Efficient Digital Neurons for Large Scale Cortical Architectures

James E. Smith
University of Wisconsin-Madison
jes@ece.wisc.edu

Abstract

Digital neurons are implemented with the goal of supporting research and development of architectures which implement the computational paradigm of the neocortex.

Four spiking digital neurons are implemented at the register transfer level in a manner that permits side-by-side comparisons. Two of the neurons contain two stages of exponential decay, one for synapse conductances and one for membrane potential. The other two neurons contain only one stage of exponential decay for membrane potential.

The two stage neurons respond to an input spike with a change in membrane potential that has a non-infinite leading edge slope; the one stage neurons exhibit a change in membrane potential with an abrupt, infinite leading edge slope. This leads to a behavioral difference when a number of input spikes occur in very close time proximity. However, the one stage neurons are as much as a factor of ten more energy efficient than the two stage neurons, as measured by the number of dynamic add-equivalent operations.

A new two stage neuron is proposed. This neuron reduces the number of decay components and implements decays in both stages via piece-wise linear approximation. Together, these simplifications yield two stage neuron behavior with energy efficiency that is only about a factor of two worse than the simplest one stage neuron.

1. Introduction

Computer architects and designers have begun the challenging task of constructing large scale computing systems targeted at emulating the operation of the mammalian neocortex. Simply put, the neocortex is the sensing, thinking, perceiving part of the brain. The discovery of its computational paradigm, along with the subsequent development of efficient systems that can emulate it, will be a truly revolutionary achievement in automated computation [32].

The motivation for architecture research on neuron-based computing devices is two-fold. 1) Currently, there is a need to support large scale experimentation in order to help discover the brain's computational paradigm [12]. 2) Eventually, as the paradigm is revealed, there will be demand for practical implementations of an entirely new type of computer – provided that the neuron level of abstraction is found to provide a good basis for efficient implementations.

Architectural approaches for constructing a silicon brain range from massive interconnections of conventional processors to ASICs and custom logic arrays. As with conventional von Neumann computing, these approaches balance the tradeoffs among generality, speed, and efficiency. In this work, the focus is toward the more hardware-oriented FPGA/ASIC/custom end of the architecture spectrum, where simplicity and efficiency are key.

We consider register transfer level (RTL) designs of spiking neurons. *Two stage* designs incorporate decay for both synaptic conductance and membrane potential (Section 2 provides an overview of neuron operation). Simpler *one stage* designs include only decay of membrane potential. Consequently the commonly used one stage designs are about an order of magnitude more energy efficient than the two stage designs.

However, there is a behavioral difference between the two stage and one stage designs that manifests itself in the relationship between input spikes appearing closely together in time and the resulting output spike latency. Depending on the larger system in which the neuron model is placed and the assumptions made by the system developer, this behavioral difference may or may not be significant.

In this paper, four existing neuron models, two with one stage and two with two stages, are implemented with behavior and efficiency as objectives. Using a consistent implementation style allows straightforward side-by-side comparisons. Through simulations both behavior and complexity are studied for the four implementations.

A new two stage digital neuron is then proposed. It provides timing behavior similar to the other two stage neurons, but is much more efficient. Although still less efficient than the simplest one stage implementation, it closes the gap to about a factor of two. Considering that other components of an interconnected cortical system will also consume significant energy, this makes the proposed two stage neuron a strong candidate for future large scale cortical architectures where two stage timing behavior is deemed important.

1.1 Related Work

Large scale simulations will be an important part of discovering the brain's function [12][18][20][30]. Some of this work is focused on biological accuracy, which, for example, will lead to better understanding and treatment of human maladies such as autism and Alzheimer's disease. In contrast, the goal of the work reported here is to better understand (and eventually replicate) the brain's computational paradigm; this means that the neuron models will be more abstract, and simpler, than the highly complex biologically accurate models.

978-1-4799-4394-4/14/$31.00 © 2014

Some approaches to cortical emulation use computer systems employing conventional processors and implement neuron operation purely in software. These include massively parallel computer systems[1][30], GPUs [10][31], and large server clusters [18]. The spiNNaker system [36] uses ARM processors embedded in special purpose chips employing Address Event Representation AER [8] for communicating spikes. An advantage of these software-based systems is that they enable broad flexibility in neuron modeling. Their neuron models can be more complex with little effect on overall efficiency because of the instruction-level overheads, both in numbers of bookkeeping instructions and lower level instruction processing.

In contrast, the digital neurons studied here are focused on hardware architectures specifically developed for neural implementation. This will allow faster and much more energy efficient implementations. Of projects of this type, the high profile DARPA SyNAPSE program funds two large scale hardware-based efforts with the eventual goal of implementing 10 billion digital neurons. The IBM effort [2][7][39] employs one stage digital neurons. The HP effort [24] is based on memristor technology. The applications for the digital neurons studied in this paper are aligned with the DARPA SyNAPSE objectives. Another approach targeted at large scale array implementation with analog neurons is the European FACETS project [38].

Also in the hardware domain are a number of FPGA-based projects where energy efficiency is a goal. For example, the work by Emery et al. [9] interconnects a large number of neurons via AER and uses single stage neurons similar to those studied here. The work by Upegi et al. [43] also uses single stage neurons for simplicity. The design described by Hellmich et al. [16] goes a step further and uses non-leaky neurons, which leads to a very simple neuron model, but which also separates it from the work reported here (as well as all the other related work). One thing the hardware-based implementations have in common is simple, energy efficient neurons.

Biological neurons communicate via action potentials (voltage spikes). Although spike rates were long thought to encode all the important information and are the basis of classical artificial neural network theory [26], it has become increasingly clear that individual spikes, and the precise relative timing of spikes, is a critical part of the brain's computational process [28][34][42]. This is not to diminish the importance of rate-based neuron research; it has led to many useful "brain-inspired" applications. However, the eventual goal is discovery and accurate emulation of a more biologically plausible computational paradigm, so in this paper, we consider spiking neurons, as does the above-cited work.

There are good arguments in favor of analog neurons [14] [38]. These arguments primarily center on energy efficiency. However, as with conventional computation, digital implementations have a number of significant advantages over analog implementations. It is not the goal here to argue the merits of digital versus analog neurons, however. Rather, the objective is to study efficient digital neuron designs and to balance efficiency and functionality with the eventual goal of practical biological scale digital architectures.

This work uses single compartment neuron models. These models capture behavior primarily at the neuron body, and form the basis for many computational studies; [21] [44][45] are a few arbitrarily-chosen examples. For biological accuracy, however, more complex multi-compartment models are required. For example, these models include dendrite behavior and interneuron delays. The single compartment neurons studied here can form the basis for computation. Adding interneuron delays is relatively straightforward. Dendritic computation can also be appended to a single compartment model, as done in the Hierarchical Temporal Memory model [15], for example.

Finally, the model due to Izhikevich [19] and the related Adaptive Exponential Model [5] lie at an interesting point in the efficiency versus complexity tradeoff curve. The Izhikevich model is an empirical model that attempts to characterize membrane potential in a more biologically accurate way via quadratic terms. Considering its biological accuracy, it is relatively simple and is used in some high performance, software-based designs; in GPUs [10][31], for example. However, for more hardware-based implementations, it is a step up in complexity beyond models studied here. Furthermore, the model structure is based on a number of parameters that, for good comparisons, must be carefully fit to the models of the type used here, and this fitting is known to be difficult [35]. Consequently, these models are an interesting topic for future exploration of tradeoffs, but they are not covered here.

1.2 Overview

As background, Section 2 briefly summarizes biological neuron behavior. Section 3 describes four digital neuron models to be studied: two one stage models and two two stage models. Section 4 discusses spiking neuron behavior and differentiates the behavior of one and two stage neuron models. Section 5 describes evaluation methods, the spike train benchmark to be used, and a metric for comparing spike trains. Section 6 briefly compares the behavior of the neuron models by observing their output spike trains when benchmark input spike trains are applied. Section 7 evaluates energy efficiency using an addition-equivalent operation count metric. Section 8 proposes a new two stage neuron with the behavior characteristics of the two stage neurons but with significantly better efficiency.

2. Biological Neuron Behavior

2.1 Neuron Components

Figure 1 illustrates general behavior of biological neurons. In the figure, a *synapse* connects a *pre-synaptic neuron* to a *post-synaptic neuron*. In reality, a neuron will have thousands of such synapses, connecting it to hundreds or thousands of other neurons. However, at any given time only about 10% of all synapses will affect neuron function [3].

As noted on the pre-synaptic neuron, the neuron *body* is surrounded by a *membrane* and is fed by inputs, the *dendrites*. It also has an output, the *axon* which is connected to the dendrites of many other neurons via synapses.

2.2 Dynamic Operation

Figure 1 shows three "probes" that indicate voltage levels at certain points. First, focus attention on the waveform shown at the axon of the pre-synaptic neuron and the waveform at the membrane of the post-synaptic neuron. Consider a sequence of events that starts with a spike, or action potential, being emitted from the pre-synaptic neuron. The spike travels along the axon and reaches the synapse connecting it with the post-synaptic neuron. At the synapse, it effectively opens a conductive gate via a relatively complex biological process. The conductive gate allows ions to flow into the post-synaptic neuron body, thereby raising the membrane potential (see the Excitatory Post Synaptic Potential (EPSP), waveform). Although not shown, a spike received at a synapse may alternatively invoke an Inhibitory Post Synaptic Potential (IPSP), which reduces the neuron's membrane potential. "PSP" refers to either inhibitory or excitatory post synaptic potentials.

A synapse has an associated "efficacy" or "weight", which controls its relative conductivity. A stronger synapse has higher conductivity, resulting in a PSP with higher amplitude.

After the conductive synapse gate is opened, it immediately starts to close with exponential decay, so the flow of ions into the post-synaptic neuron gradually diminishes. At the same time, ions leak from the neuron body, thereby decreasing the membrane potential with exponential decay, but with a longer time constant than the closing of conductive synapse gates. This combination of exponential decays with different time constants gives the EPSP its distinctive shape, as shown in the figure.

Finally, consider the more detailed waveform shown at the right side of Figure 1. As multiple spikes are received at a neuron's input synapses, each of them will invoke a PSP on the post-synaptic neuron. If received relatively closely together in time, the PSPs will accumulate, as shown in the graph, raising the total membrane potential. If the potential reaches a key level, the *threshold voltage*, then an avalanche effect takes place, and the neuron emits an output spike. Immediately following the output spike, there is a *refractory period*, during which the neuron cannot fire again. If there are insufficient input spikes to raise the membrane potential to the threshold voltage, it will eventually decay back to the rest potential and no spike will be emitted.

In the waveform at the right side of Figure 1, typical voltage levels are shown. The refractory time is on the order of a few msec. In the illustration, only about three input spikes in close proximity are sufficient to raise the body potential to the threshold. In reality, the number is about an order of magnitude higher, but can vary over a wide range.

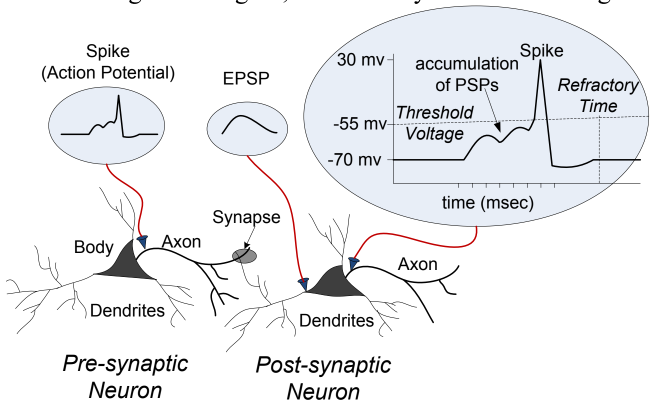


Figure 1. Neuron operation. Two neurons are connected via a synapse. Attached "probes" illustrate dynamic operation.

2.3 Synaptic Plasticity

Synaptic weight plasticity is a critical part of the neuronal learning process – the weights can change dynamically, depending on spiking behavior. In this paper, synaptic weights are static, although distributed over a range of values. This is justifiable in two scenarios. First, with offline training, the computational device is trained and the weights are established prior to operation; during normal operation they do not change. Second, even with dynamic online training and plastic synapses, weights are often modeled in a pseudo-static manner, with updates occurring at a much coarser time scale than neuron core operation [21] [38]. For example, in [21], weight updates are performed once every 10,000 time steps (assuming .1 msec time steps as done here).

3. Spiking Neuron Models

Much of the work in neuron modeling, beginning with the earliest work, is targeted at understanding the behavior of biological neurons. However, evolution under biological constraints suggests that some (perhaps many) aspects of biological neuron operation are extraneous with respect to the underlying computational paradigm. Hence, it is important to recognize the divergence between biological and computational models that arise as simplifications are made in order to improve computational speed and efficiency.

Regardless of whether the objective is biological or computational accuracy, neuron models have a common ancestry. In the next subsection, the classical Hodgkin Huxley model is first described to establish a mathematical framework. Then a series of four spiking neuron models and their efficient digital implementations are described.

3.1 Hodgkin Huxley Model

The Hodgkin Huxley (HH) Model [17] is the classic neuron model, for which its developers won the 1963 Nobel Prize in medicine. At the top level, the HH model is based on an RC circuit (Figure 2) which characterizes the membrane potential. From left to right, the RC model consists of 1) a conductance and reversal voltage (g_t^I and V^I) for inhibitory synapses; the conductance is a function of time, as denoted by the t subscript, 2) a conductance and reversal voltage (g_t^E and V^E) for excitatory synapses (note the difference in polarity for the two reversal voltages), 3) a membrane (leakage) conductance and rest voltage (g^M and V^{rest}), and 4) a membrane capacitance C^M and voltage V_t. In biologically realistic models, coupled differential equations describing the variable conductances are quite complex and are not given here. In the following, significantly simplified versions of the conductance equations are used.

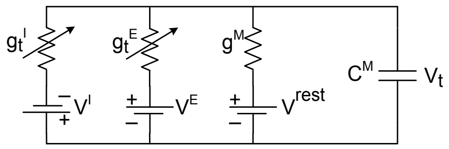


Figure 2. RC circuit which is the basis for the HH Model.

A feature of the HH model is biological accuracy, e.g., it characterizes the entire dynamic waveform for the membrane potential. However, from a computational perspective, it appears that the spiking input-output behavior is what is important. That is, all one really needs is functional input/output spiking behavior that can support the neuron's computational capabilities. This leads to simpler models as are widely used in computational neuroscience.

3.2 Leaky Integrate and Fire (LIF) Models

As originally defined [41], an LIF model incorporates a membrane potential that decays exponentially with some time constant. Over time, a number of different LIF-based models have been developed. What typically distinguishes LIF models is the modeling of synapse conductance in response to input spikes. The simplest LIF neurons do not directly model synapse conductances; an input spike causes a step change in the membrane potential. These simple LIF neurons are discussed in Section 3.4.

A commonly used (and more accurate) LIF model contains synaptic conductances that decay with time. To distinguish this specific LIF model, it is referred to in this paper as the *DLIF model*, an LIF model with *decaying* synaptic conductances. The DLIF model is based on a differential equation that describes membrane potential at time t, V_t.

(1) $C^M \, dV_t/dt = -g^M (V_t - V^{rest}) - g_t^E(V_t - V^E) - g_t^I (V_t - V^I)$

If Vt > V^{th} (threshold voltage), the neuron fires a spike and resets to the V^{rest}*, where it remains for refractory time t_{ref}.*

If an excitatory input spike is received through synapse i at time t, then $s_{ti} = 1$; else $s_{ti} = 0$. Summing over all the input synapses i, at time t, the excitatory conductance g_t^E is:

$$g_t^E \leftarrow g_{t-1}^E + \sum s_{ti} w_i g^E_{max}$$

Where w_i is the weight of synapse i ($0 \le w_i \le 1$), and g^E_{max} is the maximum excitatory conductance. Similarly, for an inhibitory synapse:

$$g_t^I \leftarrow g_{t-1}^I + \sum s_{ti} w_j g^I_{max}$$

Meanwhile, the synaptic conductances decays satisfy the differential equations:

$$\tau_E \, dg_t^E/dt = -g_t^E \quad \text{and} \quad \tau_I \, dg_t^I/dt = -g_t^I$$

where τ_E and τ_I are the respective time constants.

Beginning with the mathematical description just given, one can derive an efficient digital implementation for computing the neuron membrane potential, and, consequently, the spiking behavior of the DLIF neuron.

3.2.1 Voltage Shifting and Scaling

The first optimization for digital implementation efficiency shifts voltage levels so that $V^{rest} = 0$, this shifts the other voltage levels accordingly:

$$V^E \leftarrow V^E - V^{rest}\ ; \quad V^I \leftarrow V^I - V^{rest}$$

This optimization is commonly done. We continue on with additional optimizations, however. We scale the V values by dividing by threshold voltage V^{th}. This yields membrane potentials represented as fixed point fractions, so testing for crossing the firing threshold is reduced to detecting when the membrane potential becomes 1 or greater.
Note: to simplify notation in the remainder of the paper the same notation is used for both the pre- and post-shifted and scaled versions of voltage levels.

3.2.2 Maintaining Membrane Potential

Setting $V^{rest} = 0$ in equation (1) and dividing through by C^M yields the following.

$$dV_t/dt = - g^M V_t / C^M - g_t^E(V_t - V^E) / C^M - g_t^I (V_t - V^I) / C^M$$

For synchronous operation, we form a discrete time version with time step Δt:

$$(V_t - V_{t-1})/\Delta t = -V_{t-1}(g^M + g_t^E + g_t^I)/ C^M + (V^E g_t^E + V^I g_t^I)/ C^M$$

Defining membrane leakage time constant: $\tau_m = C^M/g^M$ and applying a series of straightforward algebraic operations:

(2) $V_t = V_t\text{-}1[(1- \Delta t/\tau_m) - (g_t^E \Delta t/C^M + g_t^I \Delta t/C^M)]$
$\qquad + (V^E g_t^E \Delta t/C^M + V^I g_t^I \Delta t/C^M)$

Equation (2) will be mapped directly into a digital implementation. Next, we deal with digital synapses.

3.2.3 Synapse Conductances

The synapse conductances g_t^E and g_t^I are modeled as:

$$\tau_E \, dg_t^E/dt = -g_t^E \quad \text{and} \quad \tau_I \, dg_t^I/dt = -g_t^I$$

Converting to discrete form, plus simple algebra yields:

$$g_t^E = g_{t-1}^E \ (1 - \Delta t /\tau_E)\ ; \qquad g_t^I = g_{t-1}^I \ (1 - \Delta t /\tau_I)$$

Adding the weighted synaptic inputs each clock cycle:

(3) $g_t^E = g_{t-1}^E(1-\Delta t /\tau_E) + \sum s_{ti} w_i \, g_M^E;$
(4) $g_t^I = g_{t-1}^I(1 - \Delta t /\tau_I) + \sum s_{ti} w_j g_M^I$

3.2.4 Putting It All Together

In equation (2), $\Delta t/C^M$ is a constant. We use the distributive property to push this constant back into equations (3) and (4) and define: $W_i = w_i (g^M \Delta t/C^M)$. This meta-weight eliminates a number of implied constant multiplications in (2). It also means that the modified versions of (3) and (4) produce dimensionless quantities that are voltage multipliers, rather than conductances. To avoid confusion, the "*g*" quantities are replaced with "*d*" quantities. Then, we have:

$$d_t^E = d_{t-1}^E (1 - \Delta t / \tau_E) + \sum s_{ti} W_i$$
$$d_t^I = d_{t-1}^I (1 - \Delta t / \tau_I) + \sum s_{tj} W_j$$
$$V_t = V_{t-1} [(1 - \Delta t/\tau_M) - (d_t^E + d_t^I)] + (V^E d_t^E + V^I d_t^I)$$

The EPSP for a single input spike is illustrated in Figure 3a. The RTL that implements these equations is in Figure 4a. In Figure 4 the numbers in braces {*n*} identify sources of dynamic operation counts to be referenced later (Section 7).

The DLIF is a *two stage* model. The synapse stage (on the left) has latches holding the dimensionless multiplier (d) values, along with a multiplicative decay, followed by the membrane stage (on the right) with a latch level that holds the membrane potential and its multiplicative decay.

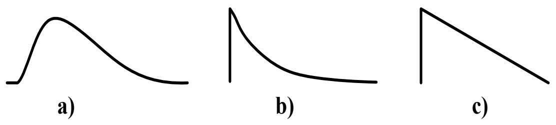


Figure 3. Excitatory Post Synaptic Potentials in response to a single input spike:
a) DLIF and DSRM0 neurons
b) SLIF neuron,
c) LLIF neuron.

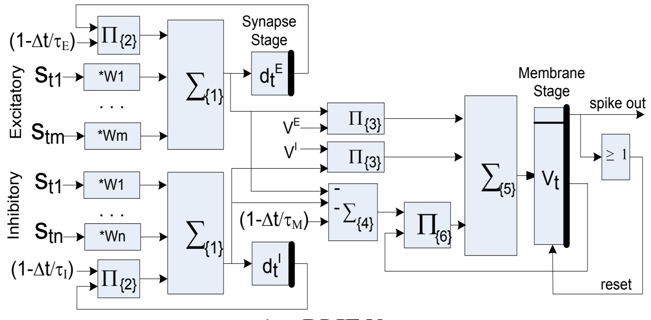
3.3 Spike Response Models (SRM0)

SRM0 is the "zeroth order" version of the more general Spike Response Model [13][25]. In the SRM0 model individual spike responses (Figure 3a) are assumed to be independent and are simply summed to yield the membrane potential. Besides leading to a simplification of the DLIF digital neuron, the SRM0 assumption also simplifies analysis of neuron computational capabilities.

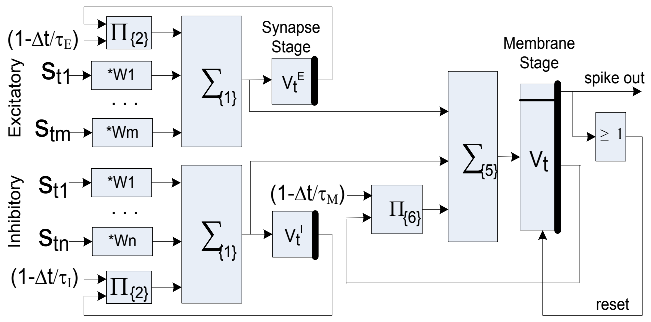
If we use the RC circuit in Figure 2 to characterize the response to individual spikes and make the SRM0 assumption regarding independence of spike responses then, after several steps of derivation omitted for brevity, we arrive at a neuron which we call the *DSRM0* neuron. It is similar to the DLIF neuron except the term $-(d_t^E + d_t^I)\ \Delta t/C^M$ in the DLIF neuron is not present in the DSRM0 neuron. This removed term reflects interactions among spike responses that affect the membrane potential in a relatively small way. Removing this term is not only a simplification by itself, but it also allows the multiplications of constants V^E and V^I to be pushed back into the synapse meta-weights to yield the following.

$$W_i = w_i (V^E g^E_{max}\ \Delta t/ C^M) \text{ and } W_j = w_j (V^I g^I_{max}\ \Delta t/ C^M)$$

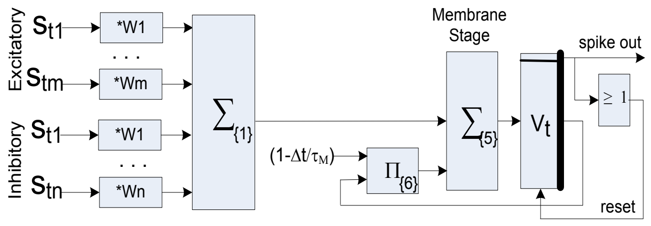
This also changes the quantities being maintained in the synapse stage to voltages. Scaling voltages so that $V^{th} = 1$, as was done with the DLIF neuron, yields the implementation in Figure 4b.



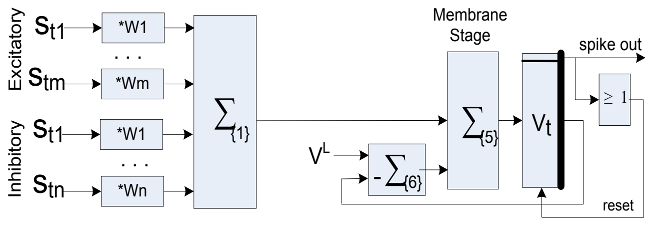
a) DLIF Neuron



b) DSRM0 Neuron



c) SLIF Neuron



d) LLIF Neuron

Figure 4. A complexity-ordered sequence of digital neurons. Refractory logic is not shown.

3.4 LIF with Step Inputs (SLIF) Model

For the next in the sequence of digital neuron simplifications, we begin with the neuron model defined by Stein [41]. This model uses a simple step to model the effect an input spike has on the membrane potential, yielding:

$$V_t = V_{t-1} (1 - \Delta t/\tau_m) + \sum s_{tj}\, v^E_{max}\, w_j + \sum s_{ti}\, v^I_{max}\, w_i$$

Where v^E_{max} and v^I_{max} are a spike's maximum contributions to membrane potential (occurring when $w_i = 1$ or $w_j = 1$, respectively). As a simplification, we define meta-weights $W_i = w_i*v^E_{max}$, $W_j = w_j*v^I_{max}$, in a manner similar to that used for the previous neurons. This is the *SLIF* neuron; its EPSP for a single input spike is in Figure 3b and the RTL implementation is in Figure 4c.

3.5 Linear Leak Integrate and Fire (LLIF)

The fourth neuron model to be considered is a further simplification which performs linear decay of the membrane potential rather than exponential decay[33][39][43]. This means that the decay can be performed by subtracting a constant rather than multiplying by a constant. The equation for membrane potential follows.

$$V_t = \max(V_{t-1} - V^L, 0) + \sum s_{ti}\, v^E_{max}\, w_j + \sum s_{tj}\, v^I_{max}\, w_j$$

V^L is a constant leak value which is repeatedly subtracted from the membrane voltage. This neuron is referred to as a Linear Leak Integrate and Fire (*LLIF*) neuron. Its EPSP for a single input spike is in Figure 3c and the RTL implementation is in Figure 4d.

4. Neuron Behavior

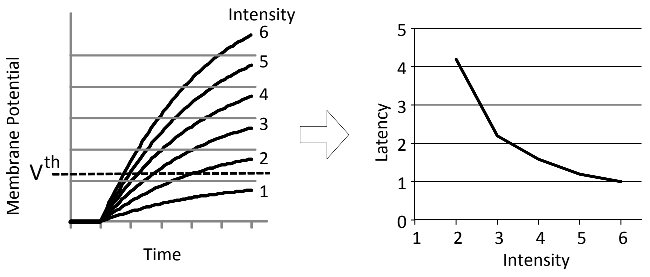
In this section, the four neuron implementations are analyzed in terms of spike-based input-output behavior. Then, in subsequent sections, their spike response behaviors and relative computational efficiencies are compared.

The motivation for using spiking neurons is that precise timing relationships are often critical for the brain's computational paradigm. This is articulated in papers [4][29][42] summarizing experimental data that strongly suggests that timing of individual spikes (rather than rates alone) are essential for some of the low latency computations that a mammalian brain performs. Maass [28] provides strong theoretical arguments for spike-based computation, as opposed to rate-based computation.

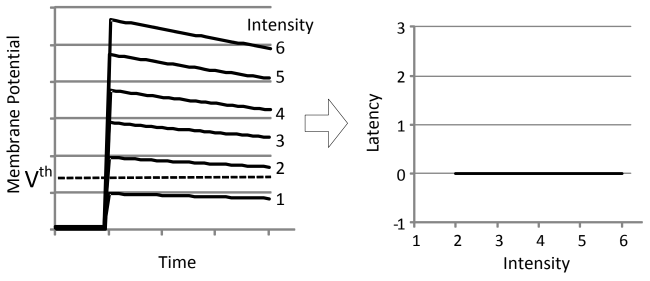
Maass also identified the leading edge slope of a PSP as a potentially significant computational feature for individual neurons [27]. Qualitatively, the key difference between the one and two stage neurons is that the PSP leading edge of the one stage neurons has an infinite slope and the PSP of the two stage neurons has a non-infinite leading edge slope (refer to Figure 4). This difference leads to different input-output transfer characteristics. This is illustrated via an example in Figure 5.

The leading edge slope affects input-output behavior when a number of spikes occur closely in time so that their leading edge slopes overlap. Informally, we will say that a neuron's input spike *intensity* is greater if there are more input spikes appearing closer together in time. Figure 5a left shows membrane potential for the DSRM0 neuron where a number of spikes (ranging from one to six) occur simultaneously; this case, chosen for purposes of illustration, exhibits maximum leading edge overlap. The response to fewer spikes takes longer to reach the threshold V^{th} than the response to more spikes. Figure 5a right shows the resulting relationship between the input intensity (number of simultaneous spikes) and the output spike latency. Although it does not satisfy the SRM0 assumption of PSP independence, the response for the DLIF neuron is very similar.

In contrast, Figure 5b left shows the spike response and latency vs. intensity relationships for the SLIF neuron. In this case, the latency is the same regardless of the intensity, as long as the intensity is sufficient to reach the threshold. Although it does not fit satisfy the SRM0 assumption of PSP independence, the response for the LLIF neuron is similar.



a) Two Stage Neuron (DSRM0)



b) One Stage Neuron (SLIF)

Figure 5. Behavior of neuron models as the number of simultaneous input spikes (intensity) is increased.

Some modelers prefer to construct systems that include background "noise" which is added to a neuron model's behavior [6][37]. Consequently, a set of simulations were performed where Gaussian noise was added to the membrane potentials at each time step. The sigma value for Gaussian noise was chosen so that with no input spike stimulus, the spontaneous output spike rate was about 5 Hz. Then, the number of simultaneous input spikes was varied, and the latency to the output spike was measured. Results were averaged over 500 trials. The results as latency vs. intensity are in Figure 6.

The x-axis shows spike intensity and the y-axis shows the average latency. Because of the presence of noise, when intensity is relatively low, some trials result in no output spike; this is indicated by dotted lines in the curves for the cases where fewer than 50% of the trials result in a spike. With added noise, the profile of the SLIF neuron now has the same general shape of the DSRM0 profile. However, the DSRM0 behavior remains distinguishable from the behavior of the SLIF neuron. First, at the end of the curve with high intensity, the response of the SLIF neuron is somewhat flatter than the response of the DSRM0 neuron; which is qualitatively similar to the noise-free case, but less extreme. At lower intensity, the SLIF neuron shows an upward curve, but a higher fraction of the responses show no output spike. Hence, there remains a behavioral difference between the one stage and two stage neurons in the presence of added background noise, although the difference is smaller than the noise-free case, and overall shapes of the curves are now more similar.

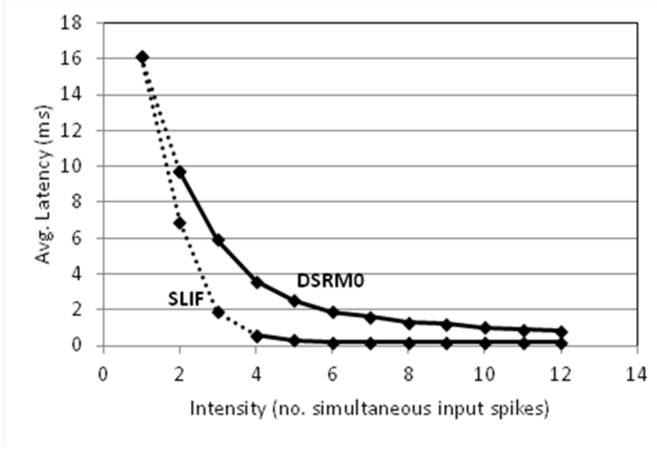


Figure 6. Average latency vs. spike intensity (# simultaneous spikes) for SLIF neuron (lower curve) and DSRM0 neuron (upper curve). Dotted lines indicate cases where fewer than 50% of trials resulted in an output spike.

A final, somewhat related, point is that using a one stage neuron with its step input can simplify event driven modeling. When a spike arrives at a one stage neuron's input, it can immediately be determined if and when the neuron will generate an output spike. Hence, a spike event can be placed immediately in event queue (s) without the need for later adjustments to the queue(s). In this paper, however, we do not consider event driven simulation, rather we assume clock cycle simulation.

5. Evaluation Methods

The four digital neurons just described differ with respect to spiking behavior and energy efficiency. In this section, methods for evaluating both spiking behavior and efficiency are described.

5.1 Simulator

Evaluations are simulation-based. Neuron simulation models were coded using GNU Octave (a Matlab clone). The models take multiple spike trains as inputs and generate a single output spike train. As a byproduct, simulation models maintain important internal neuron values. For example, the membrane potential is common to all the neuron models.

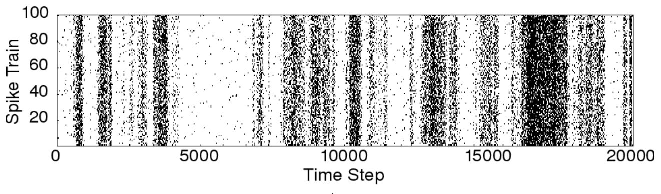
5.2 Spike Train Benchmark

The simulation models are configured to have 100 synapses each, with a ratio of 80% excitatory and 20% inhibitory (a typical ratio found in a real cortex region).

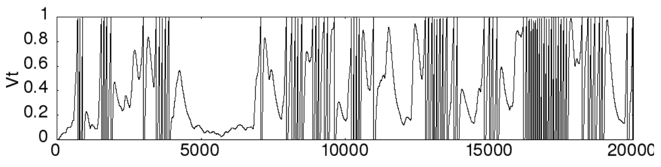
The benchmark consists of correlated, randomly distributed input spike trains as per Vogels et al. [44]. The method first generates a "white noise" signal: random numbers between -.5 and .5. Then, this random sequence is filtered and normalized to -1, 1. Next, the signal is rectified and renormalized to .1. Finally 5*Δt (.5 msec) is added to assure a background firing rate of 5 HZ. This waveform is used for generating all 100 input spike trains. For each spike train, at each time step, a random number between 0 and 1 is generated. If the random number is less than the waveform value then a spike is generated; else there is no spike. The raster diagram for all 100 spike trains is shown in Figure 7a.

For benchmark runs, synapse weights were set according to a lognormal distribution [40]. The mean for excitatory synapses is .55, and for inhibitory synapses it is .24. These weights, coupled with the 80/20 distribution of excitation and inhibition, yield output spike trains containing about 70 spikes for the 20,000 time step (2 seconds) simulations. Figure 7b shows the membrane potential for the DLIF neuron; the threshold is normalized to one, and the places spikes occur is evident.

Note that the above normalization value of .1 was chosen to produce an overall spike rate of about 35 Hz, a value in the normal active range. If a more realistic number of synaptic inputs were modeled, say one thousand, then the normalization value would be reduced to produce roughly the same number of output spikes. Consequently, the overall synaptic activity reported here (Section 7) would be essentially unchanged for the larger number of synapses.



a)



b)

Figure 7. a) Raster diagram of benchmark input spike trains. Trains 1-79 are excitatory, and trains 80-100 are inhibitory. b) Resulting membrane potential for DLIF neuron.

5.3 Comparing Spike Trains

For comparing output spike trains, the coincidence measure defined by Gerstner's group is used [22][25].

$$\Gamma = (N_{coinc} - \langle N_{coinc} \rangle) * .5 (N_{data} + N_{model}) * (1/N)$$

Where N_{data} is the number of spikes in the reference spike train (DLIF will be the reference), N_{model} is the number of spikes in the spike train to be compared with the reference, and N_{coinc} is the number of coincidences between the spike trains with precision δ. $\langle N_{coinc} \rangle = 2f\delta N_{data}$ is the number of coincidences that would occur for spikes generated by a homogeneous Poisson process with the same rate as the model spike train. The last factor, 1/N, normalizes Γ to a maximum of one ($N = 1 - 2f\delta$). Γ = 1 only if both spike trains are entirely coincident (within time δ). δ = 2 msec as in [22]. Because the refractory time, t_{ref} = 5 msec, "coincidence" is unambiguous. If the number of coincidences is the same as if the model were a homogeneous Poisson process with the same number of spikes (i.e. random chance), then Γ= 0. Note that Γ can be negative, i.e., the coincidence can be worse than random chance would predict.

5.4 Model Calibration

An important consideration when comparing neuron models is that structural differences in the models lead to different sets of model parameters. Consequently, one would like to establish model parameters so that input/output behavior among models is as similar as possible, thereby enabling more meaningful efficiency and functionality comparisons.

To establish parameters across the models, the most complex neuron, DLIF, is used as a reference. Then, given the reference and its parameters, the parameters of the other neurons chosen by maximizing their Γ metric.

Parameters for the DLIF neuron are in Table 1. These numbers are within the ranges of real neurons, keeping in mind that in real neurons the parameter ranges are fairly wide. The table shows the original parameter value, and the value used in the simulation model after it has been shifted and/or scaled to make the implementation more efficient (see Section 3.2.1).

Table 1. Simulated DLIF Neuron Parameters

DLIF Parameter	**Value *original***	**Value *shifted/scaled***
g^E	.14 nS	.014 nS (scaled)
g^I	.35 nS	.035 nS (scaled)
g^M	10 nS	1 nS (scaled)
V^{rest}	-60 mV	0 mV (shifted &scaled)
V^{th}	-50 mV	1 mV (shifted & scaled)
V^E	0 mV	6 mV (shifted & scaled)
V^I	-80 mV	-2mV (shifted & scaled)
t_{ref}	5 msec	5 msec
τ_M	20 msec	20 msec
τ_E	5 msec	5 msec
τ_I	10 msec	10 msec

The fitting process for the other neurons is straightforward. Parameters for the other neurons are fit as follows.

1) DSRM0: the time constants are adjusted, but are kept in the same ratio as in the DLIF neuron. All the other parameters are the same as in the DLIF neuron.
2) SLIF: The DLIF parameters are used to determine the maximum EPSP and IPSP values that can be achieved. Then, these values are multiplied by a fitted scaling parameter to determine the step change in membrane potential. The membrane decay time constant is the second fitted parameter.
3) LLIF: The step increase in membrane potential is modeled as in the SLIF neuron. The linear decrement value V^L is an additional fitted parameter.

Also, the one stage neurons have an inherently lower latency than the two stage neurons because their membrane potential changes instantaneously in response to an input spike. This causes a systematic time shift in the neurons' output spikes. Consequently, to get the best fit, it is necessary to time shift (delay) the output spike trains of the one stage neurons. Parameters for the neurons, including the time shifts, are given in Table 2. Also given are the corresponding Γ values.

Table 2. Parameters Fit to DLIF Neuron

DSRM0 Fitted Parameters	**Γ = 1.0**
τ_M	17.675 msec
τ_E	4.15 msec
τ_I	8.3 msec
SLIF Fitted Parameters	**Γ = .87**
V^E_{max}	.19 mV
V^I_{max}	-.3mV
τ_M	12.5 msec
time shift	+1.5msec
LLIF Fitted Parameters	**Γ = .80**
V^E_{max}	.21 mV
V^I_{max}	-.33 mV
V^L	.06 mV
time shift	+1.2 msec

6. Behavior Comparison

The Γ measure, as given in Table 2, provides one way of comparing output spike similarity. The DSRM0 neuron yields maximum similarity to the DLIF reference neuron, Γ = 1.0. Scaling the time constants for both the synaptic and membrane decays in the DSRM0 neuron can largely compensate for the structural simplification with respect to the DLIF neuron.

The one stage neurons have less similarity with the DLIF reference neuron, but their Γ values of .80 and .87 are still relatively close (as compared with results in [22], for example), and, to be fair, this measure by itself is not especially significant. Real neurons of the same type typically deviate from each other more than the one stage neurons deviate from the DLIF neuron.

To illustrate the effect of the PSP leading edge slope discussed in Section 4, the benchmark spike trains were excerpted and used in the following way. First, a region of high spiking activity was identified – between time steps 16500 and 16650. Then, in that region the neurons were simulated for a series of runs, where the number of excitatory spike trains was incrementally increased from 40 to 80, while all 20 inhibitory spike trains were active. Hence, each run incrementally increased the overlapping spike responses, but with some randomness in the actual spike patterns.

The results are in Figure 8 for three neurons: DLIF, LLIF, and PSRM0 (see Section 8). To reduce clutter, the other two neurons are not shown, but DSRM0 is similar to DLIF, and SLIF is similar to LLIF, as would be expected.

The behavior difference due to the PSP leading edge slope is apparent, but is more subtle than in the explicitly constructed example in Figure 5. Each set of data points essentially plots the relative latency as a function of the number of input spike trains. The two stage DLIF neuron shows a greater ability to resolve differences in intensity than LLIF, a one stage neuron. Quantitatively, the DLIF neuron resolves the input intensities to output spikes at 31 different time steps, while the LLIF neuron has output spikes at 20 different time steps. One small region where the difference is clear is enclosed in an ellipse. This region shows behavior very much like the behavior predicted in Figure 5.

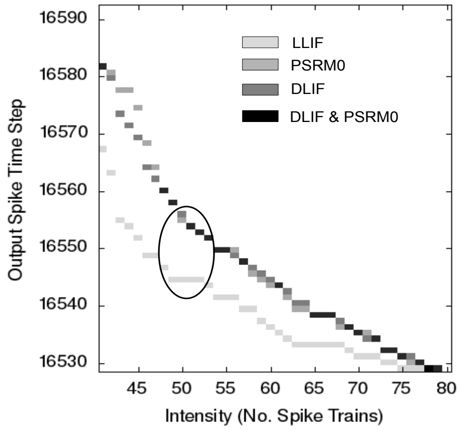


Figure 8. Latency (time to first spike) vs. spike intensity (number of active input excitatory spike trains).

7. Efficiency

A digital neuron with high energy efficiency is of critical importance when constructing a large scale system. The primary operations in the RTL implementations are addition and multiplication, so we use a simple efficiency metric which counts the number of addition-equivalent operations.

7.1 Reducing Multiplications to Additions

Multiplications are converted to addition-equivalents, and, in the process, the implementations are made more efficient via multiplier recoding. The multiplications in the DLIF and DSRM0 neurons have constant operands (e.g., the decay values). These constants are not very precise in the first place (they are biologically based), and, with multiplier recoding, they can be approximated with a relatively small number of bits, thereby reducing each multiplication to a small number of additions (e.g., two or three).

For example, from Table 2, for the DSRM0 neuron, τ_M =.017675, and Δt = .1 msec, so the decay multiplier is $1-\Delta t/\tau_M$ = 0.994342291. This is approximately 0.994628906 = 1- .00000001011 in binary. Implementing the multiplication is three additions (and/or subtractions) of four operands, all shifted versions of the scaled membrane potential.

If one performs truncation and recoding on a decay value, the reduction of implied multiplications to numbers of additions is shown in Figure 9. Each symbol in the figure corresponds to a decay value (on the x-axis), and the row the symbol appears in (y-axis) is the number additions required to achieve a multiplication by the given decay value.

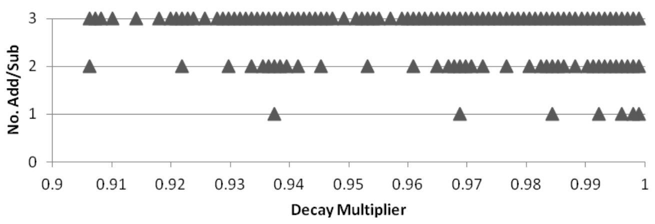


Figure 9. Decay multiplications reduced to additions.

In row 1, the values correspond to the subtraction of a single shifted version of the membrane potential from itself. I.e. the top value in the first column is $1-2^{-11}$ (all fractions were truncated to 11 bits). This set of decay values formed with only one addition appears overly coarse. The set for two additions will often be sufficient (row 2), but there are still some significant gaps. With three additions (row 3), it appears likely that all the practical decay values can be approximated with sufficient precision. For example, if we reduce the decay multiplications in the DSRM0 neuron to three addition-equivalents each and compare output spike trains for the benchmark with the full precision constants given in Table 2, the similarity metric Γ= 1.0.

7.2 Results

After decay multiplications are replaced by three additions, add-equivalent dynamic operation counts for the benchmark are shown in Figure 10. The neuron in the rightmost column will be discussed in Section 8. It is assumed that a zero operand causes the corresponding addition to be gated off and is not included in the count. Note that for the DLIF neuron, one can also convert the multiplications by constant V^E and V^I to three additions. Operation counts are annotated with the braced numbers in the Figure 4 schematics.

{1} ***Synapse Adds*** – The two stage neurons perform more operations in adding synapse weights due to the additional operands into the synapse adders in their first stage.

{2} ***Synapse Decays*** – These are performed only by the two stage neurons, and this is where more operations are performed than in any other part of the implementations.

{3}, {4} ***Compute Potential Changes*** and ***Compute Decay Value*** – The DLIF neuron performs these two operations which model interaction among spike responses (see Section 3.3). The two multiplications by the reversal voltages consume a large number of operations, yet, as observed earlier, they have relatively little effect on spiking behavior.

{5} ***Sum Membrane Inputs*** – Both the DLIF and DSRM0 neurons must sum data from stage one before passing it to stage two. This is not done in the one stage neurons.

{6} ***Membrane Decay*** – All except the LLIF neuron use a multiplicative decay (three add-equivalents) while the LLIF neuron performs a single subtraction of the linear leak value. Furthermore, the LLIF membrane potential is zero more often, so the decay operation is gated off more often. Hence, the LLIF neuron performs about a quarter the membrane decay operations as the two stage neurons.

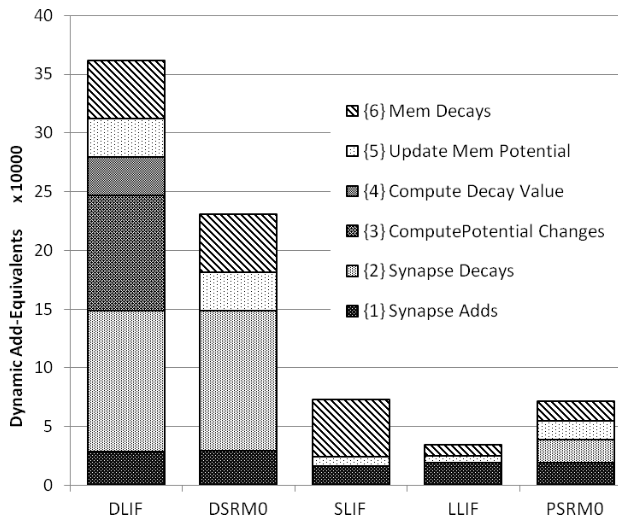


Figure 10. Dynamic operation counts for the benchmark spike trains. Numbers in braces identify components in Figure 4 and Figure 11 RTL.

To summarize briefly: the LLIF neuron requires fewer than a tenth the operations of the DLIF neuron. The DSRM0 neuron requires fewer operations than the DLIF, but it still requires almost eight times as many operations as the LLIF neuron. Meanwhile, the SLIF neuron requires about twice the number of operations as the LLIF neuron.

With regard to the above analysis, there is an important caveat: these efficiency numbers are just for the neuron and synapses. The energy required to distribute output spikes throughout a large network of neurons will be significant compared with the core neuron energy.

8. An Efficient Two Stage Neuron

Depending on the overall design objectives and assumptions, a designer may want a two stage neuron for computational reasons (Section 4), yet, as just shown, there is a very large gap between the one and two stage neurons when it comes to efficiency. In this section, two ways of reducing efficiency of a two stage neuron are proposed and evaluated. When both features are used in combination, the result is a new, highly efficient two stage neuron.

8.1 Combined Synaptic Decay

The DSRM0 neuron is the starting point. Because the largest single contributor to the DSRM0's total operation count is synapse decays, it is the first target for simplification. Then in the next subsection, both synapse and membrane decay operations are simplified.

Excitatory and inhibitory conductances decay with different time constants. Roughly half the synaptic decay operations can be eliminated if they both decay with the same time constant. So, one can use τ_E for both and compensate for the decrease in τ_I by adjusting (increasing) the inhibitory synapse weights. Through simulations, it was determined that the conductances can be combined with little change in input/output behavior if the inhibitory weights are increased by a factor of 1.90, which makes intuitive sense because the inhibitory time constant is decreased by a factor of 2.0. When this optimization is implemented in the DSRM0 neuron, it cuts the operations for synapse decays and updating membrane potential roughly in half (Figure 10 {2} and {5}).

8.2 Piecewise Linear Approximation of Decay

In Section 7.1, the decay multiplication was reduced to three (or fewer) additions. In this section, we go further and reduce the decay operation to a single subtraction. Rather than subtracting a single leak value throughout the range of membrane potentials as with the LLIF neuron, we use a series of leak values that form a piecewise linear approximation to exponential decay.

If the decay time constant is τ, then the half-life is $\tau\ln2$. In discrete terms, decaying the scaled membrane potential from 2^{-n} to 2^{-n-1} will take $\tau\ln2/\Delta t$ time intervals. Therefore, within the range of 2^{-n} to 2^{-n-1} one can use constant decrement amount $2^{-n-1} *\Delta t / \tau\ln2$. For example, to decay the membrane potential with τ_M = 20 msec, Δt = .1 msec, the membrane decay operand is D^M = .00347. So, for membrane potentials $2^{-n} > V_t \geq 2^{-n-1}$, the decrement value is 2^{-n-1} * .00347. At each time step, the implementation produces the decay decrement value as a binary shift of the constant D^M. The implementation logic determines the position of the leading one in the value being decayed (e.g., the membrane potential V_t). If the weight of this bit position is 2^{-n-1}, then the decrement value is D^M shifted right by n-1 bit positions.

In a two stage neuron, one can apply this piecewise linear approximation in both stages: the synapse decay logic and the membrane decay logic.

8.3 Two Stage Neuron with Piece-wise Linear Decay

If the simplifications in the two preceding subsections are combined, the result is a neuron with piecewise-linear decay (denoted *PSRM0*); see Figure 11. This is a two stage neuron that has the latency vs. intensity relationship as in Figure 5a.

After parameter fitting the decay constants and comparing the output spike trains with the DLIF neuron, its Γ metric is .82 for the spike train benchmark; in the same ballpark as the LLIF and SLIF neurons. Its functional behavior is very similar to the DLIF neuron (Figure 8). Finally, Figure 10 shows that the PSRM0 neuron requires slightly more than a factor of 2 more add-equivalent operations than the LLIF neuron. So, we get functional characteristics of a two stage neuron, with efficiency that is much closer to that of the LLIF neuron than the other two stage neurons.

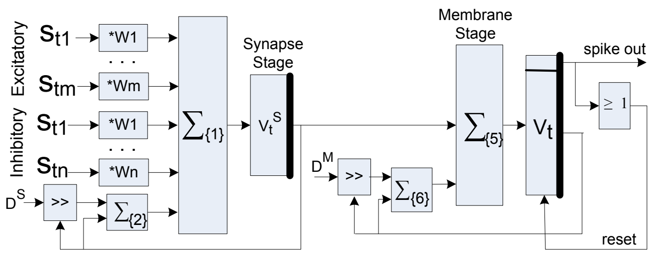


Figure 11. PSRM0 neuron with combined synapses and piece-wise linear decay.

Finally, note that logic for the shift operations is not counted in operation estimates; it will require more detailed design to get a better energy estimate. However, the shift constants are at least quasi-static, because the shift value is unchanged as decaying values pass through a given range.

9. Conclusions

The focus in this paper has been on digital neurons to support architectures for discovery and future implementation of the brain's computational paradigm. So, conclusions are drawn from that perspective.

Two stage neurons provide a PSP with a sloping leading edge, while the one stage neurons with their single step inputs to membrane potential do not. This leads to a behavioral difference which may or may not be important depending on the system in which the neuron model is placed and the assumptions made by the designer. On the other hand one stage neurons are much more energy efficient and may have advantages in event-driven system implementations. Furthermore, if background noise is added and spiking behavior is averaged over a number of trials, the behavioral differences are less than in the noise-free case.

Given that the brain's computing paradigm is not yet known, a computer architect/researcher must make an important decision: either the PSP leading edge slope is important for implementing the computational paradigm being studied, or it is not. If it is, then one should use a two stage neuron. If not, then a one stage neuron is the clear winner.

Of the two stage neurons, the DSRM0 neuron is about 30% more efficient than the DLIF. And, it can be argued that the extra operations in the DLIF neuron are only an artifact of biological implementation, and are unnecessary for capturing the brain's computational paradigm. Computational neuroscientists implicitly make this argument when they use the SRM0 model.

By 1) combining the synaptic decays (and compensating by adjusting weights) and 2) implementing piecewise linear decay, the PSRM0 neuron provides a sloping leading edge PSP and gives efficiencies several factors better than the DSRM0 implementation. This makes it an excellent choice for a large scale system where the behavior of two stage neurons is desired.

If one were to choose a one stage neuron, then the LLIF neuron is twice as efficient as the SLIF, and does not appear to have any significant computational disadvantages. So, for sheer efficiency, the LLIF neuron is the winner. Moreover, from a functional perspective, multiple LLIF neurons can be interconnected to give more complex spike timing relationships, but efficiency becomes worse [7].

The selection of a two stage neuron also adds fuel to the analog vs. digital debate. Using a two stage neuron will only widen the efficiency gap highlighted by Joubert et al. [23]. However, as has already been pointed out, one should consider the efficiency of the entire system, including the passing of spikes through a large interconnection structure, before making any final decisions regarding efficiency.

Finally, an assumption underpinning all research into hardware neuron implementations is that individual neurons will form the basic building blocks for future large scale systems. It may turn out that once the paradigm is better understood, a higher level of abstraction, say the cortical column, may be a better basis (see [11] for example), and individual neurons do not have to be modeled. If this is the case, then the functionality / efficiency tradeoffs shift significantly.

10. Acknowledgements

I thank Mikko Lipasti, Atif Hashmi, and Andy Nere for providing the impetus for pursuing this line of research. I am also very grateful to Doug Burger and Olivier Temam for guidance during the revision process. I very much appreciate the helpful comments from the anonymous reviewers, and I acknowledge one of the reviewers for recommending the inclusion of background noise in Section 4.

REFERENCES

[1] Ananthanarayanan, R., S. K. Esser, H. D. Simon, and D. S. Modha, "The Cat is Out of the Bag: Cortical Simulations with 10^9 Neurons, 10^{13} Synapses", *IEEE Conference on High Performance Computing Networking, Storage and Analysis*, pp. 1-12, 2009.

[2] Arthur, J. V., P. A. Merolla, F. Akopyan, R. Alvarez, A. Cassidy, S. Chandra, S. Esser, N. Imamy, W. Risk, D. Rubin, R. Manohary, and D. Modha, "Building Block of a Programmable Neuromorphic Substrate: A Digital Neurosynaptic Core", *2012 International Joint Conference on Neural Networks*, pp. 1-8, 2012.

[3] Barbour, B., N. Brunel, V. Hakim, and J.-P. Nadal, "What Can We Learn from Synaptic Weight Distributions", *TRENDS in Neurosciences,* 30, no. 12, pp. 622-629, 2007.

[4] Bohte, S. M., "The Evidence for Neural Information Processing with Precise Spike-times: A Survey", *Natural Computing,* 3, no. 2, pp. 195-206, 2004.

[5] Brette, R., and W. Gerstner, "Adaptive Exponential Integrate-and-Fire Model as an Effective Description of Neuronal Activity," Journal *of Neurophysiology,* 94.5, pp. 3637-3642, 2005.

[6] Brunel, N., "Persistent Activity and the Single-Cell Frequency-Current Curve in a Cortical Network Model", *Network: Computation in Neural Systems*, 11, no. 4 pp. 261-280, 2000.

[7] Cassidy, A. S., P. Merolla, J. V. Arthur, S. K. Esser, B. Jackson, R. Alvarez-Icaza, P. Datta, J. Sawaday, T. M. Wong, V. Feldman, A. Amir, D. B.-D. Rubinx, F. Akopyan, E. McQuinn, W. P. Risk, and D. S. Modha, "Cognitive Computing Building Block: A Versatile and Efficient Digital Neuron Model for Neurosynaptic Cores", *International Joint Conference on Neural Networks*, 2013.

[8] Deiss, S. R., R. J. Douglas, and A. M. Whatley, "A Pulse-Coded Communications Infrastructure for Neuromorphic Systems", *Pulsed Neural Networks*, pp. 157-178, 1999.

[9] Emery, R., A. Yakovlev, and G. Chester, "Connection-Centric Network for Spiking Neural Networks", *3rd ACM/IEEE International Symposium on Networks-on-Chip*, pp. 144-152, 2009.

[10] Fidjeland, A. K., E. B. Roesch, M. P. Shanahan, W. Luk, "Nemo: A Platform For Neural Modelling of Spiking Neurons Using GPUs," *Application-specific Systems, Architectures and Processors*, 2009.

[11] George, D., and J. Hawkins, "Towards A Mathematical Theory Of Cortical Micro-Circuits", *PLoS Computational Biology* 5.10, 2009.

[12] Gerstner, W., H. Sprekeler, and G. Deco, "Theory and Simulation in Neuroscience", *Science* 338.6103, pp. 60-65, 2012.

[13] Gerstner, W., and W. M. Kistler, *Spiking Neuron Models: Single Neurons, Populations, Plasticity*, Cambridge University Press, 2002.

[14] Giacomo I., B. Linares-Barranco, T. J. Hamilton, A. van Schaik, R. Etienne-Cummings, T. Delbruck, S.-C. Liu, P. Dudek, P. Häfliger, Sylvie Renaud, J. Schemmel, G. Cauwenberghs, J. Arthur, K. Hynna, F. Folowosele, S. Saighi, T. Serrano-Gotarredona, J. Wijekoon, Y.Wang and K. Boahen, "Neuromorphic Silicon Neuron Circuits", *Frontiers in Neuroscience* 5, 2011.

[15] Hawkins, J., S. Ahmad, and D. Dubinsky, "Hierarchical Temporal Memory Including HTM Cortical Learning Algorithms", *Technical Report*, Numenta, Inc, Palto Alto, 2010.

[16] Hellmich, H. H., M. Geike, P. Griep, P. Mahr, M. Rafanelli, and H. Klar, "Emulation Engine for Spiking Neurons and Adaptive Synaptic Weights", *IEEE International Joint Conference on Neural Networks*, pp. 3261-3266, 2005.

[17] Hodgkin, A., and A. Huxley, "A Quantitative Description of Membrane Current and Its Application to Conduction and Excitation in Nerve", *The Journal of Physiology*, 117.4, pp. 500-544, 1952.

[18] Izhikevich, E. M., and G. M. Edelman, "Large-Scale Model Of Mammalian Thalamocortical Systems", *Proceedings of The National Academy Of Science*, 105.9, pp. 3593-3598, 2008.

[19] Izhikevich, E. M , "Simple Model of Spiking Neurons", *IEEE Transactions on Neural Networks,* 14.6, pp. 1569-1572, 2003.

[20] Izhikevich, E. M., J. Gally, and G. Edelman, "Spike-Timing Dynamics of Neuronal Groups", *Cerebral Cortex,* 14.8, pp. 933-944, 2004.

[21] Izhikevich, E. M., "Polychronization: Computation with Spikes", *Neural Computatio*n, 18.2, pp. 245-282, 2006.

[22] Jolivet, R., R. Kobayashi, A. Rauch, R. Naud, S. Shinomoto, W. Gerstner, "A Benchmark Test for a Quantitative Assessment of Simple Neuron Models", *Journal of Neuroscience Methods,* 169, pp. 417-424, 2008.

[23] Joubert, A., B. Belhadj, O. Temam, and R. Héliot, "Hardware Spiking Neurons Design: Analog or Digital?", *International Joint Conference on Neural Networks,* pp. 1-5, 2012.

[24] Kim, K.-H., S. Gaba, D. Wheeler, J. M. Cruz-Albrecht, T. Hussain, N. Srinivasa, and W. Lu, "A Functional Hybrid Memristor Crossbar-Array/CMOS System for Data Storage and Neuromorphic Applications", *NanoLletters* 12.1, pp. 389-395, 2011.

[25] Kistler, W. M., W. Gerstner, and J. L. van Hemmen, "Reduction Of The Hodgkin-Huxley Equations to a Single-Variable Threshold Model", *Neural Computation* 9.5, pp. 1015-1045, 1997.

[26] Lippmann, R., "An Introduction to Computing with Neural Nets", *ASSP Magazine*, IEEE 4.2, pp. 4-22, 1987.

[27] Maass, W., "Networks of Spiking Neurons: The Third Generation of Neural Network Models," *Neural Networks* 10.9, pp.1659-1671, 1997.

[28] Maass, W. "Computing with Spikes." *Special Issue on Foundations of Information Processing of TELEMATIK* 8.1 pp. 32-36, 2002.

[29] Mainen, Z. F., and T. J. Sejnowski, "Reliability of Spike Timing in Neocortical Neurons", *Science* 268, pp. 1503-1506, 1995.

[30] Markram, H., "The Blue Brain Project", *Nature Reviews Neuroscience,* 7, no. 2 pp. 153-160, 2006.

[31] Nageswaran, J., N. Dutt, J. L. Krichmar, A. Nicolau, A. Veidenbaum, "Efficient Simulation of Large-Scale Spiking Neural Networks Using CUDA Graphics Processors", *International Joint Conference on Neural Networks,* 2009.

[32] National Academy of Engineering, "Reverse-Engineer the Brain", http://www.engineeringchallenges.org/cms/8996/9109.aspx, 2012.

[33] Nere, A., A. Hashmi, M. H. Lipasti, and G. Tononi, "Bridging the Semantic Gap: Emulating Biological Neuronal Behaviors with Simple Digital Neurons", *HPCA*, pp. 472-483. 2013.

[34] Paugam-Moisy, H. and S. M. Bohte, "Computing with Spiking Neuron Networks," *Handbook of Natural Computing*, Springer, 2009.

[35] Pospischil, M., Z. Piwkowska, T. Bal, A. Destexhe , "Comparison of Different Neuron Models to Conductance-Based Post-Stimulus Time Histograms Obtained in Cortical Pyramidal Cells Using Dynamic-Clamp in Vitro", *Biological cybernetics* 105.2 , pp.167-180, 2011.

[36] Rast, A. D., M. Khan, X. Jin, L. A. Plana, and S. B. Furber, "A Universal Abstract-Time Platform for Real-Time Neural Networks*", The Relevance of the Time Domain to Neural Network Models*. Springer US, 12, pp.135-157, 2012.

[37] Rauch, A., G. La Camera, H. R. Luscher, W. Senn, and S. Fusi, "Neocortical Pyramidal Cells Respond as Integrate-And-Fire Neurons to in Vivo-Like Input Currents," *Journal of neurophysiology* 90, no. 3 pp. 1598-1612, 2003.

[38] Schemmel, Johannes, D. Bruderle, A. Grubl, M. Hock, K. Meier, and S. Millner, "A Wafer-Scale Neuromorphic Hardware System for Large-Scale Neural Modeling", *International Symposium on Circuits and Systems*, pp. 1947-1950, 2010.

[39] Seo, J.-S., B. Brezzo, Y. Liu, B. D. Parker, S. K. Esser, R. K. Montoye, B. Rajendran. J. A. Tierno, L. Chang, D. S. Modha, and D. J. Friedman, "A 45nm CMOS Neuromorphic Chip with a Scalable Architecture for Learning in Networks of Spiking Neurons", *Custom Integrated Circuits Conference*, pp. 1-4, 2011.

[40] Song, S., P. J. Sjöström, M. Reigl, S. Nelson, and D. B. Chklovskii, "Highly Nonrandom Features of Synaptic Connectivity in Local Cortical Circuits", *PLoS Biology*, 3(3), 2005.

[41] Stein, R. B. "A Theoretical Analysis Of Neuronal Variability", *Biophysical Journal*, 5.2, pp. 173-194, 1965.

[42] Thorpe, Simon J., and M. Imbert. "Biological Constraints on Connectionist Modelling", *Connectionism in perspective* pp. 63-92, 1989.

[43] Upegui, A., C. A. Peña-Reyes, and E. Sanchez, "An FPGA Platform For On-Line Topology Exploration of Spiking Neural Networks", *Microprocessors and Microsystems,* 29.5, pp. 211-223, 2005.

[44] Vogels, T. P., H. Sprekeler, F. Zenke, C. Clopath, and W.Gerstner, "Inhibitory Plasticity Balances Excitation and Inhibition in Sensory Pathways and Memory Networks", *Science* 334.6062, pp.1569-1573, 2011

[45] Vogels, T. P., and L. F. Abbott, "Signal Propagation and Logic Gating in Networks of Integrate-and-Fire Neurons," The *Journal of Neuroscience* 25.46 pp. 10786-10795, 2005.