 - P(T_1 > t \wedge T_2 > t \wedge \ldots \vee T_{N-1} > t) \\
&= 1 - \int_t^{\frac{N}{\lambda}} \int_t^{\frac{N}{\lambda}} \ldots \int_t^{\frac{N}{\lambda}} \left(\frac{\lambda}{N}\right)^N dt_1 dt_2 \ldots dt_N \\
&= 1 - \left(\frac{\lambda}{N}\right)^N \left(\frac{N}{\lambda} - t\right)^N \\
&= 1 - \left(1 - \frac{\lambda t}{N}\right)^N \underset{N\to\infty}{=} 1 - e^{-\lambda t}
\end{aligned}
$$

recognizing that $\lim_{N\to\infty}\left(1 + \frac{x}{N}\right)^N = e^x$. This exponential distribution describes other ISIs in the merged spike-train as well since its first ISI is not special (i.e., its ISIs are identically distributed). It also arises when the spike rates are not equal, provided that consecutive spikes are never from the same neuron, which is satisfied if no neuron fires more than N times faster than the mean.

To see that ISIs are also independent for sufficiently large N, consider the merged, equal-rate periodic spike-trains again. The number of unique ISIs is exactly N; once all N unique ISIs have been traversed, they will repeat. However, this will take a longer and longer time as N tends to infinity. Once this time-scale exceeds several synaptic time constants, the ISIs can be effectively considered to be independent.[2]

p and d-thinning (Fig. 2.1, THINNING), produce more Poissonian and more periodic spike-trains for decreasing weight w, respectively (Fig. 2.1, P-THIN & D-THIN). That is, the resulting spike-train's

[2]It is crucial that the individual spike-train *rates* tend to 0 as the number of spike-trains tends to infinity in contrast to scaling the spike *amplitudes* (represented by delta functions). If only the spikes themselves are scaled, the superposed independent sources will not converge to a Poisson process—even in the limit [26]. As a consequence, spike times will synchronize more strongly through multiple, feed-forward layers of a neural network [14].

ISIs' (X) coefficient of variation ($\mathrm{CV}(X)$) approaches 1 and 0, respectively.[3] An ISI in a p-thinned spike-train, is given by $\Delta T_{\mathrm{out}} = \sum_j^S \Delta T_{\mathrm{in}_j}$, where ΔT_{in_j} are the IID pre-thinned ISIs, and S is the number of ISIs skipped before a spike makes it through. Since S is geometrically distributed with parameter w,

$$\begin{aligned}
\mathbf{E}\left[\Delta T_{\mathrm{out}}\right] &= \mathbf{E}\left[\Delta T_{\mathrm{in}}\right]/w \\
\mathrm{Var}(\Delta T_{\mathrm{out}}) &= (w\mathrm{Var}(\Delta T_{\mathrm{in}}) + \mathbf{E}[\Delta T_{\mathrm{in}}]^2(1-w))/w^2 \\
\mathrm{CV}_{\mathrm{pthin}}(\Delta T_{\mathrm{out}}) &= \sqrt{w\frac{\mathrm{Var}(\Delta T_{\mathrm{in}})}{\mathbf{E}[\Delta T_{\mathrm{in}}]^2} + (1-w)} = \sqrt{w\mathrm{CV}(\Delta T_{\mathrm{in}})^2 + (1-w)}
\end{aligned}$$

w interpolates between $\mathrm{CV}_{\mathrm{poi}}(\Delta T_{\mathrm{out}}) = 1$ and $\mathrm{CV}(\Delta T_{\mathrm{in}})$. As $w \to 0$, $\mathrm{CV}_{\mathrm{pthin}}(\Delta T_{\mathrm{out}}) \to 1$, while as $w \to 1$, $\mathrm{CV}_{\mathrm{pthin}}(\Delta T_{\mathrm{out}}) \to \mathrm{CV}(\Delta T_{\mathrm{in}})$ (i.e, all input spikes are output). For example, when a periodic spike-train ($\mathrm{CV}_{\mathrm{per}}(\Delta T_{\mathrm{out}}) = 1$) is p-thinned, $\mathrm{CV}_{\mathrm{pthin}}(\Delta T_{\mathrm{out}}) = \sqrt{1-w}$ (Table 2.1, p-thinned periodic), becoming more Poisson as $w \to 0$.

In the case of d-thinning, the spike train becomes more periodic with $\Delta T_{\mathrm{out}} = \sum_j^{1/w} \Delta T_{\mathrm{in}_j}$.[4] The sum now contains a deterministic number ($1/w$) of ΔT_{in},

$$\begin{aligned}
\mathbf{E}\left[\Delta T_{\mathrm{out}}\right] &= \mathbf{E}\left[\Delta T_{\mathrm{in}}\right]/w \\
\mathrm{Var}(\Delta T_{\mathrm{out}}) &= \mathrm{Var}(\Delta T_{\mathrm{in}})/w \\
\mathrm{CV}_{\mathrm{dthin}}(\Delta T_{\mathrm{out}}) &= \sqrt{w\frac{\mathrm{Var}(\Delta T_{\mathrm{in}})}{\mathbf{E}[\Delta T_{\mathrm{in}}]^2}} = \sqrt{w}CV(\Delta T_{\mathrm{in}})
\end{aligned}$$

Now w interpolates between $\mathrm{CV}_{\mathrm{per}}(\Delta T_{\mathrm{out}}) = 0$ and $\mathrm{CV}(\Delta T_{\mathrm{in}})$. As $w \to 0$, $\mathrm{CV}_{\mathrm{dthin}}(\Delta T_{\mathrm{out}}) \to 0$, while as $w \to 1$, $\mathrm{CV}_{\mathrm{dthin}}(\Delta T_{\mathrm{out}}) \to \mathrm{CV}(\Delta T_{\mathrm{in}})$ (as with $\mathrm{CV}_{\mathrm{pthin}}$). For example, when a Poisson spike-train ($\mathrm{CV}_{\mathrm{poi}}(\Delta T_{\mathrm{in}}) = 1$) is d-thinned, $\mathrm{CV}_{\mathrm{dthin}}(\Delta T_{\mathrm{out}}) = \sqrt{w}$, becoming more periodic as $w \to 0$ (Table 2.1, d-thinned Poisson).

2.2 Discrete to Continuous

Spiking neural networks encode signals in their spike rates, which requires estimating the underlying spike rate from the train of spikes with methods that respect causality and track temporal changes in the rate. The first requirement precludes estimating the instantaneous spike-rate as it requires

[3] $\mathrm{CV}(X) = \sqrt{\mathrm{Var}(X)}/\mathbf{E}[X]$, where $\mathbf{E}[X]$ is X's expectation (i.e, mean), and $\mathrm{Var}(X)$ is its variance. For ISIs in a periodic spike-train, $\mathrm{CV}_{\mathrm{per}}(X) = 0$, and for ISIs in a Poisson spike-train, $\mathrm{CV}_{\mathrm{poi}}(X) = 1$ (Table 2.1).

[4] When ΔT_{in} is exponentially distributed, ΔT_{out}, will be gamma—or more specifically Erlang—distributed as the sum of IID exponentials.

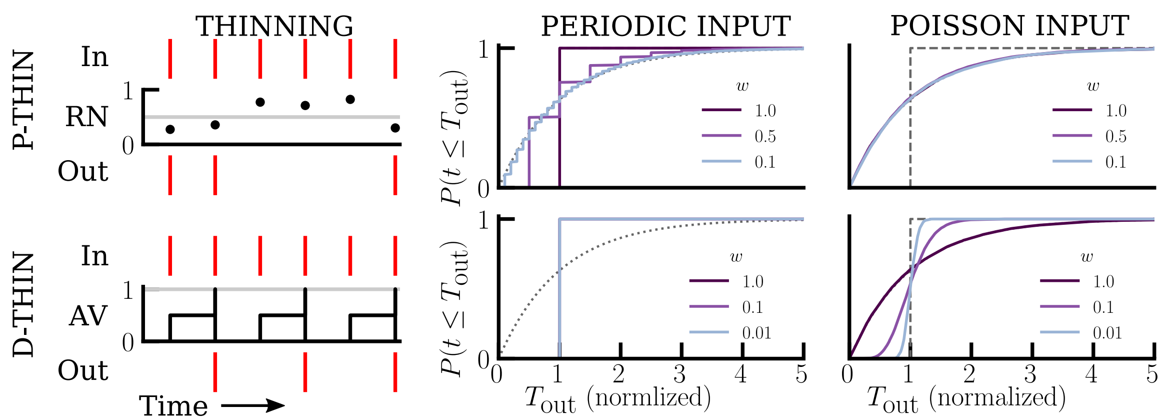


Figure 2.3: Probabilistic (P) vs Deterministic (D) Thinning.
P-THIN: Each input spike draws a random number (RN) from a uniform distribution in {0:1}. If RN is less than the weight (0.5), then the spike is forwarded out. D-THIN: Each input spike increments an accumulated value (AV) by the weight (0.5). If AV exceeds a threshold (1), then the spike is forwarded out and AV is decremented by the threshold. P-THIN PERIODIC vs D-THIN POISSON: As the weight decreases, cumulative distribution functions (CDFs) of empirical output ISIs (T_{out}), sampled and plotted as in Fig. 2.1, transition from degenerate (dashed lines) to exponential (dotted lines) versus exponential to degenerate. P-THIN POISSON and D-THIN PERIODIC: CDFs do not change with the weight.

Table 2.1: Renewal Process ISI CV and Synapse SNR

Renewal Process	ISI CV	Synapse SNR
Poisson	1	$\sqrt{2\lambda\tau}$
periodic	0	$\sqrt{\frac{2\lambda\tau}{\coth\left(\frac{1}{2\lambda\tau}\right)-2\lambda\tau}}$
p-thinned periodic	$\sqrt{1-w}$	$\sqrt{\frac{2\lambda\tau}{1-p+p\coth\left(\frac{p}{2\lambda\tau}\right)-2\lambda\tau}}$
d-thinned Poisson	$\sqrt{w}$	$\sqrt{\frac{2\lambda\tau}{\frac{(1+k\lambda\tau)^k+(k\lambda\tau)^k}{(1+k\lambda\tau)^k-(k\lambda\tau)^k}-2\lambda\tau}}$

knowing the next future spike-time. The second requirement limits the number of ISI samples available to the time window times the spike-rate.

We consider methods that obtain an estimate, X, of the rate by computing a running average of spikes, weighting a spike that happened T_j seconds in the past by $h(T_j)$. That is,

$$X = \sum_{j=0}^{\infty} h(T_j) \tag{2.1}$$

defined at a point in time after an infinite number of spikes from a fixed-rate spike-train have arrived, which allows us to ignore the initial transient and focus on the steady-state mean. We quantify the estimate's quality using its signal-to-noise ratio:

$$\mathrm{SNR}(X) = \mathbf{E}[X]/\sqrt{\mathrm{Var}(X)} \tag{2.2}$$

We consider the simplest implementation of $h(T_j)$: a first-order, low-pass filter (LPF) driven by a train of impulses, $s(t) = \sum_j \delta(t - t_j)$, where t_j is the jth spike's time. That is,

$$\tau \frac{dx}{dt} = -x + s \tag{2.3}$$

where $x(t)$ is the LPF's state (and output) and τ is its time-constant. Upon receiving the jth spike, $\delta(t - t_j)$, $x(t)$ jumps by $1/\tau$ and then decays at the rate $1/\tau$—only $x(t)$'s *mean* value settles to the spike rate—so larger τ result in cleaner but slower estimates. In this case, $h(t) = \frac{1}{\tau} e^{-t/\tau} u(t)$, which yields weights that decay by a factor of e every τ seconds; $u(t)$ is the unit-step function. In addition to depending on τ, the rate-estimate's variability also depends on the incoming ISIs' variability (Fig. 2.2).

We derive $\mathrm{SNR}(X)$ for low-pass filtered Poisson, periodic, p-thinned periodic, and d-thinned Poisson spike-trains (Table 2.1 & Fig. 2.2). We do not consider p-thinned Poisson nor d-thinned periodic spike-trains because their statistics do not change with the weight (see Fig. 2.1). It suffices to find the first and second moments of X for each process since $\mathrm{Var}(X) = \mathbf{E}[X^2] - \mathbf{E}[X]^2$. To express T_j in terms of IID random variables, we define $\Delta T_j = T_j - T_{j-1}$ for $j > 0$ so that $T_j = \sum_{n=0}^{j} \Delta T_n$. Note that $\Delta T_0 = T_0$, the elapsed time from the most recent spike, is not an ISI and so is not generally drawn from the same distribution as the ISIs. Substituting into (2.1) yields

$$
\begin{aligned}
X &= \sum_{j=0}^{\infty} h\left(\sum_{l=0}^{j} \Delta T_l\right) \\
&= \frac{1}{\tau} e^{\frac{-\Delta T_0}{\tau}} + \frac{1}{\tau} e^{-\frac{\Delta T_0 + \Delta T_1}{\tau}} + \frac{1}{\tau} e^{-\frac{\Delta T_0 + \Delta T_1 + \Delta T_2}{\tau}} + \ldots \\
&= \frac{1}{\tau} \sum_{j=0}^{\infty} \prod_{l=0}^{j} e^{-\frac{\Delta T_l}{\tau}}
\end{aligned} \tag{2.4}
$$

Therefore,

$$
\begin{aligned}
X^2 &= \left(\frac{1}{\tau} \sum_{j=0}^{\infty} \prod_{l=0}^{j} e^{-\frac{\Delta T_l}{\tau}}\right)^2 \\
&= \frac{1}{\tau^2} \sum_{j=0}^{\infty} \prod_{l=0}^{j} e^{\frac{-2\Delta T_l}{\tau}} + \frac{2}{\tau^2} \sum_{j=0}^{\infty} \sum_{l=j+1}^{\infty} \prod_{m=0}^{j} e^{\frac{-2\Delta T_m}{\tau}} \prod_{n=j+1}^{l} e^{\frac{-\Delta T_n}{\tau}} \qquad \text{breaking into diagonal and cross terms,} \\
&= \frac{1}{\tau^2} \sum_{j=0}^{\infty} \prod_{l=0}^{j} e^{\frac{-2\Delta T_l}{\tau}} + \frac{2}{\tau^2} \sum_{j=0}^{\infty} \prod_{m=0}^{j} e^{\frac{-2\Delta T_m}{\tau}} \sum_{l=j+1}^{\infty} \prod_{n=j+1}^{l} e^{\frac{-\Delta T_n}{\tau}}
\end{aligned}
$$

Assuming ΔT_0 is identically distributed to the ISIs (i.e., that the ISIs are memoryless as in the Poisson process), from (2.4),

$$
\begin{aligned}
\mathbf{E}[X] &= \mathbf{E}\left[\frac{1}{\tau} \sum_{j=0}^{\infty} \prod_{l=0}^{j} e^{\frac{-\Delta T_l}{\tau}}\right] \\
&= \frac{1}{\tau} \sum_{j=0}^{\infty} \mathbf{E}\left[\prod_{l=0}^{j} e^{-\frac{\Delta T_l}{\tau}}\right] \\
&= \frac{1}{\tau} \sum_{j=0}^{\infty} \mathbf{E}\left[e^{\frac{-\Delta T}{\tau}}\right]^{j+1} \qquad \Delta T_j \text{ are IID} \\
&= \frac{1}{\tau} \sum_{j=1}^{\infty} \varphi_{\Delta T}\left(\frac{i}{\tau}\right)^j
\end{aligned} \tag{2.5}
$$

noting that $\mathbf{E}\left[e^{-\Delta T/\tau}\right] = \varphi_{\Delta T}(s)|_{s=i/\tau}$, the characteristic function of ΔT evaluated at i/τ.[5] From (2.5),

[5]The characteristic function is related to the Fourier transform: $\varphi_X(s) = \mathcal{F}_{f_X}(\frac{-s}{2\pi})$ since $\varphi_X(s) = \int e^{isx} f_X(x)dx$ for random variable X with probability density function $f_X(x)$ and $\mathcal{F}_g(\omega) = \int e^{-2\pi i x \omega} g(x)dx$ for function $g(x)$.

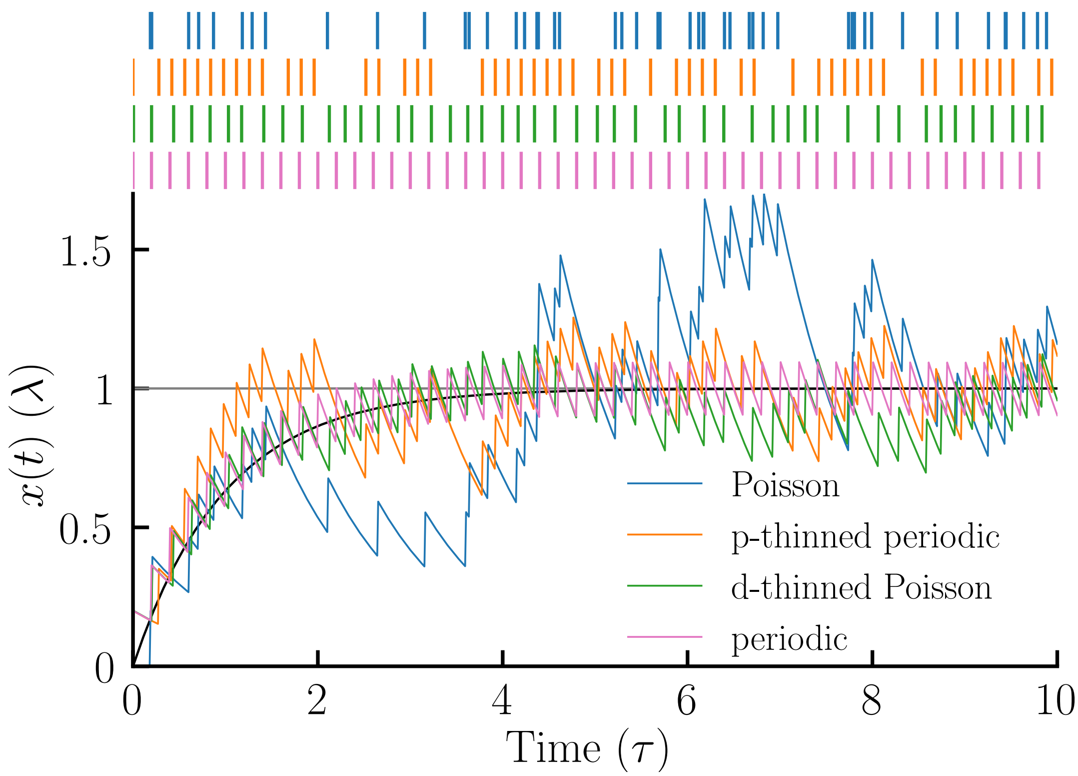


Figure 2.4: Renewal Process-Driven Synapse

Top: Spikes generated by Poisson, p-thinned periodic ($p = 0.7$), d-thinned Poisson ($k = 20$), and periodic point-processes. The underlying spike rates are matched at $\lambda = 5$ spikes per τ. Bottom: Synaptic response to spikes is governed by first-order dynamics (2.3). The mean output converges to λ, and, in this example, the variability decreases as we go from Poisson to periodic (ordered as above).

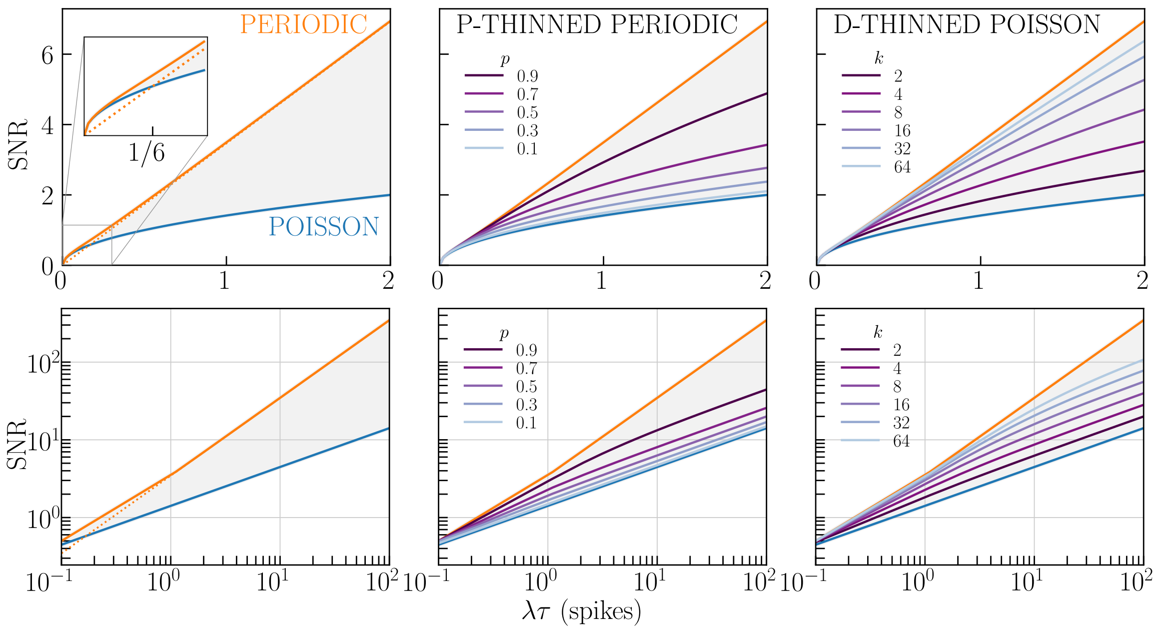


Figure 2.5: Rate-Estimate SNR.

Periodic & poisson: SNR scales linearly and as the square-root, respectively, with $\lambda\tau$, the number of spikes per synaptic time-constant τ, for $\lambda\tau > 1/6$. Below this number (inset), periodic switches from linear (dotted line) to square-root. p-thinned periodic: SNR transitions from periodic to Poisson behavior (shaded region) as p decreases. d-thinned poisson: SNR transitions from Poisson to periodic behavior as k increases.

$$
\begin{aligned}
&\mathbf{E}[X^2] \\
&\quad = \mathbf{E}\left[\frac{1}{\tau^2} \sum_{j=0}^{\infty} \prod_{l=0}^{j} e^{\frac{-2\Delta T_l}{\tau}} + \frac{2}{\tau^2} \sum_{j=0}^{\infty} \prod_{m=0}^{j} e^{\frac{-2\Delta T_m}{\tau}} \sum_{l=j+1}^{\infty} \prod_{n=j+1}^{l} e^{\frac{-\Delta T_n}{\tau}} \right] \\
&\quad = \frac{1}{\tau^2} \sum_{j=1}^{\infty} \mathbf{E}\Big[e^{\frac{-2\Delta T}{\tau}} \Big]^j + \frac{2}{\tau^2} \sum_{j=1}^{\infty} \mathbf{E}\Big[e^{\frac{-2\Delta T}{\tau}} \Big]^j \sum_{l=j+1}^{\infty} \mathbf{E}\Big[e^{\frac{-\Delta T}{\tau}} \Big]^{l-j}
\end{aligned}
$$

substituting $m = l - j$

$$
\begin{aligned}
&\quad = \frac{1}{\tau^2} \sum_{j=1}^{\infty} \mathbf{E}\Big[e^{\frac{-2\Delta T}{\tau}} \Big]^j + \frac{2}{\tau^2} \sum_{j=1}^{\infty} \mathbf{E}\Big[e^{\frac{-2\Delta T}{\tau}} \Big]^j \sum_{m=1}^{\infty} \mathbf{E}\Big[e^{\frac{-\Delta T}{\tau}} \Big]^{m} \\
&\quad = \frac{1}{\tau^2} \sum_{j=1}^{\infty} \varphi_{\Delta T}\left(\frac{2i}{\tau}\right)^j \left(1 + 2 \sum_{m=1}^{\infty} \varphi_{\Delta T}\left(\frac{i}{\tau}\right)^m \right) \qquad (2.6)
\end{aligned}
$$

When ΔT_0 is not identically distributed to the ISIs (i.e., the ISI distribution has memory), we split out ΔT_0 from (2.4) and (2.5),

$$
X = \frac{1}{\tau} \sum_{j=0}^{\infty} \prod_{l=0}^{j} e^{\frac{-\Delta T_l}{\tau}} = \frac{1}{\tau} e^{\frac{-\Delta T_0}{\tau}} \left(1 + \sum_{j=1}^{\infty} \prod_{l=1}^{j} e^{\frac{-\Delta T_l}{\tau}} \right)
$$

$$
X^2 = \frac{1}{\tau^2} e^{\frac{-2\Delta T_0}{\tau}} \left(1 + 2 \sum_{j=1}^{\infty} \prod_{l=1}^{j} e^{\frac{-\Delta T_l}{\tau}} + \sum_{j=1}^{\infty} \prod_{l=1}^{j} e^{\frac{-2\Delta T_l}{\tau}} + 2 \sum_{j=1}^{\infty} \prod_{m=1}^{j} e^{\frac{-2\Delta T_m}{\tau}} \sum_{l=j+1}^{\infty} \prod_{n=j+1}^{l} e^{\frac{-\Delta T_n}{\tau}} \right)
$$

which results in,

$$\begin{aligned}
\mathbf{E}[X] &= \frac{1}{\tau}\mathbf{E}\left[e^{\frac{-\Delta T_0}{\tau}}\right]\left(1+\sum_{j=1}^{\infty}\mathbf{E}\left[e^{\frac{-\Delta T}{\tau}}\right]^j\right) \\
&= \frac{1}{\tau}\varphi_{\Delta T_0}\left(\frac{i}{\tau}\right)\left(1+\sum_{j=1}^{\infty}\varphi_{\Delta T}\left(\frac{i}{\tau}\right)^j\right) \qquad (2.7)\\
\mathbf{E}[X^2] &= \frac{1}{\tau^2}\mathbf{E}\left[e^{\frac{-2\Delta T_0}{\tau}}\right]\left(1+2\sum_{j=1}^{\infty}\mathbf{E}\left[e^{\frac{-\Delta T}{\tau}}\right]^j\right. \\
&\quad \left.+\sum_{j=1}^{\infty}\mathbf{E}\left[e^{\frac{-2\Delta T}{\tau}}\right]^j+2\sum_{j=1}^{\infty}\mathbf{E}\left[e^{\frac{-2\Delta T}{\tau}}\right]^j\sum_{l=1}^{\infty}\mathbf{E}\left[e^{\frac{-\Delta T}{\tau}}\right]^l\right) \\
&= \frac{1}{\tau^2}\varphi_{\Delta T_0}\left(\frac{2i}{\tau}\right)\left(1+2\sum_{j=1}^{\infty}\varphi_{\Delta T}\left(\frac{i}{\tau}\right)^j\right. \\
&\quad \left.+\sum_{j=1}^{\infty}\varphi_{\Delta T}\left(\frac{2i}{\tau}\right)^j+2\sum_{j=1}^{\infty}\varphi_{\Delta T}\left(\frac{2i}{\tau}\right)^j\sum_{l=1}^{\infty}\varphi_{\Delta T}\left(\frac{i}{\tau}\right)^l\right) \qquad (2.8)
\end{aligned}$$

2.2.1 Poisson Process

With Poisson input, SNR scales with the square-root of the number of spikes per synaptic time-constant. For an exponential distribution with rate parameter λ, $\varphi_{\Delta T}(s) = \frac{\lambda}{\lambda - is}$. Using (2.2) and $s = i/\tau$,

$$\mathbf{E}[X] = \frac{1}{\tau}\sum_{j=1}^{\infty}\left(\frac{\lambda\tau}{\lambda\tau+1}\right)^j = \frac{1}{\tau}\frac{\frac{\lambda\tau}{\lambda\tau+1}}{1-\frac{\lambda\tau}{\lambda\tau+1}} = \lambda$$

That is, the expected rate-estimate matches the input rate. Using (2.6) and $s = 2i/\tau$,

$$\begin{aligned}
\mathbf{E}[X^2] &= \frac{1}{\tau^2}\sum_{j=1}^{\infty}\left(\frac{\lambda\tau}{\lambda\tau+2}\right)^j\left(1+2\sum_{m=1}^{\infty}\left(\frac{\lambda\tau}{\lambda\tau+1}\right)^m\right) \\
&= \frac{1}{\tau^2}\frac{\lambda\tau}{2}(1+2\lambda\tau) = \frac{\lambda}{2\tau}+\lambda^2
\end{aligned}$$

Therefore, from (2.2),

$$\begin{aligned}
\mathrm{Var}(X) &= \lambda/2\tau \\
\mathrm{SNR}_{\mathrm{poi}}(X) &= \sqrt{2\lambda\tau}
\end{aligned}$$

(Table 2.1, Poisson & Fig. 2.2, POISSON), which we confirm with numerical simulations (Fig. 2.2.1,

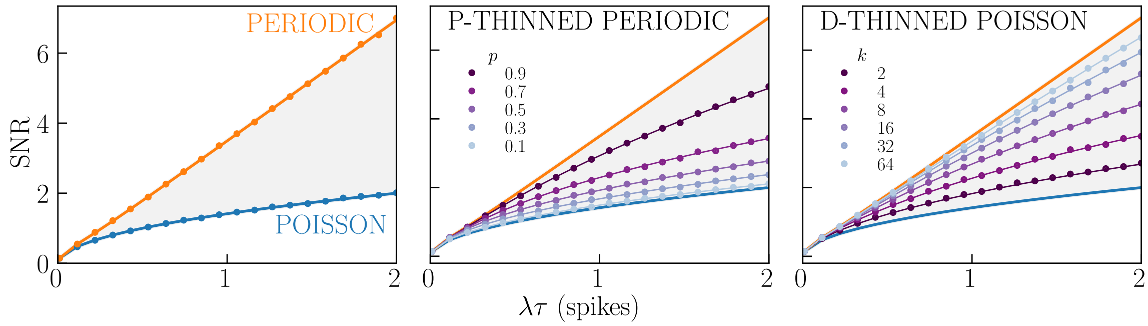


Figure 2.6: Theory vs numerical simulations.
Empirical SNR measured and averaged over 10,000 spike-train realizations with $30\lambda\tau$ spikes each. The results (circles) match the corresponding theoretical SNR (solid lines).

POISSON). Note that the relevant quantity is $\lambda\tau$, the number of spikes arriving within a time constant, not the spike rate λ or the synaptic time constant τ individually.

2.2.2 Periodic Process

With periodic input, SNR scales linearly with $\lambda\tau$. While the ISIs are equal, ΔT_0 is uniformly distributed between 0 and $1/\lambda$, so that $\varphi_{\Delta T_0}(s) = \frac{e^{is/\lambda}-1}{is/\lambda}$. From T_0 into the past, every subsequent spike is $1/\lambda$ further from the present, so from (2.7) and (2.8),

$$\mathbf{E}[X] = \frac{1}{\tau}\frac{e^{\frac{-1}{\lambda\tau}}-1}{-1/\lambda\tau}\left(1+\sum_{j=1}^{\infty} e^{\frac{-j}{\lambda\tau}}\right) = \lambda$$

$$\begin{aligned}\mathbf{E}[X^2] &= \frac{1}{\tau^2}\frac{e^{\frac{-2}{\lambda\tau}}-1}{-2/\lambda\tau}\left(1+2\sum_{j=1}^{\infty} e^{\frac{-j}{\lambda\tau}}+\sum_{j=1}^{\infty} e^{\frac{-2j}{\lambda\tau}}+2\sum_{j=1}^{\infty} e^{\frac{-2j}{\lambda\tau}}\sum_{l=1}^{\infty} e^{\frac{-l}{\lambda\tau}}\right)\\ &= \frac{\lambda}{2\tau}\frac{1+e^{\frac{-1}{\lambda\tau}}}{1-e^{\frac{-1}{\lambda\tau}}} = \frac{\lambda}{2\tau}\coth\left(\frac{1}{2\lambda\tau}\right)\end{aligned}$$

Therefore, from (2.2),

$$\begin{aligned}\mathrm{Var}(X) &= \frac{\lambda}{2\tau}\coth\left(\frac{1}{2\lambda\tau}\right)-\lambda^2 = \frac{\lambda^2}{2\lambda\tau}\left(\coth\left(\frac{1}{2\lambda\tau}\right)-2\lambda\tau\right)\\ \mathrm{SNR}_{\mathrm{per}}(X) &= \sqrt{\frac{2\lambda\tau}{\coth\left(\frac{1}{2\lambda\tau}\right)-2\lambda\tau}}\end{aligned}$$

(Table 2.1, periodic & Fig. 2.2, PERIODIC), which we confirm with numerical simulations (Fig. 2.2.1, PERIODIC).

When $\lambda\tau$ is small and large, $\mathrm{SNR}_{\mathrm{per}}(X)$ scales as the square-root and linearly, respectively. As $\lambda\tau$ approaches zero, $\mathrm{SNR}_{\mathrm{per}}(X)$ scales similarly to Poisson SNR ($\sqrt{\lambda\tau}$):

$$\lim_{\lambda\tau\to 0}\mathrm{SNR}_{\mathrm{per}}(X) = \lim_{\lambda\tau\to 0}\sqrt{\frac{2\lambda\tau}{\coth\left(\frac{1}{2\lambda\tau}\right) - 2\lambda\tau}} = \sqrt{2\lambda\tau}$$

(i.e., periodic and Poisson inputs are indistinguishable from an SNR perspective). For high $\lambda\tau$, the limit is not as straightforward—$\lim_{\lambda\tau\to\infty}\coth(1/2\lambda\tau)$ is not defined—but we find that $\mathrm{SNR}_{\mathrm{per}} \to 2\sqrt{3}\lambda\tau$ (see Appendix A.1). That is, SNR scales linearly with $\lambda\tau$ at high $\lambda\tau$. These two approximations intersect at $\lambda\tau = 1/6$ (Fig. 2.2, PERIODIC, inset). The low $\lambda\tau$ approximation is within 10% and 1% of the actual SNR for $\lambda\tau < 0.0950$ and $\lambda\tau < 0.00995$, respectively. The high $\lambda\tau$ approximation is within 10% and 1% of the actual SNR for $\lambda\tau > 0.253$ and $\lambda\tau > 0.902$, respectively. In summary,

$$\mathrm{SNR}_{\mathrm{per}}(\lambda\tau) \approx \begin{cases} \sqrt{2\lambda\tau} & \lambda\tau \leq 1/6 \\ 2\sqrt{3}\lambda\tau & \lambda\tau > 1/6 \end{cases}$$

2.2.3 p-Thinned Periodic Process

With p-thinned periodic input, SNR transitions from periodic to Poisson with decreasing p ($= w$). Instead of using (2.7) and (2.8) directly, we modify X by introducing a random (indicator) variable

$$I_j = \begin{cases} 1 & \text{with probability } p \\ 0 & \text{with probability } 1-p \end{cases}$$

to indicate whether the jth pre-thinned spike is kept or not:

$$X = \frac{1}{\tau}e^{-\frac{\Delta T_0}{\tau}}\left(I_0 + \sum_{j=1}^{\infty} e^{-\frac{j}{\lambda_p\tau}} I_j\right)$$

where ΔT_0 is the time to the first, pre-thinned spike and λ_p is the rate of the pre-thinned spike-train. Only spikes whose corresponding indicator is 1 will affect the current state. Therefore,

$$\begin{aligned}
\mathbf{E}[X] &= \mathbf{E}\left[\frac{1}{\tau}e^{\frac{-\Delta T_0}{\tau}}\left(I_0+\sum_{j=1}^{\infty}e^{\frac{-j}{\lambda_p\tau}}I_j\right)\right] \\
&= \frac{1}{\tau}\mathbf{E}\left[e^{\frac{-\Delta T_0}{\tau}}\right]\left(\mathbf{E}[I_0]+\sum_{j=1}^{\infty}e^{\frac{-j}{\lambda_p\tau}}\mathbf{E}[I_j]\right) && (T_0,\ I_j \text{ are independent}) \\
&= \frac{1}{\tau}\frac{1-e^{\frac{-1}{\lambda_p\tau}}}{1/\lambda_p\tau}\left(p+\sum_{j=0}^{\infty}e^{\frac{-j}{\lambda_p\tau}}p\right) = p\lambda_p = \lambda
\end{aligned}$$

For $\mathbf{E}[X^2]$,

$$\begin{aligned}
\mathbf{E}[X^2] &= \mathbf{E}\left[\left(\frac{1}{\tau}e^{\frac{-\Delta T_0}{\tau}}\left(I_0+\sum_{j=1}^{\infty}e^{\frac{-j}{\lambda_p\tau}}I_j\right)\right)^2\right] \\
&= \frac{\lambda_p}{2\tau}\left(1-e^{-2/\lambda_p\tau}\right)\mathbf{E}\left[\left(\sum_{j=0}^{\infty}e^{\frac{-j}{\lambda_p\tau}}I_j\right)^2\right]
\end{aligned}$$

The expectation can be broken into diagonal and cross terms.

$$\begin{aligned}
\mathbf{E}\left[\left(\sum_{j=0}^{\infty}e^{\frac{-j}{\lambda_p\tau}}I_j\right)^2\right] &= \mathbf{E}\left[\sum_{j=0}^{\infty}e^{\frac{-2j}{\lambda_p\tau}}I_j^2+\sum_{j=0,j\neq l}^{\infty}\sum_{l=0}^{\infty}e^{\frac{-(j+l)}{\lambda_p\tau}}I_jI_l\right] \\
&= \sum_{j=0}^{\infty}e^{\frac{-2j}{\lambda_p\tau}}\mathbf{E}\left[I_j^2\right]+\sum_{j=0,j\neq l}^{\infty}\sum_{l=0}^{\infty}e^{\frac{-(j+l)}{\lambda_p\tau}}\mathbf{E}[I_jI_l] \\
&= \sum_{j=0}^{\infty}e^{\frac{-jk}{\lambda_p\tau}}p+\sum_{j=0,j\neq l}^{\infty}\sum_{l=0}^{\infty}e^{\frac{-(j+l)}{\lambda_p\tau}}p^2 && (I_j,\ I_l \text{ independent for } j\neq l) \\
&= \frac{p}{1-e^{\frac{-2}{\lambda_p\tau}}}+\frac{p^2}{\left(1-e^{\frac{-1}{\lambda_p\tau}}\right)^2}-\frac{p^2}{1-e^{\frac{-2}{\lambda_p\tau}}}
\end{aligned}$$

After substitution back into $\mathbf{E}[X^2]$,

$$\mathbf{E}[X^2] = \frac{\lambda_p p}{2\tau}\left(1 - p + p\frac{1+e^{-1/\lambda_p\tau}}{1-e^{-1/\lambda_p\tau}}\right) = \frac{\lambda}{2\tau}\left(1 - p + p\coth\left(\frac{p}{2\lambda\tau}\right)\right)$$

Therefore,

$$\begin{aligned}\mathrm{Var}(X) &= \frac{\lambda}{2\tau}\left(1 - p + p\coth\left(\frac{p}{2\lambda\tau}\right) - 2\lambda\tau\right)\\ \mathrm{SNR}_{\mathrm{pthin}}(X) &= \sqrt{\frac{2\lambda\tau}{1 - p + p\coth\left(\frac{p}{2\lambda\tau}\right) - 2\lambda\tau}}\end{aligned}$$

(Table 2.1, p-thinned periodic & Fig. 2.2, P-THIN.), which we confirm with numerical simulations (Fig. 2.2.1, P-THIN.). As $p \to 1$, $\mathrm{SNR}_{\mathrm{pthin}} \to \mathrm{SNR}_{\mathrm{per}}$ (i.e., no spikes are dropped), and as $p \to 0$, $\mathrm{SNR}_{\mathrm{pthin}} \to \mathrm{SNR}_{\mathrm{poi}}$ (see Appendix A.2). Therefore,

$$\mathrm{SNR}_{\mathrm{pthin}}(X) \approx \begin{cases}\mathrm{SNR}_{\mathrm{poi}}(X) & \text{at low } p\\ \mathrm{SNR}_{\mathrm{per}}(X) & \text{at high } p\end{cases}$$

2.2.4 d-Thinned Poisson Process

With d-thinned Poisson input, SNR transitions from Poisson to periodic behavior with increasing k $(= 1/w)$. The d-thinned Poisson process is not memoryless, necessitating the use of (2.7) and (2.8) and considering ΔT_0 and ΔT separately. Expressing $\Delta T_0 = U\Delta T^*$, where U selects uniformly within random interval ΔT^* (i.e., $U \sim \mathrm{Uniform}(0,1)$), which is not identically distributed to ΔT, an example of the *inspection paradox* [40]. Although we select a point in time without bias, our point in time is biased towards falling within larger ISIs simply because they are larger. Accounting for this bias, ΔT^*'s probability density function (PDF) is $f_{\Delta T^*}(t) = \frac{t f_{\Delta T}(t)}{\mathrm{E}[\Delta T]}$. Scaling density function $f_{\Delta T}(t)$ by t captures this bias and dividing by $\mathbf{E}[\Delta T]$ renormalizes the scaled density to a valid probability density (i.e. $\int f_{\Delta T^*} = \int t f_{\Delta T}(t)dt = \mathbf{E}[\Delta T]$). As a result,

$$\begin{aligned}\mathbf{E}\left[e^{\frac{-\Delta T_0}{\tau}}\right] &= \lambda\tau\left(1 - \left(\frac{k\lambda\tau}{1+k\lambda\tau}\right)^k\right)\\ \mathbf{E}\left[e^{\frac{-2\Delta T_0}{\tau}}\right] &= \frac{\lambda\tau}{2}\left(1 - \left(\frac{k\lambda\tau}{2+k\lambda\tau}\right)^k\right)\end{aligned}$$

and since $\varphi_{\Delta T}(s) = \left(\frac{k\lambda}{k\lambda - is}\right)^{k\lambda}$, (2.7) and (2.8) produce

$$\begin{aligned}
\mathbf{E}[X] &= \frac{1}{\tau}\lambda\tau\left(1-\left(\frac{k\lambda\tau}{1+k\lambda\tau}\right)^k\right)\sum_{j=0}^{\infty}\left(\frac{k\lambda\tau}{k\lambda\tau+1}\right)^{k\lambda j} = \lambda \\
\mathbf{E}[X^2] &= \frac{1}{\tau^2}\frac{\lambda\tau}{2}\left(1-\left(\frac{k\lambda\tau}{2+k\lambda\tau}\right)^k\right)\sum_{j=0}^{\infty}\left(\frac{k\lambda\tau}{2+k\lambda\tau}\right)^{kj}\left(1+2\sum_{l=1}^{\infty}\left(\frac{k\lambda\tau}{1+k\lambda\tau}\right)^{kl}\right) \\
&= \frac{\lambda}{2\tau}\frac{1+\left(\frac{k\lambda\tau}{1+k\lambda\tau}\right)^k}{1-\left(\frac{k\lambda\tau}{1+k\lambda\tau}\right)^k} = \frac{\lambda}{2\tau}\frac{(1+k\lambda\tau)^k+(k\lambda\tau)^k}{(1+k\lambda\tau)^k-(k\lambda\tau)^k}
\end{aligned}$$

therefore,

$$\begin{aligned}
\mathrm{Var}(X) &= \frac{\lambda}{2\tau}\left(\frac{(1+k\lambda\tau)^k+(k\lambda\tau)^k}{(1+k\lambda\tau)^k-(k\lambda\tau)^k}-\frac{2k\lambda\tau}{k}\right) \\
\mathrm{SNR}_{\mathrm{dthin}}(X) &= \sqrt{\frac{2\lambda\tau}{\frac{(1+k\lambda\tau)^k+(k\lambda\tau)^k}{(1+k\lambda\tau)^k-(k\lambda\tau)^k}-2\lambda\tau}}
\end{aligned}$$

(Table 2.1, d-thinned Poisson & Fig. 2.2, D-THIN.), which we confirm with numerical simulations (Fig. 2.2.1, D-THIN.). Alternatively (see Appendix A.3), $\mathrm{SNR}_{\mathrm{dthin}} = h\ \mathrm{SNR}_{\mathrm{poi}}(X)$, where

$$h = \sqrt{\frac{1+k^2\lambda\tau+\ldots+\frac{1}{2}k^{k-1}(k-1)(\lambda\tau)^{k-2}+k^k(\lambda\tau)^{k-1}}{1+(k^2-2)\lambda\tau+\ldots+\frac{1}{6}k^{k-2}(k-1)(k+4)(\lambda\tau)^{k-2}+k^{k-1}(\lambda\tau)^{k-1}}}$$

At high $\lambda\tau$, d-thinning preserves the input's information, while at low $\lambda\tau$, a d-thinned Poisson spike-train is indistinguishable from a Poisson spike-train with the same rate:

$$h \underset{\lambda\tau\to\infty}{=} \sqrt{\frac{k^k(\lambda\tau)^{k-1}}{k^{k-1}(\lambda\tau)^{k-1}}} = \sqrt{k} \qquad \text{and} \qquad h \underset{\lambda\tau\to 0}{=} 1$$

Therefore, $\mathrm{SNR}_{\mathrm{dthin}}(X) \underset{\lambda\tau\to\infty}{=} \sqrt{k}\mathrm{SNR}_{\mathrm{poi}}(X) = \sqrt{2k\lambda\tau}$, which is the input spike-train's SNR, and $\mathrm{SNR}_{\mathrm{dthin}}(X) \underset{\lambda\tau\to 0}{=} \mathrm{SNR}_{\mathrm{poi}}(X)$.

At $k = 1$, $\mathrm{SNR}_{\mathrm{dthin}}(X) = \mathrm{SNR}_{\mathrm{poi}}(X)$ (i.e., all Poissonian input spikes are output). As $k \to \infty$, $\mathrm{SNR}_{\mathrm{dthin}}(X) \to \mathrm{SNR}_{\mathrm{per}}(X)$. However, $k \to \infty$ means processing many input, pre-thinned spikes for each output spike. When accounting for the energy costs associated with processing spikes, $\mathrm{SNR}_{\mathrm{dthin}}$ scales as $\sqrt[3]{(3c/2b)(\lambda\tau)^2}$ where c is the fanout of the thinned spike-train and b is the cost of processing each pre-thinned spike. After optimizing for costs, we find that $\mathrm{SNR}_{\mathrm{dthin}}$ scales as $(\lambda\tau)^{3/4}$.

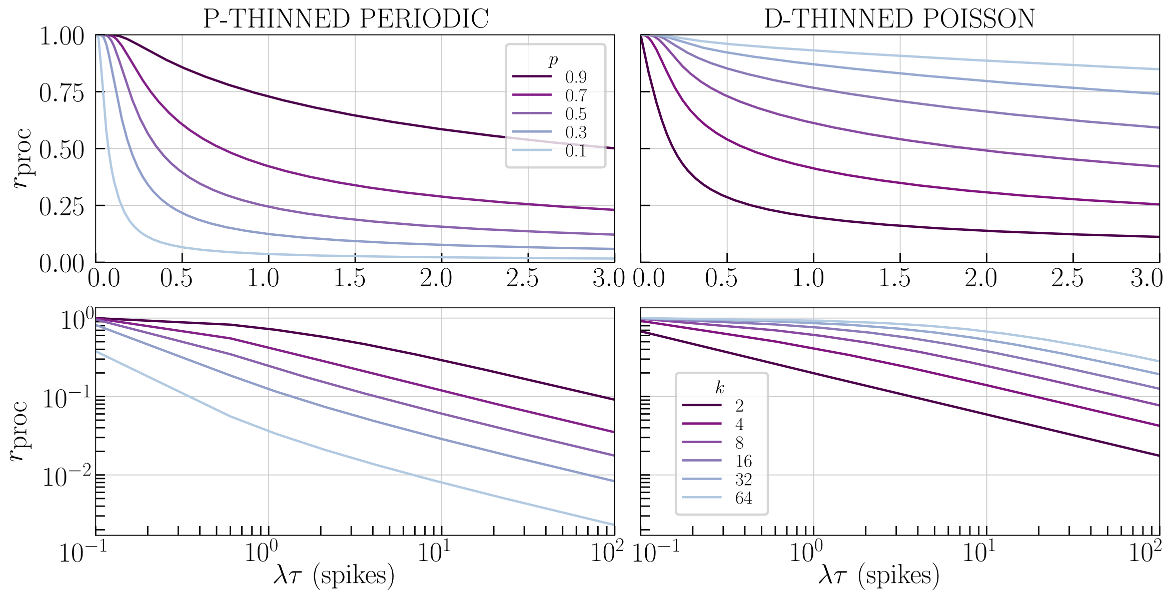


Figure 2.7: SNR Approximation.
Relative position of $\text{SNR}_{\text{pthin}}$ and $\text{SNR}_{\text{dthin}}$ between SNR_{poi} and SNR_{per} computed via (2.9). P-THINNED PERIODIC: For low $\lambda\tau$, $\text{SNR}_{\text{pthin}}$ is more closely approximated by SNR_{per} than SNR_{poi} (curves are near $r_{\text{proc}} = 1$), but as $\lambda\tau$ increases or p decreases, $\text{SNR}_{\text{pthin}}$ becomes more closely approximated by SNR_{poi} (curves approach $r_{\text{proc}} = 0$). D-THINNED POISSON: For low $\lambda\tau$, $\text{SNR}_{\text{dthin}}$ is more closely approximated by SNR_{per}, but as $\lambda\tau$ increases, $\text{SNR}_{\text{dthin}}$ becomes more closely approximated by SNR_{poi}. However, as k increases (i.e, w decreases), $\text{SNR}_{\text{dthin}}$ moves closer to SNR_{per}, in contrast to $\text{SNR}_{\text{pthin}}$.

2.2.5 Approximation Quality

While p-thinned periodic approaches Poisson and d-thinned Poisson approaches periodic as $w \to 0$, the convergence depends on $\lambda\tau$. To quantify this, consider

$$r_{\text{proc}}(\lambda\tau, w) = \frac{\text{SNR}_{\text{proc}}(X(\lambda\tau, w)) - \text{SNR}_{\text{poi}}(X(\lambda\tau))}{\text{SNR}_{\text{per}}(X(\lambda\tau)) - \text{SNR}_{\text{poi}}(X(\lambda\tau))} \tag{2.9}$$

which measures where the SNR of proc (either pthin or dthin) lies between the Poisson and periodic limits at a given $\lambda\tau$ (Fig. 2.2.5). At $r_{\text{proc}} = 0$, $\text{SNR}_{\text{proc}} = \text{SNR}_{\text{poi}}$, at $r_{\text{proc}} = 1$, $\text{SNR}_{\text{proc}} = \text{SNR}_{\text{per}}$, and at $r_{\text{proc}} = 0.5$, SNR_{proc} is halfway between SNR_{per} and SNR_{poi}.

2.3 Optimizing for Power

The advantages of weighted-summation by d over p-thinning ultimately translate into lower power consumption. Power is given by $P = c_{\text{in}}\lambda_{\text{in}} + c_{\text{out}}\lambda_{\text{out}}$ where $c_{\text{in(out)}}$ and $\lambda_{\text{in(out)}}$ are the energy cost per input (output) spike and input (output) spike rate, respectively. c_{in} for p-thinning is set as each input spike incurs a soma communication, weight lookup, random sample, and comparison (between the weight and the sample). c_{in} for d-thinning is set as each input spike incurs a soma communication, weight lookup, accumulation lookup, addition (between the weight and accumulation state), and thresholding. c_{out} is the same for both p and d-thinning: each output spike incurs a target lookup and set of synapse communications (i.e. the spike fans out).

Power for summing-and-weighting as implemented by merging and p-thinning is minimized when $p = 1$, (i.e., not weighting at all). Due to the Poisson statistics $\left(\text{SNR}_{\text{poi}}(X) = \sqrt{2\lambda\tau}\right)$,

$$P_{\text{pthin}} = c_{\text{in}}\frac{\text{SNR}_{\text{tgt}}^2}{2\tau p} + c_{\text{out}}\frac{\text{SNR}_{\text{tgt}}^2}{2\tau} = \frac{1}{2\tau}\left(\frac{c_{\text{in}}}{p} + c_{\text{out}}\right)\text{SNR}_{\text{tgt}}^2$$

for a target SNR (SNR_{tgt}) since $\lambda_{\text{out}} = \text{SNR}_{\text{tgt}}^2/2\tau$ and $\lambda_{\text{in}} = \lambda_{\text{out}}/p = \text{SNR}_{\text{tgt}}^2/2\tau p$. However, regardless of c_{in} or c_{out}, power consumption is minimized when $p = 1$ since p is restricted to $\{0, 1\}$.

In contrast, d-thinning incentives weighting for sufficient $\lambda\tau$. Since $\text{SNR}_{\text{dthin}}(X) \underset{\lambda\tau\to 0}{=} \sqrt{2\lambda\tau}$, and $\text{SNR}_{\text{dthin}}(X) \underset{\lambda\tau\to\infty}{=} \sqrt{2k\lambda\tau}$,

$$P_{\text{dthin}} \underset{\lambda\tau\to 0}{=} c_{\text{in}}\frac{\text{SNR}_{\text{tgt}}^2}{2\tau}k + c_{\text{out}}\frac{\text{SNR}_{\text{tgt}}^2}{2\tau} = \frac{1}{2\tau}\left(c_{\text{in}}k + c_{\text{out}}\right)\text{SNR}_{\text{tgt}}^2$$

$$P_{\text{dthin}} \underset{\lambda\tau\to\infty}{=} c_{\text{in}}\frac{\text{SNR}_{\text{tgt}}^2}{2\tau} + c_{\text{out}}\frac{\text{SNR}_{\text{tgt}}^2}{2k\tau} = \frac{1}{2\tau}\left(c_{\text{in}} + \frac{c_{\text{out}}}{k}\right)\text{SNR}_{\text{tgt}}^2$$

At low $\lambda\tau$, d-thinning power is minimized by $k = 1$ (i.e., not weighting at all) as with p-thinning. However, at high $\lambda\tau$, power decreases with increasing k (i.e., weighting improves power consumption). Between the extremes, the minimum-power k switches from $k = 1$ to $k > 1$ depending on c_{in} and c_{out} (Fig. 2.3).

2.4 Discussion

Although neuromorphic engineers look to biology for inspiration and biological neurons show trial-to-trial variability often modeled by Poisson statistics, Poisson spike-trains are poor at communicating information about their underlying rate. Therefore, injecting noise (via p-thinning) simply to mimic Poisson statistics observed in biological data (e.g., trial-to-trial variability of synaptic transmission) is an ill-founded pursuit (outside the corresponding biological context [8, 16, 43]).

We might seem to be claiming to violate the data-processing inequality as the accumulator "cleans up" a Poisson spike-train, but that's only an artifact of comparing the SNRs between spike trains resulting from p-thinning and d-thinning. If we instead compared the p-thinning and d-thinning SNR to their, respective, pre-thinned SNRs, we would see that both procedures decrease SNR (i.e,

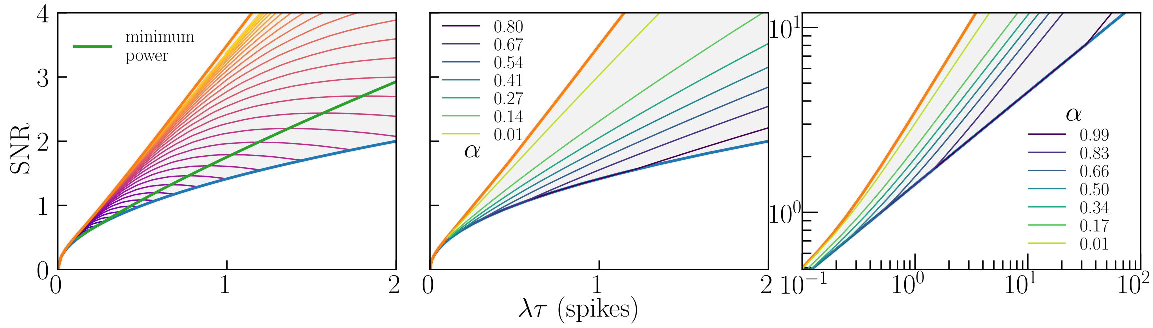


Figure 2.8: d-Thinned Poisson Power.
Set $c_{\text{in}} = \alpha$ and $c_{\text{out}} = 1 - \alpha$ so that $0 \leq \alpha \leq 1$ sweeps linearly between relative output and input energy costs. LEFT: $\alpha = 0.5$ (i.e., input and output spikes cost the same) iso-power contours. The minimum power curve passes through each contour's peak At low $\lambda\tau$, the minimum power curve follows $\text{SNR}_{\text{poi}}(X)$, but as $\lambda\tau$ increases, it lifts above $\text{SNR}_{\text{poi}}(X)$ as weighting lowers the power requirements for a given SNR. CENTER and RIGHT: Minimum power curves vary with α. The relative input-to-output cost determines when it becomes more efficient to start weighting to reduce the output spike-rate.

we respect the data-processing inequality). Our claim is on the relative drop in SNR—d-thinning produces a smaller relative decrease in SNR than p-thinning. Therefore, given a choice of p-thinning or d-thinning for weighting and a choice of summing before weighting or weighting before summing, summing and then d-thinning is the clear favorite. It results in $\text{SNR}_{\text{dthin}}$ scaling with the $3/4$ power of $\lambda\tau$ instead of as the square root—the case of p-thinning or (d-thinning) before summing. When Poisson statistics arise, they must arise from with contextual mechanisms embedded within larger frameworks.

3

A Serial H-Tree Router for Two-Dimensional Arrays

3.1 Router Functionality and Overhead

Advances in CMOS fabrication processes enable increases in the number and complexity of computational units in highly distributed and parallel architectures (e.g., neuromorphic processors; [2,37,39]), which calls for a corresponding increase in scalability and sophistication of routing mechanisms. Router area should be a reasonable fraction of the total system—router architecture is therefore dictated by client complexity. In this regard, high-overhead routers (e.g., meshes with parallel interfaces) capable of communicating arbitrary data-types at high bandwidths, are unsuited for intermediate-complexity clients with lower data-rate requirements. To provide multiple-data-type functionality for such clients, we adapt existing low-overhead routers.

Low-overhead routers contain a transmitter and a receiver [4]. The transmitter merges data from all of the clients into a single stream and adds source-identifying addresses to each datum to form a packet. The receiver takes a stream of packets, parses each destination-identifying address, and delivers the datum to the specified client.

For N clients, a low-overhead router's circuitry scales as $\mathcal{O}(\sqrt{N})$ by sharing resources. Clients are tiled in a two-dimensional (2D) array and share row and column wires within the array and transceiver circuitry at the edge of the array [4, 27] (Fig. 3.1, GRID ADDR). Sharing works correctly if certain timing assumptions are met [5, 6], but these assumptions are difficult to satisfy for long wires, which are susceptible to phenomena such as *charge relaxation*, whereby a significant voltage difference arises between the wire's two ends [25]. For this reason, grids do not readily scale to large arrays.

Our router presented herein switches from grid addresses to tree paths (Fig. 3.1, TREE PATH),

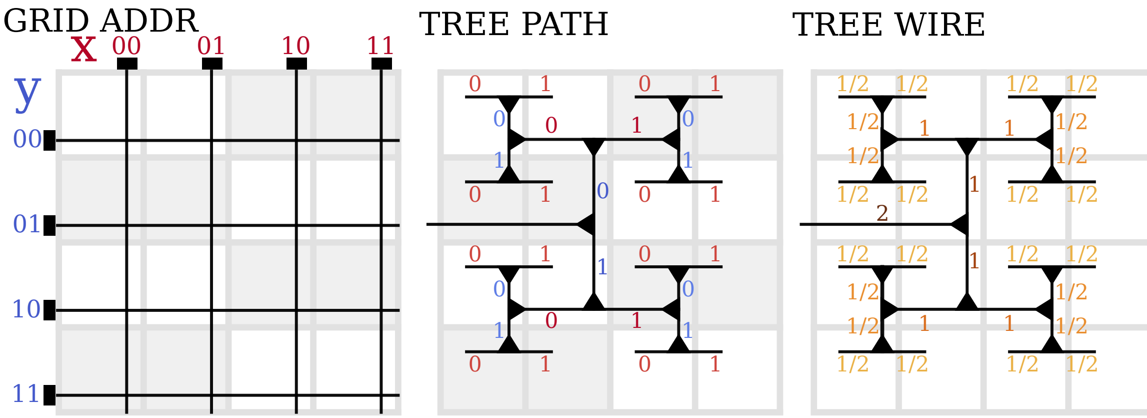


Figure 3.1: Grid Addresses and Tree Paths
Clients (white and gray squares) are tiled in a 2D array and routed to (or from) using a grid or a tree. GRID ADDR: A client's address is encoded by concatenating its x and y positions (in binary). Addressing circuitry is placed at the array's edge (black rectangles) and scales as $\mathcal{O}(\sqrt{N})$ for N clients ($N = 16$ shown). TREE PATH: A client's path is encoded by traversing the tree from the root to leaf (indexed by $n = 3$ and $n = 0$, respectively, in Algorithms 1 & 2). Each up and left (down and right) branch appends a 0 (1). Shaded squares indicate differences between tree and grid binary-number assignments (e.g., the bottom left client's grid-address is 0011, but its tree-path is 1010). Routing circuitry is embedded within the array (black triangles) and scales as $\mathcal{O}(N)$. TREE WIRE: Wire segments are annotated with their lengths.

trading an increase in logic circuitry for enhanced scalabilty and functionality. The increase in logic circuitry—from $\mathcal{O}(\sqrt{N})$ to $\mathcal{O}(N)$ for N clients—is worthwhile for emerging intermediate-complexity clients that use thick-oxide transistors for ultra-low power analog computation and much smaller thin-oxide transistors for ultra-fast digital communication [3]. The enhanced scalability arises because its asynchronous implementation's timing assumptions are easily met. And the enhanced functionality arises because its serial protocol supports multiple datatypes, whereas the grid's parallel protocol limits payload size. For backward compatibility, converting grid addresses into tree paths and vice versa is straightforward (Algorithms 1 & 2).

In Section 3.2, we describe how grid addresses and tree paths are encoded, show that both require $\mathcal{O}(N)$ wiring, and justify our choice of a 4-ary tree over a binary tree. In Section 3.3, we describe the serial link our router uses. In Section 3.4, we describe the logical design of the router's nodes. In Section 3.5, we describe how the router's leaves were customized for a neuromorphic application. In Section 3.6, we describe the router's logical and physical synthesis and its verification and validation. In Section 3.7, we conclude the chapter with a discussion of our results.

Algorithm 1 Converts Path (p) to Address (x,y)

Require: $l = \text{length}(p)$
  for $n = 0$ to $l/2 - 1$ **do**
    $x[n] \leftarrow p[2n]$
    $y[n] \leftarrow p[2n+1]$
  end for

Algorithm 2 Converts Address (x,y) to Path (p)

Require: $l = \text{length}(p)$
  for $n = 0$ to $l/2 - 1$ **do**
    $p[2n] \leftarrow x[n]$
    $p[2n+1] \leftarrow y[n]$
  end for

3.2 Tree Paths versus Grid Addresses

A 2D array can be routed to (or from) using a grid or a tree. We consider N clients, of unit width and height, arranged on a square grid (Fig. 3.1, WIRING).

Given equal link-widths, the tree requires less wiring than the grid. To calculate the length of the tree's wiring, W_t, we start from the N unit centers: N $\frac{1}{2}$-unit segments project horizontally from each center. At the second level, $\frac{N}{2}$ $\frac{1}{2}$-unit segments project vertically. At the third level, $\frac{N}{4}$ 1-unit segments project horizontally. This geometric pattern continues up to the root; each level alternates between horizontal and vertical orientation and halves the number of segments from the previous, lower level, while doubling the segment-length every other level. Overall, we have

$$W_\text{t} = \frac{1}{2}\left(N + \frac{N}{2}\right) + 1\left(\frac{N}{4} + \frac{N}{8}\right) + 2\left(\frac{N}{16} + \frac{N}{32}\right) + \ldots = \frac{3}{2}N$$

as N scales up (coloring matches Fig. 3.1, WIRING). For comparison, in the grid, each client adds 2 units of wire so that $W_\text{g} = 2N$. Therefore, $W_\text{t} = \frac{3}{4}W_\text{g}$: the tree uses up to 25% less wiring than the grid.[1] While the tree's segments become longer as we move from leaves to root, they are shared among more and more leaves.

However, the primary trade-off is the tree's larger transistor-count ($\mathcal{O}(N)$ versus $\mathcal{O}(\sqrt{N})$ for the grid), determined by the node-count times the transistors-per-node. To reduce the node-count, we opted for a 4-ary tree over a binary tree. A binary tree has $N-1$ nodes whereas a 4-ary tree has $\frac{N-1}{3}$ nodes. In general, a k-ary tree has $\frac{N-1}{k-1}$ nodes and $\log_k(N)$ levels. Consequently, switching divides the node-count by three, halves the number of levels, halves the latency, and doubles the unpipelined throughput.

If switching from binary to 4-ary doubles the transistors-per-node,[2] and divides the node count by three, we would expect to decrease the overall transistor count by 33%. However, nodes are not homogeneous; leaf nodes are tailored to clients' needs. The total transistor count in an N-client

[1] In emerging 3D processes, with wire-segments traveling along three axes, segment-count still halves at each level towards the root, but segment-length only doubles every third level (c.f. every other level in 2D). As a result, $W_\text{t} = \frac{7}{24}W_\text{g} \approx 0.29W_\text{g}$: the tree uses up to 71% less wiring.

[2] Doubling occurs if combinational gates (e.g., NANDs or NORs)—whose transistor count is 2× their fan-in—dominate. For sequential gates—whose state-holding elements are not replicated—the increase is sublinear. When gates are treed to build wider gates, the increase is supralinear.

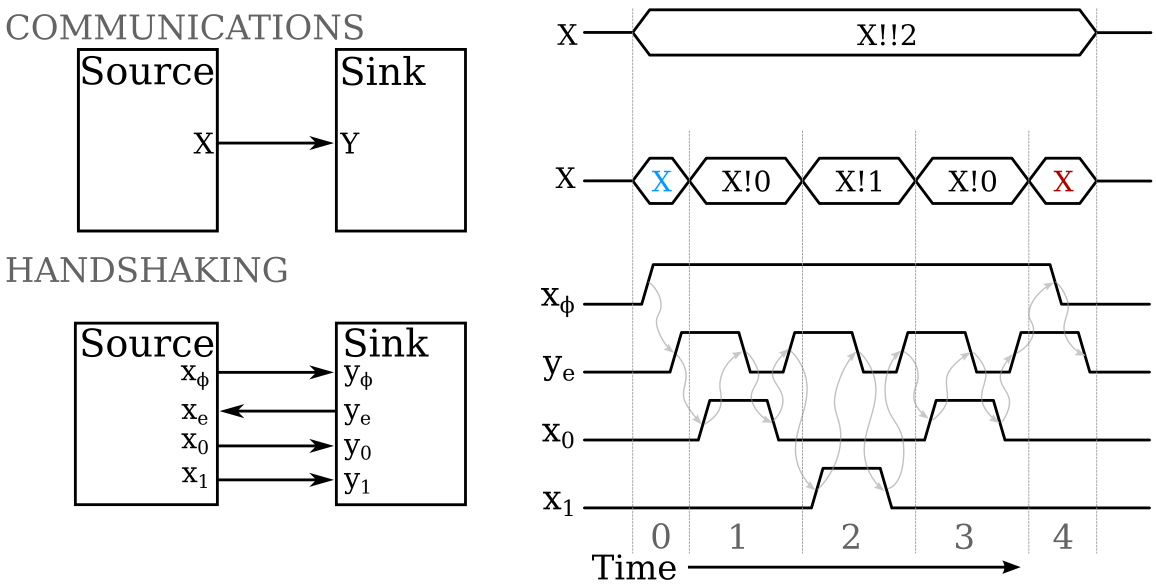


Figure 3.2: Serial Link Description at Two Levels of Abstraction
HANDSHAKING: *Source* drives control line x_ϕ and data lines x_0 and x_1. *Sink* drives control line y_e. Arrows point from driver to listener. Two-phase handshakes (time slots 0 and 4) initiate and terminate packet communication; four-phase handshakes (1, 2, and 3) send the packet's bits as 1-of-2 codes. All transitions are acknowledged (curved gray arrows), so the protocol is delay-insensitive. y_e's last transition is acknowledged by x_ϕ's initial transition in the next packet. A three-bit packet (010) is communicated in this example, but the protocol supports arbitrary-sized packets and 1-of-D codes using D data lines. COMMUNICATIONS: A channel connects *Source*'s output port (X) to *Sink*'s input port (Y). *Source*'s dataless communications (X and X) initiate and terminate packet transmission; such communications are colored blue and red, respectively, throughout the text. Its datafull communications ($X!0$ and $X!1$) send the packet's bits. The entire communication sequence may be consolidated into the single operation ($X!!2$).

k-ary tree whose leaf and intermediate nodes have $T_{\mathrm{L}k}$ and $T_{\mathrm{I}k}$ transistors each, respectively, is

$$T_{\mathrm{tot}k} = \frac{N-1}{k-1}\frac{(1-k/N)T_{\mathrm{I}k} + (k-1)T_{\mathrm{L}k}}{k-k/N} \tag{3.1}$$

For $T_{\mathrm{L}k} = T_{\mathrm{I}k} = T_k$, this expression reduces to $\frac{N-1}{k-1}T_k$: the total number of nodes times the transistors-per-node. Note that the ratio of leaf to intermedaite nodes is $k-1$:$1-k/N$, which approaches 1:1 and 3:1 for binary and 4-ary trees, respectively, as N increases. Thus, based on $T_{\mathrm{L}k}$'s and $T_{\mathrm{I}k}$'s values for our designs (Tab. 3.1),[3] which have different mixes of combinational and sequential logic and treed gates, switching from binary to 4-ary increases the average transistor count of the transmitter's and receiver's nodes by 2.6× and 1.6×, respectively. As a result, their overall transistor count reduces by 13.3% and 45%, respectively (see Tab. 3.1).

[3]The leaf node's communication is dataless—it requests or acknowledges.

	Transmitter		Receiver	
k	2	4	2	4
$T_{\mathrm{L}k}$	78	208	30	54
$T_{\mathrm{I}k}$	91	255	64	148
$T_{\mathrm{tot}4}/T_{\mathrm{tot}2}$	0.867		0.550	

Table 3.1: Node Transistor Counts & Binary:4-ary Ratios for Large N

We built our serial tree-router with quasi-delay-insensitive (QDI) circuits. The only timing assumption made is the *isochronic fork*. Signal-propagation delay along branches of such forks are assumed to be *equally* insignificant (hence the *iso* combining form; precise definitions may be found in [30,31]). This assumption is the minimal one necessary for useful computation with asynchronous circuits (i.e., Turing complete). No assumptions are made about signal-propagation delays through gates or nonisochronic wires, except that they are positive and finite.

3.3 Serial Communication Protocol

To keep link-width constant, we use serial communication. The path-length grows as we move from leaf to root in a tree. Hence, codes communicated over links closer to the root have more bits than those communicated over links closer to the leaves. A parallel protocol thus requires wider links (i.e. more wires) towards the root, whereas a serial protocol makes do with a constant width. Further, the latter allows us to communicate more than just the encoded paths; we can communicate data (e.g., configuration settings) as well.

Our serial-link follows a fully delay-insensitive version of the *bundled-data* protocol in [6] (Fig. 3.2, HANDSHAKING). For example, the following source generates a random bitstream and segments it into packets of arbitrary length (see Table 3.2 for syntax):

$$\begin{aligned} x_\phi\uparrow;\,[x_e];*[[&\text{true} \longrightarrow x_0\uparrow;\,[\neg x_e];\,x_0\downarrow;\,[x_e] \\ |\,&\text{true} \longrightarrow x_1\uparrow;\,[\neg x_e];\,x_1\downarrow;\,[x_e] \\ |\,&\text{true} \longrightarrow x_\phi\downarrow;\,[\neg x_e];x_\phi\uparrow;\,[x_e]]] \end{aligned}$$

Handshakes that demarcate the beginning and end of packet transmission are colored blue and red, respectively. If the x_ϕ branch is executed immediately, or consecutively, the packet contains no data. A sink that consumes the source's data operates as follows.

$$\begin{aligned} [y_\phi];\,y_e\uparrow;*[[&y_0 \vee y_1 \longrightarrow y_e\downarrow;\,[\neg y_0 \wedge \neg y_1];\,y_e\uparrow \\ [\,]\quad &\neg y_\phi \longrightarrow y_e\downarrow;[y_\phi];\,y_e\uparrow]] \end{aligned}$$

Selection (deterministic) is used instead of arbitration (nondeterministic) because the source guarantees mutual exclusion between the branches.

At a higher level of abstraction, we describe the source's and sink's operation simply in terms of *communications* on ports connected by a channel (Fig. 3.2, COMMUNICATIONS). For the source:

$$X;*[[\text{true} \longrightarrow X!0\,|\,\text{true} \longrightarrow X!1\,|\,\text{true} \longrightarrow X;X]]$$

X and X correspond to two-phase handshakes (on x_ϕ and x_e) that demarcate the packet (see Table 3.3 for notation). $X!$ corresponds to repeated four-phase handshakes (on $x_{0,1}$ and x_e) that send the payload. For the sink:

$$Y;*[[\overline{Y?} \longrightarrow Y?[]\overline{Y} \longrightarrow Y;Y]]$$

Note the unconventional use of the probe to check whether a *datafull* communication is pending. This probe ($\overline{Y?}$) corresponds to $y_0 \vee y_1$, whereas the *dataless* communication probe ($\overline{Y}$) corresponds to $\neg y_\phi$.

We introduce ?? and !! operators to describe serial read and write communications concisely (Fig. 3.2, COMMUNICATIONS). The source and sink are described as

$$*[[\text{true} \longrightarrow X!!\text{null}\,|\,\text{true} \longrightarrow X!!\text{Rand}()]] \parallel *[[Y??]]$$

where null is an empty string (i.e. the packet is empty) and Rand() returns a random, nonnegative integer.

3.4 Router Logical Design

The router consists of a transmitter and a receiver, both composed of a tree of nodes (Fig. 3.4.1). The transmitter merges packets from the clients into a single stream for transmission to the environment. The receiver does the inverse; it splits each packet in the stream off to the targeted client. For conciseness, we describe the nodes' operation for a binary tree (TX(2) and RV(2)). It is straightforward to extend these processes to a k-ary tree (TX(k) and RV(k)).

For a design space exporation see appendices B, C, and D. Designs were evaluated by their transistor costs assuming an array of 4096 neurons, 1024 synapses, and 256 memory banks. With each group of 1 syanpse and 4 neurons uses 28 bits of memory.

3.4.1 Transmitter

A transmitter node merges packet streams from its children into a single packet stream for its parent (another node one level closer to the root, unless the node is itself the root):

$$\begin{aligned}&\text{TX}(2)\\ \equiv\ &*[[\overline{C_0} \longrightarrow P!!(0 \oplus C_0??)\,|\,\overline{C_1} \longrightarrow P!!(1 \oplus C_1??)]]\end{aligned}$$

A packet from port $C_{0,1}$ is interpreted as a string; $a \oplus b$ prepends a to b (e.g., $1 \oplus 01 = 101$). Prepending a child's port index at that node in the tree to its data builds the overall path from leaf to root.

Expanding !! and ?? operators, and separating out arbitration, TX(2) becomes

$$
\begin{aligned}
&*[[\overline{C_0} \longrightarrow P; P!0;\ C_0; [\overline{C_0} \longrightarrow P;\ C_0]\\
&\quad |\ \overline{C_1} \longrightarrow P; P!1;\ C_1; [\overline{C_1} \longrightarrow P;\ C_1]]\\
&\|*[[\overline{C_0}? \longrightarrow P!C_0? [] \overline{C_1}? \longrightarrow P!C_1?]]
\end{aligned}
$$

Communications that demarcate when packet transmission begins and ends at the child and parent ports are colored blue and red, respectively. Putting $P!\{0,1\}$ before $C_{0,1}$ ensures that the child's index is prepended to the packet before the child's data are forwarded with the $P!C_{0,1}?$ communications.

Further expansion yields the following HSE.

$$
\begin{aligned}
&*[[c_{0\phi} \longrightarrow p_\phi\uparrow; [p_e]; p_0\uparrow; [\neg p_e]; p_0\downarrow; [p_e]; c_{0e}\uparrow;\\
&\quad [\neg c_{0\phi}]; p_\phi\downarrow; [\neg p_e]; c_{0e}\downarrow\\
&\ |\ c_{1\phi} \longrightarrow p_\phi\uparrow; [p_e]; p_1\uparrow; [\neg p_e]; p_1\downarrow; [p_e]; c_{1e}\uparrow;\\
&\quad [\neg c_{1\phi}]; p_\phi\downarrow; [\neg p_e]; c_{1e}\downarrow]],\\
&*[[c_{00} \longrightarrow p_0\uparrow; [\neg p_e]; c_{0e}\downarrow; [\neg c_{00}]; p_0\downarrow; [p_e]; c_{0e}\uparrow\\
&\ []\ c_{01} \longrightarrow p_1\uparrow; [\neg p_e]; c_{1e}\downarrow; [\neg c_{01}]; p_1\downarrow; [p_e]; c_{1e}\uparrow\\
&\ []\ c_{10} \longrightarrow p_0\uparrow; [\neg p_e]; c_{0e}\downarrow; [\neg c_{10}]; p_0\downarrow; [p_e]; c_{0e}\uparrow\\
&\ []\ c_{11} \longrightarrow p_1\uparrow; [\neg p_e]; c_{1e}\downarrow; [\neg c_{11}]; p_1\downarrow; [p_e]; c_{1e}\uparrow]]
\end{aligned}
$$

Note that the initial parent communication completes ($[p_e]$) and a code is transmitted to the parent before the initial child communication is acknowledged ($c_{0e}\uparrow$ or $c_{1e}\uparrow$). After that, the selection process relays the child's data.

We proceed by factorizing the arbitration process into the arbiter itself and the remaining child-parent communication:

$$
\begin{aligned}
&*[[c_{0\phi} \longrightarrow s_0\uparrow; [\neg c_{0\phi}]; s_0\downarrow\ |\ c_{1\phi} \longrightarrow s_1\uparrow; [\neg c_{1\phi}]; s_1\downarrow]],\\
&*[[s_0 \wedge \neg u \longrightarrow p_\phi\uparrow; [p_e]; w_0\uparrow; p_0\uparrow; [\neg p_e]; u\uparrow; w_0\downarrow;\\
&\quad p_0\downarrow; [p_e]; c_{0e}\uparrow; [\neg s_0]; p_\phi\downarrow; [\neg p_e]; c_{0e}\downarrow; u\downarrow\\
&\ []\ s_1 \wedge \neg u \longrightarrow p_\phi\uparrow; [p_e]; w_1\uparrow; p_1\uparrow; [\neg p_e]; u\uparrow; w_1\downarrow;\\
&\quad p_1\downarrow; [p_e]; c_{1e}\uparrow; [\neg s_1]; p_\phi\downarrow; [\neg p_e]; c_{1e}\downarrow; u\downarrow]]
\end{aligned}
$$

$s_{0,1}$ are introduced to store the selection result; $w_{0,1}$ are introduced to distinguish the state immediately after $[p_e]$ (prepending the index) from that immediately after $[p_e]$ (acknowledging the child); and u is introduced to preserve mutual exclusion in the selection process when its branches are implemented as concurrent processes. It prevents the s_1 branch (i.e. $p_\phi\uparrow; [p_e]; \ldots$) from beginning before the s_0 branch completes (i.e. $p_\phi\downarrow; [\neg p_e]; \ldots$) when $c_{1\phi}$ is high and the arbiter executes $s_1\uparrow$ immediately after $s_0\downarrow$.

Our 4-ary transmitter tree's node uses a four-way arbiter (Fig. 3.4.1). Three mutual-exclusion elements are interconnected in a binary decision-tree by handshaking circuitry ($c_{0:3\phi}$ and $s_{0:3}$ connect to the two ARB2's $\_c_{0,1\mathrm{i}}$ inputs and $c_{0,1\mathrm{o}}$ outputs, respectively) [32]. CHP and HSE are omitted for brevity. For comparison, a binary tree's node requires just one mutual-exclusion element—with no additional overhead. Although the arbiter design used here contains no pipelining, a greedy, but fair arbiter is described in Appendix E.2.

The transmitter node's HSE (sans ARB(4)) is implemented by the following production rule set (PRS).

$$\begin{array}{lcl} \neg u \wedge (s_0 \vee s_1) & \rightarrow & p_\phi\uparrow \\ (c_{0e} \wedge \neg s_0) \vee (c_{1e} \wedge \neg s_1) & \rightarrow & p_\phi\downarrow \end{array}$$

$$\begin{array}{lcl} s_0 \wedge p_e \wedge \neg u & \rightarrow & w_0\uparrow \\ s_1 \wedge p_e \wedge \neg u & \rightarrow & w_1\uparrow \end{array} \qquad \begin{array}{lcl} u & \rightarrow & w_0\downarrow \\ u & \rightarrow & w_1\downarrow \end{array}$$

$$\begin{array}{lcl} c_{00} \vee c_{10} \vee w_0 & \rightarrow & p_0\uparrow \\ c_{01} \vee c_{11} \vee w_1 & \rightarrow & p_1\uparrow \end{array} \qquad \begin{array}{lcl} \neg(c_{00} \vee c_{10} \vee w_0) & \rightarrow & p_0\downarrow \\ \neg(c_{01} \vee c_{11} \vee w_1) & \rightarrow & p_1\downarrow \end{array}$$

$$(w_0 \vee w_1) \wedge \neg p_e \rightarrow u\uparrow \qquad \neg(c_{0e} \vee c_{1e} \vee p_\phi) \rightarrow u\downarrow$$

$$\begin{array}{lcl} s_0 \wedge u \wedge p_e \wedge \neg c_{1e} & \rightarrow & c_{0e}\uparrow \\ s_1 \wedge u \wedge p_e \wedge \neg c_{0e} & \rightarrow & c_{1e}\uparrow \end{array} \qquad \begin{array}{lcl} \neg p_e & \rightarrow & c_{0e}\downarrow \\ \neg p_e & \rightarrow & c_{1e}\downarrow \end{array}$$

3.4.2 Receiver

A receiver node splits a packet stream from its parent into packet streams for its children (another node one level closer to the leaves, unless the node is a leaf itself):

$$\begin{array}{l} \quad \mathrm{RV}(2) \\ \equiv *[[P??(s,d) \bullet [s = 0 \longrightarrow C_0!!d \,[\!]\, s = 1 \longrightarrow C_1!!d]]] \end{array}$$

It uses the packet's first word (written into s) to decide which child to send the remainder of the packet (written into d); s has 1 bit for a binary tree or 2 bits for a 4-ary tree (RV(4)).

We expand RV(2)'s ?? and !! communications as follows.

$$\begin{array}{l} P;P?s \bullet [s = 0 \longrightarrow C_0 \,[\!]\, s = 1 \longrightarrow C_1]; \\ *[[\overline{P?} \wedge s = 0 \longrightarrow C_0!P? \\ \;\; [\!]\, \overline{P?} \wedge s = 1 \longrightarrow C_1!P? \\ \;\; [\!]\, \overline{P} \longrightarrow P \bullet [s = 0 \longrightarrow C_0 \,[\!]\, s = 1 \longrightarrow C_1]; \\ \qquad P;P?s \bullet [s = 0 \longrightarrow C_0 \,[\!]\, s = 1 \longrightarrow C_1]]] \end{array}$$

This process can be expanded further as

$$
\begin{array}{l}
*[[p_0 \wedge s_0 \longrightarrow c_{00}\uparrow;\ [\neg c_{0e}];\ p_e\downarrow;\ [\neg p_0];\ c_{00}\downarrow;\ [c_{0e}];\ p_e\uparrow \\
\quad [\,]\, p_1 \wedge s_0 \longrightarrow c_{01}\uparrow;\ [\neg c_{0e}];\ p_e\downarrow;\ [\neg p_1];\ c_{01}\downarrow;\ [c_{0e}];\ p_e\uparrow \\
\quad [\,]\, p_0 \wedge s_1 \longrightarrow c_{10}\uparrow;\ [\neg c_{1e}];\ p_e\downarrow;\ [\neg p_0];\ c_{10}\downarrow;\ [c_{1e}];\ p_e\uparrow \\
\quad [\,]\, p_1 \wedge s_1 \longrightarrow c_{11}\uparrow;\ [\neg c_{1e}];\ p_e\downarrow;\ [\neg p_1];\ c_{11}\downarrow;\ [c_{1e}];\ p_e\uparrow \\
\quad [\,]\, \neg p_\phi \longrightarrow s_0\downarrow, s_1\downarrow; \\
\qquad c_{0\phi}\downarrow, c_{1\phi}\downarrow;\ [\neg c_{0e} \wedge \neg c_{1e}];\ p_e\downarrow \boxed{;} [p_\phi];\ p_e\uparrow; \\
\qquad [p_0 \longrightarrow s_0\uparrow;\ p_e\downarrow;\ [\neg p_0];\ c_{0\phi}\uparrow;\ [c_{0e}];\ p_e\uparrow \\
\qquad [\,]\, p_1 \longrightarrow s_1\uparrow;\ p_e\downarrow;\ [\neg p_1];\ c_{1\phi}\uparrow;\ [c_{1e}];\ p_e\uparrow]]]
\end{array}
$$

After reset, the process resumes at $\boxed{;}$.

To realize these five branches as five concurrent processes, we must preclude the the first four from starting immediately after the fifth process executes $s_{0,1}\uparrow$. We accomplish this by replacing $s_{0,1}$ in their guards with $c_{0,1\phi}$, which also indicate the selected child.

$$
\begin{array}{l}
*[[p_0 \wedge c_{0\phi} \longrightarrow c_{00}\uparrow;\ [\neg c_{0e}];\ p_e\downarrow;\ [\neg p_0];\ c_{00}\downarrow;\ [c_{0e}];\ p_e\uparrow \\
\quad [\,]\, p_1 \wedge c_{0\phi} \longrightarrow c_{01}\uparrow;\ [\neg c_{0e}];\ p_e\downarrow;\ [\neg p_1];\ c_{01}\downarrow;\ [c_{0e}];\ p_e\uparrow \\
\quad [\,]\, p_0 \wedge c_{1\phi} \longrightarrow c_{10}\uparrow;\ [\neg c_{1e}];\ p_e\downarrow;\ [\neg p_0];\ c_{10}\downarrow;\ [c_{1e}];\ p_e\uparrow \\
\quad [\,]\, p_1 \wedge c_{1\phi} \longrightarrow c_{11}\uparrow;\ [\neg c_{1e}];\ p_e\downarrow;\ [\neg p_1];\ c_{11}\downarrow;\ [c_{1e}];\ p_e\uparrow \\
\quad [\,]\, \neg p_\phi \longrightarrow s_0\downarrow, s_1\downarrow;\ ss\downarrow;\ v\downarrow; \\
\qquad c_{0\phi}\downarrow, c_{1\phi}\downarrow;\ [\neg c_{0e} \wedge \neg c_{1e}];\ p_e\downarrow \boxed{;} [p_\phi];\ p_e\uparrow; \\
\qquad [p_0 \longrightarrow s_0\uparrow;\ ss\uparrow;\ p_e\downarrow;\ [\neg p_0];\ v\uparrow;\ c_{0\phi}\uparrow;\ [c_{0e}];\ p_e\uparrow \\
\qquad [\,]\, p_1 \longrightarrow s_1\uparrow;\ ss\uparrow;\ p_e\downarrow;\ [\neg p_1];\ v\uparrow;\ c_{1\phi}\uparrow;\ [c_{1e}];\ p_e\uparrow]]]]
\end{array}
$$

v is introduced to allow $c_{0,1\phi}$ to be combinational and ss is introduced to reduce the length of p_e's pull-up and pull-down chains (see PRS below).

The following PRS implements the receiver node's HSE.

$$
\begin{array}{lcl}
p_\phi \wedge \neg ss \vee c_{0e} \vee c_{1e} & \rightarrow & p_e\uparrow \\
(\neg p_\phi \vee ss) \wedge \neg c_{0e} \wedge \neg c_{1e} & \rightarrow & p_e\downarrow
\end{array}
$$

$$
\begin{array}{lclclcl}
s_0 \vee s_1 & \rightarrow & ss\uparrow & \qquad & \neg s_0 \wedge \neg s_1 & \rightarrow & ss\downarrow \\[1ex]
p_0 \wedge \neg v & \rightarrow & s_0\uparrow & & \neg p_\phi & \rightarrow & s_0\downarrow \\
p_1 \wedge \neg v & \rightarrow & s_1\uparrow & & \neg p_\phi & \rightarrow & s_1\downarrow \\[1ex]
ss \wedge \neg p_0 \wedge \neg p_1 & \rightarrow & v\uparrow & & \neg ss & \rightarrow & v\downarrow \\[1ex]
v \wedge s_0 & \rightarrow & c_{0\phi}\uparrow & & \neg v \vee \neg s_0 & \rightarrow & c_{0\phi}\downarrow \\
v \wedge s_1 & \rightarrow & c_{1\phi}\uparrow & & \neg v \vee \neg s_1 & \rightarrow & c_{1\phi}\downarrow \\[1ex]
p_0 \wedge c_{0\phi} & \rightarrow & c_{00}\uparrow & & \neg p_0 \vee \neg c_{0\phi} & \rightarrow & c_{00}\downarrow \\
p_1 \wedge c_{0\phi} & \rightarrow & c_{01}\uparrow & & \neg p_1 \vee \neg c_{0\phi} & \rightarrow & c_{01}\downarrow \\
p_0 \wedge c_{1\phi} & \rightarrow & c_{10}\uparrow & & \neg p_0 \vee \neg c_{1\phi} & \rightarrow & c_{10}\downarrow \\
p_1 \wedge c_{1\phi} & \rightarrow & c_{11}\uparrow & & \neg p_1 \vee \neg c_{1\phi} & \rightarrow & c_{11}\downarrow
\end{array}
$$

3.5 Router Application

We connected a 2D array of spiking-neuron clusters to a datapath using our asynchronous serial tree-router, a natural choice for spike communication. A product of continuous, noisy analog dynamics at biological timescales, spikes are relatively infrequent (sub-kHz) and asynchronous (there's no clock). Each cluster contains 16 spike-generating soma circuits, 4 spike-consuming synapse circuits, and a configuration memory.

Clusters are tiled in a 16×16 array. To service the 4,096 somas, 1,024 synapses, and 256 memories, the transmitter's and receiver's trees are six ($4^6 = 4096$) and five ($4^5 = 1024$) levels deep, respectively. Half of the receiver's 1,024 output ports suffice to service all 1,024 synapses because each port supplies 2 bits (a 1-of-4 code) whereas each synapse needs only 1 bit (indicates whether a spike is excitatory or inhibitory). Thus, 512 ports are left over to service the 256 memories. We customized the transmitter's and receiver's leaf nodes to suit this application as follows.

3.5.1 Transmitter Leaf

The transmitter leaf transmits a soma's spike up the tree by creating a packet containing the soma's index. With no other data to convey, the transmitter node is simplified to

$$\begin{aligned}\mathrm{TXL}(2) \equiv *[[\overline{C_0} &\longrightarrow C_0\bullet(P;P!0);\ [\overline{C_0} \longrightarrow C_0 \bullet P]\\ |\ \overline{C_1} &\longrightarrow C_1\bullet(P;P!1);\ [\overline{C_1} \longrightarrow C_1 \bullet P]]\end{aligned}$$

Note that our design actually instantiates TXL(4). We omit its HSE and PRS for brevity.

Somas lock up the transmitter during spike emission. When emitting a spike, a soma initiates packet transmission with a two-phase handshake ($C_{0,1}$). Afterwards, the soma enters a refractory period for up to a few milliseconds before executing another two-phase handshake ($C_{0,1}$) to terminate transmission. If communications within the transmitter are slackless, the soma will lock up the transmitter during its refractory period. To prevent this, we insert a buffer (i.e., latch) between the soma and the transmitter's leaf.

3.5.2 Receiver Leaf

The receiver leaf services four synapses as well as a configuration memory (via a deserializer). We repurpose two of the receiver node's 2-bit ports to service the four synapses and use a third port to communicate with the memory:

$$
\begin{aligned}
\mathrm{RVL}(4) \equiv\ & P; P?s \bullet [s = 2 \longrightarrow C_2 \,[]\, s \neq 2 \longrightarrow \mathrm{skip}];\\
& *[[\overline{P?} \wedge s = 0 \longrightarrow C_0!P?\\
& \quad []\,\overline{P?} \wedge s = 1 \longrightarrow C_1!P?\\
& \quad []\,\overline{P?} \wedge s = 2 \longrightarrow C_2!P?\\
& \quad []\,\overline{P} \longrightarrow P \bullet [s = 2 \longrightarrow C_2 \,[]\, s \neq 2 \longrightarrow \mathrm{skip}];\\
& \qquad P; P?s \bullet [s = 2 \longrightarrow C_2 \,[]\, s \neq 2 \longrightarrow \mathrm{skip}]]]
\end{aligned}
$$

Only the third port (C_2) continues with the serial protocol. HSE and PRS are omitted for brevity.

Synapses, like somas, require buffering. Depending on its analog biasing, a synapse may take up to a few milliseconds to acknowledge an input spike. We add a full cycle of slack to the otherwise slackless communication from root to synapse. One half-cycle is built into the leaf; a standard weak-precharge half-buffer provides the other.

The configuration memory accepts 6 bits of address and 2 bits of data (its 128 bits are organized into eight rows and eight 2-bit-wide columns). These 8 bits are encoded in four 1-of-4 codes that the deserializer receives in series from the receiver leaf and presents in parallel to the memory.

3.5.3 Serial–Parallel Conversion

The deserializer converts M sequentially delivered 1-of-4 codes into a M×1-of-4 parallel code using a chain of M DEs (Fig. 3.4.2, DESERIAL). For 1-of-2 codes, DE's HSE is:

$$
\begin{aligned}
*[[s_i]; [& x_0 \longrightarrow y_0\uparrow; x_a\uparrow; [\neg x_0]; s_o\uparrow; x_a\downarrow; [\neg s_i]; y_0\downarrow; s_o\downarrow\\
[]\,& x_1 \longrightarrow y_1\uparrow; x_a\uparrow; [\neg x_1]; s_o\uparrow; x_a\downarrow; [\neg s_i]; y_1\downarrow; s_o\downarrow]]
\end{aligned}
$$

For each conversion, $s_{i,o}$ propagate an event along the chain twice. The first time, serial input codes are latched to build the parallel output. The second time, the parallel output is cleared, which happens once a C-element that closes the chain receives the environment's acknowledge. PRS for a 1-of-2 version of DE is as follows.

$$
\begin{array}{lcllcl}
\neg s_o \wedge s_i \wedge x_0 & \rightarrow & y_0\uparrow & \neg s_i & \rightarrow & y_0\downarrow\\
\neg s_o \wedge s_i \wedge x_1 & \rightarrow & y_1\uparrow & s_o \vee \neg v_y & \rightarrow & x_a\downarrow\\
\neg s_o \wedge v_y & \rightarrow & x_a\uparrow & \neg s_i & \rightarrow & y_1\downarrow\\
y_0 \vee y_1 & \rightarrow & v_y\uparrow & \neg y_0 \wedge \neg y_1 & \rightarrow & v_y\downarrow\\
v_y \wedge \neg x_0 \wedge \neg x_1 & \rightarrow & s_o\uparrow & \neg v_y & \rightarrow & s_o\downarrow
\end{array}
$$

v_y is introduced to shorten transistor chains.

The serializer does the converse of the deserializer: It uses a chain of M SEs to slice a M×1-of-4 parallel code into M 1-of-4 codes and forwards them sequentially to the environment using SEQ (Fig. 3.4.2, SERIAL). For 1-of-2 codes, SE's HSE is:

$$
\begin{array}{l}
*[[s_i];\\
[x_0 \longrightarrow y_0{\uparrow};\, [y_a];\, u{\uparrow};\, y_0{\downarrow};\, [\neg y_a];\, s_o{\uparrow};\, [\neg s_i];\, u{\downarrow};\, [\neg x_0];\, s_o{\downarrow}\\
[\!]\, x_1 \longrightarrow y_1{\uparrow};\, [y_a];\, u{\uparrow};\, y_1{\downarrow};\, [\neg y_a];\, s_o{\uparrow};\, [\neg s_i];\, u{\downarrow};\, [\neg x_1];\, s_o{\downarrow}\\
]]
\end{array}
$$

u is added to distinguish states before and after the $y_{0,1}$-y_a handshake. As with the deserializer, $s_{i,o}$ propagate an event along the chain twice for each conversion. The first time, each SE relays a code to SEQ. The second time, each SE checks that its parallel-input slice is cleared. SE's PRS is:

$$
\begin{array}{lcllcl}
x_0 \wedge \neg u \wedge s_i & \rightarrow & y_0{\uparrow} & u \wedge \neg s_o & \rightarrow & y_0{\downarrow}\\
x_1 \wedge \neg u \wedge s_i & \rightarrow & y_1{\uparrow} & u \wedge \neg s_o & \rightarrow & y_1{\downarrow}\\
s_i \wedge y_a & \rightarrow & u{\uparrow} & \neg s_i & \rightarrow & u{\downarrow}\\
u \wedge \neg y_a & \rightarrow & s_o{\uparrow} & \neg u \wedge \neg x_0 \wedge \neg x_1 & \rightarrow & s_o{\downarrow}
\end{array}
$$

SEQ's HSE is:

$$
\begin{array}{l}
*[[s_i];\, s_o{\uparrow};\, [x_0 \vee x_1];\, y_\phi{\uparrow};\, [\neg s_i \wedge y_e];\, y_\phi{\downarrow};\, [\neg y_e];\, s_o{\downarrow}],\\
*[[x_0 \wedge y_e \longrightarrow y_0{\uparrow};\, [\neg y_e];\, x_a{\uparrow};\, [\neg x_0];\, y_0{\downarrow};\, [y_e];\, x_a{\downarrow}\\
\quad [\!]\, x_1 \wedge y_e \longrightarrow y_1{\uparrow};\, [\neg y_e];\, x_a{\uparrow};\, [\neg x_1];\, y_1{\downarrow};\, [y_e];\, x_a{\downarrow}]]
\end{array}
$$

By closing the chain, it initiates ($y_\phi{\uparrow}$; $[y_e]$) and terminates ($y_\phi{\downarrow}$; $[\neg y_e]$) packet transmission on the event's first and second pass, respectively. In between, it forwards codes that SEs provide. SEQ's PRS is:

$$
\begin{array}{lcllcl}
x_0 \vee x_1 & \rightarrow & y_\phi{\uparrow} & \neg s_i \wedge y_e & \rightarrow & y_\phi{\downarrow}\\
(y_0 \vee y_1) \wedge \neg y_e & \rightarrow & x_a{\uparrow} & y_e & \rightarrow & x_a{\downarrow}\\
y_e \wedge x_0 & \rightarrow & y_0{\uparrow} & \neg x_0 & \rightarrow & y_0{\downarrow}\\
y_e \wedge x_1 & \rightarrow & y_1{\uparrow} & \neg x_1 & \rightarrow & y_1{\downarrow}\\
\neg s_i \wedge \neg y_e \wedge \neg y_\phi & \rightarrow & s_o{\downarrow} & s_i & \rightarrow & s_o{\uparrow}
\end{array}
$$

3.6 Synthesis and Validation

For logical synthesis, we described logical hierarchy and PRS in the Asynchronous Compiler Tools (ACT) language.[4] We verified logical correctness with PRSIM, a discrete-event simulator that executes PRS with randomized delays [1], and then checked for logical-physical consistency in the presence of transistor parasitic capacitances with CoSIM, a PRSIM–SPICE co-simulator [1].

For physical synthesis, we decomposed our ACT into standard cells and generated their layouts with cellTK [22]. Encounter (*Cadence*) place-and-routed lower-level router circuitry—4 transmitter leaves (to service 16 somas), their parent node, a receiver leaf (to service 4 synapses and an SRAM), and a deserializer (to interface with the SRAM)—in the lower 43% ($547\mu m^2$) of the neuron-cluster

[4] Available at `https://github.com/samfok/AER_serial_tree_router`

tile ($1,261\mu m^2$). Of this router area, 14% ($76\mu m^2$) was reserved (cutout) for higher-level circuitry (Fig. 3.4.2, TILE).

We placed tiles in a 16×16-array and placed the router's higher-level nodes in their cutouts, along with repeaters to drive long wires (Fig. 3.4.2, TILE16×16, and H-TREES). We extracted parasitic resistances and capacitances from this layout and performed simulations to check for spurious transitions and to predict the router's maximum throughput.

Our postlayout simulations predicted that the transmitter and receiver could communicate up to 42.5 and 50.8 Mspike/s, respectively (Fig. 3.6).[5] Codes from (or to) nodes lower in the tree take longer (e.g., 4.31 ns from the transmitter tree's leaves versus 1.28 ns from its root) because the number of communications involved increases (from 6 at the leaf to 1 at the root). On average, 4.5 four-phase communications are performed, including one for the 2 two-phase communications that demarcate the packet. At 4 phases per communication and 4 transitions per phase, transversing six nodes involves 432 transitions.[6] Thus, the 42.5 Mspike/s cycle-rate corresponds to 56 ps per transition, in line with expectations for a 28-nm process.

Post-fabrication in a 28-nm, fully depleted, silicon-on-insulator process, we brought the chip up and validated the router's functionality. (Fig. 3.6). From two chips, we measured maximum throughputs of 27.4 and 26.1 Mspikes/s for the transmitter and 18.1 and 18.5 Mspikes/s for the receiver.[7] Differences between simulations the chip measurements are explained by additional delays introduced by unpipelined datapath communications.

3.7 Discussion

Pioneering researchers developed transmitters and receivers to write and read spikes to and from 1D or 2D arrays of silicon neurons using the *address-event representation* (AER; [4, 24, 27, 41]). A neuron's address is transmitted every time it spikes, hence the name *address-event*. In 1D, the spike is identified by a unique address assigned to each neuron. In 2D, the spike is identified by the neuron's row and column addresses and, in first-generation designs, these addresses are transmitted in parallel.

Second-generation designs communicated row and column addresses in series. In addition to saving wires by multiplexing, this so-called *word-serial* protocol supports packets with an arbitrary number of words. Thus, additional column addresses could be appended to communicate multiple spikes read from or written to the same row in parallel [5, 6, 25]. This so-called *burst-mode* offered higher throughput, servicing arrays containing as many as 64k somas and 256k synapses [2] at rates up to 43.4M spike/s (ignoring off-chip delays) [7]. Array or chip addresses could be prepended

[5] In operation, somas generate up to 500 spike/s each, and synapses consume up to 1000 spikes/s each, so we expect the transmitter and receiver to communicate 2 and 1 Mspikes/s, respectively

[6] In our PRSIM simulations, we counted 422 transitions for the transmitter and serializer and 481 transitions for the deserializer and receiver.

[7] Equivalently, the fabricated router can service arrays of up to 53.5k somas and 18.3k synapses, respectively

to further expand the address-space. Thus, an address-event-based router could service multiple arrays distributed across multiple chips [11, 36, 38]. Further, data as well as addresses could be communicated over the link (or bus) connecting the transmitter to the receiver [17]. In this fashion, multiple spikes read in parallel from small groups of neurons have been transmitted using a single dataword, boosting throughput, which had plateaued at 50M spike/s [10], to 300M spike/s [42].

To communicate configuration datawords to or from individual neurons—or clusters thereof—we could widen the neuronal interface, but the additional bandwidth would be largely wasted. Datawords use all of the wires but occur rarely, whereas spikes occur frequently but only use one (e.g., a soma's output) or two (e.g., a synapse's excitatory or inhibitory input) wires. We thus keep the neuronal interface narrow and transmit data serially, saving wires by taking more time. In addition to supporting multiple data-types efficiently, a serial protocol places no limit on the number of bits a dataword can have, unlike a parallel protocol.

We did away with timing-assumptions by switching from row-column addresses to tree paths. Striking a balance between node-count and node-complexity, we chose a 4-ary over a binary tree, which reduced transistor-count by 19.1% overall. Returns diminish for higher degrees because realizing wider gates requires treeing narrower gates (with no more than four transistors in series).[8] Although its thin-oxide transistors outnumber the neuron-cluster's thick-oxide transistors 1.9:1, the router takes up only 43% of the total area because thick-oxide transistors are much larger than thin oxide transistors.

Throughput may be enhanced substantially by pipelining the otherwise slackless communication from leaf to root (transmitter) or root to leaf (receiver) and between router and datapath. Pipelining can be added to the current design at no additional area cost by replacing repeaters with latches or placing latches in unused tile cutouts (see Appendix E.1. Subsequent codes would take no more time than the first one, which takes 1.28 (transmitter) or 1.8 ns (receiver) (Fig. 3.6). Therefore, with seven communications per spike, pipelining would increase throughput from 42.5M to 111.6M spike/s (transmitter) or from 50.7M to 79.4M spike/s (receiver).

[8]Gates with longer chains operate much slower, increasing the duration that downstream gates pass short-circuit currents.

	Assignment
$x\uparrow$ / $x\downarrow$	Set boolean variable x to true / false
	Program Composition
$s_1; s_2$	Execute segment s_1 and then s_2
s_1, s_2	Execute s_1 concurrently with s_2
$*[s]$	Execute s repeatedly
	Boolean Operations
x / $\neg x$	Return the value of x / negated value of x
$e_1 \wedge e_2$	Return the logical-and of e_1 and e_2
$e_1 \vee e_2$	Return the logical-or of e_1 and e_2
	Branching
$[e]$	Wait until boolean expression e is true
$[e_1 \rightarrow s_1]$	When e_1 becomes true, execute s_1
$[e_1 \rightarrow s_1 \mid e_2 \rightarrow s_2]$	If boolean e_1 (e_2) is true, execute s_1 (s_2)
	If both are true, execute either s_1 or s_2
	If both are false, wait
$[e_1 \rightarrow s_1 \,\square\, e_2 \rightarrow s_2]$	If boolean e_1 (e_2) is true, execute s_1 (s_2)
	Assume e_1 and e_2 cannot both be true
	If both are false, wait

Table 3.2: Handshaking Expansion (HSE) Syntax

	Assignment
$x := d$	Set variable x to d's value
	Communication
X	Communicate on port X (dataless)
$X!x$	Write value of x to X
$X?x$	Read value from X to x
$Y!X?$	Read value from X and write it to port Y
$\overline{X}$	*True* if a communication is pending and *false* if not
	Program Composition
$S_1; S_2$	Execute segment S_1 and then S_2
$S_1 \parallel S_2$	Execute S_1 in parallel with S_2
$S_1 \bullet S_2$	Overlap the execution of S_1 and S_2 (called *bullet*)
$*[S]$	Execute S repeatedly
	Boolean Operations
$\neg, \wedge, \vee$	Same as in Table 3.2
$x = d$	Return *true* if x's value equals d's and *false* if not
	Branching
$\rightarrow, \mid, \square$	Same as in Table 3.2

Table 3.3: Communicating Hardware Processes (CHP) Syntax

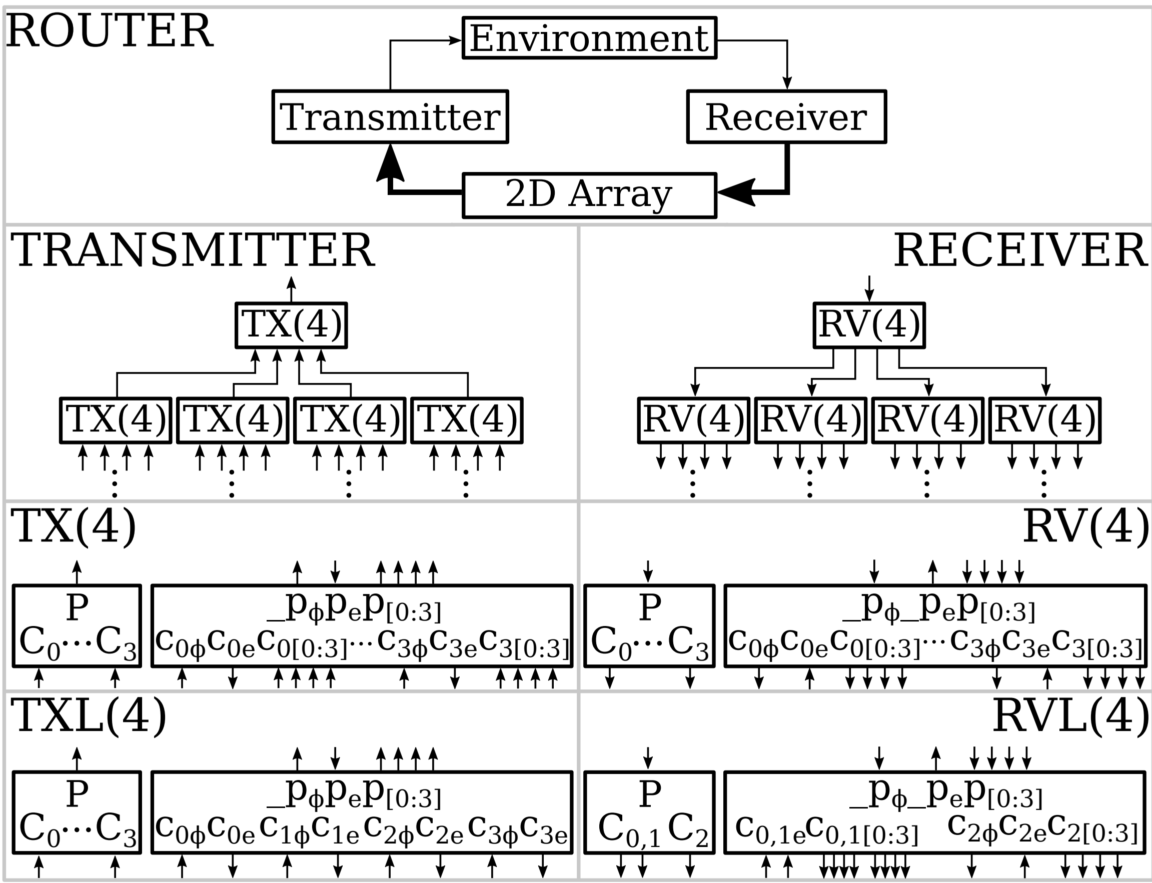


Figure 3.3: Router Process Decomposition

ROUTER: Facilitates communication between clients tiled in a 2D array and an external environment using a transmitter and a receiver. TRANSMITTER and RECEIVER: A pair of 4-ary trees provide an input and output port at their leaves for each client. TX(4) and RV(4): CHP ports (left) and HSE signals (right) that interface processes running in TRANSMITTER's and RECEIVER's nodes with their environment. TXL(4) and RVL(4): Same as previous but for processes in the leaves.

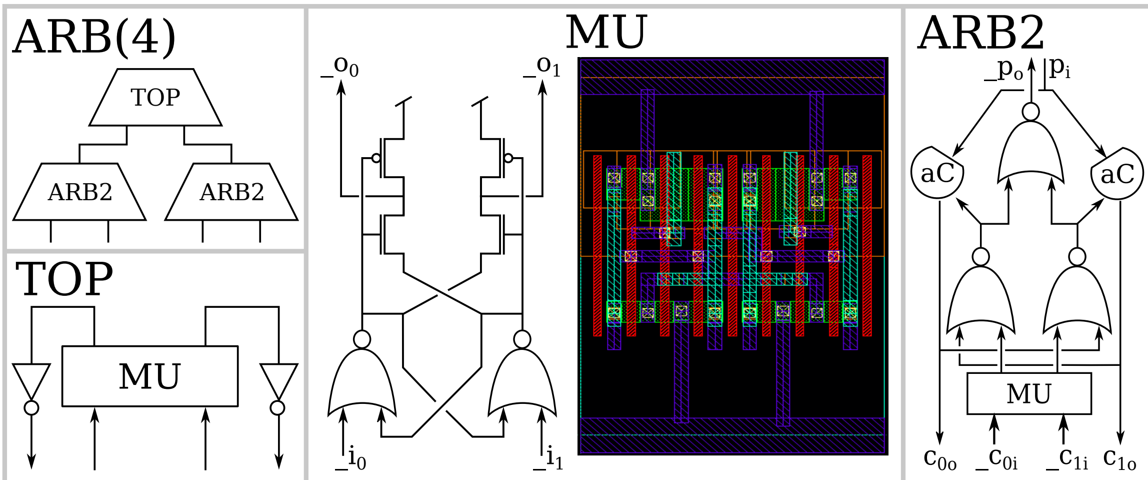


Figure 3.4: Four-Way Arbiter

ARB(4): Selects one of four clients with one TOP and two ARB2s; k clients require $k-2$ ARB2s connected in a binary tree. TOP: Performs two-way selection with a MU. MU: Selects one of two active-low (indicated by underscore prefix) inputs ($\_i_0$ and $\_i_1$) using cross-coupled NOR gates. Four additional transistors filter out metastable signals before toggling the outputs ($\_o_0$ and $\_o_1$). MU's custom standard-cell layout is shown. ARB2: Relays its childrens' requests to its parent and relays its parent's grant to a requesting child, selected beforehand by MU. The two, lower NOR gates ensure that handshakes on $c_{0\mathrm{i,o}}$ and $c_{1\mathrm{i,o}}$ do not overlap; aC are asymmetric C-elements.

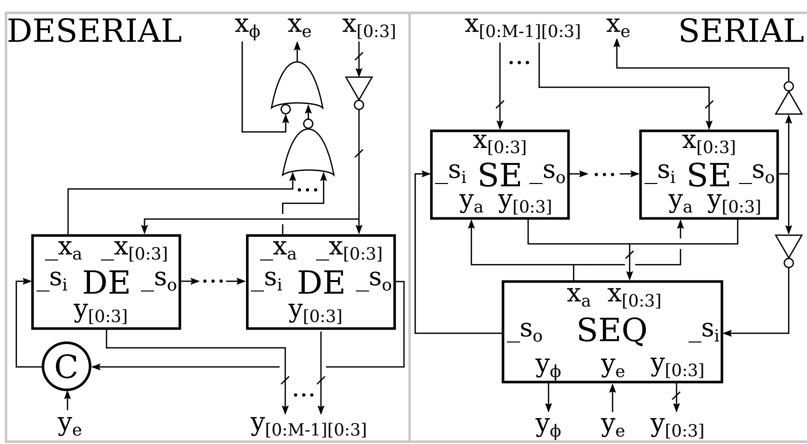


Figure 3.5: Serial–Parallel Conversion

DESERIAL: Serial input fans out to a chain of M DEs. An event moves from one to the next with each serial input; it loops back around through the C-element when parallel output occurs. SERIAL: Parallel input is divided among a chain of M SEs. As an event moves from one to the next, it outputs its data to SEQ. The event loops back around through SEQ when serial transmission is complete.

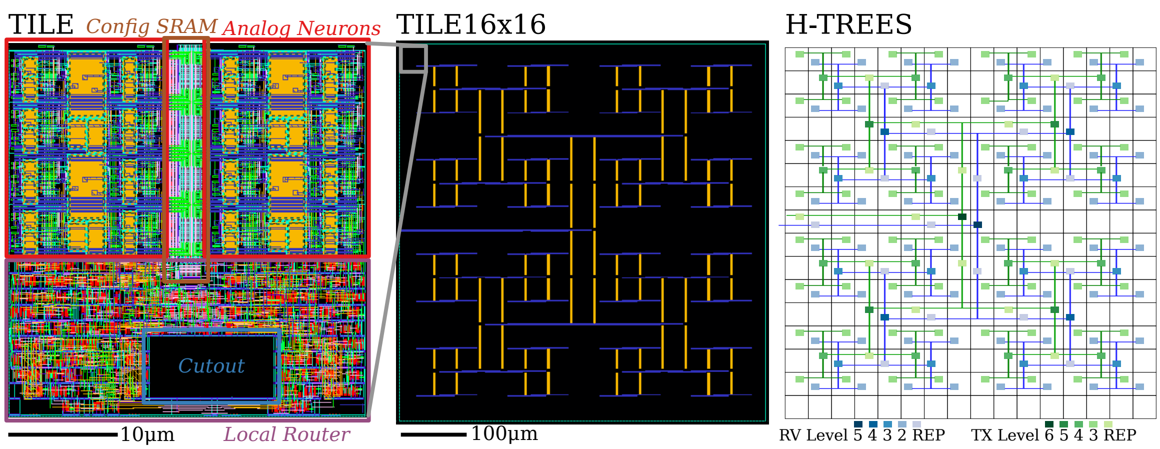


Figure 3.6: Tree Router Layout

TILE: Neuron cluster and low-level, local router circuits. *Analog Neurons*: Circuitry for 16 somas and 4 synapses. *Config SRAM*: Sixty-four 2-bit words—tiled in 8 rows and 8 columns—for analog circuitry configuration. *Local Router*: Four transmitter-tree leaves, their parent, a receiver-tree leaf, and a deserializer (SRAM interface). *Cutout*: Populated as needed with the transmitter or receiver trees' higher-level nodes or repeaters. TILE16×16: Full 2D array. Digital signals enter and exit on its left side, where the datapath is attached. To minimize crosstalk with analog circuitry, H-tree wires runs over the TILEs' *Local Router*s and *Config SRAM*s. A one-tile horizontal displacement between the two H-trees makes wiring possible with just metal layers 5 (yellow) and 6 (purple). H-TREES: Placement of two H-trees' higher-level nodes and repeaters (green for transmitter and blue for receiver) in tiles (white squares) with routing overlaid. TX: Transmitter; RV: Receiver; REP: Repeater.

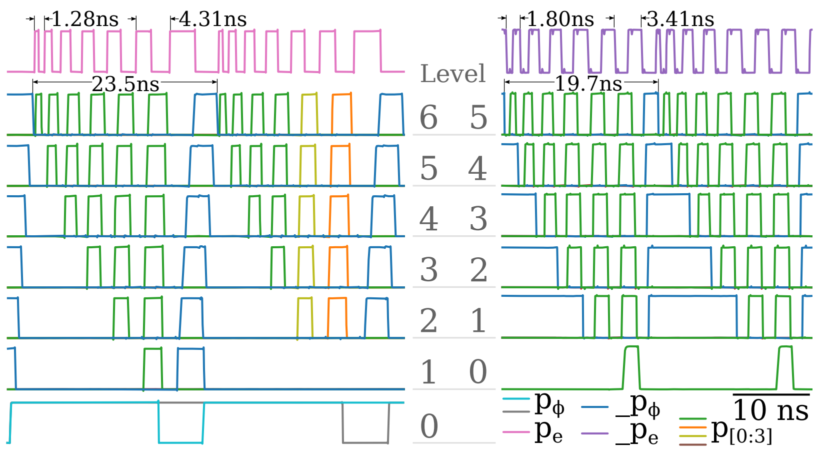


Figure 3.7: Postlayout Transmitter and Receiver SPICE Simulations
Left: SOMA 0 and 9 (000000 and 000021 in 4-ary) spike simultaneously (Level 0). Their p_ϕ signals propagate up to Level 1 (only SOMA 0's parent is shown) and Level 2, where SOMA 0's subtree is selected. Thus, its p_ϕ signal propagates to the root (Level 6). Enabled by the environment's p_e signal (top), which propagates down the tree (not shown), each node forwards its requesting child's 1-of-4 coded index ($p_{0:3}$) and then forwards indices forwarded by the child (they are all 0 for SOMA 0). Each node clears its $\_p_\phi$ once its child clears its $\_p_\phi$, signaling that there are no more indices to be forwarded. A node is then free to select another requesting subtree, as happens at Level 2 for SOMA 9's subtree. *Right*: The environment sends two inhibitory spikes to SYNAPSE 0 by injecting two packets containing its path appended with 0 (i.e., 000000) at the root. Each node selects the child indexed by the first 1-of-4 code ($p_{0:3}$) and forwards the remaining codes to that child after lowering $\_p_\phi$. Note that the leaf (Level 1) does not propagate $\_p_\phi$ to the synapse (Level 0).

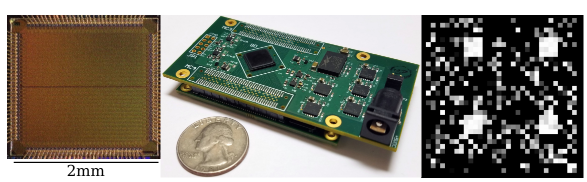


Figure 3.8: Fabrication and validation
Left: Test chip containing analog neurons, the router presented herein, and a digital datapath. *Center*: Test board piggybacked on an FPGA development board (Opal Kelly) that provides a USB link to a host computer. *Right*: Visualization of a 32×32-soma patch of the chip's spiking activity. Each small square represents a soma; its brightness reflects the soma's spike rate. The four, bright soma clusters are receiving excitatory input from spikes delivered to nearby synapses.

4

Conclusions

In this thesis, I have described the theoretical underpinnings for Braindrop's accumulative hardware for spike-train weighting and summating and have detailed the physical router hardware for communicating spikes as well as programming packets to and from Braindrop's neuron array. With its completion, Braindrop affords neuromorphic engineers both a sufficient number of neurons for nontrivial tasks and, different from Neurogrid, a well-mapped, systematic means of configuring those neurons. It is my hope that Braindrop enbables neuromorphic engineers to convincingly demonstrate to the wider world that the brain still has something to teach us about computing.

Appendix A

Spike Summing and Weighting

A.1 periodic SNR approximation

For $\lambda\tau \to \infty$, $\mathrm{SNR}_{\mathrm{periodic}} \to 2\sqrt{3}\lambda\tau$. Seeing this is not so straightforward since $\lim_{\lambda\tau\to\infty} \coth\left(\frac{1}{2\lambda\tau}\right)$ is not defined, so we move forward using Taylor series approximations. Recalling that $\coth\left(\frac{1}{2\lambda\tau}\right) = \left(1+e^{-1/\lambda\tau}\right)\left(1-e^{-1/\lambda\tau}\right)^{-1}$,

$$
\begin{aligned}
\mathrm{SNR}(X) &= \sqrt{\frac{2\lambda\tau}{\frac{1+e^{-1/\lambda\tau}}{1-e^{-1/\lambda\tau}} - 2\lambda\tau}} = \sqrt{\frac{2\lambda\tau}{\frac{1+1-\frac{1}{\lambda\tau}+\frac{1}{2}\frac{1}{(\lambda\tau)^2}-\frac{1}{6}\frac{1}{(\lambda\tau)^3}+\dots}{1-1+\frac{1}{\lambda\tau}-\frac{1}{2}\frac{1}{(\lambda\tau)^2}+\frac{1}{6}\frac{1}{(\lambda\tau)^3}+\dots} - 2\lambda\tau}} \\
&= \sqrt{\frac{1}{\frac{1}{2\lambda\tau}\frac{2-\frac{1}{\lambda\tau}+\frac{1}{2}\frac{1}{(\lambda\tau)^2}-\frac{1}{6}\frac{1}{(\lambda\tau)^3}+\dots}{\frac{1}{\lambda\tau}-\frac{1}{2}\frac{1}{(\lambda\tau)^2}+\frac{1}{6}\frac{1}{(\lambda\tau)^3}+\dots} - 1}} \\
&= \sqrt{\frac{1}{\frac{2-\frac{1}{\lambda\tau}+\frac{1}{2}\frac{1}{(\lambda\tau)^2}-\frac{1}{6}\frac{1}{(\lambda\tau)^3}+\dots}{2-\frac{1}{\lambda\tau}+\frac{1}{3}\frac{1}{(\lambda\tau)^2}-\frac{1}{12}\frac{1}{(\lambda\tau)^3}+\dots} - 1}}
\end{aligned}
$$

The Taylor series approximations begin to differ with the $\frac{1}{(\lambda\tau)^2}$ coefficients; we drop higher order terms that converge to 0 much faster.

$$\mathrm{SNR}(X) \underset{\lambda\tau\to\infty}{=} \sqrt{\frac{1}{\frac{2-\frac{1}{\lambda\tau}+\frac{1}{2}\frac{1}{(\lambda\tau)^2}}{2-\frac{1}{\lambda\tau}+\frac{1}{3}\frac{1}{(\lambda\tau)^2}}-1}} = \sqrt{\frac{1}{\frac{2-\frac{1}{\lambda\tau}+\frac{1}{3}\frac{1}{(\lambda\tau)^2}+\frac{1}{6}\frac{1}{(\lambda\tau)^2}}{2-\frac{1}{\lambda\tau}+\frac{1}{3}\frac{1}{(\lambda\tau)^2}}-1}}$$

$$= \sqrt{\frac{1}{1+\frac{\frac{1}{6}\frac{1}{(\lambda\tau)^2}}{2-\frac{1}{\lambda\tau}+\frac{1}{3}\frac{1}{(\lambda\tau)^2}}-1}} \underset{\lambda\tau\to\infty}{=} \sqrt{\frac{1}{\frac{\frac{1}{6}\frac{1}{(\lambda\tau)^2}}{2}}} = 2\sqrt{3}\lambda\tau$$

A.2 p-thinning SNR approximation

As p tends to 0, the p-thinned periodic SNR tends towards the Poisson SNR. As before, we Taylor series expand coth and find where terms begin to differ to find $\lim_{p\to 0}\mathrm{SNR}(X)$.

$$\lim_{p\to 0}\mathrm{SNR}(X) = \lim_{p\to 0}\sqrt{\frac{2\lambda\tau}{1+p\frac{1+\left(1-\frac{p}{\lambda\tau}+\frac{1}{2}\left(\frac{p}{\lambda\tau}\right)^2-\frac{1}{6}\left(\frac{p}{\lambda\tau}\right)^3+\ldots\right)}{1-\left(1-\frac{p}{\lambda\tau}+\frac{1}{2}\left(\frac{p}{\lambda\tau}\right)^2-\frac{1}{6}\left(\frac{p}{\lambda\tau}\right)^3+\ldots\right)}-2\lambda\tau}}$$

$$= \lim_{p\to 0}\sqrt{\frac{2\lambda\tau}{1+2\lambda\tau\frac{2-\frac{p}{\lambda\tau}+\frac{1}{2}\left(\frac{p}{\lambda\tau}\right)^2-\frac{1}{6}\left(\frac{p}{\lambda\tau}\right)^3+\ldots}{2-\frac{p}{\lambda\tau}+\frac{1}{3}\left(\frac{p}{\lambda\tau}\right)^2-\frac{1}{12}\left(\frac{p}{\lambda\tau}\right)^3-\ldots}-2\lambda\tau}}$$

$$= \lim_{p\to 0}\sqrt{\frac{2\lambda\tau}{1+2\lambda\tau\frac{2-\frac{p}{\lambda\tau}+\frac{1}{3}\left(\frac{p}{\lambda\tau}\right)^2+\frac{1}{6}\left(\frac{p}{\lambda\tau}\right)^2}{2-\frac{p}{\lambda\tau}+\frac{1}{3}\left(\frac{p}{\lambda\tau}\right)^2}-2\lambda\tau}}$$

$$= \lim_{p\to 0}\sqrt{\frac{2\lambda\tau}{1+2\lambda\tau\left(1+\frac{\frac{1}{6}\left(\frac{p}{\lambda\tau}\right)^2}{2-\frac{p}{\lambda\tau}+\frac{1}{3}\left(\frac{p}{\lambda\tau}\right)^2}\right)-2\lambda\tau}} = \sqrt{2\lambda\tau}$$

A.3 d-thinning SNR expansion

We express $\mathrm{SNR}_{\mathrm{dthin}}(X)$ in terms of $\mathrm{SNR}_{\mathrm{poi}}(X)$, by considering $\mathrm{SNR}_{\mathrm{dthin}}(X) = \mathrm{SNR}_{\mathrm{poi}}(X)/g$ and expanding binomials:

$$g^2 = \frac{(1+k\lambda\tau)^k+(k\lambda\tau)^k}{(1+k\lambda\tau)^k-(k\lambda\tau)^k} - 2\lambda\tau = \frac{\sum_{j=0}^{k}\binom{k}{j}(k\lambda\tau)^j+(k\lambda\tau)^k}{\sum_{j=0}^{k}\binom{k}{j}(k\lambda\tau)^j-(k\lambda\tau)^k} - 2\lambda\tau$$

$$= \frac{1+(k^2-2)\lambda\tau+\ldots+\frac{1}{6}k^{k-2}(k-1)(k+4)(\lambda\tau)^{k-2}+k^{k-1}(\lambda\tau)^{k-1}}{1+k^2\lambda\tau+\ldots+\frac{1}{2}k^{k-1}(k-1)(\lambda\tau)^{k-2}+k^k(\lambda\tau)^{k-1}}$$

By taking a fourth-order Taylor series approximation in the denominator of $\mathrm{SNR}_{\mathrm{dthin}}$,

$$\begin{aligned}\mathrm{SNR}(X) &\approx \sqrt{\frac{2\lambda\tau}{1 + \frac{k^4-10k^2+9}{15(2\lambda\tau)^4} + \frac{-k^4+20k^2-19}{45(2\lambda\tau)^3} + \frac{k^2-1}{3(2\lambda\tau)^2} + \frac{k^2-1}{3(2\lambda\tau)}}} \\ &\approx \sqrt{2\lambda\tau/(1+k^2/3(2\lambda\tau))} \qquad \text{assuming } k \gg 1 \text{ and } \lambda\tau \gg 1\end{aligned}$$

Appendix B

AER Transmitter Design Space

This appendix explores the router's AER transmitter (AEXT) design space. From the implemented transmitter design described in Section 3.4.1, the designs described here are earlier iterations and listed in approximately reverse chronological order.

B.1 AEXT Control Data decomposed (CD)

In this design, the control and data are separated; there is a control tree and a data tree (cf. AEXT ASPR and AEXT PSAR with combined control and data).

B.1.1 AEXT CD noTW CYC

The transmitter (AEXT) control-data decomposed (CD) without tailword (noTW) cyclic signaling (CYC) design make more efficient use of the control signaling (relative to AEXT CD noTW) by doing away with the data enable/acknowledge entirely.
Radix 2 accounting (2047 intermediate nodes, 2048 leaf nodes):

intermediate nodes			
component	transistors/component	components/node	transistors/node
NODE	90	1	
		total transistors/intermediate node	90

leaf nodes			
component	transistors/component	components/node	transistors/node
LEAF	74	1	74
		total transistors/leaf node	74

(90 transistors/intermediate node * 2047 intermediate nodes + 74 transistors/leaf node * 2048 leaf nodes) / 4096 neurons = **82.0 transistors/neuron**

Radix 4 accounting (341 intermediate nodes, 1024 leaf nodes):

intermediate nodes			
component	transistors/component	components/node	transistors/node
NODE	274	1	274
		total transistors/intermediate node	274
leaf nodes			
component	transistors/component	components/node	transistors/node
LEAF	218	1	218
		total transistors/leaf node	218

(274 transistors/intermediate node * 341 intermediate nodes + 218 transistors/leaf node * 1024 leaf nodes) / 4096 neurons = **77.3 transistors/neuron**

AEXT CD noTW CYC NODE

Intermediate node of AEXT tree.

$$
\begin{array}{l}
*[[c0 \longrightarrow po\uparrow; [pi];\\
\qquad\qquad w0\uparrow; [\neg pi]; u\uparrow; w0\downarrow; [pi];\\
\qquad\qquad c0o\uparrow; [\neg c0]; po\downarrow; [\neg pi]; c0o\downarrow; u\downarrow\\
\quad [\!] c1 \longrightarrow po\uparrow; [pi];\\
\qquad\qquad w1\uparrow; [\neg pi]; u\uparrow; w1\downarrow; [pi];\\
\qquad\qquad c1o\uparrow; [\neg c1]; po\downarrow; [\neg pi]; c1o\downarrow; u\downarrow\\
\ ]]
\end{array}
$$

$$
\begin{array}{l}
*[[c00 \vee c10 \vee w0 \longrightarrow p0\uparrow; [\neg pi]; c0o\downarrow; [\neg c00 \wedge \neg c10 \wedge \neg w0]; p0\downarrow; [pi \wedge c0]; c0o\uparrow\\
\quad [\!] c01 \vee c11 \vee w1 \longrightarrow p1\uparrow; [\neg pi]; c1o\downarrow; [\neg c01 \wedge \neg c10 \wedge \neg w1]; p1\downarrow; [pi \wedge c1]; c1o\uparrow\\
\ ]]
\end{array}
$$

It's helpful to consider the projection of the HSE on to the parent control and data lines.

$$
\begin{array}{l}
*[po\uparrow; [pi];\\
\qquad [\neg pi]; [pi];\\
\qquad (P\uparrow; [\neg pi]; P\downarrow; [pi]) \times (m-1)\\
\quad po\downarrow; [\neg pi]\\
\ ]
\end{array}
$$

The first line propagates the child request up the tree and waits for the parents to acknowledge.
The second line is the node outputting a new head word.
The third line repeats $(m - 1)$ times where m is this node's level in the tree.
The fourth line propagates the child reset up the tree.

$$\begin{array}{lcl} \neg u \wedge (c0 \vee c1) & \rightarrow & po\uparrow \\ (c0o \wedge \neg c0) \vee (c1o \wedge \neg c1) & \rightarrow & po\downarrow \end{array}$$

$$\begin{array}{lcl} c0 \wedge pi \wedge \neg u & \rightarrow & w0\uparrow \\ u & \rightarrow & w0\downarrow \end{array} \qquad \begin{array}{lcl} c1 \wedge pi \wedge \neg u & \rightarrow & w1\uparrow \\ u & \rightarrow & w1\downarrow \end{array}$$

$$\begin{array}{lcl} (w0 \vee w1) \wedge \neg pi & \rightarrow & u\uparrow \\ \neg c0o \wedge \neg c1o \wedge \neg po & \rightarrow & u\downarrow \end{array}$$

$$\begin{array}{lcl} c0 \wedge u \wedge pi \wedge \neg c1o & \rightarrow & c0o\uparrow \\ \neg pi & \rightarrow & c0o\downarrow \end{array} \qquad \begin{array}{lcl} c1 \wedge u \wedge pi \neg c0o & \rightarrow & c1o\uparrow \\ \neg pi & \rightarrow & c1o\downarrow \end{array}$$

$$\begin{array}{lcl} c00 \vee c10 \vee w0 & \rightarrow & p0\uparrow \\ \neg c00 \wedge \neg c10 \wedge \neg w0 & \rightarrow & p0\downarrow \end{array} \qquad \begin{array}{lcl} c01 \vee c11 \vee w1 & \rightarrow & p1\uparrow \\ \neg c01 \wedge \neg c11 \wedge \neg w1 & \rightarrow & p1\downarrow \end{array}$$

Radix 2 transistor approximate accounting:

rule	transistor count	comments
$c[0,1]$	12	2-way arbiter
p_o	11	
$w[0,1]$	16	
u	10	
$c[0,1]_o$	18	
$p[0,1]$	12	
total	79	

Radix 4 transistor approximate accounting:

rule	transistor count	comments
$c[0,1,2,3]$	92	4-way unpipelined arbiter
p_o	19	
$w[0,1,2,3]$	32	
u	10	
$c[0,1,2,3]_o$	44	
$p[0,1,2,3]$	40	
total	237	

CMOS-implementable PRS

$$
\begin{array}{lcl@{\qquad\qquad}lcl}
\_u \wedge (c0 \vee c1) & \rightarrow & \_po\downarrow & \neg\_po & \rightarrow & po\uparrow \\
(\neg\_c0o \wedge \neg c0) \vee (\neg\_c1o \wedge \neg c1) & \rightarrow & \_po\uparrow & \_po & \rightarrow & po\downarrow
\end{array}
$$

$$
\begin{array}{lcl@{\qquad\qquad}lcl}
c0 \wedge pi \wedge \_u & \rightarrow & \_w0\downarrow & c1 \wedge pi \wedge \_u & \rightarrow & \_w1\downarrow \\
\neg\_u & \rightarrow & \_w0\uparrow & \neg\_u & \rightarrow & \_w1\uparrow
\end{array}
$$

$$
\begin{array}{lcl@{\qquad\qquad}lcl}
(\neg\_w0 \vee \neg\_w1) \wedge \neg pi & \rightarrow & u\uparrow & \neg u & \rightarrow & \_u\uparrow \\
\_c0o \wedge \_c1o \wedge \_po & \rightarrow & u\downarrow & u & \rightarrow & \_u\downarrow
\end{array}
$$

$$
\begin{array}{lcl@{\qquad\qquad}lcl}
c0 \wedge u \wedge pi \wedge \_c1o & \rightarrow & \_c0o\downarrow & c1 \wedge u \wedge pi\_c0o & \rightarrow & \_c1o\downarrow \\
\neg pi & \rightarrow & \_c0o\uparrow & \neg pi & \rightarrow & \_c1o\uparrow
\end{array}
$$

$$
\begin{array}{lcl@{\qquad\qquad}lcl}
\neg\_c0o & \rightarrow & c0o\uparrow & \neg\_c1o & \rightarrow & c1o\uparrow \\
\_c0o & \rightarrow & c0o\downarrow & \_c1o & \rightarrow & c1o\downarrow
\end{array}
$$

$$
\begin{array}{lcl@{\qquad\qquad}lcl}
\neg\_c00 \vee \neg\_c10 \vee \neg\_w0 & \rightarrow & p0\uparrow & \neg\_c01 \vee \neg\_c11 \vee \neg\_w1 & \rightarrow & p1\uparrow \\
\_c00 \wedge \_c10 \wedge \_w0 & \rightarrow & p0\downarrow & \_c01 \wedge \_c11 \wedge \_w1 & \rightarrow & p1\downarrow
\end{array}
$$

$$
\begin{array}{lcl@{\qquad\qquad}lcl}
\neg p0 & \rightarrow & \_p0\uparrow & \neg p1 & \rightarrow & \_p1\uparrow \\
p0 & \rightarrow & \_p0\downarrow & p1 & \rightarrow & \_p1\downarrow
\end{array}
$$

Note that the root NODE does not create $\_p[0,1]$. We simply present a normal-sense p_i, p_o, and $p[0,1]$ interface to the environment.

We could make another another version of CMOS-implementable PRS to alternate with this version and eliminate the inverters creating p_o, $c[0,1]_o$, and $_p[0,1]$. However, that version would have a PMOS pull up chain for $c[0,1]_o$ that doesn't scale. With a radix 2 tree the chain is already 4 transistors long. With a radix 4 tree the chain is 6 transitors long. We want to keep PMOS chains 3 long or shorter.

Radix 2 transistor accounting:

rule	transistor count	comments
$c[0,1]$	12	2-way arbiter
$\_p_o$	11	
p_o	2	
$\_w[0,1]$	16	
u	10	
$\_u$	2	
$\_c[0,1]_o$	18	
$c[0,1]_o$	4	
$p[0,1]$	12	
$\_p[0,1]$	4	
total	91	

Radix 4 transistor accounting:

rule	transistor count	comments
$c[0,1,2,3]$	92	4-way unpipelined arbiter
$\_p_o$	19	
p_o	2	
$\_w[0,1,2,3]$	32	
u	10	
$\_u$	2	
$\_c[0,1,2,3]_o$	44	
$c[0,1,2,3]_o$	8	
$p[0,1,2,3]$	40	
$\_p[0,1,2,3]$	8	
total	257	

AEXT CD noTW CYC NODE (reference implementation)

Intermediate node of AEXT tree

$$
\begin{array}{l}
*[[c0 \longrightarrow q0\uparrow; po\uparrow; [pi];\\
\qquad w0\uparrow; [\neg pi]; u\uparrow; w0\downarrow; [pi];\\
\qquad c0o\uparrow; [\neg c0]; q0\downarrow; po\downarrow; u\downarrow; [\neg pi]; c0o\downarrow\\
\;[]\, c1 \longrightarrow q1\uparrow; po\uparrow; [pi];\\
\qquad w1\uparrow; [\neg pi]; u\uparrow; w1\downarrow; [pi];\\
\qquad c1o\uparrow; [\neg c1]; q1\downarrow; po\downarrow; u\downarrow; [\neg pi]; c1o\downarrow\\
\;]]
\end{array}
$$

$$
\begin{array}{l}
*[[c00 \lor c10 \lor w0 \longrightarrow p0\uparrow; [\neg pi]; c0o\downarrow; [\neg c00 \land \neg c10 \land \neg w0]; p0\downarrow; [pi \land q0]; c0o\uparrow\\
\;\;[]\, c01 \lor c11 \lor w1 \longrightarrow p1\uparrow; [\neg pi]; c1o\downarrow; [\neg c01 \land \neg c10 \land \neg w1]; p1\downarrow; [pi \land q1]; c1o\uparrow\\
\;]]
\end{array}
$$

$$
\begin{array}{lcl@{\qquad}lcl}
c0 \land \neg c1o & \to & q0\uparrow & c1 \land \neg c0o & \to & q1\uparrow\\
\neg c0 \land c0o & \to & q0\downarrow & \neg c1 \land c1o & \to & q1\downarrow
\end{array}
$$

$$
\begin{array}{lcl}
q0 \lor q1 & \to & po\uparrow\\
\neg q0 \land \neg q1 & \to & po\downarrow
\end{array}
$$

$$
\begin{array}{lcl@{\qquad}lcl}
q0 \land pi \land \neg u & \to & w0\uparrow & q1 \land pi \land \neg u & \to & w1\uparrow\\
u & \to & w0\downarrow & u & \to & w1\downarrow
\end{array}
$$

$$
\begin{array}{lcl}
(w0 \lor w1) \land \neg pi & \to & u\uparrow\\
\neg po & \to & u\downarrow
\end{array}
$$

$$
\begin{array}{lcl@{\qquad}lcl}
q0 \land u \land pi & \to & c0o\uparrow & \neg q1 \land u \land pi & \to & c1o\uparrow\\
(\neg u \lor p0 \lor p1) \land \neg pi & \to & c0o\downarrow & (\neg u \lor p0 \lor p1) \land \neg pi & \to & c1o\downarrow
\end{array}
$$

$$
\begin{array}{lcl@{\qquad}lcl}
c00 \lor c10 \lor w0 & \to & p0\uparrow & c01 \lor c11 \lor w1 & \to & p1\uparrow\\
\neg c00 \land \neg c10 \land \neg w0 & \to & p0\downarrow & \neg c01 \land \neg c11 \land \neg w1 & \to & p1\downarrow
\end{array}
$$

Radix 2 transistor accounting:

rule	transistor count	comments
$c[0,1]$	12	2-way arbiter
$q[0,1]$	16	
p_o	4	
$w[0,1]$	16	
u	8	
$c[0,1]_o$	22	
$p[0,1]$	12	
total	90	

Radix 4 transistor accounting:

rule	transistor count	comments
$c[0,1,2,3]$	92	4-way unpipelined arbiter
$q[0,1,2,3]$	40	
p_o	8	
$w[0,1,2,3]$	32	
u	10	
$c[0,1,2,3]_o$	52	
$p[0,1,2,3]$	40	
total	274	

Radix 2 transistor accounting:

AEXT CD noTW CYC LEAF

Leaf node of AEXT tree

$$
\begin{array}{l}
*[[c0 \longrightarrow po{\uparrow};\, [pi];\\
\qquad p0{\uparrow};\, [\neg pi];\, u{\uparrow};\, p0{\downarrow};\, [pi];\\
\qquad c0o{\uparrow};\, [\neg c0];\, po{\downarrow};\, [\neg pi];\, c0o{\downarrow};\, u{\downarrow}\\
\;[]\, c1 \longrightarrow po{\uparrow};\, [pi];\\
\qquad p1{\uparrow};\, [\neg pi];\, u{\uparrow};\, p1{\downarrow};\, [pi];\\
\qquad c1o{\uparrow};\, [\neg c1];\, po{\downarrow};\, [\neg pi];\, c1o{\downarrow};\, u{\downarrow}\\
\;]]
\end{array}
$$

PRS

$$
\begin{array}{lcl}
\neg u \wedge c0 \vee c1 & \rightarrow & po{\uparrow}\\
(c0o \wedge \neg c0) \vee (c1o \wedge \neg c1) & \rightarrow & po{\downarrow}
\end{array}
$$

$$
\begin{array}{lcl}
c0 \wedge pi \wedge \neg u & \rightarrow & p0{\uparrow}\\
u & \rightarrow & p0{\downarrow}
\end{array}
\qquad
\begin{array}{lcl}
c1 \wedge pi \wedge \neg u & \rightarrow & p1{\uparrow}\\
u & \rightarrow & p1{\downarrow}
\end{array}
$$

$$
\begin{array}{lcl}
(p0 \vee p1) \wedge \neg pi & \rightarrow & u{\uparrow}\\
\neg c0o \wedge \neg c1o \wedge \neg po & \rightarrow & u{\downarrow}
\end{array}
$$

$$
\begin{array}{lcl}
c0 \wedge u \wedge pi \wedge \neg c1o & \rightarrow & c0o{\uparrow}\\
\neg u \wedge \neg pi & \rightarrow & c0o{\downarrow}
\end{array}
\qquad
\begin{array}{lcl}
c1 \wedge u \wedge pi \wedge \neg c0o & \rightarrow & c1o{\uparrow}\\
\neg u \wedge \neg pi & \rightarrow & c1o{\downarrow}
\end{array}
$$

Radix 2 transistor approximate accounting:

rule	transistor count	comments
$c[0,1]$	12	2-way arbiter
p_o	11	
$p[0,1]$	16	
u	10	
$c[0,1]_o$	20	
total	69	

Radix 4 transistor approximate accounting:

rule	transistor count	comments
$c[0,1,2,3]$	92	4-way unpipelined arbiter
p_o	17	
$p[0,1,2,3]$	32	
u	14	
$c[0,1,2,3]_o$	48	
total	203	

CMOS-implementable PRS

$$\begin{array}{lll} \_u \wedge c0 \vee c1 & \rightarrow \_po{\downarrow} & \neg\_po \rightarrow po{\uparrow} \\ (\neg\_c0o \wedge \neg c0) \vee (\neg\_c1o \wedge \neg c1) & \rightarrow \_po{\uparrow} & \_po \rightarrow po{\downarrow} \end{array}$$

$$\begin{array}{ll} c0 \wedge pi \wedge \_u \rightarrow \_p0{\downarrow} & c1 \wedge pi \wedge \_u \rightarrow \_p1{\downarrow} \\ \neg\_u \rightarrow \_p0{\uparrow} & \neg\_u \rightarrow \_p1{\downarrow} \end{array}$$

$$\begin{array}{ll} (\neg\_p0 \vee \neg\_p1) \wedge \neg pi \rightarrow u{\uparrow} & \neg u \rightarrow \_u{\uparrow} \\ \_c0o \wedge \_c1o \wedge \_po \rightarrow u{\downarrow} & u \rightarrow \_u{\downarrow} \end{array}$$

$$\begin{array}{ll} c0 \wedge u \wedge pi \wedge \_c1o \rightarrow \_c0o{\downarrow} & \neg\_c0o \rightarrow c0o{\uparrow} \\ \neg u \wedge \neg pi \rightarrow \_c0o{\uparrow} & \_c0o \rightarrow c0o{\downarrow} \end{array}$$

$$\begin{array}{ll} c1 \wedge u \wedge pi \wedge \_c0o \rightarrow \_c1o{\downarrow} & \neg\_c1o \rightarrow c1o{\uparrow} \\ \neg u \wedge \neg pi \rightarrow \_c1o{\uparrow} & \_c1o \rightarrow c1o{\downarrow} \end{array}$$

Radix 2 transistor accounting:

rule	transistor count	comments
$c[0,1]$	12	2-way arbiter
$\_p_o$	11	
p_o	2	
$\_p[0,1]$	16	
u	10	
$\_u$	2	
$\_c[0,1]_o$	20	
$c[0,1]_o$	4	
total	77	

Radix 4 transistor accounting:

rule	transistor count	comments
$c[0,1,2,3]$	92	4-way unpipelined arbiter
$\_p_o$	17	
p_o	2	
$\_p[0,1,2,3]$	32	
u	14	
$\_u$	2	
$\_c[0,1,2,3]_o$	48	
$c[0,1,2,3]_o$	8	
total	215	

AEXT CD noTW CYC LEAF (reference implementation)

Leaf node of AEXT tree

$$
\begin{array}{l}
*[[c0 \longrightarrow q0\uparrow; po\uparrow; [pi];\\
\qquad p0\uparrow; [\neg pi]; u\uparrow; p0\downarrow; [pi];\\
\qquad c0o\uparrow; [\neg c0]; q0\downarrow; po\downarrow; u\downarrow; [\neg pi]; c0o\downarrow\\
\;[]\, c1 \longrightarrow q1\uparrow; po\uparrow; [pi];\\
\qquad p1\uparrow; [\neg pi]; u\uparrow; p1\downarrow; [pi];\\
\qquad c1o\uparrow; [\neg c1]; q1\downarrow; po\downarrow; u\downarrow; [\neg pi]; c1o\downarrow\\
\;]]
\end{array}
$$

$$
\begin{array}{lcllcl}
c0 \wedge \neg c1o & \rightarrow & q0\uparrow & \qquad c1 \wedge \neg c0o & \rightarrow & q1\uparrow\\
\neg c0 \wedge c0o & \rightarrow & q0\downarrow & \qquad \neg c1 \wedge c1o & \rightarrow & q1\downarrow
\end{array}
$$

$$
\begin{array}{lcl}
q0 \vee q1 & \rightarrow & po\uparrow\\
\neg q0 \wedge \neg q1 & \rightarrow & po\downarrow
\end{array}
$$

$$
\begin{array}{lcl}
q0 \wedge pi \wedge \neg u & \rightarrow & p0\uparrow \\
u & \rightarrow & p0\downarrow
\end{array}
\qquad
\begin{array}{lcl}
q1 \wedge pi \wedge \neg u & \rightarrow & p1\uparrow \\
u & \rightarrow & p1\downarrow
\end{array}
$$

$$
\begin{array}{lcl}
(p0 \vee p1) \wedge \neg pi & \rightarrow & u\uparrow \\
\neg po & \rightarrow & u\downarrow
\end{array}
$$

$$
\begin{array}{lcl}
q0 \wedge u \wedge pi & \rightarrow & c0o\uparrow \\
\neg u \wedge \neg pi & \rightarrow & c0o\downarrow
\end{array}
\qquad
\begin{array}{lcl}
q1 \wedge u \wedge pi & \rightarrow & c1o\uparrow \\
\neg u \wedge \neg pi & \rightarrow & c1o\downarrow
\end{array}
$$

Radix 2 transistor accounting:

rule	transistor count	comments
$c[0,1]$	12	2-way arbiter
$q[0,1]$	16	
p_o	4	
$p[0,1]$	16	
u	8	
$c[0,1]_o$	18	
total	74	

Radix 4 transistor accounting:

rule	transistor count	comments
$c[0,1,2,3]$	92	4-way unpipelined arbiter
$q[0,1,2,3]$	40	
p_o	8	
$p[0,1,2,3]$	32	
u	10	
$c[0,1,2,3]_o$	36	
total	218	

B.1.2 AEXT CD noTW

This design has no tail word.

Radix 2 accounting (2047 intermediate nodes, 2048 leaf nodes):

intermediate nodes			
component	transistors/component	components/node	transistors/node
CTRL	86	1	86
MERGE	28	1	28
total transistors/intermediate node			114
leaf nodes			
component	transistors/component	components/node	transistors/node
LEAF	64	1	64
total transistors/leaf node			64

(114 transistors/intermediate node * 2047 intermediate nodes + 64 transistors/leaf node * 2048 leaf nodes) / 4096 neurons = **89.0 transistors/neuron**

Radix 4 accounting (341 intermediate nodes, 1024 leaf nodes):

intermediate nodes			
component	transistors/component	components/node	transistors/node
CTRL	246	1	246
MERGE	60	1	60
total transistors/intermediate node			306
leaf nodes			
component	transistors/component	components/node	transistors/node
LEAF	204	1	204
total transistors/leaf node			204

(306 transistors/intermediate node * 341 intermediate nodes + 204 transistors/leaf node * 1024 leaf nodes) / 4096 neurons = **76.5 transistors/neuron**

AEXT CD noTW CTRL

$$
\begin{array}{l}
*[[C0 \bullet P; M; C0 \bullet P \\
\quad | \, C1 \bullet P; M; C1 \bullet P \\
\;]]
\end{array}
$$

$$
\begin{array}{l}
*[[c0 \longrightarrow po\!\uparrow; [pi]; c0o\!\uparrow; \\
\qquad\qquad mw\!\uparrow; [we]; w0\!\uparrow; [\neg we]; w0\!\downarrow; mw\!\downarrow; \\
\qquad\qquad m0\!\uparrow; [\neg c0]; po\!\downarrow; [\neg pi]; c0o\!\downarrow; m0\!\downarrow \\
\,[]\, c1 \longrightarrow po\!\uparrow; [pi]; c1o\!\uparrow; \\
\qquad\qquad mw\!\uparrow; [we]; w1\!\uparrow; [\neg we]; w1\!\downarrow; mw\!\downarrow; \\
\qquad\qquad m1\!\uparrow; [\neg c1]; po\!\downarrow; [\neg pi]; c1o\!\downarrow; m1\!\downarrow \\
]]
\end{array}
$$

$$
\begin{array}{l}
\texttt{*[[}c0 \longrightarrow q0{\uparrow}; po{\uparrow}; \texttt{[}pi\texttt{]};\\
\qquad mw{\uparrow}; c0o{\uparrow}; \texttt{[}we\texttt{]}; w0{\uparrow}; \texttt{[}\neg we\texttt{]}; m0{\uparrow}; mw{\downarrow}; w0{\downarrow}\\
\qquad \texttt{[}\neg c0\texttt{]}; q0{\downarrow}; po{\downarrow}; \texttt{[}\neg pi\texttt{]}; m0{\downarrow}; c0o{\downarrow}\\
\texttt{[]}c1 \longrightarrow q1{\uparrow}; po{\uparrow}; \texttt{[}pi\texttt{]};\\
\qquad mw{\uparrow}; c1o{\uparrow}; \texttt{[}we\texttt{]}; w1{\uparrow}; \texttt{[}\neg we\texttt{]}; m1{\uparrow}; mw{\downarrow}; w1{\downarrow}\\
\qquad \texttt{[}\neg c1\texttt{]}; q1{\downarrow}; po{\downarrow}; \texttt{[}\neg pi\texttt{]}; m1{\downarrow}; c1o{\downarrow}\\
\texttt{]]}
\end{array}
$$

sequence

$$mx{\downarrow}; cxo{\downarrow}$$

required to lower control before releasing child to lower word line.

$$
\begin{array}{lcl}
c0 \wedge \neg m0 \wedge \neg c1o & \rightarrow & q0{\uparrow}\\
\neg c0 \wedge m0 \wedge \neg w0 & \rightarrow & q0{\downarrow}
\end{array}
\qquad
\begin{array}{lcl}
c1 \wedge \neg m1 \wedge \neg c0o & \rightarrow & q1{\uparrow}\\
\neg c1 \wedge m1 \wedge \neg w1 & \rightarrow & q1{\downarrow}
\end{array}
$$

$$
\begin{array}{lcl}
q0 \vee q1 & \rightarrow & po{\uparrow}\\
\neg q0 \wedge \neg q1 & \rightarrow & po{\downarrow}
\end{array}
$$

$$
\begin{array}{lcl}
q0 \wedge mw & \rightarrow & c0o{\uparrow}\\
\neg m0 \wedge \neg pi & \rightarrow & c0o{\downarrow}
\end{array}
\qquad
\begin{array}{lcl}
q1 \wedge mw & \rightarrow & c1o{\uparrow}\\
\neg m1 \wedge \neg pi & \rightarrow & c1o{\downarrow}
\end{array}
$$

$$
\begin{array}{lcl}
c0o \wedge we & \rightarrow & w0{\uparrow}\\
\neg mw & \rightarrow & w0{\downarrow}
\end{array}
\qquad
\begin{array}{lcl}
c1o \wedge we & \rightarrow & w1{\uparrow}\\
\neg mw & \rightarrow & w1{\downarrow}
\end{array}
$$

$$
\begin{array}{lcl}
pi \wedge \neg m0 \wedge \neg m1 & \rightarrow & mw{\uparrow}\\
\neg pi \vee m0 \vee m1 & \rightarrow & mw{\downarrow}
\end{array}
$$

$$
\begin{array}{lcl}
w0 \wedge \neg we & \rightarrow & m0{\uparrow}\\
\neg pi & \rightarrow & m0{\downarrow}
\end{array}
\qquad
\begin{array}{lcl}
w1 \wedge \neg we & \rightarrow & m1{\uparrow}\\
\neg pi & \rightarrow & m1{\downarrow}
\end{array}
$$

Radix 2 transistor accounting:

rule	transistor count	comments
$c[0,1]$	12	2-input arbiter
$q[0,1]$	20	
p_o	4	
$c[0,1]_o$	16	
$w[0,1]$	14	
mw	6	
$m[0,1]$	14	
total	86	

Radix 4 transistor accounting:

rule	transistor count	comments
$c[0,1,2,3]$	92	4-input arbiter
$q[0,1,2,3]$	48	
p_o	8	
$c[0,1,2,3]_o$	32	
$w[0,1,2,3]$	28	
mw	10	
$m[0,1,2,3]$	28	
total	246	

AEXT CD noTW MERGE

$*[[pe];$
$\quad [m0 \longrightarrow c0e\uparrow;$
$\quad\quad [c00 \longrightarrow p0\uparrow; [\neg pe]; c0e\downarrow; [\neg c00]; p0\downarrow$
$\quad\quad [\!] c01 \longrightarrow p1\uparrow; [\neg pe]; c0e\downarrow; [\neg c01]; p1\downarrow]$
$\quad [\!] m1 \longrightarrow c1e\uparrow;$
$\quad\quad [c10 \longrightarrow p0\uparrow; [\neg pe]; c1e\downarrow; [\neg c10]; p0\downarrow$
$\quad\quad [\!] c11 \longrightarrow p1\uparrow; [\neg pe]; c1e\downarrow; [\neg c11]; p1\downarrow]$
$\quad [\!] mw \longrightarrow cwe\uparrow$
$\quad\quad [cw0 \longrightarrow p0\uparrow; [\neg pe]; cwe\downarrow; [\neg cw0]; p0\downarrow$
$\quad\quad [\!] cw1 \longrightarrow p1\uparrow; [\neg pe]; cwe\downarrow; [\neg cw1]; p1\downarrow]$
$\ ]]$

$$\begin{array}{lcl} pe \wedge m0 & \rightarrow & c0e\uparrow \\ \neg pe \vee \neg m0 & \rightarrow & c0e\downarrow \end{array} \qquad \begin{array}{lcl} pe \wedge mw & \rightarrow & cwe\uparrow \\ \neg pe \vee \neg mw & \rightarrow & cwe\downarrow \end{array}$$

$$\begin{array}{lcl} pe \wedge m1 & \rightarrow & c1e\uparrow \\ \neg pe \vee \neg m1 & \rightarrow & c1e\downarrow \end{array}$$

$$\begin{array}{lcl} c00 \vee c10 \vee cw0 & \rightarrow & p0\uparrow \\ \neg c00 \wedge \neg c10 \wedge \neg cw0 & \rightarrow & p0\downarrow \end{array} \qquad \begin{array}{lcl} c01 \vee c11 \vee cw1 & \rightarrow & p1\uparrow \\ \neg c01 \wedge \neg c11 \wedge \neg cw1 & \rightarrow & p1\downarrow \end{array}$$

Radix 2 transistor accounting:

rule	transistor count	comments
$c[0,1,w]e$	12	
$p[0,1]$	12	
total	24	

Radix 4 transistor accounting:

rule	transistor count	comments
$c[0,1,2,3,w]e$	20	
$p[0,1,2,3]$	40	
total	60	

AEXT CD noTW LEAF

$*[[C0 \bullet P;\ W!0;\ C0 \bullet P$
$\ \ |\ C1 \bullet P;\ W!1;\ C1 \bullet P$
$]]$

$*[[c0 \longrightarrow po\!\uparrow;\ [pi];\ c0o\!\uparrow;$
$\quad [we];\ w0\!\uparrow;\ [\neg we];\ w0\!\downarrow$
$\quad [\neg c0];\ po\!\downarrow;\ [\neg pi];\ c0o\!\downarrow$
$[]\,c1 \longrightarrow po\!\uparrow;\ [pi];\ c1o\!\uparrow;$
$\quad [we];\ w1\!\uparrow;\ [\neg we];\ w1\!\downarrow$
$\quad [\neg c1];\ po\!\downarrow;\ [\neg pi];\ c1o\!\downarrow$
$]]$

$*[[c0 \longrightarrow q0\!\uparrow;\ po\!\uparrow;\ [pi];\ c0o\!\uparrow;$
$\quad [\neg c0 \wedge we];\ w0\!\uparrow;\ [\neg we];\ q0\!\downarrow;\ po\!\downarrow;\ [\neg pi];\ w0\!\downarrow;$
$\quad c0o\!\downarrow$
$[]\,c1 \longrightarrow q1\!\uparrow;\ po\!\uparrow;\ [pi];\ c1o\!\uparrow;$
$\quad [\neg c1 \wedge we];\ w1\!\uparrow;\ [\neg we];\ q1\!\downarrow;\ po\!\downarrow;\ [\neg pi];\ w1\!\downarrow;$
$\quad c1o\!\downarrow$
$]]$

The sequence of

$[\neg we];\ po\!\downarrow;\ [\neg pi];\ wx\!\downarrow;$

is very important. Parent nodes need to reset control of merge before lowering word line.

$$\begin{array}{lcl} c0 \wedge \neg c1o & \rightarrow & q0\!\uparrow \\ w0 \wedge \neg we & \rightarrow & q0\!\downarrow \end{array} \qquad \begin{array}{lcl} c0 \wedge \neg c1o & \rightarrow & q1\!\uparrow \\ w1 \wedge \neg we & \rightarrow & q1\!\downarrow \end{array}$$

$$\begin{array}{lcl} q0 \vee q1 & \rightarrow & po\!\uparrow \\ \neg q0 \wedge \neg q1 & \rightarrow & po\!\downarrow \end{array}$$

$$\begin{array}{lcl} q0 \wedge \neg c0 \wedge we & \rightarrow & w0\!\uparrow \\ \neg pi & \rightarrow & w0\!\downarrow \end{array} \qquad \begin{array}{lcl} q1 \wedge \neg c1 \wedge we & \rightarrow & w1\!\uparrow \\ \neg pi & \rightarrow & w1\!\downarrow \end{array}$$

$$q0 \wedge pi \rightarrow c0o\uparrow \qquad q1 \wedge pi \rightarrow c1o\uparrow$$
$$\neg w0 \wedge \neg pi \rightarrow c0o\downarrow \qquad \neg w1 \wedge \neg pi \rightarrow c1o\downarrow$$

Radix 2 transistor accounting:

rule	transistor count	comments
$c[0,1]$	12	2-input arbiter
$q[0,1]$	16	
p_o	4	
$w[0,1]$	16	
$c[0,1]_o$	16	
total	64	

Radix 4 transistor accounting:

rule	transistor count	comments
$c[0,1,2,3]$	92	4-input arbiter
$q[0,1,2,3]$	40	
p_o	8	
$w[0,1,2,3]$	32	
$c[0,1,2,3]_o$	32	
total	204	

B.1.3 AEXT CD TW

This design has a tail word.

Radix 2 accounting (4095 nodes / 4096 neurons):

component	transistors/component	components/node	transistors/node
CTRL	84	1	84
MERGE	28	1	28
FWDT	15	2	30
		total transistors/node	142

142 transistors/node * 4095 nodes / 4096 neurons = 142.0 transistors/neuron

We also need

8 transistors / INT * 1 INT / neuron = 8 transistors/neuron

This gives us

142 + 8 = **150 transistors/neuron**

Radix 4 transistor accounting (1365 nodes / 4096 neurons):

component	transistors/component	components/node	transistors/node
CTRL	238	1	238
MERGE	68	1	68
FWDT	15	4	60
		total transistors/node	366

366 transistors/node * 1365 nodes / 4096 neurons = 122.0 transistors/neuron

We also need

8 transistors / INT * 1 INT / neuron = 8 transistors/neuron

This gives us

122 + 8 = **130 transistors/neuron**

AEXT CD TW CTRL

Control.

$$
\begin{array}{l}
*[[\overline{C0} \longrightarrow P; mh \bullet H!0; m0 \bullet F0; C0 \\
\;\;|\, \overline{C1} \longrightarrow P; mh \bullet H!1; m1 \bullet F1; C1 \\
\;]]
\end{array}
$$

$$
\begin{array}{l}
*[[c0 \longrightarrow po\uparrow; [pi]; \\
\qquad mw\uparrow; [we]; w0\uparrow; [\neg we]; mw\downarrow; w0\downarrow; \\
\qquad m0\uparrow; f0o\uparrow; [f0i]; m0\downarrow; f0o\downarrow; [\neg f0i] \\
\qquad c0o\uparrow; [\neg c0]; po\downarrow; [\neg pi]; c0o\downarrow \\
\;\square\, c1 \longrightarrow po\uparrow; [pi]; \\
\qquad mw\uparrow; [we]; w1\uparrow; [\neg we]; mw\downarrow; w1\downarrow; \\
\qquad m1\uparrow; f1o\uparrow; [f1i]; m1\downarrow; f1o\downarrow; [\neg f1i] \\
\qquad c1o\uparrow; [\neg c1]; po\downarrow; [\neg pi]; c1o\downarrow \\
]]
\end{array}
$$

$$
\begin{array}{l}
*[[c0 \longrightarrow po\uparrow; [pi]; \\
\qquad (c0o\uparrow; [\neg c0]), \\
\qquad (mw\uparrow; [we]; w0\uparrow; [\neg we]; m0\uparrow; mw\downarrow; w0\downarrow; \\
\qquad f0o\uparrow; [f0i]; po\downarrow; [\neg pi]); \\
\qquad m0\downarrow; f0o\downarrow; c0o\downarrow; [\neg f0i] \\
\;\square\, c1 \longrightarrow po\uparrow; [pi]; \\
\qquad (c1o\uparrow; [\neg c1]), \\
\qquad (mw\uparrow; [we]; w1\uparrow; [\neg we]; m1\uparrow; mw\downarrow; w1\downarrow; \\
\qquad f1o\uparrow; [f1i]; po\downarrow; [\neg pi]); \\
\qquad m1\downarrow; f1o\downarrow; c1o\downarrow; [\neg f1i] \\
]]
\end{array}
$$

$$\begin{array}{lcl} (c0 \vee c1) \wedge \neg f1i \wedge \neg f0i & \rightarrow & po\uparrow \\ f1i \vee f0i & \rightarrow & po\downarrow \end{array}$$

$$\begin{array}{lcl} pi \wedge \neg m0 \wedge \neg m1 & \rightarrow & mw\uparrow \\ \neg pi \vee m0 \vee m1 & \rightarrow & mw\downarrow \end{array} \qquad \begin{array}{lcl} w1 \wedge \neg we & \rightarrow & m1\uparrow \\ \neg pi \wedge \neg c1 & \rightarrow & m1\downarrow \end{array}$$

$$\begin{array}{lcl} w0 \wedge \neg we & \rightarrow & m0\uparrow \\ \neg pi \wedge \neg c0 & \rightarrow & m0\downarrow \end{array}$$

$$\begin{array}{lcl} c0o \wedge we & \rightarrow & w0\uparrow \\ \neg mw & \rightarrow & w0\downarrow \end{array} \qquad \begin{array}{lcl} c1o \wedge we & \rightarrow & w1\uparrow \\ \neg mw & \rightarrow & w1\downarrow \end{array}$$

$$\begin{array}{lcl} m0 \wedge \neg w0 & \rightarrow & f0o\uparrow \\ \neg m0 \vee w0 & \rightarrow & f0o\downarrow \end{array} \qquad \begin{array}{lcl} m1 \wedge \neg w1 & \rightarrow & f1o\uparrow \\ \neg m1 \vee w1 & \rightarrow & f0o\downarrow \end{array}$$

$$\begin{array}{lcl} c0 \wedge pi \wedge \neg c1o & \rightarrow & c0o\uparrow \\ \neg f0o \wedge \neg pi & \rightarrow & c0o\downarrow \end{array} \qquad \begin{array}{lcl} c1 \wedge pi \wedge \neg c0o & \rightarrow & c1o\uparrow \\ \neg f1o \wedge \neg pi & \rightarrow & c1o\downarrow \end{array}$$

Radix 2 transistor accounting:

rule	transistor count	comments
$c[0,1]$	12	2-input arbiter
p_o	10	
mw	6	
$m[0,1]$	16	
$w[0,1]$	14	
$f[0,1]_o$	8	
$c[0,1]_o$	18	
total	84	

Radix 4 transistor accounting:

rule	transistor count	comments
$c[0,1,2,3]$	92	4-input arbiter
p_o	16	
mw	10	
$m[0,1,2,3]$	32	
$w[0,1,2,3]$	28	
$f[0,1,2,3]_o$	16	
$c[0,1,2,3]_o$	44	
total	238	

AEXT CD TW MERGE

Controlled merge.

$$
\begin{array}{l}
*[[\overline{M0} \longrightarrow *[P!(C0?)] \\
\quad [] \overline{M1} \longrightarrow *[P!(C1?)] \\
\quad [] \overline{Mh} \longrightarrow P!(H?) \\
\;]]
\end{array}
$$

$$
\begin{array}{l}
*[[pe]; \\
\quad [m0 \longrightarrow c0e\uparrow; \\
\qquad [c00 \longrightarrow p0\uparrow; [\neg pe]; c0e\downarrow; [\neg c00]; p0\downarrow \\
\qquad [] c01 \longrightarrow p1\uparrow; [\neg pe]; c0e\downarrow; [\neg c01]; p1\downarrow \\
\qquad [] c0t \longrightarrow pt\uparrow; [\neg pe]; c0e\downarrow; [\neg c0t]; pt\downarrow] \\
\quad [] m1 \longrightarrow c1e\uparrow; \\
\qquad [c10 \longrightarrow p0\uparrow; [\neg pe]; c1e\downarrow; [\neg c10]; p0\downarrow \\
\qquad [] c11 \longrightarrow p1\uparrow; [\neg pe]; c1e\downarrow; [\neg c11]; p1\downarrow \\
\qquad [] c1t \longrightarrow pt\uparrow; [\neg pe]; c1e\downarrow; [\neg c1t]; pt\downarrow] \\
\quad [] mw \longrightarrow cwe\uparrow \\
\qquad [cw0 \longrightarrow p0\uparrow; [\neg pe]; cwe\downarrow; [\neg cw0]; p0\downarrow \\
\qquad [] cw1 \longrightarrow p1\uparrow; [\neg pe]; cwe\downarrow; [\neg cw1]; p1\downarrow] \\
\;]]
\end{array}
$$

$$
\begin{array}{lll}
pe \wedge m0 & \rightarrow & c0e\uparrow \\
\neg pe \vee \neg m0 & \rightarrow & c0e\downarrow
\end{array}
\qquad
\begin{array}{lll}
pe \wedge mw & \rightarrow & cwe\uparrow \\
\neg pe \vee \neg mw & \rightarrow & cwe\downarrow
\end{array}
$$

$$
\begin{array}{lll}
pe \wedge m1 & \rightarrow & c1e\uparrow \\
\neg pe \vee \neg m1 & \rightarrow & c1e\downarrow
\end{array}
$$

$$
\begin{array}{lll}
c00 \vee c10 \vee cw0 & \rightarrow & p0\uparrow \\
\neg c00 \wedge \neg c10 \wedge \neg cw0 & \rightarrow & p0\downarrow
\end{array}
\qquad
\begin{array}{lll}
c0t \vee c1t & \rightarrow & pt\uparrow \\
\neg c0t \wedge \neg c1t & \rightarrow & pt\downarrow
\end{array}
$$

$$
\begin{array}{lll}
c01 \vee c11 \vee cw1 & \rightarrow & p1\uparrow \\
\neg c01 \wedge \neg c11 \wedge \neg cw1 & \rightarrow & p1\downarrow
\end{array}
$$

Radix 2 transistor accounting:

rule	transistor count	comments
$c[0,1,w]e$	12	
$p[0,1]$	12	
pt	4	
total	28	

Radix 4 transistor accounting:

rule	transistor count	comments
$c[0,1,2,3,w]e$	20	
$p[0,1,2,3]$	40	
pt	8	
total	68	

AEXT CD TW WORD

Output a word.

$*[Y!(X?)]$

$*[[ye];xe\uparrow;$
$\quad [x0 \longrightarrow y0\uparrow;[\neg ye];xe\downarrow;[\neg x0];y0\downarrow$
$\quad [\!]x1 \longrightarrow y1\uparrow;[\neg ye];xe\downarrow;[\neg x1];y1\downarrow$
$\;]]$

$$\begin{array}{lcl} ye & \rightarrow & xe\uparrow \\ \neg ye & \rightarrow & xe\downarrow \end{array}$$

$$\begin{array}{lcl} x0 & \rightarrow & y0\uparrow \\ \neg x0 & \rightarrow & y0\downarrow \end{array} \qquad \begin{array}{lcl} x1 & \rightarrow & y1\uparrow \\ \neg x1 & \rightarrow & y1\downarrow \end{array}$$

Radix 2 transistor accounting:

rule	transistor count	comments
total	0	all wires

Radix 4 transistor accounting:

rule	transistor count	comments
total	0	all wires

AEXT CD TW FWDT

Forward and detect tail.

$*[C\uparrow;$
$\quad [X=0 \longrightarrow Y!X$
$\quad [\!]X=1 \longrightarrow Y!X$
$\quad [\!]X=t \longrightarrow Y!X;C\downarrow$
$\;]]$

$$
\begin{array}{l}
*[[ci \wedge ye]; xe\uparrow; \\
\quad [x0 \longrightarrow y0\uparrow; [\neg ye]; xe\downarrow; [\neg x0]; y0\downarrow \\
\quad [\!]\, x1 \longrightarrow y1\uparrow; [\neg ye]; xe\downarrow; [\neg x1]; y1\downarrow \\
\quad [\!]\, xt \longrightarrow yt\uparrow; [\neg ye]; xe\downarrow; co\uparrow; [\neg xt \wedge \neg ci]; yt\downarrow; co\downarrow \\
\ ]]
\end{array}
$$

how do I express this in CHP?

$$
\begin{array}{lcl@{\qquad\qquad}lcl}
ci \wedge ye & \rightarrow & xe\uparrow & yt \wedge \neg xe & \rightarrow & co\uparrow \\
\neg ci \vee \neg ye & \rightarrow & xe\downarrow & \neg yt \vee xe & \rightarrow & co\downarrow
\end{array}
$$

$$
\begin{array}{lcl@{\qquad\qquad}lcl}
x0 & \rightarrow & y0\uparrow & xt & \rightarrow & yt\uparrow \\
\neg x0 & \rightarrow & y0\downarrow & \neg xt \wedge \neg ci & \rightarrow & yt\downarrow
\end{array}
$$

$$
\begin{array}{lcl}
x1 & \rightarrow & y1\uparrow \\
\neg x1 & \rightarrow & y1\downarrow
\end{array}
$$

Radix 2 transistor accounting:

rule	transistor count	comments
xe	4	
c_o	4	
$y[0,1]$	0	wires
yt	7	
total	15	

Radix 4 transistor accounting:

rule	transistor count	comments
xe	4	
c_o	4	
$y[0,1,2,3]$	0	wires
yt	7	
total	15	

AEXT CD TW INT

Neuron interface.

$$*[N; C; D]$$

$$
\begin{array}{l}
*[[ni]; co\uparrow; [ci \wedge de]; dt\uparrow; no\uparrow; \\
\quad [\neg ni]; co\downarrow; [\neg ci \wedge \neg de]; dt\downarrow; no\downarrow]
\end{array}
$$

$$
\begin{array}{lll}
ni & \rightarrow & co\uparrow \\
\neg ni & \rightarrow & co\downarrow
\end{array}
\qquad
\begin{array}{lll}
dt & \rightarrow & no\uparrow \\
\neg dt & \rightarrow & no\downarrow
\end{array}
$$

$$
\begin{array}{lll}
ci \wedge de & \rightarrow & dt\uparrow \\
\neg ci \wedge \neg de & \rightarrow & dt\downarrow
\end{array}
$$

Transistor accounting:

rule	transistor count	comments
c_o	0	wires
dt	8	
n_o	0	wires
total	8	

B.2 AEXT Control Data Combined

In these designs, the control and data are not decomposed.

B.3 AEXT ASPR NODE

The active-sender, passive-receiver (ASPR) design assumes that the children actively send data to their passively receveiving parents. The following describes a monolithic node process encapsulating that idea.

$$
\begin{array}{l}
NODE \equiv \\
*[[h \longrightarrow \\
\quad [\overline{C0} \longrightarrow s := 0, P!(0); \\
\quad |\overline{C1} \longrightarrow s := 1, P!(1)]; \\
\quad h := \mathit{false} \\
\ [] \neg h \longrightarrow \\
\quad [s = 0 \longrightarrow C0?x; P!x \\
\quad []s = 1 \longrightarrow C1?x; P!x \\
\quad]; h := x.tail \\
\ ]]
\end{array}
$$

B.4 AEXT ASPR PFWD/MERGE (PM)

The monolithic NODE can be decomposed into PFWD, which prepends a word to the packet indicating which branch the packet is coming from and MERGE processes, and MERGE, which arbitrates between incoming packet streams and outputs them one at a time.

Radix 2 accounting (4095 nodes / 4096 neurons):

component	transistors/component	components/node	transistors/node
PFWD	53	2	106
MERGE	90	1	90
		total transistors/node	196

196 transistors/node * 4095 nodes / 4096 neurons = **196.0 transistors/neuron**

The leaf node PFWDs only need to communicate the tail bit and their prepend bit and can leave off the other bit. This saves 2 production rules, or 14 transistors per node. With 2048 leaf nodes, this saves us 28672 transistors.

(196 transistors/node * 4095 nodes - 28672) / 4096 neurons = **189.0 transistors/neuron**

Radix 4 transistor accounting (1365 nodes / 4096 neurons):

component	transistors/component	components/node	transistors/node
PFWD	73	4	292
MERGE	288	1	288
		total transistors/node	580

580 transistors/node * 1365 nodes / 4096 neurons = **193.3 transistors/neuron**

The leaf node PFWDs only need to communicate the tail bit and their prepend bit and can leave off the other bit. This saves 4 production rules, or 32 transistors per node. With 1024 leaf nodes, this saves us 32768 transistors.

(698 transistors/node * 4095 nodes - 32768) / 4096 neurons = **185.3 transistors/neuron**

B.5 AEXT ASPR PM PFWD unpipelined

HSE

$$
\begin{array}{l}
*[h \wedge (x0 \vee x1 \vee xt) \longrightarrow q\uparrow; yp\uparrow; [yi]; h\downarrow; yp\downarrow; [\neg yi]; q\downarrow \\
\quad [\!]\, \neg h \wedge \neg q \wedge x0 \longrightarrow y0\uparrow; [yi]; xo\uparrow; [\neg x0]; y0\downarrow; [\neg yi]; xo\downarrow \\
\quad [\!]\, \neg h \wedge \neg q \wedge x1 \longrightarrow y1\uparrow; [yi]; xo\uparrow; [\neg x1]; y1\downarrow; [\neg yi]; xo\downarrow \\
\quad [\!]\, \neg h \wedge \neg q \wedge xt \longrightarrow yt\uparrow; [yi]; xo\uparrow; [\neg xt]; h\uparrow; yt\downarrow; [\neg yi]; xo\downarrow]
\end{array}
$$

PRS

$$
\begin{array}{lclclcl}
\neg xt \wedge yt & \rightarrow & h\uparrow & \qquad & h \wedge (x0 \vee x1 \vee xt) & \rightarrow & q\uparrow \\
q \wedge yi & \rightarrow & h\downarrow & & \neg h \wedge \neg yi & \rightarrow & q\downarrow
\end{array}
$$

$$
\begin{array}{lcl}
\neg q \wedge yi & \rightarrow & xo\uparrow \\
q \vee \neg yi & \rightarrow & xo\downarrow
\end{array}
$$

$$
\begin{aligned}
h \wedge q &\rightarrow yp\uparrow \\
\neg h \wedge q \wedge yi &\rightarrow yp\downarrow
\end{aligned}
$$

$$
\begin{aligned}
\neg h \wedge \neg q \wedge x0 &\rightarrow y0\uparrow & \qquad \neg h \wedge \neg q \wedge xt &\rightarrow yt\uparrow \\
\neg q \wedge \neg x0 &\rightarrow y0\downarrow & \qquad h &\rightarrow yt\downarrow
\end{aligned}
$$

$$
\begin{aligned}
\neg h \wedge \neg q \wedge x1 &\rightarrow y1\uparrow \\
\neg q \wedge \neg x1 &\rightarrow y1\downarrow
\end{aligned}
$$

Radix 2 transistor accounting:

rule	transistor count	comments
h	8	
q	10	
x_o	4	
yp	5	OR'ed with a y rule below which also provides the staticizer
$y[0,1]$	18	
yt	8	
total	53	

Radix 4 transistor accounting:

rule	transistor count	comments
h	8	
q	12	
x_o	4	
yp	5	OR'ed with a y rule below which also provides the staticizer
$y[0,1,2,3]$	36	
yt	8	
total	73	

B.6 AEXT ASPR PM PFWD pipelined hq

This version has fewer state variables than AEXT PFWD hu, but the pull-up and pull-down chains are too long.

CHP

$$
\begin{array}{l}
PFWD \equiv \\
\quad h := true; \\
\quad *[[h \wedge \overline{X} \longrightarrow Y!(\text{header}); h\downarrow \\
\quad\quad [] \neg h \wedge \overline{X} \longrightarrow Y!(X?) \bullet [X = t \longrightarrow h\uparrow]; \\
\quad\quad] \\
\quad]
\end{array}
$$

HSE

$$
\begin{array}{l}
*[h \wedge (x0 \vee x1 \vee xt) \longrightarrow yp\uparrow; [yi]; q\uparrow; yp\downarrow; [\neg yi]; h\downarrow; q\downarrow \\
\quad [] \neg h \wedge x0 \longrightarrow y0\uparrow; xo\uparrow; [yi]; y0\downarrow; [\neg x0]; xo\downarrow; [\neg yi] \\
\quad [] \neg h \wedge x1 \longrightarrow y1\uparrow; xo\uparrow; [yi]; y1\downarrow; [\neg x1]; xo\downarrow; [\neg yi] \\
\quad [] \neg h \wedge xt \longrightarrow yt\uparrow; xo\uparrow; [yi]; h\uparrow; yt\downarrow; [\neg xt]; xo\downarrow; [\neg yi]]
\end{array}
$$

PRS

$$
\begin{array}{lcl@{\qquad\qquad}lcl}
h \wedge yi \wedge yp & \to & q\uparrow & yt \wedge xo \wedge yi & \to & h\uparrow \\
\neg h & \to & q\downarrow & q \wedge \neg yi & \to & h\downarrow
\end{array}
$$

$$
\begin{array}{lcl}
\neg h \wedge (y0 \vee y1 \vee yt) & \to & xo\uparrow \\
\neg x0 \wedge \neg x1 \wedge \neg xt \wedge \neg y0 \wedge \neg y1 \wedge \neg yt & \to & xo\downarrow
\end{array}
$$

$$
\begin{array}{lcl}
h \wedge \neg q \wedge \neg yi \wedge \neg xo \wedge (x0 \vee xi \vee xt) & \to & yp\uparrow \\
q & \to & yp\downarrow
\end{array}
$$

$$
\begin{array}{lcl@{\qquad\qquad}lcl}
\neg h \wedge \neg q \wedge \neg yi \wedge x0 \wedge \neg xo & \to & y0\uparrow & \neg h \wedge \neg q \wedge \neg yi \wedge x1 \wedge \neg xo & \to & y1\uparrow \\
yi \wedge xo & \to & y0\downarrow & yi \wedge xo & \to & y1\downarrow
\end{array}
$$

$$
\begin{array}{lcl}
\neg h \wedge \neg q \wedge \neg yi \wedge xt \wedge \neg xo & \to & yt\uparrow \\
h \wedge xo & \to & yt\downarrow
\end{array}
$$

Radix 2 transistor accounting:

rule	transistor count	comments
q	8	
h	9	
xo	14	
yp	8	OR'ed with a y rule below which also provides the staticizer
$y[0,1]$	22	
yt	11	
total	72	

Radix 4 transistor accounting:

rule	transistor count	comments
q	8	
h	9	
xo	20	
yp	10	OR'ed with a y rule below which also provides the staticizer
$y[0,1,2,3]$	44	
yt	11	
total	102	

B.7 AEXT ASPR PM PFWD hu

This version has more state variables than AEXT PFWD hq, but has reasonable pull-up and pull-down chains.

CHP

$$
\begin{array}{l}
PFWD \equiv \\
\quad *[[h \wedge \overline{X} \longrightarrow Y!(\text{header}); h\downarrow \\
\quad\quad [] \neg h \wedge \overline{X} \longrightarrow X?u \bullet Y!u, [u = t \longrightarrow h\uparrow] \\
\quad\quad] \\
\quad]
\end{array}
$$

HSE

$$
\begin{array}{l}
*[[h \wedge (x0 \vee x1 \vee xt) \longrightarrow yp\uparrow; [yi]; h\downarrow; yp\downarrow; [\neg yi] \\
\quad [] \neg h \wedge x0 \longrightarrow u0\uparrow; (xo\uparrow; [\neg x0]), (y0\uparrow; [yi]); u0\downarrow; (y0\downarrow; [\neg yi]), xo\downarrow \\
\quad [] \neg h \wedge x1 \longrightarrow u1\uparrow; (xo\uparrow; [\neg x1]), (y1\uparrow; [yi]); u1\downarrow; (y1\downarrow; [\neg yi]), xo\downarrow \\
\quad [] \neg h \wedge xt \longrightarrow ut\uparrow; (xo\uparrow; [\neg xt]), (yt\uparrow; h\uparrow; [yi]); ut\downarrow; (yt\downarrow; [\neg yi]), xo\downarrow \\
\quad] \\
]
\end{array}
$$

$$
\begin{array}{l}
*[\,[h \wedge (x0 \vee x1 \vee xt) \longrightarrow yp\uparrow; [yi]; h\downarrow; yp\downarrow; [\neg yi] \\
\quad [\!]\, \neg h \wedge x0 \longrightarrow u0\uparrow; [\neg x0 \wedge yi]; u0\downarrow; [\neg yi] \\
\quad [\!]\, \neg h \wedge x1 \longrightarrow u1\uparrow; [\neg x1 \wedge yi]; u1\downarrow; [\neg yi] \\
\quad [\!]\, \neg h \wedge xt \longrightarrow ut\uparrow; [\neg xt \wedge h \wedge yi]; ut\downarrow; [\neg yi] \\
\quad] \\
\;]
\end{array}
$$

$$
\begin{array}{l}
*[u0 \longrightarrow xo\uparrow, y0\uparrow; [\neg u0]; y0\downarrow, xo\downarrow \\
\quad [\!]\, u1 \longrightarrow xo\uparrow, y1\uparrow; [\neg u1]; y1\downarrow, xo\downarrow \\
\quad [\!]\, ut \longrightarrow xo\uparrow, (yt\uparrow; h\uparrow); [ut\downarrow]; yt\downarrow, xo\downarrow \\
\quad]
\end{array}
$$

PRS

$$
\begin{array}{lcl}
yt & \rightarrow & h\uparrow \\
yp \wedge yi & \rightarrow & h\downarrow
\end{array}
$$

$$
\begin{array}{lcl}
h \wedge (x0 \vee x1 \vee xt) \wedge \neg yi \wedge \neg yt & \rightarrow & yp\uparrow \\
\neg h \wedge yi \wedge \neg un & \rightarrow & yp\downarrow
\end{array}
$$

$$
\begin{array}{lcl}
u0 \vee u1 \vee ut & \rightarrow & xo\uparrow \\
\neg u0 \wedge \neg u1 \wedge \neg ut & \rightarrow & xo\downarrow
\end{array}
$$

$$
\begin{array}{lclclcl}
\neg h \wedge x0 \wedge \neg yi & \rightarrow & u0\uparrow & \qquad & \neg h \wedge x1 \wedge \neg yi & \rightarrow & u1\uparrow \\
\neg x0 \wedge yi & \rightarrow & u0\downarrow & \qquad & \neg x1 \wedge yi & \rightarrow & u1\downarrow
\end{array}
$$

$$
\begin{array}{lcl}
\neg h \wedge xt \wedge \neg yi & \rightarrow & ut\uparrow \\
h \wedge \neg xt \wedge yi & \rightarrow & ut\downarrow
\end{array}
$$

$$
\begin{array}{lclclcl}
u0 & \rightarrow & y0\uparrow & \qquad & u1 & \rightarrow & y1\uparrow \\
\neg h \wedge \neg u0 & \rightarrow & y0\downarrow & \qquad & \neg h \wedge \neg u1 & \rightarrow & y1\downarrow
\end{array}
$$

$$
\begin{array}{lcl}
ut & \rightarrow & yt\uparrow \\
\neg ut & \rightarrow & yt\downarrow
\end{array}
$$

Radix 2 transistor accounting:

rule	transistor count	comments
h	7	
yp	9	OR'ed with a y rule below which also provides the staticizer
xo	6	
$u[0,1]$	18	
ut	10	
$y[0,1]$	14	
yt	6	
total	70	

Radix 4 transistor accounting:

rule	transistor count	comments
h	7	
yp	11	OR'ed with a y rule below which also provides the staticizer
xo	10	
$u[0,1,2,3]$	36	
ut	10	
$y[0,1,2,3]$	28	
yt	6	
total	108	

B.8 AEXT ASPR PM MERGE unpipelined

HSE

$$
\begin{array}{l}
*[[\neg a0 \wedge \neg a1 \wedge (c00 \vee c01 \vee c0t) \longrightarrow a0\uparrow \\
\quad | \neg a0 \wedge \neg a1 \wedge (c10 \vee c11 \vee c1t) \longrightarrow a1\uparrow]]
\end{array}
$$

$$
\begin{array}{l}
*[[\ a0 \wedge c00 \longrightarrow p0\uparrow; [pi]; c0o\uparrow; [\neg c00]; p0\downarrow; [\neg pi]; c0o\downarrow \\
\quad [] a0 \wedge c01 \longrightarrow p1\uparrow; [pi]; c0o\uparrow; [\neg c01]; p1\downarrow; [\neg pi]; c0o\downarrow \\
\quad [] a0 \wedge c0t \longrightarrow pt\uparrow; [pi]; c0o\uparrow; [\neg c0t]; a0\downarrow; pt\downarrow; [\neg pi]; c0o\downarrow \\
\quad [] a1 \wedge c10 \longrightarrow p0\uparrow; [pi]; c1o\uparrow; [\neg c10]; p0\downarrow; [\neg pi]; c1o\downarrow \\
\quad [] a1 \wedge c11 \longrightarrow p1\uparrow; [pi]; c1o\uparrow; [\neg c11]; p1\downarrow; [\neg pi]; c1o\downarrow \\
\quad [] a1 \wedge c1t \longrightarrow pt\uparrow; [pi]; c1o\uparrow; [\neg c1t]; a1\downarrow; pt\downarrow; [\neg pi]; c1o\downarrow \\
\quad]]
\end{array}
$$

PRS

$$\begin{array}{lcl} (c00 \vee c01 \vee c0t) \wedge \neg a0 & \rightarrow & a0i\uparrow \\ a0 & \rightarrow & a0i\downarrow \end{array} \qquad \begin{array}{lcl} a0o \wedge \neg a1 \wedge \neg pi & \rightarrow & a0\uparrow \\ \neg a0o \wedge pt \wedge \neg c0t & \rightarrow & a0\downarrow \end{array}$$

$$\begin{array}{lcl} (c10 \vee c11 \vee c1t) \wedge \neg a1 & \rightarrow & a1i\uparrow \\ a1 & \rightarrow & a1i\downarrow \end{array} \qquad \begin{array}{lcl} a1o \wedge \neg a0 \wedge \neg pi & \rightarrow & a1\uparrow \\ \neg a1o \wedge pt \wedge \neg c1t & \rightarrow & a1\downarrow \end{array}$$

$$\begin{array}{lcl} (a0 \wedge c00 \vee a1 \wedge c10) & \rightarrow & p0\uparrow \\ (a0 \wedge \neg c00 \vee a1 \wedge \neg c10) & \rightarrow & p0\downarrow \end{array} \qquad \begin{array}{lcl} (a0 \wedge c0t \vee a1 \wedge c1t) & \rightarrow & pt\uparrow \\ \neg a0 \wedge \neg a1 & \rightarrow & pt\downarrow \end{array}$$

$$\begin{array}{lcl} (a0 \wedge c00 \vee a1 \wedge c10) & \rightarrow & p0\uparrow \\ (a0 \wedge \neg c00 \vee a1 \wedge \neg c10) & \rightarrow & p0\downarrow \end{array}$$

$$\begin{array}{lcl} a0 \wedge pi & \rightarrow & c0o\uparrow \\ \neg a0 \vee \neg pi & \rightarrow & c0o\downarrow \end{array} \qquad \begin{array}{lcl} a1 \wedge pi & \rightarrow & c1o\uparrow \\ \neg a1 \vee \neg pi & \rightarrow & c1o\downarrow \end{array}$$

Radix 2 transistor accounting:

rule	transistor count	comments
$a[0,1]i$	16	can be combinational with radix 2
$a[0,1]o$	12	2-way arbiter
$a[0,1]$	20	
$p[0,1]$	24	
pt	10	
$c[0,1]o$	8	
total	90	

Radix 4 transistor accounting:

rule	transistor count	comments
$a[0,1,2,3]i$	44	
$a[0,1,2,3]o$	92	4-way unpipelined arbiter
$a[0,1,2,3]$	40	
$p[0,1,2,3]$	80	
pt	16	
$c[0,1,2,3]o$	16	
total	288	

B.9 AEXT ASPR PM MERGE pipelined a_a

I don't like this version because the pullup chains for the state variables are too long and won't scale to higher radix encoding.

CHP

$$
\begin{array}{l}
MERGE \equiv \\
\quad *[[h \longrightarrow [\overline{C0} \longrightarrow a := 0 \mid \overline{C1} \longrightarrow a := 1]; h\downarrow \\
\quad\quad [] \neg h \wedge a = 0 \longrightarrow P!(C0?) \\
\quad\quad [] \neg h \wedge a = 0 \longrightarrow P!(C1?) \\
\quad\quad]]
\end{array}
$$

HSE

$$
\begin{array}{l}
*[[\neg a0 \wedge \neg a1 \wedge (c00 \vee c01 \vee c0t) \longrightarrow a0\uparrow \\
\quad \mid \neg a0 \wedge \neg a1 \wedge (c10 \vee c11 \vee c1t) \longrightarrow a1\uparrow]]
\end{array}
$$

$$
\begin{array}{l}
*[[\;\; a0 \wedge c00 \longrightarrow p0\uparrow; c0o\uparrow; [pi \wedge \neg c00]; p0\downarrow; c0o\downarrow; [\neg pi] \\
\quad [] a0 \wedge c01 \longrightarrow p1\uparrow; c0o\uparrow; [pi \wedge \neg c01]; p1\downarrow; c0o\downarrow; [\neg pi] \\
\quad [] a0 \wedge c0t \longrightarrow pt\uparrow; c0o\uparrow; [pi \wedge \neg c0t]; a0\downarrow; pt\downarrow; c0o\downarrow; [\neg pi] \\
\quad [] a1 \wedge c10 \longrightarrow p0\uparrow; c1o\uparrow; [pi \wedge \neg c10]; p0\downarrow; c1o\downarrow; [\neg pi] \\
\quad [] a1 \wedge c11 \longrightarrow p1\uparrow; c1o\uparrow; [pi \wedge \neg c11]; p1\downarrow; c1o\downarrow; [\neg pi] \\
\quad [] a1 \wedge c1t \longrightarrow pt\uparrow; c1o\uparrow; [pi \wedge \neg c1t]; a1\downarrow; pt\downarrow; c1o\downarrow; [\neg pi] \\
\quad]]
\end{array}
$$

PRS

$$
\begin{array}{lcl@{\qquad}lcl}
\neg a0 \wedge (c00 \vee c01 \vee c0t) \wedge \neg c0o & \rightarrow & a0i\uparrow & a0o \wedge \neg a1 \wedge \neg c1o \wedge \neg pi & \rightarrow & a0\uparrow \\
a0 & \rightarrow & a0i\downarrow & \neg a0o \wedge pi \wedge pt \wedge \neg c0t & \rightarrow & a0\downarrow
\end{array}
$$

$$
\begin{array}{lcl@{\qquad}lcl}
\neg a1 \wedge (c10 \vee c11 \vee c1t) \wedge \neg c1o & \rightarrow & a1i\uparrow & a1o \wedge \neg a0 \wedge \neg c0o \wedge \neg pi & \rightarrow & a1\uparrow \\
a1 & \rightarrow & a1i\downarrow & \neg a1o \wedge pi \wedge pt \wedge \neg c1t & \rightarrow & a1\downarrow
\end{array}
$$

$$
\begin{array}{lcl@{\qquad}lcl}
\neg pi \wedge (a0 \wedge c00 \vee a1 \wedge c10) & \rightarrow & p0\uparrow & \neg pi \wedge (a0 \wedge c0t \vee a1 \wedge c1t) & \rightarrow & pt\uparrow \\
pi \wedge (\neg\_a0 \wedge \neg c00 \vee \neg\_a1 \wedge \neg c10) & \rightarrow & p0\downarrow & \_a0 \wedge \_a1 & \rightarrow & pt\downarrow
\end{array}
$$

$$
\begin{array}{lcl}
\neg pi \wedge (a0 \wedge c01 \vee a1 \wedge c11) & \rightarrow & p1\uparrow \\
pi \wedge (\neg\_a0 \wedge \neg c01 \vee \neg\_a1 \wedge \neg c11) & \rightarrow & p1\downarrow
\end{array}
$$

$$
\begin{array}{lcl@{\qquad}lcl}
a0 \wedge (p0 \vee p1 \vee pt) & \rightarrow & c0o\uparrow & a1 \wedge (p0 \vee p1 \vee pt) & \rightarrow & c1o\uparrow \\
\neg p0 \wedge \neg p1 \wedge \neg pt & \rightarrow & c0o\downarrow & \neg p0 \wedge \neg p1 \wedge \neg pt & \rightarrow & c1o\downarrow
\end{array}
$$

Radix 2 transistor accounting:

rule	transistor count	comments
$a[0,1]i$	20	
$a[0,1]o$	12	2-way arbiter
$a[0,1]$	24	
$p[0,1]$	28	
pt	11	
$c[0,1]o$	22	
total	117	

Radix 4 transistor accounting:

rule	transistor count	comments
$a[0,1,2,3]i$	48	
$a[0,1,2,3]o$	92	4-way unpipelined arbiter
$a[0,1,2,3]$	48	
$p[0,1,2,3]$	88	
pt	17	
$c[0,1,2,3]o$	60	
total	353	

B.10 AEXT ASPR PM MERGE pipelined ah

This one has acceptable pullup/pulldown chains, but I'm worried about making it CMOS implementable

HSE

$$
\begin{aligned}
&*[[\neg a0 \wedge (c00 \vee c01 \vee c0t) \longrightarrow a0\uparrow; h\downarrow; [\neg a0]; h\uparrow \\
&\quad |\neg a1 \wedge (c10 \vee c11 \vee c1t) \longrightarrow a1\uparrow; h\downarrow; [\neg a1]; h\uparrow]]
\end{aligned}
$$

$$
\begin{aligned}
&*[[\;\; a0 \wedge c00 \longrightarrow p0\uparrow; c0o\uparrow; [pi \wedge \neg c00]; p0\downarrow; c0o\downarrow; [\neg pi] \\
&\quad [\!]\, a0 \wedge c01 \longrightarrow p1\uparrow; c0o\uparrow; [pi \wedge \neg c01]; p1\downarrow; c0o\downarrow; [\neg pi] \\
&\quad [\!]\, a0 \wedge c0t \longrightarrow pt\uparrow; c0o\uparrow; [pi \wedge \neg c0t]; a0\downarrow; pt\downarrow; c0o\downarrow; [\neg pi] \\
&\quad [\!]\, a1 \wedge c10 \longrightarrow p0\uparrow; c1o\uparrow; [pi \wedge \neg c10]; p0\downarrow; c1o\downarrow; [\neg pi] \\
&\quad [\!]\, a1 \wedge c11 \longrightarrow p1\uparrow; c1o\uparrow; [pi \wedge \neg c11]; p1\downarrow; c1o\downarrow; [\neg pi] \\
&\quad [\!]\, a1 \wedge c1t \longrightarrow pt\uparrow; c1o\uparrow; [pi \wedge \neg c1t]; a1\downarrow; pt\downarrow; c1o\downarrow; [\neg pi] \\
&\quad]]
\end{aligned}
$$

PRS

$$\begin{array}{lcl} (c00 \lor c01 \lor c0t) \land \neg a0 & \to & a0i\uparrow \\ \neg h \land a0 & \to & a0i\downarrow \end{array} \qquad \begin{array}{lcl} h \land a0o \land \neg pi & \to & a0\uparrow \\ pt \land pi \land c0o \land \neg c0t \land \neg a0o & \to & a0\downarrow \end{array}$$

$$\begin{array}{lcl} (c10 \lor c11 \lor c1t) \land \neg a1 & \to & a1i\uparrow \\ \neg h \land a1 & \to & a1i\downarrow \end{array} \qquad \begin{array}{lcl} h \land a1o \land \neg pi & \to & a1\uparrow \\ pt \land pi \land c1o \land \neg c1t \land \neg a1o & \to & a1\downarrow \end{array}$$

$$\begin{array}{lcl} \neg a0 \land \neg a1 & \to & h\uparrow \\ a0 \lor a1 \land \neg c0o \land \neg c1o & \to & h\downarrow \end{array}$$

$$\begin{array}{lcl} \neg pi \land (a0 \land c00 \lor a1 \land c10) \land \neg h & \to & p0\uparrow \\ pi \land (a0 \land \neg c00 \lor a1 \land \neg c10) & \to & p0\downarrow \end{array} \qquad \begin{array}{lcl} \neg pi \land (a0 \land c0t \lor a1 \land c1t) \land \neg h & \to & pt\uparrow \\ \neg a0 \land \neg a1 & \to & pt\downarrow \end{array}$$

$$\begin{array}{lcl} \neg pi \land (a0 \land c01 \lor a1 \land c11) \land \neg h & \to & p1\uparrow \\ pi \land (a0 \land \neg c01 \lor a1 \land \neg c11) & \to & p1\downarrow \end{array}$$

$$\begin{array}{lcl} a0 \land (p0 \lor p1 \lor pt) & \to & c0o\uparrow \\ \neg p0 \land \neg p1 \land \neg pt & \to & c0o\downarrow \end{array} \qquad \begin{array}{lcl} a1 \land (p0 \lor p1 \lor pt) & \to & c1o\uparrow \\ \neg p0 \land \neg p1 \land \neg pt & \to & c1o\downarrow \end{array}$$

Radix 2 transistor accounting:

rule	transistor count	comments
$a[0,1]i$	20	
$a[0,1]o$	12	2-way arbiter
$a[0,1]$	24	
h	10	
$p[0,1]$	30	
pt	12	
$c[0,1]o$	22	
total	130	

Radix 4 transistor accounting:

rule	transistor count	comments
$a[0,1,2,3]i$	48	
$a[0,1,2,3]o$	92	4-way unpipelined arbiter
$a[0,1,2,3]$	48	
h	14	
$p[0,1,2,3]$	52	
pt	16	
$c[0,1,2,3]o$	100	
total	370	

B.11 AEXT ASPR PFWD PREPEND/FWD/SIMPLE_MERGE

PREPEND and FWD are further decomposed into PREPEND, FWD, and SIMPLE_MERGE.

B.12 AEXT ASPR PFWD PREPEND

HSE

$$*[[si]; yp\uparrow; [yi]; yp\downarrow; [\neg yi]; so\uparrow; [\neg si]; so\downarrow]$$

B.13 AEXT PFWD FWD

HSE

$$\begin{array}{l}
*[[x0 \vee x1 \vee xt]; so\uparrow; [si]; so\downarrow; [\neg si];\\
\quad [x0 \longrightarrow y0\uparrow; [yi]; y0\downarrow; [\neg yi]\\
\quad [\!] x1 \longrightarrow y1\uparrow; [yi]; y1\downarrow; [\neg yi]\\
\quad [\!] xt \longrightarrow yt\uparrow; [yi]; yt\downarrow; [\neg yi]\\
\;]]
\end{array}$$

B.14 AEXT ASPR PFWD SIMPLE_MERGE

Assumes that its inputs are mutually exclusive.

HSE

$$\begin{array}{l}
*[[c00 \longrightarrow p0\uparrow; [pi]; c0o\uparrow; [\neg c00]; p0\downarrow; [\neg pi]; c0o\downarrow\\
\quad [\!] c01 \longrightarrow p1\uparrow; [pi]; c0o\uparrow; [\neg c01]; p1\downarrow; [\neg pi]; c0o\downarrow\\
\quad [\!] c0t \longrightarrow pt\uparrow; [pi]; c0o\uparrow; [\neg c0t]; pt\downarrow; [\neg pi]; c0o\downarrow\\
\quad [\!] c10 \longrightarrow p0\uparrow; [pi]; c1o\uparrow; [\neg c00]; p0\downarrow; [\neg pi]; c1o\downarrow\\
\quad [\!] c11 \longrightarrow p1\uparrow; [pi]; c1o\uparrow; [\neg c01]; p1\downarrow; [\neg pi]; c1o\downarrow\\
\quad [\!] c1t \longrightarrow pt\uparrow; [pi]; c1o\uparrow; [\neg c0t]; pt\downarrow; [\neg pi]; c1o\downarrow\\
\;]]
\end{array}$$

PRS

$$\begin{array}{lcl@{\qquad\qquad}lcl}
c00 \vee c10 & \to & p0\uparrow & c0t \vee c1t & \to & pt\uparrow\\
\neg c00 \wedge \neg c10 & \to & p0\downarrow & \neg c0t \wedge \neg c1t & \to & pt\downarrow\\
\\
c01 \vee c11 & \to & p1\uparrow & & & \\
\neg c01 \wedge \neg c11 & \to & p1\downarrow & & &
\end{array}$$

$$
\begin{array}{llcl}
pi \wedge (c00 \vee c01 \vee c0t) & \rightarrow & c0o\uparrow \qquad & pi \wedge (c10 \vee c11 \vee c1t) \rightarrow c1o\uparrow \\
\neg pi & \rightarrow & c0o\downarrow \qquad & \neg pi \rightarrow c1o\downarrow
\end{array}
$$

B.15 AEXT ASPR MERGE

MERGE sequences between outputting two serialized packet streams.

```
MERGE ≡
 *[[¬l ∧ ¬r ⟶
     [L̄ ⟶ l := true
     |R̄ ⟶ r := true
     ]
   []l ∨ r ⟶
     [l ⟶ O!(L?)
     []r ⟶ O!(R?)
   ]]
 ]
```

Appendix C

AER Receiver Design Space

This appendix explores the AER receiver (AERV) design space in reverse chronological order.

C.1 Receiver tree structure

The receiver tree structure is dictated by its interface with the synapse and neuron/synapse configuration memory. We could place a memory for each group of 1 synapse and 4 neurons,
Tree structure:

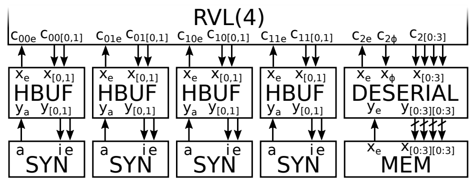
1 NODE	
4 NODE	
16 NODE	
64 NODE	
256 LEAF	
1024 SYN	1024 DESERIAL
	1024 MEM4

Accounting:

component	transistors/component	components	transistors	comments
AERV NODE	152	85	12920	
AERV LEAF	152	256	38912	
DESERIAL	216	1024	221184	6 1-of-2 words
OR	4	1024	4096	
		total	277112	
		total/neuron	67.65	

This design is expensive. We are required to use the 1-of-2 instead of 1-of-4 deserializer. The deserializer costs are derived from the number of words, which are derived from the shape of the memory as detailed in Section D.28.

To reduce overhead and use a 1-of-4 deserializer, we bundle 2 synapses (8 neurons) into each port and consolidate the memories for the 16 neurons and their synapses into a single memory:



By consolidating the memories, we further save a port on each leaf node.
Tree structure:

1 NODE
4 NODE
16 NODE
64 NODE
256 LEAF(3)
512 SYN2 256 DESERIAL
256 MEM16

Accounting:

component	transistors/component	components	transistors	comments
AERV NODE	152	85	12920	
AERV LEAF(3)	114	256	29184	
DESERIAL	126	256	32256	4 1-of-4 words
OR	4	512	2048	
		total	86408	
		total/neuron	21.10	

We could continue this line of thought by consolidating synapses and memories until all configuration data is stored in a single, monolithic memory.
Tree structure:

1 NODE(3)
2 NODE DESERIAL
8 NODE MEM4096
32 NODE
128 LEAF
128 SYN8

Accounting:

component	transistors/component	components	transistors	comments
AERV NODE(3)	114	1	114	
AERV NODE	152	42	6384	
AERV LEAF	152	128	19456	
DESERIAL	352	1	352	
OR	4	512	2048	
		total	28354	
		total/neuron	6.92	

This design is the cheapest, but would would be difficult to layout and wire.

C.2 AERV CD noTW cyclic control (CYC)

Builds off AERV CD noTW (Section C.6) by reusing the control lines to request and acknowledge the data as is done in AEXT CD noTW cyc (Section B.1.1).
Radix 2 accounting:

intermediate nodes			
component	transistors/component	components/node	transistors/node
NODE	60	1	60
		total transistors/intermediate node	60
leaf nodes			
component	transistors/component	components/node	transistors/node
LEAF	60	1	60
		total transistors/leaf node	60

(60 transistors/intermediate node * 511 intermediate nodes + 60 transistors/leaf node * 512 leaf nodes) / 4096 neurons = **15.0 transistors/neuron**
Radix 4 accounting:

intermediate nodes			
component	transistors/component	components/node	transistors/node
NODE	152	1	152
		total transistors/intermediate node	152
leaf nodes			
component	transistors/component	components/node	transistors/node
LEAF	152	1	152
		total transistors/leaf node	152

(152 transistors/intermediate node * 256 intermediate nodes + 152 transistors/leaf node * 85 leaf nodes) / 4096 neurons = **12.7 transistors/neuron**

C.3 AERV CD noTW CYC NODE

HSE

$$
\begin{array}{l}
*[[p\phi \longrightarrow po\uparrow;\\
\quad\quad [p0 \longrightarrow u0\uparrow; uu\uparrow; po\downarrow; [\neg p0]; v\uparrow; c0\phi\uparrow; [c0i]; po\uparrow\\
\quad\quad [\!]\, p1 \longrightarrow u1\uparrow; uu\uparrow; po\downarrow; [\neg p1]; v\uparrow; c1\phi\uparrow; [c1i]; po\uparrow\\
\quad\quad]\\
\quad [\!]\, p0 \wedge c0\phi \longrightarrow c00\uparrow; [\neg c0i]; po\downarrow; [\neg p0]; c00\downarrow; [c0i]; ; po\uparrow\\
\quad [\!]\, p1 \wedge c0\phi \longrightarrow c01\uparrow; [\neg c0i]; po\downarrow; [\neg p1]; c01\downarrow; [c0i]; ; po\uparrow\\
\quad [\!]\, p0 \wedge c1\phi \longrightarrow c10\uparrow; [\neg c1i]; po\downarrow; [\neg p0]; c10\downarrow; [c1i]; ; po\uparrow\\
\quad [\!]\, p1 \wedge c1\phi \longrightarrow c11\uparrow; [\neg c1i]; po\downarrow; [\neg p1]; c11\downarrow; [c1i]; ; po\uparrow\\
\quad [\!]\, \neg p\phi \longrightarrow u0\downarrow, u1\downarrow; uu\downarrow, (c0\phi\downarrow, c1\phi\downarrow; [\neg c0i \wedge \neg c1i]; po\downarrow), v\downarrow\\
\ ]]
\end{array}
$$

PRS

$$
\begin{array}{lcl}
u0 \vee u1 & \rightarrow & uu\uparrow\\
\neg u0 \wedge \neg u1 & \rightarrow & uu\downarrow
\end{array}
$$

$$
\begin{array}{lcl}
p\phi \wedge \neg uu \vee c0i \vee c1i & \rightarrow & po\uparrow\\
(\neg p\phi \vee uu) \wedge \neg c0i \wedge \neg c1i & \rightarrow & po\downarrow
\end{array}
$$

$$
\begin{array}{lcl}
p0 \wedge \neg v & \rightarrow & u0\uparrow\\
\neg p\phi & \rightarrow & u0\downarrow
\end{array}
$$

$$
\begin{array}{lcl}
p1 \wedge \neg v & \rightarrow & u1\uparrow\\
\neg p\phi & \rightarrow & u1\downarrow
\end{array}
$$

$$
\begin{array}{lcl}
uu \wedge \neg p0 \wedge \neg p1 & \rightarrow & v\uparrow\\
\neg uu & \rightarrow & v\downarrow
\end{array}
$$

$$
\begin{array}{lcl}
v \wedge u0 & \rightarrow & c0\phi\uparrow\\
\neg v \vee \neg u0 & \rightarrow & c0\phi\downarrow
\end{array}
$$

$$
\begin{array}{lcl}
v \wedge u1 & \rightarrow & c1\phi\uparrow\\
\neg v \vee \neg u1 & \rightarrow & c1\phi\downarrow
\end{array}
$$

$$
\begin{array}{lcl}
p0 \wedge c0\phi & \rightarrow & c00\uparrow \\
\neg p0 \vee \neg c0\phi & \rightarrow & c00\downarrow \\
\\
p1 \wedge c0\phi & \rightarrow & c01\uparrow \\
\neg p1 \vee \neg c0\phi & \rightarrow & c01\downarrow \\
\\
p0 \wedge c1\phi & \rightarrow & c10\uparrow \\
\neg p0 \vee \neg c1\phi & \rightarrow & c10\downarrow \\
\\
p1 \wedge c1\phi & \rightarrow & c11\uparrow \\
\neg p1 \vee \neg c1\phi & \rightarrow & c11\downarrow
\end{array}
$$

Radix 4 transistor approximate accounting:

rule	transistor count	comments
uu	8	
p_o	12	
$u[0,1,2,3]$	28	
v	10	
$c[0,1,2,3]\phi$	16	
$c[0,1,2,3][0,1,2,3]$	64	
total	138	

Alternative 0: HSE

$$
\begin{array}{l}
*[[pi];\, po\uparrow; \\
\quad [p0 \longrightarrow u0\uparrow; uu\uparrow; po\downarrow; [\neg p0]; c0o\uparrow; cco\uparrow; u0\downarrow; uu\downarrow; [c0i]; cci\uparrow; \\
\qquad po\uparrow; [\neg pi]; c0o\downarrow; cco\downarrow, ([\neg c0i]; cci\downarrow); po\downarrow \\
\quad [\!]\, p1 \longrightarrow u1\uparrow; uu\uparrow; po\downarrow; [\neg p1]; c1o\uparrow; cco\uparrow; u1\downarrow; uu\downarrow; [c1i]; cci\uparrow; \\
\qquad po\uparrow; [\neg pi]; c1o\downarrow; cco\downarrow, ([\neg c1i]; cci\downarrow); po\downarrow \\
\quad]; \\
]
\end{array}
$$

$$
\begin{array}{l}
*[[c0o \wedge p0 \longrightarrow c00\uparrow; [\neg c0i]; cci\downarrow; po\downarrow; [\neg p0]; c00\downarrow; [c0i]; cci\uparrow; po\uparrow \\
\;[\!]\, c0o \wedge p1 \longrightarrow c01\uparrow; [\neg c0i]; cci\downarrow; po\downarrow; [\neg p1]; c01\downarrow; [c0i]; cci\uparrow; po\uparrow \\
\;[\!]\, c1o \wedge p0 \longrightarrow c10\uparrow; [\neg c1i]; cci\downarrow; po\downarrow; [\neg p0]; c10\downarrow; [c1i]; cci\uparrow; po\uparrow \\
\;[\!]\, c1o \wedge p1 \longrightarrow c11\uparrow; [\neg c1i]; cci\downarrow; po\downarrow; [\neg p1]; c11\downarrow; [c1i]; cci\uparrow; po\uparrow \\
]]
\end{array}
$$

PRS

$$(pi \wedge \neg cco \vee cci) \wedge \neg uu \rightarrow po\uparrow$$
$$(\neg pi \vee cco) \wedge \neg cci \vee uu \rightarrow po\downarrow$$

$$\begin{array}{lll} p0 \wedge \neg cco \rightarrow u0\uparrow & \quad & p1 \wedge \neg cco \rightarrow u1\uparrow \\ cco \rightarrow u0\downarrow & & cco \rightarrow u1\downarrow \end{array}$$

$$u0 \vee u1 \rightarrow uu\uparrow$$
$$\neg u0 \wedge \neg u1 \rightarrow uu\downarrow$$

$$\begin{array}{lll} u0 \wedge \neg p0 \rightarrow c0o\uparrow & \quad & u1 \wedge \neg p1 \rightarrow c1o\uparrow \\ \neg pi \rightarrow c0o\downarrow & & \neg pi \rightarrow c1o\downarrow \end{array}$$

$$\begin{array}{lll} c0o \vee c1o \rightarrow cco\uparrow & \quad & c0i \vee c1i \rightarrow cci\uparrow \\ \neg c0o \wedge \neg c1o \rightarrow cco\downarrow & & \neg c0i \wedge \neg c1i \rightarrow cci\downarrow \end{array}$$

$$\begin{array}{lll} c0o \wedge p0 \rightarrow c00\uparrow & \quad & c1o \wedge p0 \rightarrow c10\uparrow \\ \neg c0o \vee \neg p0 \rightarrow c00\downarrow & & \neg c1o \vee \neg p0 \rightarrow c10\downarrow \end{array}$$

$$\begin{array}{lll} c0o \wedge p1 \rightarrow c01\uparrow & \quad & c1o \wedge p1 \rightarrow c11\uparrow \\ \neg c0o \vee \neg p1 \rightarrow c01\downarrow & & \neg c1o \vee \neg p1 \rightarrow c11\downarrow \end{array}$$

Radix 2 transistor approximate accounting:

rule	transistor count	comments
p_o	8	
$u[0,1]$	14	
uu	4	could flatten in p_o
$c[0,1]_o$	14	
cc_o	4	
cc_i	4	could flatten in p_o
$c[0,1][0,1]$	16	
total	64	60 if cc_i and uu flattened in p_o

Radix 4 transistor approximate accounting:

rule	transistor count	comments
p_o	8	
$u[0,1,2,3]$	28	
uu	8	
$c[0,1,2,3]_o$	28	
cc_o	8	
cc_i	8	
$c[0,1,2,3][0,1,2,3]$	64	
total	152	flattening cc_i and uu would make p_o pullup chain too long

Radix 3, 1-of-4 out transistor approximate accounting: This is used in some of the receiver designs.

rule	transistor count	comments
p_o	8	
$u[0,1,2]$	21	
uu	6	
$c[0,1,2]_o$	21	
cc_o	6	
cc_i	6	
$c[0,1,2][0,1,2,3]$	48	
total	116	114 if cc_i or uu flattened in p_o (can't flatten both; pullup chain too long)

CMOS-implementable PRS

$$\begin{aligned}
&\neg _cci \rightarrow __cci\uparrow \\
&_cci \rightarrow __cci\downarrow
\end{aligned}$$

$$\begin{aligned}
&\neg cco \rightarrow _cco\uparrow \\
&cco \rightarrow _cco\downarrow
\end{aligned}$$

$$\begin{aligned}
&(pi \wedge _cco \vee __cci) \wedge _uu \rightarrow _po\downarrow \\
&(\neg pi \vee \neg _cco) \wedge \neg __cci \vee \neg _uu \rightarrow _po\uparrow
\end{aligned}$$

$$\begin{aligned}
&\neg p0 \rightarrow _p0\uparrow \qquad && \neg p1 \rightarrow _p1\uparrow \\
&p0 \rightarrow _p0\downarrow && p1 \rightarrow _p1\downarrow
\end{aligned}$$

$$\begin{aligned}
&\neg _cco \rightarrow __cco\uparrow \\
&_cco \rightarrow __cco\downarrow
\end{aligned}$$

$$\begin{aligned}
&\neg _p0 \wedge \neg __cco \rightarrow u0\uparrow \qquad && \neg _p1 \wedge \neg __cco \rightarrow u1\uparrow \\
&__cco \rightarrow u0\downarrow && __cco \rightarrow u1\downarrow
\end{aligned}$$

$$\begin{aligned} &u0 \vee u1 &&\rightarrow \_uu\downarrow \\ &\neg u0 \wedge \neg u1 &&\rightarrow \_uu\uparrow \end{aligned}$$

$$\begin{aligned} &u0 \wedge \_p0 \rightarrow \_c0o\downarrow && u1 \wedge \_p1 \rightarrow \_c1o\downarrow \\ &\neg pi \rightarrow \_c0o\uparrow && \neg pi \rightarrow \_c1o\uparrow \end{aligned}$$

$$\begin{aligned} &\neg\_c0o \vee \neg\_c1o \rightarrow cco\uparrow && c0i \vee c1i \rightarrow \_cci\downarrow \\ &\_c0o \wedge \_c1o \rightarrow cco\downarrow && \neg c0i \wedge \neg c1i \rightarrow \_cci\uparrow \end{aligned}$$

$$\begin{aligned} &\neg\_c0o \wedge \neg\_p0 \rightarrow c00\uparrow && \neg\_c1o \wedge \neg\_p0 \rightarrow c10\uparrow \\ &\_c0o \vee \_p0 \rightarrow c00\downarrow && \_c1o \vee \_p0 \rightarrow c10\downarrow \end{aligned}$$

$$\begin{aligned} &\neg\_c0o \wedge \neg\_p1 \rightarrow c01\uparrow && \neg\_c1o \wedge \neg\_p1 \rightarrow c11\uparrow \\ &\_c0o \vee \_p1 \rightarrow c01\downarrow && \_c1o \vee \_p1 \rightarrow c11\downarrow \end{aligned}$$

$$\begin{aligned} &\neg\_c0o \rightarrow c0o\uparrow && \neg\_c1o \rightarrow c1o\uparrow \\ &\_c0o \rightarrow c0o\downarrow && \_c1o \rightarrow c1o\downarrow \end{aligned}$$

$$\begin{aligned} &\neg\_po \rightarrow po\uparrow \\ &\_po \rightarrow po\downarrow \end{aligned}$$

Radix 2 transistor accounting:

rule	transistor count	comments
$\_\_cc_i$	2	
$\_cc_o$	2	
p_o	8	
$\_p[0,1]$	4	
$\_\_cc_o$	2	
$u[0,1]$	14	
$\_uu$	4	
$\_c[0,1]_o$	14	
cc_o	4	
$\_cc_i$	4	
$c[0,1][0,1]$	16	
$c[0,1]_o$	4	
p_o	2	
total	80	

Radix 4 transistor accounting:

rule	transistor count	comments
$\_\_cc_i$	2	
$\_cc_o$	2	
p_o	8	
$\_p[0,1,2,3]$	8	
$\_\_cc_o$	2	
$u[0,1,2,3]$	28	
$\_uu$	8	
$\_c[0,1,2,3]_o$	28	
cc_o	8	
$\_cc_i$	8	
$c[0,1,2,3][0,1,2,3]$	64	
$c[0,1,2,3]_o$	8	
p_o	2	
total	176	

C.4 AERV CD noTW CYC LEAF

This leaf design does not transmit the c_o signals to the neuron. Rather data just shows up on the cxx lines and the neurons acknowledge on the c_i lines.

$$
\begin{array}{l}
*[[pi];\, po\uparrow;\\
\quad [p0 \longrightarrow u0\uparrow;\, uu\uparrow;\, po\downarrow;\, [\neg p0];\, c0\uparrow;\, cc\uparrow;\, u0\downarrow;\, uu\downarrow;\\
\qquad po\uparrow;\, [\neg pi];\, c0\downarrow;\, cc\downarrow;\, po\downarrow\\
\quad [\!]\, p1 \longrightarrow u1\uparrow;\, uu\uparrow;\, po\downarrow;\, [\neg p1];\, c1\uparrow;\, cc\uparrow;\, u1\downarrow;\, uu\downarrow;\\
\qquad po\uparrow;\, [\neg pi];\, c1\downarrow;\, cc\downarrow;\, po\downarrow\\
\quad];\\
]
\end{array}
$$

$$
\begin{array}{l}
*[[c0 \wedge p0 \longrightarrow c00\uparrow;\, [c0i];\, cci\uparrow;\, po\downarrow;\, [\neg p0];\, c00\downarrow;\, [\neg c0i];\, cci\downarrow;\, po\uparrow\\
\;[\!]\, c0 \wedge p1 \longrightarrow c01\uparrow;\, [c0i];\, cci\uparrow;\, po\downarrow;\, [\neg p1];\, c01\downarrow;\, [\neg c0i];\, cci\downarrow;\, po\uparrow\\
\;[\!]\, c1 \wedge p0 \longrightarrow c10\uparrow;\, [c1i];\, cci\uparrow;\, po\downarrow;\, [\neg p0];\, c10\downarrow;\, [\neg c1i];\, cci\downarrow;\, po\uparrow\\
\;[\!]\, c1 \wedge p1 \longrightarrow c11\uparrow;\, [c1i];\, cci\uparrow;\, po\downarrow;\, [\neg p1];\, c11\downarrow;\, [\neg c1i];\, cci\downarrow;\, po\uparrow\\
]]
\end{array}
$$

PRS

$$
\begin{array}{lcl}
(pi \vee cco) \wedge \neg uu \wedge \neg cci & \rightarrow & po\uparrow\\
\neg pi \wedge \neg cco \vee uu \vee cci & \rightarrow & po\downarrow
\end{array}
$$

$$\begin{array}{lcl} \neg cc \wedge p0 & \rightarrow & u0\uparrow \\ cc & \rightarrow & u0\downarrow \end{array} \qquad \begin{array}{lcl} \neg cc \wedge p1 & \rightarrow & u1\uparrow \\ cc & \rightarrow & u1\downarrow \end{array}$$

$$\begin{array}{lcl} u0 \vee u1 & \rightarrow & uu\uparrow \\ \neg u0 \wedge \neg u1 & \rightarrow & uu\downarrow \end{array}$$

$$\begin{array}{lcl} u0 \wedge \neg p0 & \rightarrow & c0\uparrow \\ \neg pi & \rightarrow & c0\downarrow \end{array} \qquad \begin{array}{lcl} u1 \wedge \neg p1 & \rightarrow & c1\uparrow \\ \neg pi & \rightarrow & c1\downarrow \end{array}$$

$$\begin{array}{lcl} c0 \vee c1 & \rightarrow & cc\uparrow \\ \neg c0 \wedge \neg c1 & \rightarrow & cc\downarrow \end{array} \qquad \begin{array}{lcl} c0i \vee c1i & \rightarrow & cci\uparrow \\ \neg c0i \wedge \neg c1i & \rightarrow & cci\downarrow \end{array}$$

$$\begin{array}{lcl} c0 \wedge p0 & \rightarrow & c00\uparrow \\ \neg c0 \vee \neg p0 & \rightarrow & c00\downarrow \end{array} \qquad \begin{array}{lcl} c1 \wedge p0 & \rightarrow & c10\uparrow \\ \neg c1 \vee \neg p0 & \rightarrow & c10\downarrow \end{array}$$

$$\begin{array}{lcl} c0 \wedge p1 & \rightarrow & c01\uparrow \\ \neg c0 \vee \neg p1 & \rightarrow & c01\downarrow \end{array} \qquad \begin{array}{lcl} c1 \wedge p1 & \rightarrow & c11\uparrow \\ \neg c1 \vee \neg p1 & \rightarrow & c11\downarrow \end{array}$$

Radix 2 transistor approximate accounting:

rule	transistor count	comments
p_o	8	
$u[0,1]$	14	
uu	4	could flatten in p_o
$c[0,1]_o$	14	
cc_o	4	
cc_i	4	could flatten in p_o
$c[0,1][0,1]$	16	
total	64	60 if cc_i and uu flattened in p_o

Radix 4 transistor approximate accounting:

rule	transistor count	comments
p_o	8	
$u[0,1,2,3]$	28	
uu	8	
$c[0,1,2,3]_o$	28	
cc_o	8	
cc_i	8	
$c[0,1,2,3][0,1,2,3]$	64	
total	152	flattening cc_i and uu would make p_o pullup chain too long

Radix 3, 1-of-4 out transistor approximate accounting: This is used in some of the receiver designs.

rule	transistor count	comments
p_o	8	
$u[0,1,2]$	21	
uu	6	
$c[0,1,2]_o$	21	
cc_o	6	
cc_i	6	
$c[0,1,2][0,1,2,3]$	48	
total	116	114 if cc_i or uu flattened in p_o (can't flatten both; pullup chain too long)

CMOS-implementable PRS

$$\begin{aligned} \neg pi &\rightarrow \_pi\uparrow \\ pi &\rightarrow \_pi\downarrow \end{aligned}$$

$$\begin{aligned} \neg\_cci &\rightarrow \_\_cci\uparrow \\ \_cci &\rightarrow \_\_cci\downarrow \end{aligned}$$

$$\begin{aligned} (\neg\_pi \vee \neg\_cc) \wedge \neg uu \wedge \neg\_\_cci &\rightarrow po\uparrow \\ \_pi \wedge \_cc \vee uu \vee \_\_cci &\rightarrow po\downarrow \end{aligned}$$

$$\begin{aligned} \_cc \wedge p0 &\rightarrow \_u0\downarrow & \_cc \wedge p1 &\rightarrow \_u1\downarrow \\ \neg\_cc &\rightarrow \_u0\uparrow & \neg\_cc &\rightarrow \_u1\uparrow \end{aligned}$$

$$\begin{aligned} \neg\_u0 \vee \neg\_u1 &\rightarrow uu\uparrow \\ \_u0 \wedge \_u1 &\rightarrow uu\downarrow \end{aligned}$$

$$\begin{aligned} \neg\_u0 \wedge \neg p0 &\rightarrow c0\uparrow & \neg\_u1 \wedge \neg p1 &\rightarrow c1\uparrow \\ \_pi &\rightarrow c0\downarrow & \_pi &\rightarrow c1\downarrow \end{aligned}$$

$$\begin{aligned} c0 \vee c1 &\rightarrow \_cc\downarrow & c0i \vee c1i &\rightarrow \_cci\downarrow \\ \neg c0 \wedge \neg c1 &\rightarrow \_cc\uparrow & \neg c0i \wedge \neg c1i &\rightarrow \_cci\uparrow \end{aligned}$$

$$\begin{aligned} c0 \wedge p0 &\rightarrow \_c00\downarrow & c1 \wedge p0 &\rightarrow \_c10\downarrow \\ \neg c0 \vee \neg p0 &\rightarrow \_c00\uparrow & \neg c1 \vee \neg p0 &\rightarrow \_c10\uparrow \end{aligned}$$

$$\begin{aligned} c0 \wedge p1 &\rightarrow \_c01\downarrow & c1 \wedge p1 &\rightarrow \_c11\downarrow \\ \neg c0 \vee \neg p1 &\rightarrow \_c01\uparrow & \neg c1 \vee \neg p1 &\rightarrow \_c11\uparrow \end{aligned}$$

$$\begin{array}{lll} \neg\_c00 & \rightarrow & c00\uparrow \\ \_c00 & \rightarrow & c00\downarrow \end{array} \qquad \begin{array}{lll} \neg\_c10 & \rightarrow & c10\uparrow \\ \_c10 & \rightarrow & c10\downarrow \end{array}$$

$$\begin{array}{lll} \neg\_c01 & \rightarrow & c01\uparrow \\ \_c01 & \rightarrow & c01\downarrow \end{array} \qquad \begin{array}{lll} \neg\_c11 & \rightarrow & c11\uparrow \\ \_c11 & \rightarrow & c11\downarrow \end{array}$$

Alternative 1: HSE

$$\begin{array}{l}
*[[p\phi \longrightarrow po\uparrow; \\
\quad\quad [p0 \longrightarrow u0\uparrow; uu\uparrow; po\downarrow; [\neg p0]; v\uparrow; c0s\uparrow; , po\uparrow \\
\quad\quad [\!]\, p1 \longrightarrow u1\uparrow; uu\uparrow; po\downarrow; [\neg p1]; v\uparrow; c1s\uparrow; , po\uparrow \\
\quad\quad] \\
\quad [\!]\, p0 \wedge c0s \longrightarrow c00\uparrow; [c0i]; cci\uparrow; po\downarrow; [\neg p0]; c00\downarrow; [\neg c0i]; cci\downarrow; po\uparrow \\
\quad [\!]\, p1 \wedge c0s \longrightarrow c01\uparrow; [c0i]; cci\uparrow; po\downarrow; [\neg p1]; c01\downarrow; [\neg c0i]; cci\downarrow; po\uparrow \\
\quad [\!]\, p0 \wedge c1s \longrightarrow c10\uparrow; [c1i]; cci\uparrow; po\downarrow; [\neg p0]; c10\downarrow; [\neg c1i]; cci\downarrow; po\uparrow \\
\quad [\!]\, p1 \wedge c1s \longrightarrow c11\uparrow; [c1i]; cci\uparrow; po\downarrow; [\neg p1]; c11\downarrow; [\neg c1i]; cci\downarrow; po\uparrow \\
\quad [\!]\, \neg p\phi \longrightarrow u0\downarrow, u1\downarrow; uu\downarrow, c0s\downarrow, c1s\downarrow; v\downarrow, po\downarrow \\
\ ]]
\end{array}$$

$$\begin{array}{lll} u0 \vee u1 & \rightarrow & uu\uparrow \\ \neg u0 \wedge \neg u1 & \rightarrow & uu\downarrow \end{array}$$

$$\begin{array}{lll} c0i \vee c1i & \rightarrow & cci\uparrow \\ \neg c0i \wedge \neg c1i & \rightarrow & cci\downarrow \end{array}$$

$$\begin{array}{lll} p\phi \wedge \neg uu \vee v \wedge \neg cci & \rightarrow & po\uparrow \\ (\neg p\phi \vee uu) \wedge (\neg v \vee cci) & \rightarrow & po\downarrow \end{array}$$

$$\begin{array}{lll} p0 \wedge \neg v & \rightarrow & u0\uparrow \\ \neg p\phi & \rightarrow & u0\downarrow \end{array}$$

$$\begin{array}{lll} p1 \wedge \neg v & \rightarrow & u1\uparrow \\ \neg p\phi & \rightarrow & u1\downarrow \end{array}$$

$$\begin{array}{lll} uu \wedge \neg p0 \wedge \neg p1 & \rightarrow & v\uparrow \\ \neg uu \wedge \neg c0s \wedge \neg c1s & \rightarrow & v\downarrow \end{array}$$

$$\begin{array}{lll} v \wedge u0 & \rightarrow & c0s\uparrow \\ \neg v \vee \neg u0 & \rightarrow & c0s\downarrow \end{array}$$

$$\begin{array}{lll} v \wedge u1 & \rightarrow & c1s\uparrow \\ \neg v \vee \neg u1 & \rightarrow & c1s\downarrow \end{array}$$

$$p0 \wedge c0s \rightarrow c00\uparrow$$
$$\neg p0 \vee \neg c0s \rightarrow c00\downarrow$$

$$p1 \wedge c0s \rightarrow c01\uparrow$$
$$\neg p1 \vee \neg c0s \rightarrow c01\downarrow$$

$$p0 \wedge c1s \rightarrow c10\uparrow$$
$$\neg p0 \vee \neg c1s \rightarrow c10\downarrow$$

$$p1 \wedge c1s \rightarrow c11\uparrow$$
$$\neg p1 \vee \neg c1s \rightarrow c11\downarrow$$

Radix 2 transistor approximate accounting:

rule	transistor count	comments
uu	4	
cc_i	4	could flatten in p_o
p_o	8	
$u[0,1]$	14	
v	10	
$c[0,1]s$	8	
$c[0,1][0,1]$	16	
total	69	67 if cc_i flattened in p_o

Radix 4 transistor approximate accounting:

rule	transistor count	comments
uu	8	
cc_i	8	could flatten in p_o
p_o	8	
$u[0,1,2,3]$	28	
v	14	
$c[0,1,2,3]s$	16	
$c[0,1,2,3][0,1,2,3]$	64	
total	146	144 if cc_i flattened in p_o

C.5 AERV CD noTW CYC LEAF (no data)

If we don't need to send any data to the neurons we can make very cheap leaf nodes.

$$
\begin{array}{l}
*[[pi];\ po\uparrow;\\
\quad [p0 \longrightarrow c0o\uparrow;\ [c0i];\ po\downarrow;\ [\neg p0];\ u\uparrow;\ po\uparrow;\ [\neg pi];\ c0o\downarrow [\neg c0i];\ u\downarrow;\ po\downarrow\\
\quad [\!]\, p1 \longrightarrow c1o\uparrow;\ [c1i];\ po\downarrow;\ [\neg p1];\ u\uparrow;\ po\uparrow;\ [\neg pi];\ c1o\downarrow [\neg c1i];\ u\downarrow;\ po\downarrow\\
\quad];\\
]
\end{array}
$$

PRS

$$
\begin{array}{lcl}
(pi \wedge \neg c0i \wedge \neg c1i) \vee u & \rightarrow & po\uparrow \\
(\neg pi \vee c0i \vee c1i) \wedge \neg u & \rightarrow & po\downarrow
\end{array}
$$

$$
\begin{array}{lcl@{\qquad\qquad}lcl}
p0 & \rightarrow & c0o\uparrow & p1 & \rightarrow & c1o\uparrow \\
\neg pi & \rightarrow & c0o\downarrow & \neg pi & \rightarrow & c1o\downarrow
\end{array}
$$

$$
\begin{array}{lcl}
c0i \wedge \neg p0 \vee c1i \wedge \neg p1 & \rightarrow & u\uparrow \\
(\neg c0i \vee p0) \wedge (\neg c1i \vee p1) & \rightarrow & u\downarrow
\end{array}
$$

Radix 2 transistor approximate accounting:

rule	transistor count	comments
p_o	8	
$c[0,1]_o$	12	
$u[0,1]$	8	
total	28	

Radix 4 transistor approximate accounting:

rule	transistor count	comments
p_o	12	
$c[0,1,2,3]_o$	24	
$u[0,1,2,3]$	16	
total	52	

CMOS-implementable PRS

$$
\begin{array}{lcl@{\qquad\qquad}lcl}
\neg c0i & \rightarrow & \_c0i\uparrow & \neg c1i & \rightarrow & \_c1i\uparrow \\
c0i & \rightarrow & \_c0i\downarrow & c1i & \rightarrow & \_c1i\downarrow
\end{array}
$$

$$
\begin{array}{lcl}
(pi \wedge \_c0i \wedge \_c1i) \vee u & \rightarrow & \_po\downarrow \\
(\neg pi \vee \neg\_c0i \vee \neg\_c1i) \wedge \neg u & \rightarrow & po\uparrow
\end{array}
$$

$$
\begin{array}{lcl@{\qquad\qquad}lcl}
p0 & \rightarrow & \_c0o\downarrow & p1 & \rightarrow & \_c1o\downarrow \\
\neg pi & \rightarrow & \_c0o\uparrow & \neg pi & \rightarrow & \_c1o\uparrow
\end{array}
$$

$$
\begin{array}{lcl}
\neg\_c0i \wedge \neg p0 \vee \neg\_c1i \wedge \neg p1 & \rightarrow & u\uparrow \\
(\_c0i \vee p0) \wedge (\_c1i \vee p1) & \rightarrow & u\downarrow
\end{array}
$$

$$
\begin{array}{lcl}
\neg\_c0o & \rightarrow & c0o\uparrow \\
\_c0o & \rightarrow & c0o\downarrow
\end{array}
\qquad
\begin{array}{lcl}
\neg\_c1o & \rightarrow & c1o\uparrow \\
\_c1o & \rightarrow & c1o\downarrow
\end{array}
$$

$$
\begin{array}{lcl}
\neg\_po & \rightarrow & po\uparrow \\
\_po & \rightarrow & po\downarrow
\end{array}
$$

C.6 AERV Control Data decomposed (CD) no tailword (noTW)

Separating control from data and removing the tail word reduced the number of transistors in the transmitter. We'll try to apply these same techniques to the receiver. Specifically, we'll want something that can interface with the control/data decomposed, no tail word, cyclic control, transmitter developed in Section B.1.1.

The accounting depends on whether we need the receiver to deliver payload or not. Without payload, we can simplify the leaf node circuitry. With payload, we'll need a more complicated interface with the neuron to be developed.

First we'll consider the case without payload:

Radix 2 accounting (2047 intermediate nodes, 2048 leaf nodes):

intermediate nodes			
component	transistors/component	components/node	transistors/node
SPLIT	30	1	30
CTRL	36	1	36
total transistors/intermediate node			66
leaf nodes			
component	transistors/component	components/node	transistors/node
LEAF	30	1	30
total transistors/leaf node			30

(66 transistors/intermediate node * 2047 intermediate nodes + 30 transistors/leaf node * 2048 leaf nodes) / 4096 neurons = **48.0 transistors/neuron**

Radix 4 accounting (341 intermediate nodes, 1024 leaf nodes):

intermediate nodes			
component	transistors/component	components/node	transistors/node
SPLIT	90	1	90
CTRL	68	1	68
total transistors/intermediate node			158
leaf nodes			
component	transistors/component	components/node	transistors/node
LEAF	58	1	58
total transistors/leaf node			58

(158 transistors/intermediate node * 341 intermediate nodes + 58 transistors/leaf node * 1024 leaf nodes) / 4096 neurons = **27.7 transistors/neuron**

Now we'll consider the case where we have to deliver payload. In this case, we cannot use the simplified leaf nodes because there is data to be sent to the neuron. In addition, we'll need to develop more circuitry per neuron to set bits. We haven't specified what or how data will be set, so this will be developed in the future.

Radix 2 accounting (4095 nodes):

(66 transistors/node * 4095 nodes) / 4096 neurons = **66.0 transistors/neuron**

Radix 4 accounting (1365 nodes):

(158 transistors/node * 1365 nodes) / 4096 neurons = **52.6 transistors/neuron**

C.7 AERV CD noTW SPLIT

$*[[\overline{C0} \wedge S = 0 \longrightarrow C0!(P?)$
$\;\;[]\overline{C1} \wedge S = 1 \longrightarrow C1!(P?)$
$\;\;[]\overline{C2} \wedge S = 2 \longrightarrow C2!(P?)$
$]]$

$$
\begin{array}{l}
*[[c0e \longrightarrow pe\uparrow; \\
\quad [s0 \wedge p0 \longrightarrow c00\uparrow; [\neg c0e]; pe\downarrow; [\neg p0]; c00\downarrow \\
\quad [\!]s0 \wedge p1 \longrightarrow c01\uparrow; [\neg c0e]; pe\downarrow; [\neg p1]; c01\downarrow \\
\quad] \\
\;[\!]c1e \longrightarrow pe\uparrow; \\
\quad [s1 \wedge p0 \longrightarrow c10\uparrow; [\neg c1e]; pe\downarrow; [\neg p0]; c10\downarrow \\
\quad [\!]s1 \wedge p1 \longrightarrow c11\uparrow; [\neg c1e]; pe\downarrow; [\neg p1]; c11\downarrow \\
\quad] \\
\;[\!]c2e \longrightarrow pe\uparrow; \\
\quad [s2 \wedge p0 \longrightarrow c20\uparrow; [\neg c2e]; pe\downarrow; [\neg p0]; c20\downarrow \\
\quad [\!]s2 \wedge p1 \longrightarrow c21\uparrow; [\neg c2e]; pe\downarrow; [\neg p1]; c21\downarrow \\
\quad] \\
\;]]
\end{array}
$$

$$
\begin{array}{lcl}
c0e \vee c1e \vee c2e & \rightarrow & pe\uparrow \\
\neg c0e \wedge \neg c1e \wedge \neg c2e & \rightarrow & pe\downarrow
\end{array}
$$

$$
\begin{array}{lcl@{\qquad\qquad}lcl}
s0 \wedge p0 & \rightarrow & c00\uparrow & s1 \wedge p1 & \rightarrow & c11\uparrow \\
\neg s0 \vee \neg p0 & \rightarrow & c00\downarrow & \neg s1 \vee \neg p1 & \rightarrow & c11\downarrow \\
\\
s0 \wedge p1 & \rightarrow & c01\uparrow & s2 \wedge p0 & \rightarrow & c20\uparrow \\
\neg s0 \vee \neg p1 & \rightarrow & c01\downarrow & \neg s2 \vee \neg p0 & \rightarrow & c21\downarrow \\
\\
s1 \wedge p0 & \rightarrow & c10\uparrow & s2 \wedge p1 & \rightarrow & c21\uparrow \\
\neg s1 \vee \neg p0 & \rightarrow & c10\downarrow & \neg s2 \vee \neg p1 & \rightarrow & c21\downarrow
\end{array}
$$

Radix 2 transistor accounting:

rule	transistor count	comments
pe	6	
$c[0,1,2][0,1]$	24	
total	30	

Radix 4 transistor accounting:

rule	transistor count	comments
pe	10	
$c[0,1,2,3,4][0,1,2,3]$	80	
total	90	

C.8 AERV CD noTW CTRL

$$
\begin{array}{l}
*[[\overline{P} \longrightarrow X?u \bullet S := 2 \\
\quad [u = 0 \longrightarrow S := 0; C0; P \\
\quad [\!]\, u = 1 \longrightarrow S := 1; C1; P \\
\quad] \\
]]
\end{array}
$$

$$
\begin{array}{l}
*[[pi]; xe\uparrow; s2\uparrow; \\
\quad [x0 \longrightarrow u0\uparrow [\!]\, x1 \longrightarrow u1\uparrow]; xe\downarrow; [\neg x0 \wedge \neg x1]; s2\downarrow; \\
\quad [u0 \longrightarrow s0\uparrow; c0o\uparrow; [c0i]; po\uparrow; [\neg pi]; u0\downarrow; s0\downarrow; c0o\downarrow; [\neg c0i] \\
\quad [\!]\, u1 \longrightarrow s1\uparrow; c1o\uparrow; [c1i]; po\uparrow; [\neg pi]; u1\downarrow; s1\downarrow; c1o\downarrow; [\neg c1i] \\
\quad]; po\downarrow \\
]
\end{array}
$$

$$
\begin{array}{lcl}
pi \wedge \neg u0 \wedge \neg u1 & \rightarrow & xe\uparrow \\
\neg pi \vee u0 \vee u1 & \rightarrow & xe\downarrow
\end{array}
$$

$$
\begin{array}{lcl}
xe \vee x0 \vee x1 & \rightarrow & s2\uparrow \\
\neg xe \wedge \neg x0 \wedge \neg x1 & \rightarrow & s2\downarrow
\end{array}
$$

$$
\begin{array}{lcl@{\qquad}lcl}
x0 & \rightarrow & u0\uparrow & x1 & \rightarrow & u1\uparrow \\
\neg pi & \rightarrow & u0\downarrow & \neg pi & \rightarrow & u1\downarrow
\end{array}
$$

$$
\begin{array}{lcl@{\qquad}lcl}
u0 \wedge \neg s2 & \rightarrow & s0\uparrow & u1 \wedge \neg s2 & \rightarrow & s1\uparrow \\
\neg u0 \vee s2 & \rightarrow & s0\downarrow & \neg u1 \vee s2 & \rightarrow & s1\downarrow
\end{array}
$$

$$
\begin{array}{lcl@{\qquad}lcl}
s0 & \rightarrow & c0o\uparrow & s1 & \rightarrow & c1o\uparrow \\
\neg s0 & \rightarrow & c0o\downarrow & \neg s1 & \rightarrow & c1o\downarrow
\end{array}
$$

$$
\begin{array}{lcl}
c0i \vee c1i & \rightarrow & po\uparrow \\
\neg c0i \wedge \neg c1i & \rightarrow & po\downarrow
\end{array}
$$

Radix 2 transistor accounting:

rule	transistor count	comments
xe	6	
$s2$	6	
$u[0,1]$	12	
$s[0,1]$	8	
$c[0,1]_o$	0	$s[0,1] = c[0,1]_o$
p_o	4	
total	36	

Radix 4 transistor accounting:

rule	transistor count	comments
xe	10	
$s4$	10	
$u[0,1,2,3]$	24	
$s[0,1,2,3]$	16	
$c[0,1,2,3]o$	0	$s[0,1,2,3] = c[0,1,2,3]_o$
po	8	
total	68	

C.9 AERV CD noTW LEAF

In the case that we don't need to deliver payload to the neuron, we can use this LEAF process to interface with the neuron.

$$
\begin{array}{l}
*[[\overline{P} \longrightarrow X?u \\
\quad [u = 0 \longrightarrow C0; P \\
\quad [\!]\, u = 1 \longrightarrow C1; P \\
\quad] \\
]]
\end{array}
$$

$$
\begin{array}{l}
*[[pi]; xe\uparrow; \\
\quad [x0 \longrightarrow u0\uparrow [\!]\, x1 \longrightarrow u1\uparrow]; xe\downarrow; [\neg x0 \wedge \neg x1]; \\
\quad [u0 \longrightarrow c0o\uparrow; [c0i]; po\uparrow; [\neg pi]; u0\downarrow; c0o\downarrow; [\neg c0i] \\
\quad [\!]\, u1 \longrightarrow c1o\uparrow; [c1i]; po\uparrow; [\neg pi]; u1\downarrow; c1o\downarrow; [\neg c1i] \\
\quad]; po\downarrow \\
]
\end{array}
$$

$$
\begin{array}{lcl}
pi \wedge \neg u0 \wedge \neg u1 & \rightarrow & xe\uparrow \\
\neg pi \vee u0 \vee u1 & \rightarrow & xe\downarrow
\end{array}
$$

$$
\begin{array}{lcl}
x0 & \rightarrow & u0\uparrow \\
\neg pi & \rightarrow & u0\downarrow
\end{array}
\qquad
\begin{array}{lcl}
x1 & \rightarrow & u1\uparrow \\
\neg pi & \rightarrow & u1\downarrow
\end{array}
$$

$$
\begin{array}{lcl}
u0 \wedge \neg x0 & \rightarrow & c0o\uparrow \\
\neg u0 \vee x0 & \rightarrow & c0o\downarrow
\end{array}
\qquad
\begin{array}{lcl}
u1 \wedge \neg x1 & \rightarrow & c1o\uparrow \\
\neg u1 \vee x1 & \rightarrow & c1o\downarrow
\end{array}
$$

$$
\begin{array}{lcl}
c0i \vee c1i & \rightarrow & po\uparrow \\
\neg c0i \wedge \neg c1i & \rightarrow & po\downarrow
\end{array}
$$

Radix 2 transistor accounting:

rule	transistor count	comments
xe	6	
$u[0,1]$	12	
$c[0,1]_o$	8	
p_o	4	
total	30	

Radix 4 transistor accounting:

rule	transistor count	comments
xe	10	
$u[0,1,2,3]$	24	
$c[0,1,2,3]o$	16	
po	8	
total	58	

C.10 AERV ASPR BCAST pipelined

HSE

strict cpcp

$$\begin{array}{l}
BCAST \equiv \\
*[[\ p0 \longrightarrow c00\uparrow, c10\uparrow; po\uparrow; [c0i \wedge c1i \wedge \neg p0]; c00\downarrow, c10\downarrow; po\downarrow; [\neg c0i \wedge \neg c1i]; \\
\quad [] p1 \longrightarrow c01\uparrow, c11\uparrow; po\uparrow; [c0i \wedge c1i \wedge \neg p1]; c01\downarrow, c11\downarrow; po\downarrow; [\neg c0i \wedge \neg c1i]; \\
\quad [] pt \longrightarrow c0t\uparrow, c1t\uparrow; po\uparrow; [c0i \wedge c1i \wedge \neg pt]; c0t\downarrow, c1t\downarrow; po\downarrow; [\neg c0i \wedge \neg c1i]; \\
\ ]]
\end{array}$$

$$\begin{array}{l}
BCAST \equiv \\
*[[\ p0 \longrightarrow q0\uparrow; c00\uparrow, c10\uparrow; po\uparrow; [c0i \wedge c1i \wedge \neg p0]; q0\downarrow; c00\downarrow, c10\downarrow; po\downarrow; [\neg c0i \wedge \neg c1i] \\
\quad [] p1 \longrightarrow q1\uparrow; c01\uparrow, c11\uparrow; po\uparrow; [c0i \wedge c1i \wedge \neg p1]; q1\downarrow; c01\downarrow, c11\downarrow; po\downarrow; [\neg c0i \wedge \neg c1i] \\
\quad [] pt \longrightarrow qt\uparrow; c0t\uparrow, c1t\uparrow; po\uparrow; [c0i \wedge c1i \wedge \neg pt]; qt\downarrow; c0t\downarrow, c1t\downarrow; po\downarrow; [\neg c0i \wedge \neg c1i] \\
\ ]]
\end{array}$$

PRS

$$\begin{array}{lll} q0 & \rightarrow & c00\uparrow \\ \neg q0 & \rightarrow & c00\downarrow \end{array} \qquad \begin{array}{lll} q0 & \rightarrow & c10\uparrow \\ \neg q0 & \rightarrow & c10\downarrow \end{array}$$

$$\begin{array}{lll} q1 & \rightarrow & c01\uparrow \\ \neg q1 & \rightarrow & c01\downarrow \end{array} \qquad \begin{array}{lll} q1 & \rightarrow & c11\uparrow \\ \neg q1 & \rightarrow & c11\downarrow \end{array}$$

$$\begin{array}{lll} qt & \rightarrow & c0t\uparrow \\ \neg qt & \rightarrow & c0t\downarrow \end{array} \qquad \begin{array}{lll} qt & \rightarrow & c1t\uparrow \\ \neg qt & \rightarrow & c1t\downarrow \end{array}$$

$$\begin{array}{lll} \neg c0i \wedge \neg c1i \wedge p0 & \rightarrow & q0\uparrow \\ c0i \wedge c1i \wedge \neg p0 & \rightarrow & q0\downarrow \end{array} \qquad \begin{array}{lll} \neg c0i \wedge \neg c1i \wedge pt & \rightarrow & qt\uparrow \\ c0i \wedge c1i \wedge \neg pt & \rightarrow & qt\downarrow \end{array}$$

$$\begin{array}{lll} \neg c0i \wedge \neg c1i \wedge p1 & \rightarrow & q1\uparrow \\ c0i \wedge c1i \wedge \neg p1 & \rightarrow & q1\downarrow \end{array}$$

HSE

output ordering cpcp parallelized

$$\begin{array}{l} BCAST \equiv \\ *[[\ p0 \longrightarrow ([\neg c0i]; c00\uparrow), ([\neg c1i]; c10\uparrow); po\uparrow; [\neg p0]; ([c0i]; c00\downarrow), ([c1i]; c10\downarrow); po\downarrow; \\ \quad [\!]\, p1 \longrightarrow ([\neg c0i]; c01\uparrow), ([\neg c1i]; c11\uparrow); po\uparrow; [\neg p1]; ([c0i]; c01\downarrow), ([c1i]; c11\downarrow); po\downarrow; \\ \quad [\!]\, pt \longrightarrow ([\neg c0i]; c0t\uparrow), ([\neg c1i]; c1t\uparrow); po\uparrow; [\neg pt]; ([c0i]; c0t\downarrow), ([c1i]; c1t\downarrow); po\downarrow; \\ \ ]] \end{array}$$

PRS

$$\begin{array}{lll} \neg c0i \wedge p0 & \rightarrow & c00\uparrow \\ c0i \wedge \neg p0 & \rightarrow & c00\downarrow \end{array} \qquad \begin{array}{lll} \neg c1i \wedge p0 & \rightarrow & c10\uparrow \\ c1i \wedge \neg p0 & \rightarrow & c10\downarrow \end{array}$$

$$\begin{array}{lll} \neg c0i \wedge p1 & \rightarrow & c01\uparrow \\ c0i \wedge \neg p1 & \rightarrow & c01\downarrow \end{array} \qquad \begin{array}{lll} \neg c1i \wedge p1 & \rightarrow & c11\uparrow \\ c1i \wedge \neg p1 & \rightarrow & c11\downarrow \end{array}$$

$$\begin{array}{lll} \neg c0i \wedge pt & \rightarrow & c0t\uparrow \\ c0i \wedge \neg pt & \rightarrow & c0t\downarrow \end{array} \qquad \begin{array}{lll} \neg c1i \wedge pt & \rightarrow & c1t\uparrow \\ c1i \wedge \neg pt & \rightarrow & c1t\downarrow \end{array}$$

$$\begin{array}{lll} VN(C) & \rightarrow & po\uparrow \\ \neg VN(C) & \rightarrow & po\downarrow \end{array}$$

po is the output of a VN detector

HSE

swap ordering of p and c in reset output ordering cppc parallelized

$$
\begin{array}{l}
BCAST \equiv \\
*[[\ p0 \longrightarrow ([\neg c0i]; c00\uparrow), ([\neg c1i]; c10\uparrow); (po\uparrow; [\neg p0]; po\downarrow); ([c0i]; c00\downarrow), ([c1i]; c10\downarrow) \\
\quad [] p1 \longrightarrow ([\neg c0i]; c01\uparrow), ([\neg c1i]; c11\uparrow); (po\uparrow; [\neg p1]; po\downarrow); ([c0i]; c01\downarrow), ([c1i]; c11\downarrow) \\
\quad [] pt \longrightarrow ([\neg c0i]; c0t\uparrow), ([\neg c1i]; c1t\uparrow); (po\uparrow; [\neg pt]; po\downarrow); ([c0i]; c0t\downarrow), ([c1i]; c1t\downarrow) \\
\ ]]
\end{array}
$$

PRS

$$
\begin{array}{lll@{\qquad}lll}
\neg c0i \wedge p0 & \rightarrow & c00\uparrow & \neg c1i \wedge p0 & \rightarrow & c10\uparrow \\
c0i \wedge \neg p0 \wedge \neg po & \rightarrow & c00\downarrow & c1i \wedge \neg p0 \wedge \neg po & \rightarrow & c10\downarrow \\
\\
\neg c0i \wedge p1 & \rightarrow & c01\uparrow & \neg c1i \wedge p1 & \rightarrow & c11\uparrow \\
c0i \wedge \neg p1 \wedge \neg po & \rightarrow & c01\downarrow & c1i \wedge \neg p1 \wedge \neg po & \rightarrow & c11\downarrow \\
\\
\neg c0i \wedge pt & \rightarrow & c0t\uparrow & \neg c1i \wedge pt & \rightarrow & c1t\uparrow \\
c0i \wedge \neg pt \wedge \neg po & \rightarrow & c0t\downarrow & c1i \wedge \neg pt \wedge \neg po & \rightarrow & c1t\downarrow
\end{array}
$$

$$
\begin{array}{lll}
(c00 \wedge c10 \vee c01 \wedge c11 \vee c0t \wedge c1t) \wedge (p0 \vee p1 \vee pt) & \rightarrow & po\uparrow \\
\neg p0 \wedge \neg p1 \wedge \neg pt & \rightarrow & po\downarrow
\end{array}
$$

instability on down phases of c because p input can rise at anytime Could probably fix with state variables.

C.11 AERV ASPR BCAST unpipelined

HSE

$$
\begin{array}{l}
BCAST \equiv \\
*[[\ p0 \longrightarrow c00\uparrow, c10\uparrow; [c0i \wedge c1i]; po\uparrow; [\neg p0]; c00\downarrow, c10\downarrow; [\neg c0i \wedge \neg c1i]; po\downarrow \\
\quad [] p1 \longrightarrow c01\uparrow, c11\uparrow; [c0i \wedge c1i]; po\uparrow; [\neg p1]; c01\downarrow, c11\downarrow; [\neg c0i \wedge \neg c1i]; po\downarrow \\
\quad [] pt \longrightarrow c0t\uparrow, c1t\uparrow; [c0i \wedge c1i]; po\uparrow; [\neg pt]; c0t\downarrow, c1t\downarrow; [\neg c0i \wedge \neg c1i]; po\downarrow \\
\ ]]
\end{array}
$$

PRS

$$
\begin{array}{lll}
p0 & \rightarrow & c00\uparrow \\
\neg p0 & \rightarrow & c00\downarrow
\end{array}
\qquad
\begin{array}{lll}
p0 & \rightarrow & c10\uparrow \\
\neg p0 & \rightarrow & c10\downarrow
\end{array}
$$

$$
\begin{array}{lll}
p1 & \rightarrow & c01\uparrow \\
\neg p1 & \rightarrow & c01\downarrow
\end{array}
\qquad
\begin{array}{lll}
p1 & \rightarrow & c11\uparrow \\
\neg p1 & \rightarrow & c11\downarrow
\end{array}
$$

$$
\begin{array}{lll}
pt & \rightarrow & c0t\uparrow \\
\neg pt & \rightarrow & c0t\downarrow
\end{array}
\qquad
\begin{array}{lll}
pt & \rightarrow & c1t\uparrow \\
\neg pt & \rightarrow & c1t\downarrow
\end{array}
$$

$$
\begin{array}{lll}
c0i \wedge c1i & \rightarrow & po\uparrow \\
\neg c0i \wedge \neg c1i & \rightarrow & po\downarrow
\end{array}
$$

C.12 AERV PSAR

This makes the circuitry much simpler

C.13 AERV PSAR decomposed into ROUTE, READ_HEAD, FWD_BODY (RHB)

ROUTE sends a parent's signal to one of its children depending on which child requests. Assumes requests are mutually exclusive.

READ_HEAD reads the head word and signals FWD_BODY which way to forward the body packet

FWD_BODY forwards words to the children based on command from DEC

Radix 2 accounting (4095 nodes / 4096 neurons):

component	transistors/component	components/node	transistors/node
ROUTE	39	1	39
READ_HEAD	29	1	29
FWD_BODY	51	1	51
		total transistors/node	119

119 transistors/node * 4095 nodes / 4096 neurons = **119.0 transistors/neuron**

Radix 4 transistor accounting (1365 nodes / 4096 neurons):

component	transistors/component	components/node	transistors/node
ROUTE	71	1	71
READ_HEAD	53	1	53
FWD_BODY	125	1	125
		total transistors/node	249

249 transistors/node * 1365 nodes / 4096 neurons = **83.0 transistors/neuron**

However, we can still send a payload to the neurons with 1-of-2 data instead of 1-of-4 data at the leaf nodes. There are 1024 leaf nodes. This will simplify the leaf node ROUTE and FWD_BODY components because their children only need to see the 1-bit payload and tail. Each leaf node ROUTE can lose 1 bit (i.e. 2 data lines or 2 asymmetric c-elements or 14 transistors). Each leaf node FWD_BODY can lose 1 bit (i.e. 2 data lines or 2 AND-gates or 8 transistors) for each of 4 children. Therefore we can subtract

1024*(14+8*4) = 47104 transistors.

Leaving out the high bit from the leaf nodes yields

(249 transistors/node * 1365 nodes - 47104 transistors) / 4096 neurons = **71.5 transistors/neuron**

C.14 AERV PSAR RHB ROUTE unpipelined

Note that when communicating with READ_HEAD, ROUTE does not need to send the tail bit; READ_HEAD should never see a tail bit.

CHP

$$
\begin{array}{l}
*[[\overline{C0!};\, C0!(P?) \\
\quad [] \overline{H!};\, H!(P?)] \\
\;]
\end{array}
$$

HSE

$$
\begin{array}{l}
*[[c0e \vee he];\, pe\uparrow \\
\quad [\;\; p0 \wedge c0e \longrightarrow c00\uparrow;\, [\neg c0e];\, pe\downarrow;\, [\neg p0];\, c00\downarrow \\
\quad [] p1 \wedge c0e \longrightarrow c01\uparrow;\, [\neg c0e];\, pe\downarrow;\, [\neg p1];\, c01\downarrow \\
\quad [] pt \wedge c0e \longrightarrow c0t\uparrow;\, [\neg c0e];\, pe\downarrow;\, [\neg pt];\, c0t\downarrow \\
\quad [] p0 \wedge he \longrightarrow h0\uparrow;\, [\neg he];\, pe\downarrow;\, [\neg p0];\, h0\downarrow \\
\quad [] p1 \wedge he \longrightarrow h1\uparrow;\, [\neg he];\, pe\downarrow;\, [\neg p1];\, h1\downarrow \\
\quad] \\
\;]
\end{array}
$$

PRS

$$
\begin{aligned}
c0e \vee he &\rightarrow pe\uparrow \\
\neg c0e \wedge \neg he &\rightarrow pe\downarrow
\end{aligned}
$$

$$
\begin{aligned}
p0 \wedge c0e &\rightarrow c00\uparrow & \quad p0 \wedge he &\rightarrow h0\uparrow \\
\neg p0 &\rightarrow c00\downarrow & \quad \neg p0 &\rightarrow h0\downarrow
\end{aligned}
$$

$$
\begin{aligned}
p1 \wedge c0e &\rightarrow c01\uparrow & \quad p1 \wedge he &\rightarrow h1\uparrow \\
\neg p1 &\rightarrow c01\downarrow & \quad \neg p1 &\rightarrow h1\downarrow
\end{aligned}
$$

$$
\begin{aligned}
pt \wedge c0e &\rightarrow c0t\uparrow \\
\neg pt &\rightarrow c0t\downarrow
\end{aligned}
$$

Radix 2 accounting:

rule	transistor count	comments
pe	4	
$c0[0,1,t]$	21	
$h[0,1]$	14	
total	39	

Radix 4 transistor accounting:

rule	transistor count	comments
pe	8	
$c0[0,1,2,3,t]$	35	
$h[0,1,2,3]$	28	
total	71	

C.15 AERV PSAR RHB READ_HEAD

HSE

$$
\begin{aligned}
&*[[si];\, xe\uparrow; \\
&\quad [\ \ x0 \longrightarrow u0\uparrow;\, xe\downarrow;\, [\neg x0];\, s0\uparrow;\, [\neg si];\, u0\downarrow;\, s0\downarrow \\
&\quad [\!]\, x1 \longrightarrow u1\uparrow;\, xe\downarrow;\, [\neg x1];\, s1\uparrow;\, [\neg si];\, u1\downarrow;\, s1\downarrow \\
&\quad] \\
&\ ]
\end{aligned}
$$

PRS

$$
\begin{aligned}
si \wedge \neg u0 \wedge \neg u1 &\rightarrow xe\uparrow \\
u0 \vee u1 &\rightarrow xe\downarrow
\end{aligned}
$$

$$
\begin{array}{lll@{\qquad\qquad}lll}
x0 & \rightarrow & u0\uparrow & u0 \wedge \neg x0 & \rightarrow & s0\uparrow \\
\neg si & \rightarrow & u0\downarrow & \neg u0 \vee x0 & \rightarrow & s0\downarrow \\
\\
x1 & \rightarrow & u1\uparrow & u1 \wedge \neg x1 & \rightarrow & s1\uparrow \\
\neg si & \rightarrow & u1\downarrow & \neg u1 \vee x1 & \rightarrow & s1\downarrow
\end{array}
$$

Radix 2 accounting:

rule	transistor count	comments
xe	9	
$u[0,1]$	12	
$s[0,1]$	8	
total	29	

Radix 4 transistor accounting:

rule	transistor count	comments
xe	13	
$u[0,1,2,3]$	24	
$s[0,1,2,3]$	16	
total	53	

C.16 AERV PSAR RHB FWD_BODY unpipelined

HSE

$$
\begin{array}{l}
*\texttt{[[}\neg s0 \wedge \neg s1 \longrightarrow so\uparrow; \\
\quad \texttt{[]} s0 \longrightarrow \texttt{[}c0e\texttt{]}; pe\uparrow; \\
\qquad \texttt{[}\ \ p0 \longrightarrow c00\uparrow; \texttt{[}\neg c0e\texttt{]}; pe\downarrow; \texttt{[}\neg p0\texttt{]}; c00\downarrow \\
\qquad \texttt{[]} p1 \longrightarrow c01\uparrow; \texttt{[}\neg c0e\texttt{]}; pe\downarrow; \texttt{[}\neg p1\texttt{]}; c01\downarrow \\
\qquad \texttt{[]} pt \longrightarrow c0t\uparrow; \texttt{[}\neg c0e\texttt{]}; pe\downarrow; \texttt{[}\neg pt\texttt{]}; so\downarrow; \texttt{[}\neg s0\texttt{]}; c0t\downarrow \\
\qquad \texttt{]} \\
\quad \texttt{[]} s1 \longrightarrow \texttt{[}c1e\texttt{]}; pe\uparrow; \\
\qquad \texttt{[}\ \ p0 \longrightarrow c10\uparrow; \texttt{[}\neg c1e\texttt{]}; pe\downarrow; \texttt{[}\neg p0\texttt{]}; c10\downarrow \\
\qquad \texttt{[]} p1 \longrightarrow c11\uparrow; \texttt{[}\neg c1e\texttt{]}; pe\downarrow; \texttt{[}\neg p1\texttt{]}; c11\downarrow \\
\qquad \texttt{[]} pt \longrightarrow c1t\uparrow; \texttt{[}\neg c1e\texttt{]}; pe\downarrow; \texttt{[}\neg pt\texttt{]}; so\downarrow; \texttt{[}\neg s1\texttt{]}; c1t\downarrow \\
\qquad \texttt{]} \\
\quad\ \texttt{]} \\
\ \texttt{]}
\end{array}
$$

PRS

$$
\begin{array}{lll}
s0 \wedge c0e \vee s1 \wedge c1e & \rightarrow & pe\uparrow \\
s0 \wedge \neg c0e \vee s1 \wedge \neg c1e & \rightarrow & pe\downarrow
\end{array}
\qquad
\begin{array}{lll}
\neg c0t \wedge \neg c1t & \rightarrow & so\uparrow \\
\neg pt \wedge (c0t \vee c1t) & \rightarrow & so\downarrow
\end{array}
$$

$$
\begin{array}{lll}
p0 \wedge s0 & \rightarrow & c00\uparrow \\
\neg p0 \vee \neg s0 & \rightarrow & c00\downarrow
\end{array}
\qquad
\begin{array}{lll}
p0 \wedge s1 & \rightarrow & c10\uparrow \\
\neg p0 \vee \neg s1 & \rightarrow & c10\downarrow
\end{array}
$$

$$
\begin{array}{lll}
p1 \wedge s0 & \rightarrow & c01\uparrow \\
\neg p1 \vee \neg s0 & \rightarrow & c01\downarrow
\end{array}
\qquad
\begin{array}{lll}
p1 \wedge s1 & \rightarrow & c11\uparrow \\
\neg p1 \vee \neg s1 & \rightarrow & c11\downarrow
\end{array}
$$

$$
\begin{array}{lll}
pt \wedge s0 & \rightarrow & c0t\uparrow \\
\neg s0 & \rightarrow & c0t\downarrow
\end{array}
\qquad
\begin{array}{lll}
pt \wedge s1 & \rightarrow & c1t\uparrow \\
\neg s1 & \rightarrow & c1t\downarrow
\end{array}
$$

Radix 2 accounting:

rule	transistor count	comments
pe	12	
s_o	9	
$c[0,1][0,1]$	16	
$c[0,1]t$	14	
total	51	

Radix 4 transistor accounting:

rule	transistor count	comments
pe	20	
s_o	13	
$c[0,1,2,3][0,1,2,3]$	64	
$c[0,1,2,3]t$	28	
total	125	

C.17 AERV PSAR RHB FWD_BODY pipelined

HSE

$$
\begin{array}{l}
*[\,so\uparrow;\ [s0 \vee s1];\ pe\uparrow; \\
\quad [\ \ p0 \wedge s0 \wedge c0e \longrightarrow c00\uparrow;\ pe\downarrow;\ [\neg p0 \wedge \neg c0e];\ c00\downarrow \\
\quad [\!]\, p1 \wedge s0 \wedge c0e \longrightarrow c01\uparrow;\ pe\downarrow;\ [\neg p1 \wedge \neg c0e];\ c01\downarrow \\
\quad [\!]\, pt \wedge s0 \wedge c0e \longrightarrow c0t\uparrow;\ pe\downarrow;\ [\neg pt \wedge \neg c0e];\ so\downarrow;\ [\neg s0];\ c0t\downarrow \\
\quad [\!]\, p0 \wedge s1 \wedge c1e \longrightarrow c10\uparrow;\ pe\downarrow;\ [\neg p0 \wedge \neg c1e];\ c10\downarrow \\
\quad [\!]\, p1 \wedge s1 \wedge c1e \longrightarrow c11\uparrow;\ pe\downarrow;\ [\neg p1 \wedge \neg c1e];\ c11\downarrow \\
\quad [\!]\, pt \wedge s1 \wedge c1e \longrightarrow c1t\uparrow;\ pe\downarrow;\ [\neg pt \wedge \neg c1e];\ so\downarrow;\ [\neg s1];\ c1t\downarrow \\
\quad] \\
]
\end{array}
$$

PRS

$$
\begin{array}{lcl}
s0 \vee s1 \wedge \neg q & \rightarrow & pe\uparrow \\
q & \rightarrow & pe\downarrow
\end{array}
\qquad
\begin{array}{lcl}
c00 \vee c01 \vee c0t \vee c10 \vee c11 \vee c1t & \rightarrow & q\uparrow \\
\neg c00 \wedge \neg c01 \wedge \neg c0t \wedge \neg c10 \wedge \neg c11 \wedge \neg c1t & \rightarrow & q\downarrow
\end{array}
$$

$$
\begin{array}{lcl}
\neg q & \rightarrow & so\uparrow \\
\neg pt \wedge (\neg c0e \wedge c0t \vee \neg c1e \wedge c1t) & \rightarrow & so\downarrow
\end{array}
$$

$$
\begin{array}{lcl}
p0 \wedge s0 \wedge c0e & \rightarrow & c00\uparrow \\
\neg p0 \wedge \neg c0e & \rightarrow & c00\downarrow \\
\\
p1 \wedge s0 \wedge c0e & \rightarrow & c01\uparrow \\
\neg p1 \wedge \neg c0e & \rightarrow & c01\downarrow \\
\\
pt \wedge s0 \wedge c0e & \rightarrow & c0t\uparrow \\
\neg s0 & \rightarrow & c0t\downarrow
\end{array}
\qquad
\begin{array}{lcl}
p0 \wedge s1 \wedge c1e & \rightarrow & c10\uparrow \\
\neg p0 \wedge \neg c0e & \rightarrow & c10\downarrow \\
\\
p1 \wedge s1 \wedge c1e & \rightarrow & c11\uparrow \\
\neg p1 \wedge \neg c0e & \rightarrow & c11\downarrow \\
\\
pt \wedge s1 \wedge c1e & \rightarrow & c1t\uparrow \\
\neg s1 & \rightarrow & c1t\downarrow
\end{array}
$$

Radix 2 accounting:

rule	transistor count	comments
total	90	

Radix 4 transistor accounting:

rule	transistor count	comments
total	262	

C.18 AERV PSAR decomposed into ROUTE PULL_CTRL PULL (RCP)

ROUTE sends a parent's signal to one of its children depending on which child requests. Assumes requests are mutually exclusive. This is same ROUTE as above.

PULL_CTRL reads the head word and indicates which PULL should request data from ROUTE.
PULL requests data from ROUTE and passes the data to the child.

Radix 2 accounting (4095 nodes / 4096 neurons):

component	transistors/component	components/node	transistors/node
ROUTE	62	1	62
PULL_CTRL	28	1	28
PULL	14	2	28
total transistors/node			118

118 transistors/node * 4095 nodes / 4096 neurons = **118.0 transistors/neuron**
Radix 4 transistor accounting (1365 nodes / 4096 neurons):

component	transistors/component	components/node	transistors/node
ROUTE	178	1	178
PULL_CTRL	58	1	58
PULL	14	4	56
total transistors/node			292

292 transistors/node * 1365 nodes / 4096 neurons = **97.3 transistors/neuron**

However, we can still send a payload to the neurons with 1-of-2 data instead of 1-of-4 data at the leaf nodes. There are 1024 leaf nodes. This will simplify the leaf node ROUTE component because its children only need to see the 1-bit payload and tail. Each leaf node ROUTE can lose 1 bit (i.e. 2 data lines or 2 asymmetric c-elements or 14 transistors) for each of 4 children. Therefore we can subtract 1024*14*4 = 57344 transistors
Leaving out the high bit from the leaf nodes yields

(292 transistors/node * 1365 nodes - 57344 transistors)/ 4096 neurons = **83.3 transistors/neuron**

C.19 AERV PSAR RCP ROUTE

This decomposition largely reuses the unpipelined ROUTE in Section C.14 above. For this decomposition, ROUTE connects to [radix] instances of PULL and 1 instance of PULL_CTRL.
Radix 2 accounting:

rule	transistor count	comments
pe	6	
$c[0,1][0,1,t]$	42	
$h[0,1]$	14	
total	62	

Radix 4 transistor accounting:

rule	transistor count	comments
pe	10	
$c[0,1,2,3][0,1,2,3,t]$	140	
$h[0,1,2,3]$	28	
total	178	

C.20 AERV PSAR PULL_CTRL

CHP

HSE

$$\begin{array}{l} *[[s0i \wedge s1i];\, xe\uparrow; \\ \quad [\;\; x0 \longrightarrow u0\uparrow;\, xe\downarrow;\, [\neg x0];\, s0o\uparrow;\, [\neg s0i];\, u0\downarrow;\, s0o\downarrow \\ \quad [\!]\, x1 \longrightarrow u1\uparrow;\, xe\downarrow;\, [\neg x1];\, s1o\uparrow;\, [\neg s0i];\, u1\downarrow;\, s0o\downarrow \\ \quad]] \end{array}$$

PRS

$$\begin{array}{lll} s0i \wedge s1i \wedge q & \to & xe\uparrow \\ \neg s0i \vee \neg s1i \vee \neg q & \to & xe\downarrow \end{array} \qquad \begin{array}{lll} \neg u0 \wedge \neg u1 & \to & q\uparrow \\ u0 \vee u1 & \to & q\downarrow \end{array}$$

$$\begin{array}{lll} x0 & \to & u0\uparrow \\ \neg s0i & \to & u0\downarrow \end{array} \qquad \begin{array}{lll} x1 & \to & u1\uparrow \\ \neg s1i & \to & u1\downarrow \end{array}$$

$$\begin{array}{lll} u0 \wedge \neg x0 & \to & s0o\uparrow \\ \neg u0 \vee x0 & \to & s0o\downarrow \end{array} \qquad \begin{array}{lll} u1 \wedge \neg x1 & \to & s1o\uparrow \\ \neg u1 \vee x1 & \to & s1o\downarrow \end{array}$$

Radix 2 accounting:

rule	transistor count	comments
xe	8	no q
$u[0,1]$	12	
$s[0,1]_o$	8	
total	28	

Radix 4 transistor accounting:

rule	transistor count	comments
xe	10	
q	8	
$u[0,1,2,3]$	24	
$s[0,1,2,3]_o$	16	
total	58	

C.21 AERV PSAR RCP PULL unpipelined

HSE

$$
\begin{array}{l}
*[\neg si \longrightarrow so\uparrow;\\
\ \ \| si \longrightarrow [ye]; xe\uparrow\\
\ \ \ \ [\ \ x0 \longrightarrow y0\uparrow; [\neg ye]; xe\downarrow; [\neg x0]; y0\downarrow\\
\ \ \ \ \| x1 \longrightarrow y1\uparrow; [\neg ye]; xe\downarrow; [\neg x1]; y1\downarrow\\
\ \ \ \ \| xt \longrightarrow yt\uparrow; [\neg ye]; xe\downarrow; [\neg xt]; so\downarrow; [\neg si]; yt\downarrow\\
\ \ \ \ ]\\
\ \ ]
\end{array}
$$

PRS

$$
\begin{array}{lll}
\neg yt \wedge \neg si & \rightarrow & so\uparrow\\
yt \wedge \neg xt & \rightarrow & so\downarrow
\end{array}
\qquad\qquad
\begin{array}{lll}
si \wedge ye & \rightarrow & xe\uparrow\\
\neg si \vee \neg ye & \rightarrow & xe\downarrow
\end{array}
$$

$$
\begin{array}{lll}
x0 & \rightarrow & y0\uparrow\\
\neg x0 & \rightarrow & y0\downarrow
\end{array}
\qquad\qquad
\begin{array}{lll}
xt & \rightarrow & yt\uparrow\\
\neg si & \rightarrow & yt\downarrow
\end{array}
$$

$$
\begin{array}{lll}
x1 & \rightarrow & y1\uparrow\\
\neg x1 & \rightarrow & y1\downarrow
\end{array}
$$

Radix 2 transistor accounting:

rule	transistor count	comments
s_o	4	
xe	4	
$y[0,1]$	0	wires
yt	6	
total	14	

Radix 4 transistor accounting:

rule	transistor count	comments
s_o	4	
xe	4	
$y[0,1,2,3]$	0	wires
yt	6	
total	14	

Appendix D

AER Interface Design Space

The router interfaces include conversions between the serial protocol as well as deserializers and serializers.

D.1 OUT e1ofN

Interfaces AEXT/AERV serial format to e1ofN channel.

$$*[[xi \wedge ye]; xo\uparrow; [\neg xi \wedge \neg ye]; xo\downarrow]$$

$$\begin{array}{l} *[[x0 \longrightarrow y0\uparrow; [\neg x0]; y0\downarrow \\ \;\; [] x1 \longrightarrow y1\uparrow; [\neg x1]; y1\downarrow \\ \;]] \end{array}$$

PRS

$$\begin{array}{lcl} xi \wedge ye & \rightarrow & xo\uparrow \\ \neg xi \vee \neg ye & \rightarrow & xo\downarrow \end{array}$$

$$\begin{array}{lclclcl} x0 & \rightarrow & y0\uparrow & \quad & x1 & \rightarrow & y1\uparrow \\ \neg x0 & \rightarrow & y0\downarrow & \quad & \neg x1 & \rightarrow & y1\downarrow \end{array}$$

CMOS-implementable PRS version 0

$$\begin{array}{lclclcl} xi \wedge ye & \rightarrow & \_xo\downarrow & \quad & \neg\_xo & \rightarrow & xo\downarrow \\ \neg xi \vee \neg ye & \rightarrow & \_xo\uparrow & \quad & \_xo & \rightarrow & xo\uparrow \end{array}$$

$$\begin{array}{lclclcl} x0 & \rightarrow & y0\uparrow & \quad & x1 & \rightarrow & y1\uparrow \\ \neg x0 & \rightarrow & y0\downarrow & \quad & \neg x1 & \rightarrow & y1\downarrow \end{array}$$

CMOS-implementable PRS version 1

$$
\begin{array}{lcl@{\qquad\qquad}lcl}
xi \wedge ye & \rightarrow & \_xo\downarrow & \neg\_xo & \rightarrow & xo\downarrow \\
\neg xi \vee \neg ye & \rightarrow & \_xo\uparrow & \_xo & \rightarrow & xo\uparrow
\end{array}
$$

$$
\begin{array}{lcl@{\qquad\qquad}lcl}
\neg\_x0 & \rightarrow & y0\uparrow & \neg\_x1 & \rightarrow & y1\uparrow \\
\_x0 & \rightarrow & y0\downarrow & \_x1 & \rightarrow & y1\downarrow
\end{array}
$$

Radix 2 transistor accounting:

rule	transistor count	comments
xo	4	
$y[0,1]$	0	wires
total	4	

Radix 4 transistor accounting:

rule	transistor count	comments
xo	4	
$y[0,1]$	0	wires
total	4	

D.2 OUT a1ofN

Interfaces AEXT/AERV serial format to a1ofN channel.

$$*[[xi];\, xo\uparrow;\, [\neg xi];\, xo\downarrow]$$

$$
\begin{array}{l}
*[[x0 \longrightarrow y0\uparrow;\, [ya];\, xo\downarrow;\, [\neg x0];\, y0\downarrow;\, [\neg ya];\, xo\uparrow \\
\;\;[]\, x1 \longrightarrow y1\uparrow;\, [ya];\, xo\downarrow;\, [\neg x1];\, y1\downarrow;\, [\neg ya];\, xo\uparrow \\
\;]]
\end{array}
$$

PRS

$$
\begin{array}{lcl}
xi \wedge \neg ya & \rightarrow & xo\uparrow \\
\neg xi \vee ya & \rightarrow & xo\downarrow
\end{array}
$$

$$
\begin{array}{lcl@{\qquad\qquad}lcl}
x0 & \rightarrow & y0\uparrow & x1 & \rightarrow & y1\uparrow \\
\neg x0 & \rightarrow & y0\downarrow & \neg x1 & \rightarrow & y1\downarrow
\end{array}
$$

CMOS-implementable PRS version 0

$$
\begin{array}{lcl@{\qquad\qquad}lcl}
xi \wedge \_ya & \rightarrow & \_xo\downarrow & \neg\_xo & \rightarrow & xo\uparrow \\
\neg xi \vee \neg\_ya & \rightarrow & \_xo\uparrow & \_xo & \rightarrow & xo\downarrow
\end{array}
$$

$$
\begin{array}{llll}
\_x0 \rightarrow \_y0\downarrow & \qquad & \_x1 \rightarrow \_y1\downarrow \\
\neg\_x0 \rightarrow \_y0\uparrow & \qquad & \neg\_x1 \rightarrow \_y1\uparrow
\end{array}
$$

Radix 2 transistor accounting:

rule	transistor count	comments
$\_x_o$	4	
x_o	2	
$\_y[0,1]$	0	wires
total	6	

Radix 4 transistor accounting:

rule	transistor count	comments
$\_x_o$	4	
x_o	2	
$\_y[0,1]$	0	wires
total	6	

D.3 Deserializer

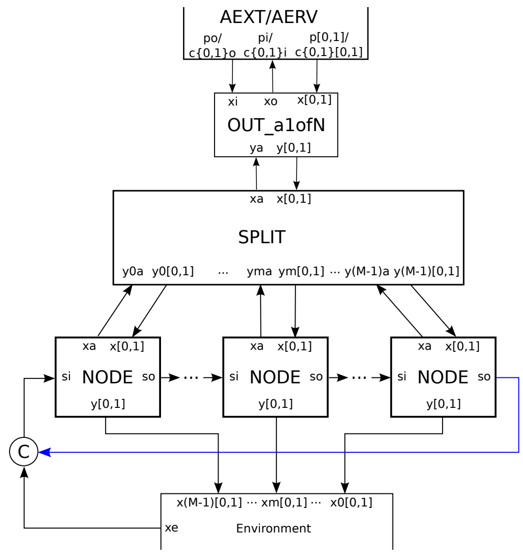
The deserializer converts 1-of-N serial data into M-1-of-N parallel data.

```
*[X?y_m;
    [m < M − 1 ⟶ m := m + 1
    []m = M − 1 ⟶ Y!y; m := 0
    ]
 ]
```

We place a deserializer at output of the transmitter to interface with the datapath circuitry. It is the first in a series of processes that communicate with the outside environment:
transmitter → deserializer → 1-of-4-to-1-of-2 converter (if needed) → [Datapath] → serializer → receiver
We also place a deserializer at the output of the receiver to interface with the neuron configuration memory.

D.4 Ring Deserializer

This design uses a ring of nodes receiving data from a central splitter to sequence words into their respective place in the parallel output. We decompose this process into SPLIT and NODE. This design has a slightly cheaper 1-of-4 implementation than the chain deserializer of Section D.9. However, the data signals in SPLIT and the environment enable signal have fanouts that grow with the number of words. The below figure shows the decomposition for packets containing M 1-of-2 words.



An OUT a1ofN process (if necessary and described above) first converts the AEXT/AERV serial communication protocol to the standard a1ofN protocol.

1-of-2 approximate scaling:

component	transistors/component	components/deserializer	transistors/deserializer
OUT a1ofN	4	1	4
SPLIT	$3M-2$	1	$3M-2$
NODE	32	M	$32M$
C	8	1	8
		approx. transistors/deserializer	$35M+10$

1-of-4 approximate scaling:

component	transistors/component	components/deserializer	transistors/deserializer
OUT a1ofN	4	1	4
SPLIT	$3M-2$	1	$3M-2$
NODE	54	M	$54M$
C	8	1	8
		approx. transistors/deserializer	$57M+10$

For the transmitter to handle 4096 neurons encoded as 1-of-2 or 1-of-4 words, we would need 12 and 6 NODEs, respectively.

1-of-2 accounting:

component	transistors/component	components/deserializer	transistors/deserializer
OUT a1ofN	4	1	4
SPLIT	30	1	30
NODE	32	12	384
C	8	1	8
		total transistors/deserializer	426

1-of-4 accounting:

component	transistors/component	components/deserializer	transistors/deserializer
OUT a1ofN	4	1	4
SPLIT	16	1	16
NODE	54	6	324
C	8	1	8
		total transistors/deserializer	352

D.5 SPLIT

SPLIT takes incoming words and routes them to their respective locations in the parallel output. For M words per packet,

$$\begin{aligned}
&*[[x0 \longrightarrow y00\uparrow, .., y(M-1)0\uparrow; [y0a \vee .. \vee y(M-1)a]; xa\uparrow;\\
&\quad [\neg x0]; y00\downarrow, .., y(M-1)1\downarrow; [\neg y0a \wedge .. \wedge \neg y(M-1)a]; xa\downarrow\\
&\;\; [\!]\, x1 \longrightarrow y01\uparrow, .., y(M-1)1\uparrow; [y0a \vee .. \vee y(M-1)a]; xa\uparrow;\\
&\quad [\neg x0]; y01\downarrow, .., y(M-1)1\downarrow; [\neg y0a \wedge .. \wedge \neg y(M-1)a]; xa\downarrow\\
&\;]]
\end{aligned}$$

For a 2-word packet,

$$\begin{aligned}
&*[[x0 \longrightarrow y00\uparrow, y10\uparrow; [y0a \vee y1a]; xa\uparrow;\\
&\quad [\neg x0]; y00\downarrow, y01\downarrow; [\neg y0a \wedge \neg y1a]; xa\downarrow\\
&\;\; [\!]\, x1 \longrightarrow y01\uparrow, y11\uparrow; [y0a \vee y1a]; xa\uparrow;\\
&\quad [\neg x0]; y01\downarrow, y11\downarrow; [\neg y0a \wedge \neg y1a]; xa\downarrow\\
&\;]]
\end{aligned}$$

PRS

$$\begin{array}{lll@{\qquad\qquad}lll}
x0 & \rightarrow & y00\uparrow & x1 & \rightarrow & y01\uparrow\\
\neg x0 & \rightarrow & y00\downarrow & \neg x1 & \rightarrow & y01\downarrow\\
\\
x0 & \rightarrow & y10\uparrow & x1 & \rightarrow & y11\uparrow\\
\neg x0 & \rightarrow & y10\downarrow & \neg x1 & \rightarrow & y11\downarrow
\end{array}$$

$$y0a \vee y1a \rightarrow xa\uparrow$$
$$\neg y0a \wedge \neg y1a \rightarrow xa\downarrow$$

1-of-2 transistor approximate accounting:

rule	transistor count	comments
$y[0..M-1][0,1]$	0	wires
xa	$8(M-1)/3$	4-ary OR-tree approx.
approx. total	$3M-2$	

1-of-4 transistor approximate accounting:

rule	transistor count	comments
$y[0..M-1][0,1,2,3]$	0	wires
xa	$8(M-1)/3$	4-ary OR-tree approx.
approx. total	$3M-2$	

CMOS-implementable PRS

$$\_x0 \rightarrow \_y00\uparrow \qquad \_x1 \rightarrow \_y01\uparrow$$
$$\neg\_x0 \rightarrow \_y00\downarrow \qquad \neg\_x1 \rightarrow \_y01\downarrow$$

$$\_x0 \rightarrow \_y10\uparrow \qquad \_x1 \rightarrow \_y11\uparrow$$
$$\neg\_x0 \rightarrow \_y10\downarrow \qquad \neg\_x1 \rightarrow \_y11\downarrow$$

$$y0a \vee y1a \rightarrow \_xa\downarrow$$
$$\neg y0a \wedge \neg y1a \rightarrow \_xa\uparrow$$

$$\neg\_y00 \rightarrow y00\uparrow \qquad \neg\_y01 \rightarrow y01\uparrow$$
$$\_y00 \rightarrow y00\downarrow \qquad \_y01 \rightarrow y01\downarrow$$

$$\neg\_y10 \rightarrow y10\uparrow \qquad \neg\_y11 \rightarrow y11\uparrow$$
$$\_y10 \rightarrow y10\downarrow \qquad \_y11 \rightarrow y11\downarrow$$

D.6 NODE

NODE latches data from SPLIT.

$$
\begin{array}{l}
*[[si];\\
\quad [x0 \longrightarrow y0\uparrow; xa\uparrow; [\neg x0]; s\uparrow; so\uparrow; xa\downarrow; [\neg si]; y0\downarrow; s\downarrow; so\downarrow\\
\quad [\!] x1 \longrightarrow y1\uparrow; xa\uparrow; [\neg x1]; s\uparrow; so\uparrow; xa\downarrow; [\neg si]; y1\downarrow; s\downarrow; so\downarrow\\
\quad]\\
]
\end{array}
$$

The s state variable is necessary for bubble reshuffling. It breaks a cycle of isochronic branches with an odd number of bubbles (See Section D.8), which would be impossible to make CMOS-implementable.

PRS

$$\begin{array}{lll} \neg s \wedge si \wedge x0 & \rightarrow & y0\uparrow \\ \neg si & \rightarrow & y0\downarrow \end{array} \qquad \begin{array}{lll} \neg s \wedge si \wedge x1 & \rightarrow & y1\uparrow \\ \neg si & \rightarrow & y1\downarrow \end{array}$$

$$\begin{array}{lll} \neg so \wedge vy & \rightarrow & xa\uparrow \\ so \vee \neg vy & \rightarrow & xa\downarrow \end{array}$$

$$\begin{array}{lll} vy \wedge \neg x0 \wedge \neg x1 & \rightarrow & s\uparrow \\ \neg vy & \rightarrow & s\downarrow \end{array}$$

$$\begin{array}{lll} s & \rightarrow & so\uparrow \\ \neg s & \rightarrow & so\downarrow \end{array}$$

$$\begin{array}{lll} y0 \vee y1 & \rightarrow & vy\uparrow \\ \neg y0 \wedge \neg y1 & \rightarrow & vy\downarrow \end{array}$$

1-of-2 transistor approximate accounting:

rule	transistor count	comments
$y[0,1][0,1]$	16	
xa	4	
s_o	8	
vy	4	
total	32	

1-of-4 transistor approximate accounting:

rule	transistor count	comments
$y[0,1,2,3][0,1,2,3]$	32	
xa	4	
s_o	10	
vy	8	
total	54	

CMOS-implementable PRS

$$\begin{array}{lll} \neg\_s & \rightarrow & \_\_s\uparrow \\ \_s & \rightarrow & \_\_s\downarrow \end{array}$$

$$\neg\_\_s \wedge \neg\_si \wedge \neg\_x0 \rightarrow y0\uparrow \qquad \neg\_\_s \wedge \neg\_si \wedge \neg\_x1 \rightarrow y1\uparrow$$
$$\_si \rightarrow y0\downarrow \qquad \_si \rightarrow y1\downarrow$$

$$\_so \wedge vy \rightarrow \_xa\downarrow$$
$$\neg\_so \vee \neg vy \rightarrow \_xa\uparrow$$

$$vy \wedge \_x0 \wedge \_x1 \rightarrow \_s\downarrow$$
$$\neg vy \rightarrow \_s\uparrow$$

$$\_\_s \rightarrow \_so\downarrow$$
$$\neg\_\_s \rightarrow \_so\uparrow$$

$$\neg y0 \rightarrow \_y0\uparrow \qquad \neg y1 \rightarrow \_y1\uparrow$$
$$y0 \rightarrow \_y0\downarrow \qquad y1 \rightarrow \_y1\downarrow$$

$$\neg\_y0 \vee \neg\_y1 \rightarrow vy\uparrow$$
$$\_y0 \wedge \_y1 \rightarrow vy\downarrow$$

D.7 C

C in the ring deserializer of Section D.4 is a C-element taking in the environment enable signal and the last node's s_o signal to produce first node's s_i signal. s_i indicates whether we are in the up or down phase of the serial-to-parallel conversion.

PRS

$$\neg so \wedge xe \rightarrow si\uparrow$$
$$so \wedge \neg xe \rightarrow si\downarrow$$

A C-element costs 8 transistors.

CMOS-implementable PRS

$$\_so \wedge xe \rightarrow \_si\downarrow$$
$$\neg\_so \wedge \neg xe \rightarrow \_si\uparrow$$

D.8 RING

In the interest of bubble reshuffling, the deserial ring NODEs and C-element should be described in a single process.

$$
\begin{array}{l}
*[[ye];s0\uparrow;\\
\quad [x0 \longrightarrow y00\uparrow;x0a\uparrow;[\neg x0]\\
\quad [\!]x1 \longrightarrow y01\uparrow;x0a\uparrow;[\neg x1]\\
\quad];s01\uparrow;s1\uparrow;x0a\downarrow;\\
\quad ...\\
\quad [x0 \longrightarrow ym0\uparrow;xma\uparrow;[\neg x0]\\
\quad [\!]x1 \longrightarrow ym1\uparrow;xma\uparrow;[\neg x1]\\
\quad];sm(m+1)\uparrow;s(m+1)\uparrow;xma\downarrow;\\
\quad ...\\
\quad [x0 \longrightarrow y(M-1)0\uparrow;x(M-1)a\uparrow;[\neg x0]\\
\quad [\!]x1 \longrightarrow y(M-1)1\uparrow;x(M-1)a\uparrow;[\neg x1]\\
\quad];s(M-1)M\uparrow;sM\uparrow;x(M-1)a\downarrow;\\
\quad [\neg ye];s0\downarrow;\\
\quad y00\downarrow,y01\downarrow;s01\downarrow;s1\downarrow;\\
\quad ...\\
\quad ym0\downarrow,ym1\downarrow;sm(m+1)\downarrow;s(m+1)\downarrow\\
\quad ...\\
\quad y(M-1)0\downarrow,y(M-1)1\downarrow;s(M-1)M\downarrow;sM\downarrow\\
]
\end{array}
$$

Recall in Section D.6 that we had an apparently extraneous state variable s. If we removed the state variable, the NODE processes would share share isochronic branches, which would create a cycle of isochronic branches among NODES. Further, this cycle would have an odd number of bubbles and be impossible to make CMOS-implementable.

PRS

Y data:

$$
\begin{array}{lcl@{\qquad\qquad}lcl}
\neg s01 \wedge s0 \wedge x0 & \rightarrow & y00\uparrow & \neg s01 \wedge s0 \wedge x1 & \rightarrow & y01\uparrow\\
\neg s0 & \rightarrow & y00\downarrow & \neg s0 & \rightarrow & y01\downarrow
\end{array}
$$

...

$$
\begin{array}{lcl@{\qquad\qquad}lcl}
\neg sm(m+1) \wedge sm \wedge x0 & \rightarrow & ym0\uparrow & \neg sm(m+1) \wedge sm \wedge x1 & \rightarrow & ym1\uparrow\\
\neg sm & \rightarrow & ym0\downarrow & \neg sm & \rightarrow & ym1\downarrow
\end{array}
$$

...

$$
\begin{array}{lcl@{\qquad\qquad}lcl}
\neg s(M-1)M \wedge s(M-1) \wedge x0 & \rightarrow & y(M-1)0\uparrow & \neg s(M-1)M \wedge s(M-1) \wedge x1 & \rightarrow & y(M-1)1\uparrow\\
\neg s(M-1) & \rightarrow & y(M-1)0\downarrow & \neg s(M-1) & \rightarrow & y(M-1)1\downarrow
\end{array}
$$

X acknowledge:

$$\neg s01 \wedge v0y \;\rightarrow\; x0a\!\uparrow$$
$$s01 \vee \neg v0y \;\rightarrow\; x0a\!\downarrow$$

...

$$\neg sm(m+1) \wedge vmy \;\rightarrow\; xma\!\uparrow$$
$$sm(m+1) \vee \neg vmy \;\rightarrow\; xma\!\downarrow$$

...

$$\neg s(M-1)M \wedge v(M-1)y \;\rightarrow\; x(M-1)a\!\uparrow$$
$$s(M-1)M \vee \neg v(M-1)y \;\rightarrow\; x(M-1)a\!\downarrow$$

S buffer states:

$$v0y \wedge \neg x0 \wedge \neg x1 \;\rightarrow\; s01\!\uparrow$$
$$\neg v0y \;\rightarrow\; s01\!\downarrow$$

...

$$vmy \wedge \neg x0 \wedge \neg x1 \;\rightarrow\; sm(m+1)\!\uparrow$$
$$\neg vmy \;\rightarrow\; sm(m+1)\!\downarrow$$

...

$$v(M-1)y \wedge \neg x0 \wedge \neg x1 \;\rightarrow\; s(M-1)M\!\uparrow$$
$$\neg v(M-1)y \;\rightarrow\; s(M-1)M\!\downarrow$$

S input states:

$$\neg sM \wedge ye \;\rightarrow\; s0\!\uparrow$$
$$sM \wedge \neg ye \;\rightarrow\; s0\!\downarrow$$

$$s01 \;\rightarrow\; s1\!\uparrow$$
$$\neg s01 \;\rightarrow\; s1\!\downarrow$$

...

$$sm(m+1) \;\rightarrow\; s(m+1)\!\uparrow$$
$$\neg sm(m+1) \;\rightarrow\; s(m+1)\!\downarrow$$

...

$$s(M-1)M \;\rightarrow\; sM\!\uparrow$$
$$\neg s(M-1)M \;\rightarrow\; sM\!\downarrow$$

Y valid detectors:

$$y00 \vee y01 \;\rightarrow\; v0y\!\uparrow$$
$$\neg y00 \wedge \neg y01 \;\rightarrow\; v0y\!\downarrow$$

...

$$\begin{array}{lcl} ym0 \vee ym1 & \rightarrow & vmy\uparrow \\ \neg ym0 \wedge \neg ym1 & \rightarrow & vmy\downarrow \end{array}$$

...

$$\begin{array}{lcl} y(M-1)0 \vee y(M-1)1 & \rightarrow & v(M-1)y\uparrow \\ \neg y(M-1)0 \wedge \neg y(M-1)1 & \rightarrow & v(M-1)y\downarrow \end{array}$$

CMOS-implementable PRS

Y data:

$$\begin{array}{lcl} \neg\_s01 & \rightarrow & \_\_s01\uparrow \\ \_s01 & \rightarrow & \_\_s01\downarrow \end{array}$$

$$\begin{array}{lcl} \neg\_\_s01 \wedge \neg\_s0 \wedge \neg\_x0 & \rightarrow & y00\uparrow \\ \_s0 & \rightarrow & y00\downarrow \end{array} \qquad \begin{array}{lcl} \neg\_\_s01 \wedge \neg\_s0 \wedge \neg\_x1 & \rightarrow & y01\uparrow \\ \_s0 & \rightarrow & y01\downarrow \end{array}$$

...

$$\begin{array}{lcl} \neg\_sm(m+1) & \rightarrow & \_\_sm(m+1)\uparrow \\ \_sm(m+1) & \rightarrow & \_\_sm(m+1)\downarrow \end{array}$$

$$\begin{array}{lcl} \neg\_\_sm(m+1) \wedge \neg\_sm \wedge \neg\_x0 & \rightarrow & ym0\uparrow \\ \_sm & \rightarrow & ym0\downarrow \end{array} \qquad \begin{array}{lcl} \neg\_\_sm(m+1) \wedge \neg\_sm \wedge \neg\_x1 & \rightarrow & ym1\uparrow \\ \_sm & \rightarrow & ym1\downarrow \end{array}$$

...

$$\begin{array}{lcl} \neg\_s(M-1)M & \rightarrow & \_\_s(M-1)M\downarrow \\ \_s(M-1)M & \rightarrow & \_\_s(M-1)M\downarrow \end{array}$$

$$\begin{array}{lcl} \neg\_\_s(M-1)M \wedge \neg\_s(M-1) \wedge \neg\_x0 & \rightarrow & y(M-1)0\uparrow \\ \_s(M-1) & \rightarrow & y(M-1)0\downarrow \end{array} \qquad \begin{array}{lcl} \neg\_\_s(M-1)M \wedge \neg\_s(M-1) \wedge \neg\_x1 & \rightarrow & y(M - \\ \_s(M-1) & \rightarrow & y(M - \end{array}$$

X acknowledge:

$$\begin{array}{lcl} \neg\_\_s01 \wedge \neg\_v0y & \rightarrow & x0a\uparrow \\ \_\_s01 \vee \_v0y & \rightarrow & x0a\downarrow \end{array}$$

...

$$\begin{array}{lcl} \neg\_\_sm(m+1) \wedge \neg\_vmy & \rightarrow & xma\uparrow \\ \_\_sm(m+1) \vee \_vmy & \rightarrow & xma\downarrow \end{array}$$

...

$$\begin{array}{lcl}
\neg__s(M-1)M \wedge \neg_v(M-1)y & \rightarrow & x(M-1)a\uparrow \\
__s(M-1)M \vee _v(M-1)y & \rightarrow & x(M-1)a\downarrow
\end{array}$$

S buffer states:

$$\begin{array}{lcl}
\neg_v0y & \rightarrow & __v0y \\
_v0y & \rightarrow & __v0y
\end{array}
\qquad
\begin{array}{lcl}
__v0y \wedge _x0 \wedge _x1 & \rightarrow & _s01\downarrow \\
___v0y & \rightarrow & _s01\uparrow
\end{array}$$

...

$$\begin{array}{lcl}
\neg_vmy & \rightarrow & __vmy \\
_vmy & \rightarrow & __vmy
\end{array}
\qquad
\begin{array}{lcl}
__vmy \wedge _x0 \wedge _x1 & \rightarrow & _sm(m+1)\downarrow \\
___vmy & \rightarrow & _sm(m+1)\uparrow
\end{array}$$

...

$$\begin{array}{lcl}
\neg_v(M-1)y & \rightarrow & __v(M-1)y \\
_v(M-1)y & \rightarrow & __v(M-1)y
\end{array}
\qquad
\begin{array}{lcl}
__v(M-1)y \wedge _x0 \wedge _x1 & \rightarrow & _s(M-1)M\downarrow \\
\neg__v(M-1)y & \rightarrow & _s(M-1)M\uparrow
\end{array}$$

S input/output states:

$$\begin{array}{lcl}
_sM \wedge ye & \rightarrow & _s0\downarrow \\
\neg_sM \wedge \neg ye & \rightarrow & _s0\uparrow
\end{array}$$

$$\begin{array}{lcl}
\neg__s01 & \rightarrow & _s1\uparrow \\
__s01 & \rightarrow & _s1\downarrow
\end{array}$$

...

$$\begin{array}{lcl}
\neg__sm(m+1) & \rightarrow & _s(m+1)\uparrow \\
__sm(m+1) & \rightarrow & _s(m+1)\downarrow
\end{array}$$

...

$$\begin{array}{lcl}
\neg__s(M-1)M & \rightarrow & _sM\uparrow \\
__s(M-1)M & \rightarrow & _sM\downarrow
\end{array}$$

Y valid detectors:

$$\begin{array}{lcl}
y00 \vee y01 & \rightarrow & _v0y\downarrow \\
\neg y00 \wedge \neg y01 & \rightarrow & _v0y\uparrow
\end{array}$$

...

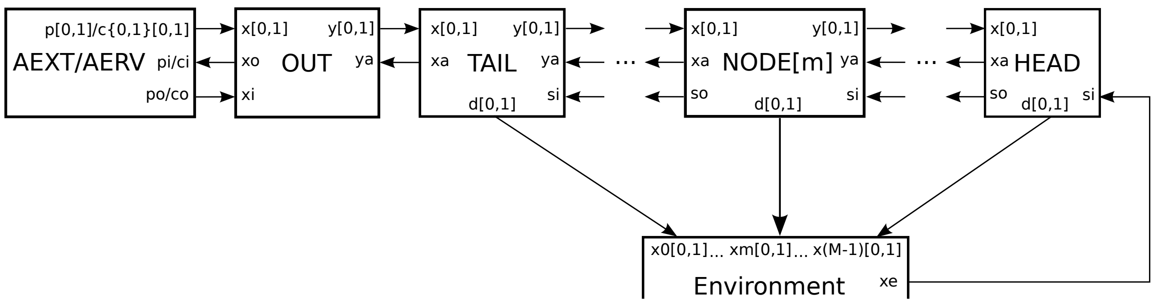
$$\begin{array}{lcl}
ym0 \vee ym1 & \rightarrow & _vmy\downarrow \\
\neg ym0 \wedge \neg ym1 & \rightarrow & _vmy\uparrow
\end{array}$$

...

$$\begin{array}{lcl}
y(M-1)0 \vee y(M-1)1 & \rightarrow & _v(M-1)y\downarrow \\
\neg y(M-1)0 \wedge \neg y(M-1)1 & \rightarrow & _v(M-1)y\uparrow
\end{array}$$

D.9 CHAIN Deserializer

This design uses a chain of nodes to sequence words into their respective place in the parallel output. That is, it takes as input a1ofN data and outputs eMx1ofN data. We decompose this process into HEAD, NODE, and TAIL processes. Although it has a bit more expensive 1-of-4 implementation than the split ring of Section D.4, its signal fanouts remain constant as the number of words in the packet grows. The below figure shows the decomposition for packets consisting of M 1-of-2 groups.



An OUT a1ofN process (described above) first converts the AEXT/AERV serial communication protocol to the standard a1ofN protocol. Each link in the chain outputs one of the words in the parallel output.

1-of-2 approximate scaling:

component	transistors/component	components/deserializer	transistors/deserializer
OUT a1ofN	4	1	4
HEAD	28	1	28
NODE	38	$M-2$	$38(M-2)$
TAIL	34	1	34
		transistors/deserializer	$38M-10$

1-of-4 approximate scaling:

component	transistors/component	components/deserializer	transistors/deserializer
OUT a1ofN	4	1	4
HEAD	50	1	50
NODE	68	$M-2$	$68(M-2)$
TAIL	66	1	66
		transistors/deserializer	$68M-16$

For the transmitter encoding 4096 neurons as 1-of-2 or 1-of-4 words, we would need 12 and 6 links in the chain, respectively.

1-of-2 approximate accounting:

component	transistors/component	components/deserializer	transistors/deserializer
OUT a1ofN	4	1	4
HEAD	28	1	28
NODE	38	10	380
TAIL	34	1	34
		total transistors/deserializer	446

1-of-4 approximate accounting:

component	transistors/component	components/deserializer	transistors/deserializer
OUT a1ofN	4	1	4
HEAD	50	1	50
NODE	68	4	272
TAIL	66	1	66
		total transistors/deserializer	392

D.10 HEAD

$$*[[si \wedge X]; D\uparrow; xa\uparrow; [\neg X]; so\uparrow; xa\downarrow; [\neg si]; D\downarrow; so\downarrow]$$

PRS

$$si \wedge x0 \rightarrow d0\uparrow \qquad si \wedge x1 \rightarrow d1\uparrow$$
$$\neg si \rightarrow d0\downarrow \qquad \neg si \rightarrow d1\downarrow$$

$$\neg x0 \wedge \neg x1 \wedge vd \rightarrow so\uparrow$$
$$x0 \vee x1 \vee \neg vd \rightarrow so\downarrow$$

$$\neg so \wedge vd \rightarrow xa\uparrow$$
$$so \vee \neg vd \rightarrow xa\downarrow$$

$$d0 \vee d1 \rightarrow vd\uparrow$$
$$\neg d0 \wedge \neg d1 \rightarrow vd\downarrow$$

1-of-2 transistor approximate accounting:

rule	transistor count	comments
$d[0,1]$	14	
s_o	6	
xa	4	
vd	4	
total	28	

1-of-4 transistor approximate accounting:

rule	transistor count	comments
$d[0,1,2,3]$	28	
s_o	10	
xa	4	
vd	8	
total	50	

CMOS-implementable PRS

$$\begin{array}{lll} si \wedge x0 & \rightarrow & \_d0\downarrow \\ \neg si & \rightarrow & \_d0\uparrow \end{array} \qquad \begin{array}{lll} si \wedge x1 & \rightarrow & \_d1\downarrow \\ \neg si & \rightarrow & \_d1\uparrow \end{array}$$

$$\begin{array}{lll} \neg\_d0 & \rightarrow & d0\uparrow \\ \_d0 & \rightarrow & d0\downarrow \end{array} \qquad \begin{array}{lll} \neg\_d1 & \rightarrow & d1\uparrow \\ \_d1 & \rightarrow & d1\downarrow \end{array}$$

$$\begin{array}{lll} \neg x0 \wedge \neg x1 \wedge \neg\_vd & \rightarrow & so\uparrow \\ x0 \vee x1 \vee \_vd & \rightarrow & so\downarrow \end{array}$$

$$\begin{array}{lll} \neg so & \rightarrow & \_so\uparrow \\ so & \rightarrow & so\downarrow \end{array}$$

$$\begin{array}{lll} \_so \wedge vd & \rightarrow & \_xa\downarrow \\ \neg\_so \vee \neg vd & \rightarrow & \_xa\uparrow \end{array}$$

$$\begin{array}{lll} \neg\_d0 \vee \neg\_d1 & \rightarrow & vd\uparrow \\ \_d0 \wedge \_d1 & \rightarrow & vd\downarrow \end{array}$$

$$\begin{array}{lll} \neg vd & \rightarrow & \_vd\uparrow \\ vd & \rightarrow & \_vd\downarrow \end{array}$$

$$\begin{array}{lll} \neg\_xa & \rightarrow & xa\uparrow \\ \_xa & \rightarrow & xa\downarrow \end{array}$$

1-of-2 transistor accounting:

rule	transistor count	comments
$\_d[0,1]$	14	
$d[0,1]$	4	
s_o	6	
$\_s_o$	2	
$\_xa$	4	
vd	4	
$\_vd$	2	
xa	2	
total	38	

1-of-4 transistor accounting:

rule	transistor count	comments
$\_d[0,1,2,3]$	28	
$d[0,1,2,3]$	8	
s_o	10	
$\_s_o$	2	
$\_xa$	4	
vd	8	
$\_vd$	2	
xa	2	
total	64	

D.11 NODE

$$
\begin{array}{l}
*[[\neg si \wedge X \longrightarrow Y\uparrow; [ya]; xa\uparrow; [\neg X]; Y\downarrow; [si \longrightarrow [\neg ya]; xa\downarrow [] \neg ya \longrightarrow xa\downarrow] \\
\quad [] si \wedge X \longrightarrow D\uparrow; xa\uparrow; [\neg X]; so\uparrow; xa\downarrow; [\neg si]; D\downarrow; so\downarrow \\
\;]]
\end{array}
$$

PRS

$$
\begin{array}{lcl@{\qquad\qquad}lcl}
\neg si \wedge x0 & \rightarrow & y0\uparrow & \neg si \wedge x1 & \rightarrow & y1\uparrow \\
si \vee \neg x0 & \rightarrow & y0\downarrow & si \vee \neg x1 & \rightarrow & y1\downarrow
\end{array}
$$

$$
\begin{array}{lcl@{\qquad\qquad}lcl}
si \wedge x0 & \rightarrow & d0\uparrow & si \wedge x1 & \rightarrow & d1\uparrow \\
\neg si & \rightarrow & d0\downarrow & \neg si & \rightarrow & d1\downarrow
\end{array}
$$

$$
\begin{array}{lcl}
\neg x0 \wedge \neg x1 \wedge vd & \rightarrow & so\uparrow \\
x0 \vee x1 \vee \neg vd & \rightarrow & so\downarrow
\end{array}
$$

$$\begin{array}{lcl} ya \vee \neg so \wedge vd & \rightarrow & xa\uparrow \\ \neg ya \wedge (so \vee \neg vd) & \rightarrow & xa\downarrow \end{array}$$

$$\begin{array}{lcl} d0 \vee d1 & \rightarrow & vd\uparrow \\ \neg d0 \wedge \neg d1 & \rightarrow & vd\downarrow \end{array}$$

1-of-2 transistor approximate accounting:

rule	transistor count	comments
$y[0,1]$	8	
$d[0,1]$	14	
s_o	6	
xa	6	
vd	4	
total	38	

1-of-4 transistor approximate accounting:

rule	transistor count	comments
$y[0,1,2,3]$	16	
$d[0,1,2,3]$	28	
s_o	10	
xa	6	
vd	8	
total	68	

CMOS-implementable PRS

$$\begin{array}{lcl} \neg x0 & \rightarrow & \_x0\downarrow \\ x0 & \rightarrow & \_x0\uparrow \end{array} \qquad \begin{array}{lcl} \neg x1 & \rightarrow & \_x1\downarrow \\ x1 & \rightarrow & \_x1\uparrow \end{array}$$

$$\begin{array}{lcl} \neg si \wedge \neg \_x0 & \rightarrow & y0\uparrow \\ si \vee \_x0 & \rightarrow & y0\downarrow \end{array} \qquad \begin{array}{lcl} \neg si \wedge \neg \_x1 & \rightarrow & y1\uparrow \\ si \vee \_x1 & \rightarrow & y1\downarrow \end{array}$$

$$\begin{array}{lcl} \neg si & \rightarrow & \_si\uparrow \\ si & \rightarrow & \_si\downarrow \end{array}$$

$$\begin{array}{lcl} \neg \_si \wedge \neg \_x0 & \rightarrow & d0\uparrow \\ \_si & \rightarrow & d0\downarrow \end{array} \qquad \begin{array}{lcl} \neg \_si \wedge \neg \_x1 & \rightarrow & d1\downarrow \\ \_si & \rightarrow & \_d1\uparrow \end{array}$$

$$\begin{array}{lcl} \neg \_vd & \rightarrow & vd\uparrow \\ \_vd & \rightarrow & vd\downarrow \end{array}$$

$$\begin{aligned} \_x0 \wedge \_x1 \wedge vd &\rightarrow \_so\downarrow \\ \neg\_x0 \vee \neg\_x1 \vee \neg vd &\rightarrow \_so\uparrow \end{aligned}$$

$$\begin{aligned} \neg\_so &\rightarrow so\uparrow \\ \_so &\rightarrow so\downarrow \end{aligned}$$

$$\begin{aligned} \neg so &\rightarrow \_\_\_so\uparrow \\ so &\rightarrow \_\_\_so\downarrow \end{aligned}$$

$$\begin{aligned} ya \vee \_\_\_so \wedge vd &\rightarrow \_xa\downarrow \\ \neg ya \wedge (\neg\_\_\_so \vee \neg vd) &\rightarrow \_xa\uparrow \end{aligned}$$

$$\begin{aligned} d0 \vee d1 &\rightarrow \_vd\downarrow \\ \neg d0 \wedge \neg d1 &\rightarrow \neg vd\uparrow \end{aligned}$$

$$\begin{aligned} \neg\_xa &\rightarrow xa\uparrow \\ \_xa &\rightarrow xa\downarrow \end{aligned}$$

1-of-2 transistor accounting:

rule	transistor count	comments
$\_x[0,1]$	4	
$y[0,1]$	8	
$\_s_i$	2	
$d[0,1]$	14	
vd	2	
$\_s_o$	6	
s_o	2	
$\_\_\_s_o$	2	
$\_xa$	6	
$\_vd$	4	
xa	2	
total	38	

1-of-4 transistor accounting:

rule	transistor count	comments
$\_x[0,1,2,3]$	8	
$y[0,1,2,3]$	16	
$\_s_i$	2	
$d[0,1,2,3]$	28	
vd	2	
$\_s_o$	10	
s_o	2	
$\_xa$	6	
$\_vd$	8	
xa	2	
total	84	

D.12 TAIL

$$*[[\neg si \wedge X \longrightarrow Y\uparrow; [ya]; xa\uparrow; [\neg X]; Y\downarrow; [si \longrightarrow [\neg ya]; xa\downarrow [] \neg ya \longrightarrow xa\downarrow]$$
$$[] si \wedge X \longrightarrow D\uparrow; xa\uparrow; [\neg si \wedge \neg X]; D\downarrow; xa\downarrow$$
$$]]$$

PRS

$$\begin{array}{lcl} \neg si \wedge x0 \wedge \neg d0 & \rightarrow & y0\uparrow \\ si \vee \neg x0 \vee d0 & \rightarrow & y0\downarrow \end{array} \qquad \begin{array}{lcl} \neg si \wedge x1 \wedge \neg d1 & \rightarrow & y1\uparrow \\ si \vee \neg x1 \vee d1 & \rightarrow & y1\downarrow \end{array}$$

$$\begin{array}{lcl} si \wedge x0 & \rightarrow & d0\uparrow \\ \neg si \wedge \neg x0 & \rightarrow & d0\downarrow \end{array} \qquad \begin{array}{lcl} si \wedge x1 & \rightarrow & d1\uparrow \\ \neg si \wedge \neg x1 & \rightarrow & d1\downarrow \end{array}$$

$$\begin{array}{lcl} ya \vee d0 \vee d1 & \rightarrow & xa\uparrow \\ \neg ya \wedge \neg d0 \wedge \neg d1 & \rightarrow & xa\downarrow \end{array}$$

1-of-2 transistor approximate accounting:

rule	transistor count	comments
$y[0,1]$	12	
$d[0,1]$	16	
xa	6	
total	34	

1-of-4 transistor approximate accounting:

rule	transistor count	comments
$y[0,1,2,3]$	24	
$d[0,1,2,3]$	32	
xa	10	
total	66	

CMOS-implementable PRS

$$\neg si \wedge \neg \_x0 \wedge \neg d0 \;\rightarrow\; y0\uparrow \qquad \neg si \wedge \neg \_x1 \wedge \neg d1 \;\rightarrow\; y1\uparrow$$
$$si \vee \_x0 \vee d0 \;\rightarrow\; y0\downarrow \qquad si \vee x1 \vee d1 \;\rightarrow\; y1\downarrow$$

$$\neg si \;\rightarrow\; \_si\uparrow$$
$$si \;\rightarrow\; \_si\downarrow$$

$$\neg \_si \wedge \neg \_x0 \;\rightarrow\; d0\uparrow \qquad \neg \_si \wedge \neg \_x1 \;\rightarrow\; d1\uparrow$$
$$\_si \wedge \_x0 \;\rightarrow\; d0\downarrow \qquad \_si \wedge \neg \_1 \;\rightarrow\; d1\downarrow$$

$$ya \vee d0 \vee d1 \;\rightarrow\; \_xa\downarrow$$
$$\neg ya \wedge \neg d0 \wedge \neg d1 \;\rightarrow\; \_xa\uparrow$$

1-of-2 transistor accounting:

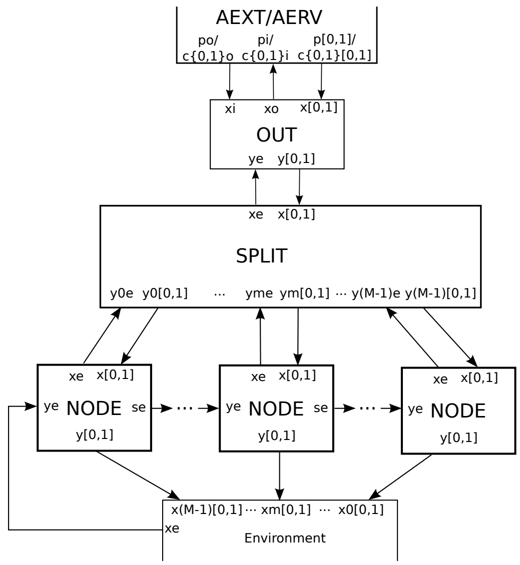
rule	transistor count	comments
$y[0,1]$	12	
$\_s_i$	2	
$d[0,1]$	16	
$\_xa$	6	
total	36	

1-of-4 transistor accounting:

rule	transistor count	comments
$y[0,1,2,3]$	24	
$\_s_i$	2	
$d[0,1,2,3]$	32	
xa	10	
total	68	

D.13 Split Chain Deserializer

This design uses a chain of nodes pulling data from a central splitter to sequence words into their respective place in the parallel output. We decompose this process into SPLIT and NODE. The below figure shows the decomposition for packets containing M 1-of-2 words.



An OUT e1ofN process (described above) first converts the AEXT/AERV serial communication protocol to the standard e1ofN protocol.

1-of-2 approximate scaling:

component	transistors/component	components/deserializer	transistors/deserializer
OUT e1ofN	4	1	4
SPLIT	$17M - 2$	1	$17M - 2$
NODE (int)	28	$M - 1$	$28(M - 1)$
NODE (end)	20	1	20
		approx. transistors/deserializer	$45M - 6$

1-of-4 approximate scaling:

component	transistors/component	components/deserializer	transistors/deserializer
OUT e1ofN	4	1	4
SPLIT	$31M - 2$	1	$31M - 2$
NODE (int)	50	$M - 1$	$50(M - 1)$
NODE (end)	38	1	38
		approx. transistors/deserializer	$81M - 10$

For the transmitter to handle 4096 neurons encoded as 1-of-2 or 1-of-4 words, we would need 12 and 6 NODEs, respectively.

1-of-2 accounting:

component	transistors/component	components/deserializer	transistors/deserializer
OUT e1ofN	4	1	4
SPLIT	198	1	198
NODE (int)	28	11	308
NODE (end)	20	1	20
		total transistors/deserializer	530

1-of-4 accounting:

component	transistors/component	components/deserializer	transistors/deserializer
OUT e1ofN	4	1	4
SPLIT	184	1	184
NODE (int)	50	5	250
NODE (end)	38	1	38
		total transistors/deserializer	476

D.14 SPLIT

SPLIT takes incoming words and routes them to their respective locations in the parallel output. For M words per packet,

$$
\begin{array}{l}
*\mathtt{[[}y0e \longrightarrow xe\uparrow; \\
\quad\quad \mathtt{[}x0 \longrightarrow y00\uparrow; \mathtt{[}\neg y0e\mathtt{]}; xe\downarrow; \mathtt{[}\neg x0\mathtt{]}; y00\downarrow \\
\quad\quad \mathtt{[]}x1 \longrightarrow y01\uparrow; \mathtt{[}\neg y0e\mathtt{]}; xe\downarrow; \mathtt{[}\neg x1\mathtt{]}; y01\downarrow \\
\quad\quad \mathtt{]} \\
\quad \mathtt{[]}\ \ldots \\
\quad \mathtt{[]}y(M-1)e \longrightarrow xe\uparrow; \\
\quad\quad \mathtt{[}x1 \longrightarrow y(M-1)0\uparrow; \mathtt{[}\neg y(M-1)e\mathtt{]}; xe\downarrow; \mathtt{[}\neg x0\mathtt{]}; y(M-1)0\downarrow \\
\quad\quad \mathtt{[]}x1 \longrightarrow y(M-1)1\uparrow; \mathtt{[}\neg y(M-1)e\mathtt{]}; xe\downarrow; \mathtt{[}\neg x1\mathtt{]}; y(M-1)1\downarrow \\
\quad\quad \mathtt{]} \\
\quad \mathtt{]} \\
\mathtt{]}
\end{array}
$$

For a 2-word packet,

$$
\begin{array}{l}
*[[y0e \longrightarrow xe\uparrow;\\
\quad [x0 \longrightarrow y00\uparrow; [\neg y0e]; xe\downarrow; [\neg x0]; y00\downarrow\\
\quad [\!]x1 \longrightarrow y01\uparrow; [\neg y0e]; xe\downarrow; [\neg x1]; y01\downarrow\\
\quad]\\
\;[\!]y1e \longrightarrow xe\uparrow;\\
\quad [x1 \longrightarrow y10\uparrow; [\neg y1e]; xe\downarrow; [\neg x0]; y10\downarrow\\
\quad [\!]x1 \longrightarrow y11\uparrow; [\neg y1e]; xe\downarrow; [\neg x1]; y11\downarrow\\
\quad]\\
]]
\end{array}
$$

$$
\begin{array}{lcl}
y0e \vee y1e & \rightarrow & xe\uparrow\\
\neg y0e \wedge \neg y1e & \rightarrow & xe\downarrow
\end{array}
$$

$$
\begin{array}{lcl@{\qquad\qquad}lcl}
y0e \wedge x0 & \rightarrow & y00\uparrow & y0e \wedge x1 & \rightarrow & y01\uparrow\\
\neg x0 & \rightarrow & y00\downarrow & \neg x1 & \rightarrow & y01\downarrow
\end{array}
$$

$$
\begin{array}{lcl@{\qquad\qquad}lcl}
y1e \wedge x0 & \rightarrow & y10\uparrow & y1e \wedge x1 & \rightarrow & y11\uparrow\\
\neg x0 & \rightarrow & y10\downarrow & \neg x1 & \rightarrow & y11\downarrow
\end{array}
$$

1-of-2 transistor accounting:

rule	transistor count	comments
xe	$8(M-1)/3$	4-ary OR-tree approx.
$y[0..M-1][0,1]$	$14M$	
approx. total	$17M-2$	

1-of-4 transistor accounting:

rule	transistor count	comments
xe	$8(M-1)/3$	4-ary OR-tree approx.
$y[0..M-1][0,1,2,3]$	$28M$	
approx. total	$31M-2$	

D.15 NODE

NODE latches data from SPLIT. For beginning and intermediate NODEs,

$$*[[ye]; xe\uparrow; [X]; Y\uparrow; xe\downarrow; [\neg X]; se\uparrow; [\neg ye]; Y\downarrow; se\downarrow]$$

$$
\begin{array}{lcl}
ye \wedge \neg vy & \rightarrow & xe\uparrow\\
\neg ye \vee vy & \rightarrow & xe\downarrow
\end{array}
$$

$$x0 \to y0\uparrow \qquad x1 \to y1\uparrow$$
$$\neg ye \to y0\downarrow \qquad \neg ye \to y1\downarrow$$

$$\neg x0 \wedge \neg x1 \wedge vy \to se\uparrow$$
$$\neg vy \to se\downarrow$$

$$y0 \vee y1 \to vy\uparrow$$
$$\neg y0 \wedge \neg y1 \to vy\downarrow$$

1-of-2 transistor accounting:

rule	transistor count	comments
xe	4	
$y[0,1]$	12	
se	8	
vy	4	
total	28	

1-of-4 transistor accounting:

rule	transistor count	comments
xe	4	
$y[0,1,2,3]$	28	
se	10	
vy	8	
total	50	

The end NODE of a chain does not need to forward an se signal,

$$*[[ye];xe\uparrow;[X];Y\uparrow;xe\downarrow;[\neg X \wedge \neg ye];Y\downarrow]$$

$$ye \wedge \neg y0 \wedge \neg y1 \to xe\uparrow$$
$$\neg ye \vee y0 \vee y1 \to xe\downarrow$$

$$x0 \to y0\uparrow \qquad x1 \to y1\uparrow$$
$$\neg x0 \wedge \neg ye \to y0\downarrow \qquad \neg x1 \wedge \neg ye \to y1\downarrow$$

1-of-2 transistor accounting:

rule	transistor count	comments
xe	6	
$y[0,1]$	14	
total	20	

1-of-4 transistor accounting:

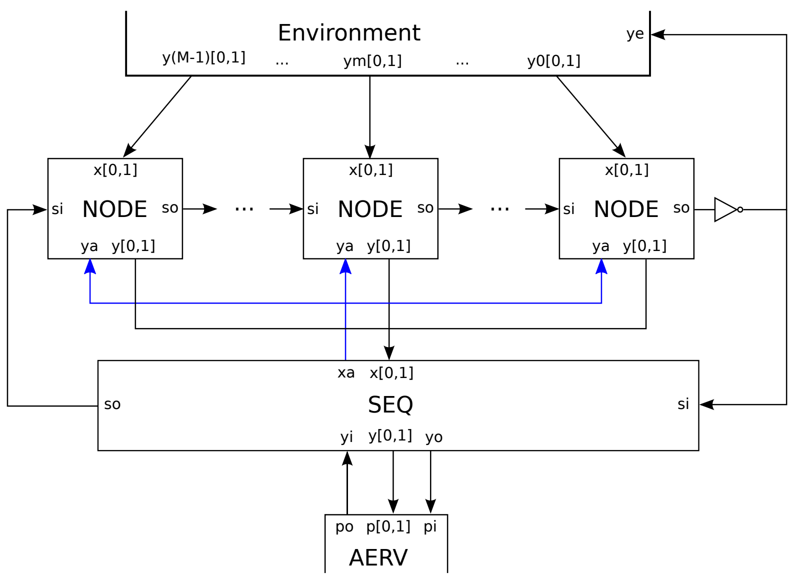
rule	transistor count	comments
xe	10	
$y[0,1,2,3]$	28	
total	38	

D.16 Serializer

The serializer converts M-1-of-N parallel data into in 1-of-N serial data. We cannot create standard a1ofN or e1ofN input interfaces to the serial protocol (like we could with output interfaces in Sections D.1 and D.2) because the serial protocol requires a signal indicating the transitions between packets, which the standard a1ofN and e1ofN channels cannot provide.

D.17 Ring serializer

The ring serializer uses a ring of NODES to sequence the words of an eMx1ofN channel. A C-element indicates the transition between the up and down phases of the sequencing.



Note the shared ye and $y[0,1]$ between the NODE processes.

1-of-2 approximate scaling:

component	transistors/component	components/serializer	transistors/serializer
NODE	34	M	$34M$
SEQ	38	1	38
		approx. transistors/serializer	$34M + 38$

1-of-4 approximate scaling:

component	transistors/component	components/serializer	transistors/serializer
NODE	54	M	$54M$
SEQ	56	1	56
		approx. transistors/serializer	$54M + 56$

For the receiver to handle 4096 neurons (and no data) encoded as 1-of-2 or 1-of-4 words, we would need 12 and 6 NODEs, respectively.

1-of-2 accounting:

component	transistors/component	components/serializer	transistors/serializer
NODE	34	12	408
SEQ	38	1	38
		total transistors/serializer	446

1-of-4 accounting:

component	transistors/component	components/serializer	transistors/serializer
NODE	54	6	324
SEQ	56	1	56
		total transistors/serializer	380

D.18 Ring serializer NODE

$$
\begin{array}{l}
*[[si];\\
\quad [\ \ x0 \longrightarrow y0\uparrow; [ya]; u\uparrow; y0\downarrow; [\neg ya]; so\uparrow; [\neg si]; u\downarrow; [\neg x0]; so\downarrow\\
\quad [\!]\, x1 \longrightarrow y1\uparrow; [ya]; u\uparrow; y1\downarrow; [\neg ya]; so\uparrow; [\neg si]; u\downarrow; [\neg x1]; so\downarrow\\
\ \ ]]
\end{array}
$$

PRS

$$
\begin{array}{lcl@{\qquad\qquad}lcl}
x0 \wedge \neg u \wedge si & \rightarrow & y0\uparrow & x1 \wedge \neg u \wedge si & \rightarrow & y1\uparrow\\
u \wedge \neg so & \rightarrow & y0\downarrow & u \wedge \neg so & \rightarrow & y1\downarrow
\end{array}
$$

$$
\begin{array}{lcl}
si \wedge ya & \rightarrow & u\uparrow\\
\neg si & \rightarrow & u\downarrow
\end{array}
$$

$$
\begin{aligned}
u \wedge \neg ya &\rightarrow so\uparrow \\
\neg u \wedge \neg x0 \wedge \neg x1 &\rightarrow so\downarrow
\end{aligned}
$$

1-of-2 transistor accounting:

rule	transistor count	comments
$y[0,1]$	18	
u	7	
s_o	9	
total	34	

1-of-4 transistor accounting:

rule	transistor count	comments
$y[0,1,2,3]$	36	
u	7	
s_o	11	
total	54	

CMOS-implementable PRS

$$
\begin{aligned}
\neg\_u &\rightarrow \_\_u\uparrow \\
\_u &\rightarrow \_\_u\downarrow
\end{aligned}
$$

$$
\begin{aligned}
\neg so &\rightarrow \_so\uparrow \\
so &\rightarrow \_so\downarrow
\end{aligned}
$$

$$
\begin{aligned}
\neg\_x0 \wedge \neg\_\_u \wedge \neg\_si &\rightarrow y0\uparrow & \qquad \neg\_x1 \wedge \neg\_\_u \wedge \neg\_si &\rightarrow y1\uparrow \\
\_\_u \wedge \_so &\rightarrow y0\downarrow & \qquad \_\_u \wedge \_so &\rightarrow y1\downarrow
\end{aligned}
$$

$$
\begin{aligned}
\neg\_si &\rightarrow \_\_si\uparrow \\
\_si &\rightarrow \_\_si\downarrow
\end{aligned}
$$

$$
\begin{aligned}
\_\_si \wedge ya &\rightarrow \_u\downarrow \\
\neg\_\_si &\rightarrow \_u\uparrow
\end{aligned}
$$

$$
\begin{aligned}
\neg\_\_u &\rightarrow \_\_\_u\uparrow \\
\_\_u &\rightarrow \_\_\_u\downarrow
\end{aligned}
$$

$$
\begin{aligned}
\neg\_\_\_u \wedge \neg ya &\rightarrow so\uparrow \\
\_\_\_u \wedge \_x0 \wedge \_x1 &\rightarrow so\downarrow
\end{aligned}
$$

$$
\begin{aligned}
\neg y0 &\rightarrow \_y0\uparrow & \qquad \neg y1 &\rightarrow \_y1\uparrow \\
y0 &\rightarrow \_y0\downarrow & \qquad y1 &\rightarrow \_y1\downarrow
\end{aligned}
$$

D.19 Ring serializer sequencer

$$
\begin{array}{l}
*[[si];so\uparrow;[x0 \vee x1];yo\uparrow;\\
\quad [\neg si \wedge yi];yo\downarrow;[\neg yi];so\downarrow]
\end{array}
$$

$$
\begin{array}{l}
*[[x0 \longrightarrow y0\uparrow;[\neg yi];xa\uparrow;[\neg x0];y0\downarrow;[yi];xa\downarrow\\
\quad [\!]\, x1 \longrightarrow y1\uparrow;[\neg yi];xa\uparrow;[\neg x1];y1\downarrow;[yi];xa\downarrow\\
\ ]]
\end{array}
$$

PRS

$$
\begin{array}{lcl}
x0 \vee x1 & \rightarrow & yo\uparrow\\
\neg si \wedge yi & \rightarrow & yo\downarrow
\end{array}
$$

$$
\begin{array}{lcl}
si & \rightarrow & so\uparrow\\
\neg si \wedge \neg yi \wedge \neg yo & \rightarrow & so\downarrow
\end{array}
$$

$$
\begin{array}{lcl}
yi \wedge x0 & \rightarrow & y0\uparrow\\
\neg x0 & \rightarrow & y0\downarrow
\end{array}
\qquad
\begin{array}{lcl}
yi \wedge x1 & \rightarrow & y1\uparrow\\
\neg x1 & \rightarrow & y1\downarrow
\end{array}
$$

$$
\begin{array}{lcl}
(y0 \vee y1) \wedge \neg yi & \rightarrow & xa\uparrow\\
yi & \rightarrow & xa\downarrow
\end{array}
$$

1-of-2 transistor accounting:

rule	transistor count	comments
y_o	8	
s_o	8	
$y[0,1]$	14	
xa	8	
total	38	

1-of-4 transistor accounting:

rule	transistor count	comments
y_o	10	
s_o	8	
$y[0,1,2,3]$	28	
xa	10	
total	56	

CMOS-implementable PRS

$$\begin{aligned}
\neg_x0 \vee \neg_x1 &\rightarrow yo\uparrow \\
_si \wedge yi &\rightarrow yo\downarrow
\end{aligned}$$

$$\begin{aligned}
\neg_si &\rightarrow __si\uparrow \\
_si &\rightarrow __si\downarrow
\end{aligned}$$

$$\begin{aligned}
__si &\rightarrow _so\downarrow \\
\neg__si \wedge \neg yi \wedge \neg yo &\rightarrow _so\uparrow
\end{aligned}$$

$$\begin{aligned}
\neg_x0 &\rightarrow __x0\uparrow & \neg_x1 &\rightarrow __x1\uparrow \\
_x0 &\rightarrow __x0\downarrow & _x1 &\rightarrow __x1\downarrow
\end{aligned}$$

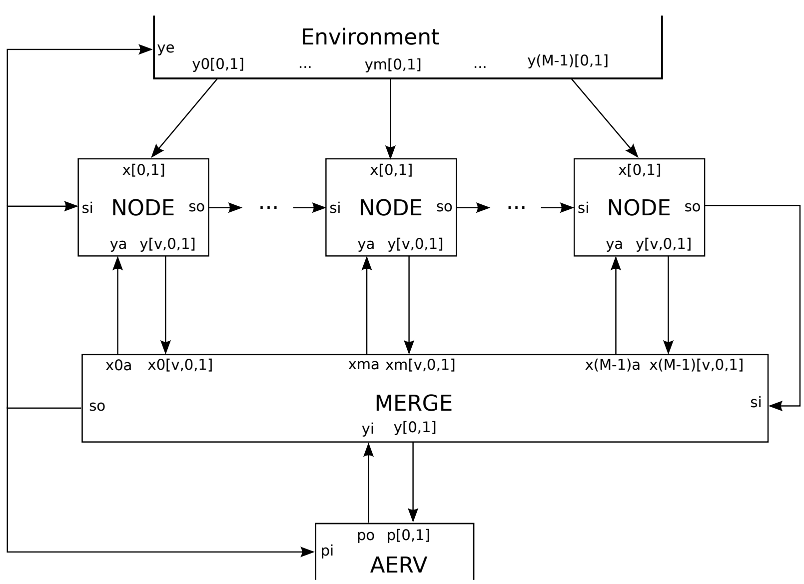
$$\begin{aligned}
yi \wedge __x0 &\rightarrow _y0\downarrow & yi \wedge __x1 &\rightarrow _y1\downarrow \\
\neg__x0 &\rightarrow _y0\uparrow & \neg__x1 &\rightarrow _y1\uparrow
\end{aligned}$$

$$\begin{aligned}
(\neg_y0 \vee \neg_y1) \wedge \neg yi &\rightarrow xa\uparrow \\
yi &\rightarrow xa\downarrow
\end{aligned}$$

$$\begin{aligned}
\neg_y0 &\rightarrow y0\uparrow & \neg_y1 &\rightarrow y1\uparrow \\
_y0 &\rightarrow y0\downarrow & _y1 &\rightarrow y1\downarrow
\end{aligned}$$

D.20 Ring serializer 2

This design uses more conventional channels and more sequencing than the previous serial ring. Since there's only one serializer per neuron array, the cost per neuron is still trivial.



D.21 Ring serializer 2 NODE

$$
\begin{array}{l}
*[[\ si \wedge x0 \longrightarrow y0\uparrow, yv\uparrow; [ya]; u\downarrow; y0\downarrow, yv\downarrow; [\neg ya]; so\uparrow; [\neg si \wedge \neg x0]; u\uparrow; so\downarrow \\
\ \ [] si \wedge x1 \longrightarrow y1\uparrow, yv\uparrow; [ya]; u\downarrow; y1\downarrow, yv\downarrow; [\neg ya]; so\uparrow; [\neg si \wedge \neg x1]; u\uparrow; so\downarrow \\
\ ]]
\end{array}
$$

PRS

$$
\begin{array}{lcl@{\qquad\qquad}lcl}
si \wedge x0 \wedge u & \rightarrow & y0\uparrow & si \wedge x1 \wedge u & \rightarrow & y1\uparrow \\
\neg si \vee \neg x0 \vee \neg u & \rightarrow & y0\downarrow & \neg si \vee \neg x1 \vee \neg u & \rightarrow & y1\downarrow
\end{array}
$$

$$
\begin{array}{lcl}
si \wedge (x0 \vee x1) \wedge u & \rightarrow & yv\uparrow \\
\neg si \vee \neg x0 \wedge \neg x1 \vee \neg u & \rightarrow & yv\downarrow
\end{array}
$$

$$
\begin{array}{lcl}
\neg si \wedge \neg x0 \wedge \neg x1 & \rightarrow & u\uparrow \\
ya & \rightarrow & u\downarrow
\end{array}
$$

$$
\begin{array}{lcl}
\neg u \wedge \neg ya & \rightarrow & so\uparrow \\
u \vee ya & \rightarrow & so\downarrow
\end{array}
$$

CMOS-implementable PRS

$$
\begin{array}{lcl@{\qquad\qquad}lcl}
si \wedge x0 \wedge u & \rightarrow & \_y0\downarrow & si \wedge x1 \wedge u & \rightarrow & \_y1\downarrow \\
\neg si \vee \neg x0 \vee \neg u & \rightarrow & \_y0\uparrow & \neg si \vee \neg x1 \vee \neg u & \rightarrow & \_y1\uparrow
\end{array}
$$

$$
\begin{array}{lcl}
si \wedge (x0 \vee x1) \wedge u & \rightarrow & \_yv\downarrow \\
\neg si \vee \neg x0 \wedge \neg x1 \vee \neg u & \rightarrow & \_yv\uparrow
\end{array}
$$

$$
\begin{array}{lcl}
\neg si \wedge \neg x0 \wedge \neg x1 & \rightarrow & u\uparrow \\
ya & \rightarrow & u\downarrow
\end{array}
$$

$$
\begin{array}{lcl}
\neg u \wedge \neg ya & \rightarrow & so\uparrow \\
u \vee ya & \rightarrow & so\downarrow
\end{array}
$$

$$
\begin{array}{lcl@{\qquad\qquad}lcl}
\neg \_y0 & \rightarrow & y0\uparrow & \neg \_y1 & \rightarrow & y1\uparrow \\
\_y0 & \rightarrow & y0\downarrow & \_y1 & \rightarrow & y1\downarrow
\end{array}
$$

D.22 Ring serializer 2 MERGE

for M words,

$$*[[\neg si \wedge \neg yi];\, so{\uparrow};\, [si \wedge yi];\, so{\downarrow}]$$

$$\begin{array}{l}
*[[\ \ x00 \longrightarrow u0{\uparrow},\, [yi];\, y0{\uparrow};\, vy{\uparrow};\, [x0v \wedge \neg yi];\, x0a{\uparrow};\, ([\neg x00];\, u0{\downarrow};\, y0{\downarrow};\, vy{\downarrow}),\, [\neg x0v];\, x0a{\downarrow} \\
\quad \Box\, x01 \longrightarrow u1{\uparrow},\, [yi];\, y1{\uparrow};\, vy{\uparrow};\, [x0v \wedge \neg yi];\, x0a{\uparrow};\, ([\neg x01];\, u1{\downarrow};\, y1{\downarrow};\, vy{\downarrow}),\, [\neg x0v];\, x0a{\downarrow} \\
\quad \ldots \\
\quad \Box\, xm0 \longrightarrow u0{\uparrow},\, [yi];\, y0{\uparrow};\, vy{\uparrow};\, [xmv \wedge \neg yi];\, xma{\uparrow};\, ([\neg xm0];\, u0{\downarrow};\, y0{\downarrow};\, vy{\downarrow}),\, [\neg xmv];\, xma{\downarrow} \\
\quad \Box\, xm1 \longrightarrow u1{\uparrow},\, [yi];\, y1{\uparrow};\, vy{\uparrow};\, [xmv \wedge \neg yi];\, xma{\uparrow};\, ([\neg xm1];\, u1{\downarrow};\, y1{\downarrow};\, vy{\downarrow}),\, [\neg xmv];\, xma{\downarrow} \\
\quad \ldots \\
\quad \Box\, x(M{\downarrow}1)0 \longrightarrow u0{\uparrow},\, [yi];\, y0{\uparrow};\, vy{\uparrow};\, [x(M{\downarrow}1)v \wedge \neg yi];\, x(M{\downarrow}1)a{\uparrow};\, ([\neg x(M{\downarrow}1)0];\, u0{\downarrow};\, y0{\downarrow};\, vy{\downarrow}),\, [\neg x(M{\downarrow}1)v] \\
\quad \Box\, x(M{\downarrow}1)1 \longrightarrow u1{\uparrow},\, [yi];\, y1{\uparrow};\, vy{\uparrow};\, [x(M{\downarrow}1)v \wedge \neg yi];\, x(M{\downarrow}1)a{\uparrow};\, ([\neg x(M{\downarrow}1)1];\, u1{\downarrow};\, y1{\downarrow};\, vy{\downarrow}),\, [\neg x(M{\downarrow}1)v] \\
\ \ ]]
\end{array}$$

for $M = 2$,

$$*[[\neg si \wedge \neg yi];\, so{\uparrow};\, [si \wedge yi];\, so{\downarrow}]$$

$$\begin{array}{l}
*[[\ \ x00 \longrightarrow u0{\uparrow},\, [yi];\, y0{\uparrow};\, vy{\uparrow};\, [x0v \wedge \neg yi];\, x0a{\uparrow};\, ([\neg x00];\, u0{\downarrow};\, y0{\downarrow};\, vy{\downarrow}),\, [\neg x0v];\, x0a{\downarrow} \\
\quad \Box\, x01 \longrightarrow u1{\uparrow},\, [yi];\, y1{\uparrow};\, vy{\uparrow};\, [x0v \wedge \neg yi];\, x0a{\uparrow};\, ([\neg x01];\, u1{\downarrow};\, y1{\downarrow};\, vy{\downarrow}),\, [\neg x0v];\, x0a{\downarrow} \\
\quad \Box\, x10 \longrightarrow u0{\uparrow},\, [yi];\, y0{\uparrow};\, vy{\uparrow};\, [x1v \wedge \neg yi];\, x1a{\uparrow};\, ([\neg x10];\, u0{\downarrow};\, y0{\downarrow};\, vy{\downarrow}),\, [\neg x1v];\, x1a{\downarrow} \\
\quad \Box\, x11 \longrightarrow u1{\uparrow},\, [yi];\, y1{\uparrow};\, vy{\uparrow};\, [x1v \wedge \neg yi];\, x1a{\uparrow};\, ([\neg x11];\, u1{\downarrow};\, y1{\downarrow};\, vy{\downarrow}),\, [\neg x1v];\, x1a{\downarrow} \\
\ \ ]]
\end{array}$$

$$\begin{array}{lcl}
\neg si \wedge \neg yi & \to & so{\uparrow} \\
si \wedge yi & \to & so{\downarrow}
\end{array}$$

$$\begin{array}{lcl@{\qquad\qquad}lcl}
x00 \vee x10 & \to & u0{\uparrow} & x01 \vee x11 & \to & u1{\uparrow} \\
\neg x00 \wedge \neg x10 & \to & u0{\downarrow} & \neg x01 \wedge \neg x11 & \to & u1{\downarrow}
\end{array}$$

$$\begin{array}{lcl@{\qquad\qquad}lcl}
u0 \wedge yi & \to & y0{\uparrow} & u1 \wedge yi & \to & y1{\uparrow} \\
\neg u0 & \to & y0{\downarrow} & \neg u1 & \to & y1{\downarrow}
\end{array}$$

$$\begin{array}{lcl}
y0 \vee y1 & \to & vy{\uparrow} \\
\neg y0 \wedge \neg y1 & \to & vy{\uparrow}
\end{array}$$

$$\begin{array}{lcl@{\qquad\qquad}lcl}
x0v \wedge \neg yi \wedge vy & \to & x0a{\uparrow} & x1v \wedge \neg yi \wedge vy & \to & x1a{\uparrow} \\
\neg x0v \wedge \neg vy & \to & x0a{\downarrow} & \neg x1v \wedge \neg vy & \to & x1a{\downarrow}
\end{array}$$

CMOS-implementable PRS

$$\begin{array}{lcl}
\neg si \wedge \neg yi & \to & so{\uparrow} \\
si \wedge yi & \to & so{\downarrow}
\end{array}$$

$$\begin{array}{lll} x00 \vee x10 & \rightarrow & u0\uparrow \\ \neg x00 \wedge \neg x10 & \rightarrow & u0\downarrow \end{array} \qquad \begin{array}{lll} x01 \vee x11 & \rightarrow & u1\uparrow \\ \neg x01 \wedge \neg x11 & \rightarrow & u1\downarrow \end{array}$$

$$\begin{array}{lll} u0 \wedge yi & \rightarrow & \_y0\downarrow \\ \neg u0 & \rightarrow & \_y0\uparrow \end{array} \qquad \begin{array}{lll} u1 \wedge yi & \rightarrow & \_y1\downarrow \\ \neg u1 & \rightarrow & \_y1\uparrow \end{array}$$

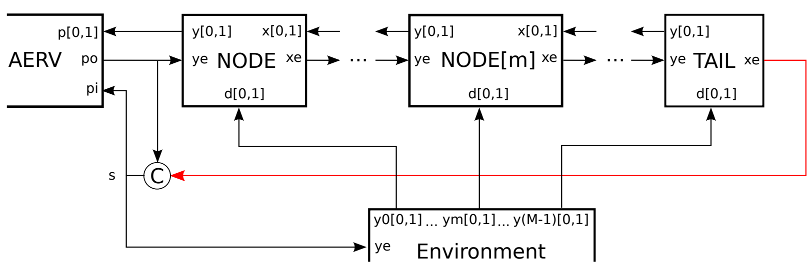
$$\begin{array}{lll} \neg\_y0 & \rightarrow & y0\uparrow \\ \_y0 & \rightarrow & y0\downarrow \end{array} \qquad \begin{array}{lll} \neg\_y1 & \rightarrow & y1\uparrow \\ \_y1 & \rightarrow & y1\downarrow \end{array}$$

$$\begin{array}{lll} y0 \vee y1 & \rightarrow & \_vy\downarrow \\ \neg y0 \wedge \neg y1 & \rightarrow & \_vy\uparrow \end{array}$$

$$\begin{array}{lll} \neg\_x0v \wedge \neg yi \wedge \neg\_vy & \rightarrow & x0a\uparrow \\ \_x0v \wedge \_vy & \rightarrow & x0a\downarrow \end{array} \qquad \begin{array}{lll} \neg\_x1v \wedge \neg yi \wedge \neg\_vy & \rightarrow & x1a\uparrow \\ \_x1v \wedge \_vy & \rightarrow & x1a\downarrow \end{array}$$

D.23 Chain serializer

The chain serializer uses a chain of NODES to sequence the words of an eMx1ofN channel. A C-element indicates the transition between the up and down phases of the sequencing.



1-of-2 approximate scaling:

component	transistors/component	components/serializer	transistors/serializer
NODE	39	$M-1$	$39(M-1)$
TAIL	33	1	33
C	8	1	8
		approx. transistors/serializer	$39M+2$

1-of-4 approximate scaling:

component	transistors/component	components/serializer	transistors/serializer
NODE	65	$M-1$	$65(M-1)$
TAIL	53	1	53
C	8	1	8
		approx. transistors/serializer	$65M-4$

For the receiver to handle 4096 neurons (and no data) encoded as 1-of-2 or 1-of-4 words, we would need 11 and 5 NODEs, respectively.
1-of-2 accounting:

component	transistors/component	components/serializer	transistors/serializer
NODE	39	11	429
TAIL	33	1	33
C	8	1	8
		total transistors/serializer	470

1-of-4 accounting:

component	transistors/component	components/serializer	transistors/serializer
NODE	65	5	325
TAIL	53	5	53
C	8	1	8
		total transistors/serializer	386

D.24 Chain serializer NODE

```
*[[ye];
  [¬u ⟶
    [d0 ⟶ y0↑; [¬ye]; u↑; y0↓
    []d1 ⟶ y1↑; [¬ye]; u↑; y1↓
    ]
  []u ⟶ xe↑;
    [x0 ⟶ y0↑; [¬ye]; xe↓; [¬x0]; y0↓
    []x1 ⟶ y1↑; [¬ye]; xe↓; [¬x1]; y1↓
    []¬d0 ∧ ¬d1 ⟶ u↓; [¬ye]; xe↓
    ]
 ]]
```

PRS

$$
\begin{array}{lll@{\qquad}lll}
ye \wedge \neg u \wedge d0 \vee x0 & \rightarrow & y0\uparrow & ye \wedge \neg u \wedge d1 \vee x1 & \rightarrow & y1\uparrow \\
\neg ye \wedge u \wedge \neg x0 & \rightarrow & y0\downarrow & \neg ye \wedge u \wedge \neg x1 & \rightarrow & y1\downarrow
\end{array}
$$

$$
\begin{array}{lll@{\qquad}lll}
y0 \vee y1 & \rightarrow & vy\uparrow & d0 \vee d1 & \rightarrow & vd\uparrow \\
\neg y0 \wedge \neg y1 & \rightarrow & vy\downarrow & \neg d0 \wedge \neg d1 & \rightarrow & vd\downarrow
\end{array}
$$

$$
\begin{array}{lll}
vy \wedge vd & \rightarrow & u\uparrow \\
\neg vy \wedge \neg vd & \rightarrow & u\downarrow
\end{array}
$$

$$\begin{aligned} ye \wedge u \wedge \neg vy &\rightarrow xe\uparrow \\ \neg ye \wedge (\neg u \vee vy) &\rightarrow xe\downarrow \end{aligned}$$

1-of-2 transistor approximate accounting:

rule	transistor count	comments
$y[0,1]$	20	
u	9	
xe	10	
total	39	

1-of-4 transistor approximate accounting:

rule	transistor count	comments
$y[0,1]$	40	
u	13	
xe	12	
total	65	

CMOS-implementable PRS

$$\begin{aligned} \neg u &\rightarrow \_u\uparrow \\ u &\rightarrow \_u\downarrow \end{aligned}$$

$$\begin{aligned} ye \wedge \_u \wedge d0 \vee x0 &\rightarrow \_y0\downarrow & \quad ye \wedge \_u \wedge d1 \vee x1 &\rightarrow \_y1\downarrow \\ \neg ye \wedge \neg\_u \wedge \neg x0 &\rightarrow \_y0\uparrow & \quad \neg ye \wedge \neg\_u \wedge \neg x1 &\rightarrow \_y1\uparrow \end{aligned}$$

$$\begin{aligned} \neg\_y0 &\rightarrow \_\_y0\uparrow & \quad \neg\_y1 &\rightarrow \_\_y1\uparrow \\ \_y0 &\rightarrow \_\_y0\downarrow & \quad \_y1 &\rightarrow \_\_y1\downarrow \end{aligned}$$

$$\begin{aligned} \_\_y0 \vee \_\_y1 &\rightarrow \_vy\downarrow & \quad d0 \vee d1 &\rightarrow \_vd\downarrow \\ \neg\_\_y0 \wedge \neg\_\_y1 &\rightarrow \_vy\uparrow & \quad \neg d0 \wedge \neg d1 &\rightarrow \_vd\uparrow \end{aligned}$$

$$\begin{aligned} \neg\_vy \wedge \neg\_vd &\rightarrow u\uparrow \\ \_vy \wedge \_vd &\rightarrow u\downarrow \end{aligned}$$

$$\begin{aligned} \neg\_u &\rightarrow \_\_u\uparrow \\ \_u &\rightarrow \_\_u\downarrow \end{aligned}$$

$$\begin{aligned} ye \wedge \_\_u \wedge \_vy &\rightarrow \_xe\downarrow \\ \neg ye \wedge (\neg\_\_u \vee \neg\_vy) &\rightarrow \_xe\uparrow \end{aligned}$$

D.25 Chain serializer TAIL

$$
\begin{array}{l}
*[[ye];\\
\quad [\neg u \longrightarrow\\
\quad\quad [d0 \longrightarrow y0\uparrow; u\uparrow; [\neg ye]; y0\downarrow\\
\quad\quad [\!]\, d1 \longrightarrow y1\uparrow; u\uparrow; [\neg ye]; y1\downarrow\\
\quad\quad]\\
\quad [\!]\, u \longrightarrow xe\uparrow; [\neg d0 \wedge \neg d1]; u\downarrow; [\neg ye]; xe\downarrow\\
\ ]]
\end{array}
$$

PRS

$$
\begin{array}{lcl@{\qquad}lcl}
ye \wedge \neg u \wedge d0 & \rightarrow & y0\uparrow & ye \wedge \neg u \wedge d1 & \rightarrow & y1\uparrow\\
\neg ye \wedge u & \rightarrow & y0\downarrow & \neg ye \wedge u & \rightarrow & y1\downarrow
\end{array}
$$

$$
\begin{array}{lcl@{\qquad}lcl}
y0 \vee y1 & \rightarrow & vy\uparrow & d0 \vee d1 & \rightarrow & vd\uparrow\\
\neg y0 \wedge \neg y1 & \rightarrow & vy\downarrow & \neg d0 \wedge \neg d1 & \rightarrow & vd\downarrow
\end{array}
$$

$$
\begin{array}{lcl}
vy \wedge vd & \rightarrow & u\uparrow\\
\neg vy \wedge \neg vd & \rightarrow & u\downarrow
\end{array}
$$

$$
\begin{array}{lcl}
ye \wedge u \wedge \neg vy & \rightarrow & xe\uparrow\\
\neg ye \wedge (\neg u \vee vy) & \rightarrow & xe\downarrow
\end{array}
$$

1-of-2 transistor approximate accounting:

rule	transistor count	comments
$y[0,1]$	16	
u	9	
xe	8	
total	33	

1-of-4 transistor approximate accounting:

rule	transistor count	comments
$y[0,1]$	32	
u	13	
xe	8	
total	53	

CMOS-implementable PRS

$$
\begin{array}{lcl}
\neg u & \rightarrow & \_u\uparrow\\
u & \rightarrow & \_u\downarrow
\end{array}
$$

$$
\begin{array}{lcl@{\qquad\qquad}lcl}
ye \wedge \_u \wedge d0 & \rightarrow & \_y0\downarrow & ye \wedge \_u \wedge d1 & \rightarrow & \_y1\downarrow \\
\neg ye \wedge \neg\_u & \rightarrow & \_y0\uparrow & \neg ye \wedge \neg\_u & \rightarrow & \_y1\uparrow
\end{array}
$$

$$
\begin{array}{lcl@{\qquad\qquad}lcl}
\neg\_y0 \vee \neg\_y1 & \rightarrow & vy\uparrow & d0 \vee d1 & \rightarrow & \_vd\downarrow \\
\_y0 \wedge \_y1 & \rightarrow & vy\downarrow & \neg d0 \wedge \neg d1 & \rightarrow & \_vd\uparrow
\end{array}
$$

$$
\begin{array}{lcl}
\neg\_vd & \rightarrow & \_\_vd\uparrow \\
\_vd & \rightarrow & \_\_vd\downarrow
\end{array}
$$

$$
\begin{array}{lcl}
\neg\_vy \wedge \neg\_vd & \rightarrow & u\uparrow \\
\_vy \wedge \_vd & \rightarrow & u\downarrow
\end{array}
$$

$$
\begin{array}{lcl}
\neg\_u & \rightarrow & \_\_u\uparrow \\
\_u & \rightarrow & \_\_u\downarrow
\end{array}
$$

$$
\begin{array}{lcl}
ye \wedge \_\_u \wedge \_vy & \rightarrow & xe\uparrow \\
\neg ye \wedge (\neg\_\_u \vee \neg\_vy) & \rightarrow & xe\downarrow
\end{array}
$$

D.26 Chain serializer C

The C process in the chain serializer is a C-element and costs 8 transistors.

$$
\begin{array}{lcl}
xe \wedge po & \rightarrow & s\downarrow \\
\neg xe \wedge \neg po & \rightarrow & s\uparrow
\end{array}
$$

D.27 Serial merge

The receiver is responsible for sending spikes to neurons and data to the neuron and synapse configuration memory. This designs uses an arbiter to handle concurrent input so that we can send spikes and write to the configuration memeory on the fly.
For M inputs and 1-of-D encoding,

$$
\begin{array}{l}
*[[\\
\quad \langle | \, m : M : xmi \longrightarrow yo\uparrow; [yi]; xmo\uparrow; [\neg xmi]; yo\downarrow; [\neg yi]; xmo\downarrow \rangle \\
\;]]
\end{array}
$$

$$
\begin{array}{l}
*[[\\
\quad \langle [] m : M : \langle [] d : D : xmd \longrightarrow yd\uparrow; xmo\downarrow; [\neg xmd]; yd\downarrow; xmo\downarrow \rangle \rangle \\
\;]]
\end{array}
$$

For M=2 and D=2,

$$
\begin{array}{l}
*\texttt{[[}x0i \longrightarrow yo\uparrow;\ \texttt{[}yi\texttt{]};\ x0o\uparrow;\ \texttt{[}\neg x0i\texttt{]};\ yo\downarrow;\ \texttt{[}\neg yi\texttt{]};\ x0o\downarrow \\
\quad \texttt{|}\,x1i \longrightarrow yo\uparrow;\ \texttt{[}yi\texttt{]};\ x1o\uparrow;\ \texttt{[}\neg x1i\texttt{]};\ yo\downarrow;\ \texttt{[}\neg yi\texttt{]};\ x1o\downarrow \\
\ \texttt{]]}
\end{array}
$$

$$
\begin{array}{l}
*\texttt{[[}x00 \longrightarrow y0\uparrow;\ \texttt{[}\neg yi\texttt{]};\ x0o\downarrow;\ \texttt{[}\neg x00\texttt{]};\ y0\downarrow;\ \texttt{[}yi\texttt{]};\ x0o\downarrow \\
\quad \texttt{[]}\,x01 \longrightarrow y1\uparrow;\ \texttt{[}\neg yi\texttt{]};\ x0o\downarrow;\ \texttt{[}\neg x01\texttt{]};\ y1\downarrow;\ \texttt{[}yi\texttt{]};\ x0o\downarrow \\
\quad \texttt{[]}\,x10 \longrightarrow y0\uparrow;\ \texttt{[}\neg yi\texttt{]};\ x1o\downarrow;\ \texttt{[}\neg x10\texttt{]};\ y0\downarrow;\ \texttt{[}yi\texttt{]};\ x1o\downarrow \\
\quad \texttt{[]}\,x11 \longrightarrow y1\uparrow;\ \texttt{[}\neg yi\texttt{]};\ x1o\downarrow;\ \texttt{[}\neg x11\texttt{]};\ y1\downarrow;\ \texttt{[}yi\texttt{]};\ x1o\downarrow \\
\ \texttt{]]}
\end{array}
$$

PRS

The parent requests and grants are handled by the standard n-way arbiter with its parent ports exposed. Otherwise,

$$
\begin{array}{lcl@{\qquad\qquad}lcl}
x00 \lor x10 & \to & y0\uparrow & x01 \lor x11 & \to & y1\uparrow \\
\neg x00 \land \neg x10 & \to & y0\downarrow & \neg x01 \land \neg x11 & \to & y1\downarrow
\end{array}
$$

CMOS-implementable PRS:

$$
\begin{array}{lcl@{\qquad\qquad}lcl}
x00 \lor x10 & \to & \_y0\uparrow & x01 \lor x11 & \to & \_y1\uparrow \\
\neg x00 \land \neg x10 & \to & \_y0\downarrow & \neg x01 \land \neg x11 & \to & \_y1\downarrow
\end{array}
$$

$$
\begin{array}{lcl@{\qquad\qquad}lcl}
\neg\_y0 & \to & \_\_y0\downarrow & \neg\_y1 & \to & \_\_y1\downarrow \\
\_y0 & \to & \_\_y0\uparrow & \_y1 & \to & \_\_y1\uparrow
\end{array}
$$

D.28 Memory

Each group of 4 neurons and 1 synapse needs at least 28 bits of memory. However, given the shape and size constraints of the memory, we may end up using larger memories. This memory will only be written, not read.

A memory consists of a two dimensional array of bitcells. Rows are addressed by a read/write lines, and columns are addressed by the data itself. That is, we write an entire row at a time.

The shape of the memory dictates the size of the input we deliver to it. For each write operation, we indicate the address (which write line) and data to write. The address is encoded in 1-of-2 or 1-of-4 words. The data is communicated as it will be written. It is cheaper to have write lines than data lines because the size of the encoded address scales with the logarithm of the number of write lines. So we usually maximize the number of write lines within the aspect ratio constraint of the neuron layout. Further, using 1-of-4 encoding requires that we use at least 2 data bits.

Here are some memory sizes and their required deserializer bits:

neurons/synapses	memory bits	write lines	write bits	data bits	total bits	word size	words
4/1	32	32	5	1	6	2	6
16/4	128	64	6	2	8	4	4
16/4	128	32	5	4	9	4	4.5
4096/1024	32768	16384	14	2	16	4	8

We can reduce the number of words if the memory supports banking, a level of address heirarchy on top of the address itself.

Appendix E

Serial Router Supplement

This appendix describes some components that were not used in the serial H-Tree router described in Section 3.4

E.1 Half-Cycle Slack Buffer

The serial router as laid out did not contain any pipelining. As a result, its throughput was throttled by additional unpipelined communications that extended well into the datapath. For reference, here is a buffer process that provides a half-cycle of slack. Buffers could be placed in the cutouts of the router and between the router and datapath to improve the router's throughput.

HSE

$$
\begin{array}{l}
*[[x_\phi \longrightarrow y_\phi\uparrow; x_e\uparrow, [y_e] \\
\;\; [\!]\, x_0 \longrightarrow y_0\uparrow; x_e\downarrow; [\neg y_e \wedge \neg x_0]; y_0\downarrow; x_e\uparrow; [y_e] \\
\;\; [\!]\, x_1 \longrightarrow y_1\uparrow; x_e\downarrow; [\neg y_e \wedge \neg x_1]; y_1\downarrow; x_e\uparrow; [y_e] \\
\;\; [\!]\, \neg x_\phi \longrightarrow y_\phi\downarrow; x_e\downarrow; [\neg y_e] \\
\;]]
\end{array}
$$

PRS

$$
\begin{array}{lcl}
x_\phi \wedge \neg y_e & \to & y_\phi\uparrow \\
\neg x_\phi \wedge y_e & \to & y_\phi\downarrow
\end{array}
$$

$$
\begin{array}{lcl}
y_\phi \wedge \neg y_0 \wedge \neg y_1 & \to & x_e\uparrow \\
\neg y_\phi \vee y_0 \vee y_1 & \to & x_e\downarrow
\end{array}
$$

$$
\begin{array}{lclclcl}
x_0 \wedge y_e & \to & y_0\uparrow & \qquad & x_1 \wedge y_e & \to & y_1\uparrow \\
\neg x_0 \wedge \neg y_e & \to & y_0\downarrow & \qquad & \neg x_1 \wedge \neg y_e & \to & y_1\downarrow
\end{array}
$$

CMOS-Implementable PRS

$$\begin{array}{lcl} \neg x_\phi & \rightarrow & \_x_\phi\uparrow \\ x_\phi & \rightarrow & \_x_\phi\downarrow \end{array} \qquad \begin{array}{lcl} \neg\_x_\phi \wedge \neg y_e & \rightarrow & y_\phi\uparrow \\ \_x_\phi \wedge y_e & \rightarrow & y_\phi\downarrow \end{array}$$

$$\begin{array}{lcl} y_\phi \wedge \_y_0 \wedge \_y_1 & \rightarrow & \_x_e\downarrow \\ \neg y_\phi \vee \neg\_y_0 \vee \neg\_y_1 & \rightarrow & \_x_e\uparrow \end{array} \qquad \begin{array}{lcl} \neg\_x_e & \rightarrow & \_\_x_e\uparrow \\ \_x_e & \rightarrow & \_\_x_e\downarrow \end{array}$$

$$\begin{array}{lcl} x_0 \wedge y_e & \rightarrow & \_y_0\downarrow \\ \neg x_0 \wedge \neg y_e & \rightarrow & \_y_0\uparrow \end{array} \qquad \begin{array}{lcl} \neg\_y_0 & \rightarrow & \_\_y_0\uparrow \\ \_y_0 & \rightarrow & \_\_y_0\downarrow \end{array}$$

$$\begin{array}{lcl} x_1 \wedge y_e & \rightarrow & \_y_1\downarrow \\ \neg x_1 \wedge \neg y_e & \rightarrow & \_y_1\uparrow \end{array} \qquad \begin{array}{lcl} \neg\_y_1 & \rightarrow & \_\_y_1\uparrow \\ \_y_1 & \rightarrow & \_\_y_1\downarrow \end{array}$$

4-ary Accounting

rule	transistor count	comments
$\_x_\phi$	2	
y_ϕ	8	
$\_x_e$	10	
$\_\_x_e$	2	
$\_y[0,1,2,3]$	16	staticized by $\_\_y[0,1,2,3]$
$\_\_y[0,1,2,3]$	16	staticizes $\_y[0,1,2,3]$
total	54	

E.2 Greedy but Fair N-way Arbiter using Tree/Ring Sequencing

Although the serial router used unpipelined 4-way arbiters for minimal transistor counts, a designer may want an N-way arbiter with pipelining. This appendix presents a pipelined, N-way arbiter which unifies the ring and tree approaches by parameterizing a tree via K, its radix. When $K = 1$, the arbiter is a flat ring, and when $K = 2$, the arbiter is a binary tree. The tree is built from NODEs. The leaves of the tree service requests from N clients.

Accounting

NODES cost $30K+18$ transistors, where K is the radix of the tree.

Configuration	transistors
K-ary tree	$(30K+18)\frac{N-1}{K-1}$
Flat ring	$30N+18$
Binary tree	$78(N-1)$
4-ary tree	$46(N-1)$

E.3 NODE

For a binary tree,

$$
\begin{array}{l}
*\texttt{[[}\overline{C_0} \vee \overline{C_1}\texttt{]}; P \bullet (\\
\quad \texttt{[}\overline{C_0} \longrightarrow C_0; C_0 \,|\, \neg\overline{C_0} \longrightarrow \mathit{skip}\texttt{]};\\
\quad \texttt{[}\overline{C_1} \longrightarrow C_1; C_1 \,|\, \neg\overline{C_1} \longrightarrow \mathit{skip}\texttt{]})\\
\texttt{]}
\end{array}
$$

For a K-ary tree,

$$
\begin{array}{l}
*\texttt{[[}\langle \vee k : K : \overline{C_k}\rangle\texttt{]};\\
\quad P \bullet \langle ; k : K : \texttt{[}\overline{C_0} \longrightarrow C_0; C_0 \,|\, \neg\overline{C_0} \longrightarrow \mathit{skip}\texttt{]}\rangle\\
\texttt{]]}
\end{array}
$$

We decompose NODE into PRODUCER and CONSUMER processes.

Accounting

component	transistors/component	components/NODE	transistors/NODE
PRODUCER	18+K	1	18+K
CONSUMER	29	K	29K
		total transistors/NODE	30K+18

E.3.1 PRODUCER

PRODUCER produces parent requests upon sensing child requests and sequences between CONSUMERs. For a binary tree,

$$*\texttt{[[}\overline{C_0} \vee \overline{C_1}\texttt{]}; P \bullet \langle \bullet K : k : S_k\rangle\texttt{]]}$$

For a K-ary tree,

$$*\texttt{[[}\langle \vee K : k : \overline{C_k}\rangle\texttt{]}; P \bullet \langle \bullet K : k : S_k\rangle\texttt{]]}$$

We decompose PRODUCER into CTRL and SEQ.

Accounting

component	transistors/component	components/PRODUCER	transistors/PRODUCER
CTRL	18+K	1	18+K
SEQ	0	1	0
		total transistors/PRODUCER	18+K

CTRL

CTRL relays the child requests to the parent and initiates the sequencing between CONSUMERS.

CHP

For a binary tree,

$$*[[\overline{C_0} \vee \overline{C_1}]; P \bullet S]$$

For a K-ary tree,

$$*[[\langle \vee K : k : \overline{C_k} \rangle]; P \bullet S]$$

HSE

For a binary tree,

$$\begin{aligned}&*[[pe]; [c0r \vee c1r]; x\downarrow;\\ &\quad pr\uparrow; [\neg pe];\\ &\quad sr\uparrow; [sa]; x\uparrow; sr\downarrow; [\neg sa];\\ &\quad pr\downarrow]\end{aligned}$$

PRS

$$\begin{aligned}pe \wedge (c0r \vee c1r) &\rightarrow x\downarrow\\ sa &\rightarrow x\uparrow\end{aligned}$$

$$\begin{aligned}\neg x \vee sa &\rightarrow pr\uparrow\\ x \wedge \neg sa &\rightarrow pr\downarrow\end{aligned}$$

$$\begin{aligned}\neg x \wedge \neg pe &\rightarrow sr\uparrow\\ x \vee pe &\rightarrow sr\downarrow\end{aligned}$$

CMOS-implementable PRS

$$\begin{aligned}\_pe \wedge (c0r \vee c1r) &\rightarrow x\downarrow\\ \neg\_sa &\rightarrow x\uparrow\end{aligned}$$

$$\begin{aligned}\neg x \vee \neg\_sa &\rightarrow pr\uparrow\\ x \wedge \_sa &\rightarrow pr\downarrow\end{aligned}$$

$$\begin{aligned}\neg x \wedge \neg\_pe &\rightarrow sr\uparrow\\ x \vee \_pe &\rightarrow sr\downarrow\end{aligned}$$

$$\begin{aligned}pe &\rightarrow \_pe\downarrow & \qquad sa &\rightarrow \_sa\downarrow\\ \neg pe &\rightarrow \_pe\uparrow & \qquad \neg sa &\rightarrow \_sa\uparrow\end{aligned}$$

Accounting

rule	transistor count	comments
x	2+K+4	
pr	4	
sr	4	
pe	2	
$_sa$	2	
total	18+K	

SEQ

SEQ sequences through the consumers.

CHP

In general,

$$\mathit{SEQ}(K) \equiv$$
$$*[\mathit{SP} \bullet \langle \bullet k : K : \mathit{SC}_k \rangle]$$

For a binary tree,

$$*[\mathit{SP} \bullet \mathit{SC}_0 \bullet \mathit{SC}_1]$$

We build a K-way sequencer as K'-way sequencer tree defined recursively as

$$\mathit{SEQ}(K) \equiv \mathit{SEQ}(K') \parallel \langle \parallel k' : K' - 1 : \mathit{SEQ}(K/K') \rangle \parallel \mathit{SEQ}(K - (K' - 1)K/K')$$

where the recursion ends when $K \leq K'$. For example, a binary tree would be defined recursively as

$$\mathit{SEQ}(K) \quad = \quad \mathit{SEQ}(2) \parallel \mathit{SEQ}(K/2) \parallel \mathit{SEQ}(K - K/2)$$

The SC channels of the parent sequencer are the SP channels of the the child sequencers. The SC channels of the leaf sequencers connect to the consumers.

HSE

For a binary tree,

$$
\begin{array}{l}
*[[spr]; \\
\quad sc_0r\uparrow; [sc_0a]; sc_1r\uparrow; [sc_1a]; \\
\quad spa\uparrow; [\neg spr]; \\
\quad sc_0r\downarrow; [\neg sc_0a]; sc_1r\downarrow; [\neg sc_1a]; \\
\quad spa\downarrow \\
\;]
\end{array}
$$

In general,

$$
\begin{aligned}
&*[[spr];\\
&\quad \langle ; k' : K' : sc'_k r\uparrow; [sc'_k a] \rangle\\
&\quad spa\uparrow; [\neg spr];\\
&\quad \langle ; k' : K' : sc'_k r\downarrow; [\neg sc'_k a] \rangle\\
&\quad spa\downarrow\\
&\ ]
\end{aligned}
$$

PRS

$$
\begin{array}{lcl@{\qquad\qquad}lcl}
spr & \rightarrow & sc0r\uparrow & sc(k'-1)r & \rightarrow & spa\uparrow\\
\neg spr & \rightarrow & sc0r\downarrow & \neg sc(k'-1)r & \rightarrow & spa\downarrow
\end{array}
$$

$$
\begin{array}{lcl}
sck'a & \rightarrow & sc(k'+1)r\uparrow\\
\neg sck'a & \rightarrow & sc(k'+1)r\downarrow
\end{array}
$$

These are just wires, so all K'-way tree sequencers that implement a K-way sequencer are equivalent.

E.3.2 CONSUMER

CONSUMER checks and services requests from children.

CHP

$$
\begin{aligned}
&*[[\overline{C} \longrightarrow S; C\\
&\quad |\overline{S} \longrightarrow S]]
\end{aligned}
$$

which is a variation on the precise exceptions circuit of [28].

HSE

$$
\begin{aligned}
&*[[cr \longrightarrow [sr]; sa\uparrow; [\neg sr]; ce\downarrow; [\neg cr]; sa\downarrow; ce\uparrow\\
&\quad | sr \longrightarrow sa\uparrow; [\neg sr]; sa\downarrow\\
&\ ]]
\end{aligned}
$$

We break out nondeterministic selection.

$$
\begin{aligned}
&*[[cr \longrightarrow c\uparrow; [\neg cr]; c\downarrow\\
&\quad | sa \longrightarrow s\uparrow; [\neg sa]; s\downarrow\\
&\ ]]\\
&*[[c \longrightarrow [sr]; sta\uparrow; [\neg sr]; ce\downarrow; [\neg c]; sta\downarrow; ce\uparrow\\
&\quad [] s \longrightarrow sfa\uparrow; [\neg s]; sfa\downarrow\\
&\ ]]\\
&*[[sta \vee sfa]; sa\uparrow; [\neg sta \wedge \neg sfa]; sa\downarrow]
\end{aligned}
$$

PRS

$$\begin{array}{lcl} c \wedge sr & \rightarrow & sta\uparrow \\ \neg c & \rightarrow & sta\downarrow \end{array} \qquad \begin{array}{lcl} s & \rightarrow & sfa\uparrow \\ \neg s & \rightarrow & sfa\downarrow \end{array}$$

$$\begin{array}{lcl} \neg sta \vee sr & \rightarrow & ce\uparrow \\ sta \wedge \neg sr & \rightarrow & ce\downarrow \end{array}$$

$$\begin{array}{lcl} sta \vee sfa & \rightarrow & sa\uparrow \\ \neg sta \wedge \neg sfa & \rightarrow & sa\downarrow \end{array}$$

CMOS-implementable PRS

$$\begin{array}{lcl} c \wedge sr & \rightarrow & \_sta\downarrow \\ \neg c & \rightarrow & \_sta\uparrow \end{array} \qquad \begin{array}{lcl} s & \rightarrow & \_sfa\downarrow \\ \neg s & \rightarrow & \_sfa\uparrow \end{array}$$

$$\begin{array}{lcl} \_sta \vee sr & \rightarrow & \_ce\downarrow \\ \neg\_sta \wedge \neg sr & \rightarrow & \_ce\uparrow \end{array}$$

$$\begin{array}{lcl} \neg\_sta \vee \neg\_sfa & \rightarrow & sa\uparrow \\ \_sta \wedge \_sfa & \rightarrow & sa\downarrow \end{array}$$

Accounting

rule	transistor count	comments
$[c, s]$	12	2-way active-low arbiter
$\_sta$	7	
$\_sfa$	2	
$\_ce$	4	
sa	4	
total	29	

Bibliography

[1] Filipp Akopyan, Carlos Tadeo, Ortega Otero, and Rajit Manohar. Hybrid synchronous-asynchronous tool flow for emerging VLSI design. In *IEEE International Workshop on Logic Synthesis*, 2016.

[2] Ben Varkey Benjamin, Peiran Gao, Emmett McQuinn, Swadesh Choudhary, Anand R. Chandrasekaran, Jean Marie Bussat, Rodrigo Alvarez-Icaza, John V. Arthur, Paul A. Merolla, and Kwabena Boahen. Neurogrid: A mixed-analog-digital multichip system for large-scale neural simulations. *Proceedings of the IEEE*, 102(5):699–716, 2014.

[3] Kwabena Boahen. A neuromorph's prospectus. *Computing in Science and Engineering*, 19(2):14–28, 2017.

[4] Kwabena A Boahen. Point-to-point connectivity between neuromorphic chips using address-events. *IEEE Transactions on Circuits & Systems II: Analog and Digital Signal Processing*, 47(5):416–434, 1999.

[5] Kwabena A. Boahen. A burst-mode word-serial address-event link-I: transmitter design. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 51(7):1269–1280, 2004.

[6] Kwabena A Boahen. A burst-mode word-serial address-event link-II: receiver design. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 51(7), 2004.

[7] Kwabena A. Boahen. A burst-mode word-serial address-event link-III: testing and results. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 51(7):1292–1300, 2004.

[8] Martin Boerlin, Christian K. Machens, and Sophie Denève. Predictive coding of dynamical variables in balanced spiking networks. *PLoS Computational Biology*, 9:1–16, 2013.

[9] Alexander Borst and Frdric Theunissen. Information theory and neural coding. *Nature Neuroscience*, 2(11):947–957, 1999.

[10] Christian Brandli, Raphael Berner, Minhao Yang, Shih Chii Liu, and Tobi Delbruck. A 240 180 130 dB 3 μs latency global shutter spatiotemporal vision sensor. *IEEE Journal of Solid-State Circuits*, 49(10):2333–2341, 2014.

[11] T. Y W Choi, Paul A. Merolla, John V. Arthur, Kwabena A. Boahen, and Bertram E. Shi. Neuromorphic implementation of orientation hypercolumns. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 52-I:1049–1060, 2005.

[12] Swadesh Choudhary, Steven Sloan, Sam Fok, Alexander Neckar, Eric Trautmann, Peiran Gao, Terry Stewart, Chris Eliasmith, and Kwabena Boahen. Silicon neurons that compute. In *International Conference on Artificial Neural Networks*, pages 121–128, 2012.

[13] Matthieu Courbariaux, Yoshua Bengio, and Jean-Pierre David. Training deep neural networks with low precision multiplications. *ArXiv e-prints*, 2014.

[14] Hideyuki Cteau and Alex D. Reyes. Relation between single neuron and population spiking statistics and effects on network activity. *Physical Review Letters*, 2006.

[15] Chris Eliasmith and Charles H Anderson. *Neural engineering: computation, representation, and dynamics in neurobiological systems.* MIT press, 2004.

[16] William Ft Softky and Christof Koch. The highly irregular firing of cortical cells is inconsistent with temporal integration of random EPSPs. *The Journal of Neuroscience*, 13(l):334–350, 1993.

[17] Julius Georgiou, Andreas G Andreou, and Philippe 0 Pouliquen. A mixed analog/digital asynchronous processor for cortical computations in 3D SQl-CMOS. In *2006 IEEE International Symposium on Circuits and Systems*, pages 4 pp.–, 2006.

[18] Suzana Herculano-Houzel. The human brain in numbers: a linearly scaled-up primate brain. *Frontiers in Human Neuroscience*, 2009.

[19] Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Quantized neural networks: training neural networks with low precision weights and activations. *CoRR*, 2016.

[20] Eric Jonas and Konrad Paul Kording. Could a neuroscientist understand a microprocessor? *PLoS Computational Biology*, 13(1), 2017.

[21] Norman P. Jouppi, Al Borchers, Rick Boyle, Pierre-luc Cantin, Clifford Chao, Chris Clark, Jeremy Coriell, Mike Daley, Matt Dau, Jeffrey Dean, Ben Gelb, Cliff Young, Tara Vazir Ghaemmaghami, Rajendra Gottipati, William Gulland, Robert Hagmann, C. Richard Ho, Doug Hogberg, John Hu, Robert Hundt, Dan Hurt, Julian Ibarz, Nishant Patil, Aaron Jaffey, Alek Jaworski, Alexander Kaplan, Harshit Khaitan, Daniel Killebrew, Andy Koch, Naveen Kumar, Steve Lacy, James Laudon, James Law, David Patterson, Diemthu Le, Chris Leary, Zhuyuan Liu, Kyle Lucke, Alan Lundin, Gordon MacKean, Adriana Maggiore, Maire Mahony, Kieran Miller, Rahul Nagarajan, Gaurav Agrawal, Ravi Narayanaswami, Ray Ni, Kathy Nix, Thomas Norrie, Mark Omernick, Narayana Penukonda, Andy Phelps, Jonathan Ross, Matt Ross, Amir

Salek, Raminder Bajwa, Emad Samadiani, Chris Severn, Gregory Sizikov, Matthew Snelham, Jed Souter, Dan Steinberg, Andy Swing, Mercedes Tan, Gregory Thorson, Bo Tian, Sarah Bates, Horia Toma, Erick Tuttle, Vijay Vasudevan, Richard Walter, Walter Wang, Eric Wilcox, Doe Hyun Yoon, Suresh Bhatia, and Nan Boden. In-datacenter performance analysis of a Tensor Processing Unit. *ACM SIGARCH Computer Architecture News*, 45(2):1–12, 2017.

[22] Robert Karmazin, Carlos Tadeo Ortega Otero, and Rajit Manohar. cellTK: automated layout for asynchronous circuits with nonstandard cells. In *Proceedings - International Symposium on Asynchronous Circuits and Systems*, 2013.

[23] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. ImageNet Classification with Deep Convolutional Neural Networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[24] John Lazzaro, John Wawrzynek, M. Mahowald, Massimo Sivilotti, and Dave Gillespie. Silicon auditory processors as computer peripherals. *IEEE Transactions on Neural Networks*, 4(3):523–528, 1993.

[25] Joseph Lin and Kwabena Boahen. A delay-insensitive address-event link. In *IEEE Symposium on Asynchronous Circuits and Systems*, 2009.

[26] Benjamin Lindner. Superposition of many independent spike trains is generally not a Poisson process. *Physical Review E - Statistical, Nonlinear, and Soft Matter Physics*, 2006.

[27] M Mahowald. *An analog VLSI stereoscopic vision system*, volume 1. Springer, Boston, 1994.

[28] R. Manohar, M. Nyström, and A. J. Martin. Precise exceptions in asynchronous processors. In *Proceedings - 2001 Conference on Advanced Research in VLSI, ARVLSI 2001*, 2001.

[29] Rajit Manohar. Comparing stochastic and deterministic computing. *IEEE Computer Architecture Letters*, 14(2):119–122, 2015.

[30] Rajit Manohar and Yoram Moses. The eventual C-element theorem for delay-insensitive asynchronous circuits. In *2017 23rd IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC)*, pages 102–109, 2017.

[31] Alain Martin. The limitations to delay-insensitivity in asynchronous circuits. In *Proceedings of the 6th MIT Conference on Advanced Research in VLSI*. MIT Press, 1990.

[32] Alain J. Martin and Mika Nyström. Asynchronous techniques for system-on-chip design. *Proceedings of the IEEE*, 94(6):1089–1120, 2006.

[33] Carver Mead. *Analog VLSI and neural systems*. Addison-Wesley, 1989.

[34] Carver Mead. Neuromorphic electronic systems. *Proceedings of the IEEE*, 78(10):1629–1636, 1990.

[35] Samir Menon, Sam Fok, Alex Neckar, Oussama Khatib, and Kwabena Boahen. Controlling articulated robots in task-space with spiking silicon neurons. In *5th IEEE RAS/EMBS International Conference on Biomedical Robotics and Biomechatronics*, pages 181–186, 2014.

[36] Paul Merolla, John Arthur, Rodrigo Alvarez, Jean Marie Bussat, and Kwabena Boahen. A multicast tree router for multichip neuromorphic systems. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 61(3):820–833, 2014.

[37] Paul A Merolla, John V Arthur, Rodrigo Alvarez-Icaza, Andrew S Cassidy, Jun Sawada, Filipp Akopyan, Bryan L Jackson, Nabil Imam, Chen Guo, Yutaka Nakamura, Bernard Brezzo, Ivan Vo, Steven K Esser, Rathinakumar Appuswamy, Brian Taba, Arnon Amir, Myron D Flickner, William P Risk, Rajit Manohar, and Dharmendra S Modha. A million spiking-neuron integrated circuit with a scalable communication network and interface. *Science*, 345(6197):668–673, 2014.

[38] Paul A. Merolla, John V. Arthur, Bertram E. Shi, and Kwabena A. Boahen. Expandable networks for neuromorphic chips. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 54(2):301–311, 2007.

[39] Jongkil Park, Theodore Yu, Siddharth Joshi, Christoph Maier, and Gert Cauwenberghs. Hierarchical address event routing for reconfigurable large-scale neuromorphic systems. *IEEE Transactions on Neural Networks and Learning Systems*, 28(10):2408–2422, 2017.

[40] Sheldon Ross. *Stochastic Processes*. John Wiley & Sons, 2nd edition, 1996.

[41] Massimo Antonio Sivilotti. *Wiring considerations in analog VLSI systems, with application to field-programmable networks*. PhD thesis, California Institute of Technology, Pasadena, CA, USA, 1991.

[42] Bongki Son, Yunjae Suh, Sungho Kim, Heejae Jung, Jun Seok Kim, Changwoo Shin, Keunju Park, Kyoobin Lee, Jinman Park, Jooyeon Woo, Yohan Roh, Hyunku Lee, Yibing Wang, Ilia Ovsiannikov, and Hyunsurk Ryu. A 640480 dynamic vision sensor with a 9m pixel and 300Meps address-event representation. In *Digest of Technical Papers - IEEE International Solid-State Circuits Conference*, pages 66–67, 2017.

[43] Dominik Thalmeier, Marvin Uhlmann, Hilbert J. Kappen, and Raoul Martin Memmesheimer. Learning universal computations with spikes. *PLoS Computational Biology*, 12:1–29, 2016.

[44] John Von Neumann. *The Computer and the Brain*. Yale University Press, New Haven, 3rd edition, 2012.