Programmable Logic Construction Kits for Hyper-Real-Time Neuronal Modeling

Ruben Guerrero-Rivera
rg66@leicester.ac.uk
Center for Bioengineering, University of Leicester, Leicester LE1 7RH, U.K.

Abigail Morrison
abigail@biologie.uni-freiburg.de
Markus Diesmann
diesmann@biologie.uni-freiburg.de
Computational Neurophysics and Bernstein Center for Computational Neuroscience, Institute of Biology III, Albert-Ludwigs-University, Freiburg, Germany

Tim C. Pearce
t.c.pearce@le.ac.uk
Center for Bioengineering, University of Leicester, Leicester LE1 7RH, U.K.

Programmable logic designs are presented that achieve exact integration of leaky integrate-and-fire soma and dynamical synapse neuronal models and incorporate spike-time dependent plasticity and axonal delays. Highly accurate numerical performance has been achieved by modifying simpler forward-Euler-based circuitry requiring minimal circuit allocation, which, as we show, behaves equivalently to exact integration. These designs have been implemented and simulated at the behavioral and physical device levels, demonstrating close agreement with both numerical and analytical results. By exploiting finely grained parallelism and single clock cycle numerical iteration, these designs achieve simulation speeds at least five orders of magnitude faster than the nervous system, termed here *hyper-real-time operation*, when deployed on commercially available field-programmable gate array (FPGA) devices. Taken together, our designs form a programmable logic construction kit of commonly used neuronal model elements that supports the building of large and complex architectures of spiking neuron networks for real-time neuromorphic implementation, neurophysiological interfacing, or efficient parameter space investigations.

1 Introduction

Numerical simulation of large-scale networks of spiking neurons and real-time neuromorphic system implementation requires significant computational power. On the other hand, existing general-purpose microprocessors, although extremely versatile, rely largely on serial processing of data, which severely limits their computational throughput. This serial dependence comes about through centralized arithmetic resources (such as arithmetic logic unit or floating point unit), which are restricted to one or a small number of concurrent operations. This class of numerical simulation problem, due to its inherent parallelism, is challenging for single-core processor architectures, rendering real-time operation impossible for all but the simplest of neural systems. The fact that cluster computing approaches are so successful in speeding up neuronal simulations demonstrates this serial bottleneck in computation. Clearly, then, single-core microprocessor-based neural simulators offer flexibility but are limited by serial processing constraints.

Fully customized hardware (such as analog VLSI) potentially offers high computational power at the expense of flexibility and design iteration times. Consequently there is a need for a rapid prototyping platform for neuronal models that is extremely fast with similar flexibility to general-purpose microprocessors. Field-programmable hardware (in the forms of field programmable gate arrays (FGPAs) or field programmable analog arrays (FPAAs)) is an ideal technology to fulfill these requirements, since devices are fast (up to 1 GHz clock speeds), can be reprogrammed in milliseconds, and offer vast numbers of individual circuit elements that are inherently parallel in nature and may be configured arbitrarily.

Programmable logic promises to deliver computational speeds approaching that of custom silicon solutions while providing a flexible substrate for numerical simulation. Delivering on this promise, however, requires that field-programmable neuronal element circuit designs exploit the inherent parallelism of both the problem domain and target technology. Without this finely grained parallelism, this approach cannot compete in terms of processing performance with single-core microprocessors, in the form of pipelined and reduced instruction set compute architectures optimized for serial computation. Hence, to achieve the necessary performance in programmable logic requires deploying robust, extremely low-complexity neuron element designs that are inherently self-contained (i.e., no shared resources) and operate independently and in parallel. Furthermore, to achieve efficient operation, one should consider only single clock cycle iteration solutions (one clock cycle equals one numerical iteration) without sacrificing numerical performance. Furthermore, to achieve efficient operation, one should consider only single clock cycle iteration solutions without sacrificing numerical performance. Multicycle architectures of neuronal models have been previously discussed (Graas, Brown, & Lee, 2004). Single-cycle architectures have independently been investigated by Weinstein and Lee (2006).

FPGAs are digital integrated circuits (IC) that internally contain configurable blocks of logic, as well as programmable interblock connections

(Xilinx, 2002). Such devices can be configured in an arbitrary fashion to perform a variety of signal and data processing tasks. As a first step in the implementation, designs must satisfy specific criteria to guarantee a functional circuit, which is free from logical errors. Generally designs are specified in a hardware description language (HDL), such as VHDL or Verilog, although schematic-level entry is also possible. The HDL program must then be synthesized, which implies a process of minimization and optimization, which ends in the conversion of the sequential coded description into a collection of parallel registers (memory storage) and Boolean relationships. The gate-level functions obtained from this synthesis process are then mapped onto the physical layer of the device, which means assigning the logical design to available chip resources, depending on the chosen target device. From this process, a map file is created, which defines the placing of the logic onto the physical device, as well as the routing of the signals between logic elements (so-called place and route). Finally a "bitstream" is created, which is a file containing the information to internally configure the FPGA device (see Xilinx, 2003, and Maxfield, 2004). Design tools to aid the FPGA synthesis process are in a rapid state of development.

Here we present a set of reduced complexity programmable logic designs for exponential decay and alpha and beta function synapse with spike-time dependent plasticity (STDP) learning, as well as integrate-and-fire soma complete with axonal delays. Together these designs form a neuronal modeling kit, simulating all the major components commonly used in neuronal modeling of large-scale networks. The designs are of sufficiently low complexity to be realizable in massive numbers on a single FPGA device through multiplexing, yet feature single clock cycle numerical iteration.

We begin by making mathematical comparisons between forward-Euler (FE) and exact integration (EI) numerical iteration schemes in the case of linear ordinary differential equation (ODE) solvers. We show that this comparison leads to a surprisingly simple solution for implementing neuronal elements with exact numerical properties. By simulating and evaluating these designs, we demonstrate FPGAs as a viable technology for large-scale spiking neuron model implementation. The designs are finally tested against numerical and analytical results, verifying exact integration behavior.

2 Numerical Considerations

Neuronal dynamics are commonly modeled in digital hardware using numerical methods for solving ODEs. The simplest scheme for numerical simulation of dynamical system behavior is the FE approach. For an initial value problem (IVP) of the form

$$\dot{y}(t) = f(y, t), \qquad y(0) = y_0, \tag{2.1}$$

a first-order (linear) approximation to its solution is given by the FE approximation (Lambert, 1973)

$$\tilde{y}_{k+1} = \tilde{y}_k + \Delta f(y_k, t_k), \tag{2.2}$$

which is an iteration scheme that begins from an initial value (y_0), where k is an integer defining the iteration number ($k = 0, 1, \cdots$), and Δ is the fixed step size, in part determining the accuracy of the approximation. In general, this numerical integration scheme has a truncation error of order $O(\Delta^2)$. More sophisticated iteration schemes can be used that reduce the single step error (such as Runge-Kutta), but these require greater computational effort and hence more complex implementations.

In this context, considering a homogeneous first-order ODE of the form

$$\dot{y}(t) = ay(t), \qquad y(0) = y_0, \tag{2.3}$$

we can approximate its solution by applying the FE method as follows:

$$\tilde{y}_{k+1} = \tilde{y}_k(1 + \Delta a), \qquad \tilde{y}(0) = y_0. \tag{2.4}$$

As an alternative, there is an exact way to perform digital simulation of general linear-time-invariant systems, which is described by Rotter and Diesmann (1999). In order to obtain an EI solution of equation 2.3, we make use of this method, yielding the following expression:

$$y_{k+1} = e^{\Delta a} y_k, \qquad y(0) = y_0. \tag{2.5}$$

In general, both methods will yield different values. However, in order to find a relation between the FE and the EI method for the linear ODE case, we make use of the parameter a, which determines the time constant of the system as the relational parameter. Let us now define $\tilde{a}$ and a as the factors determining the time constants for the FE and EI solutions, respectively. If both iteration schemes begin from the same initial value y_0, then clearly they will give identical results if the following condition is met:

$$1 + \Delta\tilde{a} = e^{\Delta a}. \tag{2.6}$$

To satisfy this condition, we must first solve for $\tilde{a}$ and then substitute this value into equation 2.4 instead of a. Hence, for first-order linear-time-invariant dynamical systems such as we consider here, the FE solution is simply a timescaled version of the exact solution, which can be corrected by adopting the $\tilde{a}$ parameter in equation 2.4. In the following circuit implementations, we will exploit this fact to derive extremely low-complexity circuit designs capable of implementing an EI of common neuronal modeling components.

3 Neuron Model

We first consider the two main neuronal elements: a synapse model that reproduces postsynaptic dendritic current dynamics resulting from a presynaptic action potential and a soma model that integrates these currents over time to generate a membrane potential. We describe the implementation of both models in turn, providing the numerical details on which they are based, as well as their optimized programmable logic circuit implementations.

3.1 Synapse Model

3.1.1 Exponential Decay Synapse Model. One of the most common methods to model dendritic currents generated by a synapse in response to a spike train is through an exponential decay over multiple spike inputs occurring at times $(t_1, t_2, \ldots, t_j, \ldots t_l)$ to give the dendritic current

$$I(t) = w \sum_{j=1}^{l} H(t - t_j) e^{-\frac{t-t_j}{\tau_e}}, \tag{3.1}$$

where $H(\bullet)$ is the Heaviside function, w is the synaptic efficacy (weight) that determines the current increment in response to the arrival of each presynaptic action potential, and τ_e is the time constant of the exponential decay resulting from the arrival of each action potential.

Equation 3.1 is the solution to the first-order ODE of the form

$$\dot{I}(t) = w \sum_{j=1}^{l} \delta(t - t_j) - \frac{1}{\tau_e} I(t), \tag{3.2}$$

which can be approximated using the FE scheme for $a = -\frac{1}{\tau_e}$ as

$$I_{k+1} = w\delta_{k+1,j} + I_k \left(1 - \frac{\Delta}{\tau_e}\right). \tag{3.3}$$

This numerical scheme may be implemented directly in programmable logic by keeping a running accumulation of I over time by adding the current value to itself and subtracting a fraction of I_k at each time step. When a time step occurs in which a spike is received, the constant factor w must also be added to the accumulated value. This scheme makes for a very simple architecture with the exception of the multiplication involved in the fractioning process, which in its most general form is expensive to implement in programmable logic.

The resulting register transfer level (RTL) description of the synapse that can be directly implemented in programmable logic is shown in Figure 1A, as derived from the FE iteration scheme given in equation 3.3. The implementation assumes that within a single clock cycle (identical to the step time Δ), only a single action potential may be received at the input. This is reasonable if we assume that every synapse is connected to a single presynaptic cell, which typically has an absolute refractory period far exceeding a single time step. $n-$bit integer arithmetic may be used without loss of precision as long as we scale the circuit weight input w_B according to $w_B = k_I w$, by choosing k_I such that the bit count output of the circuit, B, extends across the integer number line $\{0, 1, 2, \ldots, 2^{n-1}\}$. In this case, the dendritic current can be recovered from the circuit output via $I(t) = B/k_I$. The real-time response of this circuit to a single action potential shown in Figure 1B is an exponential decay, with starting value equal to the weight of the synapse as shown in Figure 1C. The simulation proceeds at least three orders of magnitude faster than biological time, requiring a dimensionless speed up to factor k_t to translate between biological time constants and those achieved by the FPGA. The speed-up factors k_t are calculated based on a biological time constant for the synapse of 5 ms for Figures 1 and 2 and a biological time constant for the soma of 20 ms for Figures 3, 4, and 7.

Using the same arguments as previously, it is clear that the FE scheme describing the synaptic dynamics is equivalent to the EI solution as long as the following corrected time constant is substituted for τ_e in equation 3.3:

$$\tilde{\tau}_e = \frac{\Delta}{1 - e^{-\frac{\Delta}{\tau_e}}}. \tag{3.4}$$

The behavior of this circuit to the regular spike train shown in Figure 1D is shown in Figure 1E. The limiting value of peak synaptic current is compared to the asymptotic value (dotted line), which in the case of a regular spiking input at fixed frequency, f_{sp}, can be shown to be

$$I_{peak} = \frac{w}{1 - e^{-\frac{1}{f_{sp}\tilde{\tau}_e}}}. \tag{3.5}$$

The asymptotic peak response of the circuit is found to be within one least significant bit (LSB) of the theoretical value, I_{peak}.

3.1.2 Alpha and Beta Function Synapse Models. Alpha and beta functions are also popular physical models for synaptic dynamics (Jack, Noble, & Tsien, 1975; Gerstner & Kistler, 2002; Tuckwell, 1988). The dendritic current

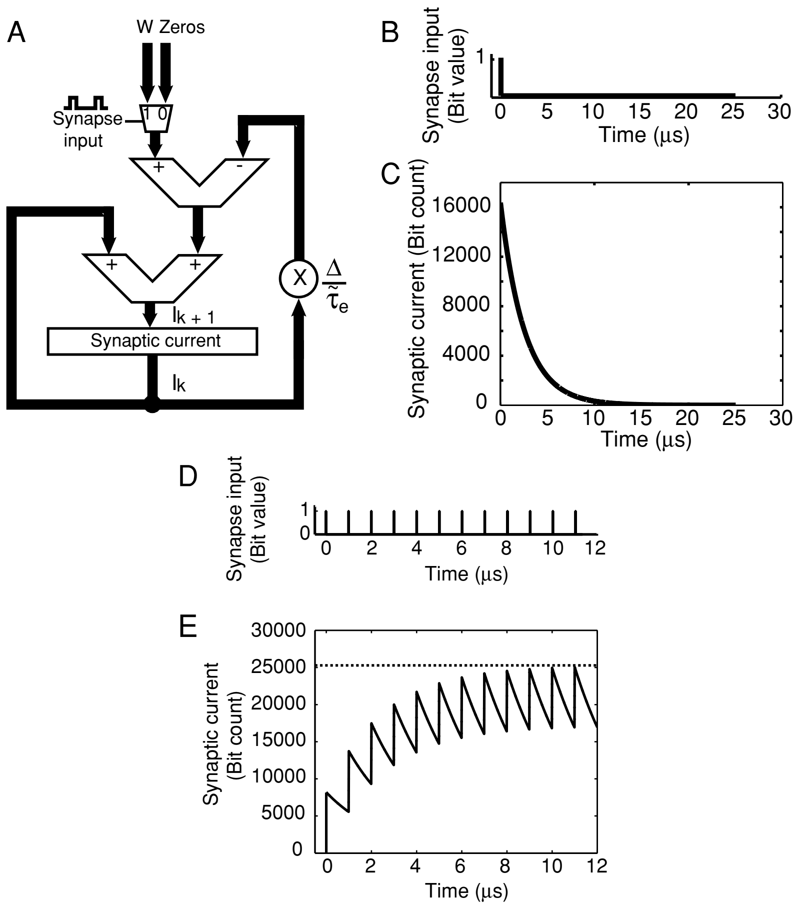


Figure 1: Exponential decay synapse implementation. (A) RTL description of the synapse architecture. Spiking synaptic input is represented by 0-1 logic levels, which controls the addition of weight value at each clock cycle. Thick, solid lines represent *m*-bit digital buses (representing unsigned integers), and the thin, solid line represents an individual control line. (B) A single spike event at time $t = 0$ occurs at the input of the synapse. (C) An exponential decay is generated in the circuit as response to the input. For a synapse with a current increment of 50 pA, the bit count output, B, can be converted to synaptic current using $I(t) = B/k_I$, where $k_I = 3.28 \times 10^{14}$ counts A^{-1}. (D) Spiking input at a regular firing frequency, $f_{sp} = 1$ MHz, used to test the synapse implementation. (E) Real-time synapse response to this regular spiking input for FPGA synapse implementation. The theoretical asymptotic value of peak synaptic current is shown by the upper dotted line. The clock frequency was set to $f_{clk} = 100$ MHz, giving a step time, $\Delta = 10$ ns, and a speed-up factor $k_t = 1953$ resulting in time constants, $\tilde{\tau}_e = 2.56\,\mu$s and weight $w = 16{,}384$ for C and $w = 8192$ for E.

generated by an alpha function synapse responding to multiple spike inputs occurring at times $(t_1, t_2, \ldots, t_j, \ldots t_l)$ is described by

$$I(t) = \frac{we}{\tau_\alpha} \sum_{j=1}^{l} H(t - t_j)(t - t_j)e^{-\frac{t-t_j}{\tau_\alpha}}, \tag{3.6}$$

where w is again the efficacy of the synapse and τ_α is the characteristic time constant for the synapse.

In a beta function model, current in a postsynaptic dendrite generated in response to multiple presynaptic spikes occurring at times $(t_1, t_2, \ldots, t_j, \ldots t_l)$ is described by

$$I(t) = w\beta \sum_{j=1}^{l} H(t - t_j)\left(e^{-\frac{t-t_j}{\tau_{\beta_1}}} - e^{-\frac{t-t_j}{\tau_{\beta_2}}}\right), \tag{3.7}$$

where τ_{β_1} and τ_{β_2} now determine the time constant of the synapse and β is a parameter adjusted to produce a (dimensionless) beta function with unit amplitude.

Rotter and Diesmann (1999) show how a matrix exponential can be used to describe the exact solution to greater than first-order linear ODEs (see appendix A), such as alpha and beta function dynamics. Due to their second-order dynamics, the matrix exponential for either the alpha (see equation A.2) or beta function (see equation A.3), consists of two exponential decay terms in the diagonal and a third off-diagonal term. This fact provides us with a simple method for implementing either function by deploying in cascade two exponential decay elements detailed in section 3.1.1.

There is, however, a nonzero off-diagonal term in both matrices, which act as a scaling function that must ordinarily be applied when coupling both exponential decay elements. We see that this matrix element is in fact a constant factor and so can be implemented by means of a multiplier in our circuit. Yet to keep the design as simple as possible, this constant factor may be directly applied to the weight of the synapse instead. Adjusting the synaptic weight in this way does not change the dynamics of the function but eliminates the necessity of multipliers between exponential decay elements, resulting in a simpler circuit. Thus, the resulting adjusted weight for the alpha function is

$$w_{adj} = \Delta e^{-\frac{\Delta}{\tau_\alpha}} w, \tag{3.8}$$

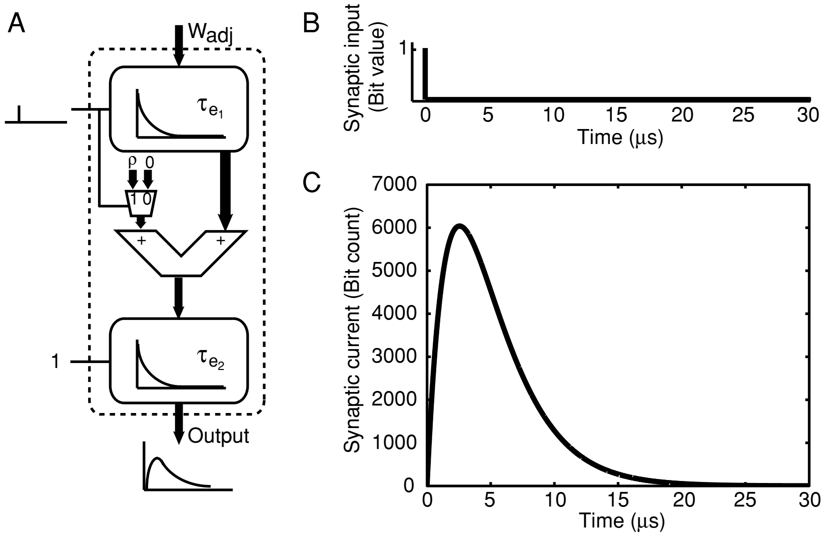


Figure 2: Alpha and beta function synapse implementation. (A) Block diagram of the implementation of an alpha and beta function generator implemented by connecting in cascade two exponential decay synapses. When $\tau_{e_1} = \tau_{e_2}$, the function generated is an alpha function; otherwise, it is a beta function. Each time an action potential arrives at the synapse, both the adjusted weight and the value of ρ are added to the circuit. The parameter ρ is used only when the alpha or beta function is fitted to a soma circuit to produce a combined neuron model; in such cases, the value of ρ is determined using equations 3.12 and 3.13. When simply performing isolated alpha function synapses, ρ has no effect in the circuit; therefore, set $\rho = 0$, and no value of ρ will be added during spikes, whereas w_{adj} is determined by equations 3.8 or 3.9 depending on the type of synapse. (B) A single spike event at time $t = 0$ occurs at the input of the synapse. (C) An alpha function is generated as real time response to the input. A 32-bit representation was used in the circuit with parameters clock frequency $f_{clk} = 100$ MHz, step time, $\Delta = 10$ ns, $k_t = 1953$, $\tilde{\tau}_\alpha = 2.56\,\mu$s, $\rho = 0$, and adjusted weight $w_{adj} = 64$.

whereas for the beta function, it is

$$w_{adj} = \frac{w\tau_{\beta_1}\tau_{\beta_2}}{\tau_{\beta_1} - \tau_{\beta_2}} \left(e^{-\frac{\Delta}{\tau_{\beta_1}}} - e^{-\frac{\Delta}{\tau_{\beta_2}}} \right). \tag{3.9}$$

The schematic for this class of synapse is shown in Figure 2A. In this case, action potentials, acting as input, are received at the first exponential decay element, while the output from the second element follows alpha and beta function dynamics. The parameter ρ, shown, will be proved to be an important parameter when the circuit is used in combination with a soma circuit to reproduce the membrane potential of a neuron, but is not relevant for isolated synapses and can then be set to zero. Figure 2C shows an alpha function generated by this circuit as a response to a single input spike applied at time $t = 0$ (see Figure 2B). Note that we have again adjusted each time constant according to equation 3.4 to guarantee an EI solution.

3.2 Soma Model. The popular integrate-and-fire model that receives synaptic input of the form $I(t)$ has a membrane potential, $V(t)$, the dynamics of which are described in between generated action potentials by

$$\dot{V}(t) = -\frac{V(t)}{\tau_m} + \frac{I(t)}{C_m}, \tag{3.10}$$

where the membrane capacitance, C_m, is a constant and τ_m is the characteristic time constant for the cell. Once $V(t)$ reaches a fixed threshold potential, V_{th}, say, at time t', $V(t')^- = V_{th}$, the soma then resets to the afterhyperpolarization potential, $V(t')^+ = V_{ahp}$, and a spike is generated by the soma and integration of the current input continues. Again, using the FE approach, the solution of equation 3.10 may be approximated as

$$V_{k+1} = V_k\left(1 - \frac{\Delta}{\tau_m}\right) + \frac{\Delta}{C_m} I_k, \tag{3.11}$$

resulting in a similar implementation to that of the exponential decay synapse, except that the dendritic input is added at each time step. In the next section, we show how, again, EI can be implemented within the FE scheme if the dynamics of I are appropriately taken into account.

The RTL description for the soma implementation, complete with spike generating mechanism, is shown in Figure 3A. A comparator is used to detect threshold crossings, which gates a single flip-flop producing a $0 \rightarrow 1 \rightarrow 0$ transition on the axon output lasting one clock cycle. If required, a fixed input bias may be added at each time step in order to determine the resting potential of the cell. As before, the bit count held in the soma potential register, B, is a linearly scaled representation of the soma potential, such that $V(t) = B/k_V$. The soma model responding to two current pulses of different magnitudes (shown in Figure 3B) is shown in Figure 3C.

3.3 Combined Neuron Model. We have now developed programmable logic circuits that implement EI solutions for all the main neuronal modeling components described by equations 3.1, 3.6, 3.7, and 3.10. These synapse and soma circuits must next be combined in such a way as to obtain an EI solution for the complete neuron model. This process is not immediate, since, unfortunately, combining individual elements with EI performance

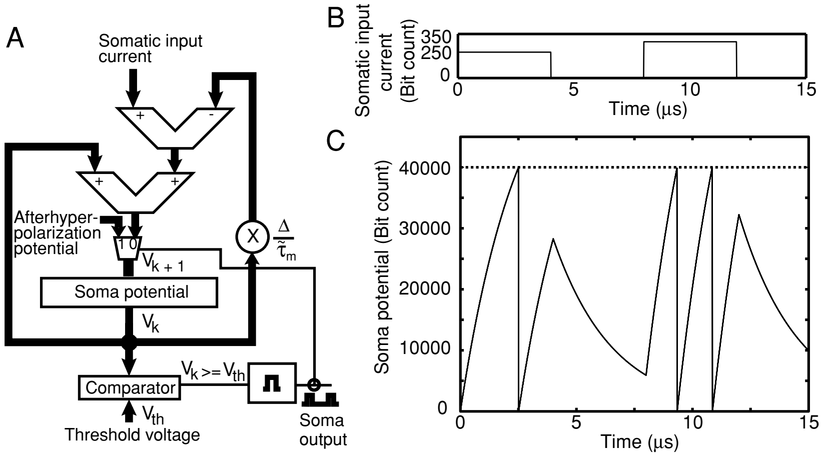


Figure 3: Integrate-and-fire soma implementation. (A) RTL description of soma architecture. Somatic input current, $I(t)$, afterhyperpolarization (reset) potential, V_{ahp}, and the threshold potential, V_{th}, are each represented by a p-bit signed integer (thick, solid lines). A fire event (spike) is represented by a $0 \rightarrow 1$ transition on the soma output line lasting a single clock cycle. (B) Somatic current pulses (see Rotter & Diesmann, 1999, sec. 3.2.3, for the appropriate EI coefficient) of $I = 250$ and 350 bit count, respectively, lasting and separated by 4 μs were used to test the soma implementation. (C) Real-time soma response to the current input for the FPGA implementation with a speed-up factor $k_t = 7813$. For a soma with a threshold of 20 mV above resting potential, the soma potential bit count, B, can be converted to soma potential $V(t) = B/k_V$, where $k_V = 2 \times 10^6$ counts V^{-1}. A 32-bit representation was used with parameters clock frequency $f_{clk} = 100$ MHz, step time, $\Delta = 10$ ns, resulting in $\tilde{\tau}_m = 2.56\,\mu$s, and $V_{th} = 40{,}000$ which is indicated by the horizontal dotted line.

does not guarantee EI system performance when coupled together. Thus, some important considerations must be taken into account before connecting these components.

First, we use the matrix exponential of the combined system given by Diesmann, Gewaltig, Rotter, and Aertsen (2001), as in equation B.1. In this case the matrix exponential describes the state-space representation of a combined neuron comprising synapse and soma. We see from this representation that a simple cascade connection of these components will not suffice, since there exist nonvanishing off-diagonal elements that act as constant factors between stages, again requiring the use of multipliers.

In order to optimize the combined neuron model for the case of alpha function synapse and soma, we apply a linear transformation (see appendix B), which permits the direct cascade of synapse and soma elements without the need of coupling factors. An important consequence of this transformation, however, is the necessity to again adjust the synaptic weight and also apply a constant addition factor (ρ) in between stages according to

$$w_{adj} = \frac{\tau_\alpha \tau_m \Delta e^{-\frac{\Delta}{\tau_\alpha}} (e^{-\frac{\Delta}{\tau_\alpha}} - e^{-\frac{\Delta}{\tau_m}})}{C(\tau_\alpha - \tau_m)} w, \tag{3.12}$$

and

$$\rho = \frac{\tau_\alpha \tau_m ((\tau_\alpha - \tau_m)\Delta e^{-\frac{\Delta}{\tau_\alpha}} - \tau_\alpha \tau_m (e^{-\frac{\Delta}{\tau_\alpha}} - e^{-\frac{\Delta}{\tau_m}}))}{C(\tau_\alpha - \tau_m)^2} w. \tag{3.13}$$

These parameters may again be calculated off-line to avoid the necessity of multipliers in the circuit, yielding the far simplified combined neuron circuit shown in Figure 4A.

Using similar arguments for the exponential decay synapse and soma case (see appendix C), we can again find an adjusted weight that results in combined EI performance:

$$w_{adj} = \frac{w \tau_e \tau_m}{C(\tau_e - \tau_m)} (e^{-\frac{\Delta}{\tau_e}} - e^{-\frac{\Delta}{\tau_m}}). \tag{3.14}$$

Now that we have determined the conditions required to perform an exact integration for each case, synapse and soma may now be combined to form a self-contained, single clock cycle operation, spiking neuron implementation. Figure 4A shows a generalized multisynapse scheme for a single neuron. The neuron comprises r synapses, which receive spikes from different presynaptic cells, generating dendritic currents that are summed in a single clock cycle and integrated to produce the membrane potential of the soma. In general, a greater number of bits will be need to be deployed at the soma as compared to the synapse, since integration of the input occurs at each time step (not just after spike arrival) and multiple synapse inputs may be summed. In order to minimize the total number of bits required in the soma, we can limit the maximum and minimum weights in the synapses to avoid overloading the soma, which, as we will see, is also a desirable characteristic for the learning algorithm we consider later.

In order to demonstrate the EI performance of the combined designs, we tested an exponential decay type synapse and soma. Input spikes were applied to the combined neuron model as seen in Figure 4B which gave rise to the resulting membrane potential shown in Figure 4C. We see that in this case, the membrane potential crosses the threshold (V_{th}) three times, generating the same number of spikes (see Figure 4D).

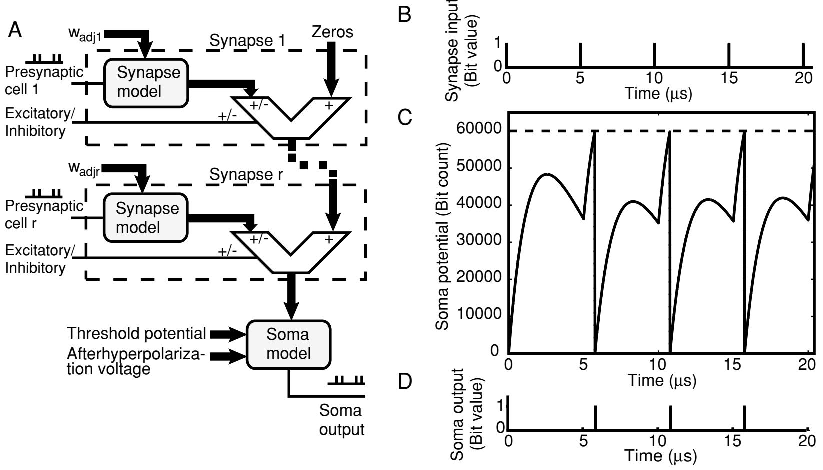


Figure 4: Combined neuron implementation. (A) RTL description of the combined neuron architecture. Spiking input is represented by 0-1 logic levels received at r synapses. Weights, w_i, for synapses ($i = 1, \dots, r$), the afterhyperpolarization (reset) potential, V_{ahp}, and the threshold potential, V_{th}, are represented by m-bit unsigned integers (thick solid lines). (B) Spiking input at a regular firing frequency, $f_{sp} = 200$ KHz, used to test the combined neuron implementation (only one synapse input ($i = r = 1$) was considered). (C) Real-time soma response to the synapse input for the FPGA combined neuron implementation. (D) Spikes generated by the combined neuron model in response to the spiking input defined in B. A 32-bit representation was used for both circuits with clock frequency $f_{clk} = 100$ MHz, step time, $\Delta = 10$ ns, and $k_t = 7813$. For the synapse, time constant of $\tilde{\tau}_e = 2.56\,\mu$s and adjusted weight $w_{adj} \approx 0.61681$; for the soma, $\tilde{\tau}_m = 2.56\,\mu$s, $C = 12$ pF, and $V_{th} = 60{,}000$ which is again indicated by the horizontal dashed line.

3.4 Axonal Delays. Axonal delays play an important role in the dynamics of spiking neuron network models (Crook, Ermentrout, Vanier, & Bower, 1997) and are simple to implement in our design. In general, axonal delays will not be constant across all cells of the network. Instead, each axon should have associated with it a particular delay (see Figure 5A).

When a spike occurs at a soma output, this should not be delivered instantly to the target synapse. Rather, the action potential must be presented in some $n\Delta$ time steps after the spike occurred. In our design, axonal delays are implemented using a ring buffer (see Figure 5B). The spike history (up to time $n\Delta$) is stored in the ring buffer from each soma, and at each time step, the nth element is transmitted to the target synapse, while the current state of the soma is written to the buffer. Ring buffers have the advantage that spike history data are automatically overwritten as they become redundant.

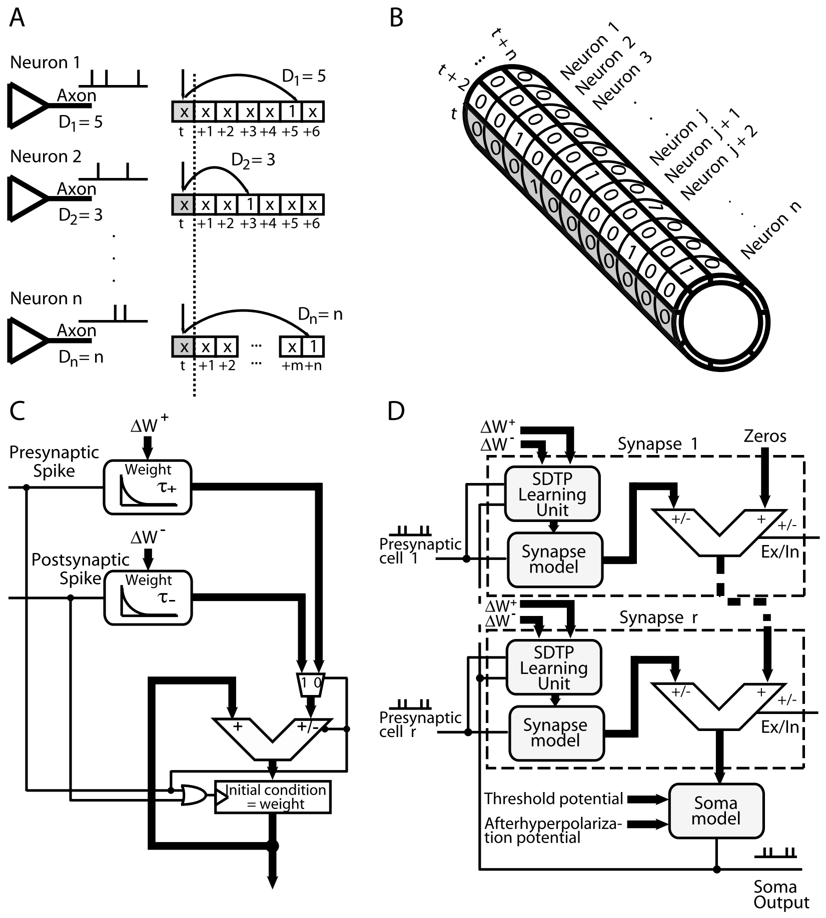


Figure 5: Axonal delay and STDP learning implementation. (A) Each axon is programmed to have a specific delay (defined by D_n for the n-axon), which is implemented using a ring buffer. (B) Action potentials are read from the ring buffer at time t; the synapses from other neurons receive the delayed spikes according with the programmed delay for each neuron. (C) RTL design of the STDP learning unit. Superpositions of exponential decays with height equal to the amount of weight modification (ΔW^+ to increment or ΔW^- to decrement) are generated; the order of presynaptic and postsynaptic fire times defines both the amount and the sign of the weight modification. (D) Neuron model with STDP learning. The STDP unit receives presynaptic spikes arriving at a specific synapse as well as the action potentials generated by its respective soma. The adjusted weight is fed to the synapse every time a presynaptic or postsynaptic spike takes place.

3.5 Learning by STDP. There are many Hebbian (correlation)-based plasticity mechanisms that can be used for learning purposes in spiking neuron models (Abbott & Nelson, 2000; Gütig, Aharonov, Rotter, & Sompolinsky, 2003). One of the most common of these is spike-timing dependent plasticity (STDP) proposed by Song, Miller, and Abbott (2000). This plasticity mechanism relies on the difference in presynaptic and postsynaptic spike times to adjust the strength of excitatory synapses.

As an example of how plasticity mechanisms may be conveniently combined with our neuronal element designs for on-chip learning in real time, we have implemented STDP with our synapses. Figure 5C shows the RTL design of the STDP unit, which contains two exponential decay elements reused from our synapse designs. The exponential decay unit situated in the upper section of the diagram receives presynaptic spiking input in the same way as the synapse itself. Each time a spike is received at this exponential decay unit, a value equal to the maximum amount of change of the synapse strength (ΔW^{+}) is added to its output, while during latency periods, this value decays exponentially according to equation 3.1. Once a spike is generated in the postsynaptic neuron, the strength of the synapse is increased by the current value output by this exponential decay unit. The second exponential decay unit behaves in the opposite way. That is, the second element has as input postsynaptic action potentials that add to the current value an amount equal to the maximum possible change of synapse weakening (ΔW^{-}). When a presynaptic action potential is received, the current value of this exponential decay unit is subtracted from the weight of the synapse. In this way, synapses compete to control the postsynaptic spike timing of the neuron. Figure 5D shows the implementation of the STDP unit in the generalized neuron model.

The weight of the synapse is limited to the range $[0, W_{\max}]$. Stable synaptic modification requires that $\Delta W^{+} < \Delta W^{-} < W_{\max}$, and experimental results suggest that $\Delta W^{-} \Delta W^{+} \approx 1.1$ (Song et al., 2000). The maximum weight $W_{\max}$ should be chosen so as not to overload the soma.

4 Results

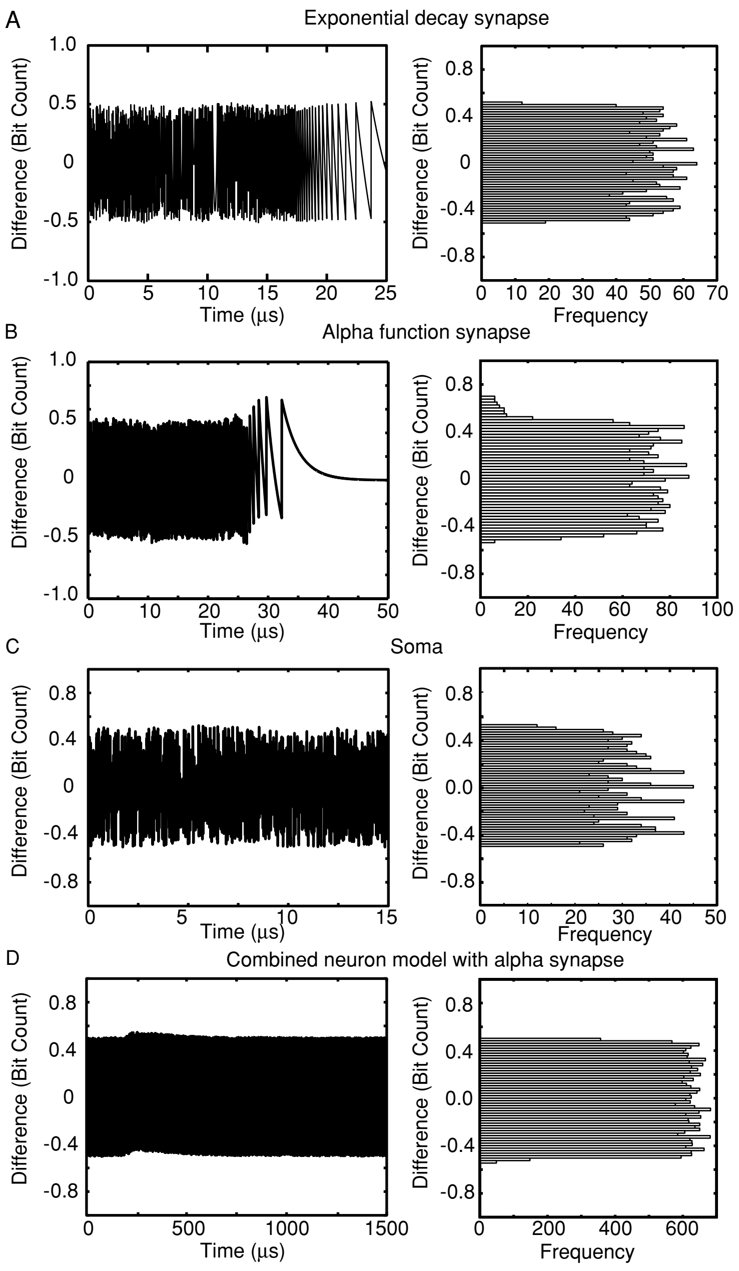
RTL designs for each neuronal component were implemented in VHDL and executed on a PCI-based FPGA development board (model ADM-XRC-II Pro, manufactured by Alpha Data Systems, UK), which hosts a Xilinx Virtex II Pro device (product code XC2VP100-5). Numerical results were tested against the equivalent exact numerical solution as calculated on a standard PC with Intel Pentium IV running at a clock speed of 3.06 GHz with 1 GB of external memory, programmed in C++. All responses shown in Figures 1B and 1C, 2B and 2C, and 3B and 3C were compared against their corresponding EI solution and the differences plotted in Figure 6 (left). In all cases, the errors show a random behavior about zero, which suggests that the difference is due to round-off as a result of the finite-length number representation. This conclusion is confirmed by the histograms of the error (see Figure 6, right), which show that in all cases, the distributions closely resemble the shape of the uniform probability density function, which is the expected behavior generated by a round-off process (Barnes, Tran, & Leung, 1985).

With increasing time, the response error becomes more regular as the output of the circuits approaches zero, since the time derivative becomes very small (itself an exponential decay). In the case of the alpha function synapse (see Figure 6B), although we see a random uniform distribution about zero of ± 0.5 bits for each of the two stages, when combined, the two errors may accumulate in the positive direction leading to an error greater than ± 0.5 bits for a single neuron. However the total error will never exceed ± 1 bit and is not systematic, since the asymptotic behavior is toward zero.

An additional experiment was carried out using a combined neuron model comprising an alpha function synapse and a soma, which was excited by a single spike at time ($t = 0$). The membrane potential was again compared against the numerical solution given by the EI method. The error between the circuit implementation and the EI solution shows the same behavior as in the preceding case: a uniform distribution due to the round-off process (see Figure 6D).

To further test the numerical performance and robustness of our designs, a combined neuron simultaneously integrating signals from five exponential decay synapses driven by Poisson processes (see Figure 7A) was implemented. The total resulting synaptic current is shown in Figure 7B. The difference between the membrane potential obtained by the EI scheme and the value generated by the circuit (see Figure 7C), again shows a random behavior about zero, ruling out any systematic error in the circuit behavior

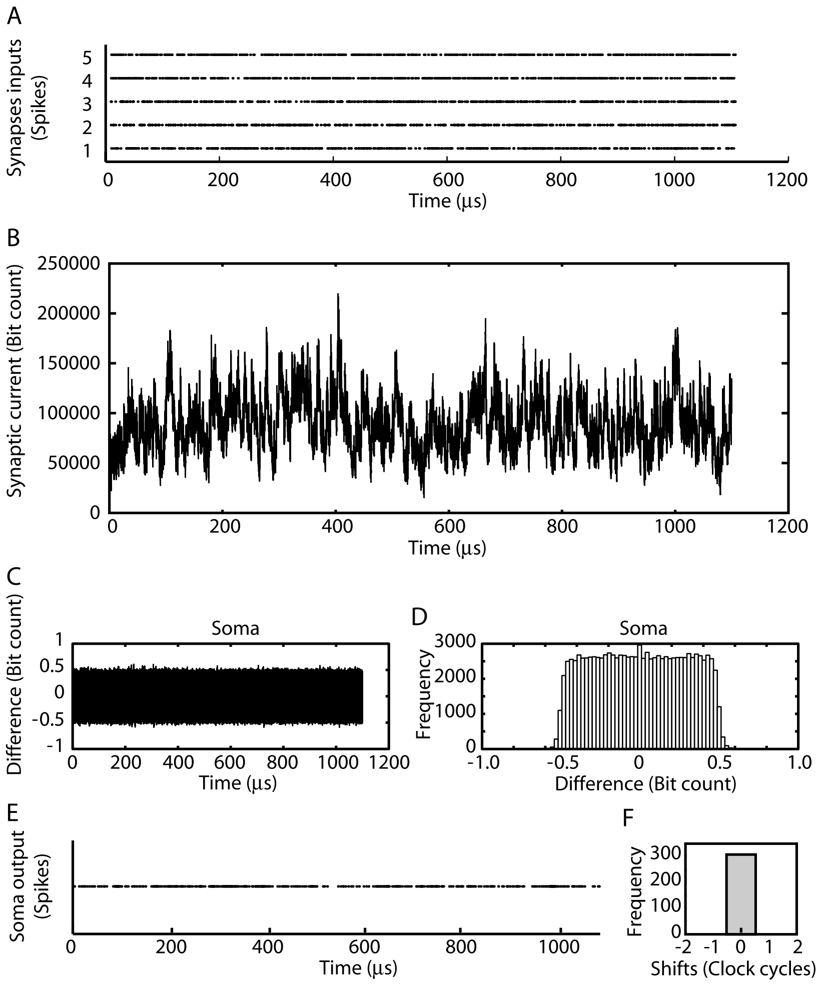
Figure 6: Differences between the numerical solution and the circuit responses. Difference over time (left) and histograms of their corresponding error distribution (right) for (A) exponential decay synapse response, (B) the alpha function synapse response, (C) the soma response, and (D) combined neuron model made up of alpha function synapse and integrate-and-fire model. The error corresponds to those responses shown in Figures 1 to 3 except for the combined neuron model, whose response comes from a single spike at ($t = 0$). The parameters of the three first circuits are the same as the ones specified for each response, while for the combined model, the parameters are $\tilde{\tau}_\alpha = 25.6\ \mu$s, $\tilde{\tau}_m = 4.096$ ms, $C = 250$ pF, adjusted weight $w_{adj} \approx 0.5089$, and $\rho \approx 0.2553$. All errors show the same characteristics: a random behavior about zero and a uniformly distributed probability density function. The clock frequency was set to $f_{clk} = 100$ MHz, giving a time step of $\Delta = 10$ ns. Numerical solutions were carried out in C++ using long double precision, using an 80 bits representation with 64 bits for the mantissa and 14 for the exponent.


A
Exponential decay synapse
Difference (Bit Count)
Time (μs)
Frequency
B
Alpha function synapse
C
Soma
D
Combined neuron model with alpha synapse

(see Figure 7D). In this case, 308 action potentials were generated by the circuits (see Figure 7E), exactly the number of spikes obtained through the EI scheme. Critically, there were no differences in firing times for both the circuit and the numerical solution, all spikes were time coincident within a single clock cycle (see Figure 7F). This confirms that for the purposes of simulation of integrate-and-fire neuron and exponential decay–based synapse dynamics, our circuit produces EI performance under realistic simulation conditions, limited only by the restricted bit length of the representation, which is the case for any digital neuronal model implementation.

In order to further test the comparative numerical performance of our FPGA implementation with PC technology, we deployed a medium-scale neuronal model described previously (Pearce et al., 2005) and tested it against its software equivalent. Briefly, this model was composed of 100 integrate-and-fire somas and 675 exponential decay synapses describing two classes of neuron in the mammalian olfactory bulb (see Figure 8A). Mitral/tufted (M/T) neuron outputs in the vertebrate olfactory system are under tight regulation through lateral inhibition, mediated by dendrodendritic interaction. This interaction effects complex synchronous firing behavior across the bulb output that is stimulus specific, reminiscent of that observed in electrophysiological studies (Friedrich & Laurent, 2001; Friedrich, Habermann, & Laurent, 2004) and demonstrated in our model (see Figures 8B and 8C). Due to the similarity of synapse time constants in this model, it was possible to implement the design on a single Virtex II Pro device (Xilinx) running at a very conservative clock speed of 33 MHz.

Figure 7: Numerical error for the combined neuron. (A) Random spikes trains applied at the input of each synapse. (B) Total synaptic current generated by the circuit. (C) The error between the exact integration numerical implementation and the circuit response for the membrane potential again shows a random behavior about zero. (D) Histogram of the error distribution, which uniform distribution corroborates the round-off effects as the only cause of the error. (E) Over 300 spikes were generated by the soma. (F) Spike timing for both the exact integration numerical implementation and circuit response. All spikes from the circuit were exactly coincident with the spike times given by the numerical solution. An m-bit representation was used with parameters $\tilde{\tau}_{e_1} = 5.12\,\mu$s, $\tilde{\tau}_{e_2} = 2.56\,\mu$s, $\tilde{\tau}_{e_3} = 1.28\,\mu$s, $\tilde{\tau}_{e_4} = 0.64\,\mu$s, $\tilde{\tau}_{e_5} = 1.28\,\mu$s, adjusted weights $w_{adj_1} \approx 0.077026$, $w_{adj_2} \approx 0.077101$, $w_{adj_3} \approx 0.077253$, $w_{adj_4} \approx 0.077558$, and $w_{adj_5} \approx 0.077253$, whereas $\tilde{\tau}_m = 2.56\,\mu$s, for the soma, capacitance $C = 12$ pF, threshold potential, $V_{th} = 65{,}000$. The clock frequency was set to $f_{clk} = 100$ MHz, giving a step time $\Delta = 10$ ns and a speed-up factor $k_t = 7813$. Numerical solutions were carried out in C++ using long double precision using an 80 bits representation with 64 bits out of them for the mantissa and 14 for the exponent.

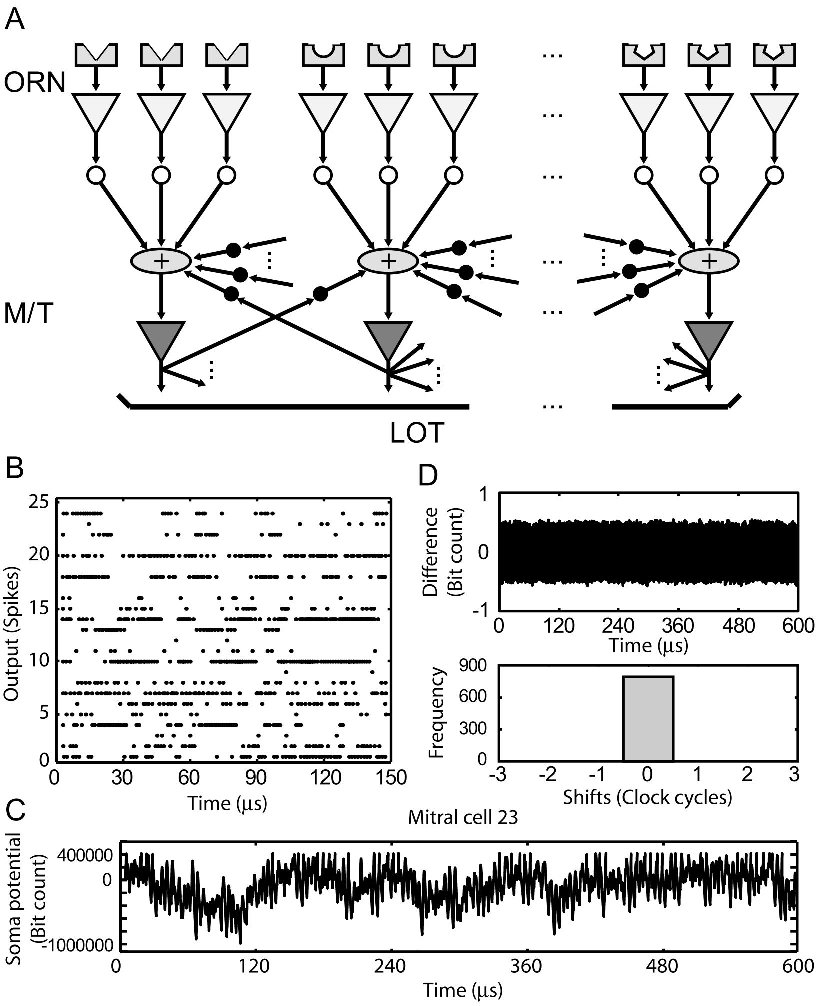

A
Synapses inputs (Spikes)
Time (μs)
B
Synaptic current (Bit count)
Time (μs)
C
Soma
Difference (Bit count)
Time (μs)
D
Soma
Frequency
Difference (Bit count)
E
Soma output (Spikes)
Time (μs)
F
Frequency
Shifts (Clock cycles)

The same network was coded in C++ using the same set of equations as implemented by the hardware, compiled using GNU gcc, and executed on an Intel Pentium IV 3.06 GHz with 1 GB RAM. As before, the error between the circuit and the software implementation is within a single bit and uniformly distributed (see Figure 8D). Spike timing analysis reveals that no incorrect translation occurs between the FPGA- and PC-based solutions.

5 Conclusion

Designs for leaky integrate-and-fire soma and exponential, alpha, and beta function synapses with STDP learning and axonal delays are presented in this article, which are suitable for implementation in programmable logic. Together these designs provide a neuronal modeling construction kit of the most popular elements that can be deployed in arbitrary configurations for

Figure 8: Simplified olfactory bulb neuronal model implementation comprising 100 somas and 675 exponential decay synapses. (A) Schematic of the olfactory bulb architecture. Twenty-five mitral/tufted (M/T) cells, represented by integrate-and-fire elements provide the main olfactory bulb output (corresponding to the lateral olfactory tract). These cells are reciprocally connected by exponential decay synapses, which mediate lateral inhibition in the model representing inhibitory coupling between M/T cells in vertebrates (open circles: excitatory synapse; closed circles: inhibitory synapse). Excitatory input to the model is provided by olfactory receptor neurons (ORNs) driven by population-coded receptor input shown by irregular polygons. Synaptic input from identical ORNs is summed to represent receptor input convergence at a single glomerulus (ellipse) (after Pearce et al., 2005). (B) Firing behavior of all 25 M/T cells in the network over time. (C) Membrane potential of a randomly selected M/T cell. (D) The error between the exact integration numerical implementation and the circuit response for the membrane potential and spike timing analysis. A 32-bit representation was used with parameters $\tilde{\tau}_{e_E} \approx 2.1$ ms for the excitatory synapses and $\tilde{\tau}_{e_I} \approx 17.1$ ms for the inhibitory synapses. The adjusted weights for the excitatory synapses were fixed to $w_{adj_E} = 256.8$, whereas the inhibitory weights w_{adj_I} were randomly chosen in a range between 1.5 and 32, inclusive. The time constants of both the ORN and mitral cells were set to $\tilde{\tau}_m \approx 10.2$ ms, with a capacitance $C = 250$ pF and threshold potential $V_{th} = 432{,}640$, where $k_V = 21.6 \times 10^6$ counts V^{-1}. For the ORNs, constant values randomly selected in a range between 4240 and 6400 were used to represent a constant concentration of an odor stimulus. The clock frequency was set to $f_{clk} = 33$ MHz, giving a step time $\Delta \approx 30.3$ ns and a speed-up factor $k_t = 3300$. Numerical solutions were carried out in C++ using long double precision using an 80 bits representation with 64 bits out of them for the mantissa and 14 for the exponent.


A
ORN
M/T
LOT
B
Output (Spikes)
Time (μs)
C
Mitral cell 23
Soma potential (Bit count)
Time (μs)
D
Difference (Bit count)
Time (μs)
Frequency
Shifts (Clock cycles)

hyper-real-time implementation. This programmable logic construction kit supports the building of large and complex architectures of spiking neuron networks by means of an efficient communication and multiplexing scheme of the neuronal elements. The circuits proposed here have the advantage that they work in parallel rather than depending on serial computation, being capable of simulating neuronal models in hyper real time. The source codes (VHDL) for these designs have been made freely available for download at http://www.neurolab.le.ac.uk/fpga/.

The implementation we present is restricted to the class of integrate-and-fire models where synapses are described by time-dependent currents. However, with conductance-based synapses, only the differential equation for the membrane potential is no longer linear time invariant. Future work will need to explore whether an implementation with exact integration (EI) for the conductances and an approximate method for the membrane equation (e.g., FE as described in section 2) can be effectively combined.

Our circuit designs have been demonstrated to perform exact integration in a single clock cycle and are also self-contained (no shared resources). The advantage of single clock cycle operation is that designs may operate faster than biological timescales (milliseconds); depending on their complexity and the extent of optimization, processing speeds may be able to approach the clock frequency of the FPGA. This hyper-real-time operation provides us with the opportunity of multiplexing the physical neuron architecture to achieve far greater neuron numbers. This is made possible since only digital spike events need to be communicated between neurons, thereby simplifying connectivity circuitry. Thus, programmable logic offers neuronal simulation speeds approaching those of fully custom silicon solutions, but not at the expense of flexibility, design times, or network capacity, since once the design process outlined in section 1 is complete, large-scale models may be downloaded and adapted in milliseconds.

Most important, by using fully parallel single clock cycle implementations, our neuronal modeling approach leverages the impressive advances in programmable logic, in terms of clock speeds and device capacities, specialized DSP components, cost, and ongoing development of advanced design tools. FPGA capabilities and operating speeds are currently under exponential growth. Over the past 10 years, capacity has increased more than 200-fold with every indication that Moore's law will continue for these devices some way into the future (Alfke & Hitesh, 2005). This should ensure the possibility for growth of neuronal modeling capability that is commensurate with advances in programmable logic.

We can estimate the total neuronal capacity of any given FPGA device by calculating the amount of resources each neuronal element would require. FPGA resources are measured in terms of slices, each of which contains the fundamental digital elements required to implement arbitrary combinational logic functions (Xilinx, 2002). Thus, the number of slices required per neuron element limits the total physical numbers implementable on a single device. Using a 16 bit representation for the synapse and a 32 bit representation for the soma requires a total of 70 slices for the (exponential decay) synapse circuit and 90 slices for the soma. Current FPGA devices exceed 50,000 slices. Therefore, it is possible to configure, for instance, over 500 physical synapses without STDP (or 250 with STDP) and 100 physical somas, running in parallel with single clock cycle iteration.

Such an arrangement would be ideal in the case of small and medium network designs for which we require hyper-real-time operation in order to understand or optimize its performance in different portions of its parameter space. Over 100,000 prototypes of the same network could be simulated in the time taken for one iteration of the biological network it represents. Hence, programmable logic provides a powerful technology for understanding and optimizing small and medium spiking neuronal models.

Synapses with a common target soma often have similar time constants in the brain. We can exploit this fact to make further gains in total synapse numbers by using only one physical synapse and convolving the incoming spikes of many virtual synapses. This is mathematically equivalent to implementing large numbers of separate synapses with identical time constants. In this way, a very large degree of synaptic convergence can be achieved, which is particularly relevant to cortical models.

In order to test the comparative speed performance of our FPGA implementation with a single PC, we ran parallel trials of the olfactory bulb model (see Figure 8) on both PC and FPGA. The identical network was coded in C using the same set of equations as implemented by the hardware, compiled using GNU gcc and executed on an Intel Pentium IV 3.06 GHz with 1 GB RAM. The PC performed 10,000 numerical iterations of the network in 0.76 s without disk access. Due to single clock cycle numerical iteration, the FPGA performed the same number of steps on-chip in 303.03 μs, giving a speed-up factor of about 2500. This provides a speed-up factor of three to four orders of magnitude, which may be further optimized either by improving the placement and routing of the logic of the design implementation or by employing a faster FPGA (e.g., Virtex IV devices offer faster internal speeds and shorter propagation delays; Xilinx, 2004). In further experiments, we were able to increase the clock frequency up to 50 MHz. In both cases, about 55% of the FPGA resources were utilized.

We may also choose to exploit this hyper-real-time operation to increase network sizes through the use of a multiplexing scheme. In this case, we create virtual exponential decay units by storing and replacing the current state at each numerical step and, optionally, the associated parameters such as weight, afterhyperpolarization potential, threshold potential, and time constant. This requires storage either on or off chip, which, depending on the access time, will determine the communication overhead associated with applying a multiplexing scheme.

FPGAs have on-chip RAM block storage with parallel access single clock cycle operation, which minimizes this overhead (Xilinx XC2VP125 device has 556 such blocks, which may be variously configured). In principle, for this Xilinx device, we can use 556, 1024 $\times$ 18-bit buffers to create 10^5 to 10^6 virtual exponential decay elements operating in biological time (18-bit precision). However, in practice, there are two issues that must be addressed when using FPGAs to build large-scale neuronal models that exploit such multiplexed architectures. Firstly, at each numerical step, we must store, update, and replace the state of each element. This imposes a time overhead, which is at least one additional clock cycle per numerical iteration step to move the data to and from on-chip memory. This reduces the total number of virtual elements that can be operated in biological time. Second, we must consider connectivity between neurons, which imposes constraints on the scale of architectures that can be deployed on a single device.

Cortical models represent a particular challenge due to highly convergent synaptic input, which is often thousands of synapses targeting each neuron. Since most cortical models adopt only one or two time constants for these convergent synapses, arriving input spikes may be convolved through only one virtual synapse for each time constant, leading to a massive reduction in resource use. Such scaling issues represent an important focus of future research for implementing FPGA neuronal models to achieve cortical model scales in real time.

Appendix A: Alpha and Beta Functions Matrices Exponentials

The general solution of the exact integration (EI) scheme (Rotter & Diesmann, 1999) for an nth order system of linear-time-invariant ODEs is described by

$$\mathbf{y}_{k+1} = e^{\mathbf{A}\triangle}\mathbf{y}_k, \qquad \mathbf{y}(0) = \mathbf{y}_0, \tag{A.1}$$

where $e^{\mathbf{A}\triangle}$ is the matrix exponential of the system. The diagonal elements of this matrix describe the step dynamics for each of the exponential decay stages to be used as part of the solution, which can be cascaded to simulate the nth order system. However, to maintain an exact solution, the remaining off-diagonal elements must be applied as coupling factors between the cascaded stages.

A.1 Matrix Exponential of the Alpha Function Synapse.

$$\mathbf{P}_\alpha = e^{\mathbf{A}\triangle} = \begin{bmatrix} e^{-\frac{\Delta}{\tau_\alpha}} & 0 \\ \Delta e^{-\frac{\Delta}{\tau_\alpha}} & e^{-\frac{\Delta}{\tau_\alpha}} \end{bmatrix}. \tag{A.2}$$

A.2 Matrix Exponential of the Beta Function Synapse.

$$\mathbf{P}_\beta = e^{\mathbf{A}\Delta} = \begin{bmatrix} e^{-\frac{\Delta}{\tau_{\beta_1}}} & 0 \\ \frac{\tau_{\beta_1}\tau_{\beta_2}(e^{-\frac{\Delta}{\tau_{\beta_1}}} - e^{-\frac{\Delta}{\tau_{\beta_2}}})}{\tau_{\beta_1}-\tau_{\beta_2}} & e^{-\frac{\Delta}{\tau_{\beta_2}}} \end{bmatrix}. \tag{A.3}$$

Appendix B: Matrix Exponential of the Combined Neuron Model Based on Alpha Function Synapse

B.1 Matrix Exponential of the Combined Neuron Model Based on Alpha Function Synapse.

$$\mathbf{P}_m = e^{\mathbf{A}\Delta} = \begin{bmatrix} e^{-\frac{\Delta}{\tau_\alpha}} & 0 & 0 \\ \Delta e^{-\frac{\Delta}{\tau_\alpha}} & e^{-\frac{\Delta}{\tau_\alpha}} & 0 \\ \frac{\tau_\alpha\tau_m((\tau_\alpha-\tau_m)\Delta e^{-\frac{\Delta}{\tau_\alpha}} - \tau_\alpha\tau_m(e^{-\frac{\Delta}{\tau_\alpha}} - e^{-\frac{\Delta}{\tau_m}}))}{C(\tau_\alpha-\tau_m)^2} & \frac{\tau_\alpha\tau_m(e^{-\frac{\Delta}{\tau_\alpha}} - e^{-\frac{\Delta}{\tau_m}})}{C(\tau_\alpha-\tau_m)} & e^{-\frac{\Delta}{\tau_m}} \end{bmatrix}. \tag{B.1}$$

B.2 Linear Transformation of the Matrix Exponential of the Combined Neuron Model Based on Alpha Function Synapse. Here we describe a linear transformation that may be applied to the matrix exponential in order to simplify the combined neuron circuit. The goal of this transformation is to reduce the hardware required to implement the combined neuron model by removing the requirement for multipliers.

First, for simplicity, let us rewrite the matrix exponential ($\mathbf{P}_m = e^{\mathbf{A}\Delta}$) in a more general form as

$$\mathbf{P}_m = \begin{bmatrix} \alpha & 0 & 0 \\ r & \beta & 0 \\ q & p & \gamma \end{bmatrix}, \tag{B.2}$$

where α, β, and γ define the three exponential decay circuits connected in cascade and p, q, and r are constant coupling factors, which call for the use of multipliers. We express the matrix exponential in the form

$$\begin{bmatrix} \alpha & 0 & 0 \\ 1 & \beta & 0 \\ 0 & 1 & \gamma \end{bmatrix}, \tag{B.3}$$

which means that the resulting combined neuron model circuit will be made up of three exponential decay elements connected in cascade but, crucially, with no multiplication factors coupling them. The best way to achieve this form without altering the dynamics is by applying a linear transformation. We must find a linear transformation that, when applied to the matrix exponential, such as equation B.1, yields a transformed matrix exponential of the form shown in equation B.3. The following transformation matrix fulfills this requirement:

$$\mathbf{Q} = \begin{bmatrix} b & 0 & 0 \\ f & c & 0 \\ 0 & 0 & d \end{bmatrix}. \tag{B.4}$$

Thus, we must now find the values of b, c, d, and f that transform equation B.1 into the form of equation B.3.

Applying this transformation matrix to the general solution, equation A.1, gives

$$\mathbf{y} = \mathbf{Q}\mathbf{z}, \tag{B.5}$$

which gives rise to the linear transformation of equation A.1 given by

$$\mathbf{z}_{k+1} = \mathbf{Q}^{-1}\mathbf{P}_m\mathbf{Q} \cdot \mathbf{z}_k, \qquad \mathbf{z}(0) = \mathbf{z}_0, \tag{B.6}$$

where

$$\mathbf{Q}^{-1}\mathbf{P}_m\mathbf{Q} = \begin{bmatrix} \alpha & 0 & 0 \\ \left(-\frac{f}{bc}\alpha + \frac{1}{c}r\right)b + \frac{f}{c}\beta & \beta & 0 \\ \frac{b}{d}q + \frac{f}{d}p & \frac{c}{d}p & \gamma \end{bmatrix}. \tag{B.7}$$

As was already mentioned, in order to keep simplicity in the circuit, the transformed matrix exponential should be in the form of equation B.3. The following conditions ensure that this is accomplished:

$$\begin{aligned} -f\alpha + br + f\beta &= c, \\ bq + fp &= 0, \\ cp &= d. \end{aligned} \tag{B.8}$$

To maintain the dynamic of the membrane potential, it is required that $d = 1$. Now we can easily solve for b, c, and f, which results in

$$b = \frac{-1}{(\beta - \alpha)q - pr} \tag{B.9}$$

$$c = \frac{1}{p} \tag{B.10}$$

$$d = 1 \tag{B.11}$$

$$f = \frac{q}{p((\beta - \alpha)q - pr)}. \tag{B.12}$$

In turn, the transformed initial conditions for a single presynaptic action potential should then be

$$\mathbf{z}(0) = \begin{bmatrix} \frac{1}{b}\frac{we}{\tau_\alpha} \\ -\frac{f}{bc}\frac{we}{\tau_\alpha} \\ 0 \end{bmatrix}. \tag{B.13}$$

In the context of the combined circuit, the first and second row of equation B.13 correspond to the initial conditions of the exponential decay units (in the same order), which produces the alpha function synapse (see Figure 2A). The first row requires us to adjust the synaptic weight according to

$$w_{adj} = \frac{1}{b}\frac{we}{\tau_\alpha}, \tag{B.14}$$

where b is defined in equation B.9 and w is the synaptic weight.

In turn, the second row gives the initial condition that must be applied to the second exponential decay unit of the alpha function synapse, which is carried out by making

$$\rho = -\frac{f}{bc}\frac{we}{\tau_\alpha}, \tag{B.15}$$

where b, c, and f are defined in equations B.9 to B.12. It is important to point out that ρ must be nonzero in combined neuron models based on alpha and beta function synapses; otherwise, $\rho = 0$ and can be neglected. Furthermore, since equation A.1 describes the dynamics of the combined neuron model in between spikes, both w_{adj} and ρ should be added every time an action potential is received at the synapse (see Figure 2A).

Finally, the soma model requires initial conditions set to zero. Meeting these conditions will guarantee an EI of a leaky integrate-and-fire model receiving dendritic currents modeled by alpha functions.

Appendix C: Exponential Decay Synapse-Based Combined Neuron Model Matrix Exponential

C.1 Matrix Exponential of the Combined Neuron Model Based on Exponential Decay Synapse.

$$\begin{bmatrix} e^{-\frac{\Delta}{\tau_e}} & 0 \\ \frac{\tau_e \tau_m (e^{-\frac{\Delta}{\tau_e}} - e^{-\frac{\Delta}{\tau_m}})}{C(\tau_e - \tau_m)} & e^{-\frac{\Delta}{\tau_m}} \end{bmatrix}. \tag{C.1}$$

Acknowledgments

This work was funded by the EU-FPV IST program in Future and Emerging Technologies IST-2001-33066 (to T.C.P.) and ESPRC grant GR/R37968/01(P)(to T.C.P.). R.G. is funded by CONACYT, Mexico. A.M. and M.D. are partially funded by DIP F1.2, EU Grant 15879 (FACETS), and BMBF Grant 01GQ0420 to BCCN Freiburg. Special thanks go to Manuel Sánchez-Montañés with whom we had many useful discussions regarding exact integration solutions and axonal delay implementations. Thanks also go to Carlo Fulvi-Mari, who provided the analytical expression for the asymptotic peak synaptic output.

References

Abbott, L. F., & Nelson, F. B. (2000). Synaptic plasticity: Taming the beast. *Nature Neuroscience, 3*, 1179–1183.

Alfke, P., & Hitesh, P. (2005). *Achieving breakthrough performance with Virtex-4, the world's fastest FPGA.* (Online conference), Accessed February 2005 at http://www.techonline.com/community/ed_resource/webcast/37558.

Barnes, C. W., Tran, B. N., & Leung, S. H. (1985). On the statistics of fixed-point roundoff error. *IEEE Trans. Acoustic, Speech, and Signal Proc., 33*, 595–606.

Crook, S. M., Ermentrout, G. B., Vanier, M. C., & Bower, J. M. (1997). The role of axonal delay in the synchronization of networks of coupled cortical oscillators. *J. Comp. Neurosci., 4*, 161–172.

Diesmann, M., Gewaltig, M., Rotter, S., & Aertsen, A. (2001). State space analysis of synchronous spiking in cortical neural networks. *Neurocomputing, 38–40*, 565–571.

Friedrich, R. W., Habermann, C. J., & Laurent, G. (2004). Multiplexing using synchrony in the zebrafish olfactory bulb. *Nat. Neuroscience, 7*, 862–871.

Friedrich, R. W., & Laurent, G. (2001). Dynamic optimization of odor representations by slow temporal patterning of mitral cell activity. *Science, 291*, 889–894.

Gerstner, W., & Kistler, W. (2002). *Spiking neuron models.* Cambridge: Cambridge University Press.

Graas, E. L., Brown, E. A., & Lee, R. H. (2004). An FPGA-based approach to high-speed simulation of conductance-based neuron models. *Neuroinformatics, 2*(4), 417–436.

Gütig, R., Aharonov, R., Rotter, S., & Sompolinsky, H. (2003). Learning input correlations through nonlinear temporally asymmetric Hebbian plasticity. *J. Neurosci., 23*, 3697–3714.

Jack, J. J. B., Noble, D., & Tsien, R. W. (1975). *Electric current flow in excitable cells.* New York: Oxford University Press.

Lambert, J. D. (1973). *Computational methods in ordinary differential equations.* New York: Wiley.

Maxfield, C. (2004). *The design warrior's guide to FPGAs.* Burlington, MA: Newnes.

Pearce, T. C., Koickal, T., Fulvi-Mari, C., Covington, J. A., Tan, F. S., Gardner, J. W., & Hamilton, A. (2005). Silicon-based neuromorphic implementation of the olfactory pathway. In *2005 2nd International IEEE Conference on Neural Engineering* (pp. 307–312). Piscataway, NJ: 2005.

Rotter, S., & Diesmann, M. (1999). Exact digital simulation of time-invariant linear systems with applications to neuronal modeling. *Biological Cybernetics, 81*, 381–402.

Song, S., Miller, K. D., & Abbott, L. F. (2000). Competitive Hebbian learning through spike-timing dependent synaptic plasticity. *Nature Neuroscience, 3*, 919–926.

Tuckwell, H. C. (1988). *Introduction to theoretical neurobiology.* Cambridge: Cambridge University Press.

Weinstein, R. K., & Lee, R. H. (2006). Architecture for high-performance FPGA implementations of neural models. *Journal of Neural Engineering, 3*, 21–34.

Xilinx. (2002). *Virtex-II pro: Platform FPGA handbook.* San Jose, CA.

Xilinx. (2003). *Development system reference guide*. San Jose, CA. Retrieved March 17, 2005, from http://toolbox.xilinx.com/docsan/xilinx6/books/docs/dev/dev.pdf

Xilinx (2004). Virtex-4 family overview. In *Virtex-4 handbook.* San Jose, CA. Retrieved October 21, 2004, from http://direct.xilinx.com/bvdocs/publications/ds112.pdf

Received August 10, 2005; accepted April 27, 2006.