Emulation of Ion Channel Dynamics in CMOS VLSI

Bassel Hanafi
Electrical & Computer Engineering Dept.
University of California, San Diego
La Jolla, CA 92093
bhanafi@ucsd.edu

Yousr Abdelmaksoud
Electrical & Computer Engineering Dept.
University of California, San Diego
La Jolla, CA 92093
yabdelma@ucsd.edu

Abstract

In this paper a novel VLSI implementation is proposed to emulate ion channels gating dynamics of neurons. The technique is demonstrated by implementing K^+ ion channels of a Morris-Lecar (ML) neuron model. The circuit emulates the opening and closing of 256 individual ion channels and is implemented in $0.18\mu m$ CMOS technology. Simulation results indicate that the circuit consumes $2.7\mu A$ from a $0.8V$ supply, and has an area estimate of $150 \times 150\mu m^2$.

1 Introduction

The fundamental computing unit of a brain is the neuron. Owing to extensive structure it has complex computing capabitities. In contrast, the fundamental computing unit of a microprocessor is a transistor, which is capable of extensive computing as well, but only at much larger power consumptions. Since neuron computing is orders of magnitude more power effeccient than transistor computing, there has always been a strong motivation to persue VLSI circuits that can both capture neuron dynamics accuratley and remain area effecient.

Previous VLSI neuron implementations reported in literature are numerous and range from those based on a simple integrate-and-fire model [1], to complex full Hodgkin-Huxley model [2], passing by the simplified two dimensional Izhikevich model [3]. But non of these models was concerned with modeling the openeing and closing dynamics of individual neuron channels, since gating variables were treated as analog continuous time variables. A more realistic representation is to model channel conductances with quantized discrete time variables controlled with the corresponding opening and closing rates $\alpha(V_m)$ and $\beta(V_m)$. In this paper an effecient digital circuit implementation is presented to capture the quantized stochastic nature of ion channels opening and closing in a neuron.

The paper is organized as follows, in section 2, concepts of the proposed emulation architecture are explained. Section 3 discusses in more details the circuit implementation of the neuron model. Simulation results are presented in section 4. And, finally, the conclusion and future work are included in section 5.

2 Morris-Lecar Model and Gate Channel Dynamics

In this section a brief introduction to the ML model dynamics and equations is provided. Next, the emulation architecture proposed in this work is explained, and system level simulation results based on behavioral models are presented.

2.1 Morris-Lecar model

The ML model [4] is a simplified reduced neuron model. The model was originally employed to model the barnacle giant muscle fiber with Ca^{+2} activation (excitation) and K^{+} inactivation (recovery). A suitable and intuitive circuit model for channel conductances in ML neuron is shown in figure 1. If the Ca^{+2} activation is assumed to be infinitley fast (instantaneous), the model can be further simplified into a two-dimensional model, and the system equations are found to be as follows:

$$\begin{aligned} C_m \frac{dV_m}{dt} &= (I_{ext} - I_K - I_{Ca} - I_L) \\ I_K &= g_K.w.(V_m - E_K) \\ I_{Ca} &= g_{Ca}.m_\infty.(V_m - E_{Ca}) \\ I_L &= g_L.(V_m - E_L) \\ \frac{dw}{dt} &= \alpha.(1-w) - \beta.w \\ &= \frac{w_\infty - w}{\tau_\infty} \end{aligned} \tag{1}$$

where m_∞, w_∞ and τ_∞ are all functions of the membrane voltage V_m and have the form:

$$\begin{aligned} w_\infty &= \frac{\alpha}{\alpha+\beta} = 0.5 \cdot (1 + tanh(V_m/30)) \\ \tau_\infty &= \frac{1}{\alpha+\beta} = \frac{5}{cosh(V_m/60)} \quad [msec] \\ m_\infty &= 0.5 \cdot (1 + tanh(\frac{1+V_m}{15})) \end{aligned} \tag{2}$$

It should be noted that based on the above equations, $\alpha(V_m)$ and $\beta(V_m)$ have exponential dependence on V_m.

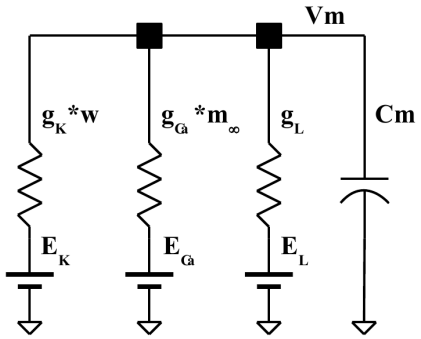

Figure 1: Morris-Lecar conductance based model

2.2 Gate Channel Dynamics

To understand how this model can capture neuron firing, external DC current is assumed to charge the membrane capacitance, and membrane potential starts ramping up. When it reaches a certain firing threshold, the fast activation Ca^{+2} channels open and the membrane voltage spikes to a peak value, after some delay the slow inactivation K^{+} channels open and the membrane potential drops slowly, until most of the Ca^{+2} channels close and the membrane potential hits a minima. Then the external current starts charging the membrane capaciance again, and the same process is repeated.

2.3 Proposed Emulation Method

Now, consider the K^+ channel of the ML model discussed above with only one gating mechanism. Thus, only one gating variable n is required. The rates $\alpha(V_m)$ and $\beta(V_m)$ are the average opening and closing state transition rates respectively, and w is the fraction of the total gates that are open.

Consider further that there is a finite number of such gates (*i.e.*, M*-gates*), such that when $w = 1$ all M gates open and when $w = 0$ all M gates oclosed. From equation 1, $w = 1$ corresponds to a total K^+ channel conductance of g_K. Consequently, for a K^+ channel composed of M gates, channel conductance can be modeled as M *switched conductances* with a conductance of $\frac{g_K}{M}$ each. This is better depicted in figure 2

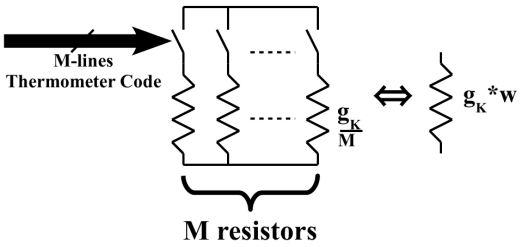

Figure 2: Switched conductance implementation of the gated K^+ conductance

Now, let n be the number, rather than the fraction, of open gates, *i.e.*, $n = M$ means that all gates are open. Clearly, n can be implemented as an M-thermometer code binary number. Modeling gated conductance this waydirectly implements the multiplication term $g_K.w$ because for n switches activated, the net K^+ conductance becomes $g_K.\frac{n}{M}$, where, $\frac{n}{M}$ is the fraction of open gates *i.e.*, w.

To be able to fully model the K^+ channel, the rate differential equation in 1 needs to be implemented. To express the equation in terms of n, the equation can be rewritten as:

$$\frac{dn}{dt} = \alpha \cdot (M - n) - \beta \cdot n \tag{3}$$

Since the first and second terms in [3] represent the transition rate from closed to open channels and from open to closed channels respectively, it is best to implement them with circuits that produce an event at a controlled rate (gate transition rates), and then integrate the number of events to give the exact number of open/closed gates. These circuits by definition are voltage/current controlled oscillators, and the integrator is a simple digital counter. Hence equation [3] can be rwritten as

$$\begin{aligned} \Delta n &\approx (\frac{1}{T_\alpha} - \frac{1}{T_\beta})\Delta t \\ n(t) &= n_o + floor(f_\alpha t) - floor(f_\beta t) \end{aligned} \tag{4}$$

where $f_\alpha = \frac{1}{T_\alpha} = \alpha.(M - n)$, and $f_\beta = \frac{1}{T_\beta} = \beta.n$.

Shown in figure 3 is the block diagram for the full system implementation. It is broken down into the following sub-blocks

- A voltage controlled oscillator (VCO) called the β oscillator (VCO_β) has an analog control signal V_m and a digital control signal n. The VCO output frequency is given by $f_\beta = \beta(V_m).n$.
- Another VCO called the α osciilator (VCO_α) has an analog control signal V_m and digital control signal $M - n$. This VCO output frequency is given by $f_\alpha = \alpha(V_m).(M - n)$.

- An asynchronous $Up/Down$ binary counter that operates as follows, a rising edge at the Up control increments the counter by 1, a falling edge at the Dn control decrements the counter by 1. The counter output saturates when it attempts to exceed M or go below 0.
- A binary-to-thermometer decoder to convert the counter output into a thermometer code to control the channel conductance.

Note that $M - n$ is implemented as $\bar{n}$ or the *one's complement* of n. As explained earlier, the combination of the VCO's and the counter yields at the counter output the number of open gates, and hence implements the rate equation [3].

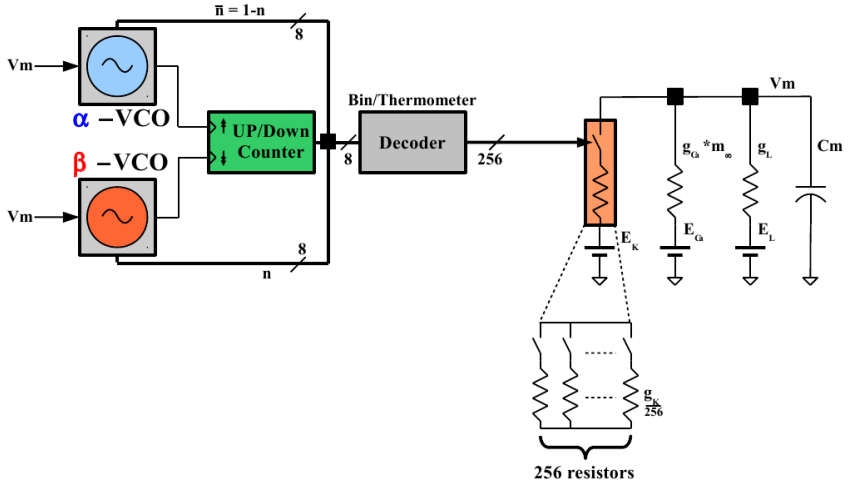

Figure 3: System level block diagram

2.4 A Simple Test Case

To verify the concept, behavioral models were built for the system in figure 3 using *VerilogA* and simulated with $CADENCE^{®}$. The model paramters used for simulations are:

$$
\begin{aligned}
&C_m = 1\mu F/cm^2 \\
&E_{Ca} = 100mV, \;\; g_{Ca} = 1.1mS/cm^2 \\
&E_K = -70mV, \;\; g_K = 2mS/cm^2 \\
&E_L = -50mV, \;\; g_L = 0.5mS/cm^2 \\
&I_{ext} = 35\mu A/cm^2 \\
&M = 256 \;\; gates
\end{aligned} \tag{5}
$$

For simplicity, the membrane area was supposed to be 1 cm^2. Since $M = 256$, the K^+ conductance is split into 256 switched conductances each of value $7.8125mS$ ($128k\Omega$). Also, for the sake of comparison the same set of equations were solved using $MATLAB^{®}$. Shown in figure 4 is the membrane voltage prediceted by both MATLAB and CADENCE.

It is clear that both results match,and thus verify the concept. Also, shown in figure 5 is the instantaneous K^+ conductance as a function of time exhibiting the quantized nature of channel conductance.

3 Circuit Implementation

In this work, the K^+ channel dynamics only is implemented in circuits, whereas, the rest of the model remains behavioral. The circuits were designed on a $0.18\mu m$ CMOS process. Although, the nominal supply voltage for this process is 1.8 V, the circuits were designed to operate from a 0.8

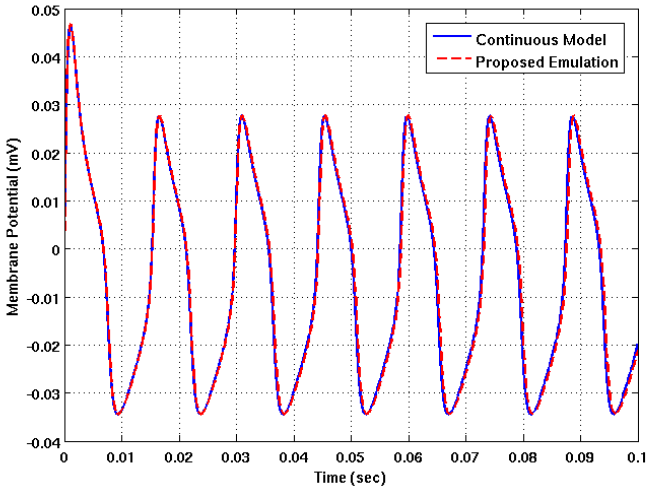

Figure 4: Behavioral system simulation using $CADENCE$ and continuous solution using $MATLAB$

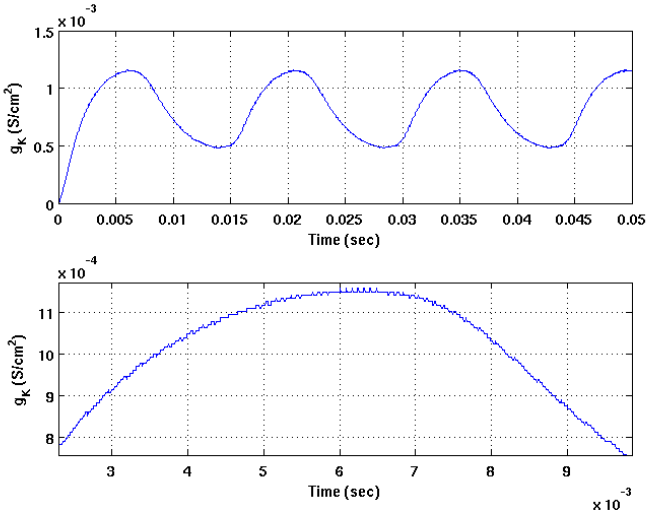

Figure 5: Zoomed version of the instantaneous net K^+ conductance $g_K \frac{n}{M}$

V supply to be able to reduce the circuit power consumption drastically. This is feasable because the circuit maximum operating frequency ($max[f_\alpha, f_\beta]$) is in the order of $100kHz$ which is much lower than the transistor cutoff frequency for this technology.

The implemented circuits are VCO_α, VCO_β, and the $Counter/Decoder$ circuit. All circuits were designed to be ultra-low power, and hence used transistor channel lengths that were many time larger than the minimum transistor channel length. Also, since the membrane potential swings between positive and negative values, while the lowest available potential for a single ended implementation is zero, the circuits were designed to operate centered around a DC shift of $350mV$.

3.1 The Oscillators

For VCO_β, an oscillator is required that has a frequency decreasing exponentially with an analog control variable V_m and increaing linearly with a digital control variable n. This can be done by

building a current controlled oscillator whose frequency depends linearly on a control current I_C, then generate I_C such that it has both an exponential dependence on V_m, and a linear dependence on n.

A practical realization for I_C is the output of a switched current source DAC whose input is a digital word n and whose reference current I_{exp} is an exponential function in V_m, such that $I_C = n \times I_{exp}$. I_{exp} is implemented using a sub-threshold $PMOS$ transistor whose gate is tied to V_m. This results in a current that falls exponentially with V_m. In general this current is mirrored to $M-1$ (M=256 in this case) current sources to build the current DAC. The current of the $M-1$ sources is then added form I_C. The circuit for generating I_C is shown in figure 6. Note that due to the limitations on the maximum channel length of the device, the $PMOS$ device generating I_{exp} is implemented as two $PMOS$ devices in series. Also, the DAC is implemented with 256 switched current sources instead of 255 such that when $n = 0$ the $I_C \neq 0$ to avoid turning the oscillator off, and since its just a single LSB shift it will not affect the operation significantly.

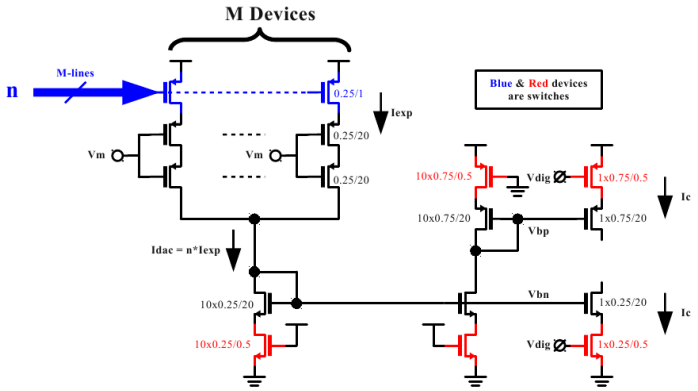

Figure 6: Schematic of the I_{DAC} generation circuit for the VCO_β

The low power current controlled oscillator is implemented as a *Schmitt-Trigger* based relaxation oscillator explained as follows:

- The oscillator is based on a schmitt-trigger that has an upper threshold V_H above which the output is Low, and a lower threshold V_L below which the output is $High$, hence it has a hysteresis voltage $V_{HYS} = V_H - V_L$.
- The output of the schmitt-trigger is used to control two current sources I_{UP} and I_{DOWN} to charge or discharge the capacitor C_V respectively.
- When the schmitt trigger output is $High$, the capacitor C_V charges linearly from the I_{UP} current source, and when the output is Low, C_V discharges linearly from the I_{DOWN} current source.
- The current sources are chosen such that $I_{UP} = I_{DOWN} = I_C$, and the capacitor voltage is fed back to be the input of the schmitt trigger.
- At steady-state, the capacitor voltage will have a trianglular waveform swinging between V_L to V_H with a slope I_C/C_V, and falling from V_H to V_L with slope of -I_C/C_V.

As a result, the oscillator frequency is given by the relation:

$$f_\beta = \frac{I_C}{2 \cdot V_{HYS} \cdot C_V} \tag{6}$$

The current controlled oscillator and its circuit details are shown in both figures 7 and 8.

VCO_α is implemented in a similar fashion to VCO_β, except now the oscillator frequency needs to have an exponentially increasing dependence on an analog control variable V_m. Thus, the same oscillator is used with the following modifications:

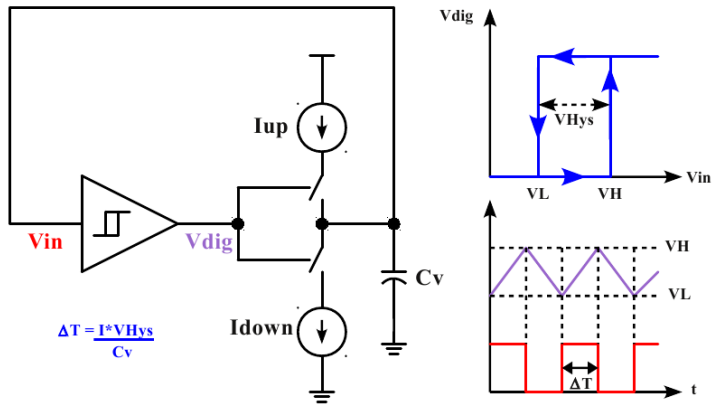

Figure 7: Schematic of current controlled oscillator

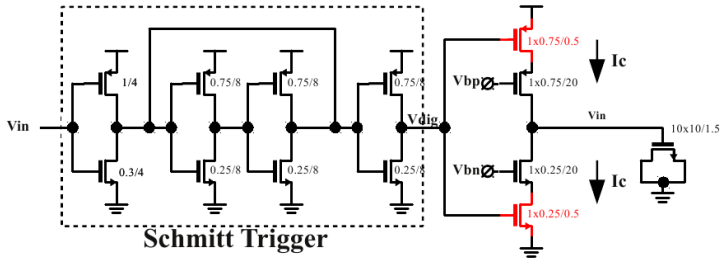

Figure 8: Circuit details of the current controlled oscillator

- An $NMOS$ transistor is used to generate I_{exp} rather than a $PMOS$ transistor.
- The digital control word is now $\bar{n}$ instead of n.

Hence, the only changes are in the current generation circuitry as shown in figure 9.

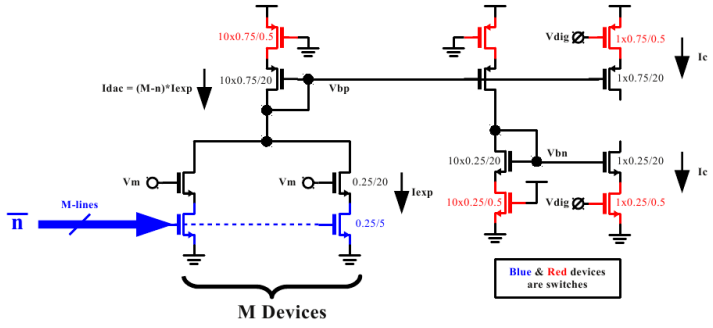

Figure 9: Schematic of the I_{DAC} generation circuit for the VCO_{α}

3.2 The Counter/Decoder Implementation

Since a binary code is not needed anywhere in the implementation, it was found more efficient to build a counter that counts in thermometer code directly rather than implementing a separate binary

counter and a binary-to-thermometer decoder. This counter is found to be nothing more than an M-stage bidirectional shift register. Shown in figure 10 is a pictorial representation of a bidirectional shift register. The register content is updated at every positive edge occuring at the Clk input. At a positive Clk edge the register samples the control signal applied to the $Up/\bar{D}n$ input. If it is high, the register shifts its content by a single bit to the right (counts up), if it is low it shifts its content by a single bit to the left (counts down). This implementation has a number of advantages:

- It saves power by combining both the counter and decoder circuitry into a single circuit.
- It provides n and $\bar{n}$ automatically, since registers readily provide the output and its complement.
- It automatically saturates the counter if it tries to exceed M-1 or go below 0

But a number of challenges still exist for this implementation. First, it is required to detect positive edges occuring from both VCO_α and VCO_β then combine them into a single output to update the shift register on either a cout up or a count down operation. Second, at a positive Clk edge the correct count direction needs to be sampled at the Up/Dn control. Shown in figure 11 is a circuit implemented to perform these two tasks. A clock combiner is a simple circuit that results in a very narrow pulse occuring at its output for every rising edge occuring at any of its two inputs. The D-FF, on the other hand, makes sure to issue a high signal to the Up/Dn control every time a rising edge occurs at VCO_β to perform a count up and is reset after a small delay. The delay elements are implemented to ensure accurate timing relations between different controls signals.

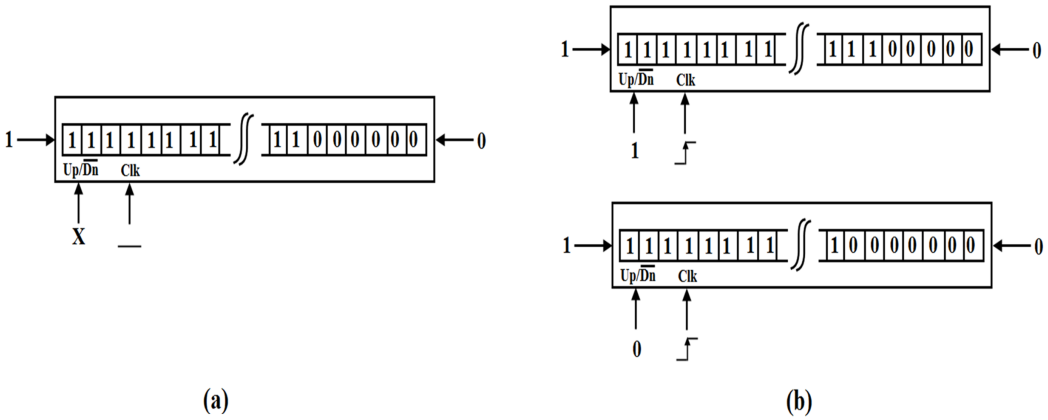

Figure 10: Bidirectional shift register operation

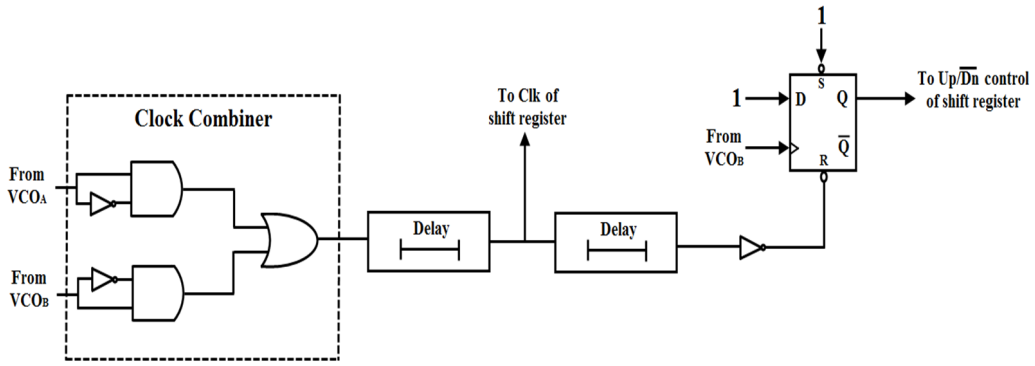

Figure 11: Control signals conditioning for the shift register

4 Simulation Results

4.1 VCO Simulations

Transistor sizing provided in the previous section was obtained after a number of iterations to fit f_α and f_β to physical values. The schmitt trigger used was previously reported in [5] and is shown in figure 8. The simulated voltage transfer characteristics is shown in figure 12 indicating that V_{HYS} is approximately $500mV$.

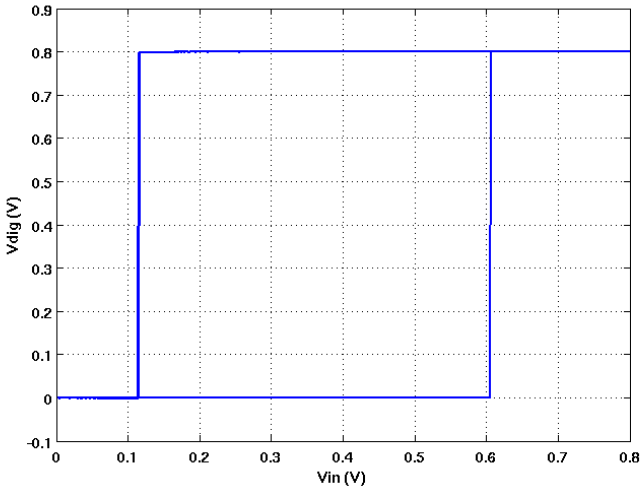

Figure 12: Schmitt trigger simulation

The DAC reference currents I_{exp} for VCO_α and VCO_β are shown in figure 13 for V_m ranging from 310 to 390 mV. Also, the DAC output current I_{DAC} for both oscillators is shown in figure 14 for $V_m = 350mV$. Note that the mirroring ratio from I_{DAC} to I_C is $1/10$ *i.e.*, $I_C = 0.1I_{DAC}$.

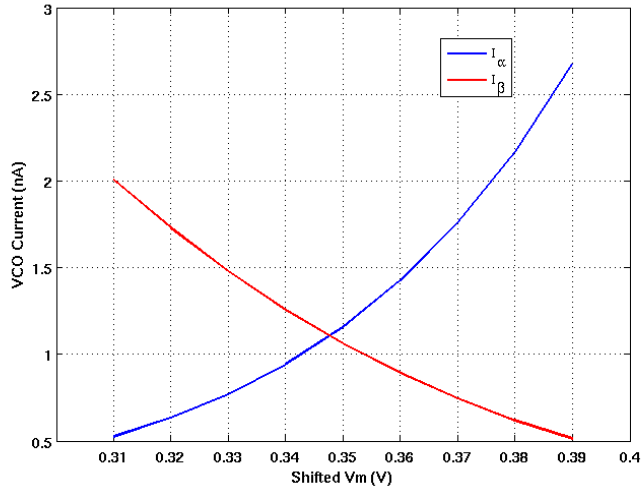

Figure 13: Simulation of I_{exp}

The internal waveforms of VCO_β are shown in figure 15 for $n = 0$ and $V_m = 350mV$.

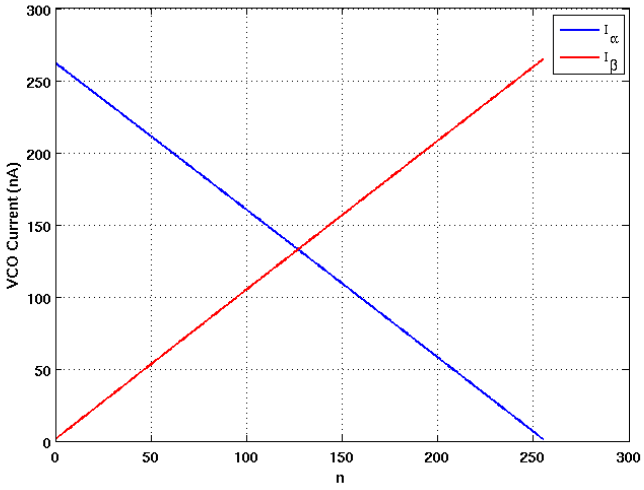

Figure 14: Simulation of I_{DAC}

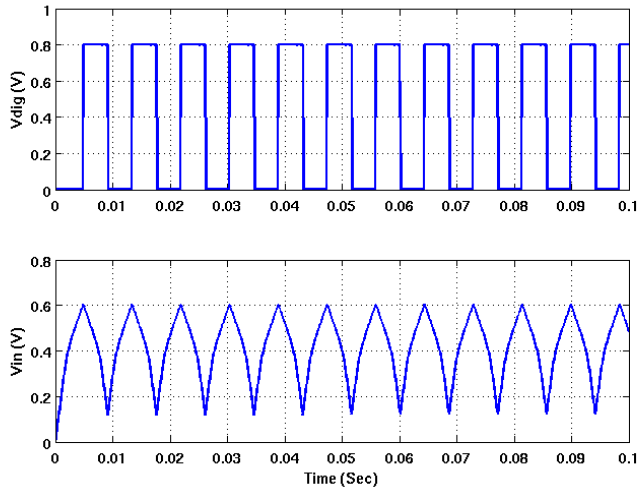

Figure 15: Internal waveforms of VCO_β at $n = 0$ and $V_m = 350mV$

Notice that the capacitor voltage is no longer a triangle waveform because the capacitor C_V is implemented with a non-linear $NMOS$ capacitor to save area. The capacitance of an NMOS transistor increases with applied transistor voltage, and this explains why the waveform slope is fast near the bottom of the wave, and slows down as the voltage increases. This of course impacts the linearity of the oscillator, but not critical to the circuit functionality.

Also, the analog control voltage V_m is swept and the frequencies f_α and f_β are recorded for $n = 0$ in the case of VCO_β and for $\bar{n} = 0$ in the case of VCO_α (*i.e.*, $f_\alpha = \alpha(V_m)$ and $f_\beta = \beta(V_m)$). The simulated α and β functions are plotted against the original functions obtained from equation 2 and are shown in figure 16. Also, an equivalent w_∞ and τ_∞ are plotted against the original ML functions as shwon in figure 17.

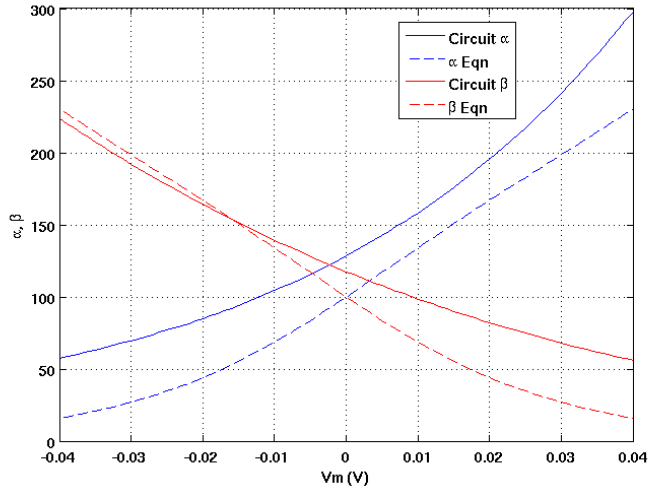

Figure 16: The synthesized $\alpha(V_m)$ and $\beta(V_m)$ vs the original ones

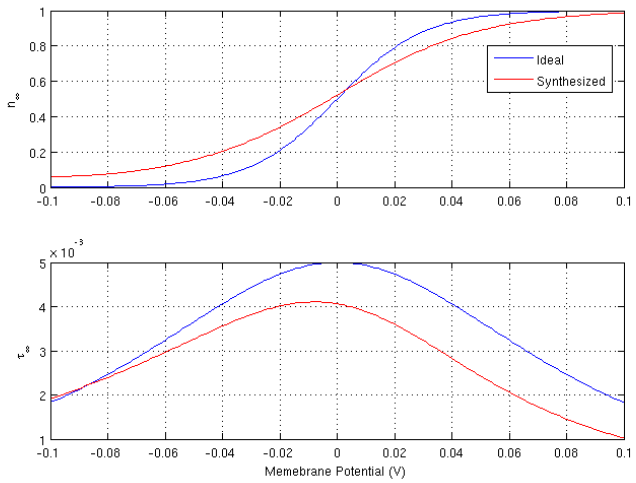

Figure 17: Equivalent w_∞ and τ_∞ of the synthesized α and β vs the original ones

4.2 System Simulations

Simulating the full circuit implementation of the system required a lot of simulation time and power, and it was decided to replace the counter by its behavioral model to reduce simulation time. At the mean time, the counter circuit was simulated separately assuming a worst case input frequency (obtained from system simulations) to verify its functionality and determine its power consumption. As was previously shown, the synthesized w_∞ and τ_∞ follow the same behavior of their original counterparts but are slightly different. This is attributed to the fact that a ML model is sensitive to its parameters, and the parameters had to be slightly modified to attain neuron firing.

$$
\begin{aligned}
&C_m = 1\mu F/cm^2 \\
&E_{Ca} = 70mV, \;\; g_{Ca} = 1.1mS/cm^2 \\
&E_K = -70mV, \;\; g_K = 2mS/cm^2 \\
&E_L = -50mV, \;\; g_L = 0.5mS/cm^2 \\
&I_{ext} = 50\mu A/cm^2 \\
&M = 256 \;\; gates
\end{aligned} \tag{7}
$$

The system was simulated with $CADENCE$, while, w_∞ and τ_∞ were extracted from continuous system simulations on MATLAB. Both results are compared in figure 18. The difference in results are mainly attributed to VCO non-linearities due to the non-linear capacitor C_V and other circuit non-linearities.

Finaly, a performance summary for the circuit is shown in table 1.

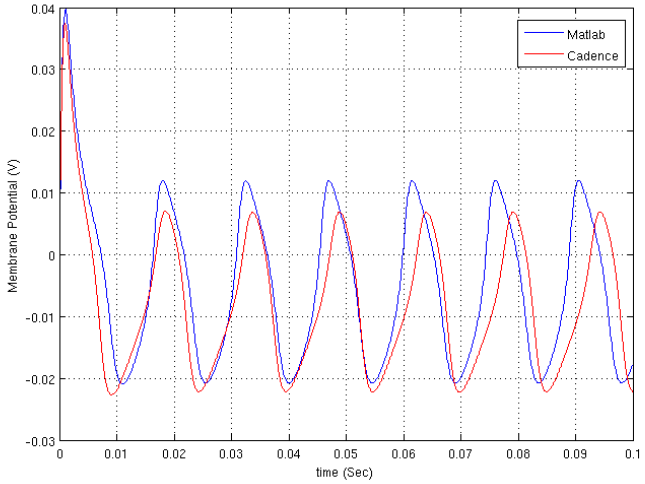

Figure 18: System simulation with the circuits of the VCO's vs continuous simulation on $MATLAB$

5 Conclusion and Future Work

In this paper a new technique for emulating gate channel dynamics on silicon was presented. Circuit implementations required for this method were explained and verified using circuit level simulation results. The implementation was based on a $0.18\mu m$ $CMOS$ technology and consumes only $2.7\mu A$ from $0.8V$ supply, and thus, is power efficient. The total area estimate is about $150\mu m^2$.

Proposals for extending the results of this work include,

1. Can implement the same rate equation with a randomizer circuit(*e.g.*, a $\Delta\Sigma$ modulator has a digital output whose average corresponds to the analog input voltage.
2. Figure out ways to further reduce the system power dissipation. Specially for implementations with larger M count, or those that need to implement the Ca^{2+} channel as well.
3. Programmable α and β exponential rate functions.
4. Extending this technique to emulate channels with more than one gating variable and/or higher powers of gating variables (*e.g.* Na^+ and K^+ channels in a Hudgen-Huxley model have fourth order gating mechanisms).

Table 1: Performance Summary

Technology	$0.18\mu m\ CMOS$
Supply	$0.8V$
VCO current (both)	$0.7\mu A$
Counter current (worst case)	$2\mu A$
Total power consumption	$2.16\mu W$
Area estimate	$150 \times 150\mu m^2$

Appendix 1: VerilogA models

Behavioral VCO

```
////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////

`include "constants.vams"
`include "disciplines.vams"

module vco_ideal_differential(Vtune,Vout_p,Vout_n);
parameter real fmin=16e9;
//parameter real fsim=16e9;
parameter real ko=1e9;
parameter real amp=3.3;
parameter real fc=1e6;  // Flicker noise corner
parameter real Qf=10;  // Resonator Quality factor
parameter real nf=100;   // Noise factor
parameter real voffs=0;

input Vtune;
output Vout_p,Vout_n;
electrical Vtune,Vout_p,Vout_n;
voltage vn,voc,vos;

real phase,npr,fo;

analog begin

fo=fmin+ko*V(Vtune);

//use if need phase noise
// >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
npr=fo/(2*Qf*amp);
npr=(npr*npr);
npr=2*nf*npr*`P_K*$temperature;
// <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<

phase = 2*`M_PI*idtmod(fo,0,1,-0.5);
```

```
V(vos)<+amp*sin(phase);
V(voc)<+amp*cos(phase);

// use if need phase noise
// >>>>>>>>>>>>>>>>>>>>>>>>>>>>
V(vn)<+flicker_noise(npr,2,"noise_leeson");
V(vn)<+flicker_noise(npr*fc,3,"noise_flick");
// <<<<<<<<<<<<<<<<<<<<<<<<<<<<

V(Vout_p)<+voffs+V(vos)+V(voc)*V(vn);
V(Vout_n)<+voffs-V(vos)-V(voc)*V(vn);

$bound_step(0.1/fo);
end

endmodule

////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////
```

Behavioral Up/Down Counter

```
////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////

`include "constants.vams"
`include "disciplines.vams"

module logic_UD_cntr(vclku,vclkd,vq);
input vclku,vclkd;
output vq;
electrical vclku,vclkd,vq;

parameter real vtrans=0.5;
parameter real tdelay=50p;
parameter real trise=50p;
parameter real tfall=50p;
parameter integer nbit=8 from [1:inf];


real q,qmx;

analog begin

@(initial_step)
begin
q = 0.0;
qmx = pow(2,nbit);
end;

@(cross (V(vclku)-vtrans,+1)) q=q+1;
@(cross (V(vclkd)-vtrans,+1)) q=q-1;

if (q>=qmx)
q=qmx;

if (q<=0.0)
q=0.0;
```

```
V(vq)<+ transition(q,tdelay,trise,tfall);

end

endmodule

////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////
```

Behavioral ADC

```
////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////

`include "constants.vams"
`include "disciplines.vams"

module logic_ADC(vin,vout,voutb);


parameter real fullscale = 256;
parameter real vhigh = 1.2;
parameter real vthresh = vhigh/2;

input vin;
output [0:7] vout;
output [0:7] voutb;
electrical vin;
electrical [0:7] vout;
electrical [0:7] voutb;

real sample, midpoint;
integer i;
integer result[0:7];


analog begin

sample = V(vin);
midpoint =fullscale/2;

for (i=7;i>=0;i=i-1) begin

if (sample >= midpoint) begin
result[i]=vhigh;
sample=sample-midpoint;
end else begin
result[i]=0;
end
sample = 2*sample;

end

V(vout[0]) <+ transition(result[0],1e-6,1e-6,1e-6);
V(vout[1]) <+ transition(result[1],1e-6,1e-6,1e-6);
V(vout[2]) <+ transition(result[2],1e-6,1e-6,1e-6);
V(vout[3]) <+ transition(result[3],1e-6,1e-6,1e-6);
V(vout[4]) <+ transition(result[4],1e-6,1e-6,1e-6);
```

```
V(vout[5]) <+ transition(result[5],1e-6,1e-6,1e-6);
V(vout[6]) <+ transition(result[6],1e-6,1e-6,1e-6);
V(vout[7]) <+ transition(result[7],1e-6,1e-6,1e-6);

V(voutb[0]) <+ transition((vhigh - result[0]),1e-6,1e-6,1e-6);
V(voutb[1]) <+ transition((vhigh - result[1]),1e-6,1e-6,1e-6);
V(voutb[2]) <+ transition((vhigh - result[2]),1e-6,1e-6,1e-6);
V(voutb[3]) <+ transition((vhigh - result[3]),1e-6,1e-6,1e-6);
V(voutb[4]) <+ transition((vhigh - result[4]),1e-6,1e-6,1e-6);
V(voutb[5]) <+ transition((vhigh - result[5]),1e-6,1e-6,1e-6);
V(voutb[6]) <+ transition((vhigh - result[6]),1e-6,1e-6,1e-6);
V(voutb[7]) <+ transition((vhigh - result[7]),1e-6,1e-6,1e-6);


end

endmodule

////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////
```

Behavioral Binary-2-Thermometer

```
////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////

`include "constants.vams"
`include "disciplines.vams"

module logic_bin2therm_8bit(vin,vo);
input [0:7] vin;
output [0:255] vo;
electrical [0:7] vin;
electrical [0:255] vo;

parameter real vhigh = 1.0;
parameter real vlow = 0.0;
parameter real vtrans = 0.5;
parameter real tdelay=50p;
parameter real trise=50p;
parameter real tfall=50p;


integer ix,in,iz;
real vip[0:7];
real vx[0:255];
//electrical vq2;

analog begin

vip[0]=V(vin[0]) > vtrans;
vip[1]=V(vin[1]) > vtrans;
vip[2]=V(vin[2]) > vtrans;
vip[3]=V(vin[3]) > vtrans;
vip[4]=V(vin[4]) > vtrans;
vip[5]=V(vin[5]) > vtrans;
vip[6]=V(vin[6]) > vtrans;
vip[7]=V(vin[7]) > vtrans;
```

```
ix = vip[0] + 2*vip[1] + 4*vip[2] + 8*vip[3] + 16*vip[4]
    + 32*vip[5] + 64*vip[6] + 128*vip[7];

if (ix >= 255)
ix = 255;

if (ix <= 0)
ix = 0;

for  (in=0;in<ix;in=in+1)
vx[in] = vhigh;

for (iz=255; iz>=ix; iz=iz-1)
vx[iz] = vlow;

V(vo[  0]) <+ transition(vx[  0],tdelay,trise,tfall);
V(vo[  1]) <+ transition(vx[  1],tdelay,trise,tfall);
V(vo[  2]) <+ transition(vx[  2],tdelay,trise,tfall);
V(vo[  3]) <+ transition(vx[  3],tdelay,trise,tfall);
V(vo[  4]) <+ transition(vx[  4],tdelay,trise,tfall);
V(vo[  5]) <+ transition(vx[  5],tdelay,trise,tfall);
V(vo[  6]) <+ transition(vx[  6],tdelay,trise,tfall);
V(vo[  7]) <+ transition(vx[  7],tdelay,trise,tfall);
V(vo[  8]) <+ transition(vx[  8],tdelay,trise,tfall);
V(vo[  9]) <+ transition(vx[  9],tdelay,trise,tfall);
V(vo[ 10]) <+ transition(vx[ 10],tdelay,trise,tfall);

// ...... CAREFUL !!!!!!!!!!!!!!!!!!!!!!!!!!
// ...... 242 lines were removed from the code
// ...... CAREFUL !!!!!!!!!!!!!!!!!!!!!!!!!!

V(vo[253]) <+ transition(vx[253],tdelay,trise,tfall);
V(vo[254]) <+ transition(vx[254],tdelay,trise,tfall);
V(vo[255]) <+ transition(vx[255],tdelay,trise,tfall);


////////////////////////////////////////////////////////

end

endmodule

////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////
```

Behavioral analog multiplier

```
////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////

`include "constants.vams"
`include "disciplines.vams"

module analog_mixer(vi1, vi2, vout);

input vi1, vi2; electrical vi1, vi2;
output vout; electrical vout;

parameter real gain=1;
```

```
real vx, vy, vob;

analog begin

vx =  V(vi1);

vy =  V(vi2);

V(vout) <+ gain * vx * vy;

end

endmodule

////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////
```

Behavioral frequency meter (used to measure frequency in simulation

```
////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////

`include "constants.vams"
`include "disciplines.vams"

module freq_mtr(vinp,vinn,v_fout);
electrical vinp,vinn, v_fout;
parameter real vthr=0;
parameter real scale=1;

integer armed,arm1;
real t0,t1,t2;
real fout;

analog begin
@(initial_step)
t0=1p;

t1=t0;

t0=last_crossing(V(vinp)-V(vinn)-vthr,1);

@ ( cross (V(vinp)-V(vinn)-vthr,1) )



fout = 1/(t0-t1);


V(v_fout) <+ fout/scale;
end
endmodule

////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////
```

References

[1] Mead, C.A. “Analog VLSI and Neural Systems,” (1989), Addison-Wesley.

[2] Yu, T. & Cauwenberghs, G. “Analog VLSI Neuromorphic Network with Programmable Membrane Channel Kinetics,” Proc. IEEE Int. Symp. Circuits and Systems, Taipei Taiwan, (2009).

[3] Kyung, M., Weiss, E., Rangan, V. & Cauwenberghs, G. “Analog VLSI Simple Model Neuron with Wide Ranging Complex Dynamics,” Proc. Joint Symp. on Neural Computation, Los Angeles, CA, (2009).

[4] Morris, C. & Lecar, H. “Voltage oscillations in the barnacle giant muscle fiber,” Biophys. J. 35, (1981), 193-213.

[5] Rabaey J., Chandrakasan A., & Nikolic B. “Digital Integrated Circuits,” (2003), Prentice-Hall.