A log-domain implementation of the Izhikevich neuron model

André van Schaik, Craig Jin, Alistair McEwan
Electrical and Information Engineering,
The University of Sydney,
Sydney, NSW 2006, Australia
andre@ee.usyd.edu.au

Tara Julia Hamilton
School of Electrical Engineering and Telecommunications
University of New South Wales
Sydney, NSW 2052, Australia

Abstract— **We present an implementation of the Izhikevich neuron model which uses two first-order log-domain low-pass filters and two translinear multipliers. The neuron consists of a leaky-integrate-and-fire core, a slow adaptive state variable and quadratic positive feedback. Simulation results show that this neuron can emulate different spiking behaviours observed in biological neurons.**

I. Background

In 2003 Eugene Izhikevich presented a neuron model that exhibited Hodgkin-Huxley-type dynamics but had the computational efficiency of an integrate-and-fire neuron model [1]. It is capable of exhibiting various spiking and bursting behaviours of known cortical neurons through the adjustment of only four parameters. As a result the Izhikevich (Iz) neuron model has become quite popular for the simulation of spiking neural networks.

The Iz model has recently been implemented very efficiently in the voltage domain by Wijekoon and Dudek using only 14 MOS transistors [2, 3]. Here we present a log-domain implementation of the Iz model, while in a companion paper we present the implementation of the Mihalas-Niebur neuron [4], a recent model showing similar spiking behaviour and computational efficiency as the Iz model, but using a more modular approach so that additional behaviours can be easily added. Both implementations use many of the same building blocks.

In these two papers we are not claiming that these neuron models are better than others, or that our implementation is necessarily better than others in the literature. Rather we are determining how well we can map the equations of the neuron model to log-domain circuits. Furthermore we want to compare the performance of the two neurons models as log-domain implementations and to that end a test chip has been implemented containing 57 identical copies of each type. Here we first present the Iz model in section II, followed by the circuit implementation in section III. In section IV we present the result of various simulations of the circuits and we conclude in section V.

II. The Izhikevich Neuron Model

The Izhikevich model is described by three equations [1]:

$$\frac{dv}{dt} = 0.04v^2 + 5v + 140 - u + I_{ex} \tag{1}$$

$$\frac{du}{dt} = a(bv - u) \tag{2}$$

$$\text{if } v > 30 \text{ mV, then } v \leftarrow c \text{ and } u \leftarrow u + d \tag{3}$$

where v represents the membrane potential of the neuron, u represents a slow membrane recovery variable accounting for the activation of K^+ ion currents and inactivation of Na^+ ion currents and I_{ex} represents the excitatory input current. The third equation shows that v and u are reset to c and $u + d$, respectively, when v reaches 30 mV.

According to [1], v and u are dimensionless variables and a, b, c, and d are dimensionless parameters, while t is expressed in [ms]. However, this leads to some inconsistencies of the units in the equations. The left hand side of equations (1) and (2) are in [ms^{-1}], while the right hand side is dimensionless. Equation (3) compares v with 30 mV. Fortunately [1] includes the Matlab code for numerical simulation of the model and this allows us to rewrite the equations using correct electrical units:

$$\tau\frac{dv}{dt} = 0.04[\mathrm{mV}^{-1}]v^2 + 5v + 140[\mathrm{mV}] - u + 1[\Omega]I_{ex}[\mathrm{mA}] \tag{4}$$

$$\tau\frac{du}{dt} = a(bv - u) \tag{5}$$

$$\text{if } v > 30 \text{ [mV], then } v \leftarrow c \text{ and } u \leftarrow u + d \tag{6}$$

Now v, u, c and d are expressed in [mV], a and b are dimensionless gains, and τ is 1 ms. Simulating this electrical model with the appropriate values for the parameters results in the same curves for v and u as the Matlab code from the paper.

To facilitate implementing the Izhikevich model on chip using log-domain filters, we first rewrite equations (4)-(6) with both v and u represented as currents, by simply replacing every term in [mV] by an equivalent one in [nA] as follows:

$$\tau \frac{dI_v}{dt} = \frac{I_v^2}{25\,[\mathrm{nA}]} + 5I_v + 140[\mathrm{nA}] - I_u + I_{ex} \tag{7}$$

$$\tau \frac{dI_u}{dt} = a(bI_v - I_u) \tag{8}$$

$$\text{if } I_v > 30\ [\mathrm{nA}]\text{, then } I_v \leftarrow c \text{ and } I_u \leftarrow I_u + d \tag{9}$$

Now c and d are also currents in [nA].

In normal operation I_v varies between -90 nA and +30 nA depending on the input current and the parameters chosen. Since log-domain filters are much simpler to implement with unidirectional currents than with bidirectional currents, we add an arbitrary offset current of +100 nA to I_v and of +100b nA to I_u. Adding a constant to these variables doesn't affect their time derivative, so we can write:

$$I_v = I_{mem} - 100\ ; \qquad I_u = I_{slow} - 100b \tag{10}$$

$$\tau_v \frac{dI_{mem}}{dt} = \frac{(I_{mem} - 100)^2}{25\,[\mathrm{nA}]} + 5(I_{mem} - 100) + 140 - (I_{slow} - 100b) + I_{ex} \tag{11}$$

$$\tau_u \frac{dI_{slow}}{dt} = a(b(I_{mem} - 100) - (I_{slow} - 100b)) \tag{12}$$

$$\text{if } I_{mem} > 130\ [\mathrm{nA}], \quad \text{then } I_{mem} \leftarrow c + 100 \text{ and } I_{slow} \leftarrow I_{slow} + d \tag{13}$$

Finally, simplifying the above equations leads to a model that can be mapped directly onto two first-order log-domain filters:

$$\tau_{mem} \frac{dI_{mem}}{dt} + I_{mem} = \frac{1}{3}\left(\frac{I_{mem}^2}{25} + 40 + 100b - I_{slow} + I_{ex}\right) \tag{14}$$

$$\tau_{slow} \frac{dI_{slow}}{dt} + I_{slow} = bI_{mem} \tag{15}$$

$$\text{if } I_{mem} > 130\ [\mathrm{nA}], \quad \text{then } I_{mem} \leftarrow c + 100 \text{ and } I_{slow} \leftarrow I_{slow} + d \tag{16}$$

where $\tau_{mem} = 1/3$ ms and $\tau_{slow} = 1/a$ ms. Typical values for the model parameters are shown in Table 1.

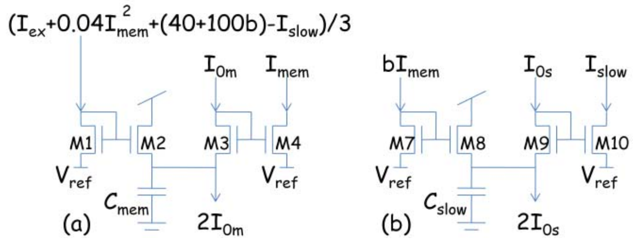

Figure 1. Tau-cell implementation of (a) the membrane variable, and (b) the slow variable.

Table 1. Typical values adapted from [1].

τ_{mem} = 1/3 ms	$\tau_{slow} = 1/a$ = 50 ms
$a = 0.02$	$c + 100 = 35$ nA
$b = 0.2$	$d = 2$ nA

III. CMOS IMPLEMENTATION

The basic building block for our log-domain implementation of a first-order low-pass filter is the tau-cell [5]. This building block is used to implement both the membrane variable (I_{mem}) and the slow variable (I_{slow}) as shown in Figure 1a and b. Assuming the circuit operates in weak inversion, it is most easily analysed using the translinear principle, which states that (for Figure 1a):

$$I_{M1} I_{M3} = I_{M2} I_{M4} \tag{17}$$

while the current through the capacitor is given by:

$$C_{mem} \frac{dV_{mem}}{dt} = I_{M2} - I_{0m} \tag{18}$$

Assuming that the gate capacitance of M3 plus M4 is significantly smaller than C_{mem} so that we can ignore the dynamics of this output mirror, we can also write:

$$I_{M4} = I_{M3} e^{\frac{V_{mem} - V_{ref}}{U_T}} \tag{19}$$

where U_T is the thermal voltage. Since I_{M3} is a constant ($I_{M3} = I_{0m}$), the derivative of I_{M4} is simply given by:

$$\frac{dI_{M4}}{dt} = \frac{I_{M4}}{U_T} \frac{dV_{mem}}{dt} \tag{20}$$

With these equations we can write:

$$\tau_{mem} \frac{dI_{M4}}{dt} + I_{M4} = I_{M1} \tag{21}$$

The time constants of the implementations of Figure 1a and b are given by:

$$\tau_{mem} = \frac{U_T C_{mem}}{I_{0m}} \qquad \tau_{slow} = \frac{U_T C_{slow}}{I_{0s}} \tag{22}$$

Note that since the tau-cell and the multipliers all use an alternating topology, i.e., only gates are connected to gates and only sources are connected to sources in the translinear loop, none of the transistors need their local substrate tied to their source [6]. This means that none of the transistors need a separate well, reducing the size of their layout.

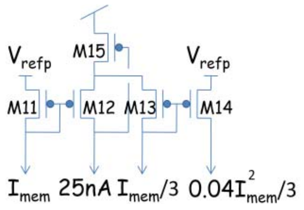

Figure 2. A translinear multiplier implementing the feedback term.

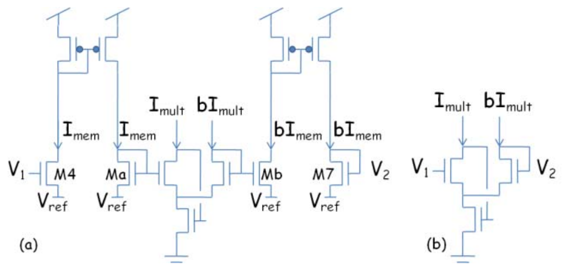

Figure 3. A translinear multiplier to create bI_{mem}. (a) Naïve implementation; (b) optimised implementation.

In order to complete the implementation, we need to create the $1/3 \times I_{mem}^2/25$ [nA] term, which can be done in the current domain using a translinear multiplier. Since all currents are positive, this can be a simple one-quadrant multiplier as shown in Figure 2.

The final term to implement is bI_{mem} as the input to the circuit of Figure 1b. This could simply be implemented by appropriate transistor scaling, but this would not allow us to vary b post fabrication. Alternatively, we could use another multiplier as in Figure 2, using I_{mem}, I_{mult}, and bI_{mult}, respectively, as inputs. However, by implementing this second multiplier with NMOS transistors we can reduce the number of transistors. Figure 3a shows the naïve implementation using an NMOS multiplier and two PMOS current mirrors. Transistors M4 and M7 are identical to M4 and M7 in Figure 1. Inspection of Figure 3a reveals that the gate voltage of Ma is identical to that of M4 (V_1). Therefore we can simply connect these two nodes together and remove the two NMOS transistors and the current mirror. The same is true for M7 (V_2) and Mb. This leads to the much smaller circuit of Figure 3b.

The final circuit is shown in Figure 4. To implement equation (16) in addition to equations (14) and (15), the spike induced adaptation of I_{mem} and I_{slow} are added by comparing a copy of I_{mem} (using M16 and M17) with the 130 nA threshold. A chain of inverters create the logic signals V_{nspike} and V_{spike}. Transistor M18 resets I_{mem} by resetting the voltage on C_{mem} to V_{reset}, thus implementing the parameter c, while the combination of transistors M19 and M20 increases I_{slow} by a small amount set by V_d when a spike occurs, thus implementing the parameter d.

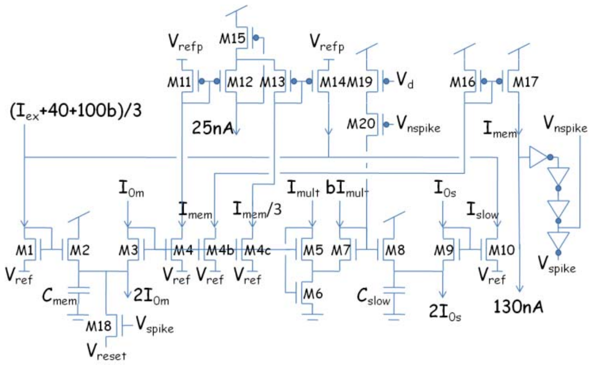

Figure 4. The Izhikevich neuron circuit.

IV. Simulation Results

We have simulated the proposed circuits using the parameters for the AMIS 0.5 μm process. All transistors are 2.8 μm wide and 4.2 μm long, unless stated otherwise. The four inverters use transistors which are 1.4 μm wide and 0.7 μm long. M18 is 5.6 μm wide and 1.4 μm long. M19 and M20 are both 14.0 μm wide and 2.8 μm and 0.7 μm long, respectively. Because the currents in the multiplier creating the quadratic feedback can get quite substantial, transistors M11-M15 are 14 μm wide and 4.2 μm long. The $I_{mem}/3$ factor could be implemented by making M4c three times as long as M4, but instead we have chosen to make M13 effectively three times as wide by repeating it three times in parallel. The capacitor values are: C_{mem} = 0.8 pF and C_{slow}= 12 pF. The total size of the neuron is 0.02 mm^2.

In the simulations, the time constants were set with I_0 = 60 pA for I_{mem} and 6 pA for I_{slow}. The resting level of the membrane current was approximately 20 nA and the resting level of the slow variable was 5 nA. V_{dd} was 3.3V, V_{reset} was 0.6 V, and V_d was 2.3 V. I_{mult} was 100 nA and bI_{mult} was varied to implement different values of b. The power consumption at rest was 2.6 μW. A single spike lasting 50 μs consumes approximately 20 μW. The reference potential for the NMOS and PMOS tau-cells was 0.4 V above ground or below V_{dd}, respectively. These reference potentials are needed so that the transistors providing the $2I_0$ currents can operate in saturation.

Figure 5 shows the simulation results. Note that for clarity we have omitted the digital spike output from the plots and spikes are indicated by the abrupt reset of I_{mem}. In Figure 5(a) the circuit behaves as a simple leaky-integrate-and-fire neuron with no adaptation (b = 0.001) and the input current is only just enough to cause the neuron to spike. This shows clearly that the membrane current can be very close to the spiking threshold for extended periods of time.

Figure 5b shows the spike frequency adaptation caused by an increase of I_{slow} (using b = 0.25) as the neuron spikes. Phasic spiking (Figure 5c) is obtained by lowering I_{ex} so that the I_{slow} adapts after initial spiking to a level where the excitatory input current is no longer enough to cause the neuron to spike. The membrane potential will remain just below the spiking threshold for as long as the constant input current remains.

Figure 5d shows accommodation in the neuron (using b = 0.5). The input current at the start is the same as in Figure 5b, which causes the neuron to spike. However, when the input current is raised to this level in three equal steps, as shown in the 60 ms to 105 ms range in Figure 5d, the neuron's threshold accommodates to each increase in input current and the neuron won't spike, even when the input current reaches its original level.

Figure 5e and Figure 5f show two types of bursting behaviour obtained when the circuits of Figure 3 are included, for different levels of excitatory current. Here V_{reset} was increased to 0.65 V to cause bursting behaviour and b was kept at 0.25. A larger input current causes Tonic Bursting (Figure 5e) while a smaller input current causes Phasic Bursting (Figure 5f).

V. Conclusions

We have presented an implementation of the Izhikevich neuron model using two first-order log-domain low-pass filters and two translinear multipliers. The equations of the model map quite naturally to this implementation. We have shown that some of the typical spike behaviours of the Izhikevich neuron can be obtained by the chip by varying the input current and the biases voltages and currents that implement the four parameters of the model.

Acknowledgments

The collaboration on this design was started at the Telluride Neuromorphic Engineering Workshop 2008. The authors thank the organisers and the sponsors of the workshop for their support.

References

[1] E. M. Izhikevich, "Simple model of spiking neurons," *IEEE Transactions on Neural Networks,* vol. 14, pp. 1569-1572, Nov 2003.

[2] J. H. B. Wijekoon and P. Dudek, "Integrated circuit implementation of a cortical neuron," in *IEEE International Symposium on Circuits and Systems*, 2008, pp. 1784-1787.

[3] J. H. B. Wijekoon and P. Dudek, "Compact silicon neuron circuit with spiking and bursting behaviour," *Neural Networks,* vol. 21, pp. 524-534, Mar-Apr 2008.

[4] S. Mihalas and N. Niebur, "A Generalized Linear Integrate-And-Fire Neural Model Produces Diverse Spiking Behaviors," *Neural Computation,* vol. 21, pp. 704-718, 2009.

[5] A. van Schaik and C. Jin, "The tau-cell: a new method for the implementation of arbitrary differential equations," in *IEEE International Symposium on Circuits and Systems - ISCAS 2003*, 2003, pp. 569 - 572.

[6] B. A. Minch, "Analysis and Synthesis of Static Translinear Circuits," Cornell University, Ithaca, NY 2000.

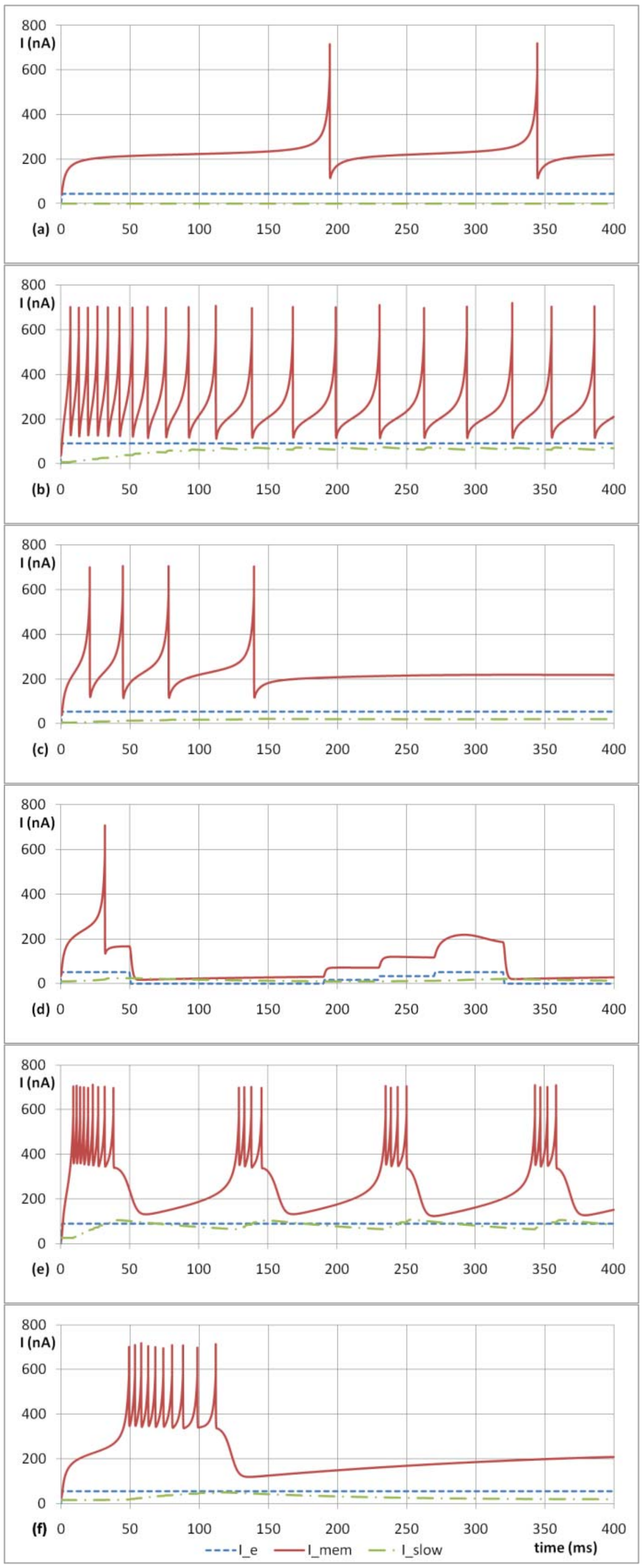

Figure 5. Examples of spike patterns. (a) Class 1; (b) Spike Frequency Adaptation; (c) Phasic Spiking; (d) Accommodation; (e) Tonic Bursting; (f) Phasic Bursting.