# Simulation and Modeling of Algorithm-Level Analog Computation

Jennifer Hasler, *Senior Member, IEEE*, and .....

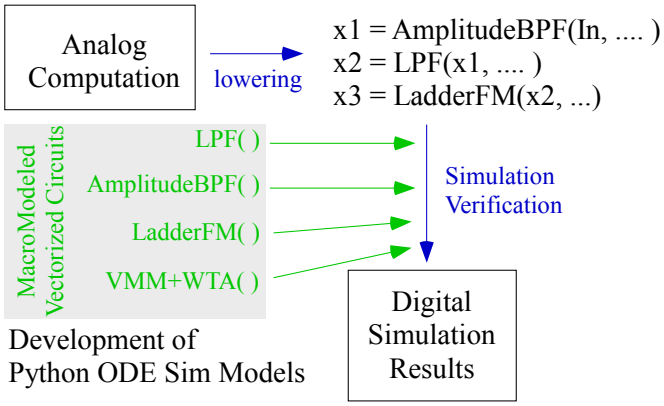


Fig. 1: When lowering analog computation into a set of algorithm blocks, a set of macromodelled blocks for the resulting vectorized circuits enables a digital simulation for system development and verification. This effort develops these macromodelled algorithm models for multiple ODE python simulation frameworks.

***Abstract*—Abstract**

This effort focuses on developing fast analog simulation techniques to enable analog simulation resulting from High-Level Synthesis (HLS) of analog computation (Fig. 1). This simulation tool is optimized for analog system design that still reasonably predicts experimental results. This analog system simulation tool builds on the numerical simulation simplifications developed for a range of digital simulations.

## I. Algorithm Models

These algorithm models arise from understanding of FG-enabled device synthesis. Each of these algorithms (Fig. 2) fit a dataflow approach (level=1 [1]). In addition to these *signal* values, there are a number of parameter values for each function. These blocks could be directly lowered to a highest-level Verilog-AMS with the same names. The blocks must enable vectorized (bus) inputs and outputs where appropriate for the computational model.

## II. Core Circuit Model Algorithms

The following subsections work through specific core circuit models in the algorithm functions, starting with the initial block diagram for a transconductance amplifier (Fig. II-A). The core circuit blocks (Fig. 3) include Low-Pass Filter (Sec. II-B), Amplitude Detector (Sec. II-C), Bandpass Filter (Sec. II-D) a core lowpass and bandpass second-order section (Sec. II-E) and a ladder filter stage (Sec. II-F. Each of the formulations uses its own input and output voltages that would be used in a particular instance. The resulting ODE simulation environment must take care of these multiple instances for the larger simulation environment.

Jennifer Hasler is with the School of Electrical and Computer Engineering (ECE), Georgia Institute of Technology, Atlanta, GA 30332-250 USA (e-mail:jennifer.hasler@ece.gatech.edu).

### A. Important FG Block: Transconductance amplifier

Transconductance Amplifiers (TA) come in two forms, both with FG current sources setting the bias current. The non-FG input model is

$$I = I_{bias} \tanh\left(\frac{\kappa V}{2U_T}\right),$$

and the FG input model is

$$I = I_{bias} \tanh\left(\frac{V}{V_L}\right),$$

where $V_L$ is proportional to 2 $U_T$ / $\kappa$ and proportional to inverse of the input capacitive division into the TA inputs ($C_T/C$). $U_T$ is the thermal voltage, $kT/q$ that is 25.8mV at room temperature, and $\kappa$ is the capacitive coupling from ($V_{fg}$ to the surface potential) TA elements are used throughout the core circuits as well as many of the larger system definitions.

### B. Low-Pass filter

A first-order low-frequency unity gain Low-Pass Filter (LPF) built from a non-FG input TA

$$C_L \frac{d_{out}}{dt} = I_{bias} \tanh\left(\frac{\kappa(V_{in} - V_{out})}{U_T}\right). \quad (1)$$

A FG input TA is similar

$$C_L \frac{d_{out}}{dt} = I_{bias} \tanh\left(\frac{V_{in} - V_{out}}{V_L}\right). \quad (2)$$

This formulation, like several of the This system shows the typical circuit parameters one would obtain with this formulation:

- $U_T$ is 25.8mV (room temperature).
- $\kappa = 0.7$ (process dependent, technically $\kappa_p$)
- $C_L$ = load capacitance that would include routing capacitance (default = 100fF).
- $I_{bias}$ = programed FG bias current

One could visualize an alternate size of functional parameters (non-FG input), parameters more typical of a computational or signal processing environment, as

- Voltage gain, $A_v$ (low frequency) = 1 (fixed).
- Corner frequency, $f_c = \frac{1}{2\pi\tau} = \frac{\kappa I_{bias}}{2\pi C_L}$.

| Function Area | Function | Input | Output | System Parameters | Circuit Parameters |
|---|---|---|---|---|---|
| **Lowpass** | LPF | n | n | cornerFreq(n) | Ibias(n) |
| **Filters:** | HopfLPF | 1 | n | centerFreq(n), Q(n) | Ibias1(n), Ibias2(n) |
| | LadderF | 1 | n | delay(n) | Ibias1(n), Ibias2(n) |
| | LadderFM | m | n×m | delay(n) | Ibias1(n), Ibias2(n) |
| **Bandpass**: | BPF | 1 | n | centerFreq(n), Q(n) | Ibias1(n), Ibias2(n) |
| **Filters**: | AdaptBPF | 1 | n | centerFreq(n), Q(n) | Ibias(n), Iq(n) |
| | AmplitudeBPF | 1 | n | centerFreq(n), Q(n), AmpDetect(n) | Ibias1(n), Ibias2(n), IbiasA(n) |
| | AmplitudeAdaptBPF | 1 | n | centerFreq(n), Q(n), AmpDetect(n) | Ibias(n), Iq(n), IbiasA(n) |
| **VMM**: | VMM | m | n | W(m×n), freq(1) | Ibias(n×m) |
| **blocks** | VMMWTA | m | n | W(m×n), Thresh(n), freq(1) | Ibias(n×m), Ithresh(n) |
| | SigVMM | m | n | W(m×n), freq(1) | Ibias(n×m) |
| | VMMDemod | m+1 | n | W(m×n), mod(1), freq1(1), freq2(1) | Ibias(n×m), Ibias2(n) |
| **Dynamics** | LinearEqIterate | n | n | A(n×n), b(n), freq(1) | IbiasA(n×n), IbiasB(n) |
| **Blocks** | NeuronHH | m | n | W(m×n) | Isyn(m×n) |
| **2-D Blocks** | LocalSepImagerConv | 1 | m | vert=V1, horiz=h1,clock | Ivert, Ihoriz, clock |
| | ClassifyBlockImage2Vector | m | n | clock, | clock |
| **Utilities** | Scanner | m+2 | 1 | clock(1), data(1) | |
| | M2V | m×n | m n | | |
| | MergeV | m , n | m+n | | |

Fig. 2: A current list of Algorithm blocks and some of their Python definition structure.

Specifying $f_c$ leaves us with a ratio of $I_{bias}$ / $C_L$; therefore, given the $C_L$ expected or calculated from a physical implementation, $I_{bias}$ is directly calculated. FG-input TA has a similar formulation for $f_c = \frac{I_{bias}}{2\pi V_L C_L}$, where the $V_L$ is given or designed, resulting in a similar ratio of $I_{bias}$ / $C_L$.

One often will see analytic modeling for considering the nonlinear behavior of this or other FG-enabled circuits where the core variables are normalized, the core variables that set many of the important parameters. For this first-order LPF,

$$x = \frac{\kappa}{2U_T}, y = \frac{\kappa}{2U_T}, \tau = \frac{2C_L U_T}{\kappa I_{bias}},$$

resulting in the expression if time is renormalized at $t_1 = t/\tau$ to a dimensional-less quantity as

$$\frac{dy}{dt} = \tanh(x - y)$$

These formulations are essential for many analytic circuit formulations, although the normalizations are not used for the numerical modeling. Often one must remember to *unnormalize* an elegant analytic formulation to make its macromodelled numerical model. Often the normalizations do provide useful intermediate variables in the ODE calculations.

### C. Amplitude Detector

An amplitude detector circuit is a modification of the LPF to have an even-nonlinearity component that will generate the even-order harmonics to follow either the minimum or the maximum of the input signal. A classical discrete circuit might use a diode and a capacitor. An on-chip with the same dynamics typically uses a source follower circuit; the circuit does not require any input current draw. Further, an amplifier is often used in the feedback for rectifying behavior to be smaller than $U_T$, but rather the amplifier gain ($A_V$) smaller than $U_T$ (rectifying behavior at $U_T/A_V$ amplitudes. We will model this concept as a first-order block as the dynamics of the amplifier are typically secondary to the dynamics of the rectifying source follower and the capacitor at $V_1$ is small. The amplifier is described by

$$V_1 = A_v(V_{in} - V_{out}).$$

When coupled to the source-follower dynamics,

$$C_L \frac{d_{out}}{dt} + I_{thp} e^{(\kappa_p(V_{dd} - V_1 - V_{T0}) - (V_{dd} - V_{out}))/U_T} = I_{bias},$$

one gets the amplitude detection (in this case amplitude minimum detection) as

$$C_L \frac{d_{out}}{dt} = I_{bias} \left(1 - e^{(V_{out}(1+1/A_v) - V_{in})/(U_T/\kappa_p A_v)}\right) \quad (3)$$

At very low-frequencies, the output ($V_{out}$) is nearly equal to the input ($V_{in}$). Parameters (other than $\kappa_p$ and $U_T$) are the load capacitance ($C_L$, default = 100fF) and the bias current ($I_{bias}$), as well as the amplifier gain $A_v$ (default of 200, typical uncascoded amplifier gain). The ratio of $C_L$ and $I_{bias}$ sets the time constant for the amplitude detection; if $C_L$ changes, then $I_{bias}$ just needs to change the same amount for identical operation.

### D. Bandpass Filter Model

The $C^4$ Bandpass Filter(BPF) topology ($C^4$ = Capacitively Coupled Current Conveyer) [2], [3] provides a tunable BPF both through programmable bias currents and capacitances (drawn and parasitic). The $C^4$ simulation model [3] starts with writing the circuit equations (Fig. 3)

$$I_1 = I_{bias} \tanh\left(-\frac{\kappa(V_1}{2U_T}\right)$$

$$I_2 = I_{fb} \tanh\left(\frac{V_{out} - V_1}{V_L}\right)$$

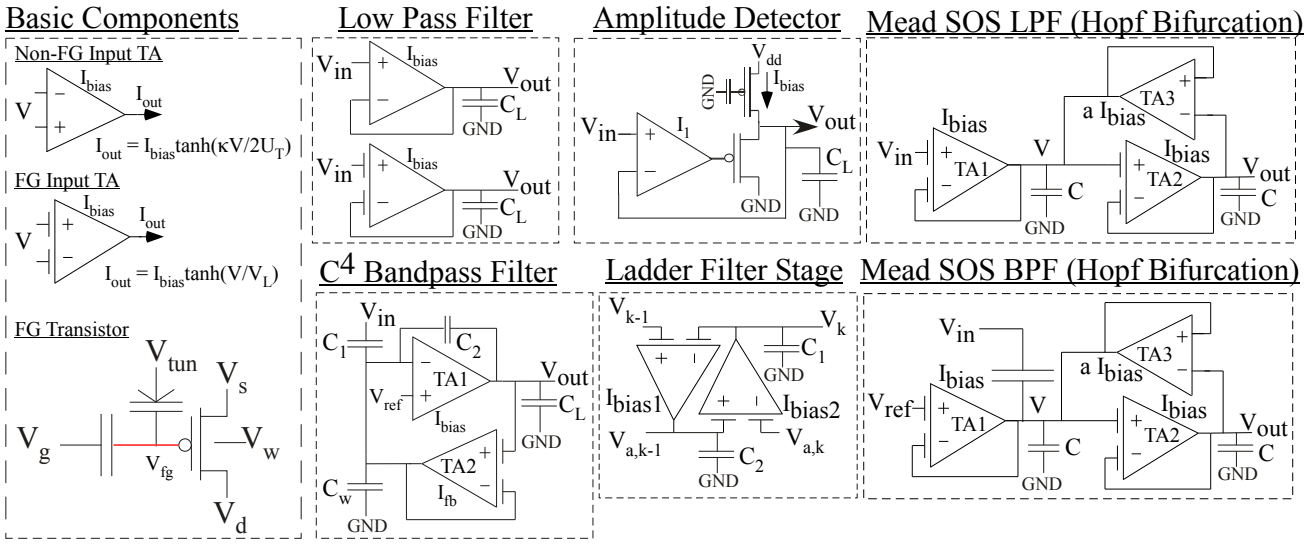


Fig. 3: Algorithmic blocks that arise from simple circuit architectures. Basic circuit components include FG transistors as well as Transconductance Amplifiers (TA) that use both FG and non-FG inputs. *Low-Pass Filter (LPF)*: The circuit uses one TA with an FG current source and explicit or parasitic (e.g. routing) capacitance. *Amplitude Detector*: The source follower creates the rectifying nonlinearity, and the amplifier increases the sensitivity to signals larger than $U_T/A_V$. *Bandpass Filter (BPF)*: The $C^4$ topology gives a range of tunable bandpass topologies. *Mead SOS*: LPF and BPF forms enabling Hopf Bifurcations. *Ladder Filter Stage*: One stage of a second-order prototype emulating an LC tap.

$$(C_L + C_2)\frac{d_{out}}{dt} = C_2\frac{d_1}{dt} + I_1$$
$$(C_1 + C_2 + C_w)\frac{d_1}{dt} = C_1\frac{d_{in}}{dt} + C_2\frac{d_{out}}{dt} + I_2 \qquad (4)$$

Using Gaussian elimination to only have state variable derivatives on the left-hand side yields

$$C_{eq}\frac{dV_{out}}{dt} = C_1\frac{d_{in}}{dt} + \frac{C_1 + C_2 + C_w}{C_2}I_1 + I_2$$
$$C_{eq}\frac{dV_1}{dt} = C_1\frac{C_L + C_2}{C_2}\frac{d_{in}}{dt} + I_2\frac{C_L + C_2}{C_2} + I_1 \qquad (5)$$

where $C_{eq} = \frac{(C_1+C_2+C_w)(C_L+C_2)-C_2^2}{C_2}$ At this stage, we have an issue to handle the input derivative signal that helps create the bandpass signal response. Each ODE system has its own unique solution. The solution for this equation system is using a variable substitution for the state variables

$$V_{1,a} = V_1 - \frac{C_1}{C_{eq}}\frac{C_L + C_2}{C_2}V_{in}$$
$$V_{out,a} = V_{out} - \frac{C_1}{C_{eq}}$$
$$I_1 = I_{bias}\tanh\left(-\frac{\kappa(V_{1,a} + \frac{C_1}{C_{eq}}\frac{C_L+C_2}{C_2}V_{in}}{2U_T}\right)$$
$$I_2 = I_{fb}\tanh\left(\frac{V_{out,a} - V_{1,a} - \frac{C_1}{C_{eq}}\frac{C_L}{C_2}V_{in}}{V_L}\right) \qquad (6)$$

The ODE model becomes

$$C_{eq}\frac{dV_{out,a}}{dt} = \frac{C_1 + C_2 + C_w}{C_2}I_1 + I_2$$
$$C_{eq}\frac{dV_1}{dt} = I_2\frac{C_L + C_2}{C_2} + I_1 \qquad (7)$$

The output voltage requires transformation,

$$V_{out} = V_{out,a} + \frac{C_1}{C_{eq}}$$

at the end of each timestep as the actual computation output. The computation has a number of parameters (drawn or parasitic)

- Two currents, $I_{bias}$ and $I_{fb}$
- Four capacitors: $C_1$, $C_2$, $C_w$, $C_L$
- $V_L$ (TA2), $\kappa_p$, and $U_T$.

A shift in the four capacitors can be compensated by the two bias currents.

### *E. LPF and BPF Mead SOS: Hopf Bifurcation Structure*

Starting from the initial concept of a positive feedback TA second-order section [4], a FG-enabled second-order LPF [5] and BPF [6] circuits. Using two FG input TA that have a large input linear range ($V_L$, measured $\approx$ 700mV in, [5], [6]) allows focusing the nonlinear range on the positive-feedback TA. The LPF three-TA circuit can be modeled as

$$\tau^2\frac{d^2V_{out}}{dt^2} + 2\tau\frac{dV_{out}}{dt}$$
$$-b\tanh\left(\tau\frac{\kappa}{2U_T}\frac{dV_{out}}{dt}\right) + V_{out} = V_{in}, \qquad (8)$$

where $\tau = CV_L/I_{bias}$ for the programmed $I_{bias}$, and $b = a\frac{\kappa V_L}{2U_T} = a\frac{C_T}{C_1}$, proportional to the current percentage (a) of the positive feedback TA to the other TAs. One can build this system as a system of first-order ODEs as a step from the original derivation [5],

$$\tau\frac{dV_{out}}{dt} + V_{out} = V,$$
$$\tau\frac{dV}{dt} + V = V_{in} + aV_L\tanh\left(\frac{\kappa(V - V_{out})}{2U_T}\right); \qquad (9)$$

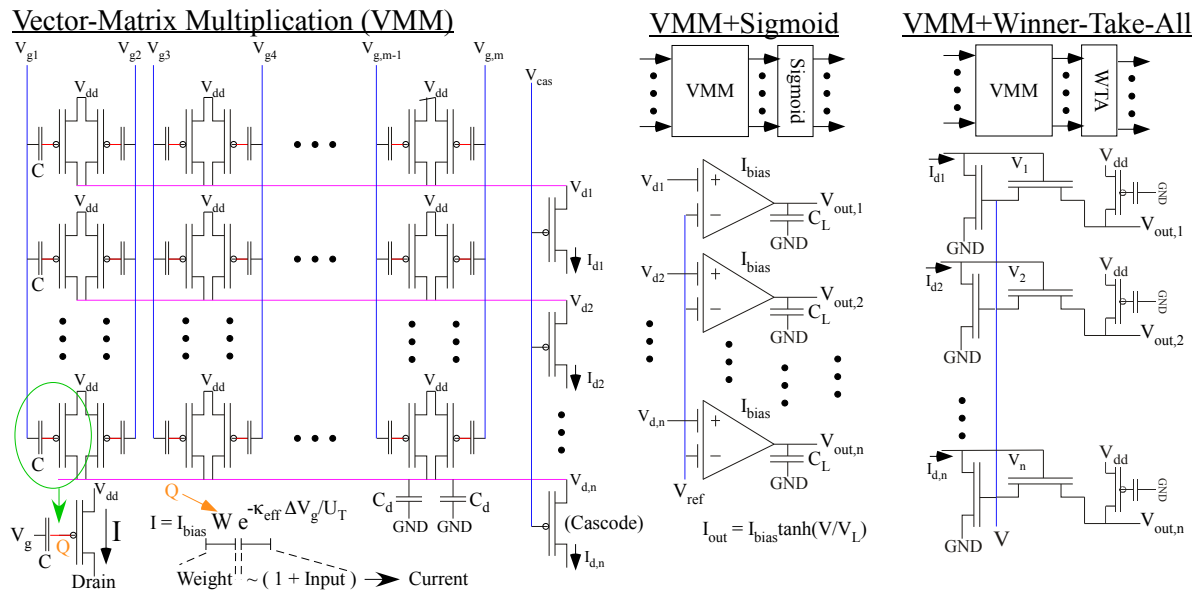


Fig. 4: Vector-Matrix Multiplication (VMM) and some blocks (Sigmoid, Winner-Take-All (WTA) ) for a consistent dataflow block. The VMM+WTA block has a constraining current source at V.

two first-order LPF formulations should be more numerically stable for the resulting simulation. This block requires a parallel bank with vectorized inputs and outputs. The BPF three-TA circuit modifies the LPF circuit by capacitively coupling into V rather than an input through TA1. This circuit is modelled as [6] with its two equation formulation as

$$\tau\sqrt{\delta}\frac{dV}{dt}+V = V_{bias}+aV_L\tanh\left(\frac{\kappa(V-V_{out})}{2U_T}\right)+\alpha\sqrt{\delta}\tau\frac{dV_{in}}{dt}$$

$$\frac{\tau}{\sqrt{\delta}}\frac{dV_{out}}{dt}+V_{out}=V, \tag{10}$$

where $C_T$ is the total capacitance at V, is C + $C_{in}$, $\delta = C_T/C$ (models unequal TA1 and TA2 load capacitances), $\tau = \sqrt{\delta}CV_L/I_{bias}$, and $\alpha$ = $C_{in}$ / $C_T$. This set of equations has the same issue for an ODE simulation as the $C^4$ amplifier in that the input signal is a derivative[1].

### F. Ladder Filter Algorithm Block

A ladder filter array is a TA & capacitor programmable emulation for a L & C network, a network that can be configured towards a delay approximation [7]. The ladder filter tap is (Fig. 3)

$$C\frac{dV_{a,k}}{dt} = I_{bias1,k}\tanh((V_{k-1}-V_k)/V_L)$$
$$C\frac{dV_k}{dt} = I_{bias2,k}\tanh((V_{a,k-1}-V_{a,k})/V_L) \tag{11}$$

The input for the first tap is $V_1 = V_{in}, V_{a,1} disconnected$ to drive the LC network, and the last tap (position n) would be

$$C\frac{dV_{a,n}}{dt} = I_{bias1,n}\tanh((V_{n-1}-V_n)/V_L)$$
$$C\frac{dV_n}{dt} = I_{bias,n}\tanh((V_{a,n-1}-V_n)V_L) \tag{12}$$

[1]Not an obvious solution at this stage; thoughts welcome

effectively similar to an optimal resistive termination.

## III. VMM Circuit Model Algorithms

A number of the algorithm blocks involve Vector-Matrix Multiplication (VMM) computations (fig:AlgorithmBlocks). A VMM computation has a vector of voltage inputs going into, and a vector of current outputs leaving from, a mesh array of stored weight values. A dataflow compatible block would require voltage outputs [1], and therefore any VMM block requires the addition of the next-stage of computation for a dataflow compatible block (Fig. 4).

### A. Core VMM Modeling

A FG device has a free parameter that is a linear shift on the threshold voltage ($V_{T0}$) effectively through a shift on the flatband voltage ($V_{fb}$). For subthreshold currents that have an exponential current resulting from the gate & source voltage, a linear shift in the gate voltage results in a multiplicative factor, effectively a multiplicative weight the transistor operation. A FG crossbar array is capable of computing a Vector-Matrix Multiplication (VMM) because a single transistor can output a current that is the product of an input signal and a representation of the stored FG charge on the device (Fig. 4). The FG node is a capacitive voltage divider from the gate ($V_g$) terminal to the FG ($V_{fg}$) terminal that can be modeled as

$$V_{fg} = \frac{C}{C_T}V_g + V_Q \tag{13}$$

where $C_T$ is the total capacitance at the floating-gate node, and $V_Q$ is due to charge at the floating-gate node. The saturated subthreshold pFET transistor channel current is

$$I = I_{th}e^{\kappa(V_{dd}-V_{fg}-V_{T0})/U_T} \tag{14}$$

for the source and well voltage at $V_{dd}$, . Choosing a useful FG bias point, $V_{fg0}$, & the variation around it $\bar{V}_{fg}$ ($V_{fg}$ =

$V_{fg0} + \bar{V}_{fg}$) with a corresponding gate bias point, $V_{g0}$, & the variation around it $\Delta V_g$ ($V_g = V_{g0} + \Delta V_g$), results in a known bias current ($I_{bias}$) for $V_{fg0}$,

$$I_{bias} = I_{th} e^{\kappa(V_{dd} - \bar{V}_{fg} - V_{T0})/U_T}. \quad (15)$$

Using this formulation, (14) becomes

$$I = I_{bias} W e^{-\kappa_{eff} \Delta V_g / U_T} \quad (16)$$

where $W = e^{-\kappa \hat{V}_{fg}/U_T}$ the effective $\kappa$ is $\kappa_{eff} = \kappa \frac{C}{C_T}$ with a corresponding linear input range of $\frac{U_T}{\kappa}\frac{C}{C_T}$. This formulation results in a multiplication of the stored weight (W) value with a form of the input signal ($\Delta V_g$).

A VMM is created from a two-dimensional array of FG-transistors through a mesh architecture that broadcasts the gate n inputs along columns and sums the m output currents along the rows (Fig. 4). The output current for the l-th row, where k goes from 1 to n would be

$$I_{d,l} = I_{bias} \sum_{k=1}^{m} W_{l,k} e^{-\kappa_{eff}(V_{g,k} - V_{g0})/U_T}$$

$$I_{d,l} = I_{bias} \sum_{k=1}^{m} W_{l,k} e^{-\kappa_{eff} \Delta V_{g,k}/U_T} \quad (17)$$

In practical terms, the average value of the programmed currents, $I_{bias}$ at the bias gate voltage for all of the inputs ($V_{g0}$) sets the important parameters, and the weights are the variation of each bias current from $I_{bias}$. $\kappa_{eff}$ is the effective $\kappa$ for all of the devices. A similar formulation can be made with inputs to the source terminals instead of the gate terminals (seen in [8]), with a similar formulation that requires sourcing the current for the computation by the input terminals.

If the inputs go into a cascoded transistor element or equivalent source node, the resulting output voltage of the cascode is

$$I_{bias} e^{(V_{d,l} - V_{d0})/U_T} = I_{bias} \sum_{k=1}^{m} W_{l,k} e^{-\kappa_{eff}(V_{g,k} - V_{g0})/U_T},$$

$$e^{\Delta V_{d,l}/U_T} = \sum_{k=1}^{m} W_{l,k} e^{-\kappa_{eff} \Delta V_{g,k}/U_T} \quad (18)$$

where $V_{d0}$ is the bias drain voltage from the computation that would include the cascode gate bias voltage ($V_{cas}$). The output currents are approximately the same as the summed currents on the line if the line is held at a nearly fixed potential; using a cascode greatly decreases any affect of the output voltage movements changes on the VMM operation. This formulation requires including the capacitors, primarily parasitic capacitances ($C_d$), along each input line

$$mC_d \frac{dV_{d,l}}{dt} + I_{bias} e^{(V_{d,l} - V_{d0})/U_T} =$$

$$I_{bias} \sum_{k=1}^{m} W_{l,k} e^{-\kappa_{eff}(V_{g,k} - V_{g0})/U_T},$$

$$\tau \frac{dV_{d,l}}{dt} + U_T e^{(V_{d,l} - V_{d0})/U_T} =$$

$$U_T \sum_{k=1}^{m} W_{l,k} e^{-\kappa_{eff}(V_{g,k} - V_{g0})/U_T}, \quad (19)$$

where $\tau = mC_d U_T / I_{bias}$. One can further model the column dynamcis and time constant assuming the transconductance of a buffer driver ($g_m$, that for a subthreshold buffer amplifier would be $\kappa I_{buf}/U_T$) as

$$nC \frac{dV_{g,k}}{dt} = g_m V_{g,k}, \tau_{col} \frac{dV_{g,k}}{dt} = V_{g,k} \quad (20)$$

where $\tau_{col} = nCU_T/\kappa I_{buf}$. This modeling closely follows experimental measurements (e.g. [9]) across multiple IC process nodes.

### *B. VMM + Sigmoid Block*

Although a VMM block could a dataflow (level=1) block by outputing $V_{d,1}$ to $V_{d,n}$ (or more likely buffering the outputs), or putting the cascode currents through a transimpedance amplifier to convert the currents to a voltage, multiple additional components can be added to the VMM blocks for a range of implementations (e.g. Neurons, NN blocks, and related structures) resulting in an efficient dataflow block. One example would be creating one form of a sigmoid function; a range of creativity is possible along these lines. An FG TA, that could be built or programmed to a wide range of linearities and offset voltages, results in a range of potential sigmoid functions.

This approach requires modeling output voltage $V_{out}$ due to the output current of the FG-input TA that when all devices are in saturation yields

$$C_L \frac{dV_{out,k}}{dt} = I_{bias} \left( \tanh\left( \frac{V_{d,k} - V_{ref} + V_Q}{V_L} \right) + \frac{V_{out,k}}{V_A} \right). \quad (21)$$

where $V_A$ is the effective early voltage of the output current source transistors. The steady state of (23) shows the

$$V_{out,k} = V_A \tanh\left( \frac{(V_{d,k} - V_{ref1})}{V_L} \right). \quad (22)$$

where $V_{ref1} = V_{ref} - V_Q$, and the input linear range ($V_L$) can be tuned for the desired sigmoidal range for a given process node. The function gives a sigmoid with a programmable offset and a gain at the center of the sigmoid of $V_A/V_L$. Typically some gain is useful as the signal size from the cascode voltages is on the order of $U_T$, so assuming that $V_{d,k} = (VMM)U_T$, the center gain (dimensional-less to V) would be $V_A(U_T/V_L) = (V_A/\kappa)(C/C_T)$. This gain can become sufficiently high that the output reaches the output voltage supplies and bounded by those supplies, although this can be controlled, if desired, by adjusting these particular parameters. One can also get capacitive coupling to the FG node, although if balanced, the differential structure eliminates the effect. Using capacitance in feedback to the negative terminal to modify the gain

$$C_L \frac{dV_{out,k}}{dt} = I_{bias} \tanh\left( \frac{V_{d,k} - V_{ref1} + A_v V_{out}}{V_L} \right), \quad (23)$$

and a more linear response without sigmoid behavior until reaching the power supply rails, where $A_v$ is the voltage

gain achieved by ratioing the capacitors. A ReLu function is effectively the same function but with a low gain (e.g. gain = 1 to the current...normalize) and without the upper voltage rail that can be created by either of these biasing techniques.

### C. VMM + WTA Block

Another important block is a VMM+WTA classifier [10] used in multiple embedded IC classifier / inference [8] as well as learning [11], [12] physical implementations where one simplification of this block leads to a softmax computation. The computation starts typically with the output cascode current vector ($I_{d,1}$ to $I_{d,n}$), where referring the computation to the output cascode voltage vector ($V_{d,1}$ to $V_{d,n}$ ). One term is at the input that we would model as a common-gate configuration

I_in

$$C_L \frac{dV_k}{dt} = I_{bias}\left(e^{\Delta V_{d,k}/U_T} - e^{\kappa_n \Delta V/U_T}(1 - e^{V_k/U_T})\right),$$

One term is the middle extended differential pair node (state variable) that primarily sees source conductances, and therefore has the fastest of the timeconstants. Assuming the dynamics node is not essential for the modeling, we get

$$V = U_T \ln\left(\sum_{l=1} e^{\kappa_n V_l}\right) - \kappa V_{bias} \approx \kappa_n(\max(V_k) - V_{bias}) \tag{24}$$

The last term would be a common-source type structure at the output with one current source being the normalized version of the input voltage,

$$C_L \frac{dV_{out,k}}{dt} = I_{prog,k} e^{\sigma(V_{dd} - V_{out,k})/U_T} - I_{bias} \frac{e^{\kappa V_k}}{\sum_{l=1} e^{\kappa V_l}} \tag{25}$$

covering most of the dynamics of this k-WTA circuit.

The computation can also use the cascode voltage nodes ($V_{d,1}$ to $V_{d,n}$ ) that can allow for a lower-gain VMM+WTA computation with a similar Common-Mode Rejection as well as a lower effect of mismatch. The current-input case has a threshold voltage mismatch that is dependent on the node winning or losing. The primary change is modifying the extended differential pair model,

$$V = U_T \ln\left(\sum_{l=1} e^{\kappa_n V_{d,l}}\right) - \kappa V_{bias} \approx \kappa_n(\max(V_{d,l}) - V_{bias}) \tag{26}$$

to achieve the desired dynamics. One can compensate for the lower gain by adding another output gain stage to the outputs. Lower gain in this form can have another gain stage on the output nodes, etc.

## IV. Next Models

Next models likely involve more neuromorphic concepts (e.g. HH neuron, channels, non-adapting synapses, etc.), although the models above are enough (I will double check a few things) to do the analog related models.

## References


[1] C. Schlottmann and J. Hasler, "High-level modeling of analog computational elements for signal processing applications," *IEEE Transactions VLSI*, vol. 22, no. 9, pp. 1945–1953, 2014.

[2] D. Graham, Hasler, R. Chawla, and P. Smith, "A low-power, programmable bandpass filter section for higher-order filter applications," *IEEE Transactions CAS I*, vol. 54, no. 6, pp. 1165–1176, 2007.

[3] M. Collins, J. . Hasler, and S. George, "An open-source toolset enabling analog–digital software codesign," *Journal of Low Power Electronics Applications*, vol. 6, no. 1, pp. 1–15, 2016.

[4] C. Mead, *Analog VLSI and Neural Systems*. Addison-Wesley, 1989.

[5] J. Hasler, "A programmable on-chip hopf bifurcation circuit," *IEEE Transactions on Circuits and Systems I*, vol. 69, no. 12, pp. 4958–4968, 2022.

[6] ——, "A programmable adaptive-q bpf circuit," *IEEE Transactions on Circuits and Systems I*, vol. 70, no. 10, pp. 3888–3898, 2023.

[7] J. Hasler and S. Shah, "An SoC FPAA based programmable, ladder-filter based, linear-phase analog filter," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 68, no. 2, pp. 592–602, 2021.

[8] J. Hasler, "Large-scale field programmable analog arrays," *Proceedings of the IEEE*, vol. 108, no. 8, pp. 1283–1302, 2020.

[9] C. Schlottmann and Hasler, "A highly dense, low power, programmable analog vector-matrix multiplier: the FPAA implementation," *IEEE Journal on Emerging CAS*, vol. 1, no. 3, pp. 403–411, 2011.

[10] S. Ramakrishnan and J. Hasler, "Vector-Matrix Multiply and Winner-Take-All as an Analog Classifier," *IEEE Transactions VLSI*, vol. 22, no. 2, pp. 353–361, 2014.

[11] J. Hasler and S. Shah, "SoC FPAA hardware implementation of a VMM+WTA embedded learning classifier," *IEEE Journal on Emerging CAS*, vol. 8, no. 1, pp. 28–37, 2018.

[12] S. Shah and J. Hasler, "VMM + WTA embedded classifiers learning algorithm implementable on SoC FPAA devices," *IEEE Journal on Emerging CAS*, vol. 8, no. 1, pp. 65–76, 2018.