ANALOG AND NEUROMORPHIC COMPUTING WITH A FRAMEWORK ON A RECONFIGURABLE PLATFORM

A Dissertation
Presented to
The Academic Faculty

By

Aishwarya Natarajan

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Electrical and Computer Engineering

Georgia Institute of Technology

December 2021

ANALOG AND NEUROMORPHIC COMPUTING WITH A FRAMEWORK ON A RECONFIGURABLE PLATFORM

Thesis committee:

Dr. Jennifer Hasler, Advisor
Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Aaron Lanterman
Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Omer Inan
Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Steven Baer
Department of Mathematics
Arizona State University

Dr. Azad Naeemi
Electrical and Computer Engineering
Georgia Institute of Technology

Date approved: December 9 2021

For Amma, Appa, Akka, Uday

ACKNOWLEDGMENTS

I would like to thank my PhD advisor Jennifer Hasler, for all the guidance and interactions we have had during the course of my graduate studies at Georgia Tech. I thank her for always believing in me and pushing me to achieve the best that I am capable of. Her invaluable technical knowledge and feedback have certainly been helpful throughout my PhD.

I would like to thank the members of my thesis committee for their insights and comments. I would like to thank Steve Baer on the knowledge imparted by him especially on the neuroscience aspects. I look forward to continue our collaborations on modeling aspects of neuromorphic systems. I would like to thank Aaron Lanterman for thoroughly reading my dissertation and giving me useful suggestions to edit the same. Further, I would also like to thank Omer Inan and Azad Naeemi for their feedback and comments during the process.

I would like to thank the members of our Integrated Computational Electronics Laboratory (ICE) lab, both the present and past members for the interactions and collaborations we have had over the years. I thank my friends who have helped me handle graduate school.

I would not be where I am today, without the constant support, motivation and encouragement from my family. They inspired me to pursue my studies and get into research. I would especially like to thank my sister and Uday for always supporting me patiently no matter what during all the times.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

The objective of the research is to demonstrate energy-efficient computing on a configurable platform, the Field Programmable Analog Array (FPAA), by leveraging analog strengths, along with a framework, to enable real-time systems on hardware. By taking inspiration from biology, fundamental blocks of neurons and synapses are built, understanding the computational advantages of such neural structures. To enable this computation and scale up from these modules, it is important to have an infrastructure that adapts by taking care of non-ideal effects like mismatches and variations, which commonly plague analog implementations. Programmability, through the presence of floating gates, helps to reduce these variations, thereby ultimately paving the path to take physical approaches to build larger systems in a holistic manner.

CHAPTER 1
ANALOG AND NEUROMORPHIC COMPUTING ON CONFIGURABLE SYSTEMS

1.1 Overview

Biologically inspired systems have revolutionized computing, since the concept was pioneered by Carver Mead, in the early 1980s. The human brain performs computation with just 20W of power comparable with supercomputers that consume megawatts of power. Neuromorphic computing is about translating these energy-efficient neural systems, as well as implementing competitive techniques, by understanding and exploiting the computational advantages of such structures. This can be accomplished by analog structures that remain true to the biophysics of neurons and perform compact low-power real-time computation.

However, in conventional analog processes, mismatch and variations may plague overall performance. Reconfigurability and programmability provided by floating gates helps solve these issues. Hence, it is significant to have a platform and framework that can support the real-time implementation of energy-efficient applications on hardware.

Keeping such goals in mind, this work aims towards high impact by implementing physical approaches of computing in configurable hardware. The analog implementations, facilitated through dense arrays of floating gates, can not only adapt by taking care of non-ideal effects like mismatches and variations, but also pave the path for power efficient computation. The framework built for such hardware, with abstractable tools and algorithms, enables circuit and system designers, as well as computational neuroscientists, to understand neural systems, especially through the fundamental blocks of Hodgkin-Huxley neurons and synapses that allow one to scale and build different topologies and larger sys-

tems in real-time, rather than through digital simulations. Hence, this work focuses on implementing such circuits and systems through Field Programmable Analog Arrays.

1.2 Programmable and Configurable Analog Systems

Reconfigurable hardware has numerous advantages. In the digital domain, Field Programmable Gate Arrays (FPGAs) have been useful for prototyping and implementing a number of applications. However, to break the energy efficiency wall, one needs robust analog approaches. Analog computing techniques result in 1000$\times$ improvement in power or energy efficiency, and a 100$\times$ improvement in area efficiency, compared to digital computation, as Mead originally predicted [1]. Field Programmable Analog Arrays (FPAA) [2], with their reconfigurability and programmability, enable the design and implementation of large-scale mixed-signal systems for diverse applications in signal processing and neuromorphic computing. The presence of a single platform facilitating a hardware-software codesign has been instrumental in implementing such applications [3]. Reconfigurability is obtained by modulating the charges on the Floating Gates (FG) on the FPAA structure. These devices are utilized as ubiquitous, small, dense, non-volatile memory structures and to perform computation throughout the SoC FPAA.

A Floating Gate circuit is one in which the gate of a FET is electrically isolated and is capacitively connected to the other nodes. In 1967 [4], FG transistors were first shown as an emerging technology that could be used for non-volatile memory storage. Charge is stored on the FG, which leads to the non-volatile behavior and it can be modulated through electron tunneling and hot-electron injection [5].

The hot-electron injection process injects charges on the floating node, thereby reducing the threshold voltage of the device. The process of Fowler-Nordheim tunneling is used by subjecting the MOS capacitor to a high voltage, to remove the electrons from the node. This voltage is increased to reduce the effective width of the barrier, which in turn increases the threshold voltage of the device. These techniques originated from the fundamental single

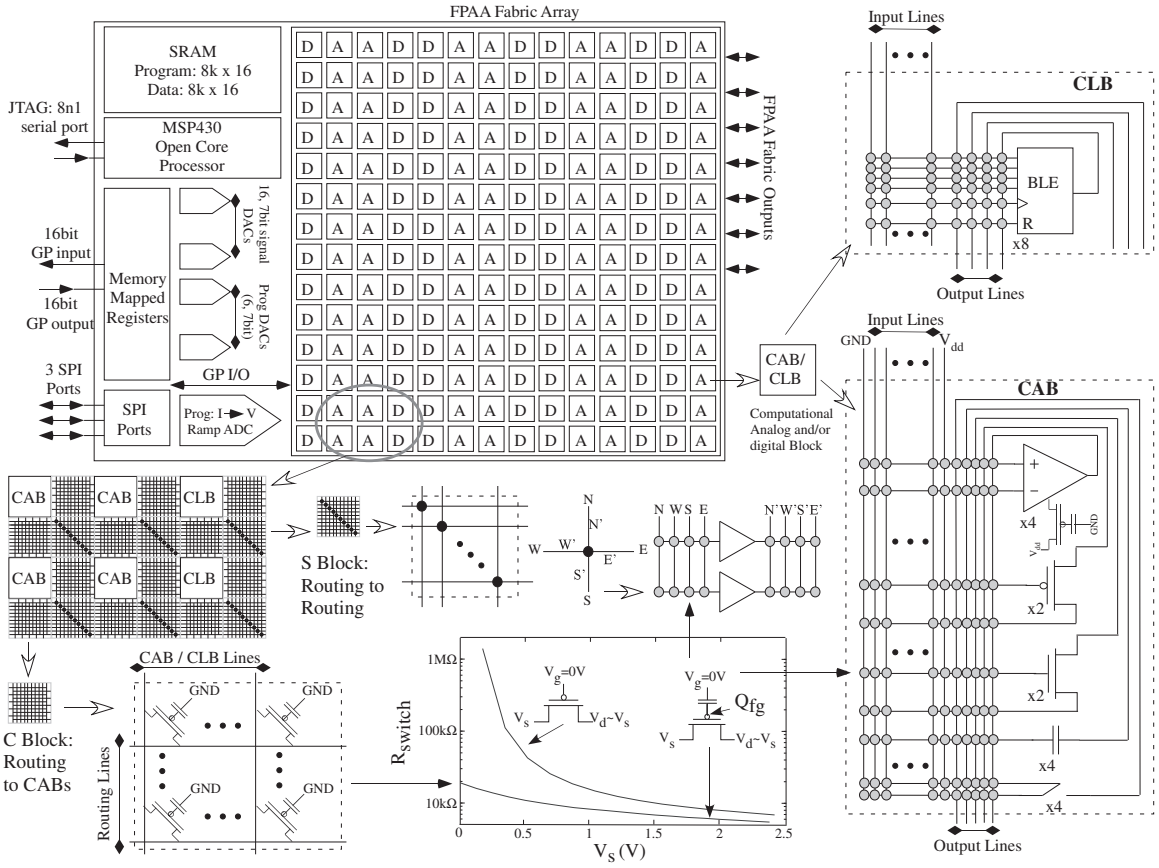


Figure 1.1: FPAA fabric consists of CABs, CLBs and other peripherals, fabricated on a 350nm process. The CAB components with the transistors, OTAs, FGOTAs, transmission gates and capacitor banks, with the routing lines consisting of the FG pFETs are shown in one CAB. The CLBs consist of eight, four input BLE lookup tables with a latch. The DACs, MSP 430 processor and other peripherals are used as a part of the infrastructure, all on the chip, to perform the programming of the FG elements.

transistor synapse learning device demonstrated in [6].

A concept of a CMOS FPAA for starting to look at promising neural network applications was first shown in [7]. The FPAAs developed have evolved through a number of generations [8, 9], all leading to the current generation [2]. The SOC FPAA [2] is a reconfigurable and programmable SOC consisting of Computational Analog blocks (CABs) and Computational Logic Blocks (CLBs), which are connected through Manhattan style routing that are composed of Connection (C) and Switch (S) blocks. It is integrated with a processor, ADCs, DACs and peripherals, as shown in Figure 1.1. Each CAB consists of nFETs, pFETs, Operational Transconductance Amplifier (OTAs), FGOTAs, transmission

gates and capacitor banks, while a CLB consists of lookup tables. There are 98 CABs and 98 CLBs. All the measurements shown throughout this work are taken from the SOC FPAA fabricated on a 350nm process.The routing switches, made of FG pFETs, are not dead weight; they may be used for computation, depending on the target application. The SoC FPAA interdigitates analog and digital computation in the same routing fabric.

Due to the presence of FGs, one can take care of mismatches and other variations that commonly plague analog implementations. Two approaches to programming [10] FPAAs have been followed through the various generations, namely the direct and indirect programming methodologies. In direct programming [5], t-gates switch a FG transistor between its application and the programming circuitry associated with it. Since this switching may decrease accuracy and speed, especially due to extra parasitics, an indirect programming [11] method is used currently, where the transistors in the circuit and the programming circuitry are different. These techniques help to eliminate most issues of threshold voltage mismatch and alleviates the need for large transistors for analog as opposed to using small transistors for digital operation to handle mismatch issues, especially as one scales analog and digital designs to smaller device nodes.

The process of compilation, programming, and taking experimental measurements is executed through an open source tool infrastructure in a Scilab/Xcos framework [12, 13]. FG programming is encapsulated in the infrastructure [14], abstracted away from the user. These tools give the user the ability to create, model, and simulate [15] analog and mixed-signal circuits and systems. They also allow multiple levels of abstraction [16], thereby being accessible to system-level designers, while still enabling circuit designers the freedom to build at a low level. They facilitate rapid development and reduce the barrier of entry for ultra-low power physical computing techniques for analog and neuromorphic applications.

1.3 Neuron models commonly used for Neuromorphic Computing

The fundamental building blocks for the design and implementation of a spiking network are neurons and synapses. A neuromorphic computing approach initially requires a biological understanding of these modules to help build them on configurable hardware. A biological neuron consists of the soma or cell body, axon, and dendrites. The neurons transmit the output through the axons, while the dendrites help receive the input and transmit the information to the soma. The information is communicated between the neurons (from a dendrite input to the axon output) through the synapses, enabled through the neurotransmitters at the gap junctions.

A key question while designing different networks in hardware is the choice of the required neuron model to be used. A number of factors need to be considered before arriving at a decision. One should consider how well the model replicates the spiking dynamics of a neuron, preserving neurocomputational features, and how that further leads to processing of information in the brain. Since the ultimate objective is to achieve a hardware implementation, the efficiency and speed of computation along with the power dissipated and area should also be kept in mind. This becomes more apparent when one instantiates multiple copies of these neuron models in a network for useful computation.

1.3.1 Hodgkin Huxley Neuron

Hodgkin and Huxley (HH) won the Nobel prize for their work [17] on eliciting a response from the nerve fiber of a squid axon. They experimentally demonstrated voltage against current relationships for the ion channels, thereby proving the existence of voltage gated channels. Hence, they deduced that variable conductances could be used to model the dynamics. Spiking behaviour and action potentials are produced from neurons due to interactions between different ions.

Voltage clamp experiments were performed, where a step voltage was applied at the

input and current measurements were taken from the ion channel. This data helped to derive the types of channels responsible for the dynamics and formulate a model. From their data [17], the spike is observed primarily due to the interaction between the sodium, Na^+ and potassium, K^+ channel. Hence, studying these dynamics, HH developed a set of equations to create a model for a neuron. The fundamental model, which is a set of non-linear ordinary differential equations,

$$C\frac{\mathrm{d}V_{mem}}{\mathrm{d}t} = I_{in} - g_{Na}(V_{mem} - V_{Na})m^3h - g_K(V_{mem} - V_K)n^4 - g_{Leak}(V_{mem} - V_{Leak}), \quad (1.1)$$

$$\frac{\mathrm{d}m}{\mathrm{d}t} = \alpha_m(1-m) - \beta_m m, \quad (1.2)$$

$$\frac{\mathrm{d}n}{\mathrm{d}t} = \alpha_n(1-n) - \beta_n n, \quad (1.3)$$

$$\frac{\mathrm{d}m}{\mathrm{d}t} = \alpha_h(1-h) - \beta_h h, \quad (1.4)$$

where C is the effective capacitance, V_{mem} is the membrane potential, g_{Na}, g_K and g_{leak} are the conductances of the sodium, potassium, and leak channels respectively, V_{Na}, V_K and V_{Leak} are the potentials of the sodium, potassium, and leak channels respectively; I_{in} is the input current, and m, n, and h are the coefficients that signify the probability that any gate of the ion channel is open at any moment, thereby relating V_{mem} to the speed of activation and inactivation of the channels. These coefficients are further determined by the α and β factors.

The set of equations is a complex model with a number of interdependent parameters. Hence, it is difficult to have direct physical realizations of the model, and earlier efforts included a number of software simulations in MATLAB and neuron simulators [18, 19].

A classical implementation of a transistor channel-based neuron was first shown in [20],

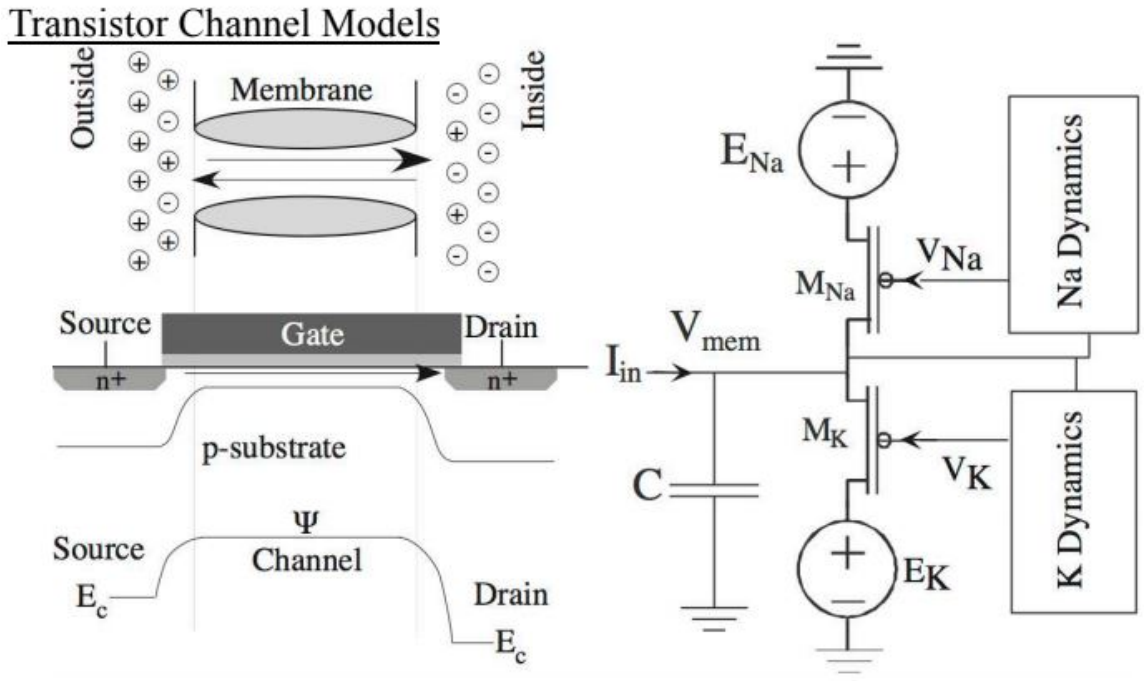


Figure 1.2: The similarity of a biological channel with a bi-lipid cell membrane to a MOS-FET, shown on the left hand side, is the primary motivation behind the ion-channel based implementation [20].The band diagram has the surface potential which is analogous to the biological nernst potential of the ions. The ion channel dynamics are realised through a set of six transistors where the Na^+ and K^+ channel are modeled as a band pass and high pass filter respectively.

exploring the similarity between a biological neuron and silicon. Figure 1.2 illustrates how the approach is inspired by this movement of charge through the biological membrane. This exponential distribution of carriers and their mechanisms of drift and diffusion of ions through a bi-lipid biological membrane are similar to the movement of charge through a transistor channel. The Nernst potentials that represent the equilibrium voltage of the Na^+ and K^+ channels. given by E_{Na} and E_K, are translated to voltages for the neuron circuit. The gating dynamics to produce V_{Na} and V_K for the respective channels was first shown in [20], where two pFETs model the Na^+ and K^+ channel.

The Na^+ channel is responsible for the rising and falling phase of the action potential due to its activating and inactivating mechanism. The step response of the Na^+ channel from the original data [17] shows that it has the characteristics of a bandpass filter. Its time

constants are defined to be τ_m and τ_h. The K^+ channel is responsible for returning the depolarized state of the neuron to its resting potential. Its time constant is defined to be τ_n.

[20] models the Na^+ and K^+ channels as bandpass and highpass filters, respectively. The K gating, with respect to how much it turned on, was quantitatively described by the 'n' activation factor, while the 'm' activation and 'h' inactivation factors determined the Na gating time constants.

1.3.2 Integrate and Fire Neuron

The integrate and fire neuron [21] is one of the simplest models for realizing a neuron circuit. Hence, it is commonly used in a number of systems [18, 22] as well as digital neuromorphic implementations [23]. Although this does not truly mimic the biological neuron, it is commonly used in rate-encoded spiking neural systems, where one does not care about preserving the computational features of a true neuron. A spike is generated as soon as V_{mem} exceeds the firing threshold of the neuron; V_{mem} is then reset to the resting potential, V_{rest}, and a refractory period is added, while the conductance is modeled to be g_{leak} with a stimulation current of I as shown in the equation below:

$$C\frac{\mathrm{d}V_{mem}}{\mathrm{d}t} = I - g_{Leak}(V_{mem} - V_{rest}). \tag{1.5}$$

1.3.3 Izhikevich Neuron

The Izhikevich neuron [24] is a model that produces a wide variety of action potentials, reproducing different spiking dynamics, including a phasic spiking where the neuron remains inactive after producing a single spike, or tonic bursting where periodic bursts of spikes are observed, and a mixed model where tonic spiking follows phasic spiking at the beginning of the stimulation. This model has been widely used to reproduce spiking using software simulators for analyzing the dynamics of populations of neurons. FPGAs have been used to reproduce this behaviour in hardware [25, 18].

There have been multiple approaches to emulating the spiking of neurons. A fundamental implementation of a silicon neuron circuit [26] was shown that builds from the classic HH equations. Reference [19] built a HH neuron by implementing a library of analog operators while [27] has shown a circuit based on a voltage-dependent ion channel model. Reference [28] directly translates the equations, which may involve a larger area due to the higher number of elements used. A neuromimetic integrated system in [29] compares biological neuron spikes with the HH model. Another approach uses gating variables inspired by the rates of the closing and opening of ion channels in a reconfigurable chip [30]. A comprehensive set of different circuit models and solutions to build neurons on silicon has been discussed in [18].

1.4 Organization of the dissertation

This work aims to use the SoC FPAA to demonstrate various analog and neuromorphic computing techniques. Chapter 2 presents experimental silicon results on the dynamics of a transistor channel based Hodgkin-Huxley neuron, inspired by the similarity between biology and silicon, by modeling the ion channels and their time constants. It introduces techniques of mismatch compensation by tuning the ion channels to produce a variety of action potential dynamics and spiking responses from different inputs.

Chapter 3 introduces our FPAA tool infrastructure, primarily the aspects of a simulator modeling the fundamental components like transistors, amplifiers, and FG devices. It demonstrates close agreement between simulated results and experimental measurements by implementing systems such as continuous-time filters and the analog front-end of a speech processing system. It further discusses models for simulating various analog circuits over temperature.

Chapter 4 describes a methodology for building networks on the FPAA from transistor channel-based neurons and synapses. It elaborates on the neuronal responses obtained through excitatory and inhibitory synapses and shows post-synaptic potentials from the

synaptic clefts and neurons. Further, the models are used to demonstrate neuromorphic systems of spiking neurons and synapses.

Chapter 5 focuses on the tuning algorithm for compensating for indirect programming mismatch, especially to set the weights on a Vector Matrix Multiplier (VMM). It discusses the constraints during the design and implementation process, accounting for the mismatch in devices. Further, it shows an application of this algorithm through a set of programmable bandpass filters with the tuned VMM.

Chapter 6 discusses an analog solution of a linear system of equations and shows the advantages of analog computing. It experimentally demonstrates these continuous-time analog solutions through varieties of transconductance amplifier based networks for different types of inputs, sizes and matrices. Finally, Chapter 7 discusses the issues to remember while building complex systems and concludes the discussion by introducing the design and layout of FG test cells in 14nm to scale to lower process nodes and build the next generation of hardware for energy-efficient computing.

CHAPTER 2

HODGKIN-HUXLEY NEURON AND DYNAMICS ON THE FPAA

2.1 Hodgkin-Huxley neuron on a reconfigurable platform

The pathway of neuromorphic computing [31] offers a number of opportunities to build circuits and systems efficiently, especially due to the analogy between biology and silicon. This work aims to present hardware results on the dynamics from a Hodgkin-Huxley (HH) neuron adapted from our original model [32] and implemented on a System on Chip (SoC) large-scale Field Programmable Analog Arrays (FPAA) fabricated on a 350nm CMOS process [2].

Hodgkin and Huxley won the Nobel prize for their work [17] on eliciting a response from the nerve fiber of a squid axon. The action potential is generated as a result of a set of voltage-gated ion channels. This exponential distribution of carriers and their mechanisms of drift and diffusion of ions through a bi-lipid biological membrane are similar to the movement of charge through a transistor channel. The left panel of Figure 1 illustrates how our approach is inspired by this movement of charge through the biological membrane and the reproducibility of the action potentials. The right panel suggests accessibility to a wide set of users either remotely or through a local board.

This work reproduces the spiking behavior and shows the nonlinearity of dynamics of HH neuron by modeling voltages, ion channels, and time constants. We show a model which is not only inspired by the physical similarity between ion channels and semiconductors but which also helps clarify a few concepts on ion-channels and neurons, as shown in Fig. 1. Rather than just simulating on a digital computer and understanding via the equations, anyone can reproduce similar results using our open-source tool infrastructure and remote system [33] [12]. One can study and analyze the system behavior by observing the

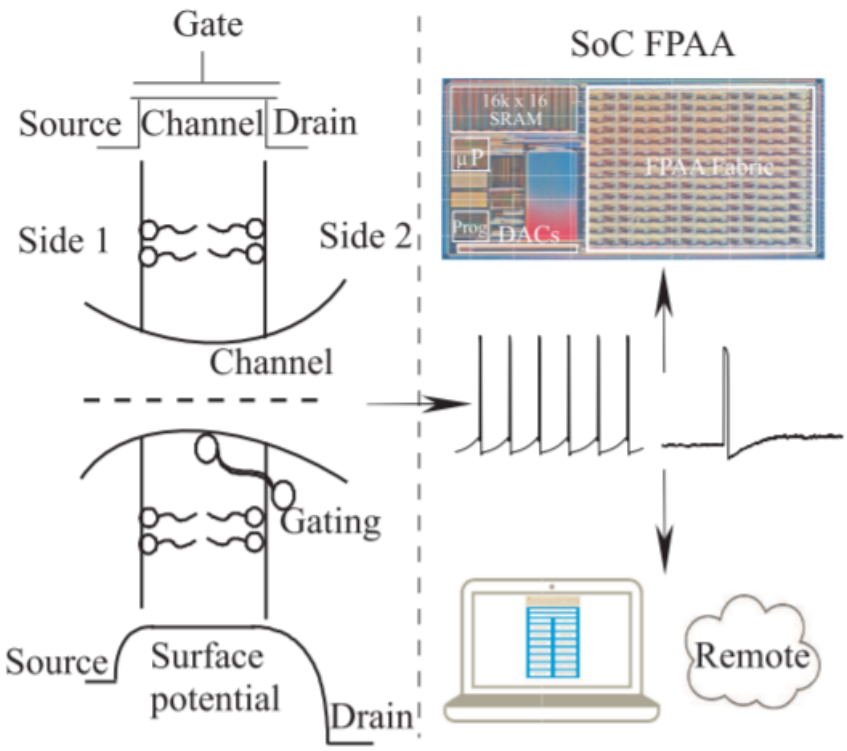


Figure 2.1: High level idea behind the implementation of the Hodgkin-Huxley neuron on the FPAA. The similarity of a biological channel with a bi-lipid cell membrane to a MOSFET, shown on the left hand side, is one of the primary motivations behind the ion-channel based implementation. The band diagram has the surface potential which is analogous to the biological nernst potential of the ions. The hardware implementation of the HH neuron model on SoC FPAA, inspired by this concept, offers multiple opportunities. The reconfigurability helps to replicate a variety of action potentials through our open toolset and thereby offering greater insights to a few concepts in neuroscience through experimental results from the hardware.

effect of tuning the ion channel parameters with ease due to a reconfigurable chip which helps the user get better insight into the causality behind the dynamics. Hence, this chapter demonstrates a system from the perspective of the user.

A software simulation is done by modeling a set of equations which has more than twenty parameters [17] to tune to obtain the right behaviour. However, in our system we mainly have three bias parameters to tune the time constants for spiking, while the other parameters like E_{Na} and E_K, the supplies to the circuit, are constant and global, thereby reducing the whole parameter set to obtain a spiking behaviour. Also, the bio-physical

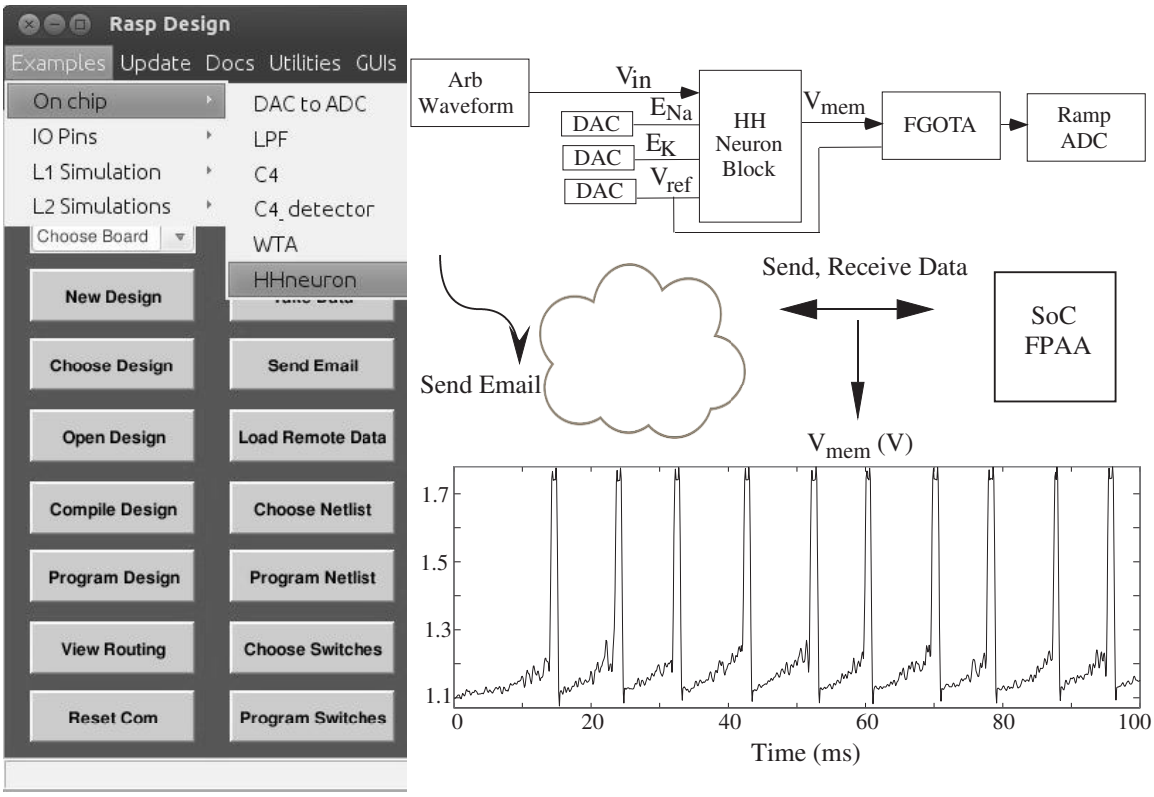


Figure 2.2: The graphical user interface from which the HH neuron design can be viewed from the examples drop-down, in the open-source tool infrastructure is shown. The FGOTA and the ramp ADC constitute the measurements setup, to amplify the membrane voltage signal and observe the spiking behavior from the neuron clearly. The example can be compiled and data can be measured by sending the email and remotely receiving the result from the FPAA to observe various dynamics. One such set of behavior is zoomed out for the clarity of the reader and shown for the membrane voltage output from the neuron.

behaviour is emulated in hardware since we are modeling the ion channels without just curve fitting and translating the mathematical equations to hardware. Unlike an ASIC, the reconfigurable nature allows us to obtain a multitude of dynamics for the HH neuron and we are able to use the same chip to scale up as well as implement the functionality of neuron models as well. The low cost and flexibility, as well eliminating the waiting for the chip to arrive after every fabrication, helps in rapid prototyping, while the remote system enables the user to obtain the relevant dynamics from the neuron as it is available in the examples in the tools. It can be replicated by others in the community, with the inner details having been abstracted away for the ease of the user, while at the same time, offering the flexibility

of access to an advanced user too.

This chapter elaborates on the modeling of the HH neuron and implementation of the circuit and the varying spiking dynamics observed through the experimental measurements obtained from the FPAA. Section 2.2 introduces the translation of our original model to the FPAA with all its constraints and opportunities. We first observe the action potentials in Section 2.3. Section 2.4 describes the similarity between a biological channel and transistor channel. We discuss the criticality of getting the correct biasing points in Section 2.5 and further discuss the physics behind the ion channels in Section 2.6. Section 2.7 gives deeper insight into different spiking dynamics across chips, while Section 2.8 concludes the discussion.

2.2 Implementation on the FPAA

The transistor channel models are modified from the circuit in [20], utilizing the current FPAA resources using nFETs, pFETs, Operational Transconductance Amplifiers (OTAs) and Floating-Gate (FG) OTAs such that the circuit is contained in one CAB, thereby occupying less than 1 percent of the IC. The current chips fabricated in a 350nm CMOS process on which the measurements have been taken consist of 98 CABs.

Hodgkin-Huxley performed voltage clamp experiments, where a step voltage was given and the current measurements were taken from the ion channel. This data helped to derive the types of channels responsible for the dynamics and formulate a model. From their data [17], it is seen that the spike is observed due to the interaction between primarily the sodium, Na^+ and potassium, K^+ channel. Their Nernst potentials, given by E_{Na} and E_K, are translated to voltages here, which are in terms of U_T, thermal voltage, and serve as supplies to the neuron circuit. The gating dynamics to produce V_{Na} and V_K for the respective channels are built from the FPAA resources, where a pFET and nFET model the Na^+ and K^+ channel respectively, operated in the subthreshold regime.

The Na^+ channel is responsible for the rising and falling phase of the action potential

due to its activating and inactivating mechanism. The step response of the Na^+ channel from the original data [17] shows that it has the characteristics of a bandpass filter. Its time constants are defined to be τ_m and τ_h. The K^+ channel is responsible for returning the depolarized state of the neuron to its resting potential and its time constant is defined to be τ_n.

Our classical circuit [20] modeled the Na^+ and K^+ channel as bandpass and highpass filters respectively. The K gating, with respect to how much it turned on, was quantitatively described by the 'n' activation factor, while the 'm' activation and 'h' inactivation factors determined the Na gating time constants. However, in the FPAA, due to the significant capacitance in the routing, the FPAA gain from V_K to V_{mem} is reduced, whereas the original structure [20] needs a higher gain, especially around higher frequencies. This inspired us to modify the design, for the K^+ channel to be a lowpass filter, whose output to the K gating nFET is represented by V_K. An FGOTA in a follower configuration is used for this purpose, since the FGs help set the DC offsets, while tuning the bias current at the right place would produce the desired τ_n.

The Na^+ channel's output to the Na gating pFET is represented by V_{Na}. The Na channel, a bandpass filter, can be considered to be a modified Capacitively Coupled Current Conveyer (C^4) filter [2]. An FG-pFET is used in the feedback to set the τ_h constant. No extra device is required for this purpose, since the feedback is a part of the local routing without occupying an extra area. Figure 2.3 demonstrates the concept behind the above implementation on the FPAA. The role of the local routing, as shown through the input and output lines to the CAB devices is focused on in Figure 2.3. The switches in the routing, the FG-pFETs are used as programmable switch elements for computation [14]. The routing tracks dominate the area and one transistor is similar in area to one OTA, both being three terminal devices, inspiring the transformation of the design of the channel gating dynamics modeled by the transistors to OTAs. Hence, a primarily transistor-based Na^+ channel is converted to an FGOTA and an FG-pFET switch on the FPAA. Due to the routing parasitic

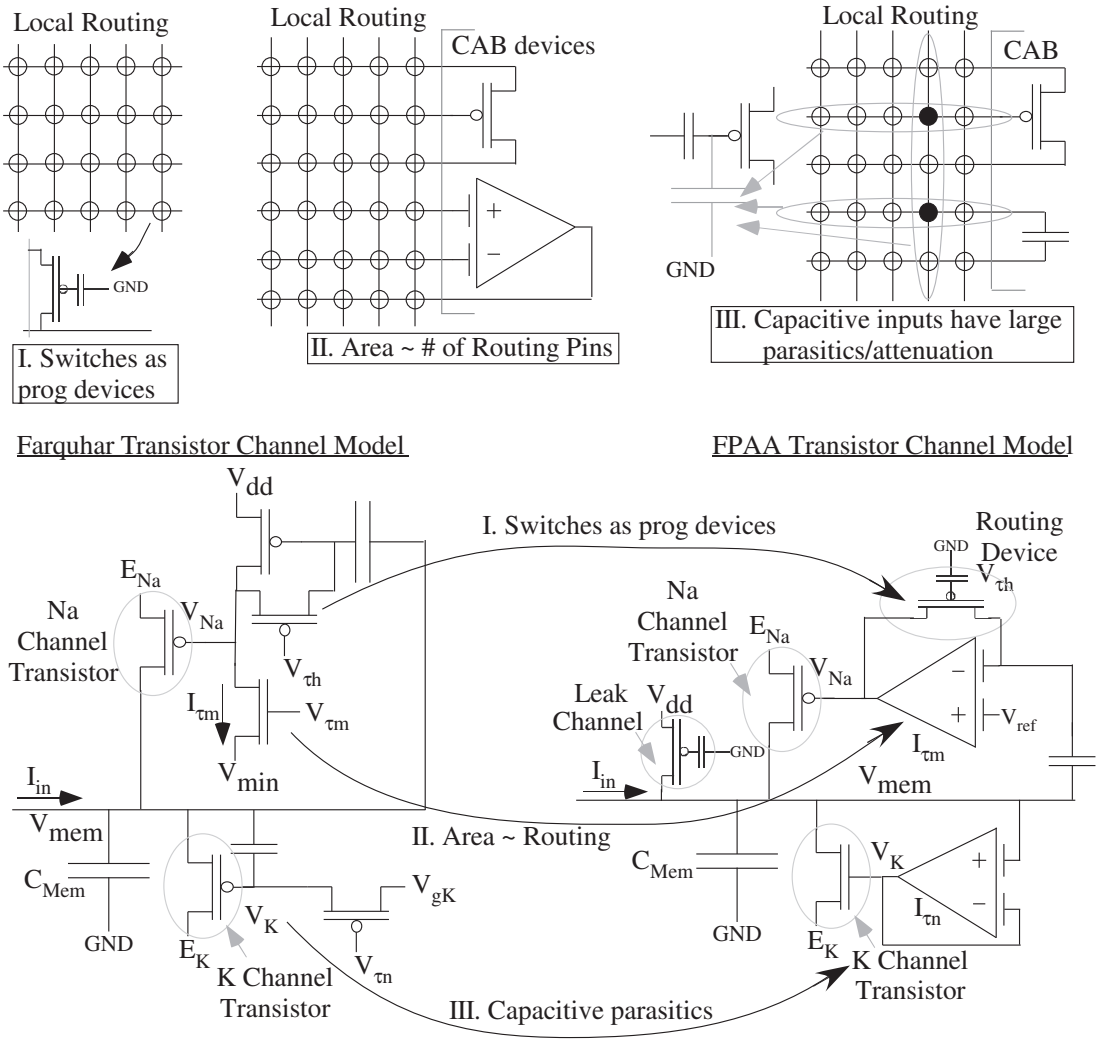


Figure 2.3: Translation from the original structure to the one on the FPAA constrained in one CAB. The switches in the routing as depicted are not deadweight but those FG-pFETs are used as programmable switch elements for computation. With the current structure on FPAA, a transistor or an OTA corresponds to a 3 terminal device, thereby an extra area is not a concern since it is decided by the number of routing pins. Keeping this in mind, the Na^+ and K^+ channel are converted on our FPAA as shown.

capacitive inputs contributing to a large attenuation, the highpass K^+ channel translates to a lowpass structure with the desired gain on our device.

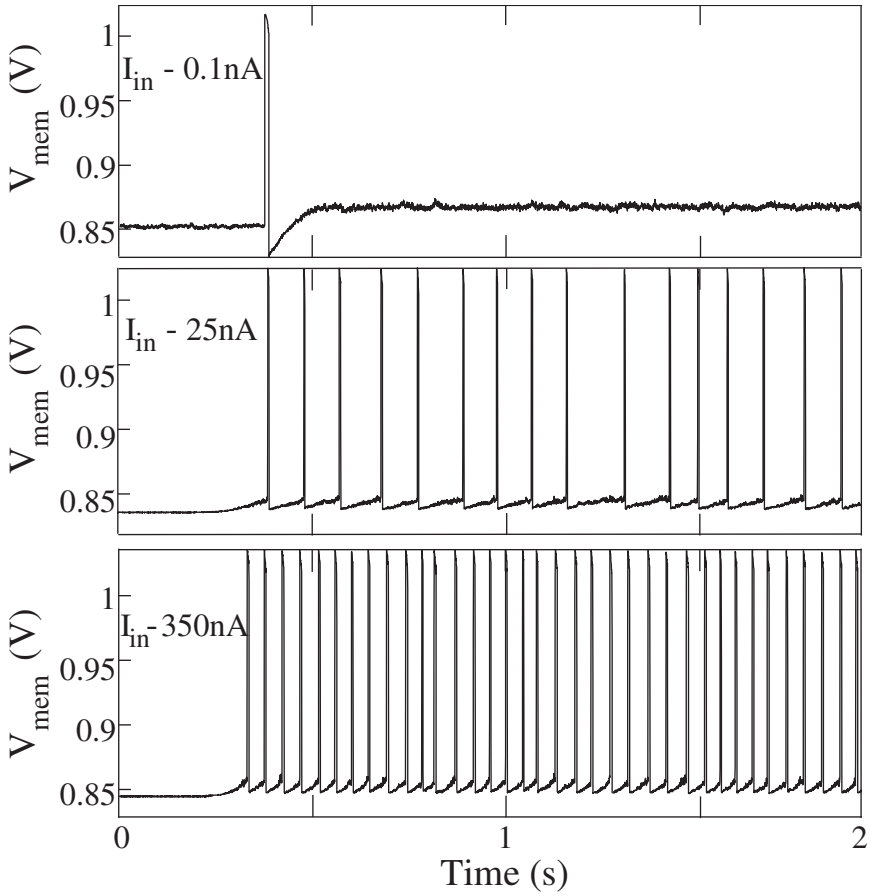


Figure 2.4: Different spiking rates and change in the inter-spike interval for a set of conditions and parameters, over a longer period of time. The effective input current is varied, thereby changing the total current contributed at the membrane voltage node. The behavior of V_{mem} changes from one spike to multiple spikes with different rates.

2.3 Action potential

Figure 2.4 shows the spiking behavior on V_{mem} in response to a step input current. The effective input current to the cell is used as a primary parameter to study this behavior [34]. We can see that the inter spike interval (ISI) decreases and the number of spikes in a specific period of time increases as the input current increases across the plots [35]. A phasic spiking is observed [24] initially for a lower current, where a single spike is observed and V_{mem} rests at its resting voltage. Continuous spiking patterns are observed as the current changes. The average firing rate is consistent but the thermal noise contributed by the transistors in the model as well as from the measurement and instrumentation circuitry

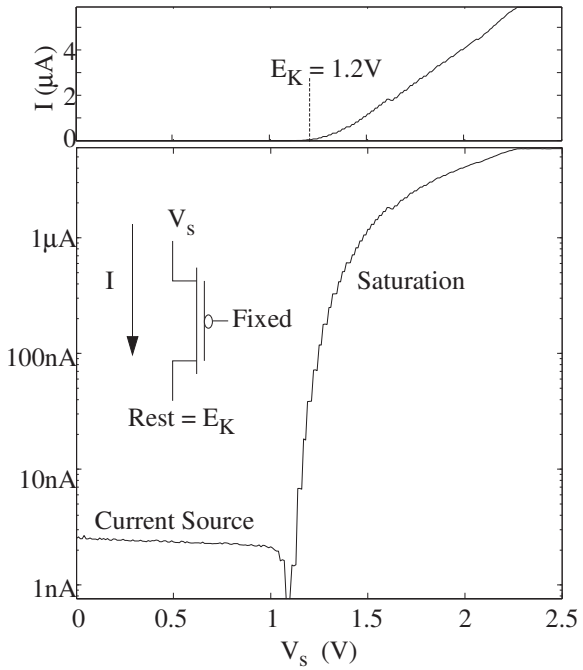


Figure 2.5: The pFET is modeled as a prototype for a biological channel. Current as a function of the source voltage is shown in absolute as well as log scale. The gating potential is fixed at 0.1V, while E_K is biased to 1.2V, through which we can identify the dual behavior of the terminal as a source and drain.

causes changes to the ISIs in the spiking pattern over time.

2.4 Passive Channel

A pFET is modeled as the prototype for a passive ion channel, which selectively allows ions to cross the membrane. The flow of ions due to the difference in the concentration in the intracellular and extracellular space [36] and the channel formation is similar to the energy band diagram as observed with a silicon channel model. A single transistor modeled as a biological passive channel aids us in deciding the regions of operation of the circuit elements, thereby initiating the process of figuring out the parameters to bias the system. The I-V relationship [15] shown in Figure 2.5, where we measure the pFET as a passive channel, helps us take a systematic approach.

Fixing the gating potential of the pFET and the drain voltage at E_K, the current flowing

through the transistor is observed as the source voltage is swept. It can be seen from the current though the pFET vs Source voltage plot in Figure 2.5 that the terminal behaves as a source and a drain node when it acts as a current source and when the transistor is in saturation, respectively. There is a constant current when the sweeping voltage is less than E_K as the source voltage crosses the resting potential of 0.85V on E_K. This can be observed in an activating excitatory synapse when the FG-pFET's source is at E_{Ca} which is $4U_T$ [36], while its drain is at the membrane voltage at a resting potential of $-2.5U_T$ and for a peak at the gate. The passive channel concept is also relevant in the case of modeling of dendrites [37], where the saturation region is exploited.

2.5 Biasing the Neuron

The complementary relationship between the voltage and current from a pFET modeled as a biological channel plays a significant role in determining the bias points for the various nodes, giving an insight into the design. The current measurements resulting from a step voltage input through the voltage clamp helps determine the gate voltage needed to produce the desired response from the ion channels, which in turn, is the starting point of the biasing of the neuron. Figure 2.6 shows the concept behind the sequence in figuring out the biases required to obtain action potentials from the neuron.

The gain from the Na^+ gating channel output to the membrane voltage is around 8. Thus, the DC of V_{Na} can be fixed to be around 250-300mV, thereby setting the source and drain DC voltages for the feedback FG-pFET and programming a large current for the feedback element. For the Na gating channel pFET to be in saturation, knowing V_{Na} and the current through the pFET in saturation in the order of nAs, we can set E_{Na} to be at 1V. This sets E_K, since the difference between E_K and E_{Na} is around 150mV. From the Nernst potential [36], we know E_K is $-3U_T$, while E_{Na} is $2U_T$. The resting potential for membrane voltage is 10mV above E_K. From E_K and the current through the K gating nFET, we can determine V_K to have a DC of around 1.25V. From these DC bias points, we

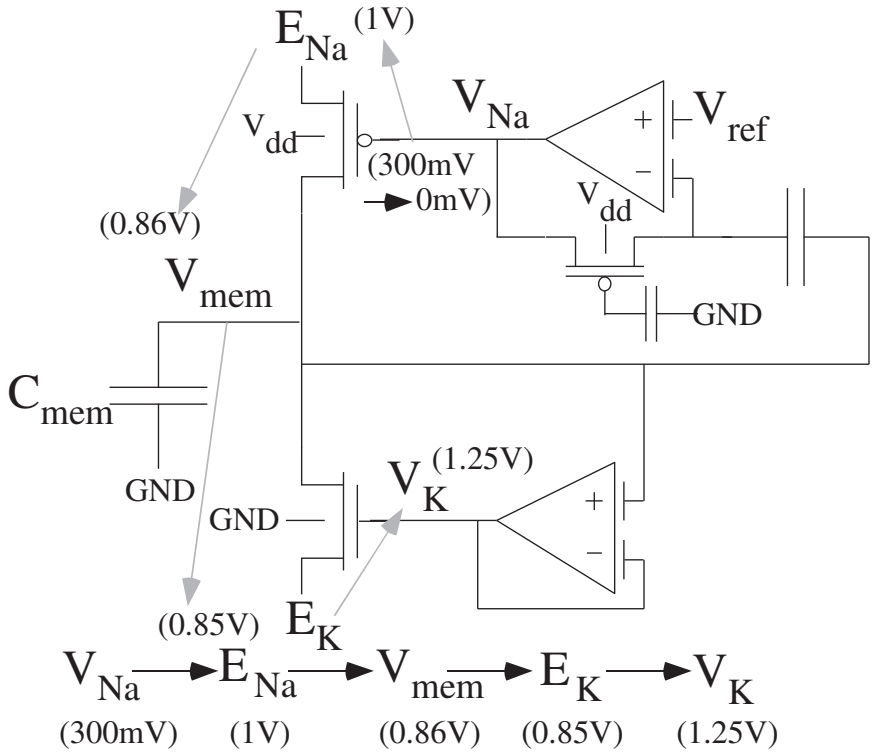


Figure 2.6: The bias levels are important for the circuit since they determine the dynamics of the action potential, such that it is between E_K and E_{Na}. Further, the FGOTAs' bias currents help to set the time constants of the K^+ lowpass filter and the Na^+ bandpass behavior. The arrows indicate the sequence followed to arrive at the various DC levels, starting from V_{Na}.

can determine how to tune the FG OTAs and the conductances to generate a spike.

The biasing on the Na^+ is critical in terms of getting the DC values, time constants, and gain right, since that sets the rest of the biasing and time constants. A gain of -8 or -12dB between V_{mem} and V_{Na} is required for it to spike, so that one does not see just an oscillatory response. The corners of the bandpass filter are tuned such that there is a factor of two to four between the two bands. It is important to make sure that the feed forward time constant due to the forward FGOTA, τ_m, is faster than the feedback time constant, τ_h, caused by the output to the middle capacitively coupled node. The DC level affects the corner frequency of the amplifier, since it is giving the DC point of the FG-pFET device that is creating the feedback. Considering the channel transistors to be in saturation with the Na^+ channel

pFET's current being 1 percent of the K^+ channel nFET, the resting potential of V_{mem} sits at E_K. A leak channel that represents the ion channels for chloride and other ions [17], modeled by an FG switch in the routing, is added to pull up the V_{mem} to $U_T/2 + E_K$, as the V_{mem} is at the transition point between the linear and nonlinear conductance regions. The time constants are set such that τ_n is slower than τ_m, biologically similar to the fact that K^+ channels respond slower than the Na^+ channels.

2.6 Physics of Sodium and Potassium Channel

Figure 2.7 illustrates the Na^+ and K^+ channel behavior clearly. The interaction of the currents from the Na^+ and K^+ produce the action potential. A step input current is injected into the node through an OTA or an FG-pFET which models a synapse. The capacitance from the line acts as the membrane capacitance, C_{mem}. Figure 2.7(a) shows the responses of the Na^+ and K^+ channel individually extending the concept of the voltage clamp experiment, where the ionic currents were observed [17] after selectively blocking the respective ion channels. Activating one channel at a time in the system, the bias parameters were calculated through simulation [15] and from the experimental results on hardware. They were selected such that we get a bandpass response for the Na^+ channel while the K^+ channel gives a lowpass response. The transient responses to a step input in the Figure 2.7(a) give insights into the separation of the time constants τ_m, τ_h and τ_n of the ion channels. From the calculations shown in Figure 2.6, we get similar DC points by adjusting the offsets through the FGOTAs.

Figure 2.7(b) shows a single spike. The different regions of the action potential are labeled in the figure. The membrane voltage initially starts at its resting voltage, about 10mV above E_K at 1. The region 2 of depolarisation, where the Na^+ ions flow in, is represented here when the Na^+ gets activated and the positive feedback through the FG-pFET kicks in and as V_{Na} drops, V_{mem} rises to E_{Na}. At 3, the K^+ channel starts to affect the response and the feedback time constant, τ_h, starts recovering. The Na^+ channel sees the input to

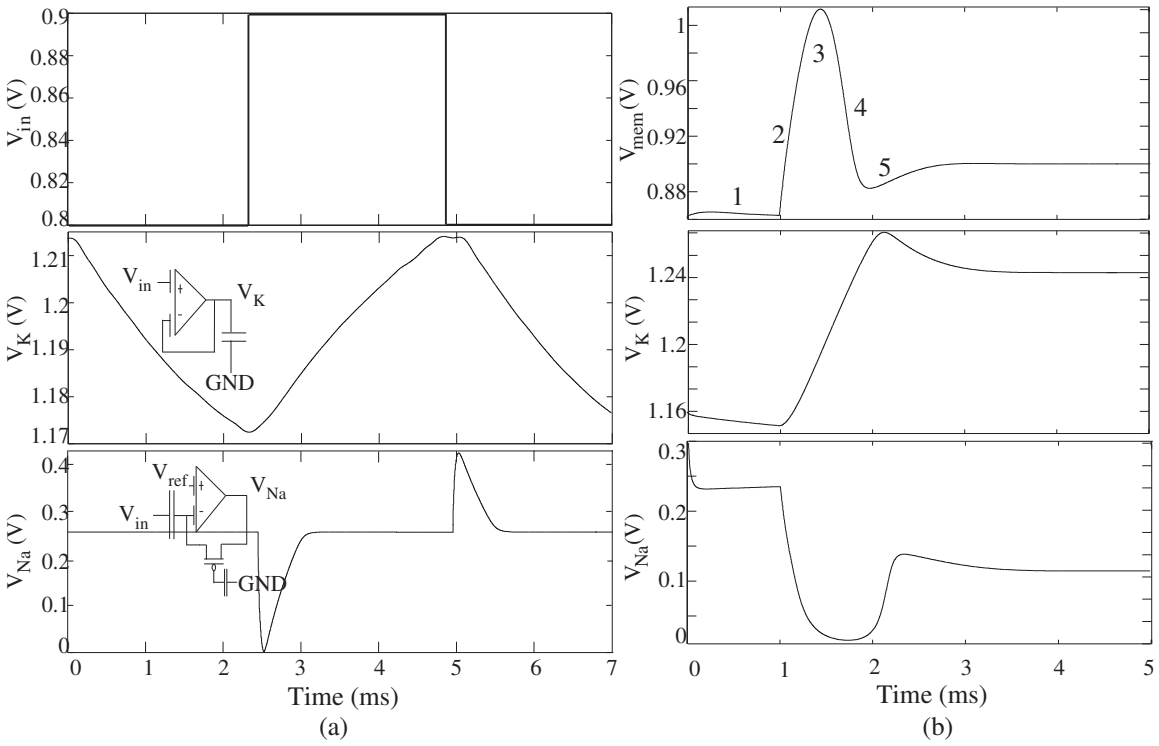


Figure 2.7: (a) The K^+ and Na^+ gating circuit and the experimental results for the step response of the channels are shown. A step input of 100mV is applied. The transient response of the K^+ and Na^+ channel depicts the low-pass and bandpass filter behavior respectively. The lowpass effect is responsible for bringing the action potential back to the resting potential. A DC level of 1.2V is achieved by tuning the offset currents in the FGOTA and the time constant can be tuned by I_{τ_n}, the current to which the FGOTA is biased. The DC level of the Na^+ channel is observed at around 300mV. The feedback time constant is controlled by the FG-pFET switch in the routing. (b) The device level simulation results for an action potential to a step input of 100 mV are shown. V_{mem}, V_K and V_{Na} are observed on one plot in response to a step input for a set of biases.

it decreasing, causing V_{Na} to rise and settle to its DC of 300mV, and τ_h pulls V_{mem} to its equilibrium at 4. At 5, the K^+ and Na^+ shut off, and we see the rise in V_{mem} as the node charges through C_{mem}. The refractory period is a factor of the input current integrating through the capacitor at the node. This spiking behavior observed is biologically similar to the ion channel behavior with the Na^+ and K^+ channel conductances changing with time due to the opening and closing of the respective ion channels, causing the depolarisation and hyperpolarisation in the membrane voltage.

Observing the results from the transitions of V_K and V_{Na} with respect to V_{mem}, we can

further understand the close coupling between the nonlinear dynamics of the ion channels, which contributes to an action potential in Figure 2.6(b) through circuit level simulations [15]. As the voltage on V_{mem} starts rising from its resting voltage to its peak, there is a fall in V_{Na}, with V_K peaking as expected, with the rise and fall governed by the time constants. The rise during the refractory period is controlled by C_{mem}.

2.7 Looking into different spiking dynamics

This section explores different dynamics, analyzing the effect of modifying time constants to achieve different firing rates. The tuning for mismatch is also studied to obtain similar dynamics across chips along with subjecting the system to other signals like ramp inputs and noise.

2.7.1 Effect of modifying time constants

Figure 2.8 shows a multiplication in the frequency of the spikes as the plots progress from (d) to (a). This adaptation in the spiking frequency is due to the change in time constants τ_m, τ_n and τ_h, which are changed by tuning the bias currents I_{τ_m} and I_{τ_n} and V_{ref}. As they change, the time taken by the Na^+ and K^+ channel to respond to changes in V_{mem} varies, leading to higher spiking frequency.

I_{τ_m} and V_{ref} change the corners of the bandpass filter responsible for the Na^+ channel, while I_{τ_n} adjusts the corner of the K^+ channel. We show how we can run our circuit at different time scales in Figure 2.8. Our tools allow one to use a single Xcos design;fine-tuning the parameters enables running at slower, faster or real-time speeds.

2.7.2 Calibrating for mismatch across chips

Mismatch in the responses between different chips or among different neurons has been beneficially exploited in some applications [38, 39, 40, 41]. As expected, the intra-neuron mismatch due to variation from CAB to CAB is less than from chip-to-chip, since the die

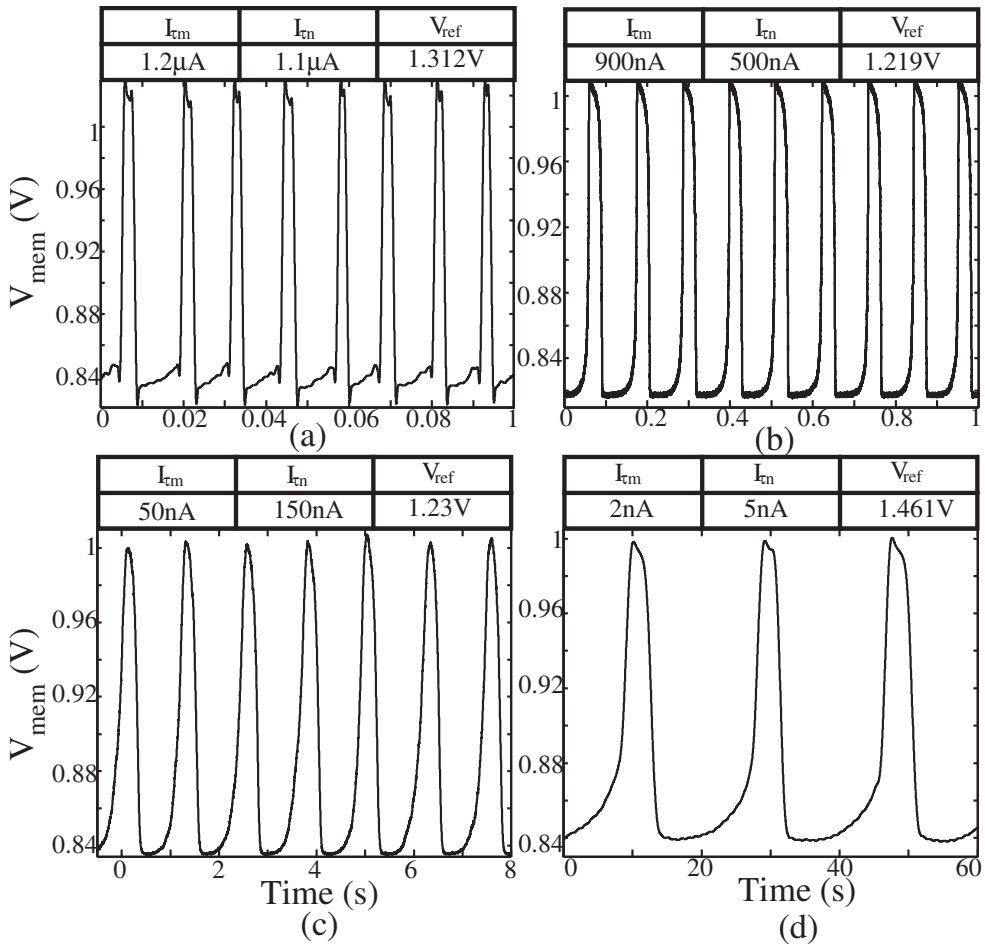


Figure 2.8: The experimental measurements' data from the FPAA for an action potential to a step input of 100 mV, exhibiting different spiking frequencies. The biases are controlled such that different rates can be obtained, scaling the I_{τ_m}, I_{τ_n} and V_{ref} accordingly depending on the rate desired. As observed from the time scale, the spiking frequencies reduce with the progression of the plot from (a) to (d).

to die variation is higher than on-die variation [42]. A classic neuromorphic problem is studied here to identify if this spiking behavior is reproducible. Experimental measurements were taken across three different chips. We try to minimize this mismatch since we wish to efficiently use as many neurons as possible for computation, without using extra resources to correct for the added variation [43]. The results with the same identical set of parameters are shown in Figure 2.9(a). The threshold voltage mismatch caused due to the indirect FG programming infrastructure can either be compensated by using the systematic automated mismatch map approach [44] in our system or other techniques as shown in [45],

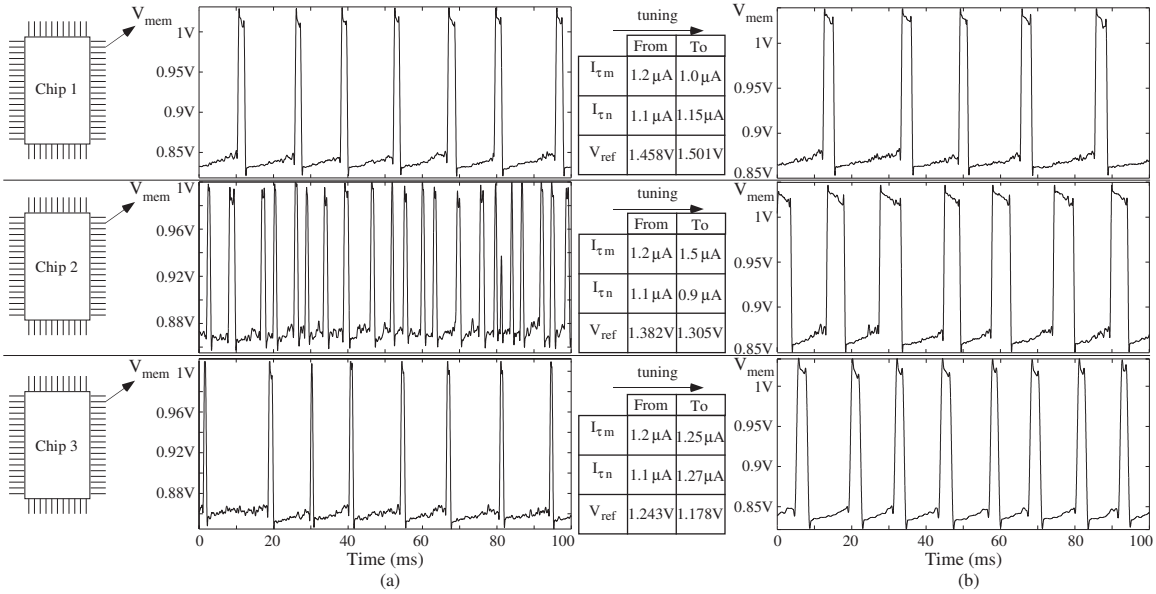


Figure 2.9: The experimental measurements' data from the FPAA for an action potential to a step input of 100 mV, from different boards. Figure 2.9(a) shows results before tuning for mismatch, while Figure 2.9(b) shows the measurements after tuning. The biases are controlled such that the DC biases match along with the different frequencies and rates from the different chips. The threshold voltage mismatches are compensated and adjusting I_{τ_m}, I_{τ_n} and V_{ref} aids in adjusting the respective time constants.

with translation of parameters in a network [46]. References [47, 48] have explored different optimization algorithms to arrive at the parameter space set for the tuning the neuron model [19]. The parameters are tuned here at the neuron level by looking at the varying time constants specific to this circuit since our platform allows one to shift biases.

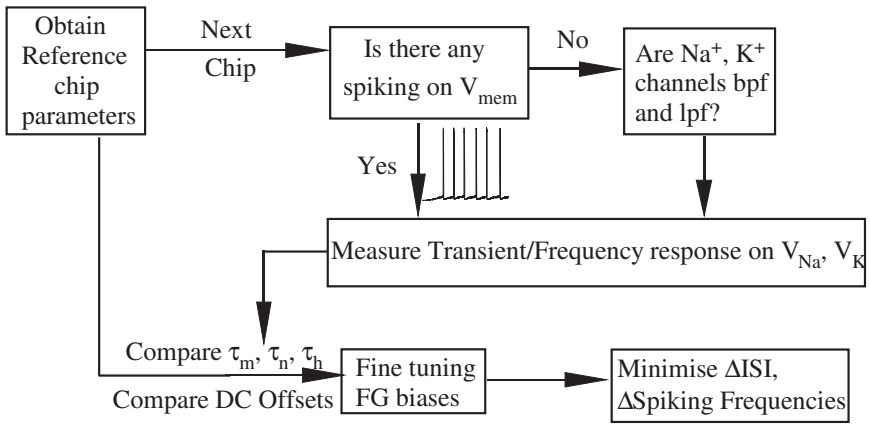


Figure 2.10: The tuning algorithm to minimise the variation between different chips to get similar spiking responses across boards is shown by analysing V_{Na}, V_K and then fine tuning the FG biases on the CAB elements of the neuron model.

CORRELATION MEASURES ACROSS CHIPS

Metrics	Before Tuning			After Tuning		
Chip Number	Chip 1	Chip 2	Chip 3	Chip 1	Chip 2	Chip 3
ISI (ms)	12.14	4.07	12.45	15.71	10.08	9.61
Spiking Frequency	82.37	245.7	80.32	63.65	99.2	104.1

Figure 2.12: The table shows metrics like ISI and spiking frequency, to demonstrate numbers to quantitatively compare the behaviors before and after tuning for mismatch.

Figure 2.10 shows the tuning methodology for mismatch minimization between the chips. We first start with obtaining the spiking response from a reference board by programming the biases and time constants of each of the channels as elaborated in Section 2.5. Once the reference parameters are obtained, these are used for compiling down the design to the respective chips. An Xcos block for debugging has been created in the library of blocks with the pins instrumented out to measure V_{Na}, V_K and V_{mem}. With the reference parameters, a check is done to observe if there is any spiking activity or not. If there are no spikes, we do a sanity check to ensure Na^+ and K^+ channel behave as bandpass and lowpass filters, respectively.

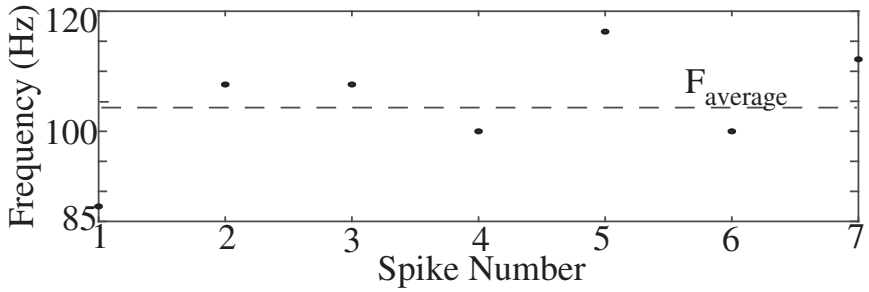


Figure 2.11: The spiking frequency for a chip represented as filled circles is plotted as time progresses. It is defined as the reciprocal of ISI here. It is not a constant due to the thermal noise of the transistors in the system. The average value across time is calculated, marked as a dotted line, to compute the spiking frequency of each chip, which is used to quantitatively correlate the behavior between the chips.

After observing the spiking behavior on V_{mem}, we calculate the variation in the ISIs or

spiking frequency to minimize its difference from the parameter chip. A sample average spiking frequency is shown in Figure 2.11. The transient or frequency response from V_{Na} and V_K helps us calculate the respective time constants τ_m, τ_h and τ_n. The FG biases on I_{τ_m}, V_{ref} and I_{τ_n} are then fine-tuned. To get equivalent spiking amplitudes between the chips, E_{Na} and E_K which are the supplies given through DACs on the respective chips, are calibrated and the DC offsets are equalised through fine tuning of the FG biases on the FG OTAs of the Na^+ and K^+ channel.

Figure 2.9 shows how the rates and frequencies are thus matched across the chips, thereby giving effectively identical behavior among all the three chips. This proves our hypothesis that the dynamics are repeatable and reproducible among the different boards too, due to the programmability offered by the FG devices in the FPAA. We have defined different metrics like variations in ISI and spiking frequencies to correlate the performance across chips in Figure 2.12.

2.7.3 Effects of Ramp and Noise inputs

Subjecting the circuit to different ramp and noisy inputs, we see different dynamics as seen in Figure 2.13. When a ramp signal is applied at the input, there is an initial delay in the spiking which is proportional to the time for which the ramp is active before it settles to a steady value. Figure 2.13(b) also shows that after the first four spikes, the inter-spike interval changes due to the delay, then we observe continuous spiking.

Random noise is input to explore the effect of noise [49, 50, 51] since it is an inherent property of a biological system [52]. The resulting V_{mem} spikes even with a noisy signal. The system is edge-triggered rather than level triggered since an action potential is observed even at an input level lower than the threshold value. In these cases, V_{mem} is primarily influenced by changes on V_{Na} since a positive feedback loop is created between V_{Na} and V_{mem}. The bandpass filter for the Na^+ channel gating destabilizes the circuit since the loop gain is higher than one. The $'h'$ variable is significant in the downward transition where the

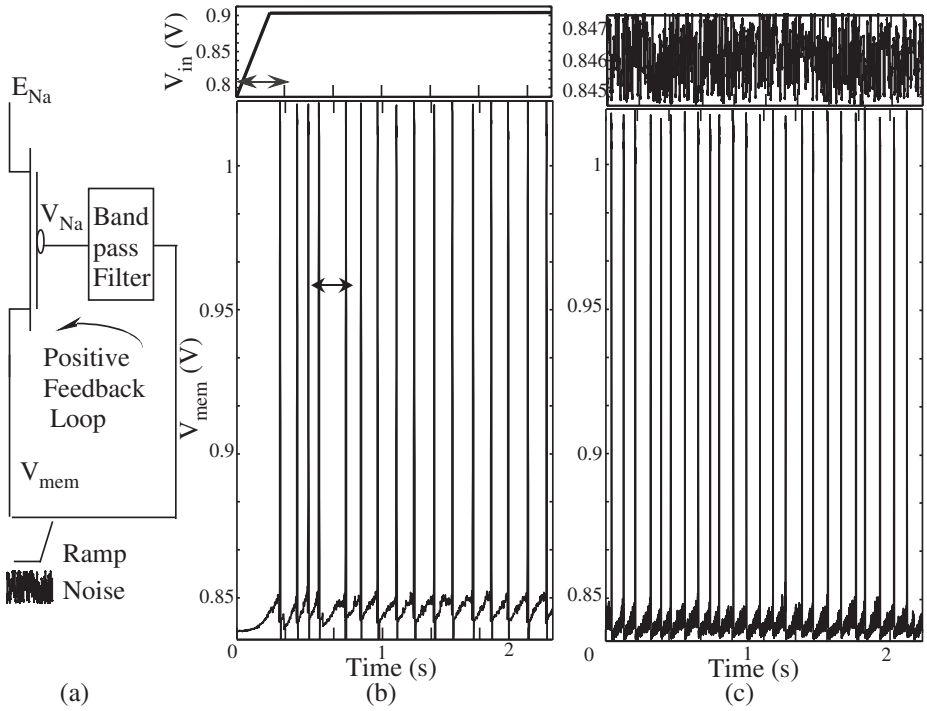


Figure 2.13: Response of the membrane voltage to a different set of inputs over a longer period of time. (a) The positive feedback loop from V_{Na} to V_{mem} destabilizes the system thereby affecting the overall dynamics of the system. The 'h' variable plays a significant role here. (b) An initial delay in V_{mem} is seen corresponding to the application of the ramp input and it continues spiking. (c) With a baseline of 0.84V and considering a threshold of 0.85V, a Gaussian noisy input generated through a random noise generator, shows that it continues spiking with higher noise levels as soon as it sees a rise in the input signal.

feedback is out of band. The positive feedback is again analogous to that caused due to the increase in g_{Na}, with more influx of the Na^+ ions.

2.8 Summary and Discussion of the neuron structure

FPAAs offer reconfigurabilty and programmability, which enables rapid prototyping in hardware. It is useful for scaling up and building large-scale networks and systems for real time applications, unlike simulations in software that may have a reduced speed. Parallel execution nature and the fact that each neuron occupies only one CAB element on the SOC makes it a useful platform. The circuit also consumes only 0.79 μW of power due to the devices being operated in the subthreshold regime. When one thinks of expanding to

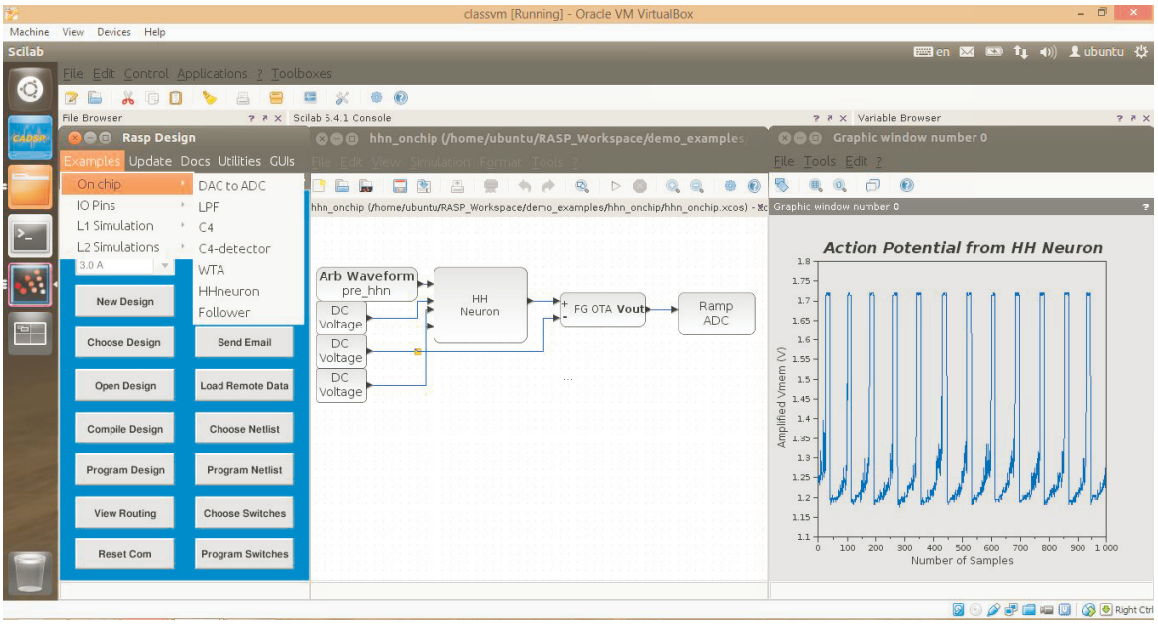


Figure 2.14: The screen-shot of the setup with the results, shows a spiking waveform measured from the ramp ADC for a set of biases. The membrane voltage signal has been amplified up through the use of an FGOTA. The screen-shot portrays the idea that the system can be designed and the output can be captured through the on-chip setup designed using the open-source tool infrastructure, encapsulated in a single virtual machine.

build larger networks for real-time applications, the neurons and synapses built from CABs and routing elements are used for computation.

Through this chapter, we attempt to bridge the gap between users in the hardware and computational communities [53], especially by providing reusable neuron blocks [54] on a mixed-signal platform. Figure 2.14 elaborates on our goal in presenting this work. One just needs to pull up the Xcos structure from our openly available toolset, compile, and get the results back from silicon through a remote system. Moreover, a simulation engine in the toolset enables users to model and simulate the system and compare results with those from the FPAA SoC.

A fundamental implementation of silicon neuron circuit [26] was built from the classic HH equations. There have been other approaches [28, 29, 30, 19] that have been adapted from this classic implementation, translating from the equations to hardware, with additional voltage-dependent ion channel based modeling [27]. A comprehensive set of differ-

ent circuit models and solutions to build neurons on silicon has been discussed in [18].

Comparing our implementation to one in a digital setting, as has been shown in FPGAs [55, 56, 57, 58], we can turn to a reduced neuron model, computationally a digital concept [59] where the number of state variables is reduced to mostly two. This is due to the ease with which two state variables can be studied for its nonlinear dynamics and one can observe the phase plane behavior [35] to observe different bifurcations depending on the chosen parameter. In the circuit, the four state variables can be brought down to two, by tuning the Na^+ channel, where τ_m moves to 0, τ_h tends to infinity, thereby making the Na channel just an amplifier with a finite gain.

The concept of multiplication-accumulation (MAC) [60], a standard operation which adds a product of two numbers to an accumulator, especially used in digital computing to compare the performance of digital signal processors, helps to throw light on the efficiency of the neuron models. The Integrate and Fire neuron [22] can be built from this, with the same number of devices, though with lower number of parameters, with 1-4 MACs optimized for a digital implementation, while the models like Izhikevich [24] and FitzHugh-Nagumo model involve 30-60 MAC per sample as they are two state variable systems. However, they may not accurately represent the behavior of neurons like the HHneuron model, which emulates the ion channel dynamics.

Though the HHneuron model may involve a larger number of Ordinary Differential Equations (ODEs), with numbers in the range of 1000 MACs per computational iteration for a digital implementation, it can be implemented in our analog formulation with a few devices and parameters unlike the full parameter set used in the HH equations, thereby reducing redundancy. The bias parameters are directly relatable to observable time constants in the dynamics. Hence, such models that truly mimic the characteristics of a biological neuron are especially useful in applications that require the preservation of neurocomputational features and can be used to build neuromorphic systems of spiking neurons.

CHAPTER 3

MODELING, SIMULATION AND IMPLEMENTATION OF CIRCUIT ELEMENTS IN AN OPEN-SOURCE TOOL SET

3.1 Modeling circuit elements for the FPAA

Field Programmable Analog Arrays (FPAA) [2], with their configurability and programmability, have enabled the design and implementation of large-scale mixed-signal systems, with a diverse set of applications, including signal processing and neuromorphic computing. The need for a single platform facilitating a hardware-software codesign is critical in implementing such applications [3], and providing flexibility.

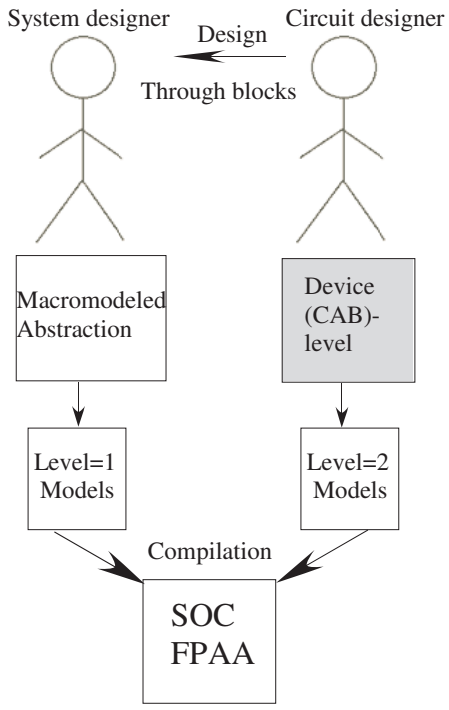


Figure 3.1: The concept of implementing the circuit/system idea from the designer's mind to simulating and obtaining results from the FPAA. It is essential to have a support for circuit design as much as a system design since the number of parameters are much higher in system design than circuit design and is significant to realize the dynamics of the system. This low-level circuit design is highlighted.

A simulator based on real data from silicon with a simple platform having minimal parameters for a comprehensive analysis, built in Xcos/Scilab is shown in this chapter. The infrastructure [12] enables one to compile the desired circuits and systems on the FPAA to obtain silicon data and simulate them at the same time to analyze and go back and forth between the simulation and data to obtain different dynamics, done either in a SPICE-similar setting with the fundamental components like transistors, capacitors, amplifiers etc, which is the primary focus of this work or build it as a system implementing the Ordinary Differential Equations (ODEs) shown in [61], depending on the user's choice, thereby incorporating the perspectives of both the circuit and system designers.

Conventional circuit simulators like SPICE and Cadence employ models that have a large number of parameters and switch between different levels of models like BSIM (e.g. level=44) [62] [63] to obtain different levels of accuracy and different computation times. However, the basic components built in our toolset are based on the EKV model, with a minimal number of parameters.

An accurate and a relatively fast simulator is essential [64] for a circuit designer, with the user having the flexibility to intuitively tune the parameters. The W/L ratios being fixed for the devices on the FPAA is abstracted through parameters like threshold current for the transistor, which is a more relevant term in this discussion; one need not hypothesize the required W/L, which is generally a significant factor to be decided on [65] in SPICE simulations.

Due to the availability of the remote system and open-source toolset encapsulated in a virtual machine [12], any user can compile their required circuits and systems, send it to a remote system that programs the design, measures the output from the given input, and sends them the results from the FPAA back in an email [33]. A design cycle of waiting for a period of time for the fabrication, as in the case of a custom chip, is not required here and we can readily take data due to access to the FPAA.

Reconfigurability is obtained by programming the charges on the floating gates on the

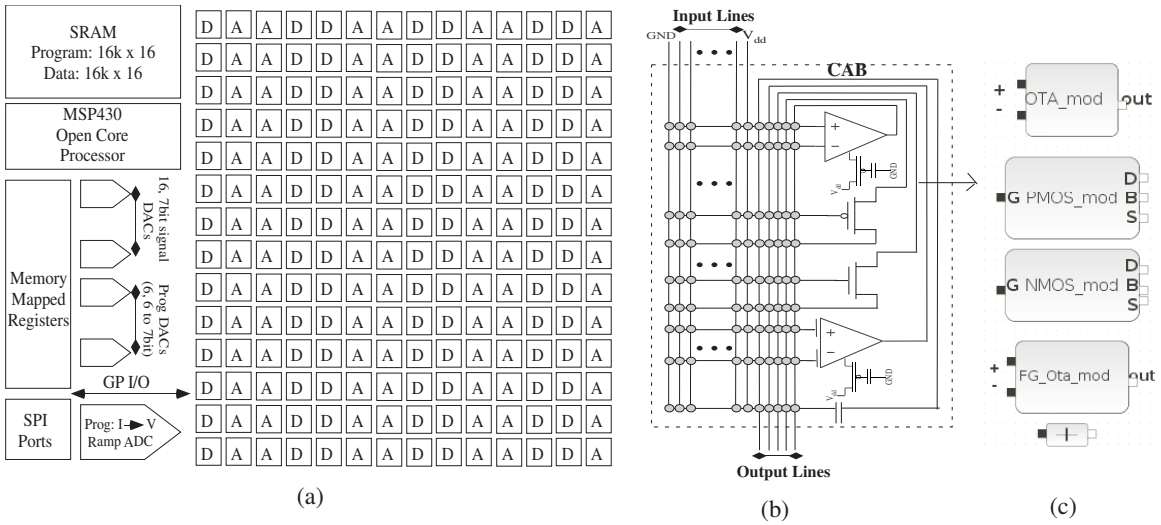


Figure 3.2: FPAA and tool infrastructure overview. (a) FPAA fabric consists of CABs, CLBs and other peripherals, fabricated on a 350nm process. (b) The CAB components consisting of nFET, pFET, OTA, FGOTA and capacitor, with the FGpFETs is shown in one CAB. (c) The Xcos palette with the blocks simulated with modelica corresponding to each CAB component is shown.

FPAA. Hence, it is important to have an idea of the bias currents and reference DC levels to be used. One can design the system beforehand using the simulation models in the toolset, set the different parameters in the system and then use the remote system to try to replicate the results.

This chapter elucidates our circuit models and their close agreement with measurements obtained from the FPAA. Section 3.2 introduces the concept of level=1 and level=2 models, while Section 3.3 describes the FPAA and the tool infrastructure used for obtaining the results. We discuss the characterization of the CAB components as level=2 blocks in Section 3.4. Section 3.5 introduces floating gate components. Validation of the level=2 blocks and building upon these form systems are further discussed in Section 3.6. Sections 3.7 and 3.8 discuss the modeling of temperature dependence of circuits using the simulator. Section 3.9 concludes the discussion, comparing this work and a conventional simulator's computation time.

3.2 Level=1 and Level=2 Models

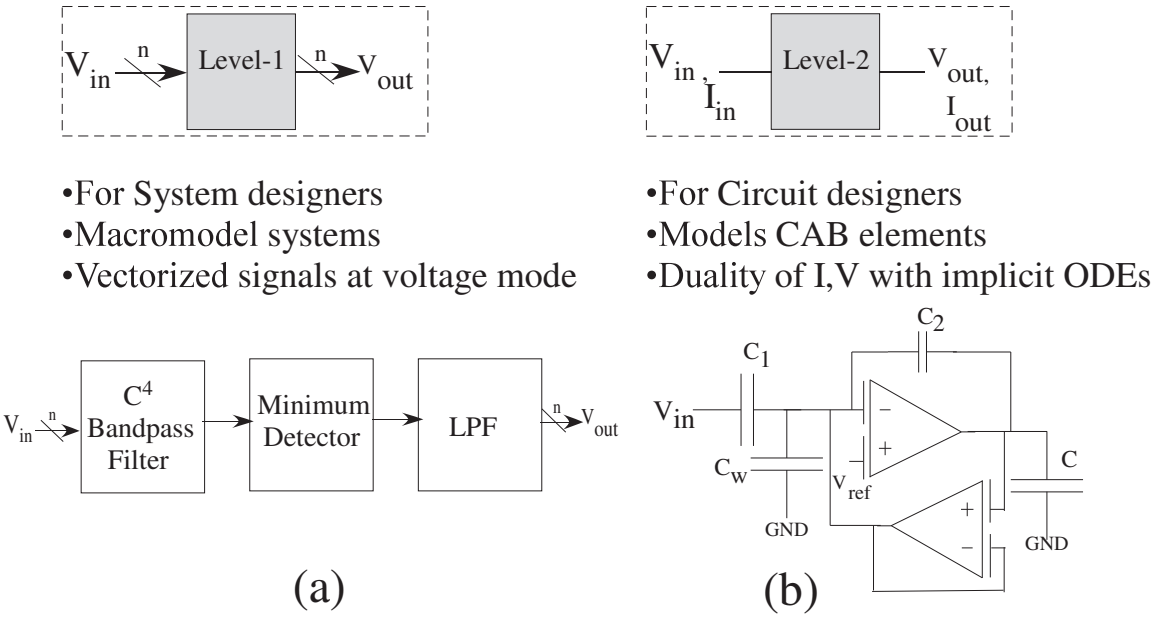


Figure 3.3: Level=1 and Level=2 blocks. (a) Level=1 blocks are used for emulating the system behavior. Level=1 operates in voltage mode and serves for designing systems at a high level. The analog front-end of a speech processing system is shown as an example of a level=1 system. (b) Level=2 blocks are used for emulating the circuit behavior. A C^4 bandpass filter is shown as an example of a level=2 circuit.

The tool infrastructure built in Scilab and Xcos [12] enables us to perform simulations and take experimental measurements for compiling various circuits and systems. The different blocks in the tools are classified as primarily level=1 and level=2 blocks. The dataflow tool representation allows for a heterogenous mixture of coarser-grain system concepts (Level=1) as well as fine-grain circuit blocks (Level=2).

Level=1 blocks aim to model systems, where the focus is on a macromodeled simulation [66], defining the ODEs and algebraic equations for the system operating with voltage inputs and outputs, with vectorized inputs and outputs. They are similar to a typical data flow graph, which defines the equations of the system, as introduced in [12] and [61].

The level=2 models aim to model the circuits and individual CAB components, through a Modelica based Scilab framework, implementing implicit ODEs with both voltage or cur-

rent mode. The goal of level=2 modeling would be to design, test, model, and build level=1 blocks that can be used by system designers. Each block represents a circuit element, either an element in the CAB (e.g. transistors, OTA) or a circuit block. Every line represents a single circuit connection providing current–voltage constraints. They primarily model the individual CAB elements, in a SPICE-like setting.

3.3 Integration with the tool infrastructure

The level=2 models are built through a Modelica based Scilab framework to implement EKV models. Modelica, an object oriented language to write and solve the ODEs and model system dynamics, enables directly implementing electrical systems as physical-level components in Xcos [67]. This modeling is similar to SPICE simulations, providing better intuition than breaking down the ODEs of the system and modeling each operation individually and integrating to form a system. A standard Runge-Kutta-4(5) (RK-45) [68] solver is used in our simulations, although other solvers can be used instead. The electrical blocks have voltage and current values in their input and output ports, which are specified in the computational function associated with the Xcos block.

A component-based approach [69] is followed here where the behavior for all the components we use has been described in the computational function associated with that block. During compilation, the Xcos/Scilab converts the Xcos schematic to a set of files. All the instances are called in a netlist-like Modelica file, which describes the connections between the nodes of all the components present in the diagram through the command called connect, analogous to an electrical netlist. The other Modelica file describes the equations of the system, obtained by following KCL and KVL at the nodes. These Modelica .mo files are compiled to a C code that describes the flow of the simulation through the event-driven flags, which further define the flow of operations from the initialization of the ODEs to running for the time period of the simulation defined by the period of the clock pulse generated and applied at regular periodic intervals.

3.4 Characterization of CAB components

The CAB components comprise nFETs, pFETs, OTAs, FGOTAs, T-gates and current mirrors. The EKV model [70] is used to model the transistors, based on fundamental device physics properties. It covers subthreshold to above threshold regions accurately [71]. FG capacitive coupling effects are also modeled, which helps in accurately simulating FG circuits and systems [5].

3.4.1 Transistors: nFET, pFET

The measurements are taken from a golden nFET and pFET, whose position is fixed on a particular CAB on the FPAA IC, to get uniform characteristics and to use the same parameters while building other circuits and systems from these transistors. To identify what terminal voltages need to be swept to determine the required parameters for the FETs, we look at the simplified EKV model in the subthreshold and above threshold regions [72].

The version of the EKV model used is shown in the table in Figure 3.4(a), where I_{th} is the current at threshold, V_{T0} is the threshold voltage, U_t is the thermal voltage, κ is the fractional change in the surface potential, V_d is the drain voltage, V_b the bulk voltage, V_g the gate voltage and V_s is the source voltage. The source voltage is fixed at a V_{dd} of 2.5 V for the pFET and ground for the nFET while V_g and V_d are swept to determine the parameters for the EKV model. These parameters are substituted in the simulation model to get a close overlap with the data.

Looking at the equations in Figure 3.4, one can identify the slopes to be determined to extract the necessary parameters. The drain current measurements against the gate voltage sweep are performed and fit to the EKV model to determine I_{th} , V_{T0} and κ, while the drain current measurements against the drain voltage sweep determine the σ value. These values are then substituted in the Modelica models. Figure 3.4 shows a close overlap between the data and the simulations.

	nFET	pFET
EKV	$I_{th}\ln^2(1+e^{(\kappa(V_g-V_{T0})-V_s+\sigma V_d)/2U_T})$ $-\ln^2(1+e^{(\kappa(V_g-V_{T0})-V_d+\sigma V_s)/2U_T})$	$I_{th}\ln^2(1+e^{(\kappa(V_b-V_g-V_{T0})-(V_b-V_s)+\sigma(V_b-V_d))/2U_T})$ $-\ln^2(1+e^{(\kappa(V_b-V_g-V_{T0})-(V_b-V_d)+\sigma(V_b-V_s))/2U_T})$
For sub V_{T0}: Ohmic Saturation	$I_{th}e^{\kappa(V_g-V_{T0})/U_T}\left(e^{-V_s/U_T}-e^{-V_d/U_T}\right)$ $I_{th}e^{(\kappa(V_g-V_{T0})-V_s+\sigma V_d)/U_T}$	$I_{th}e^{\kappa(V_b-V_g-V_{T0})/U_T}\left(e^{-(V_b-V_s)/U_T}-e^{-(V_b-V_d)/U_T}\right)$ $I_{th}e^{(\kappa(V_b-V_g-V_{T0})-(V_b-V_s)+\sigma(V_b-V_d))/U_T}$
For Above V_{T0}: Ohmic Saturation	$\frac{I_{th}}{4U_T^2}\left((\kappa(V_g-V_{T0})-V_s)^2-(\kappa(V_g-V_{T0})-V_d)^2\right)$ $\frac{I_{th}}{4U_T^2}(\kappa(V_g-V_{T0})-V_s+\sigma V_d)^2$	$\frac{I_{th}}{4U_T^2}\left((\kappa(V_b-V_g-V_{T0})-(V_b-V_s))^2-(\kappa(V_b-V_g-V_{T0})-(V_b-V_d))^2\right)$ $\frac{I_{th}}{4U_T^2}(\kappa(V_b-V_g-V_{T0})-(V_b-V_s)+\sigma(V_b-V_d))^2$

(a)

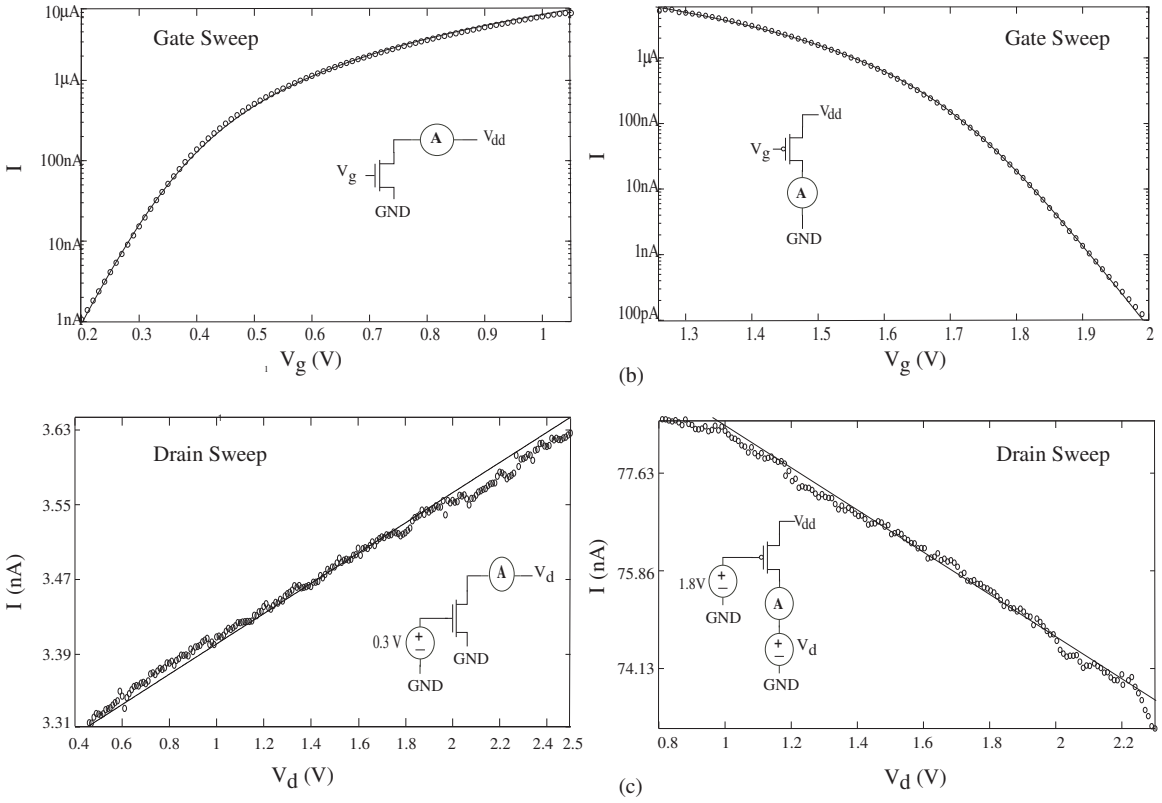
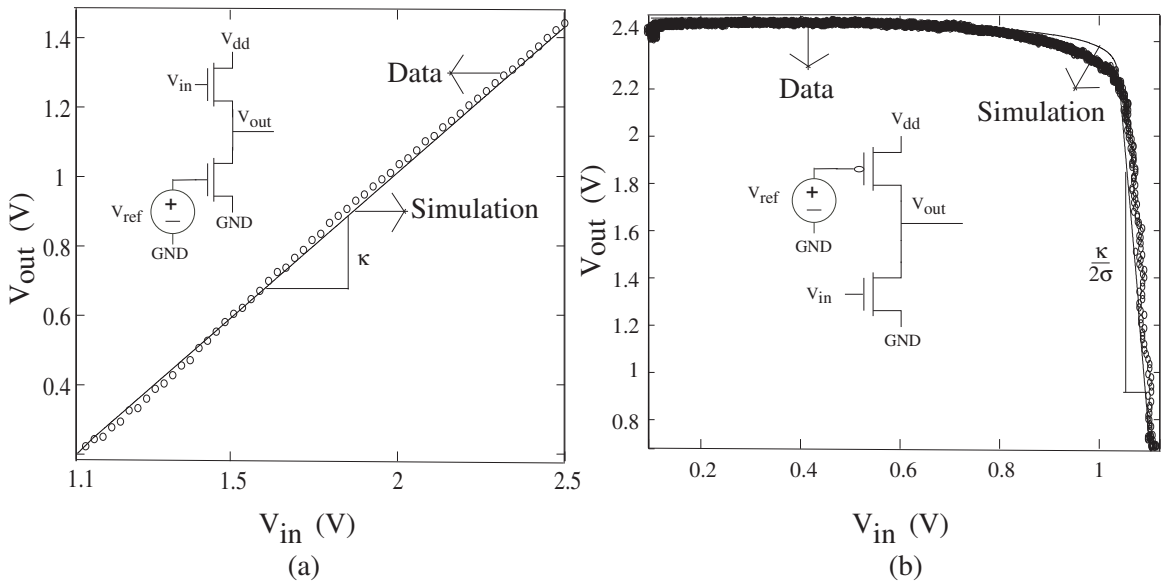


Figure 3.4: Simulated and measured current as a function of Gate voltage and drain voltage in log scale, at a V_{dd} of 2.5V. The measured data is represented as the fine discrete points while the simulation is the thick continuous line. (a) The version of the EKV model for the level=2 models of the transistors is shown. (b) The subthreshold region with a slope of $\frac{\kappa}{U_T}$ and the above threshold regions can be observed. The parameters obtained from the EKV curve-fit on the gate sweep data, used for the simulation are I_{th} of 53.58nA, V_{T0} of 0.32V and κ of 0.84 for the nFET and I_{th} of 111.20nA, V_{T0} of 0.75V and κ of 0.76 for the pFET. (c) The drain voltage is swept and the gate voltage is at 0.3V for the nFET and source voltage is fixed at a V_{dd} of 2.5V for pFET, to measure the slope in the sub-threshold region. σ extracted from the data, used for the simulation is 0.00039 and 0.0049 for nFET and pFET respectively.



	κ	I_{th}	V_{TO}	σ
pFET	0.679	111.84nA	0.866V	0.0049
nFET	0.808	53.58nA	0.313V	0.00039

(c)

Figure 3.5: Verifying and building circuits using level=2 models. (a) Simulated and measured Output voltage as a function of input voltage for a nFET source follower. A gain of κ can be observed due to the capacitive coupling. The nFET is biased at a reference voltage of 500mV. (b) Simulated and measured Output voltage as a function of input voltage for a common source amplifier. A finite gain of $\frac{\kappa}{2\sigma}$ is observed. (c) The parameters used for the simulation of the transistors in the amplifier are shown.

3.4.2 Building amplifiers from the transistors

Using the pFET and nFET models introduced in Subsection 3.4.1, experiments are performed for a nFET source follower and a common source amplifier. In the common drain amplifier, the current flows through the nFETs, determined by the bias voltage, V_{ref}. The source voltage follows the gate terminal, to which the input voltage is swept, at a gain of κ. The variation of κ with the input voltage is seen here and the depletion capacitance is almost a constant since the FET is in the subthreshold region. Only the gain of κ is highlighted in the transfer curves. The offset of κV_{ref} is not shown due to the voltage offset from the output buffer used at the output of the experimental setup. Assuming subthreshold

operation, large-scale analysis yields

$$I_{th}e^{(\kappa(V_{in}-V_{T0})-V_{out})/U_T} = I_{th}e^{(\kappa(V_{ref}-V_{T0}))/U_T}, \tag{3.1}$$

$$V_{out} = \kappa(V_{in} - V_{ref}). \tag{3.2}$$

The common source amplifier has the pFET biased to 1 V, which sets the bias current through the FETs. Again doing a large-signal analysis,

$$\begin{aligned} &I_{thp}e^{\kappa(V_{dd}-V_{ref}-V_{T0})+\sigma_p(V_{dd}-V_{out})/U_T} = \\ &I_{thn}e^{\kappa(V_{in}-V_{T0})+\sigma_n V_{out}/U_T} \end{aligned} \tag{3.3}$$

Assuming identical FETs,

$$\kappa(V_{dd} - V_{ref}) + \sigma_p V_{dd} = (\sigma_p + \sigma_n)V_{out} + \kappa V_{in},$$

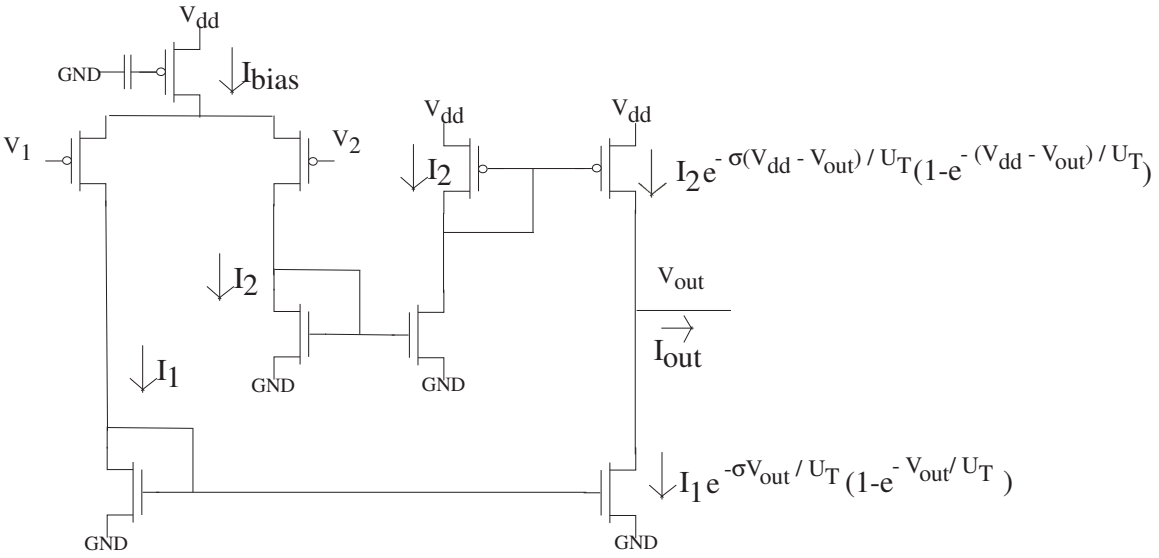
$$V_{out} = \frac{-\kappa}{\sigma_p + \sigma_n}V_{in} + V_{constant} \tag{3.4}$$

Hence, a high gain of $\kappa/2\sigma$ is observed in the transfer curve of V_{out} vs V_{in}, as shown in Figure 3.5.

3.4.3 OTA

Another CAB component, the OTA, has been built as a level=2 model. The schematic of the 9 transistor differential transconductance amplifier is shown in Figure 3.6. It consists of a pFET differential pair and a set of current mirrors. The bias current, I_{bias}, is set by the FG-pFET at the top, which acts as a current source for the amplifier [73]. Considering that the input pFET differential pair is in saturation, the I_{bias} is split into currents I_1 and I_2 in the input differential pair branches. The derivation from the exponential relationship

$$I_1 = I_{bias}(\tfrac{1}{2} + \tfrac{1}{2}\tanh\tfrac{\kappa(V_1 - V_2 + V_{offp})}{2U_T})$$
$$I_2 = I_{bias}(\tfrac{1}{2} - \tfrac{1}{2}\tanh\tfrac{\kappa(V_1 - V_2 + V_{offn})}{2U_T})$$



$$I_{out} = I_2(e^{-\sigma(V_{dd}-V_{out})/U_T}(1 - e^{-(V_{dd}-V_{out})/U_T})) - I_1(e^{-\sigma V_{out}/U_T}(1 - e^{-V_{out}/U_T}))$$

Figure 3.6: Schematic of the 9 transistor OTA in the CAB, with the I_{bias} current split into currents I_1 and I_2 in the input differential pair branches and reflected as $I_2^{"}$ and $I_1^{"}$, due to the output transistors.

to the tanh relation follows from[1] and through intuition from the graph and its solution. Since the drain current is dependent on the drain voltage in saturation due to the Early effect, the effect of σ is significant, especially in the output transistors and included in the output relation from I_{out} to V_{out}. The current is not reflected out equally from the current mirrors due to mismatches. The V_{T0} and I_{th} differences between the individual transistors are responsible for the voltage and current offsets, which are also included in the equations.

Different experiments are performed to study the behavior of the OTA, as shown in Figure 3.7. It shows the OTA as an element in the CAB, which is built from the level=2 Modelica simulation models. Each parameter of the nFET and pFET like V_{T0}, I_{th} and σ can be manipulated to vary the effect of mismatches and offsets, especially from the current mirrors. Figure 3.7(b) - Figure 3.7(d) shows the characteristics of an OTA built as a level=2

[1] [31], Chapter 5, pp. 68

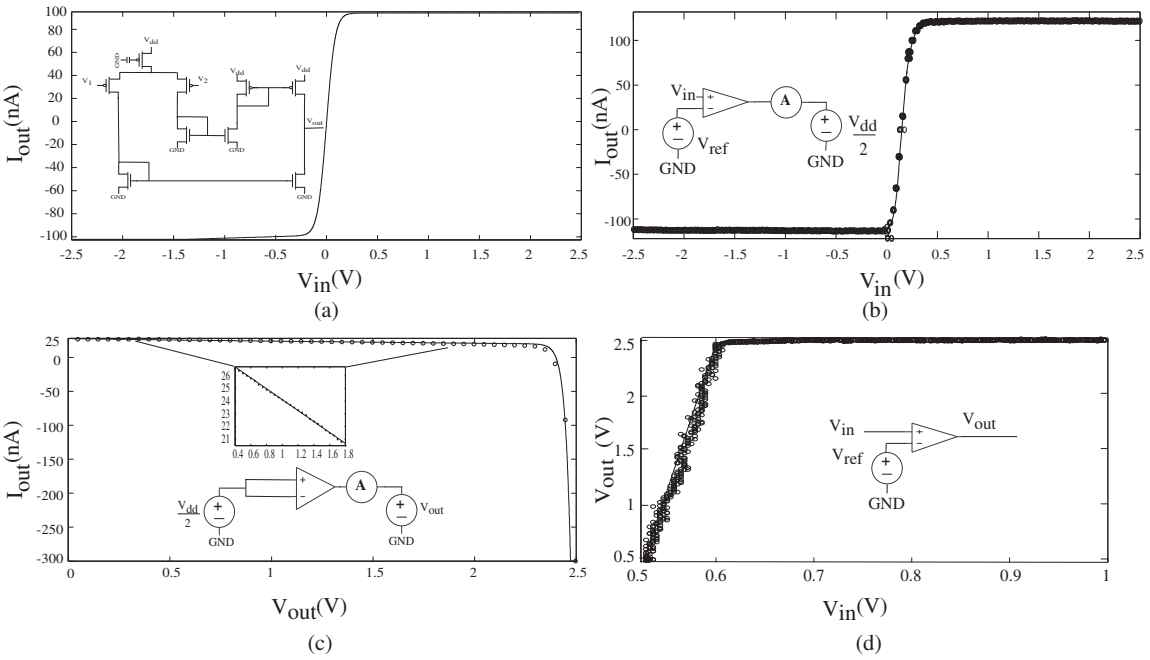
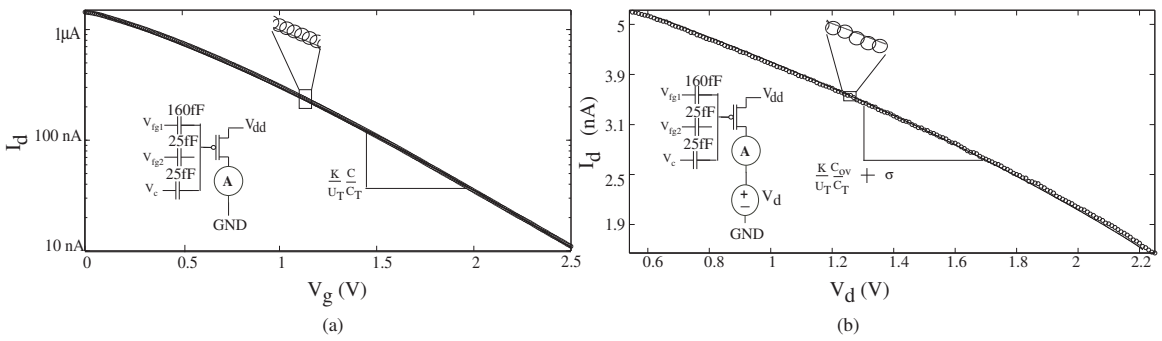


Figure 3.7: Simulated and measured characteristics of the OTA biased at 100nA. (a) Output current as a function of differential input voltage. The 9 transistor OTA is built using the level=2 modelica transistors. The tanh behavior is observed and the offset in the currents between the positive and negative inputs can be seen due to the pFET current mirror. (b) Output current as a function of differential input voltage for an OTA. The transconductance can be measured from the slope which shows a tanh behavior. The current offsets from the biased current can be seen due to the threshold mismatch in the current mirrors in the 9T OTA structure. (c) Output current as a function of output voltage. The exponential dependency of the current is seen through an upper limit near V_{dd} above which the current decreases. The finite output conductance of the OTA is observed in the finite slope in the midrange, shown in the inset zooming into the midrange. (d) Voltage Transfer Curve, Output voltage as a function of input voltage. The positive terminal is fixed at 500mV.

model. A close agreement can be seen between the experimental measurements and the simulations.

While the input voltage is swept, the output voltage is kept at midrail and the ammeter is connected in series with the output node of the OTA. As expected, we can see a tanh behavior in the I_{out} curve as a function of V_{in}. The transconductance, g_m, as a function of I_{bias} is measured from the slope of the curve. The output current to which the OTA settles at the upper and lower end is not identical due to the finite current gain from the current mirror and is reflected as an exponential variation in the threshold voltage differences.



	κ	I_{th}	V_{TO}	σ
FG pFET	0.712	512.36nA	0.854V	0.0071

(c)

Figure 3.8: Simulated and measured current as a function of Gate voltage and drain voltage of a FGpFET, from the MITE block in log scale. (a) The FG input depending on the input capacitances to be swept can be chosen. The capacitive coupling can be seen in the slope and the range of currents obtained as the input voltage is swept. The source voltage is fixed at a V_{dd} of 2.5V. A slope of $\frac{\kappa}{U_T}\frac{C}{C_T}$ can be observed. (b) The Source voltage is fixed at a V_{dd} of 2.5V. A slope of $\frac{\kappa}{U_T}\frac{C_{ov}}{C_T} + \sigma_p$ can be observed. (c) The parameters obtained through curve-fit from the data, which are then used for the simulations are shown.

V_1 and V_2 are fixed to observe the effect of V_{out} on I_{out}. The finite output conductance is shown in the zoomed in plot and can be measured from the slope of the curve, over a range of the output voltage. This output resistance is due to the early effect imposed by the finite σ of the output transistors. We can see that the current drops exponentially near the V_{dd} of 2.5 V, which is due to the point when the transistors transition from the saturation region.

The voltage transfer curve of the OTA in the open loop configuration, wherein V_{out} is obtained as a function of V_{in}, is also measured. The slope of the curve gives the gain of the OTA, which is similar to A_v = g_m R_{out} , where g_m and R_{out} have already been obtained from the I_{out} vs V_{in} and I_{out} vs V_{out} curves respectively.

3.5 Floating gate components

A FG circuit is one in which the gate of the FET is electrically isolated and capacitively connected to the other nodes [5]. Since there is no DC path from the floating node to any fixed voltage, it is difficult to model an FG in conventional SPICE simulators, where it is modeled with a DC voltage at the node with a resistor. Charge is stored on the FG, and it can be modulated through electron tunneling and hot-electron injection. The FG programming is encapsulated in the infrastructure [14], abstracted away from the user.

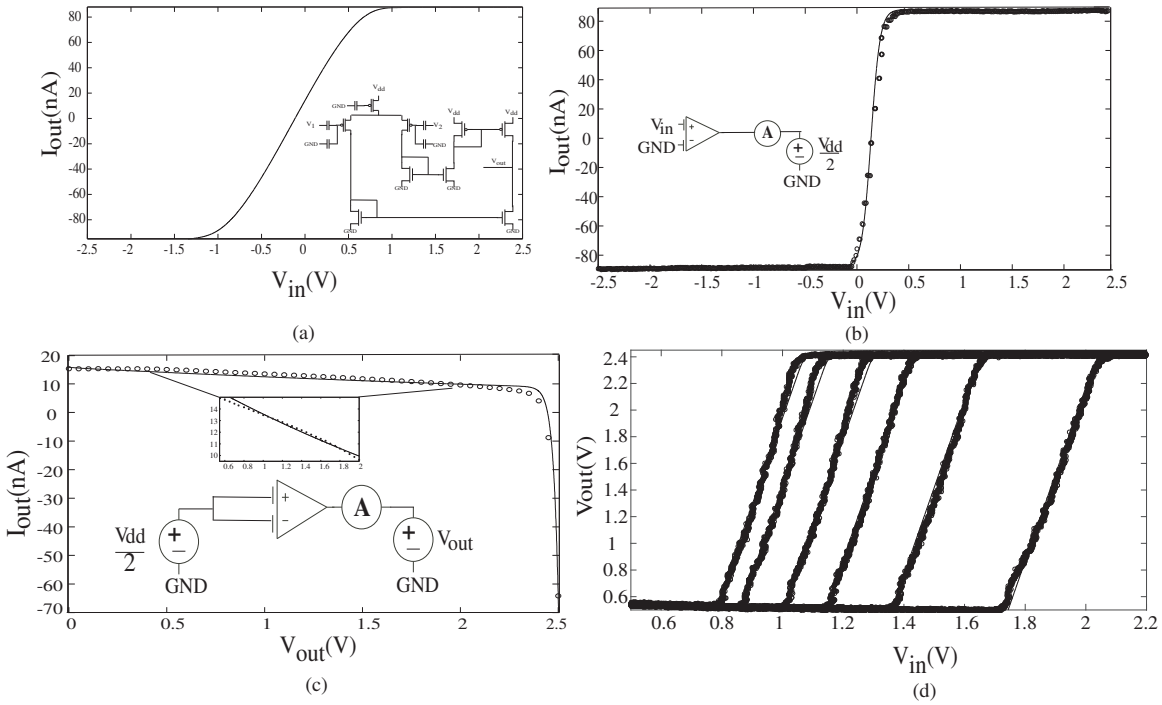
3.5.1 FGpFET

The input, routing, overlap, and oxide capacitances represented by C_1, C_w C_{ov}, C_{ox} respectively form a capacitive divider at the gate terminal with C_T being the total capacitance at the gate terminal.

$$C_T = C_1 + C_w + C_{ov} + C_{ox}(1 - \kappa)$$

$$V_{fg} = \frac{C_1}{C_T} V_g + \frac{C_{ov}}{C_T} V_d + \frac{C_w}{C_T} V_{dd}$$

Experimental measurements and simulations are performed for the FGpFET, from the Multiple-Input Translinear Element (MITE) block [9][11]. The effective value of κ seen is less compared with a pFET due to the capacitive coupling as shown in the above equations. V_{fg} is the effective FG voltage at the node due to the capacitive coupling from the gate node, with a voltage offset that depends on the initial charge stored on the node. Similar to the pFET, a gate sweep is performed to extract I_{th}, V_{T0}, and κ, and a drain sweep gives the σ value. The capacitor node to be swept can be chosen, thereby giving different capacitive coupling ratios from the gate terminal at the input to V_{fg}.



	g_m	R_{Out}
OTA	1486.5nA/V	233.4MΩ
FG OTA	285.5nA/V	277.81MΩ

(e)

Figure 3.9: Simulated and measured characteristics for an FGOTA. (a) The voltage offset is obtained for FGOTA built using the level=2 modelica FG transistors by setting the offsets at the input FGpFETs. (b) Output current as a function of differential input voltage for a level=2 FG OTA whose slope gives the transconductance and shows its tanh behavior and the linearity. (c) The output current as a function of output voltage shows the exponential dependency of the current, seen through an upper limit near V_{dd} above which the current decreases. The finite slope, shown in the inset zooming into the midrange shows the output conductance of the FGOTA. (d) Voltage Transfer Curve, Output voltage as a function of input voltage. The positive terminal is fixed at 1.5 V. The voltage offsets can be seen due to the offsets programmed at the input FGpFETs. The positive input was biased at 10nA current and the negative input bias was increased in steps of 10 from 10nA to 60nA. (e) The slopes of the curves show the g_m and R_{out} of the amplifiers.

3.5.2 FGOTA

The effective FG voltages are set by the capacitive divider at the inputs of the FGpFETS, at the differential amplifier input terminals:

$$V_{fgp} = \frac{C_1}{C_T} V_1 + V_{fgp0}$$

$$V_{fgn} = \frac{C_1}{C_T} V_1 + V_{fgn0}$$

The 9T FGOTA structure in Figure 3.9(a) built from level=2 models of FGpFETs and pFETs shows characteristics similar to that obtained from the FGOTA built as a level=2 model. In addition to the current source which can be programmed, the FGOTA has two FGpFETs at the two inputs which can be programmed [74]. Each parameter of the three programmable transistors in the 9T structure can be tweaked to get different linearities and transconductances ranges.

As shown in Figure 3.9(b), the tanh behavior and the improved linearity can be observed in the I_{out} vs V_{in} plot, whose slope gives the transconductance. The linear range of the FGOTA is higher than the OTA due to the capacitive divider at the two inputs. The drain induced barrier lowering effect (DIBL) is again responsible for the finite output conductance, measured from the slope of the I_{out} vs V_{out} curve, as shown in the zoomed in inset in Figure 3.9(c). and the sudden exponential drop in current can be seen as the output voltage approaches V_{dd}.

Another advantage of the FGOTA is that it helps to set a desired voltage offset which is programmable. The charges on the two FGpFETs at the inputs can be modulated, thereby producing the required voltage offset. The FGpFET that acts as the current source is at a fixed bias while the input FGpFET is tuned from 10nA to 60nA. These currents are mapped to voltages internally in the infrastructure. This tuning is responsible for the voltage shifts as observed in the V_{out} vs V_{in} voltage transfer curve in Figure 3.9(d).

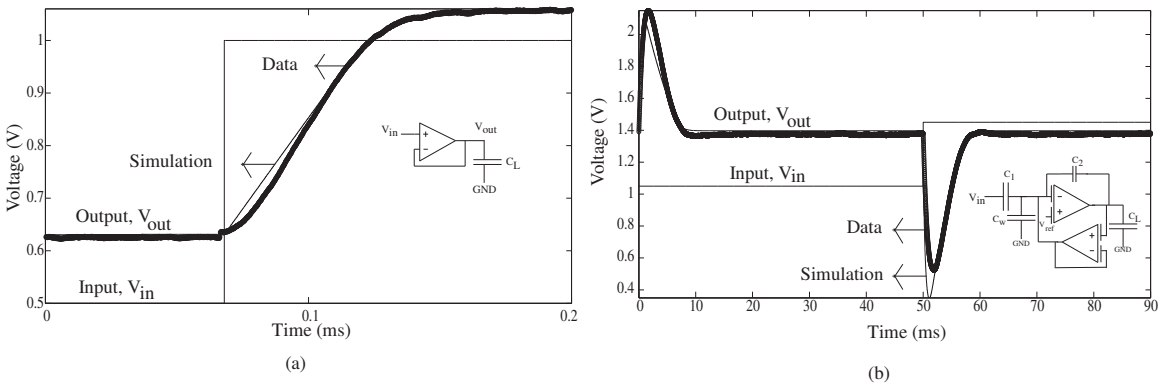


Figure 3.10: Simulated and measured Output Voltage as a function of input voltage for Continuous-time Filters. (a) Step response for a first-order low-pass filter. A step input is given to an OTA biased at 5nA, connected in a follower configuration . The extracted capacitance value, which comprises the parasitic and routing capacitance at the output of OTA for the required time constant is 460fF. (b) Step response for a Capacitively Coupled Current Conveyer, C^4 based bandpass filter. A step input from 1.05V to 1.45V is given to a FGOTA biased at 50nA, with a DC input bias, V_{ref} at 1.4V, while the feedback FGOTA is biased at 500pA, to get the required dynamics.

3.6 Verification and Building level=1 systems using level=2 models

To further verify and validate the models, in addition to the circuits built in Subsection 3.4.2, other circuits and systems are built from the level=2 models and the experimental measurements are compared with the simulations. The OTA is connected in a follower configuration, to act as a first order low pass filter, which is driving a capacitive load, considered to be the sum of the parasitic and routing capacitances at the output node of the OTA. An FGOTA could also be used here to get a better linearity. The response of the filter integrating the step input is close to the simulations, with the time constant, τ, being $2U_T\mathrm{C}/\kappa I_{bias}$,

$$\tau\frac{\mathrm{d}V_{out}}{\mathrm{d}t} = \frac{2U_T}{\kappa}\tanh\frac{\kappa(V_{in}-V_{out})}{2U_T},$$

Which when linearized,

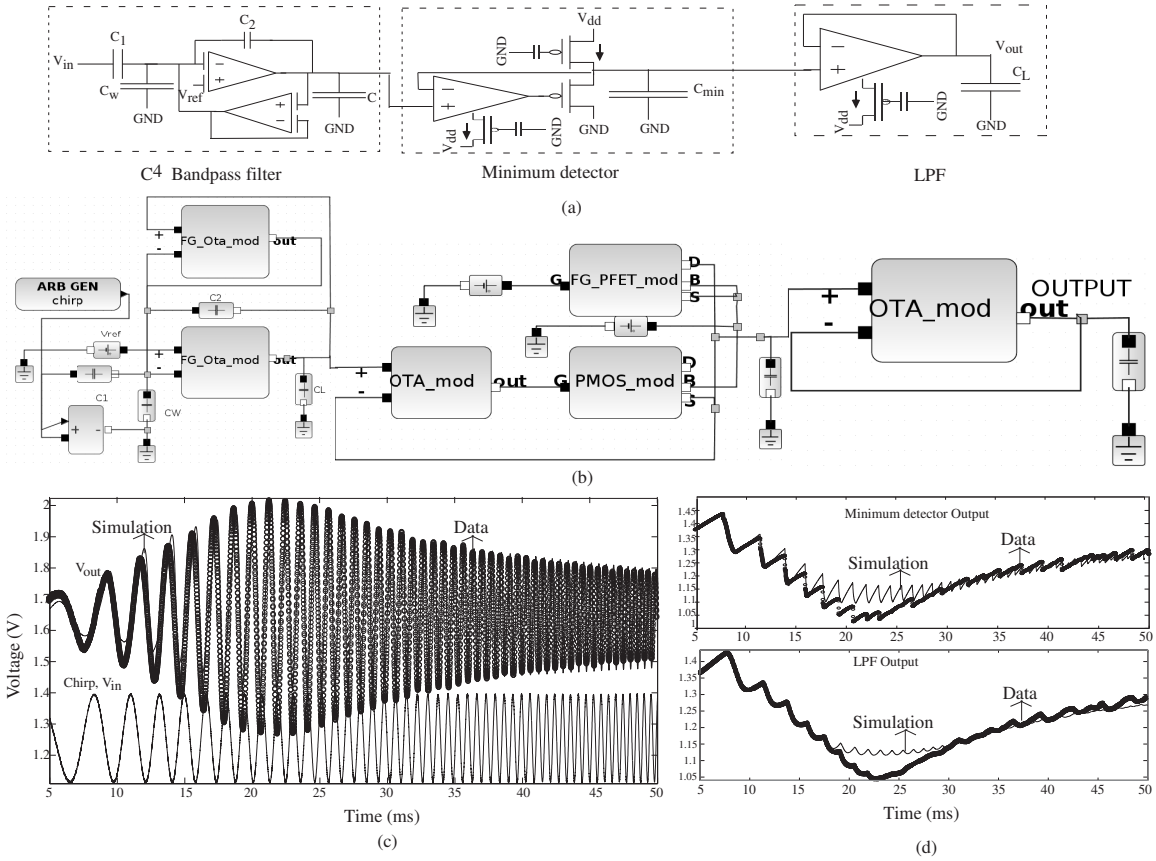


Figure 3.11: Simulated and measured dynamics for the analog front-end of a speech processing system. (a) The schematic of the system consisting of a C^4 bandpass filter, minimum detector and LPF are shown. (b) The Xcos/Scilab diagram with the level=2 models corresponding to each block, used for simulation is shown. (c) A chirp input between 1Hz and 20Khz with an offset of 200mV around 1.25V is applied at the input of the system. The C^4 limits the spectral band to a set of frequencies, set by the FGOTA in the gain stage biased at 300nA, while the feedback stage FGOTA is biased at 6nA. (d) The minimum detector tracks the minimum points while the LPF gives out a smooth result.

$$\tau \frac{\mathrm{d}V_{out}}{\mathrm{d}t} + V_{in} = V_{out}. \tag{3.5}$$

Consider the Capacitively Coupled Current Conveyer (C^4) based second-order bandpass filter [75]. The transconductances of the FGOTAs and the feedback and load capacitance set the time constants of the filter. A step input of 400mV around $V_{dd}/2$ is applied, and again there is a close overlap between the data and the simulation results. The equations for C^4 and its level=1 modeling is shown in [12].

Building on these components and systems, the analog front-end of a speech processing system is shown here in Figure 3.11 [2] [76]. It consists of the C^4 which selects a band of frequencies as determined by the bias currents set by the FGOTAs in the bandpass filters. This is fed to a minimum detector to track the amplitude at the minimum points on the envelope of the input waveform. The time constants are set by the load capacitances and the currents to which the FG-pFET and OTA are biased.

The bias current of the OTA in the minimum detector is tuned such that it stays in high gain, assuming the pFETs are in subthreshold saturation. The LPF at the output filters the ripples and smooths the minimum detector output. There is a close overlap in the C^4 output, while the dynamics of the minimum detector is very close in both the data and simulations where it closely follows the minimas. An exact fit is not obtained though, in the middle time range, since there is a slight DC offset observed in the output. Through this system, we have further emphasized that we can get an accurate correspondence between the simulations and the experimental data for larger systems.

3.7 Modeling Temperature Dependence

Many of the systems that can benefit from the computational power of an FPAA need to operate over a range of environments and temperatures. For instance, modern ubiquitous medical health assessment systems use physiologic signals collected from ambulatory subjects during daily outdoor activities [77]. Likewise, point-of-care diagnostics platforms aiming to achieve lab-quality tests in low-resource settings operate in environments with varying ambient temperatures [78, 79]. Furthermore, in assisted living applications, sensor networks are used to identify and track the daily activities of elderly residents in outdoor settings. For outdoor applications where temperature is not stable, one of the critical performance metrics affecting the computation accuracy of the FPAA would be, therefore, robustness against temperature variations. Hence, another application of this Level=2 simulator is to analyse the temperature dependencies of different circuits and the results are com-

pared with measurement data obtained from the FPAA. Temperature measurements were performed using a ZPlus (Cincinnati Sub-Zero Products LLC, Sharonville, OH, USA) temperature chamber. For each temperature value, 15 min is allowed to ensure that the FPAA die reaches the desired temperature value.

Based on the EKV model, the channel current for a pMOS and an nMOS transistor are governed by the following equations

$$\begin{aligned} I_d &= I_{thnmos} \ln^2\left(1 + e^{(\kappa(V_g - V_{T0n}) - (V_s) + \sigma(V_d))/2U_T}\right) - I_{thnmos} \ln^2\left(1 + e^{(\kappa(V_g - V_{T0n}) - (V_d) + \sigma(V_s))/2U_T}\right) \\ I_d &= I_{thpmos} \ln^2\left(1 + e^{(\kappa(V_{DD} - V_g - V_{T0p}) - (V_{DD} - V_s) + \sigma(V_{DD} - V_d))/2U_T}\right) - I_{thpmos} \ln^2\left(1 + e^{(\kappa(V_{DD} - V_g - V_{T0p}) - (V_{DD} - V_d) + \sigma(V_{DD} - V_s))/2U_T}\right) \end{aligned} \tag{3.6}$$

where I_{thnmos} and I_{thpmos} are specific currents. These are defined at threshold voltages given by $2\mu_{nmos}C_{ox}(W/L)U_T^2/\kappa$ and $2\mu_{pmos}C_{ox}(W/L)U_T^2/\kappa$, respectively, σ is the drain-induced barrier lowering coefficient; and V_d, V_s, V_{T0p}, V_{T0n}, and U_T are the drain, source, pMOS zero-bias threshold, nMOS zero-bias threshold, and thermal voltages, respectively. Temperature dependence of I_d in (Equation 3.6) arises from threshold voltages, I_{th}, and explicit U_T or KT/q . The dependence of threshold voltages on temperature could be modeled using $A_1 + A_2U_T$. I_{th} has dependence on temperature due to the mobility (μ) and presence of U_T^2, which could be modeled using $I_{thr}(\frac{T}{T_r})^a$, where $a = 0.5$. From measured data, shown in the Table 3.1, which shows

$$\frac{dI_{th}}{dT}/I_{thr} = \frac{a}{T_r}, \tag{3.7}$$

it can be seen that $a \approx 0.6$. Here, T_r is the reference temperature (298 K). The variation of V_{To} is as follows

$$\frac{dV_{to}}{dT} = -\frac{2}{\kappa} \ln\left(\frac{N_D}{\sqrt{N_c N_v}}\right) \frac{dU_T}{dT}, \tag{3.8}$$

where N_c and N_v are the effective density of electrons and holes in conduction and valence band, respectively. Their dependence on temperature is $T^{3/2}$ [80]. This is small compared to the linear dependence to U_T term. It should be noted that the model in [70] has more parameters and is much more generalized. In the case of (3.4), the model has a reduced

Table 3.1: Comparison of simulated and measured data: Percentage change over 60 °C.

Device Parameter	Measured				Simulated			
	Threshold Voltage		I_{th}		Threshold Voltage		I_{th}	
pFET	-0.28%	-1.7 mV/C	-0.2%	2000 ppm/C	-0.27%	-1.7 mV/C	-0.17%	1700 ppm/C
nFET	-0.26%	-1.1 mV/C	-0.24%	2400 ppm/C	-0.22%	-0.95 mV/C	-0.17%	1700 ppm/C

number of parameters, which allows for faster simulation, with the ability to closely predict data from the FPAA.

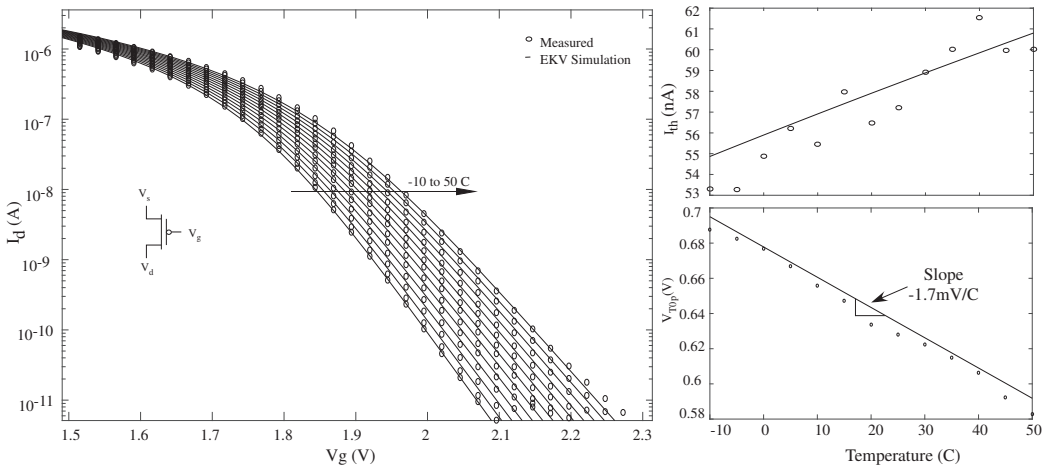


Figure 3.12: I_d vs V_g transfer characteristics of PMOS over Temperature. EKV modeling is used for modeling the transfer characteristics of a PMOS. I_{th}, current at threshold voltage, and threshold voltage are also plotted over temperature. The simulated values are consistent with the measured values. The I_{th} and threshold voltages are extracted by curve fitting onto to the output current in both measurement and simulation to be consistent.

This model has been integrated as a part of the Scilab/Xcos environment [15]. Figure 3.12 shows measurement of pFET compiled on to the FPAA and simulation performed using the EKV model. The tool incorporates the above variation in U_T, threshold voltage, and current at threshold voltage (I_{th}). Figure 3.12 compares these variations between the simulated model and the measured results over a change of 60 °C. The measurements are silicon data obtained from the FPAA fabricated in 350 nm technology. Scaling of floating gates would follow similar trends as show in [81]. Simulation parameters have to adapted as the process scales to match the data from silicon.

Using the above model for simulation, similar measurements and simulations were performed for an nFET. The variation in threshold voltage and current at threshold voltage

(I_{th}) for the devices are summarized in the Table 3.1. These parameters are extracted from the transfer characteristics using a EKV curve fit program in Scilab [82] with varying U_T. The variations are shown as percentage changes in the parameters from its value at room temperature over 60 ° C.

The consistency between the measurements and the model allows us to predict the first-order behavior of circuits and systems compiled on the FPAA with temperature, thereby enabling temperature robust circuits and system design.

3.8 Temperature Dependence of Simple Single Ended Circuits

The models developed in the previous section helps predict the behavior of circuits and systems on the FPAA. To illustrate, a common source amplifier, shown in the Figure 3.13, has a gain which is constant over temperature [83]. The EKV model shown in (3.4) could be reduced to following set of equations when $I_{sat} \ll I_{th}$:

$$I_d = I_{thnmos} e^{(\kappa(V_g - V_{T0n}) - V_s + \sigma_{nmos} V_d)/U_T}$$

$$I_d = I_{thpmos} e^{(\kappa(V_{DD} - V_g - V_{T0p}) - (V_{DD} - V_s) + \sigma_{pmos} V_d)/U_T}$$

Equating the channel current for nMOS and pMOS we have

$$I_{thnmos} e^{(\kappa(V_{bias} - V_{T0n}) + \sigma_{nmos} V_{out})/U_T} = I_{thpmos} e^{(\kappa(V_{DD} - V_{in} - V_{T0p}) + \sigma_{pmos}(V_{DD} - V_{out}))/U_T}$$

$$V_{out} = \frac{-\kappa}{\sigma_{nmos} + \sigma_{pmos}} V_{in} + V_{offset} \tag{3.9}$$

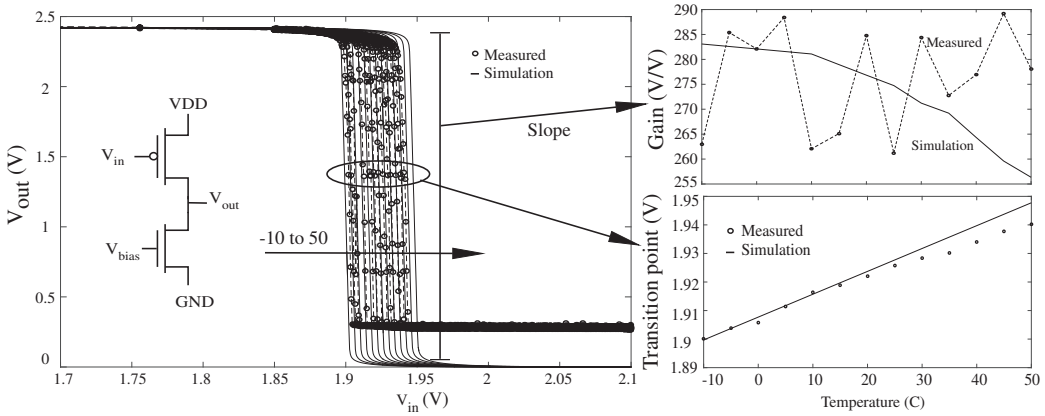


Figure 3.13: Transfer function of a common source amplifier, measured and simulated using the models developed in the previous section, with temperature. The slope, and hence the gain of the common source amplifier, is also plotted. Slope is constant with temperature. The transition of V_{out} with V_{in} for different temperature is also shown. The transition changes over temperature because the threshold voltage of nMOS and pMOS varies differently.

In (Equation 3.9), the offset V_{offset} is a manifestation of U_T, threshold voltage difference between nFET (V_{T0n}) and pFET (V_{T0p}), and $log(\frac{I_{thnmos}}{I_{thpmos}})$. Figure 3.13 shows the transfer characteristics of a common source amplifier, measured on the FPAA and simulated using the models developed before, with pFET input. As seen in Figure 3.13, the slope of the simulation and measurement remains relatively constant. The variation in the transition points of the transfer characteristics corresponding to different temperatures is associated with the offset (V_{offset}) term that depends on U_T, and also because the threshold voltage of pFETs and nFETs vary differently over temperature.

For temperature-robust transition points and gain, the circuit could be designed to use the same type of FET devices. The transfer characteristics of a two pFET-based common drain is given by

$$I_{thpmos}e^{(\kappa(V_{DD}-V_{bias}-V_{T0p})+\sigma_{pmos}(V_{DD}-V_{out}))/U_T} = I_{thpmos}e^{(\kappa(V_{DD}-V_{in}-V_{T0p})-(V_{DD}-V_{out}))/U_T}$$
$$V_{out} = \frac{\kappa}{1+\sigma_{pmos}}V_{in} - \frac{\kappa V_{bias}}{\sigma_{pmos}+1} + V_{DD} \tag{5}$$

Thus the slope of the transfer characteristics is $\approx \kappa$, since $\sigma_{pmos} << 1$, which is invari-

ant over temperature. Also, the V_{offset} term is independent of U_T, threshold voltage, and I_{thpmos}, hence making the transition points invariant over temperature. Figure 3.14 shows the transfer characteristics of a common drain circuit with a pFET input. The measurement and simulation are plotted and have similar slope, that is, the kappa of the pFET input. The inset shows a zoom-in of the transfer characteristics measured over 60 ° C.

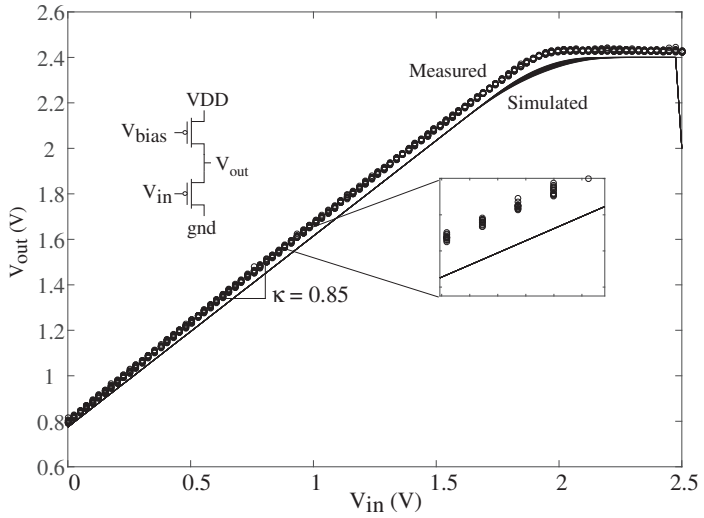


Figure 3.14: Transfer function of a common drain amplifier, measured and simulated using the models developed in the previous section, with temperature. The slope and the transition offset of the circuit is constant with temperature because the variation in threshold voltage of the two pMOS devices are similar. A slope of 0.85, which is the κ of the pMOS device, is measured from the circuit compiled on the FPAA.

The analyses of these circuits give further insight into the temperature dependence of various circuits and signal processing systems and allows us to predict and compensate for their behavior. Several FG-based current and voltage references [84] can be used that could help in reducing the variability caused due to changes in temperature.

3.9 Comparison with conventional simulators

Our open-source simulator based on EKV models obtained accurate results close to the experimental results from the FPAA. To further highlight the performance of our simulator, experiments have been performed comparing the behavior against the conventional Ngspice simulator, installed in the same virtual machine in the Ubuntu environment where the tool

Table 3.2: Comparison of simulation times with a conventional Ngspice simulator.

	With level=2 OTA	With all level=2 36T	Ngspice with EKV	Ngspice with BSIM
Sin input (20Hz) over 50ms	0.721s	1.632s	3.456s	5.384s
Chirp input over 50ms	0.945s	1.893s	4.296s	6.596s
Sin input (1Khz) over 5ms	0.548s	1.526s	2.985s	3.241s
Sin input (1Khz) over 5s	105.732s	206.132s	457.938s	508.143s

infrastructure is located. The Ngspice simulator [85] [86] uses the EKV model [87] in a 500nm process while our simulator in this work uses a 350nm process.

The experiments were performed for the front-end of a speech processing system as shown in Figure 3.11 for different set of inputs to identify the initial settling time, convergence, and compilation times. Level=2 models for the OTA were used for abstracting the 9T-OTA structure, thereby giving an accurate, yet faster simulation while the netlist for the system consisting of 36T was simulated in Ngspice with the same time resolution. Even using a 36T structure for the system in Modelica results in a faster simulation than Ngspice. The shorter input time span gives us an idea about the initial setup time and DC operating point solution time while the longer input gives an estimate of the longer computation time for larger systems in general.

A RK(45) solver is used, which determines the step size depending on the error in the solutions, thereby giving an accurate result, which can be observed from the overlap with the data from the FPAA in the results. We can see from the equations of the EKV model used in this chapter that it is an analytic function with derivatives existing and being continuous over all values. This is advantageous compared to a BSIM model with a large number of parameters, which takes a longer time to converge since the error may be higher due to the existence of discontinuities. Table I shows this aspect with the BSIM simulation time being greater than the simulation with the EKV model.

Moreover, the routing capacitances can also be estimated through transient responses by observing time constants while measuring the output relevant to the particular system. The capacitances can be modeled in the simulations, customizing depending on the CAB

location where the circuit elements have been placed, thus aiding in their characterization as well.

Our toolset allows circuit as well as system designers to implement applications. The tool directly compiles these elements into hardware, as well as simulates these circuits in the Scilab / Xcos environment, modeling the current-voltage relationships and interactions between blocks. They enable circuit designers to build system-level blocks. The models closely correspond to experimental data, thereby providing techniques to verify the blocks and their performance. Further, the application of this simulator to model circuits over temperature highlights the utility of such a framework in studying and understanding variations.

CHAPTER 4

IMPLEMENTATION OF SYNAPSES WITH HODGKIN HUXLEY NEURONS

4.1 Macromodeling a Neuron with Synapses on a Reconfigurable Platform

The foundation of building bio-inspired systems for real time applications is the design of efficient neuronal and synaptic blocks. Synapses enable the neurons to communicate with each other and allow one to implement algorithms to exploit the action potentials from neuronal cells. The Field Programmable Analog Array (FPAA) [2] System on Chip (SoC), a reconfigurable mixed signal hardware, is utilized in this chapter to show such bio-inspired circuits for energy efficient computing [31]. Tunability and programmability are achieved through the floating gate (FG) elements on the device. One could obtain different responses based on external inputs or other interconnections on the hardware itself.

This work takes advantage of the similarity between neuroscience and silicon to model synapses and neurons on the hardware. This block can be further expanded to build neural networks directly on the hardware. Analog signal processing offers high energy efficiency [59] and operates on low power as well with the analog elements operating in the sub-threshold regime.

Figure 4.1 shows the methodology behind building a spiking neural network on the FPAA from a fundamental set of neurons and synapses. The neurons are Hodgkin Huxley (HH) based models emulating ion channels. The synaptic cleft connecting the pre synaptic and post synaptic neuron is modeled to produce a triangle like response either from an external input or from another neuron. This ramp waveform is fed to a FG device in the routing that emulates the synapse and then exhibits the post synaptic response. The neuron projects to either excitatory or inhibitory synapses depending on the type of neurotransmitter a neuronal cell projects at its individual synapse. The responses of the blocks are

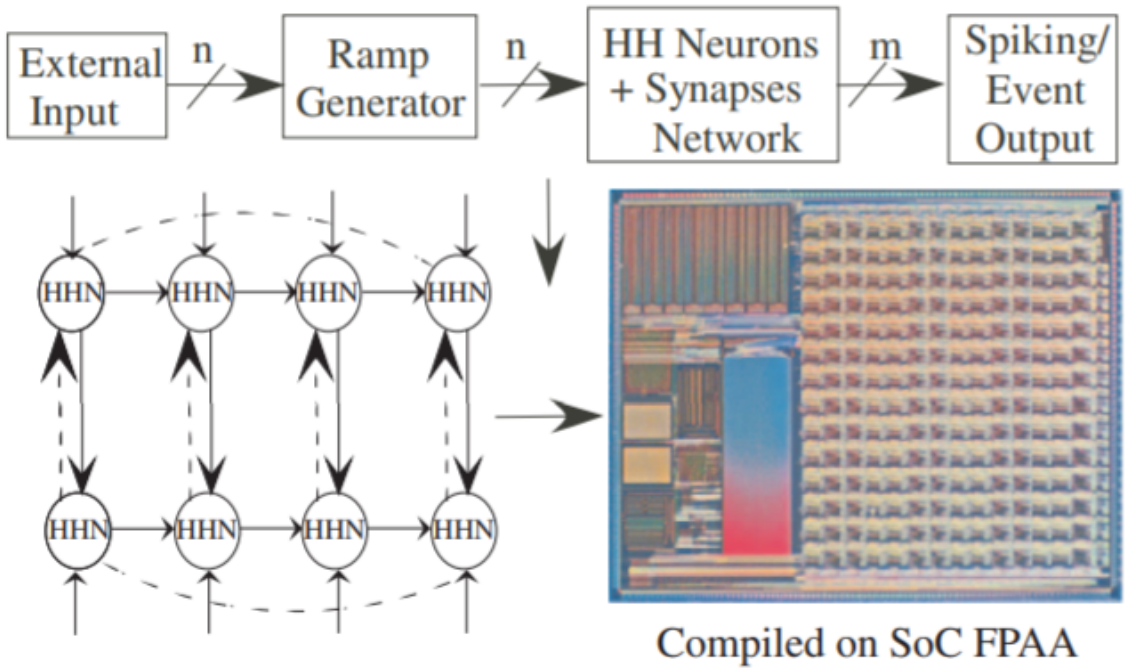


Figure 4.1: A network of neurons with external inputs and projecting excitatory and inhibitory synapses between them is shown here. These are translated to a set of macroblocks based on biological neuronal cells. The system can be compiled on the SoC FPAA to obtain experimental data. The ramp generator processes the external inputs that feed the synapses and the HH neurons' network is configured in the desired topology. Each macromodel of the HH neuron, integrated with synapses, is built in a single Computational Analog block (CAB). These level=1 macroblocks are vectorized thereby allowing higher levels of compilation and enabling larger networks to be built on the SOC FPAA.

controlled through the FG devices to generate an excitatory or inhibitory synapse and the synaptic strength is varied by modulating the charge on the floating gate.

The HH neuron, along with the synaptic clefts and synapses, is macromodeled and abstracted in the library blocks to be compiled onto the FPAA hardware. They are level=1 vectorized blocks with voltage input and voltage output [15]. The toolset enables one to build multiple neurons with synapses connecting between them by instantiating the library blocks [16]. This aids in building multiple configurations using the same single chip, rather than building custom chips for each configuration or application. The versatile place and route- algorithm used in the tools allows one to position the blocks according to the required chip topology.

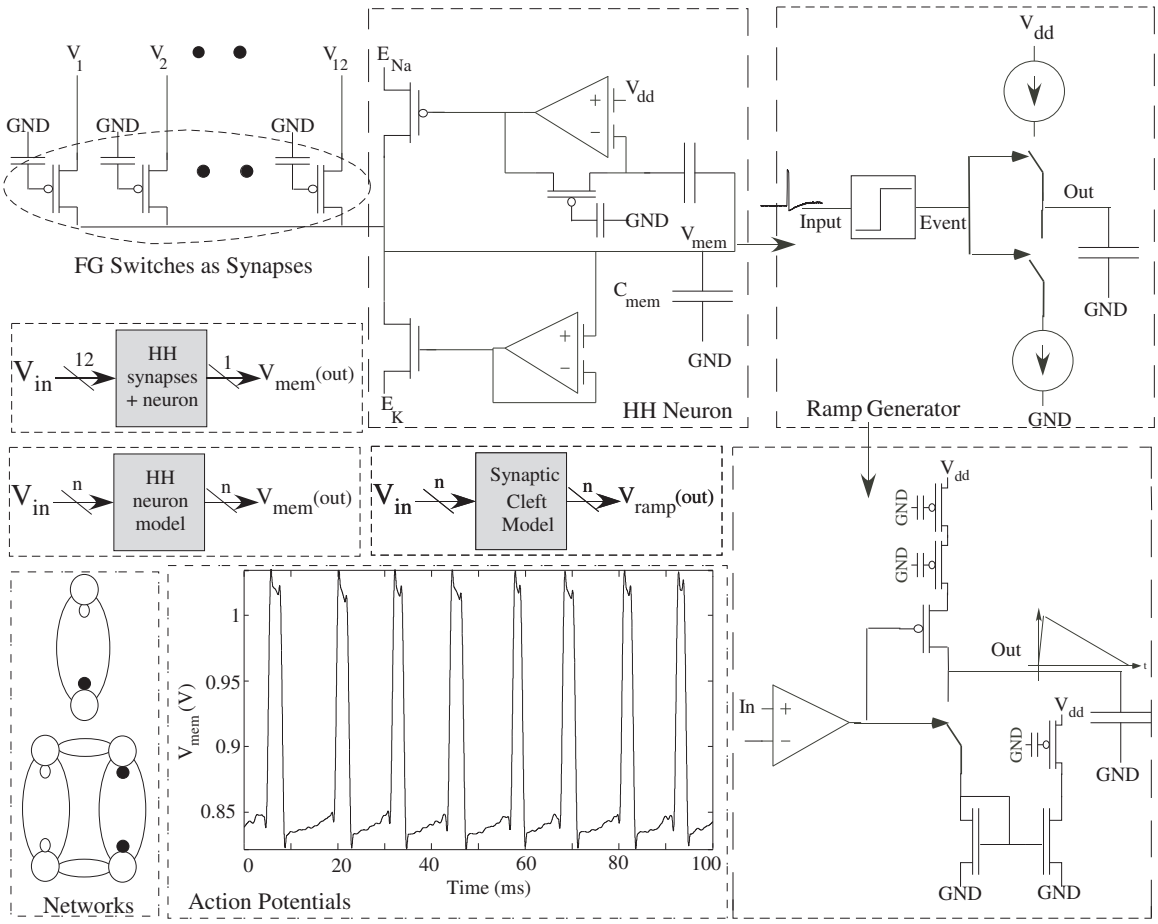


Figure 4.2: The design of the Xcos blocks, namely HH neuron block with the synapses and the synaptic cleft model, constrained to one CAB, is shown here. The synapses are through the FG pFET input currents, utilizing the routing FG switches. The ramp generator models the pre synaptic cleft response to create the triangles for the inputs to the synapses. The neuron projects to either excitatory or inhibitory synapses whose strengths and biases are controlled through the FG devices, to generate an event output or an action potential from the neuron. The FG elements are programmed with subthreshold currents. In case of transistor channel based HH neuron, the gating dynamics determined by the Na^+ bandpass circuit and the K^+ lowpass configuration are implemented through FG OTAs. The ramp generator makes use of the switches in the local routing, pFET, T-gate switches and a current mirror. The OTA detects the spike, generating an event, while the FG devices in the pull-up and pull-down design control the rise and fall time of the triangle. All the blocks are vectorized, implying multiple neurons can be implemented such that each neuron goes to each of the CABs and different size networks can be created. The experimental measurements of membrane voltage from the HH neuron model for an action potential are shown here too.

The experimental results obtained from the synaptic responses due to the neurotransmitters as well as the action potentials from the ion channel based model of HH neurons are

close to the behavior observed from biological neurons and synapses. Section 4.2 gives an overview of the circuit on the FPAA, while Section 4.3 and Section 4.4 describe the action potentials produced from the neuron and the synaptic cleft as a ramp generator. Section 4.5 addresses the excitatory and inhibitory synapses with how to approach the tuning of the parameters while Section 4.6 concludes by elaborating on building networks.

4.2 Design and Overview of the Block constrained to One CAB

Figure 4.2 illustrates the different Xcos macroblocks to build any arbitrary sized network on the chip. Each CAB is configured as one neuron along with the synaptic cleft that generates the ramp waveform connecting into the synapse that projects onto the post synaptic neuron. The FG local routing fabric, consisting of FG pFET switches models the synapses and is used for computation.

Using the available resources in one CAB of the FPAA, the design is figured out in such a way that it replicates the biological neuronal response and minimizing area overhead. Once the neuron is designed, the remaining available elements are optimally partitioned to design the circuits required for modeling the synaptic behavior.

4.3 Action Potentials from the HH Neuron Model

The transistor channel neuron model [88] inspired by Hodgkin and Huxley's work eliciting responses from a squid axon [17] is based on the ion channels in a neuronal cell. E_{Na} and E_K represent the biological supplies to the neuron which are equivalent to the Nernst ionic potentials, while C_{mem}, denotes the membrane capacitance. The depolarisation and hyper-polarisation in the membrane potential, V_{mem} happen due to the opening and closing of Na^+ and K^+ ion channels and the resulting interaction between them. Their conductances are modeled through a set of FG OTAs, nFETs, and pFETs. They are represented on the hardware as bandpass and lowpass filter respectively. The nonlinear dynamics arising out of their interaction gives rise to a continuous spiking response to an input fed through a FG

pFET in the routing as shown in Figure 4.2. The spiking frequency is varied by controlling the time constants of the Na^+ and K^+ channels.

4.4 Synaptic Cleft Modeling

Figure 4.2 shows the flow of signals from the input events between the pre synaptic and post synaptic neurons through the synaptic gap, equivalent to the movement of ions in the ion channels. A current starved inverter based structure to modulate the gate of the FG elements was shown in [89], while [90] presented integrate and fire neurons with current mode integrator based synapses and conductance based synapses in [30], and [91] shows current sink based synaptic inputs.

The ramp generator can be used to process the digital input and mimic the synaptic cleft. The action potential from the pre synaptic neuron is fed to the ramp generator. Once the neuron spikes, it is converted to an event through an OTA that behaves as a comparator with a threshold level. Depending on if one desires a ramp up or down first, the input is fed to the corresponding terminal of the OTA. As the spike is rising, the T-gate switch closes and the current mirror in the pull-down draws the current, with the rising time constant being controlled by the FG pFET. The pull-up then activates as the event falls with a time constant much slower than the rising time, tuned by the cascode FG pFET structure in the routing. Since the post synaptic potential (PSP) of biological neurons decays slowly [92, 93], the FG pFETs are biased such that it discharges slower than the rate at which it charges. This ramp is then fed to the source of the synapse element due to the exponential relation between the output current and source voltage of a FG pFET. If the external inputs are digital events, these triangle ramp generators can be used for input processing as well.

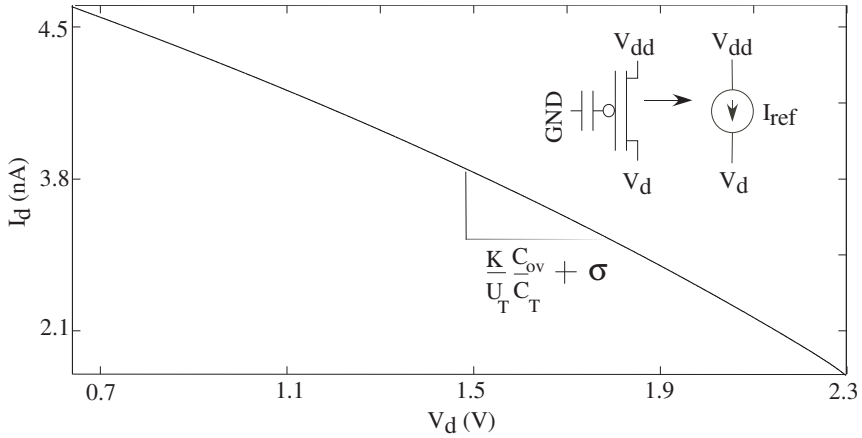


Figure 4.3: The current from a FG pFET in the routing is measured as a function of the drain voltage and the I_d - V_d curve is shown in log scale. The capacitive coupling can be seen in the slope and a range of currents is obtained as the input voltage is swept. The source voltage is fixed at a V_{dd} of 2.5 V and the FG pFET is biased such that it is in saturation. From the slope, the value of effective σ is calculated thereby giving the estimate of the early voltage which shows that the overlap capacitance in the routing is significant, which may cause a curvature in the response of the ramp generator. Hence, a cascode pFET structure is used to get a more accurate current source.

A current source as close to ideal as possible is required so that the ramp generator produces minimum curvature in the triangle fed to the synapse, be it inhibitory or excitatory. Hence, a through characterization of the FG pFET is performed. The drain voltage is swept to measure the drain current, as shown in Figure 4.3. The FG device is biased in the subthreshold region and the current in the saturation region is observed when the source to drain voltage is greater than 100mV, since we require it to behave as a current source. The source voltage is fixed at 2.5V. Due to the Early effect [70], which is a factor of the significant overlap gate to drain capacitance in the routing, the channel current depends on the drain voltage. To minimize this deviation from an ideal current source, a cascode structure is used in the pull-up design of the ramp generator, which reduces the effective σ, thereby increasing the Early voltage, making it a better current source.

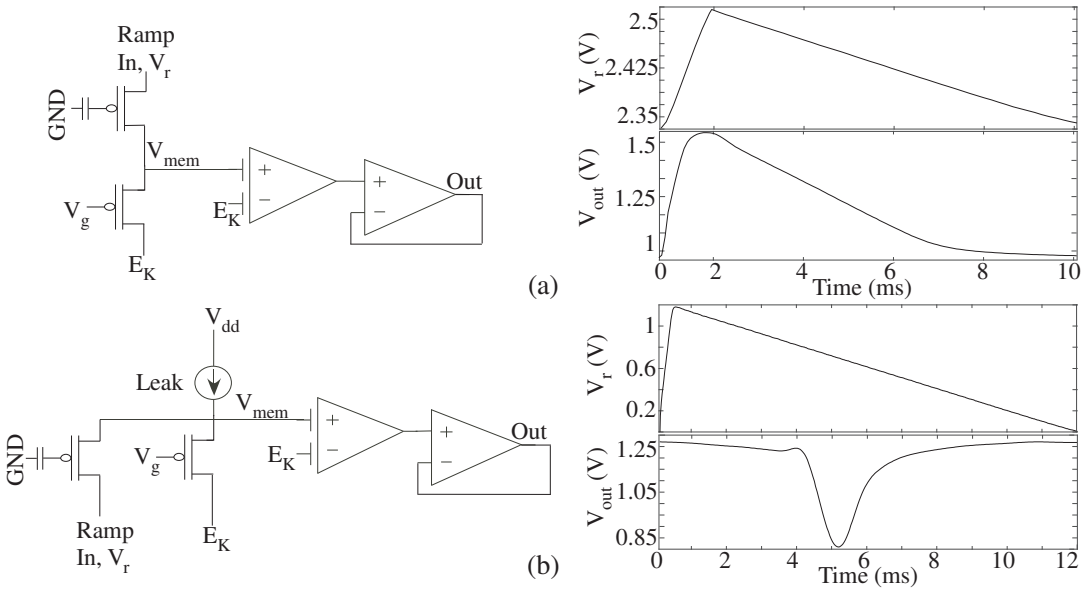


Figure 4.4: The experimental measurements' data from the FPAA for an excitatory and inhibitory synapse and the corresponding test setups are shown. The rise and fall times are controlled through the bias currents of the pFET and the current mirror in the ramp generator circuit. The change in membrane potential is gained up through the FG OTA and then buffered out to measure PSP. (a) The ramp input is fed to the source of the FG pFET for an excitatory synapse. The EPSP measured from the synapse with a passive channel rises up and decays slowly. (b) The ramp input is fed to the drain of the FG pFET for an inhibitory synapse. The inhibitory synapse is slower than the excitatory one and the IPSP, measured from the synapse with the passive membrane biased with a leak channel to draw the current, drops to E_K and then settles back at a resting potential.

4.5 Post Synaptic Responses through Excitatory and Inhibitory Synapse

As neurotransmitters are released into the synaptic cleft from the pre synaptic neuron, they attach to the receptors of the post synaptic neuron, causing the ion channels to open, resulting in ions flowing in the postsynaptic neuronal cell. This causes a change in the membrane potential [93] and an excitatory or inhibitory response is observed based on the type of ions that flow. An excitatory synapse causes deploarization in the post synaptic V_{mem}, while the inhibitory synapse reduces the V_{mem}.

Figure 4.4 shows the experimental results of the triangle created from the ramp generator as well as the PSPs. The synapse is modeled by the FG pFET in the routing inspired by the single transistor learning synapse [6], while the change in membrane potential is

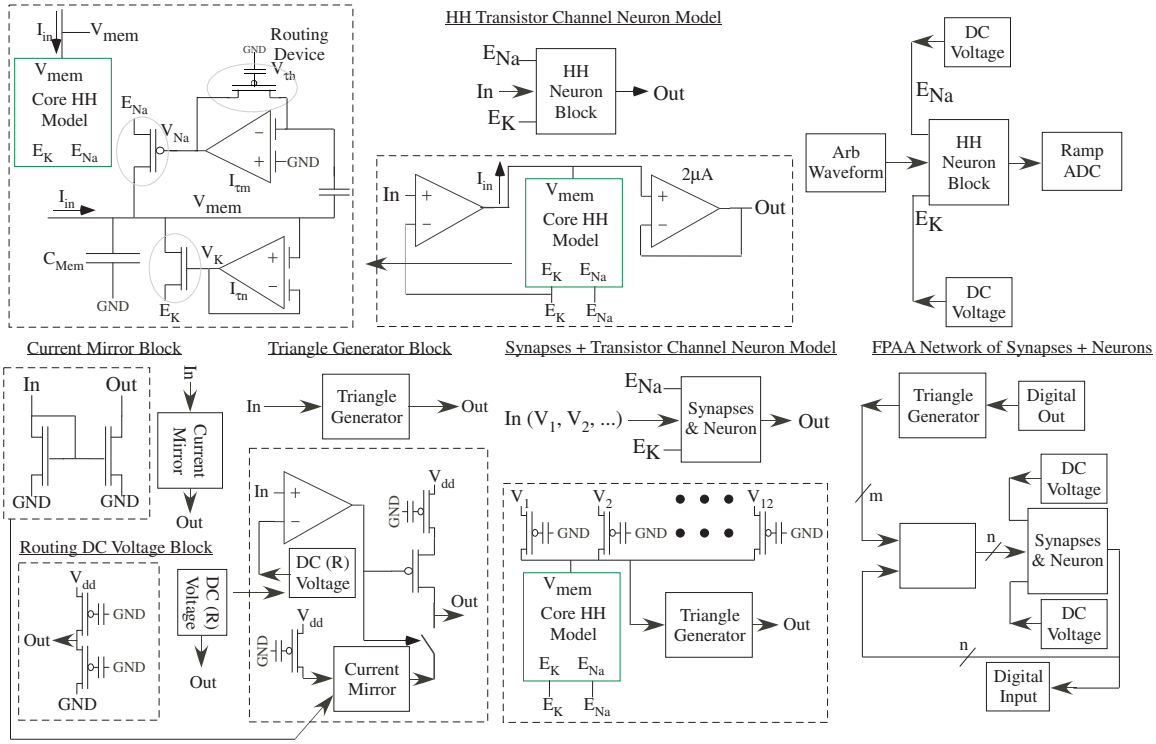


Figure 4.5: Blocks based on transistor channel model for neurobiological systems, enabling compilable networks of neurons and synapses. From one core HH model circuit, one can develop a set of parallel neuron blocks, as well as a set of parallel neuron and synapse block in the FPAA fabric. FG switch elements model the programmable synapse elements. Triangle Generator Block creates the presynaptic waveforms required for biological synapse response. The DC voltage block in Routing (R) uses two routing elements, nominally in a voltage follower configuration, to enable a programmed voltage source on any local CAB routing line (and can be routed into the fabric). The nFET current mirror block (level=2) corresponds to the nFET current mirror available in the CABs.

amplified through a FG OTA with a high gain and buffered through an OTA connected in a source follower configuration, with all the instrumentation performed on chip.

In case of an excitatory synapse shown in Figure 4.4a, the source of the FG pFET is modulated. The pFET is modeled as a biological passive channel, with its drain connected to E_K, while the EPSP measured from its source has a fast rising time and then decays slowly.

The drain of the FG pFET is modulated for an inhibitory synapse in Figure 4.4b. Hence, the ramp from the ramp generator is fed to the drain of the FG pFET and connected to a leak channel biased by a current source, which enables the FG source to draw the current.

We observe an inhibitory PSP (IPSP) that drops to E_K and then rises back and settles to a resting potential. The inhibitory synapse is designed to be slower than the excitatory one, matching biological synapses [93].

4.6 Building Networks

Figure 4.5 shows the methods of abstraction of neural blocks using the transistor channel model definitions. Different applications result in different methods of supplying the input current based on input voltage(s), which necessitates different blocks. All components are chosen to embed the structure in a single CAB, macroblocking the design.

One case uses an OTA to transform a voltage input to a single, direct current input into the neuron element(s). The membrane voltage (V_{men}) is buffered to the output. The measurement structure requires complexity similar to other circuits with an input, output, and two DC voltage biases (E_k, E_{Na}) for all circuit instances. E_K and E_{NA} are the same biological supply for all neurons and therefore shared between blocks.

A second case utilizes the local routing FG transistors as synaptic elements to combine synaptic and neuron activity. The outputs are triangle ramps which pre-compute the modeled charge concentrations reaching the post-synaptic terminal. The number of synapses is limited, in this approach, by the number of local input routing lines. The inputs require that their initial digital events have been converted to triangle ramps. The ramp element can integrate directly on the line capacitance, or can be buffered, depending on the resulting synaptic current consumption. The measurement structure shows a full layer of synaptically connected neurons, similar to the dynamics shown in custom ICs [43]. In the SoC FPAA [94], one could use this block to compile a network of 92 biologically modeled neurons, each with 12-14 synaptic inputs and 10-12 network inputs. Learning techniques from the VMM+WTA classifier could be adapted for this network [95]. One could create a software interface to directly utilize the PyNN [96] neural representation for simulation or compilation. The approach would extend to other neural network applications and ap-

proaches in a straightforward manner.

A routing DC voltage block can be built for setting DC voltages using only the routing fabric. These techniques enable dense setting of DC voltages. The circuit is nominally a FG pFET voltage follower. By characterizing one element, one gets a nearly ubiquitous voltage supply circuit that can be routed on any local line. Each CAB has local routing to V_{dd} and GND lines, so this component is always available with nearly no cost. Hence, through our open source tool infrastructure, one could consider creating networks building upon these blocks including central pattern generators [57, 56] and Winner take all (WTA) [89, 97, 98, 90] for different applications [99, 100, 53].

4.7 Synfire Chains

Networks like central pattern generators (CPGs) that mimic the locomotion of lampreys can be implemented. This is inspired by Mark Tilden's approach [101] with a coupled oscillator structure made out of highpass filter and inverters. The starting point of such a network is a synfire chain. The dynamics of these serially connected networks of neurons, where the output events are controlled by the delay of each neuron, are explored here, under the influence of different external synaptic inputs, where the spike timing is encoded in the information transmitted. Figure 4.6 shows a set of experimental measurements of action potentials for a coupled network and synfire chain. Two neurons are coupled to each other through inhibitory synapses feeding into each other. The phase of spiking is controlled through the synapses. Depending on the whether the synapses are excitatory or inhibitory, the neurons either spike in or out of phase with each other (Figure 4.6(a)). This behaviour can be exploited in oscillatory systems. A series of neurons are connected, making a synfire chain. Each neuron receives excitatory synapses from the previous neuron, while the output of the last neuron is fed back through an excitatory synapse to the first neuron. This cascaded structure allows the neurons to fire at different time delays. The propagation delay is seen through the action potentials from each neuron, which can be

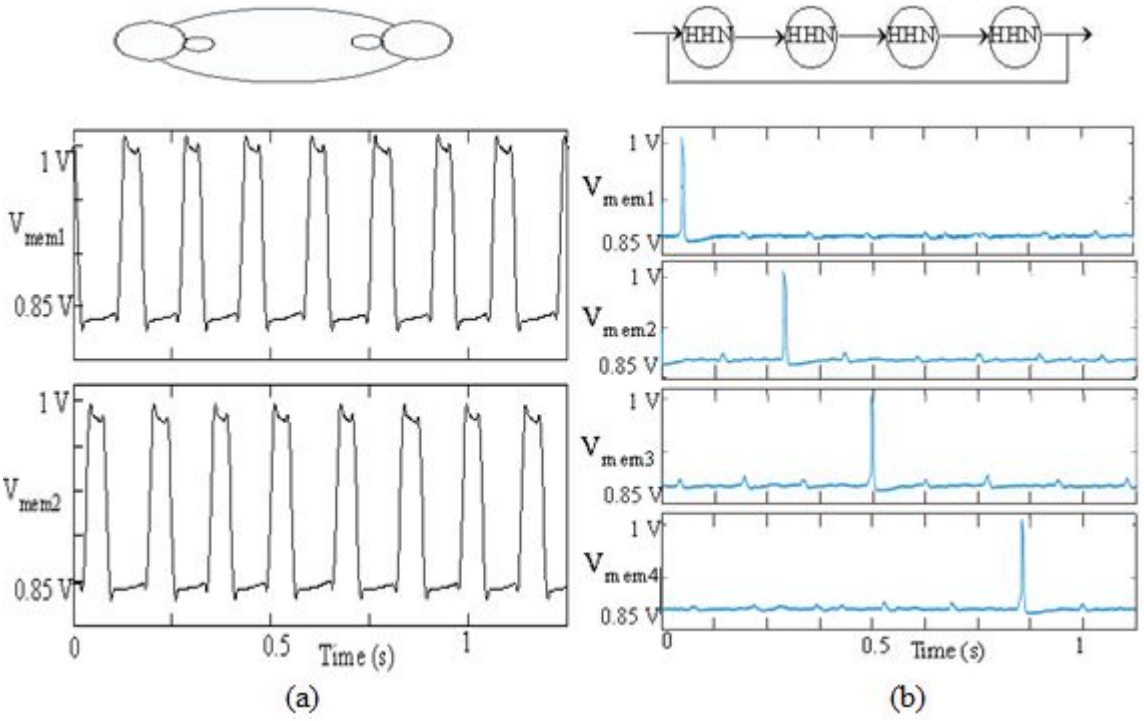


Figure 4.6: (a) Two neurons are coupled to each other through inhibitory synapses feeding into each other. The phase of spiking is controlled through the synapses. The neurons spike out of phase with each other since one neuron inhibits the other, thereby suppressing the spike occurring at the same time interval. (b) A series of neurons are connected through excitatory synapses, making a synfire chain. The propagation delay is seen through the action potentials from each neuron, further tuned through the synapses.

further controlled by modulating the synaptic strengths between the neurons.

4.8 Winner-Take-All (WTA)

We can build upon the modularity of the neurons and synapses to design different networks with a few neurons such as a WTA network. This scales up the macroblocked models in the infrastructure to build a network of different output neurons that has various synaptic inputs provided through triangle generators, which process the inputs. This allows for direct synthesis of any network topology, mapping from the design to the switches on the hardware.

The WTA is an example of one such spiking network. The outer ring consists of neu-

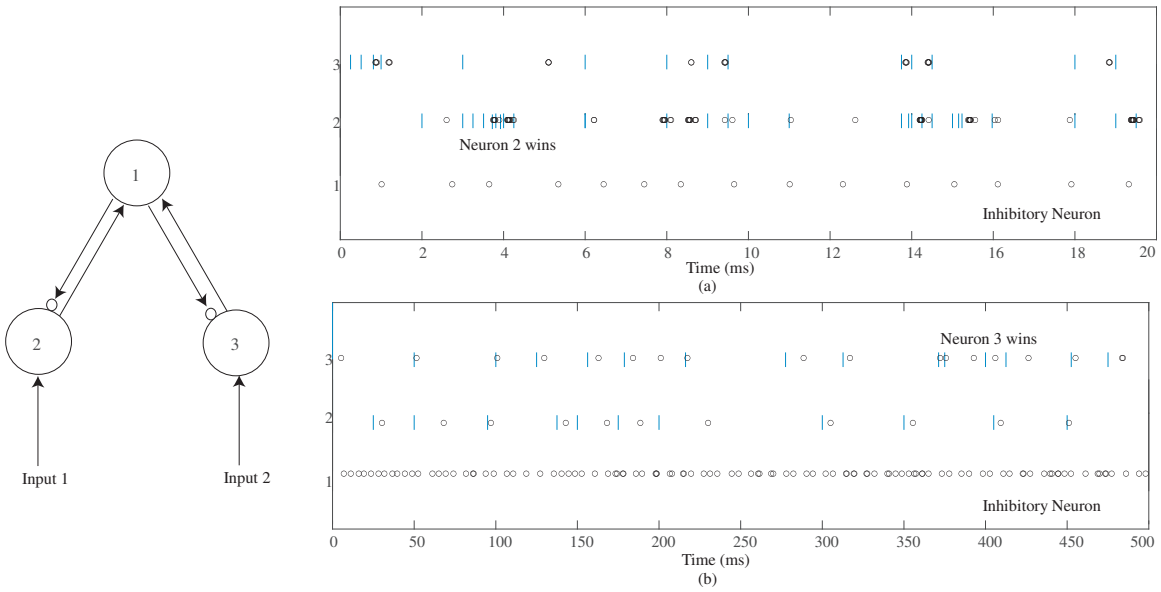


Figure 4.7: A configuration of a ring WTA network structure with three neurons is shown. The Neuron 1 is connected to the other two neurons through inhibitory synapse, while they project excitatory connections to the interneuron one. Poisson inputs represented by the solid lines are given while the circles show the output spikes. (a) A constant spiking is observed from the inhibitory interneuron. It is configured so that a single neuron (two) wins. (b) The strength of the inhibitory synapse is reduced and sparse inputs are given in a configuration to observe the transition of outputs such that the other neuron wins.

rons with excitatory synapses projecting to an interneuron that feeds back an inhibitory connection to the other neurons. As the excitatory synapse activates the interneuron, the inhibitory synapse that it projects starts to inhibit the excitatory elements. The synapses are programmed through the FG elements such that the desired excitatory cell wins after the inhibition provided by the interneuron.

Figures 4.7 and 4.8 show the spiking behaviour of a 3-neuron and 4-neuron WTA respectively. Each neuron in the CAB has been tuned to produce an action potential as shown in Figure 4.8(b). The intra CAB variation is less [102], leading to reduced mismatch among the cells, which can either be utilized [40, 46] or minimized [44] through the FG devices themselves without using extra resources for the compensation. The frequency and shape of the spike from the neuron has been modulated by tuning the channel time constants, producing a shape closely mimicking an action potential from a biological neuron.

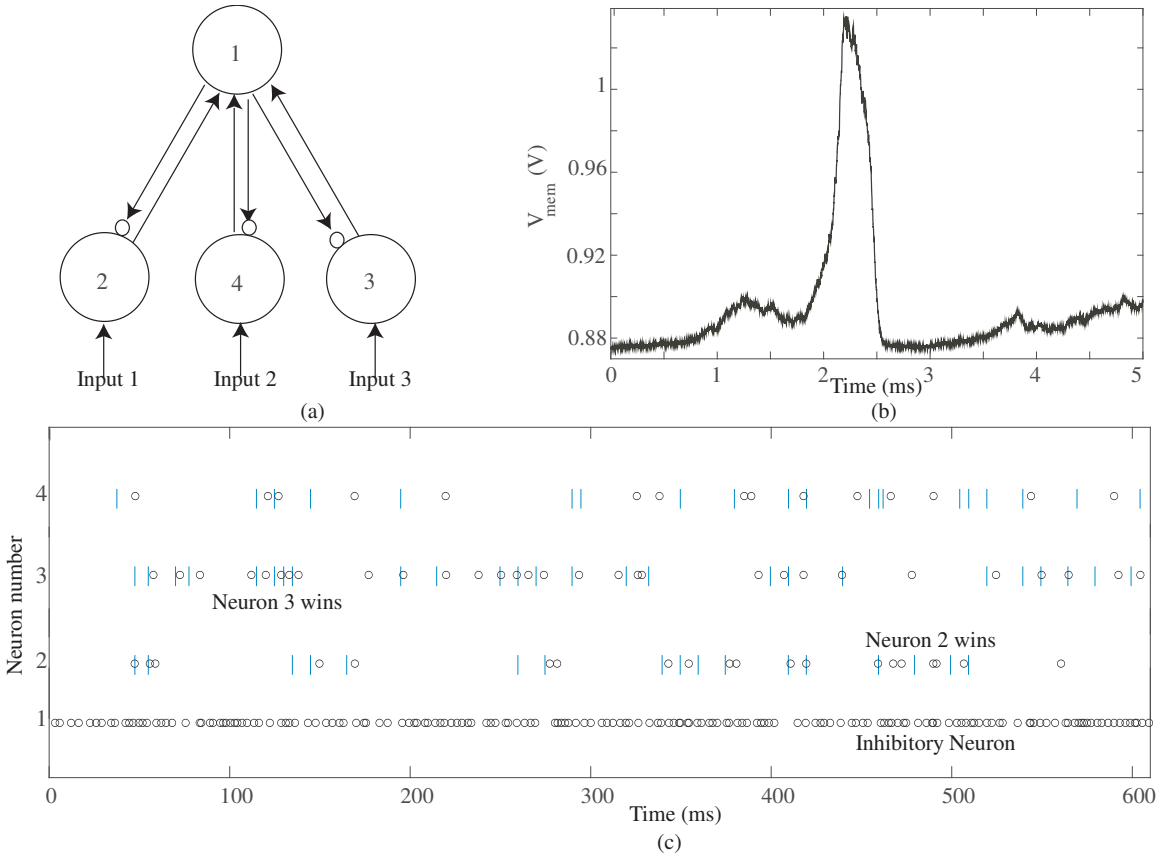


Figure 4.8: (a) A configuration of a ring WTA network structure with four neurons is shown. Neuron 1 is connected to the other three neurons through inhibitory synapse, while they project excitatory connections to the interneuron one. (b) A single spike is plotted to show the trace of an action potential of a neuron in the CAB. (c) Poisson inputs represented by the solid lines are given while the circles show the output spikes. A constant spiking is observed from the inhibitory interneuron with a low inhibition. It is configured so that two neurons win for the respective inputs.

Poisson inputs applied to the outer neurons are represented by the solid lines, while the circles show the output spikes. A constant spiking is observed from the inhibitory interneuron. As the synaptic strengths change, affecting the inhibitory and excitatory connections between the neurons, we observe different neurons winning, corresponding to their respective inputs. The winning neuron spikes correspond to the highest frequency on the input, and suppressing the other neurons for that frequency due to the lateral inhibition. Figure 4.7 shows cases for a single winner where the increased spiking on one neuron inhibits the other neuron. Further, Figure 4.8 shows two winning neurons corresponding to the fir-

ing rate on the input. The inhibition is controlled by tuning the timing of the ramp on the inhibitory synapse.

4.9 Discussion

This chapter presented a macromodeled HH neuron block integrated with excitatory or inhibitory synapses and synaptic clefts or ramp generators, analogous to biology, and demonstrated results from the FPAA fabricated on a 350nm process.

A range of algorithms can be built from the compiled blocks of channel based neurons and synapses, to explore different neuromorphic systems. One could build acoustic applications based on a spiking VMM and WTA network, where the the output of the interneuron goes into inhibitory synapses. This can be used to exhibit exclusive-OR behavior while the system classifies a frequency pattern in a phrase. Furthermore, multiple dendritic compartments can be added to the neuron, through FG pFETs in the routing fabric that model conductances and current sources that model the inhibitory and excitatory synaptic channels. By macroblocking it to be in a single CAB, it can communicate the inputs to other neurons. This could be further used for applications like keyword spotting.

CHAPTER 5
PROGRAMMABLE FILTERS WITH BUILT-IN SELF-TEST OF VECTOR MATRIX MULTIPLIERS

5.1 Built-In Self-Test of VMM blocks

Analog mismatch dominates the performance of most applications (e.g. data converters as in [103]); and therefore, most analog systems require some level of programmability, as well as an algorithm to tune that programmability, for high-performance operation. As computation moves towards more system-level capabilities, these issues become magnified. When reaching the level of a million parameters large-scale Field Programmable Analog Arrays (FPAA) [10], an automated form of tuning this programmability becomes essential. Some early steps have been considered in this directions for configurable systems [104], [105]. Some views approach the question with a belief that mismatch helps the resulting computation, such as in neuromorphic systems [106, 107], and either hope that belief works or aggregate many components to remove that mismatch (e.g. [108, 109]). This effort focuses on making a fundamental key computation, Vector-Matrix Multiplication (VMM), directly implementable on an FPAA structure with automatic tuning capabilities (Figure 5.1).

VMM is the fundamental operation throughout signal processing and neural operations where one wants to weight and sum a set of inputs [110, 111]. VMM is a core block of analog and other novel computational techniques, as seen in early neural-network mesh networks (e.g. [112, 6]), and is central to programmable filters, signal processing, classifiers, deep neural networks, and spiking networks of neurons [113, 114, 115, 116, 2, 117]. The high energy efficiency of VMM computation was first hypothesized by Mead in 1990 [1], experimentally demonstrated a decade later [118], and further experimentally verified

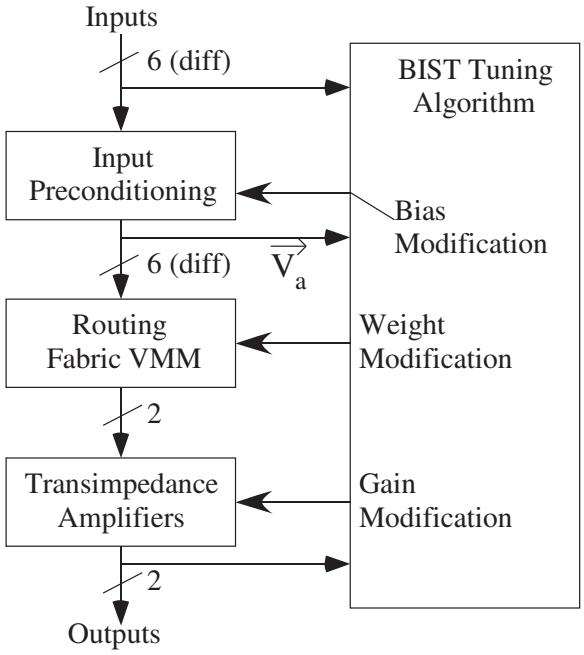


Figure 5.1: A built-in self-test algorithm for a Vector-Matrix Multiplier (VMM) and resulting infrastructure is implemented in a large-scale Field-Programmable Analog Array (FPAA) for a wide range of applications. The algorithm uses the available inputs and outputs to converge on the target weights and biases. The adaptation programs out the threshold-voltage mismatch due to indirect programming on the FPAA, as well as tunes the desired output voltage gain and amplitudes. Once the solution has been reached, the circuit can be retargeted without the BIST on-chip infrastructure.

in custom and FPAA implementations (e.g. [119, 120, 121, 122]) Tuning this algorithm greatly provides a window for tuning more complex algorithms.

This chapter focuses on a FPAA-implemented VMM Built-In Self-Test (BIST) algorithm (Figure 5.1) that includes a 6x2 VMM block and the input and output infrastructure required for using these techniques. Abstraction of these analog blocks makes this discussion tractable [16]. The VMM block and tuning capabilities are experimentally demonstrated. The following sections describe the core 6x2 VMM FPAA block (Section 5.2), the input and output interface circuitry to the VMM (Section 5.3), the effects of the mismatch due to the indirect programming (Section 5.4), self-tuning algorithm for the whole chain (Section 5.5), measurements compensating for the indirect-programming mismatch along with the analysis of the outputs of the VMM (Section 5.6) and an application using the

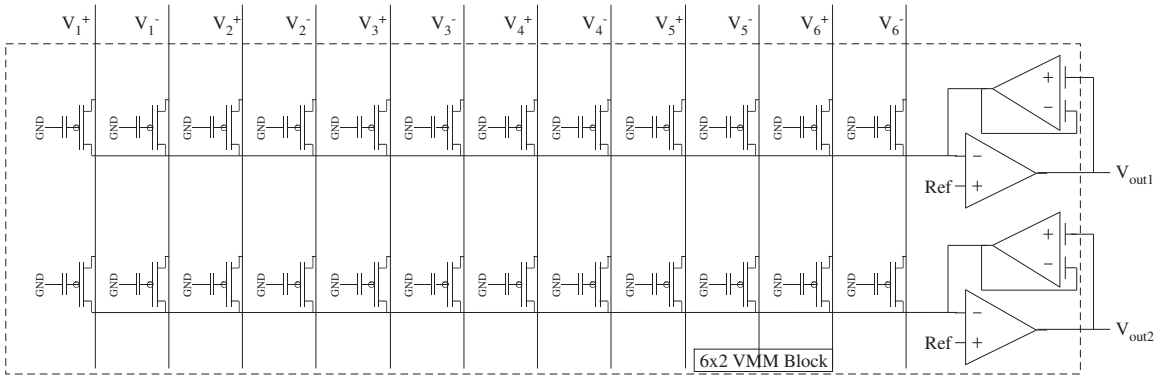


Figure 5.2: A differential 6x2 VMM including two transimpedance amplifiers to translate the currents to voltages, is compiled on the FPAA. The FPAA Computational Analog Block (CAB) routing lines implement the differential, four-quadrant VMM. The FG elements of the VMM perform the 4 quadrant multiplication, with the weights being set by the FG pFETs on the positive and negative input. The inputs V_i^+ and V_i^- that drive the sources of the VMM have to operate near V_{dd}. Six differential column lines are selected from the CAB, with 2 row output lines that go into the CAB elements that comprise the TIA, while the local routing elements in the fabric consisting of the FG elements make up the VMM structure. The output levels after the TIA are adjusted primarily through the FG OTA element at the feedback of each of the TIA, which controls the gain, offset and the operating range.

VMM with programmable filters (Section 5.7).

5.2 6x2 VMM FPAA Block

The VMM is implemented using the Floating-Gate (FG) devices in the routing fabric of the FPAA, effectively vector multiplying a set of input vectors with the stored weights through that crossbar memory (Figure 5.2). We target the VMM and interfacing infrastructure on the SoC FPAA [10, 2].

We focus our techniques on a 6x2 four-quadrant VMM block built in the routing fabric of a single Computational Analog Block (CAB) of the SoC FPAA. A larger VMM can be constructed by cascading multiple blocks as well as by incorporating other levels of routing switches. Our four-quadrant operation requires two input lines for the resulting differential input signal, resulting in six differential inputs as each CAB has 13 input signal lines. The FG pFET-based routing fabric stores the VMM weights, which are in turn set through the

FG programming infrastructure [14]. The VMM circuit operation is a source-input version of an earlier 4-quadrant multiplication element [123], modified in systems with Winner-Take-All (WTA) and VMM+WTA classifiers [124, 76].

The block has two independent output rows limited by the number of TA amplifiers in each CAB, creating a voltage output through compiled transimpedance amplifiers. The two transimpedance amplifiers use one FG TA element to build feedback around the one non-FG TA. The FG TA allows for a selectable linear range, including one setting for nearly rail-to-rail linearity (1.5V-2V on a 2.5V supply), as well as for tuning voltage offsets. The output DC voltage is referenced to V_{ref} as these amplifiers operate on a single V_{dd} (2.5V) supply. This design conforms to the system block abstraction (level=1 definition in [15]) and is confined to a single CAB.

This VMM computation multiplies a differential-signal input vector by an array of stored differential weights (Figure 5.2). A single FG pFET device with a programmed gate voltage (as in Figure 5.2) can be modeled in terms of the bias current, I_{bias}, and changes in source (V_s) and drain (V_d) voltages as

$$I = I_{bias} W e^{(\Delta V_s - \sigma \Delta V_d)/U_T} \tag{5.1}$$

where W is a weighting element due to a difference in programmed charge on the FG node (V_{fg}) or difference in threshold voltage (V_{T0}), modeled as $e^{\kappa(\Delta V_{fg} - \Delta V_{T0})/U_T}$. Further, σ models channel length modulation and/or Drain Induced Barrier Lowering (DIBL), κ is the capacitive divider from V_{fg} to surface potential, and U_T is the thermal voltage, kT/q. The actual bias levels for the transistor terminals and bias current (I_{bias}) are essential to the overall system setup and will be further addressed in later sections. In this case, W is positive or zero. Assuming *small* changes in V_s where $|\Delta V_s| \leq U_T$, (5.1) becomes

$$I = I_{bias} W \left(1 + \frac{\Delta V_s}{U_T}\right) \tag{5.2}$$

which shows a multiplication between W and ΔV_s, when $\sigma \rightarrow 0$. A four-quadrant VMM operation utilizes two devices, where a differential signal is applied to the two source volt-

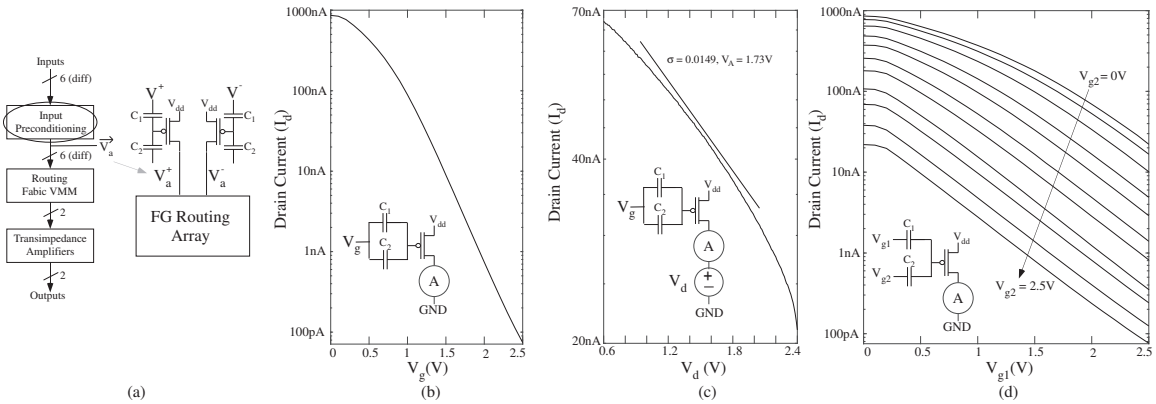


Figure 5.3: The front-end interface to drive the VMMs to produce signals close to the supply voltage uses a FG-based CAB element (MITE block) to both increase the input linearity as well as level-shift signals from mid-rail to voltages near V_{dd}. To understand this operation, we show the characterization of this device similar to its operating conditions with its source fixed at V_{dd} (2.5V). (a) Circuit block and its position in the computational algorithm. (b) Measured drain current when sweeping both input capacitances (C_1 and C_2) with the same input voltage (V_g). (c) Measured drain current when sweeping the drain voltage (V_d) resulting in rough extraction of σ_{eff} (= 0.0149). (d) Measured drain current when sweeping one gate input (C_1) resulting in extraction of κ_{eff}. The curvature increases as the input terminal approaches V_{dd} typical of a pFET drain sweep which shows that the effect of the overlap feedback (which is constant) is small.

ages and the resulting weight is the difference of the two weights:

$$I^+ = I_{bias} W^+ \left(1 + \frac{\Delta V_s}{U_T}\right)$$

$$I^- = I_{bias} W_- \left(1 - \frac{\Delta V_s}{U_T}\right)$$

$$I = I^+ + I^- = I_{bias} W_0 + \frac{I_{bias}}{U_T} \Delta W \Delta V_s, \tag{5.3}$$

where $W_0 = W^+ + W^-$ and $\Delta W = W^+ - W^-$. These formulations are consistent with earlier VMM formulations (e.g. [123]).

5.3 Interface circuitry to VMM

Since the VMM operates near V_{dd}, a set of FG devices available in the CABs (called a Multi-Input Translinear Element, MITE, as seen in [125]) level-shifts the input voltages and widens the input range through capacitive voltage division into the FG node. One device is required per input line even if shared across multiple blocks. Although the source-voltage inputs to the VMM, V_a (Figure 5.2), could be biased at almost any voltage level by the translinear principle [126], voltage levels near V_{dd} mean the currents required for programming are the closest to targeted currents, enabling easier programming of these structures. Nominally, we bias these voltages 100mV ($\approx 4U_T$) below V_{dd}, so all pFET devices are operating in saturation. A source to substrate voltage difference of 100mV for saturated subthreshold MOSFET operation roughly decreases the current by a factor of 50. If the programmed currents (I_{prog}) are subthreshold, then the target currents (I_{target}), which are smaller than the programmed currents, is a constant factor below the programmed currents (Figure 5.4) . For higher target currents, the relationship is a nonlinear function of the target current and pFET threshold current (I_{th}) as shown in the analysis below.

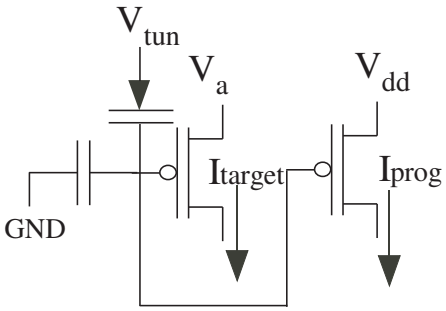
The equations for modeling programming difference between the target current and programmed current and issues on V_a:

$$I_{target} = I_{th} e^{\kappa(V_{dd} - V_g - V_{T0})/U_T} e^{-4}$$
$$I_{prog} = I_{th} \ln^2 \left(1 + e^{\kappa(V_{dd} - V_g - V_{T0})/2U_T}\right)$$
$$I_{prog} = I_{th} \ln^2 \left(1 + e^2 \sqrt{\frac{I_{target}}{I_{th}}}\right)$$

,

$$I_{prog}/I_{th} = e^4 \frac{\frac{I_{target}}{I_{th}}}{1 + e^2 \sqrt{\frac{I_{target}}{I_{th}}}}, \tag{5.4}$$

Figure 5.4 illustrates indirect programming through the two pFETs. The transistor in



I_{bias}	I_{target} (nA)	I_{prog} (nA)
$W^{+}_{1,1}$, $W^{+}_{2,1}$	3.75,3.75	100,100
$W^{-}_{1,1}$, $W^{-}_{2,1}$	1.25,1.25	40,40
$W^{+}_{1,2}$, $W^{+}_{2,2}$	3.75,3.75	105,102
$W^{-}_{1,2}$, $W^{-}_{2,2}$	1.25,1.25	43,37
$W^{+}_{1,3}$, $W^{+}_{2,3}$	3.75,3.75	97,100
$W^{-}_{1,3}$, $W^{-}_{2,3}$	1.25,1.25	38,45
$W^{+}_{1,4}$, $W^{+}_{2,4}$	3.75,3.75	100,100
$W^{-}_{1,4}$, $W^{-}_{2,4}$	1.25,1.25	40,40
$W^{+}_{1,5}$, $W^{+}_{2,5}$	3.75,3.75	95,100
$W^{-}_{1,5}$, $W^{-}_{2,5}$	1.25,1.25	40,40
$W^{+}_{1,6}$, $W^{+}_{2,6}$	3.75,3.75	100,102
$W^{-}_{1,6}$, $W^{-}_{2,6}$	1.25,1.25	40,45

Figure 5.4: Indirect programming infrastructure through the two pFET structure. The transistor in the circuit acting as the VMM device is programmed indirectly by measuring the current of the programming transistor. The source of the VMM, V_a is set at $4U_T$ from V_{dd} so that the devices are in saturation as well as to keep the programming and targeted currents close. To set the analog weights on the VMM through I_{target}, we define the current to be programmed, I_{prog}. The target and programmed current values for the VMM weight matrix, factoring in the model, are shown in the table.

the circuit acting as the VMM device is programmed indirectly by measuring the current of the programming transistor. To set the analog weight on the VMM through I_{target}, we define the current to be programmed, I_{prog} which is calculated from the model (5.4). The table lists these expected targeted values for these quantities and programmed values.

A single FG pFET can enable the desired level-shifting and gain function as part of the VMM system interface (Figure 5.3). Programming the FG charge enables level-shifting the resulting intermediate voltage (V_a) from input signals (e.g. V_1) that are operating in the middle of the power supply. The capacitive coupling to the FG node capacitively divides the input voltage (without additive noise), enabling several 100mV swing to fit in a roughly -U_T to U_T swing. The change in V_{fg} is equal to C_1/C_T times the change in the V_{g1} node,

$$\Delta V_{fg} = \frac{C_1}{C_T}\Delta V_{g1} + \frac{C_2}{C_T}\Delta V_{g2},$$

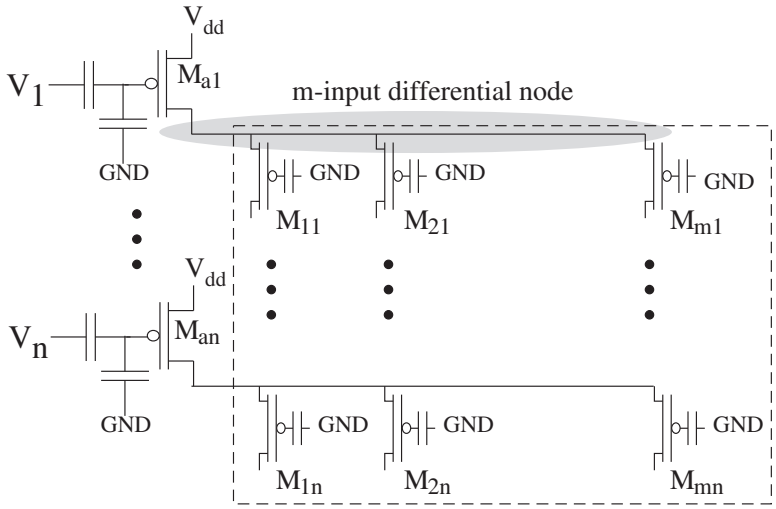


Figure 5.5: The VMM, M_{11} to M_{mn}, is driven by a set of FG pFET transistors, M_{a1} to M_{an}, potentially creating a normalization typical of differential-pair or winner-take-all (WTA) circuits. These driving FGs on the interface should act as a translator of voltage levels and not just as current sources. This prevents the differential pair effects along the columns, thereby not putting constraints on the possible input voltage level cases.

where C_1 and C_2 are the input capacitances and C_T is the total capacitance at the FG node. Sweeping either both voltages together (Figure 5.3b) or sweeping the voltages in a parametric sweep (Figure 5.3c) shows the increased linearity for subthreshold currents. The drain to gate overlap capacitance (C_{ov}) can have an effect on ΔV_{fg}, increasing the measured σ value, $\sigma_{eff} = \sigma + \frac{C_{ov}}{C_T}$; the effect for this device is small (Figure 5.3d).

The FG pFET device and the resulting pFET switch elements for the VMM computation along a column appear to function like a multi-input form of a differential-pair circuit (Figure 5.5). Although such a structure can normalize signals into a desired operating range, the circuit could, if not properly balanced, restrict the number and types of available weight matrices. The circuit should act more like a voltage signal translation between the input and V_a nodes, and less like a bank of current sources driving the VMM columns. One approach requires all VMM source voltages (V_a nodes) biased at the same voltage (2.4V), effectively eliminating different differential pair effects along different columns. The first self-test algorithm uses FG programming to tune the V_a nodes to a 2.4V bias (Figure 5.6).

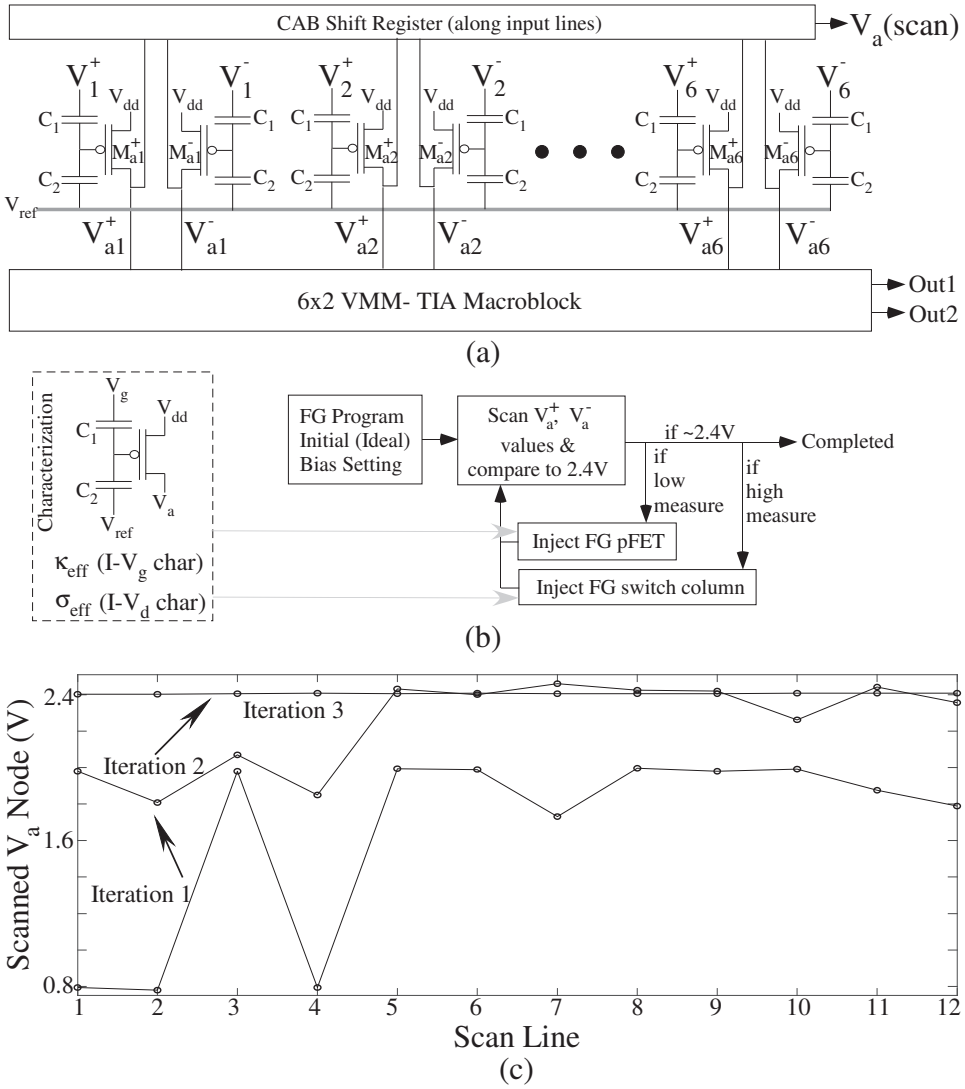


Figure 5.6: Floating-gate pFET elements used to gain and level shift the VMM input signals to the VMM source nodes. These devices are components available in SoC FPAA CABs. (a) The FG pFET outputs are made available through a shift register for characterization and resulting calibration. (b) The voltages at the output intermediate nodes are set such that they are as close to V_{dd} as possible and the FG devices are biased to avoid a differential pair effect on the columns. The levels are compared and the FG pFETs, M_{a1}^{+} or M_{a1}^{-} to M_{a6}^{+} or M_{a6}^{-} are programmed through hot-electron injection. (c) The voltages of the individual MITEs are plotted after three iterations and a significant reduction in the variation is observed from a change of 1.6V to 5mV.

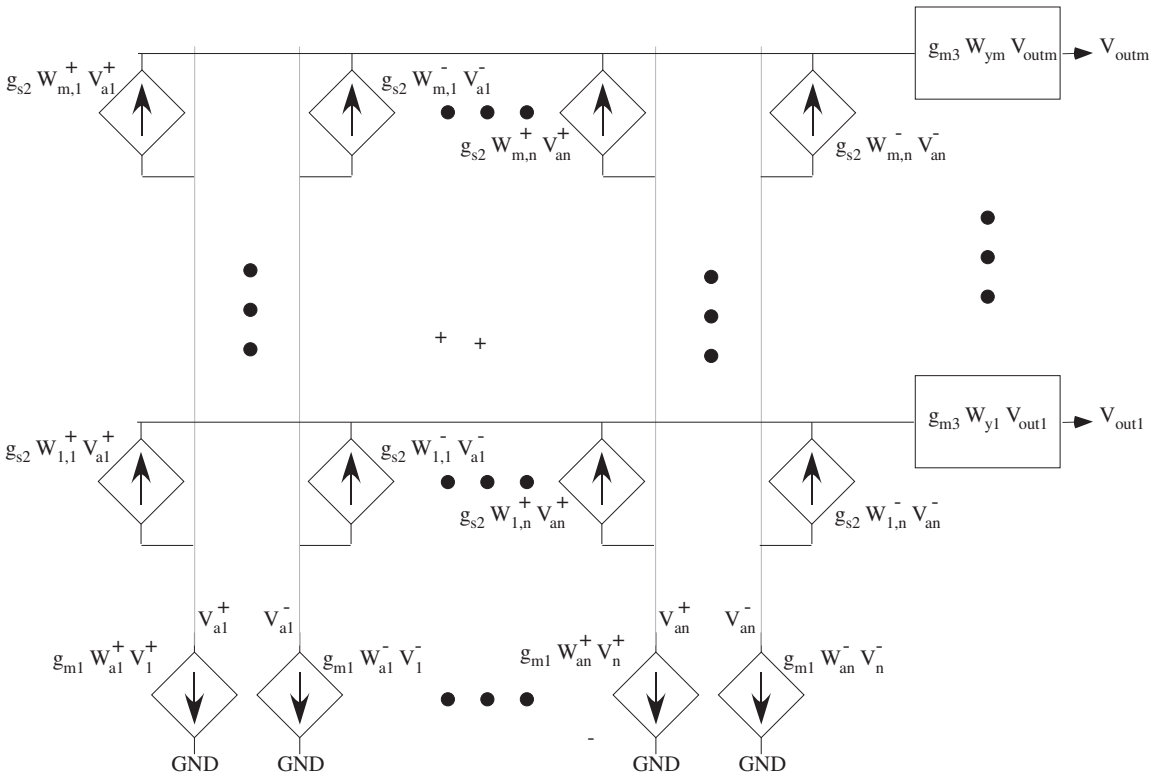
A shift-register block already available in each CAB [2] along the I/O lines into the CAB is compiled [13] to be controlled by the on-chip microprocessor. This shift register allows scanning of the V_a lines (Figure 5.6a) to be measured and the corresponding programmed

FG elements are modified (Figure 5.6b). Hot-electron injection, one of the two programming mechanisms (e.g. [14]), allows for fast (e.g. ms) programming updates that increase the transistor bias current (decreases V_{fg}). Decreasing the current bias (increasing V_{fg}) requires electron tunneling, resulting in erasing the entire chip and reprogramming the entire infrastructure. Therefore, we want our algorithms to increase bias currents to tune any improvements (Figure 5.6b). If the particular V_a line is higher than 2.4V, then the FG elements along the VMM column that are higher are programmed according to the V_a line voltage shift. If the particular V_a line is lower than 2.4V, then the input FG pFET element would be programmed according to the V_a line voltage shift.

Starting from initial expected (no-variation) targets, this algorithm can program around V_{T0} mismatch resulting from indirect programming. Three iterations of the scanned nodes (Figure 5.6c) brings the variation drops in the nodes from the desired 2.4V significantly from a range of 1.6V to 5mV. Once calibration and the built-in self test is concluded, the scanner can be eliminated.

5.4 Effect of VMM System Mismatch

A reasonable set of parameters programmed into the FG devices with the V_a nodes within a reasonable range (within a few U_T) means that our VMM structure should be functioning as a VMM within the bounds of some significant mismatch. Mismatch from programmed FG-devices is V_{T0} mismatch between the two pFET FG transistors due to indirect programming [127]; one FET is used for programming and one FET is used for operation. Indirect programming techniques easily enable a widely heterogeneous set of components including above-threshold switches [128]. This improves upon the direct programming algorithm [10], which was employed in earlier generations of the FPAAs, since that used a single transistor but required disconnecting from the rest of the system with transmission gates (T-gates). Hence, the indirect programming results in fewer switches and T-gates, leading to fewer parasitic capacitances and reduced complex interface circuitry.



(a)

	Small Signal Variable	I_{bias}	Typical
g_{m1}	input FG transconductance	5nA	0.013/MΩ
g_{s2}	switch source conductance	2.5nA	0.1/MΩ
g_{m3}	FG OTA transconductance (transZ)	30nA	0.03/MΩ

(b)

Figure 5.7: Analysis of Computation and Mismatch in the VMM block. (a) Small-signal model of the full VMM system with the infrastructure elements. (b) Summary and values of key small-signal parameters. Typical values are given for a 6x2 VMM computation. For these calculations, we assume a typical κ of 0.7, and $C_1/C_T = 0.09$. g_{m3}, the transconductance of transimpedance (transZ) amplifer assumes a 1V linear range (V_L).

We might find other V_{T0} mismatch among non-FG devices that will also affect these components.

A tuning method needs to eliminate this mismatch, and yet, this particular mismatch means that parameters are reasonably near their correct values. Linearized analysis techniques reasonably model the mismatched signal-dependent aspects (Figure 5.7) built on linearized conductances and transconductances. Each input FG switch element, M_{a1} to M_{an}, is programmed with a bias current (I_{bias}) that is the starting point for our modeling.

The user sets it as an input parameter in the Xcos design for the programming currents, further setting the weights. A fraction of that bias current flows into each switch element, M_{11} to M_{m1}, based on their weighting values, and then the resulting values are summed together as the input into the transimpedance stage. As seen in previous cases using differential weights (e.g. [123]), the positive and negative programmed weights are of a similar size, rarely more than a factor of 4 from each other. For m rows and n columns, it is reasonable to assume that each switch is biased with I_{bias} / m current, and each transimpedance amplifier is biased with 2 n I_{bias} / m current. The non-FG OTA bias current is programmed high enough for the output dynamics and has little effect on the output precision. The linearized conductances can be computed given these bias currents, including g_{m1} (= $\kappa(C_1/C_T)I_{bias}/U_T$), the programmed, identical transconductances from the input FG elements, g_{s2} (= $I_{bias}/(mU_T)$, the normalized source conductance for the switch elements, and g_{m3} (= $2nI_{bias}$ / $m\mathrm{V}_L$), the programmed, identical transconductance from the FG OTA used by the transimpedance amplifier. V_L is the linear range of FG OTA (e.g. 1V) in the transimpedance amplifier. For these programmed values, the gain from an input voltage (e.g. V_1^+) to the V_a nodes (e.g. V_a^+1) is $\kappa(C_1/C_T)$, an attenuation entirely set by capacitors.

This formulation extends to modeling device mismatch (Figure 5.7). Typical V_{T0} mismatch results in a bias current that changes less than a factor of two for these components, allowing the use of small-signal modeling. Uncompensated systems can show 5 to 50 percent system deviations using subthreshold bias currents. The weights due to programmed charge and V_{T0} mismatch are defined for the corresponding row and column. $\mathrm{W}_a^{+,-}$ are the weights due to mismatch for the input current-source transistors, while $\mathrm{W}^{+,-}$ are the programmed weights of the switch elements for the VMM, and $\mathrm{W}_y^{+,-}$ represent the weights due to programming and mismatch of the transimpedance amplifiers. Given these defini-

tions, we get the l^{th} output voltage as

$$V_{out,l} = \frac{g_{s2}}{g_{m3} W_{y,l}} \sum_{k=1}^{n} \left(W_{k,l}^{+} V_{a,k}^{+} + W_{k,l}^{-} V_{a,k}^{-} \right), \tag{5.5}$$

and the voltage for the k^{th} middle node as

$$g_{s2} V_{a,k}^{+} \sum_{g=1}^{m} W_{g,k}^{+} = g_{m1} W_{a,k}^{+} V_{k}^{+}$$

,

$$g_{s2} V_{a,k}^{-} \sum_{g=1}^{m} W_{g,k}^{-} = g_{m1} W_{a,k}^{-} V_{k}^{-}, \tag{5.6}$$

Substituting (5.6) into (5.5), we get

$$V_{out,l} = \frac{g_{m1}}{g_{m3} W_{y,l}} \sum_{k=1}^{n} \left(\frac{W_{k,l}^{+} W_{a,k}^{+}}{\sum_{g=1}^{m} W_{g,k}^{+}} V_{k}^{+} + \frac{W_{k,l}^{-} W_{a,k}^{-}}{\sum_{g=1}^{m} W_{g,k}^{-}} V_{k}^{-} \right). \tag{5.7}$$

Remembering that the inputs are differential,

$$V_{k}^{+} = -V_{k}^{-} = V_{k}/2, \text{and } V_{a,k}^{+} = -V_{a,k}^{-} = V_{a,k}/2$$

. In a small signal representation, the signal weights are

$$\Delta W_{k,l} = \frac{W_{k,l}^{+} W_{a,k}^{+}}{\sum_{g=1}^{m} W_{g,k}^{+}} - \frac{W_{k,l}^{-} W_{a,k}^{-}}{\sum_{g=1}^{m} W_{g,k}^{-}}, \tag{5.8}$$

resulting in the target case

$$V_{out,l} = \frac{g_{m1}}{g_{m3} W_{y,l}} \sum_{k=1}^{n} \Delta W_{k,l} V_{k}. \tag{5.9}$$

The output gain from input (e.g. V_1 to $V_{out,1}$) is multiplied by g_{m1}/g_{m3}.

The small-signal analysis does not show the effect of the biasing levels, which include the output bias current level. A straightforward analysis shows that the output bias current weights, W_0, would be,

$$\Delta W_{0,k,l} = \frac{W_{k,l}^{+} W_{a,k}^{+}}{\sum_{g=1}^{m} W_{g,k}^{+}} + \frac{W_{k,l}^{-} W_{a,k}^{-}}{\sum_{g=1}^{m} W_{g,k}^{-}}, \tag{5.10}$$

leading to an output bias current of $I_{bias} \sum_{k=1}^{n} \Delta W_{0,k,l}$. Offsets can be programmed out by

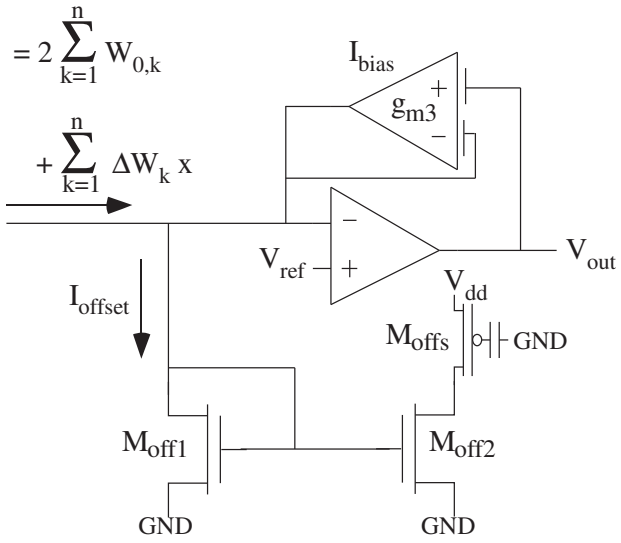


Figure 5.8: TIA circuit that includes a tunable OTA for tuning the amplitude gain, and a bias current to remove additional bias current arising from the multiplication operations, is added to the main block. We see that the constant weight values along a row are much larger than the signal values.

the FG programmed current source (Figure 5.8), as these current levels tend to be significantly larger than the input signal current levels. The DC bias current is removed through an nFET current mirror with transistors M_{off1} and M_{off2} using the FG device, M_{offs}, to set that current. This approach allows tuning the output gain by decreasing g_{m3}.

5.5 Built-in Self Test for the Full Chain

Given a set of signal inputs, known desired outputs (Figure 5.10), and calibrated V_a nodes (Figure 5.6), the resulting weights and offsets could be calibrated. The previous mathematical framework for the VMM weights (5.8) as well as output current offsets (5.10) illustrates sufficient flexibility for this calibration. Multiple adaptive algorithms could converge to a solution.

This section discusses an automatic algorithm to tune calibration mismatch enabled through the non-volatile VMM structure (Figure 5.9). The goal of a calibration algorithm is to identify and allow for quick on-chip adaptation by only increasing values (e.g. only

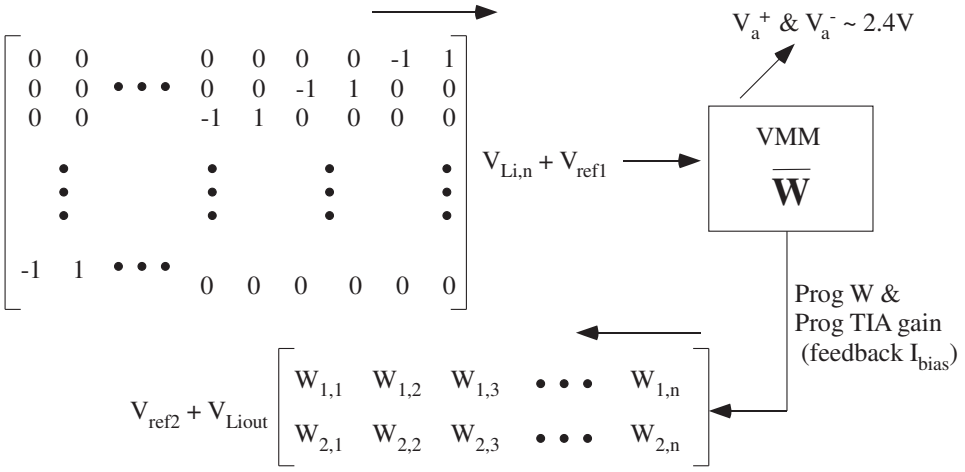


Figure 5.9: Block diagram of the data measurement for performing the VMM self-test operation . This approach inputs the identity and the negative of the identity matrix, scaled to the full-scaled range into the VMM block to correct the indirect threshold mismatch in the VMM and TIA bias currents. One could equally take an arbitrary set of vectors and perform a Least Mean Squares, LMS solution on that data using this flow.

hot-electron injection). The algorithm chooses an initial set of inputs that modulates each input component by its largest positive and negative differential input that are related to the input linear range ($V_{Lin} = C_T U_T/(\kappa C_1)$) around the required input offset voltage. This input is effectively the normalized +1 and -1 inputs (**x**) for the normalized input. The resulting normalized outputs (**y = W x**) of the VMM are the positive and negative values of the effective VMM weights (Figure 5.9). The differential signal outputs directly show the output offset voltage (V_{ref2}) and the output linear range ($V_{L,out}$). The V_a nodes are monitored so the algorithm can tune DC levels to stay near 2.4V.

A normalized weight vector of all one values illustrates the calibration algorithm (Figure 5.9), where the positive weight could be 1.5 and the negative weight could be 0.5 (W_0 = 2). The positive switch elements would be programmed to 100nA and 40nA resulting in targeted values of 3.75nA and 1.25nA. The bias currents for the positive columns would be 7nA and the negative columns would be 2.5nA. The resulting output signal for the positive and negative input would be an output current alternating between 33.25nA and 35.75nA. One would want to program the bias current structure to 34.5nA, resulting in an output

Table 5.1: Measured Output for V_a averaged over ten trials

Active Input Pair	V_{a1}^+ (V)	V_{a1}^- (V)	V_{a2}^+ (V)	V_{a2}^- (V)	V_{a3}^+ (V)	V_{a3}^- (V)	V_{a4}^+ (V)	V_{a4}^- (V)	V_{a5}^+ (V)	V_{a5}^- (V)	V_{a6}^+ (V)	V_{a6}^- (V)
V_1^+,V_1^-	2.39	2.42	2.42	2.41	2.41	2.41	2.40	2.41	2.41	2.40	2.39	2.40
V_2^+,V_2^-	2.40	2.39	2.37	2.42	2.39	2.41	2.40	2.41	2.38	2.39	2.40	2.39
V_3^+,V_3^-	2.39	2.40	2.40	2.40	2.37	2.43	2.38	2.41	2.38	2.41	2.37	2.40
V_4^+,V_4^-	2.38	2.40	2.38	2.39	2.39	2.42	2.39	2.42	2.39	2.40	2.40	2.40
V_5^+,V_5^-	2.37	2.41	2.37	2.40	2.38	2.41	2.39	2.41	2.39	2.42	2.40	2.43
V_6^+,V_6^-	2.40	2.40	2.40	2.40	2.40	2.42	2.38	2.42	2.38	2.41	2.38	2.42

current modulating between -1.25nA to 1.25nA. Biasing the 1V linear range TIA at 2.5nA would result in a 1V peak to peak output signal. These values expected after the devices are calibrated, and these are the values to initially program as the starting guess for this normalization; the corrections would be incorporation of the V_{T0} mismatch and could be directly used for programming new VMM values.

Table 5.1 shows the statistics of the measured voltages from the intermediate nodes at V_a following the self-test algorithm described in Figure 5.9. The voltages corresponding to the positive and negative components of the identity matrix is applied to the input FG switch elements that correspond to the positive and negative weights, one pair at a time, while the rest of the input elements are fed the 0 input. This pattern is fed to all 12 inputs, with +1 and -1 alternating through the positive/negative pairs. The intermediate nodes, V_a, in the system are measured through a shift register connected in the loop to make sure that they are near 2.4V. The shift from 2.4V is higher in some trials since the parasitic capacitances may shift between the program and run modes. Also, the error rates could be due to the accuracy of the 14-bit ADC used for the measurements while programming. The residual mismatch in such cases could be further tuned out by fast injection on those switches, till the nodes hit the target 2.4V as close as possible. The average over the trials show that the signals remain bounded near 2.4V during the course of the computation. This helps to make sure that the programmed weights on the VMM are as close to the targeted weights as possible.

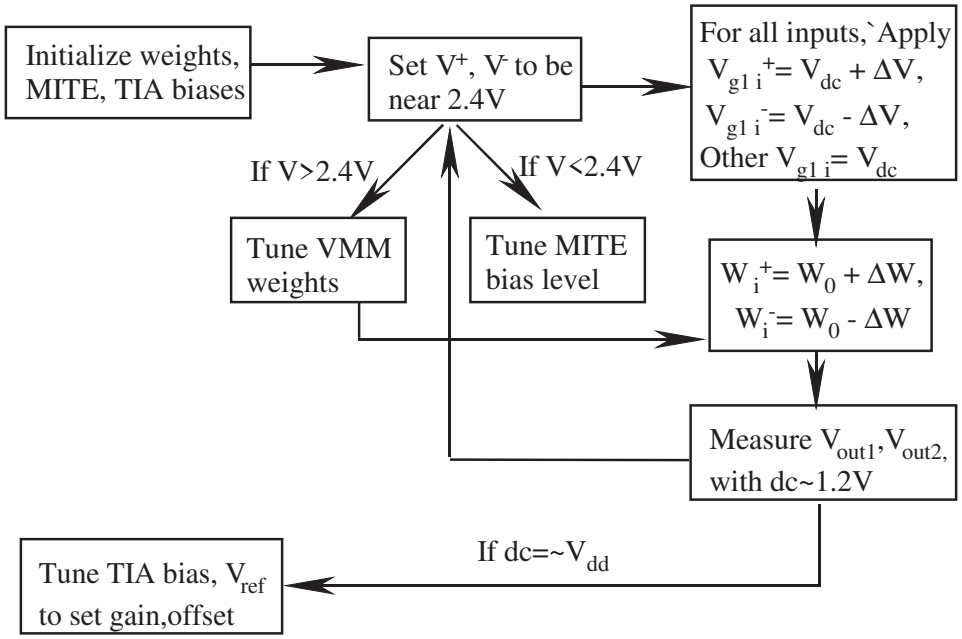


Figure 5.10: The algorithm flow for the full system, to set the weights of the VMM. It calculates the variation change with respect to the different inputs applied and then measure the intermediate nodes. This process is repeated for all inputs. After an initialization of the parameters, the source inputs to the VMM and system outputs are measured and compared and the VMM weights or the input FG biases are fine tuned accordingly depending on the variation.

5.6 Outputs of the VMM

The algorithm for the entire system chain is shown in Figure 5.10. Once the input elements are biased and the weights of the VMM are calibrated accordingly, the gain and the DC level and the operating range at the VMM outputs can be controlled through the TIA. The output values are moved through the injection on the output TIA structure. The output gain can be increased by decreasing the transconductance of the FG OTA as seen in (5.9) or reducing the bias current of the FG OTA. The offset charge between the positive and negative terminals of the differential TA is used to center the output waveform. The current mirror structure also helps to control the remaining bias current that is subtracted at the output node, thereby controlling the DC levels at the output rows. This current source is programmed to a low current initially and then injected incrementally until the desired

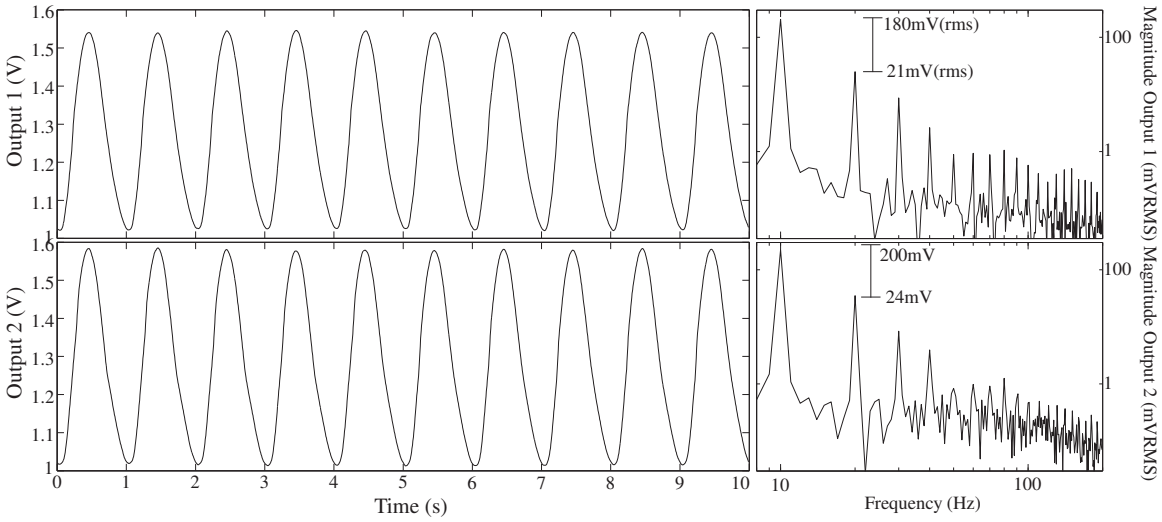


Figure 5.11: 1Hz differential sine waves are applied to the six pairs of taps. The gain and the DC level at the output is set such that it is not near the rails and distortion is as low as possible. The frequency analysis of the 1Hz signal is shown, where the output voltage levels are regressed to observe the V_{rms} magnitudes and show the primary amplitude (180mV) and second harmonic (21mV).

response is obtained. The feedforward OTA is programmed to a high value while the feedback FG OTA sets the gain and operating range at the output.

A set of sinusoidal waves are applied to the system to characterise the VMM in addition to the step reponses. Figure 5.11 shows the transient sinusoidal responses of the VMM with the application of 1Hz differential sine waves to the six pairs of taps. The output voltage is scaled such that the distortion is minimum. As all the signals are added on the taps with the weights, it can be considered as the case for distortion where the noise is at its lowest. The noise current at the output is a factor of the sum of weights and output bias current for a particular output voltage swing. The VMM can be considered as a matched filter where the largest response is always to the matched signal, as the SNR depends on the operating conditions. The frequency analysis of the 1Hz signal is shown in Figure 5.11, where the output voltage levels are regressed to observe the V_{rms} magnitudes and show the primary amplitude (180mV) and other harmonics. The noise component has a number of bias currents being added together as a single value at the input is being modulated. The

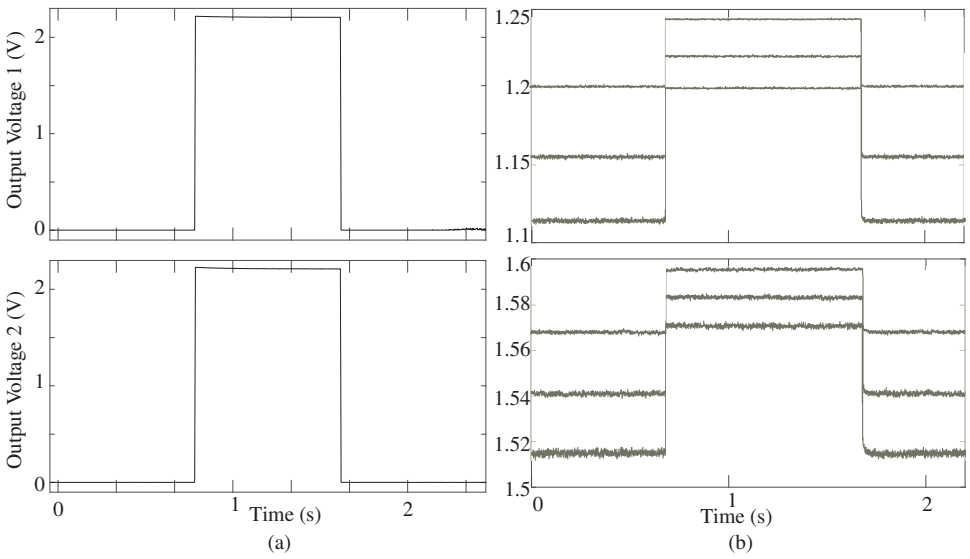


Figure 5.12: (a) The level of the outputs of the VMM row 1 and 2 after the TIA is shown, in response to a step input. A full-swing output is shown at the VMM outputs setting the FG OTA at TIA to maximum gain. A higher gain is further obtained by decreasing the bias on the FG OTA of the TIA. (b) The gain and offset can be tuned by changing the bias on the FG OTA at the feedback of the TIA. As the feedback bias current is increased, the gain drops on the output signal.

noise power increases with the number of taps and the output conductance can be tuned through the TIA, depending on how large one wants to scale the output voltage and tune for the distortion.

Figure 5.12(a) shows the full-swing outputs at the VMM, setting the gain to the maximum at the FG OTA of the TIA. Figure 5.12(b) shows the different DC levels at the outputs, which could be centered by tuning the offset bias current that is subtracted or by modulating the FG charge at the input terminals of the feedback FG OTA. From the rail to rail outputs, it is apparent that it is possible to get such a level of gain, providing a base level from where one could tune the outputs depending on the desired output gain and operating range, by a fine injection on the output. Hence, the user can choose a range of output conductances through the TIA and tune it accordingly.

5.7 Programmable Filters with VMMs

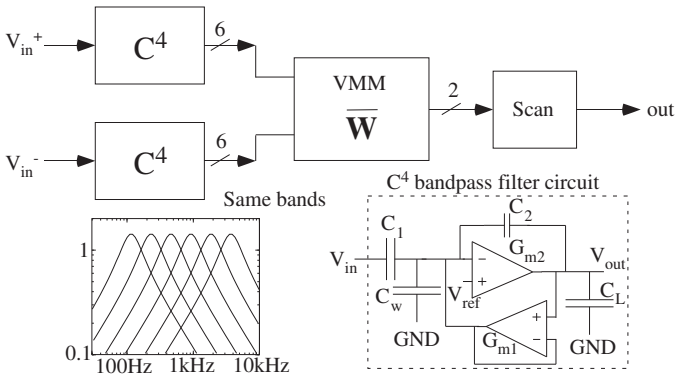
This VMM tuning algorithm could be used in a number of applications. We show a signal processing application using the calibrated VMM structure in a larger system. A bank of bandpass filters are connected to the VMM to experimentally demonstrate the programmable filters.

In case of these tunable filters, the input data would be from a microphone or similar sensor feeding into a bunch of filters, while the outputs would be measured from the VMM block. This system, similar to custom programmable filter efforts [113], is typical of audio MP3 encoders [129], where sub-band coding is achieved through bandpass filters [130] that decompose incoming bands into a set of frequencies, while the VMM helps to choose a particular set of responses. A weighted sum [131] is experimentally demonstrated, through a combination of filters and multipliers.

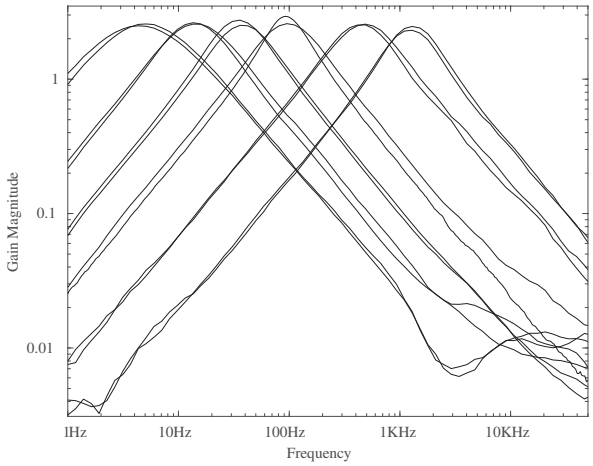
Figure 5.13(a) shows a system where two pairs of six filters are connected to the positive and negative inputs of the differential VMM. The system uses six bandpass filters with differential input and output signals, setting up the VMM for the two resulting bandpass filter functions. The capacitively coupled current conveyer (C^4) based bandpass filters [75] are tuned such that the pairs of six taps produce identical responses, with each set tuned to produce exponentially spaced center frequencies between 1Hz and 100KHz, as shown in Figure 5.13(b). The respective corner frequencies are set by the transconductances on the feedback and feedforward FG OTA of the C^4 (Figure 5.13(a)). They have been biased such that the filters themselves produce similar outputs in terms of gain and quality factors, while producing responses that are uniformly spaced from each other.

The transfer function of the C^4 is

$$\frac{V_{out}}{V_{in}} = \frac{s^2 C_1 C_2 - s G_{m2} C_1}{s^2(C_p C_T - C_2^2) + s(G_{m1} C_p + G_{m2} C_2 - G_{m1} C_2) + G_{m1} G_{m2}}. \tag{5.11}$$



(a)



(b)

Figure 5.13: Application of a self-tested VMM structure. (a) Block diagram for a programmable filter utilizing the calibrated weight values is shown. The differential 6x2 VMM is connected to two pairs of a bank of six filters each. The system outputs for different filter functions are scanned out and read from the two VMM outputs. The schematic of a C^4 bandpass filter with two FG OTAs whose transconductances control the cut-off frequencies, is shown. (b) The output of the set of twelve filters are shown here. Each set of six bands are fed to the corresponding six differential inputs of the VMM. The individual filter responses from the bank of C^4 bandpass filters, are tuned such that both of them have the same set of bands and cut-off frequencies.

where C_1, C_L, C_2 and C_W are the input, load, feedback and routing capacitances respectively while $C_p = C_2 + C_L$ and $C_T = C_1 + C_2 + C_W$. G_{m1} and G_{m2} are the transconductances that set the lower and upper frequencies respectively.

Table 5.2 compares the measured and expected center frequencies of the two pairs of filter banks. They have been tuned such that the measured values are close to the expected

Table 5.2: Comparison of measured and expected center frequencies of the filter banks

Filter Tap	Measured		Expected	Error (%)	
	Bank 1	Bank 2	Bank 1,2	Bank 1	Bank 2
1	4.92Hz	4.66Hz	6.78Hz	27.4	31.2
2	19.23Hz	18.46Hz	21.23Hz	9.4	13.0
3	42.29Hz	35.93Hz	38.76Hz	9.1	7.3
4	109.69Hz	104.72Hz	105.16Hz	4.3	0.4
5	486.25Hz	471.38Hz	475.22Hz	2.3	0.8
6	1.49kHz	1.42kHz	1.46kHz	2.1	2.7

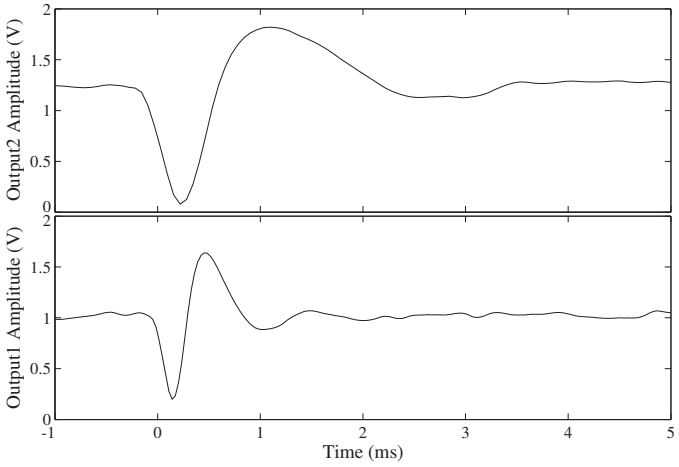


Figure 5.14: Step responses from the system structure with the filters and VMM corresponding to two outputs for a weight matrix, are plotted to study the transient behaviour. The rising and decaying on the transient waveform is characteristic of a bandpass filter behavior. The measured rise and fall times from the step response match the expected cut-off frequencies for the system. It is apparent from the time scale that it passes the middle and higher set of bands. The feedback FG OTA on the TIA is used to set the DC level on the output rows and center the waveform with a reasonable gain.

ones from the bandpass filters. The error rates deviating from the expected values are defined in the table. The transconductances of the feedforward and feedback FG OTA on the filters aid in tuning the lower and higher corner frequencies, thereby controlling the center frequencies and their deviations from the expected analytical values.

The weights of the VMM have been tuned as discussed in Section 5.5. It is essential to have a calibrated VMM in such larger systems. Fortunately, the resulting infrastructure can be removed after calibrating the device once. Since the dynamic operation range of this

VMM has been extended, the frequency responses from the C^4 can be directly multiplied through the VMM to obtain desirable filter behaviours. Further, the BIST strategy enables one to set the correct biases on the positive and negative coefficients of the VMM weight matrix. These weighted outputs are further summed to generate a single response. The differential weights can be tuned such that either a particular single response or a weighted combination of filters is chosen. Each of the output rows of the differential VMM can be biased to produce two different sets of programmable functions, that ultimately select a desired set of corners in the system frequency response.

Initially, the VMM is tuned such that it passes one band from the system wherein all six bands come through the system with single outputs sequentially. The weights corresponding to the active bands are set to the desired factor while the weights for the inactive bands are set to a low bias level so that the single band is output.

Figure 5.15 shows the specific case of a pair of the six center frequencies being active, one at a time. These have a tighter response with a higher quality factor as well as gain, since the negative weights resulted in a sharper drop in the response. A pair of bands has been analysed and their corresponding center frequencies and quality factors have been marked. The first and middle bands are active on the output rows. This test case serves as a starting point to figure out the coefficients required for grouping different bands.

Different combination of weights can be chosen to produce a number of cases for the overall system response. Table 5.3 lists these coefficients for the VMM weight matrix for all the system responses. These weight matrices were chosen to pass a particular set of bands and the tuning of the coefficients was done through the calibration algorithm. Along with frequency responses, the system behaviour has been studied experimentally through transient measurements as well. Figure 5.14 shows the system responses corresponding to the second weight matrix shown in Table 5.3, where the step inputs are applied to the system. The response to a step input is plotted and the rise and fall times over a period of 6ms is noted. This is characteristic of bandpass filter behavior and matches the expected

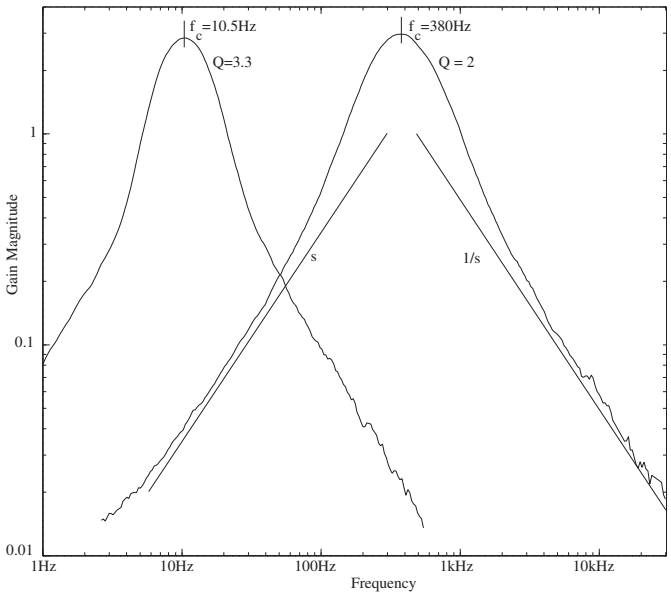


Figure 5.15: The system response is shown where the VMM is tuned such that it passes one band from the system. One could choose one band from all the six bands between 1Hz and 10KHz, that come through the system with single outputs effectively. The weights corresponding to the active bands are scaled and set while the weights for the inactive bands are set low so that a single band case comes out of the system. This case is shown where a pair of the 6 center frequencies are active at a time, namely the first and middle band, by choosing the corresponding VMM weights and the system outputs are measured. These pair of bands are analysed and plotted. These have a tighter response with a higher quality factor. The gain and their corresponding center frequencies and quality factors have been marked.

cutoff frequencies for the system. It is apparent from the time scale that it passes the middle and higher set of bands through the system. The upswing and downswing corresponding to the application of the step shows the effect of the positive and negative weights on the VMM. The feedback FG OTA on the TIA is used to set the DC level on the output rows and center the waveform with a reasonable gain. Each parameter can be individually tuned to obtain different gains and DC levels, as shown in the plots.

Frequency responses corresponding to these weight matrices are plotted as well. Figure 5.16 shows these cases, tuning for a variety of weights. An adjacent pair of bands are passed through the system on the two outputs, for the first and second weight matrix. A mixture of a lower, middle and upper frequency pairs are chosen for the responses from

Table 5.3: Coefficients of the VMM weight matrix for the programmed filter responses

Coefficients	First VMM weight matrix (nA)	Second VMM weight matrix (nA)	Third VMM weight matrix (nA)
$W^+_{1,1}$, $W^+_{2,1}$	130,0.15	0.15,0.15	125,0.15
$W^-_{1,1}$, $W^-_{2,1}$	70,0.15	0.15,0.15	75,0.15
$W^+_{1,2}$, $W^+_{2,2}$	120,0.15	125,0.15	135,0.15
$W^-_{1,2}$, $W^-_{2,2}$	75,0.15	75,0.15	70,0.15
$W^+_{1,3}$, $W^+_{2,3}$	0.15,125	120,0.15	100,0.15
$W^-_{1,3}$, $W^-_{2,3}$	0.15,75	80,0.15	40,0.15
$W^+_{1,4}$, $W^+_{2,4}$	0.15,150	0.15,105	0.15,0.15
$W^-_{1,4}$, $W^-_{2,4}$	0.15,50	0.15,45	0.15,0.15
$W^+_{1,5}$, $W^+_{2,5}$	0.15,125	0.15,120	0.15,120
$W^-_{1,5}$, $W^-_{2,5}$	0.15,75	0.15,80	0.15,80
$W^+_{1,6}$, $W^+_{2,6}$	0.15,0.15	0.15,0.15	0.15,150
$W^-_{1,6}$, $W^-_{2,6}$	0.15,0.15	0.15,0.15	0.15,50

both the output rows in the two experiments. The VMM weights corresponding to the active pair is chosen while the other bands are inactive by tuning them to a low current so that these bands are not seen in the final response.

The third weight matrix shows another experiment where three sets of bands are active at a time on one output while the other complementary bands are active on the second output of the VMM. The triplet bands case is shown, to observe the first and last set of bands separately from the system, complementing each other. The distinctive responses plotted show the range of frequencies for the twelve bandpass filters and the coefficients for the weight matrix have been targeted such that the lower corner on the second row matches the upper frequency bound on the first row output. Table 5.4 quantitatively compares the corner frequencies for the bandpass response, namely lower f_l and upper f_h frequencies of the system of the filters with the VMM, for the different weight matrices with and without the calibration algorithm. In an untuned system, the VMM weights have been initialised without compensating for the mismatch from the indirect programming. Further, the translinear FG pFETs that drive the VMM are not tuned. This causes the sources of the VMM to

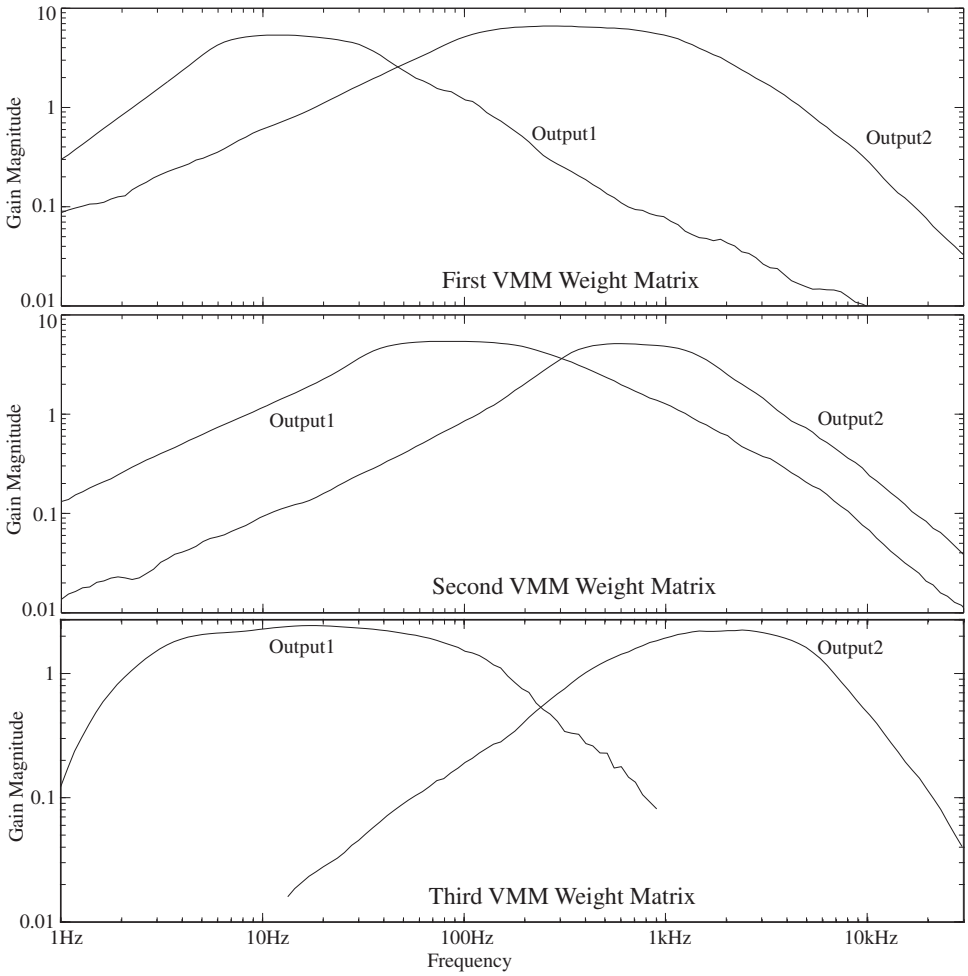


Figure 5.16: The system responses for the cases where a pair of consecutive bands are passed is shown on the two outputs for the first two VMM weight matrices. The VMM weights corresponding to the active pair is chosen while the other bands are made inactive by tuning them to a low current. The lower and middle set of bands are shown from the system for the first VMM weight matrix. The middle and higher set of frequency pairs are passed through in this response corresponding to the second VMM weight matrix. The system response is shown for a third VMM weight matrix, where three sets of bands are active at a time on one output while the other complementary bands are active on the second output of the VMM. Hence, the triplet bands case is experimentally measured, where one can observe the first and last set of bands separately from the system.

not be set close to V_{dd}, which further results in the deviation from the desired responses. Without the tuning, there are undesired deep drops between the bands in the response as well as unwanted ripples with different gains, producing a more distorted signal. Hence, the calibration not only aids in multiplying with the desired factor through the VMM, but

Table 5.4: Parameters of the system with and without calibration for the VMM weight matrices

Weight Matrix One	Output 1		Output 2	
	f_l (Hz)	f_h (Hz)	f_l (Hz)	f_h (Hz)
Untuned	3.1	2.1k	10.5	1.9k
Tuned	5.5	32.6	86.0	1.3k
Target	4.9	36.2	90	1.4k

Weight Matrix Two	Output 1		Output 2	
	f_l (Hz)	f_h (Hz)	f_l (Hz)	f_h (Hz)
Untuned	11.2	4.6k	9.7	10.1k
Tuned	30.0	314.4	314.4	1.5k
Target	36.2	325	325	1.6k

it also helps in passing the desired set of bands to the outputs.

Experimental measurements from the system have been shown to demonstrate a variety of bands related to the weighted sum of the filter responses. The BIST algorithm on the VMM aided in converging on a set of weights to find the coefficients and apply the correction for the targets, after setting the bands of the filters at the desired region. The weights can be targeted to produce a response with a higher or lower quality factor, to obtain narrow or wideband filters. Further, the system parameters enable the user to set the gain for each band and the bandwidth, as well as scale the amplitudes, to achieve interesting patterns.

Applications such as programmable filters require highly linear vector-matrix multiplication, whereas linearity and calibration might not be as significant a concern in other applications such as for classifiers with VMM+WTA [124]. BIST is advantageous where one really needs a linear, tuned circuit. Such architectures can be used in place of digital signal processors to filter inputs from sensors and produce multiple programmable functions, all on a single platform that embeds the tuning of parameters along with the desired computation aspects.

5.8 Discussion

The algorithm proposed in this work is applied and tested via implementation on a reconfigurable mixed-signal platform. The internal biases for the system can be set through the on-chip supplies and Digital to Analog Converters (DACs), while the outputs can be recorded either through the Analog to Digital converters (ADCs) compiled on-chip or IO

pads.

A built-in self-test algorithm has been shown for a differential 6x2 VMM to adapt the weights. The tuning process extends the dynamic range to translate the wide-swing input voltages to the required ranges for the source-driven inputs of the differential VMM. The process converges on the target coefficients for the weight matrix through fast injection, taking care of the threshold voltage mismatch due to indirect programming. After the initial programming duration during the first run for the entire system of switches, the iterative tuning is faster (in the order of ms) since the algorithm performs fast injection on the switch. The gain and operating range of the outputs can also be set.

The VMM has been used in a system with bandpass filters to show a variety of programmable filter dynamics, which could be tuned depending on the application. The bandpass filters could receive the input data from sensors, with the system outputs measured from the VMM block, for an efficient end-to-end computation. The variation over temperature can be compensated by using an FG based bootstrap reference circuit which biases the FG transistors [84, 132]. All the core circuits on the FPAA are referenced to ground or V_{dd}. Hence, the supply voltage variations are minimised due to the system being relative to the supplies. Typically, precision voltage references [84, 133] can be used as well to control the variation in power supply.

Due to the programmability on the FPAA, the floating gates directly enable eliminating mismatch due to threshold voltage or transistor widths and lengths. Other process variations can be tuned out after fabrication. It is critical to handle mismatch issues and variations in an analog system to obtain the desired performance and make sure that they do not become a bottleneck, to take advantage of the energy efficiency aspects of having used such systems. The self-test technique addresses this crucial question of automatically tuning the parameters for a fundamental in-memory computing block. Thereby, this algorithm could be further generalized to other analog devices and platforms as well, extending this idea to handle the mismatch for complex, large-scale systems.

CHAPTER 6

CONTINUOUS-TIME, CONFIGURABLE ANALOG LINEAR SYSTEM SOLUTIONS WITH TRANSCONDUCTANCE AMPLIFIERS

6.1 Framing Analog Solutions of Linear Equations

The solution of linear equations is a fundamental digital computation technique (Figure 6.1), that is considered difficult using analog computation [134]:

$$\mathbf{Ax} = \mathbf{b}, \tag{6.1}$$

where **A** is the input matrix, **b** is the input vector, and **x** is the solution vector. The classic digital solution uses Gaussian elimination. This computation shows all of the strength of digital processing, including the pivots, to get the maximum accuracy for the decomposition [135]. Numerical tools such as MATLAB are dedicated to these ubiquitous operations. Computing benchmarks are based on solving linear equations (e.g. LINPACK [136]). Since digital computers are well suited to solving (6.1), a problem is considered analytically solved when reduced to solving (6.1)[134].

If programmable analog techniques competitively solve (6.1) experimentally, analog techniques can span a large range of numerical analysis techniques [134, 137], enabled through analog algorithmic techniques[138]. Since a direct analog equivalent to Gaussian elimination involves numerous discrete steps and memory manipulations (e.g. pivots) that require the storage of high-resolution intermediate values, this effort considers a different approach to solving (6.1). This Ordinary Differential Equations (ODE) converges to the solution of (6.1):

$$\tau \frac{d\mathbf{x}}{dt} + \mathbf{Ax} = \mathbf{b}, \tag{6.2}$$

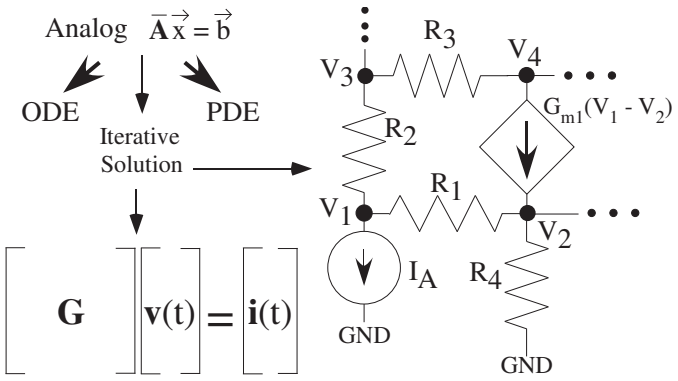


Figure 6.1: Digital and analog linear equation solution techniques include both direct (e.g. Gaussian elimination, **L U** decomposition) and iterative techniques. Analog techniques can utilize iterative techniques by using differential equations that converge to the solution. Resistive circuits with linear dependent and independent sources can be directly transformed and then solved by a set of linear equations; this work reverses that perspective to solve linear equations by transforming these equations to a circuit that converges to the linear equation solution. This transformation requires using a set of Transconductance Amplifiers (TA) as voltage-controlled current sources, enabling direct implementation in a programmable and configurable platform (e.g. FPAA), thereby enabling the solution of a wide range of matrices.

where τ is the network time constant. One discretization is

$$\mathbf{x}[n] = \mathbf{x}[n-1] + \epsilon \left(\mathbf{b} - \mathbf{A}\mathbf{x}[n]\right), \tag{6.3}$$

where ϵ is a function of τ and the time step. Sometimes, iterative digital techniques in some cases require fewer operations than digital Gaussian elimination, particularly for sparse matrix solutions as well as for embedded computations (e.g. [139]). Equation 6.2 and 6.3 converge for positive eigenvalues of **A**; one can find related iterative equations for other **A** [135].

This work focuses on analog solutions to (6.1) using (6.2). Reconfigurable and programmable Transconductance Amplifiers (TA) implement (6.2), and provide a platform for analyzing the analog numerics. An SoC Field-Programmable Analog Array (FPAA) [2] provides the experimental demonstration platform; these approaches could be implemented in earlier FPAA devices (e.g. [140]) or custom silicon. Engineering students are taught that static resistive circuits *with independent and dependent voltage- and current-*

sources are directly formulated and solved as a system of matrix equations like (6.1). This work experimentally demonstrates and characterizes the other side of this statement that configurable TA configurations solve (6.1) through (6.2) for any stable **A** (Figure 6.1). A resistor-only network is limited in the possible **A** matrix, while resistors with op-amps may have stability, mismatch, or parasitic concerns in physical implementations [141, 142, 143, 144, 145]. These approaches encompass early theoretical discussions of analog computing for solving (6.1) [146, 147], as well as theoretical discussions using multiple-output TA-based recurrent neural networks for solving (6.1) [148, 149]. Electronic circuits are used to illustrate solutions (e.g. Hopfield networks [150, 151]), simulating linear solutions for a reduced class of problems [152, 153, 152], and considering specialized cases (e.g. elliptic Partial Differential Equations (PDE) [154, 155].)

If an analog linear equation solver is built, one would want this computation to have at least a fraction of the typical 1000$\times$ improvement in computational energy efficiency (e.g. [1]) compared to digital techniques (e.g. [156]), as well as typical area efficiencies. Vector-Matrix Multiplication (VMM) shows these efficiencies compared with digital computation [157, 158]. This chapter will discuss the potential energy efficiencies as well as algorithmic complexities.

This work moves to unify analog solutions of linear systems into a low-energy, compilable approach, including discussing its wider numerical analysis and resulting circuit techniques. The discussion moves towards transconductance-based (Figure 6.2), and resistive-based iterative methods for solving linear equations (Section 6.2), describing the relevant FPAA implementation details (Section 6.3), as well as demonstrating analog solutions of representative linear systems (Section 6.4). The resulting analog linear equation solution circuits are effectively analog filters or control systems, and we discuss the connection to these approaches (Section 6.5). Finally, the discussion moves towards analyzing computational efficiencies of linear equation solvers and concludes the discussion in Sections 6.6 and 6.7.

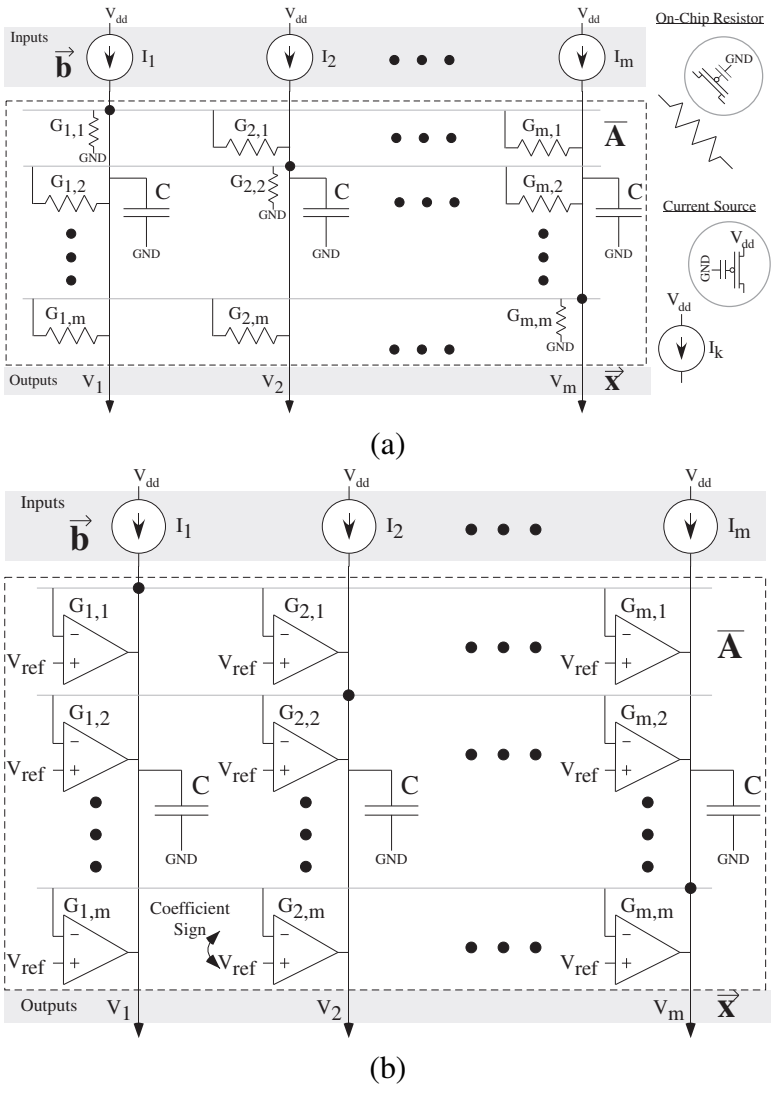


Figure 6.2: Potential continuous-time circuit architectures to solve systems of linear equations. Shaded areas show the input **b** and output **x** regions. Connection dots show connections at connecting wire intersections. A *T* would be a connection. (a) Linear equation solution built from resistors and current sources. The programmable resistors and current sources can be implemented with programmable transistors. (b) Linear equation solution built from Transconductance Amplifiers (TA). The constraint matrix of voltage-controlled current sources is set by individual conductances and the corresponding bias currents.

6.2 Physical ODE Solutions of Linear Equations

Analog linear equation solutions could be implemented using resistive coupling between nodes (Figure 6.2a), and/or transconductance amplifier coupling between nodes (Figure 6.2b).

FPAAs efficiently implement resistive and transconductance networks, including exploiting routing fabric elements [2, 159].

A linear system built of resistors creates a diagonally dominant system with negative non-diagonal coefficients. Undergraduate engineering students would recognize that low-frequency circuits of resistors and supplies are modeled by linear system of equations. A generic resistor network for solving a system of linear equations (Figure 6.2a) is modeled as

$$I_l = -\sum_{k \neq l}^{m} G_{l,k} V_k + V_l \sum_{k=1}^{m} G_{l,k} + C\frac{dV_l}{dt}, \tag{6.4}$$

where $G_{k,l}$ is the inter-node conductance, $G_{k,k}$ is the node conductance, and k, l represent the matrix indices of **A** that are of size m. This stable ODE converges to the solution.

Physical computation uses continuous variables as representations. These variables are often scaled, including the units, to abstract the computation from physical values. For example, users solving linear systems often prefer representing their problem as a normalized variable, x, that varies from 0 to 1, rather than keeping track of an arbitrary range of currents (e.g. 14.6nA and 31.7nA). Knowing the abstraction, tools should remove these low level details from the user. Normalizing these variables converts the physical equations to mathematical equations. The smallest of the diagonal sums, $\sum_{k=1}^{m} G_{l,k}$, typically normalizes the resulting equations. The time constant τ is set by a capacitance, C, over a value proportional to the conductances. The relationships connect (6.4) to (6.2):

$$a_{l,k} \propto -G_{l,k} \;\; k \neq l$$

,

$$a_{k,k} \propto \sum_{k=1}^{m} G_{l,k}, b_k \propto I_k. \tag{6.5}$$

Input currents are signed, and the output voltages are signed values around some bias point. All diagonal values will be greater than 1; all off-diagonal values will be negative.

Transconductance amplifiers (TA) achieve a larger **A** range compared to resistive el-

ements by using voltage-controlled current sources and not just resistor elements. A TA operating in its linearized region is expressed as

$$I_{out} = G(V^+ - V^-), \tag{6.6}$$

where G is the circuit's transconductance parameter. When the 9-transistor TA (as in [31]) in the SoC FPAA [2] is programmed in the typical case with a subthreshold bias current (I_{bias}), the resulting transconductance, and coupling between nodes (k,l), would be $G_{k,l} = \frac{\kappa I_{bias}}{U_T}$. where κ is the capacitive division between gate and surface potential for a MOSFET (e.g. [31]), and U_T is the thermal voltage, kT/q (≈25mV at T=300K). Similarly, an expression for G can be derived for different bias current regions (e.g. above-threshold bias currents) with smaller increases in G for further increases in I_{bias}. An on-chip amplifier, such as the TA in on-chip custom [31] and configurable [2, 10] designs, becomes a voltage or transconductance amplifier depending on the resulting circuit operation as they are often unbuffered designs. The SoC FPAA has 196 non-FG input TAs and 196 FG input TAs [2]. As the SoC FPAA TA bias current sources are set with an internal FG element, these TAs do not have a fourth terminal (e.g. [31]) to set the bias current.

Linearized TAs operating with subthreshold or near-threshold currents are governed by

$$C_L\frac{dV_{out}}{dt} + \frac{\kappa I_{bias}}{2U_T}(V_{out} - V_{ref} + V_{off1})$$

$$= \frac{\kappa I_{bias2}}{2U_T}(V_{in}(t) - V_{ref} + V_{off2}), \tag{6.7}$$

including the TA input voltage offsets (V_{off1}, V_{off2}). For a large input voltage where the TA becomes a current source,

$$C_L\frac{dV_{out}}{dt} + \frac{\kappa I_{bias}}{2U_T}(V_{out} - V_{ref} + V_{off1}) = I_{bias2}. \tag{6.8}$$

The steady state solutions for the linearized and current source cases are

$$V_{out} = V_{ref} - V_{off1} + \frac{I_{bias2}}{I_{bias}}V_{off2}$$

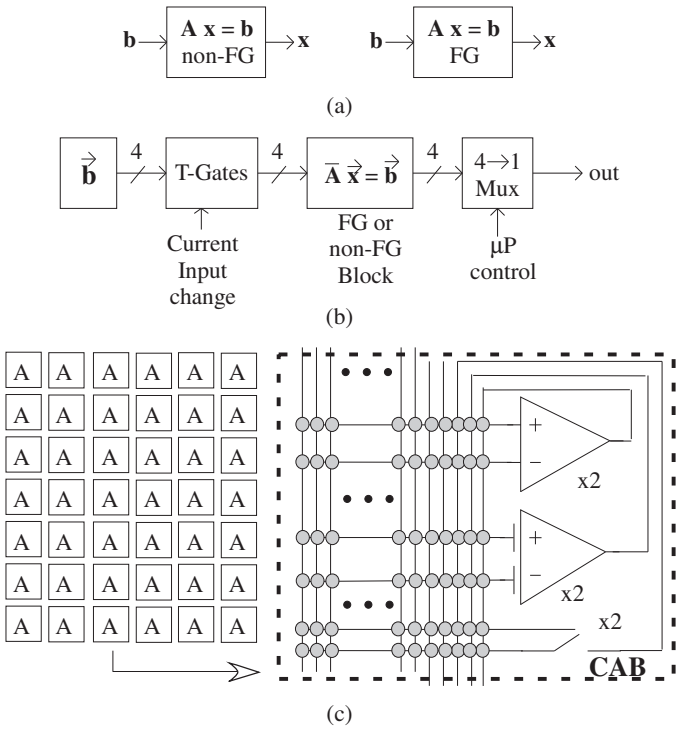


Figure 6.3: Analog linear equation solution in the SoC FPAA [2]. (a) High-level tool (Scilab) blocks for the Floating-Gate (FG) and non-FG TA based linear equation solution. (b) A representative block diagram setup for computing and measuring a linear equation solution, where each line represents a parameterized bus of inputs or outputs. This diagram shows a 4x4 **A** as used in later experimental examples. (c) Arrays of FG and non-FG TAs in SoC FPAA Computational Analog Blocks (CAB) perform the linear equation solutions.

and

$$V_{out} = V_{ref} - V_{off1} + \frac{2U_T}{\kappa}\frac{I_{bias2}}{I_{bias}}, \tag{6.9}$$

respectively.

A generic TA network for (6.2) (Figure 6.2b) for the l-th row with subthreshold I_{bias} is modeled as

$$C\frac{dV_l}{dt} = I_l - \sum_{k=1}^{m} G_{l,k}V_k \tag{6.10}$$

The normalizations connect (6.10) to (6.2):

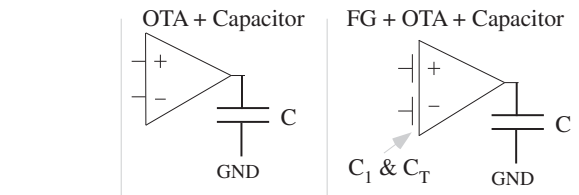
$$a_{l,k} \propto G_{l,k}, b_k \propto I_k \tag{6.11}$$

One could have a different τ per row due to different conductance modeling and capacitances, potentially tuned to optimize convergence. Current sources set the **b** inputs. Positive current sources would go to V_{dd}, and negative current sources would go to GND. The ODE solution of these networks requires positive eigenvalues for **A**; related techniques can transform the resulting matrix for the solution of general **A** (e.g. [153]). Often, quantities are built around a single bias current, I_{bias}, which abstracts the resulting currents (typically one of the programmed current values), resulting in $\tau = \frac{CU_T}{\kappa I_{ref}}$. The current scaling is a question of computation speed. Each output row could be scaled accordingly, setting each row's resulting τ and normalizing the row values as required. Each row in (6.2) could have a different τ.

Physical linear equation solutions, transforming a linear system into an ODE problem, should involve problems that start as a series of linear equations. It is inefficient to take an ODE or PDE, transform it into a linear system, and then to transform it to a linear ODE circuit for the solution. As ODE / PDE applications are efficiently solved by direct implementation [134, 159], we focus on the physical solution of linear systems that originates from different applications.

6.3 Configurable Analog Linear Equation Solver

The TA-based analog linear equation solution is experimentally demonstrated in an SoC FPAA [2]. The SoC FPAA is enabled by a significant infrastructure and tool base; an extensive overview of FPAA devices is written elsewhere [10]. The SoC FPAA operates with analog supplies at 2.5V and ground. This computation is encapsulated in a linear equation block (Figure 6.3a), abstracting the analog TA computation [16] that can be targeted to an FPAA IC, as well as a macromodel simulated in a full system level simulation (level=1), in an open-source toolset in Scilab [12]. The block parametrizes the number of inputs (**b**) and outputs (**x**) and the resulting matrix (**A**). This discussion presents experimental circuit measurements and simulations



	OTA + Capacitor	FG + OTA + Capacitor
Max Signal Size	$\frac{2\,U_T}{\kappa}$	$\frac{C_T}{C_1}\frac{2\,U_T}{\kappa}$
Time-Constant (Gain =1)	$\frac{\kappa\,I_{bias}}{2\,U_T}$	$\frac{C_T}{C_1}\frac{\kappa\,I_{bias}}{2\,U_T}$
Total Noise (gain=1)	$\sqrt{\frac{q\,\kappa\,U_T}{2\,C}}$	$\sqrt{\frac{C_T}{C_1}\frac{q\,\kappa\,U_T}{2\,C}}$
SNR (amplitude, gain=1)	$\sqrt{\frac{2\,C\,U_T}{q\,\kappa}}$	$\sqrt{\frac{C_T}{C_1}\frac{2\,C\,U_T}{q\,\kappa}}$

Figure 6.4: Comparison of non-FG TA and FG TA for solving $\mathbf{Ax} = \mathbf{b}$. The FG TA results in higher voltage signals and SNR, as well as larger τ by the ratio of total capacitance (C_T) to the input coupling capacitance (C_1).

This linear block becomes part of the experimental on-chip test setup (Figure 6.3b) that includes providing the **b** input as well as multiplexing each value of the output vector **x**. TAs in the Computational Analog Blocks (CAB) have a 9 transistor circuit topology (Figure 6.3c) [2, 31]. The **A** matrix is set by the conductances of the individual TAs, that in turn are set by Floating-Gate (FG) bias currents set by the pFET bias current. The TA has nearly rail-to-rail output range, although the upper range is limited by the pFET current source remaining saturated. Typical operation occurs for signals around a 1V to 1.25V reference (V_{ref}). A TA is used to convert the input voltage (**b**) signal to a current. The reference voltage(s) can be controlled by Digital-to-Analog Converters (DAC) compiled on the FPAA [2]. The sign of each **A** matrix element determines the TA input sign, where positive values are input in the - input and the reference to the + input, while negative values are input in the + input and the reference to the - input. The measured outputs, **x**(t), are scanned and buffered out.

The choice of FG or non-FG TA depends on the required linearity and convergence (Figure 6.4). FG TA has higher linearity with a slower time constant for a given bias cur-

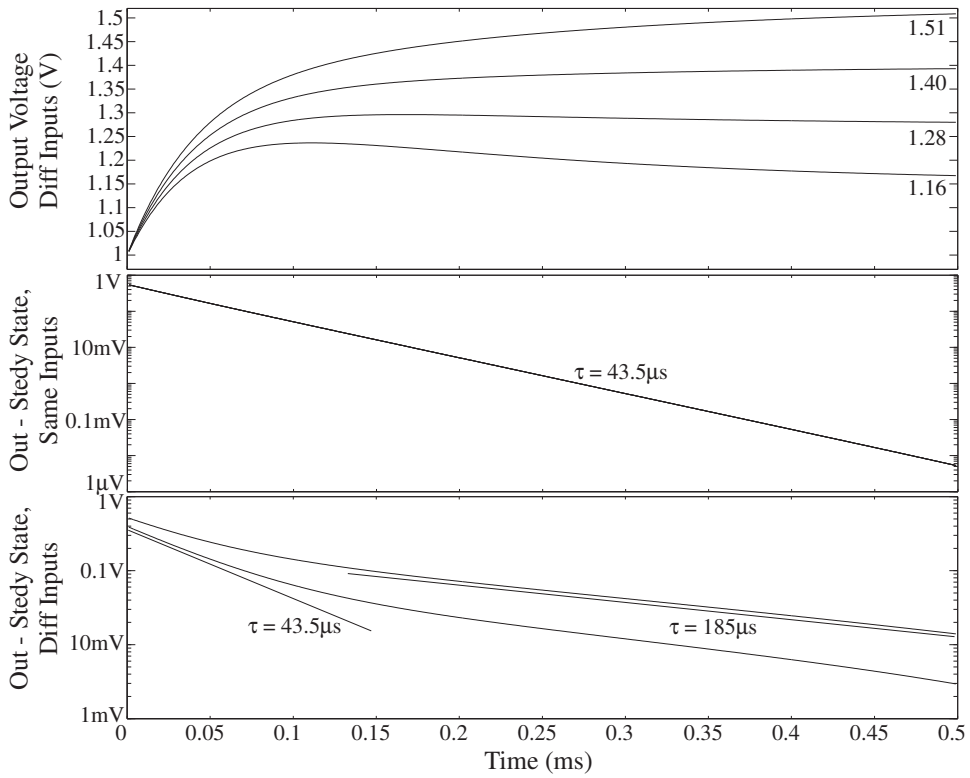


Figure 6.5: Measured system level (level=1) simulation results for a 4x4 FG TA for a matrix programmed with 100nA on the diagonals, 50nA for the off diagonal elements, and an input (**b**) that switches between no current and its particular current level. The graph shows two cases of the input vector (**b**), one case (*same inputs*) for **b** = [300nA 300nA 300nA 300nA], and a second case (*different inputs*) for **b** = [50nA 75nA 100nA 125nA]. These results are compared for an equivalent, more detailed level=2 [15] simulation model, and the results compare closely with experimental measurements.

rent, while a non-FG TA has a lower linear range and provides a faster convergence for a given bias current. There have been other discussions in the literature about a range of programmable current bias TA [160], the effect of FG capacitance coupling on core circuit parameters [161], built-in self-test algorithms [162], and application of these structures in nonlinear dynamics [163] and education [164]. We summarize these core results to facilitate our linear-equation solutions (Figure 6.4). The non-FG TA is more energy efficient for a given bias current. The FG TA is easier to instrument using 1V signals rather than 50-100mV signals. The key parameter that scales these results is the ratio of total capacitance (C_T) to the input coupling capacitance (C_1). The experimental measurements will illustrate

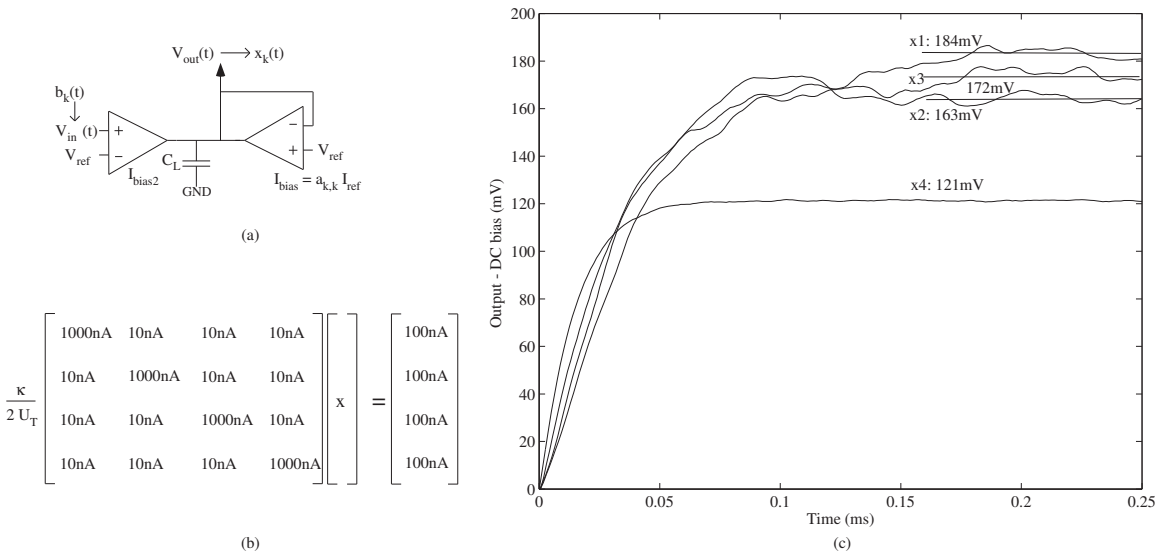


Figure 6.6: Setting up $\mathbf{Ax} = \mathbf{b}$ using TA elements. (a) Effective circuit for solving $\mathbf{Ax} = \mathbf{b}$ for a diagonal matrix $\mathbf{A}$, that corresponds to a first-order TA circuit with tunable gain based on the ratio of the bias currents. (b) Effective matrix programmed for this diagonal computation (1μA as the *1* elements and 10nA as the *0* elements), where the off diagonal elements programmed to 1% or less of the diagonal elements (1μA compared to 10nA). Effectively, the off diagonal elements can be ignored in these measurements. (c) Analysis for the convergence of $\mathbf{A}$ matrix after applying a 40mV $\mathbf{b}$ input through the TA current sources. The time constant was extracted to be roughly 47μs for each component. It is nearly identical for all the four curves, thereby getting similar convergences.

the properties of these two TA approaches (Figure 6.4) corresponding to these two blocks (Figure 6.3a). Using the FPAA routing fabric results in efficient resistive networks [2, 159], although they typically result in lower SNR and signal amplitudes than TAs.

The system model (level=1) for this TA based equation solver, where voltages are referenced to V_{ref}, and where the inputs use a similar FG-based TA, would be

$$\frac{dV_l}{dt} = \frac{I_{b,l}}{C} \tanh\left(V_{x,l}/V_L\right) - \frac{1}{C} \sum_{k=1}^{m} I_{A,l,k} \tanh\left(V_k/V_L\right), \tag{6.12}$$

where $\mathrm{I}_{b,l}$ are the bias currents for the input FG TAs (**b**), $\mathrm{V}_{x,l}$ are the input voltages (**b**), $\mathrm{I}_{A,l,k}$ are the bias currents for the matrix TAs (**A**), V_L is the linear range of the TA, and we assume a nominal value of C (e.g. 1pF) until compilation provides better data that gives a better estimate, including the circuit place and route. This model is implemented for the

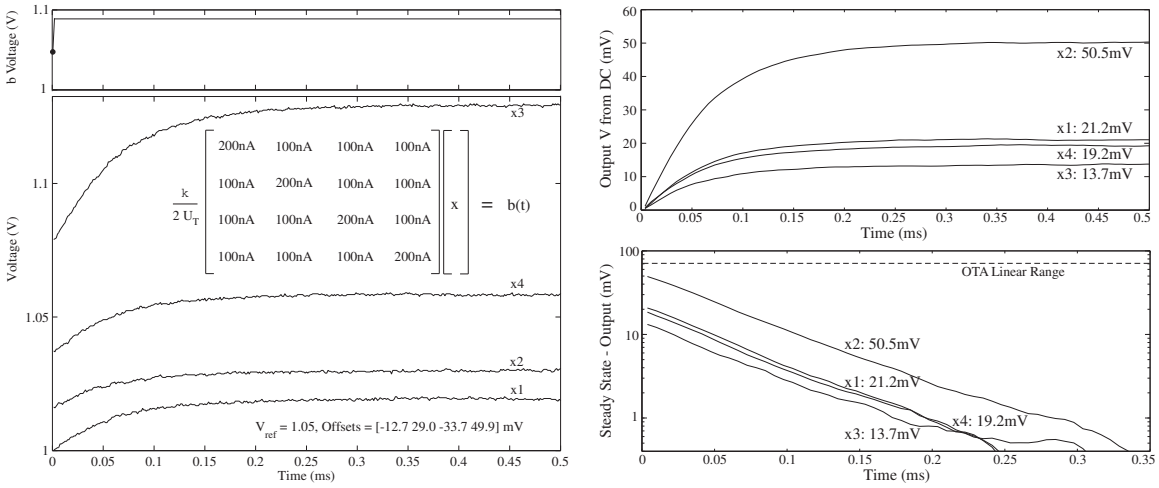


Figure 6.7: Measured convergence **x** for **A** matrix with 200nA and 100nA as the diagonal and off-diagonal elements, respectively. For a 40mV **b** input applied as current sources, the x(t) solution is plotted and is within the linear range of the TAs. The time constant (61μs) from curve-fitting the log(·) of the step responses; as expected, there are three identical eigenvalues for this matrix. One of the components along the larger eigenvalue (by a factor of 5) converges 5 times faster in 12μs.

FG TA block, and can be directly modified for the non-FG block. One can show results from this abstracted simulation model (from (Equation 6.12)), and compare it to a more detailed transistor level simulation [15] and experimental measurements (Figure 6.5).

6.4 Linear Equation Experimental Dynamics

Experimental measurements from a 4x4 **A** matrix illustrate the linear equation solver (Figure 6.3b) dynamics. The compiled 4x4 linear solver requires 16 TAs to implement the 4x4 **A** matrix, and 4 TAs to implement the **b** matrix. The **b** vectors are inputs fed to the gate input of the TA structure, step functions from a *zero* point (= fixed potential or current switched off) to a desired input for **b**. The matrix solution outputs, *x(t)* , show the dynamics when new inputs are applied.

An identity matrix for **A** illustrates the solution circuit dynamics (Figure 6.6). The circuit model (Figure 6.2, non-FG TAs) simplifies to a group of two-TA components for a diagonal **A** (Figure 6.6). The time constant is directly related to the diagonal TA bias current

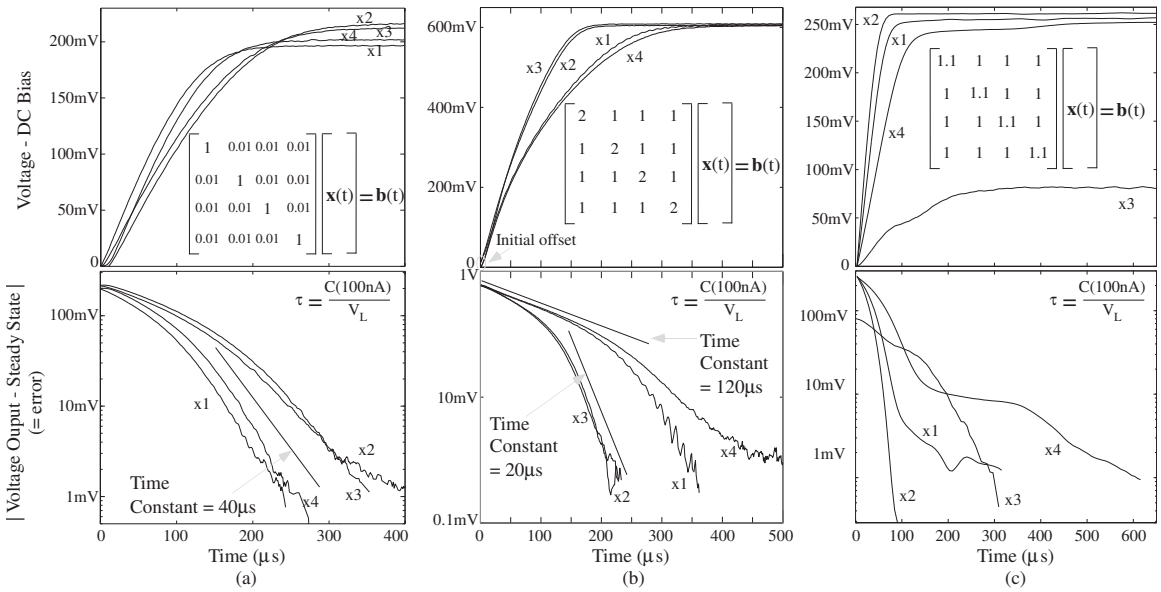


Figure 6.8: Measured solutions for a FG TA based linear equation solution circuit. Each case shows the measured results and the log trajectory (or error) towards the steady state. (a) Solution of diagonal matrix with 100nA diagonal elements. (b) Solution of 200nA diagonal elements and 100nA off-diagonal elements (c) Solution of 110nA programmed diagonal elements and programmed 100nA elements. Some mismatch resulting from indirect programming was not compensated, so the elements had some random variation.

and the FPAA routing capacitive load, showing unprogrammed mismatches. In general, we can normalize each row, effectively changing the time constant, but not affecting the final steady-state solution. Linearized TAs operating with subthreshold or near-threshold currents are described by

$$C_L \frac{dV_{out}}{dt} + \frac{\kappa I_{bias}}{2U_T}(V_{out} - V_{ref} + V_{off1})$$

$$= \frac{\kappa I_{bias2}}{2U_T}(V_{in}(t) - V_{ref} + V_{off2}), \tag{6.13}$$

where we include the input voltage offsets of the TAs (V_{off1}, V_{off2}). If the input TA has a large input voltage where that TA becomes a current source,

$$C_L \frac{dV_{out}}{dt} + \frac{\kappa I_{bias}}{2U_T}(V_{out} - V_{ref} + V_{off1}) = I_{bias2}. \tag{6.14}$$

Steady state solution for linearized and current source cases are

$$V_{out} = V_{ref} - V_{off1} + \frac{I_{bias2}}{I_{bias}} V_{off2}$$

and

$$V_{out} = V_{ref} - V_{off1} + \frac{2U_T}{\kappa} \frac{I_{bias2}}{I_{bias}}, \tag{6.15}$$

respectively. The voltage offsets could be tuned out by using FG-TA elements. If the **A** elements are operating in their linear region, one can define *zero* at any particular offset because if the outputs (**x**) are measured around an offset vector ($\mathbf{x}_0$) due to an offset ($\mathbf{b}_0$) in the input (**b**), then one can simply normalize the output around the starting zero point because we are solving a linear system, $\mathbf{A}(\mathbf{x} - \mathbf{x_0}) = \mathbf{b} - \mathbf{b_0}$.

Analyzing (6.2) illustrates the circuit dynamics that can be experimentally verified. **A** can be written as

$$\mathbf{A} = \mathbf{E}\mathbf{\Lambda}\mathbf{E}^{-1}, \tag{6.16}$$

where $\mathbf{\Lambda}$ is a diagonal matrix of eigenvalues, and **E** are the corresponding rows of normalized (power = 1) eigenvectors corresponding to the particular eigenvector. This relationship simplifies to $\mathbf{A} = \mathbf{E}\mathbf{\Lambda}\mathbf{E}^T$ for symmetric **A**. Transforming the solution **x** into a projection along the eigenvector basis, $\mathbf{x} = \mathbf{E}\mathbf{y}$ for (6.1), we project along the eigenvector basis to get

$$\tau \frac{d\mathbf{y}}{dt} + \mathbf{\Lambda}\mathbf{y} = \mathbf{E}^{-1}\mathbf{b} = \hat{\mathbf{b}},$$

$$\frac{\tau}{\lambda_k} \frac{dy_k}{dt} + y_k = \frac{\hat{b}_k}{\lambda_k}, \tag{6.17}$$

where y_k is the kth component of **y**, and λ_k is the kth eigenvalue of **A**. The matrix requires positive λ_k values, although through transformations in **A** and **b**, one could achieve positive values (**A** is positive definite). Depending on the projection of **b** on the eigenvector basis, the solution could have the effect of one or all of the eigenvectors. The time constants are scaled by λ_k, so the largest eigenvalue component will converge first, and the smallest

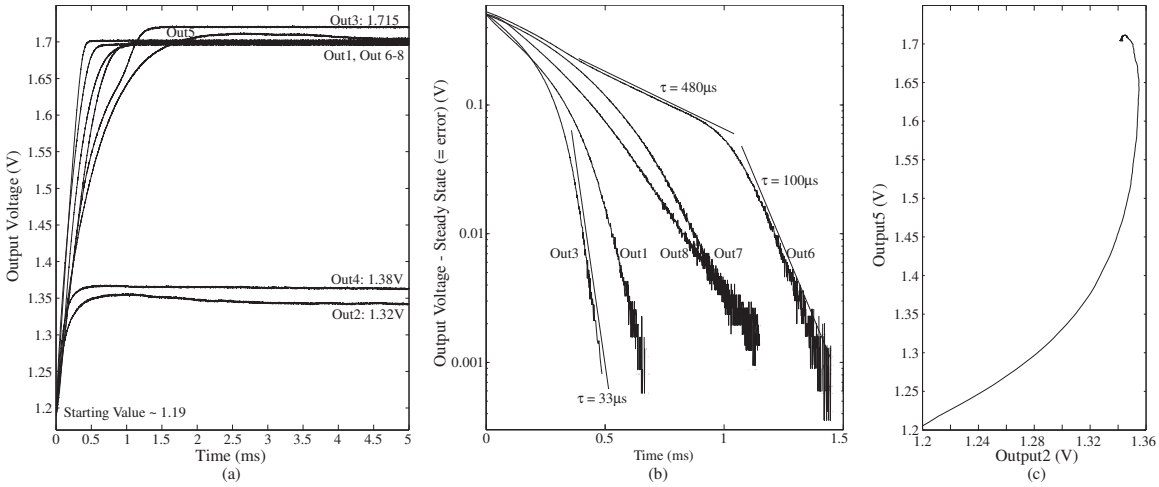


Figure 6.9: Dynamics for a compiled 8x8 **A** matrix with the diagonal elements programmed to 200nA and the off-diagonal elements programmed to 100nA. (a) Starting from an initial value, each of the eight nodes converges to their final value. (b) Several of the outputs directly converge to their steady state solution (Out1, 3, 6, 7, 8), although they might have different time constants depending on the convergence of other nodes. (c) Some of the outputs may overshoot or have a damped oscillation into their steady-state solution (e.g. Out2 vs. Out5).

eigenvalue component converges last; it is often the component most noticed in the system dynamics. As the value of τ could be normalized along each row, one could compensate for the slower response of the smaller λ_k values, although these changes will be projected onto the eigenvector basis.

Programming different **A** matrices can illustrate these dynamics (Figure 6.7). The dynamics are studied by plotting the matrix solution outputs, **x**(t) which are the steady-state responses. The effective time constant from the three eigenvalues is 61μs, obtained through curve-fitting the exponential curves from the step responses. Since one of the eigenvalues is larger by a factor of 5, it converges 5 times faster, in 12μs.

The FG TA implementation enables further investigation of the linear solution dynamics given the larger signal amplitudes and SNR. The larger signal amplitudes from the FG TA devices enable observation of the multiple time constants (Figure 6.8), particularly when we choose matrices with less symmetry and identical eigenvalues (Figure 6.8c). Scaling the currents of the **A** matrix also scales the convergence time (Figure 6.8a versus

Figure 6.8b). The larger eigenvalue spread results in a larger spread in the resulting time constants (Figure 6.8d). This case is an example in our FPAA graphical tools.

The core block can compile solutions to larger **A** matrices, including 8x8 (e.g. Figure 6.9), where the largest square matrix on an SoC FPAA would be 14 x 14 resulting from the 196 FG TAs and/or 196 non-FG TAs. A trajectory may converge to the solution through an oscillatory path (e.g. imaginary roots, elliptic paths), depending on the rows of **A**, where a particular variable might overshoot or spiral into the solution (e.g. Figure 6.9). One could imagine building TA with other elements, including routing elements, to increase the matrix size. Fewer elements are needed for a sparse matrix, where one only needs one particular element per value in the **A** matrix.

6.5 Analog Linear Solutions as Linear Filtering

A different look at (6.2) through the TA circuit configurations (Figure 6.2b) illustrates that this linear equation solver sets up a linear filter between the inputs **b** and outputs **x**. The 4x4 equation solver in Figure 6.8b) shows a low-pass frequency response when applying a sinusoid to one of the **b** inputs (Figure 6.10a), as well as a low-pass chirp signal response (Figure 6.10b). This second-order low-pass response from the higher-frequency attenuation is consistent with the step-response dynamics (Figure 6.8).

The solution of linear equations transforms into filter design and approaches used in control system implementations. One might be able to employ this transformation to optimize for a filter transfer function using a full **A** relaxing other parameter constraints, often expressed as requirements for high resonance and/or minimizing energy requirements. G_m-C filter cascades of first-order and second-order filter components match directly to the linear equation solver with TA components (Figure 6.2b). Second-order filter sections, which are cascaded for many filters, are equivalent to a linear equation solver. These circuits illustrate wave-propagating behavior where the delay would linearly scale with the number of components, consistent with cascades of TA elements, including unity gain devices or

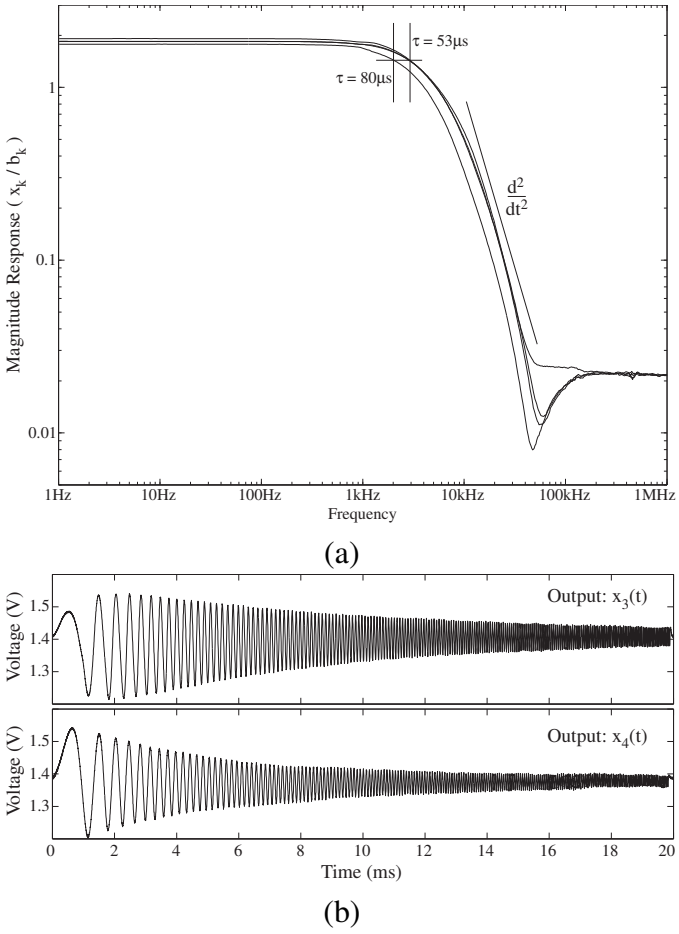


Figure 6.10: Linear equation solver characterized as a linear filter for a 4x4 FG TA configuration. The **A** matrix was the same as in Figure 6.8b. (a) Frequency response from input (**b**) to the output (**x**) vectors. (b) The output response of a linear equation solver for a chirp input (**b**), shows an expected response of a second-order low-pass filter as in (a). The chirp input sweeps from 1Hz to 20kHz over a time duration of 10ms as a 20mV sinusoidal offset around 1.25V. The output response shows the attenuation of the higher frequencies corresponding to a first-order low-pass filter.

cochlear models (e.g. [31]).

6.6 Computational Efficiencies of Linear Equation Solvers

This discussion focuses on the architectural tradeoffs and efficiencies for analog solutions of (6.2) as well as some digital alternatives. If the application comes from a directly solvable physical system (e.g. PDE computations), one would use those more natural tech-

Table 6.1: Computational Efficiency for m = 16, V_L = 1V, and V_{dd} = 2.5V

	Freq (f)	Power / node	Comp Eff MMAC(/s)/μW	SNR (power)
I_{bias}	$\propto \frac{I_{bias}}{2\pi C V_L}$	$2m^2 V_{dd} I_{bias}$	$\frac{1}{2\pi C V_L V_{dd}}$	$\frac{2V_L C m}{q}$
1nA	800Hz	1.28μW	0.32	78dB
100nA	80kHz	128μW	0.32	78dB

niques for that application. Both analog and digital techniques have similar tradeoffs for sparse computation, particularly with configurable analog capabilities [10].

Gaussian elimination can be represented as decomposing **A** into a Lower diagonal matrix (**L**), and an Upper diagonal matrix (**U**). This technique can be useful when solutions for multiple inputs (**b**) are required. Analog techniques directly solve these two matrices, linearly propagating each of the results in a similar fashion to a digital solution for **L** and **U**, retaining the typical improvement for an analog system over a digital system (e.g. [158]). This analog operation could be transformed to a Vector-Matrix Multiplication (VMM). One might digitally decompose a single **A** into **L** and **U** and further download them into the analog solver for continuous analog processing.

The computational efficiencies between digital and analog solvers is shown, in a similar way to VMM comparisons [157]. Since each TA effectively computes a Multiply-ACcumulate (MAC) operation in addition to ODE integration, we compare the MAC operations at a given frequency. Table 6.1 shows this model assuming an average I_{bias} over the m values on a row or column, and C is the single element total capacitance (C = 200fF).

Analog techniques to solving linear equations do have computational energy efficiency improvements compared with digital techniques, although not quite the 1000$\times$ advantages over digital computation in this configurable platform. The TAs in this configurable framework have higher capacitance than other algorithms, such as VMM computations [158]. A scaled down FPAA device, an optimized FPAA device, or a custom IC implementation would result in substantially smaller capacitances and similar VMM efficiencies. In a custom solution, a TA could be built using a single FG device. One would have similar

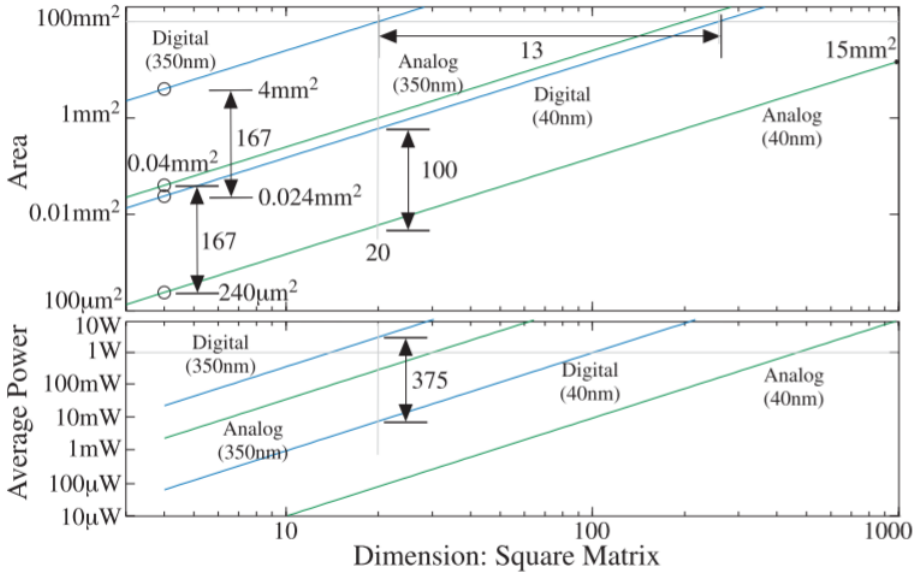


Figure 6.11: Linear equation solution comparison ($n \times n$ matrix) between analog versus digital approaches at 350nm CMOS and projected 40nm CMOS both in area and in average power consumption. The area improvement between analog approaches at 350nm and at projected 40nm CMOS is 167$\times$ the digital approach in the same process, while the power also scales correspondingly.

VMM crossbar computational efficiencies, preserving the 1000$\times$ factor for analog computation compared to the digital efficiency wall of 40MMAC(/s)/mW (16bit registers) [156], to obtain similar numerical results for analog computations [134].

These analyses allow a comparison between a custom analog and a custom digital solution at 350nm CMOS, as we have measured components in this process as well as extrapolate for 40nm CMOS [165], showing roughly a 100$\times$ area improvement and $350\times$ energy efficiency improvement (Figure 6.11).

Many linear equation formulations are transformations of linear sets of ODEs or PDEs, transforming the two dimensional space into a one-dimensional vector, enabling these solutions for digital computation. Solving this linear set of equations, either in whole or in blocks, by analog circuits seems highly inefficient, although such viewpoints are sometimes considered (e.g. [154, 155]). The physics behind solving linear systems may be useful for other ODE solutions, and these techniques should be used for those ODE appli-

cations where applicable.

6.7 Summary and Discussion

The chapter discussed solving systems of linear equations using analog computation, transforming the linear system solution to a set of ODEs. The technique is related to iterative digital methods for solving linear equations. These approaches extend the energy efficient properties of analog computing initially shown for vector-matrix multiplication to solutions of linear systems, where the vector-matrix multiplication happens through arrays of TAs.

The dynamics and convergences are studied for different matrices, through experimental measurements of the matrix output solutions from hardware. Analog solutions of linear systems typically is among the most challenging algorithm for analog computation. Hence, finding analog algorithmic solutions for linear systems opens up an entire range of high-performance analog computing. This approach allows for the solution of any positive definite **A** matrix through the use of TA devices, and not limited as in resistive coupling networks.

These techniques could be extended towards building a canonical nonlinear function solver by mixing different types of transconductance amplifiers (e.g. built on an FPAA). Different TA circuits result in different even and odd nonlinearities, allowing the direct implementation of second and third-order normal forms within a similar architectural framework. Considering such nonlinearities expands solution spaces to include oscillatory systems. This capability would further enable compilation and synthesis of nonlinear ODEs in experimental hardware, as well as a framework for further theoretical development of applications with nonlinear functions.

CHAPTER 7
DISCUSSION AND CONCLUDING REMARKS

The objective of the research is to demonstrate energy-efficient computation on a configurable platform, an FPAA, by leveraging analog strengths, along with our development of a framework, to enable real-time systems on hardware. This work demonstrates the design of fundamental blocks like Hodgkin Huxley neurons and synapses, for building up synfire chains and WTA circuits. To enable this computation and scale up from these modules, methods to address component variations have been demonstrated through our framework. Other applications such as the solution of linear systems of equations have also been demonstrated on hardware. By leveraging the strengths of silicon, these techniques provide several opportunities towards building energy-efficient systems.

7.1 Research Summary

Chapter 2 presented experimental silicon results on the dynamics of a Hodgkin-Huxley neuron, inspired by the similarity between biology and silicon, by modeling ion channels and their time constants. Further, action potential dynamics consisting of spiking responses from different inputs and with different parameters was shown.

Chapter 3 described a simulator to model fundamental components like the transistors, amplifiers and FG devices based on the EKV model. Systems including continuous-time filters and the analog front-end of a speech processing system have been built from these basic components to demonstrate a close agreement between the simulated results and the experimental measurements. Further, it also introduced models for simulating various analog circuits at different temperatures.

Chapter 4 described neuronal responses through excitatory and inhibitory synapses to observe post synaptic potentials from transistor channel neuron models. Further, the mod-

els have been used to build spiking networks such as synfire chains and WTA functions.

Chapter 5 focused on the tuning algorithm for setting the weights on VMM as well as to producing levels of voltage near the power supply rails to the source-driven VMMs, by application of a wide range of input levels. Constraints during the design and implementation process, accounting for device mismatch were discussed. Further, an application of this algorithm was demonstrated through a set of programmable bandpass filters with the tuned VMM.

Chapter 6 addressed a programmable linear equation solver. A set of differential equations using transconductance devices directly translated from circuit theory converges to the linear equation solution. These energy-efficient analog techniques are experimentally demonstrated through a set of TA based networks.

7.2 List of Contributions

- **Macromodeling and design of a level=2 simulator**: I designed a simulator to model circuit level elements, primarily CAB components and FG devices, and further, showed a close overlap between the simulated results and experimental data for the front-end of a speech processing system. This work was published in the JAICSP journal [15] and the ISCAS conference [166, 167].

- **Modeling temperature dependence for level=2 models**: The EKV models and simulator I developed were used to characterize behaviour over a range of temperatures. This was done in collaboration with Sahil Shah and Hakan Toreyin and the work resulted in publications in TVLSI [84] and JLPEA [132] journals.

- **Abstraction of analog systems**: The design of library blocks and abstraction of elements in the tool infrastructure were collaboratively done with Sahil Shah and Sihwan Kim. This work was published in the JLPEA journal [16] and WOSET workshop [168].

- **Hodgkin Huxley neuron on FPAA**: I designed a transistor channel based HH neuron model, adapted from the classical structure [20] and implemented in one CAB on the FPAA. I demonstrated the replicability of the spiking of HH Neurons through experimental measurements across three chips. This work was published in the TBIOCAS [88] journal and the ISCAS conference [102].

- **Programmable filters and built-in self-test for VMM**: I conducted experiments on chip for a BIST algorithm for a 6x2 VMM, along with the MITE interface circuitry. I used this algorithm to demonstrate a set of programmable filters with a VMM. This work was published in the ISCAS conference [169].

- **Analog solutions of linear systems of equations**: I did experimental measurements for different transconductance amplifier-based networks with various matrices and inputs to solve a linear system of equations on chip. This work was published in the ISCAS conference [170] and the TCAS-I journal [171].

- **Implementation of synapses with neurons**: I designed synaptic clefts through ramp generators to produce post-synaptic potentials for excitatory and inhibitory synapses. I built upon this neuron and synapse model to build spiking networks. This work was published in ISCAS [172].

- **RASP 3.1**: The design and fabrication of the next generation of the FPAAs in 130nm was done in collaboration with Sahil Shah and Sihwan Kim. Experimental measurements and testing has not been done.

- **Design and layout of FG test structures in 14nm**: The design and layout of the FG test cells was done in a 14nm FinFET process to characterise FG cells in a scaled down process as well as develop a standard cell library of blocks. Experimental measurements and testing has not been done.

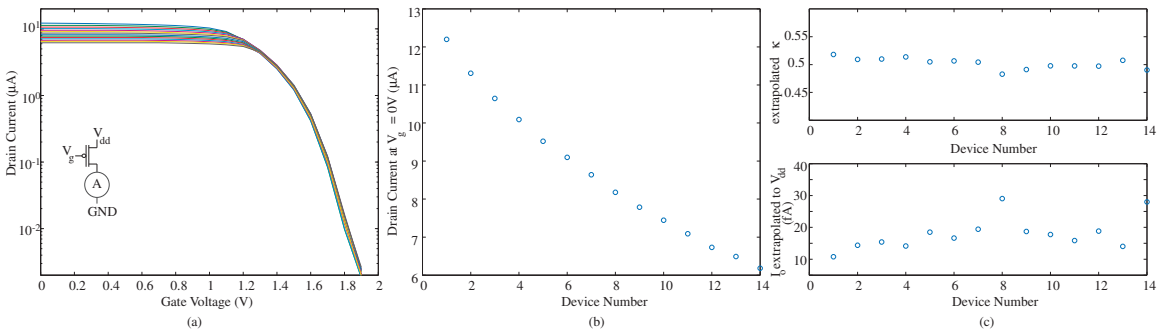


Figure 7.1: A current sweep experiment is performed for a pFET across the 14 CABs. To understand the variation across CABs, we show the characterization of this device sweeping the gate voltage. (a) The Circuit block for the experiment is shown. The gate voltage is swept and the drain voltage is at V_{dd} for the pFET in each CAB. The drain current is measured. (b) The drain current limit and it's variation is plotted corresponding to the current when gate voltage is 0V. (c) κ and I_o are extracted from the slope and the variation is plotted across CABs.

7.3 Issues to notice through experimental observations

A number of methods to compensate for variations has been discussed in prior chapters. Further experiments have been performed to analyse issues concerning variations as we build on our modules to form more complex systems. We can exploit programmability to compensate for these issues.

The resistive drop due to the routing across different CABs was measured to analyse the conductance of the switches as well as to ensure that the local supply voltage to each CAB is not shifting. A current sweep experiment is performed for a pFET across the 14 CABs. The variation of the drain current against the gate voltage is shown in Figure 7.1(a). A shift register is used to scan the drains to be measured. For the high current limit curve in Figure 7.1(b), the maximum current is decreasing with the position of the transistor on the CAB. This is because the conductance of the switches or pFETs are limiting the transistor curve, likely moving the effective drain voltage higher towards V_{dd}. The response is monotonic as expected with a constant increase in switches. Threshold voltage mismatch can be observed from the I_o extrapolated in Figure 7.1(c) as well as a small variation in κ

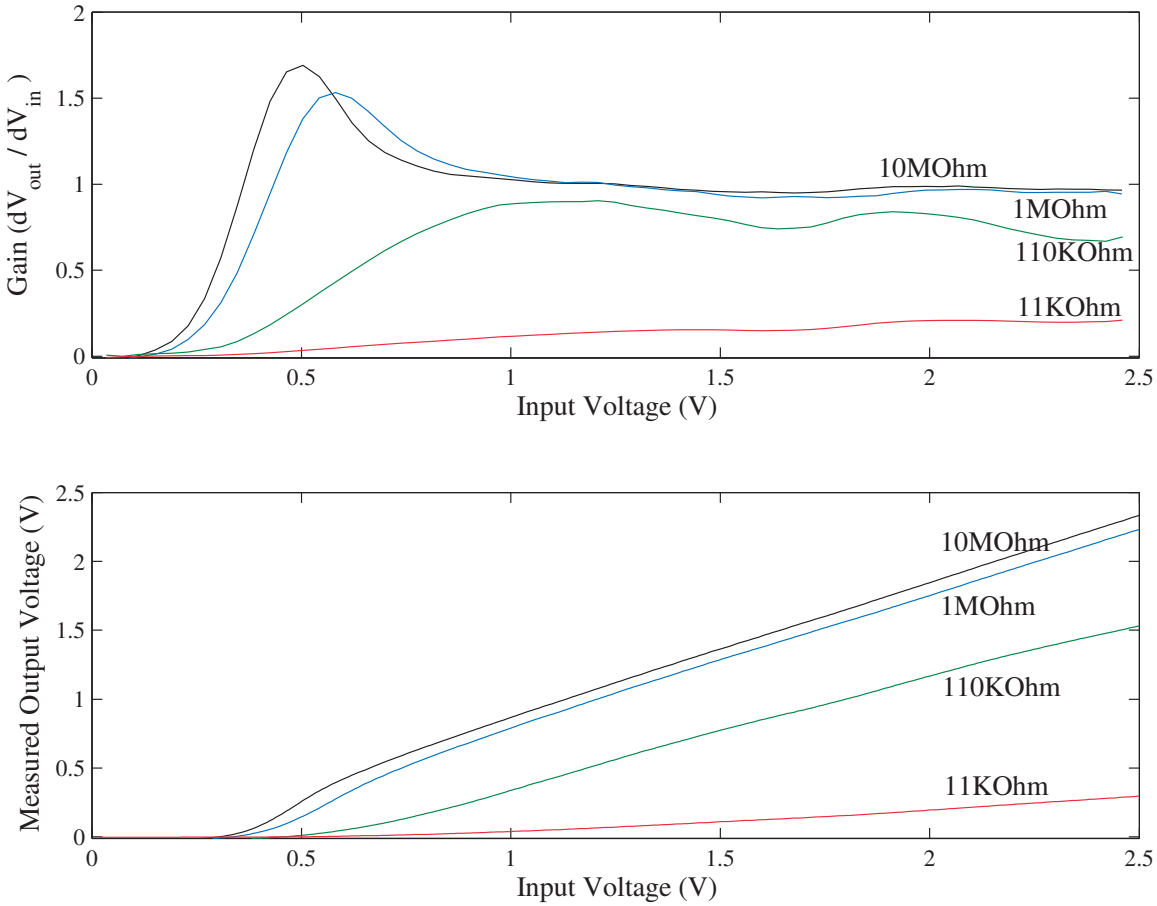


Figure 7.2: The conductance of switches is analysed and the transfer of function of output voltage against input voltage is plotted with different load resistances. The slope of the output voltage against the input voltage is plotted to analyse the gains and the resistance is observed as a function of the input voltage.

mismatch. From this experiment, the local V_{dd} seems to be roughly the same for all curves, thereby not changing drastically across the CABs.

Another significant experiment is performed to study the conductance of switches; output voltage is plotted against input voltage in Figure 7.2 for different load resistances. The resulting resistance has been measured for ten switches. The slope of the gains are fairly reasonable beyond 1V as expected. For input voltages around 500mV, the gain is greater than 1 in some cases for the set of passive elements of switches that are effectively making a voltage divider with the respective load resistances.

From the experiments, it is apparent that one must be cautious while taking measure-

ments near the power supply V_{dd} or ground. The choice of buffers or shift registers should also be considered when one observes offsets while measuring in different CABs. Hence, it remains a key requirement to tune the analog cells on the fabric with systematic characterization and calibration procedures. This will reduce the errors in analog computation as well as make sure that mismatch does not become a limiting factor. Further, these questions of mismatch bottlenecks become more important as we start scaling down for better performance metrics.

7.4 Scaling, Design and Layout of Floating-Gate Test Structures in 14nm

Previous approaches show FG functionality and characterization that is consistent from 2µm through 40nm CMOS (e.g. [165]). Scaling FG devices enables building dense, compact systems as well as offering higher frequency response and energy efficiency. It is important to test and characterize the FG devices at 14nm to study scalability as well as take steps towards developing a programmable standard cell library [173]. These standard cells will enable automation of the analog IC layout process. Programmable analog IC automation can drastically decrease time, cost, and design uncertainty similar to the impact of digital IC automation. Hence, this discussion shows a set of test structures and resulting tests that fully characterize FG devices for typical FG programming approaches (e.g. [44]) as well as lay the foundation for a standard cell library in 14nm.

The focus is developing working structures that give clear insight on developing dense 14nm FG structures, as well as enable characterization of FG electron tunneling and hot-electron injection [44]. We have used MOSFETs with the larger insulator devices, devices often used for external I/O interfaces. Each FG is a continuous poly Si with no contacts. One approach to achieving such a cell is having a single strip of vertical polysilicon going through each of the devices (Figure 7.3). All of the characterization nFETs and pFETs are the same size, where the constraints come from layout and not physics. Test structures are designed assuming pFET injection, with corresponding structures to test nFET injec-

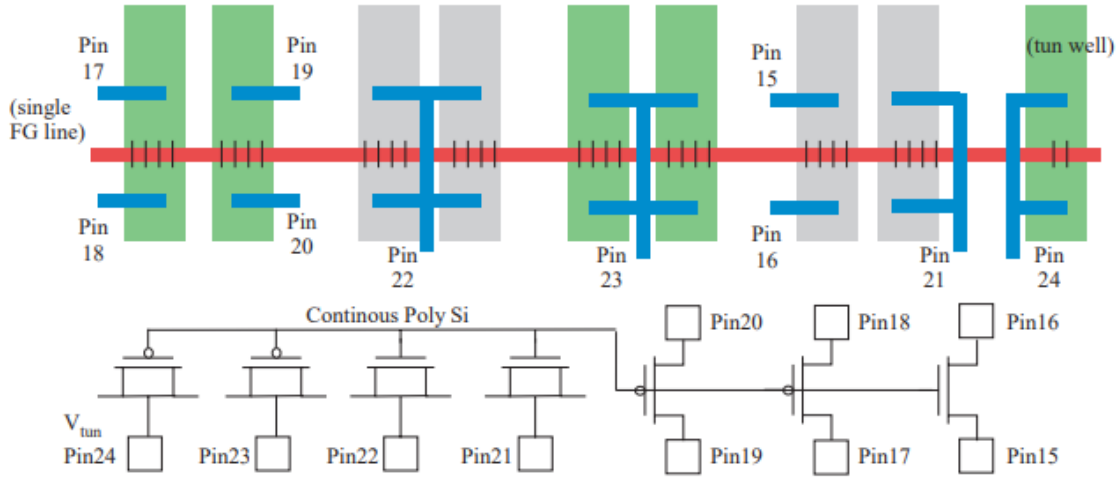


Figure 7.3: Visualizing 14nm layout of a FG pFET test device. Continuous FG is developed by continuous polysilicon or connecting layer. Input capacitor is designed to be significantly larger than the tunneling capacitor. Hence, the tunneling capacitor has a two fin thick insulator. The actual cell is rotated by 90 degrees so the polysillicon gate is a vertical bar.

tion. Further, multiple transistors are located on a FG node to measure the injection circuit concepts, as well as place nFET devices in places that could enable injection when testing for these options. Understanding and characterizing the capacitors to couple into the FG node fully describes the potential FG devices. In this 14nm process, MOS capacitors built from MOSFET transistors with n+ junction and an n-substrate and non-selected polysilicon contact layer to polysilicon are the two potential polysilicon capacitors. The first capacitor potentially has non-linear MOS capacitance effects where the second capacitor is a linear capacitor. The MOS capacitors, whether nFET or pFET based structures, are designed to be on separate strips that intersect the FG polysilicon line. The core test structure includes these different capacitance options (Figure 7.3), to test building dense structures.

The tunneling junction capacitor is a pFET-based MOS capacitor with the source, drain, and well terminals tied together. The spacing of this device would need additional spacing to ensure no breakdowns, due to the larger voltage used to induce electron tunneling. Since the tunneling capacitor is smaller than the input capacitor, it has a two finger thick insulator

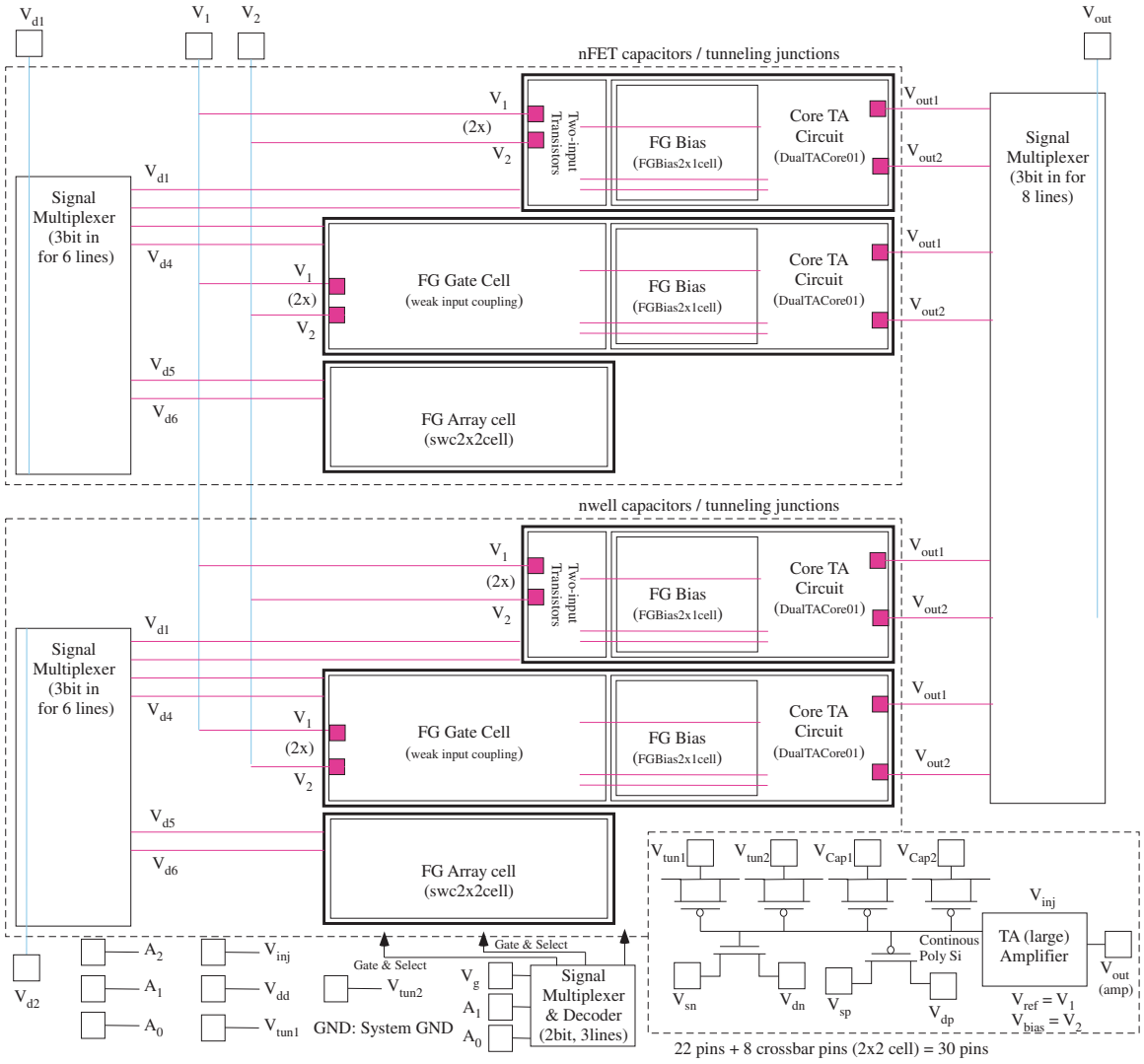


Figure 7.4: Top level structure for experimentally measuring the 14nm test cells. The test structures include instrumented pFET and nFET transistors for hot-electron injection, a single pFET device for impact-ionization characterization, and an instrumented test structure with a high-gain transistor amplifier to integrate FG currents. All transistors use thick insulator options to potentially enable long-term lifetimes. The top level has been designed to have 30 pins.

while the input capacitor has four or eight fingers.

The core structure includes an instrumented FG device, as well as a device with the FG also connected directly to the gate of an integrating amplifier structure (Figure 7.5). The high-gain transistor amplifier integrates the FG currents, enabling a direct measurement of FG currents in a static structure without requiring an ammeter. The vertical gate contin-

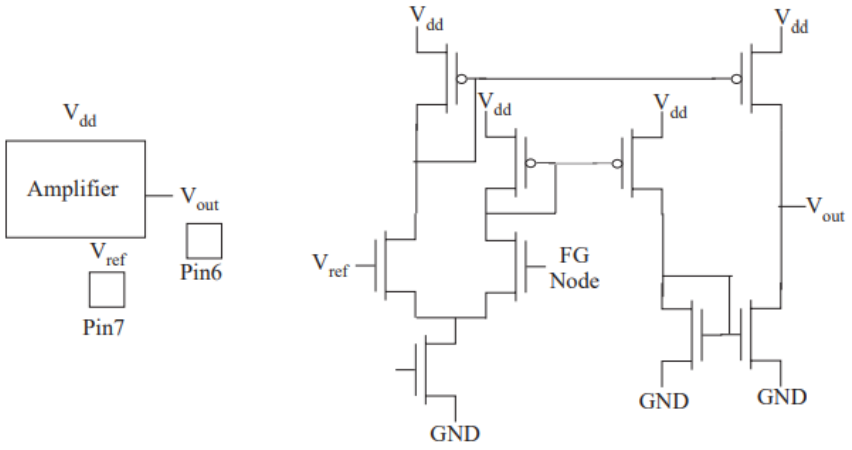


Figure 7.5: Design of the transconductance amplifier for the instrumented FG structure. The transconductance amplifier is designed with transistors with longer L to enable amplifier gain. For the integrating amplifier, an additional capacitor is connected between the FG node and the amplifier output node for a direct measurement of FG currents without requiring an ammeter.

ues through to the nFET of the one input differential pair. One would have an additional capacitor between the FG node and the amplifier output node that does not contact the polysilicon node. The amplifier structure fixes the FG node while measuring low currents through a current integrator structure. The instrumented amplifier is a 9-transistor differential transconductance amplifier (TA). The nFET differential pair has one input from the floating gate, and as a result, they have longer transistor lengths to ensure sufficient gain. The topology can be modified depending on the particular process constraints (e.g. threshold voltage mismatch issues). Each transistor tends to utilize large insulator devices as the characterization voltages are higher than standard CMOS values for the process. Further, the characterization cell has been designed with different capacitors built out of nFETs and pFETs for the tunneling junction. The FG characterization structure is implemented in a single 30pin slice that is embedded in a larger test structure (Figure 7.4). GND is the one pin shared with the outside structure. The FG characterization power supply, V_{inj}, is a separate V_{dd} pin. Often, this V_{dd} voltage must be higher than typical CMOS V_{dd} to be enough for channel current, impact-ionization current, and hot-electron injection current characterization, often pushing devices near or beyond breakdown levels of other CMOS

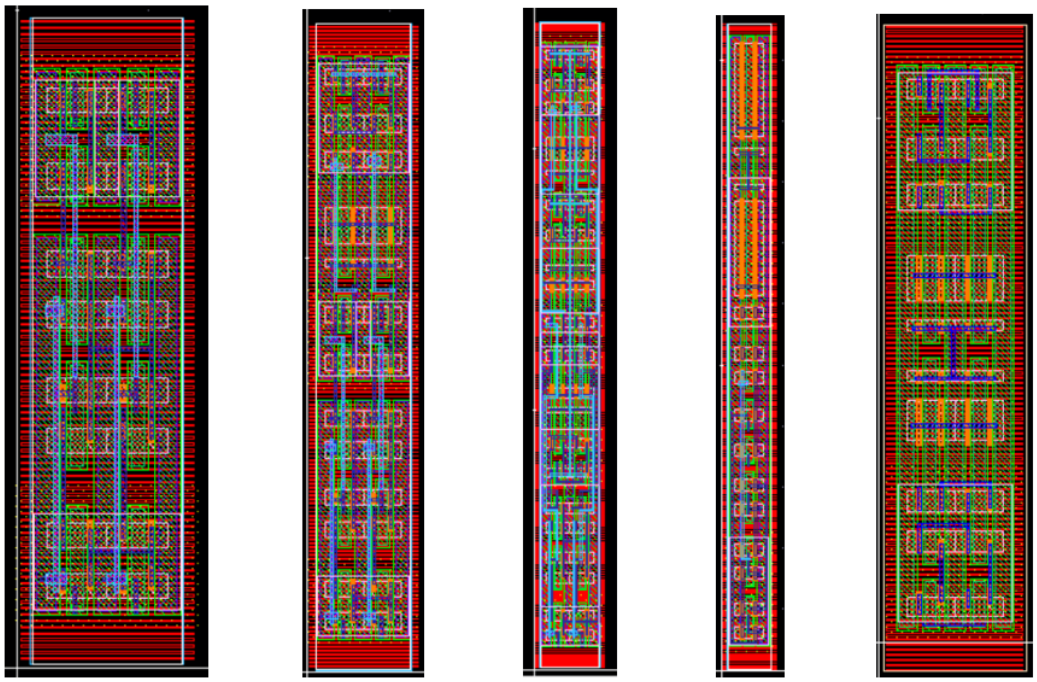


Figure 7.6: The layouts of all the primary FG cells are shown. (a) A basic 2x1 9-transistor TA with signal bias and non-FG inputs is shown. (b) A non-FG 2x1 TA block with no FG inputs and FG bias at the current source is shown. (c) A 2x1 TA block with low coupling and high-linearity FG inputs is shown. (d) The core FG characterization structure with different capacitors and the instrumentation amplifier is shown, with pins terminated out for characterization. (e) A 2x2 FG switch cell has been designed to analyse the crossbar structure. The FG elements have some I/O pins used to directly characterize sample crossbar elements externally.

structures. The high-gain transistor amplifier is the only circuit to use V_{dd} other than the protection diodes on the I/O pins. The layout structure conforms to the regular polysilicon grid even when including the amplifier components (Figure 7.5). We have a tunneling junction pin that requires significantly higher voltages and different protection structures. Two tunneling junction pins are reserved for the nFET-based MOS capacitors and the nwell based capacitors. All the tunneling lines are shared between the characterization structure and other devices. The substrate and well contacts are provided in the area outside of the cells. The signal multiplexers and decoders have been designed to provide the gate select signals to the FG cells as well as read the outputs from the TAs and crossbar cells in the test structure.

Table 7.1: The different cells designed in 14nm CMOS and their cell sizes.

Cell type	Cell name	Cell Variations	Cell Pitch (μm)	Cell Size (μm)
Transconductance Amplifiers (TA)	2x1NonFG_TA	2TA, FG bias, no FG inputs, MOScaps	7.247	1.161
	2x1FG_TA	2TA, FG bias, no FG inputs, MOScaps	11.807	1.161
	2x1NonFG_TA_nwell	2TA, FG bias, no FG inputs, nwell-caps	7.247	1.161
	2x1FG_TA_nwell	2TA, FG bias, no FG inputs, nwell-caps	11.807	1.161
	2TA_vbias	2TA, signal bias, no FG inputs	4.372	1.161
FG switch	fgswc2x2_MOScap	2x2 cell, MOScaps	5.622	1.12
	fgswc2x2_nwell	2x2 cell, nwell-caps	5.622	1.12
FG Characterization	Fgchar	Characterization, different caps	10.195	0.717

The schematics and layouts have been designed in the non-planar bulk 14nm FinFET process (Figure 7.6). All the FG cells have been designed with different variations on the capacitors. There are primarily two types, consisting of MOS capacitors or nwell capacitors. A basic 2x1 9-transistor TA with signal bias and non-FG inputs is shown in Figure 7.6(a). Figure 7.6(b) shows a non-FG 2x1 TA block with no FG inputs while the 2x1 TA block with low coupling and high-linearity FG inputs is shown in Figure 7.6(c), where both have FG biases at the current source to the TA. Figure 7.6(d) shows the core FG characterization structure with different capacitors and the instrumentation amplifier, with pins terminated out for characterization. A 2x2 FG switch cell has been designed to analyse the crossbar structure (Figure 7.6(e)). The FG elements have some I/O pins used to directly characterize sample crossbar elements externally.

The objective is to build a standard cell library of core FG cells in the 14nm CMOS FinFET process; their cell pitches and sizes are shown in Table 7.1. An initial design with

the standard layout rules has been done for a first set of measurements. Tight SRAM rules would significantly decrease the pitch and width dimensions (2×) for the 14nm cells, and an SOI based process would eliminate the well spacing between devices.

A resistive structure made from polysilicon is used to protect a tunneling junction, especially during packaging, and to minimize the resulting device current for instrumentation. The source and drain terminals are protected through reverse-bias diode protection to V_{dd} and GND, which would include MOS capacitors (other than the tunneling capacitor) and other capacitor structures. We have also laid out one pFET with its well terminal brought out to a pin in this group, to directly measure the impact ionization current from the device and correlate with hot electron injection properties. On the top level, the chip is integrated with the IO pads in a ring-like structure, and all the other pins from the chip are connected to ESD structures on the external IO pads, which would then be either wire-bonded or flip chipped onto a PCB.

The design of FG cells in 14nm, thus, initiates efforts to experimentally test and characterize the FG structures at scaled down IC technology nodes, to understand device properties, and provide opportunities to build dense arrays of configurable FG based systems. They act as a fundamental group of core blocks for a standard cell analog library in 14nm, to enable IC automation at lower technology nodes as well.

Looking forward, the work presented in this dissertation launches one into a space, offering diverse opportunities for energy-efficient computing. The analog and neuromorphic systems facilitate the creation of the next generation of systems that will enable low-power embedded computing.

REFERENCES

[1] C. A. Mead, "Neuromorphic electronic systems," *Proceedings of the IEEE*, vol. 78, pp. 1629–1636, 1990.

[2] S. George *et al.*, "A programmable and configurable mixed-mode FPAA soc," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 24, no. 6, pp. 2253–2261, Jun. 2016.

[3] G. D. Michell and R. K. Gupta, "Hardware/software co-design," *Proceedings of the IEEE*, vol. 85, no. 3, pp. 349–365, Mar. 1997.

[4] D. Kahng and S. M. Sze, "A floating gate and its application to memory devices," *The Bell System Technical Journal*, vol. 46, no. 6, pp. 1288–1295, 1967.

[5] P. E. Hasler, "Floating-gate devices, circuits, and systems, invited," in *Proceedings of the 5th IEEE International Workshop on System-on-Chip for Real-Time Applications (IWSOC 2005), 20-24 July 2004, Banff, Alberta, Canada*, 2005, pp. 482–487.

[6] P. Hasler, C. Diorio, B. A. Minch, and C. Mead, "Single transistor learning synapses," *Advances in Neural Information Processing Systems*, pp. 817–824, 1994.

[7] E. K. F. Lee and P. G. Gulak, "A cmos field-programmable analog array," *IEEE Journal of Solid-State Circuits*, vol. 26, no. 12, pp. 1860–1867, Dec. 1991.

[8] A. Basu *et al.*, "Rasp 2.8: A new generation of floating-gate based field programmable analog array," in *2008 IEEE Custom Integrated Circuits Conference*, Sep. 2008, pp. 213–216.

[9] C. Schlottmann, D. N. Abramson, and P. E. Hasler, "A MITE-based translinear FPAA," *IEEE Trans. VLSI Syst.*, vol. 20, no. 1, pp. 1–9, 2012.

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

[11] D. W. Graham, E. Farquhar, B. Degnan, C. Gordon, and P. Hasler, "Indirect programming of floating-gate transistors," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 54, no. 5, pp. 951–963, May 2007.

[12] M. Collins, J. Hasler, and S. George, "An open-source tool set enabling analog-digital-software co-design," *Journal of Low Power Electronics and Applications*, vol. 6, no. 1, p. 3, 2016.

[13] J. Hasler and A. Natarajan, "An open-source toolset for FPAA design," in *Article No. 18, Third Workshop on Open-Source EDA Technology (WOSET), 2020.*.

[14] S. Kim, J. Hasler, and S. George, "Integrated floating-gate programming environment for system-level ics," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. PP, no. 99, pp. 1–9, 2016.

[15] A. Natarajan and J. Hasler, "Modeling, simulation and implementation of circuit elements in an open-source tool set on the FPAA," *Analog Integrated Circuits and Signal Processing*, vol. 91, no. 1, pp. 119–130, 2017.

[16] J. Hasler, A. Natarajan, and S. Kim, "Enabling energy-efficient physical computing through analog abstraction and ip reuse," *Journal of Low Power Electronics and Applications*, vol. 8, no. 4, p. 47, 2018.

[17] A. L. Hodgkin and A. F. Huxley, "A quantitative description of membrane current and its application to conduction and excitation in nerve," *The Journal of Physiology*, vol. 117, no. 4, pp. 500–544, 1952.

[18] G. Indiveri *et al.*, "Neuromorphic silicon neuron circuits," *Frontiers in Neuroscience*, vol. 5, p. 73, 2011.

[19] S. Saighi, Y. Bornat, J. Tomas, G. L. Masson, and S. Renaud, "A library of analog operators based on the hodgkin-huxley formalism for the design of tunable, real-time, silicon neurons," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 5, no. 1, pp. 3–19, Feb. 2011.

[20] E. Farquhar and P. Hasler, "A bio-physically inspired silicon neuron," in *2004 IEEE International Symposium on Circuits and Systems (IEEE Cat. No.04CH37512)*, vol. 1, May 2004, I-309-I–312 Vol.1.

[21] N. Brunel and M. C. Van Rossum, "Lapicque's 1907 paper: From frogs to integrate-and-fire," *Biological cybernetics*, vol. 97, no. 5-6, pp. 337–339, 2007.

[22] G. Indiveri, "A low-power adaptive integrate-and-fire neuron circuit," in *Circuits and Systems, 2003. ISCAS '03. Proceedings of the 2003 International Symposium on*, vol. 4, May 2003, IV-820-IV–823 vol.4.

[23] M. Davies *et al.*, "Loihi: A neuromorphic manycore processor with on-chip learning," *IEEE Micro*, vol. 38, no. 1, pp. 82–99, Jan. 2018.

[24] E. M. Izhikevich, "Which model to use for cortical spiking neurons?" *IEEE Transactions on Neural Networks*, vol. 15, no. 5, pp. 1063–1070, Sep. 2004.

[25] A. A. Zhilenkov and M. V. Kotlyarevskaya, "Synthesis of model of hardware realization of izhikevich model of biological neuron on the basis of fpga," in *2018 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus)*, Jan. 2018, pp. 1040–1043.

[26] M. Mahowald and R. Douglas, "A silicon neuron," *Nature*, vol. 354, no. 6354, pp. 515–518, 1991.

[27] K. M. Hynna and K. Boahen, "Neuronal ion-channel dynamics in silicon," in *2006 IEEE International Symposium on Circuits and Systems*, May 2006.

[28] M. F. Simoni, G. S. Cymbalyuk, M. E. Sorensen, R. L. Calabrese, and S. P. DeWeerth, "A multiconductance silicon neuron with biologically matched dynamics," *IEEE Transactions on Biomedical Engineering*, vol. 51, no. 2, pp. 342–354, Feb. 2004.

[29] F. Grassia, L. Buhry, T. Levi, J. Tomas, A. Destexhe, and S. Saighi, "Tunable neuromimetic integrated system for emulating cortical neuron models," *Frontiers in Neuroscience*, vol. 5, p. 134, 2011.

[30] T. Yu, T. J. Sejnowski, and G. Cauwenberghs, "Biophysical neural spiking, bursting, and excitability dynamics in reconfigurable analog vlsi," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 5, no. 5, pp. 420–429, Oct. 2011.

[31] C. Mead, *Analog VLSI and Neural Systems*. Addison-Wesley Longman Publishing Co., Inc., 1989.

[32] E. Farquhar and P. Hasler, "A bio-physically inspired silicon neuron," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 52, no. 3, pp. 477–488, 2005.

[33] J. Hasler, S. Shah, S. Kim, I. K. Lal, and M. Collins, "Remote system setup using large-scale field programmable analog arrays (FPAA) to enabling wide accessibility of configurable devices," *Journal of Low Power Electronics and Applications*, vol. 6, no. 3, p. 14, 2016.

[34] P. Dayan and L. F. Abbott, *Theoretical Neuroscience: Computational and Mathematical Modeling of Neural Systems*. The MIT Press, 2005, ISBN: 0262541858.

[35] J. H. Wijekoon and P. Dudek, "Compact silicon neuron circuit with spiking and bursting behaviour," *Neural Networks*, vol. 21, no. 2, pp. 524–534, 2008.

[36] B. Hille *et al.*, *Ion channels of excitable membranes*. Sinauer Sunderland, MA, 2001, vol. 507.

[37] S. Nease, S. George, P. Hasler, S. Koziol, and S. Brink, "Modeling and implementation of voltage-mode cmos dendrites on a reconfigurable analog platform," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 6, no. 1, pp. 76–84, Feb. 2012.

[38] S. Sheik, E. Chicca, and G. Indiveri, "Exploiting device mismatch in neuromorphic VLSI systems to implement axonal delays," in *International Joint Conference on Neural Networks, (IJCNN) 2012*, IEEE, 2012, pp. 1940–1945.

[39] S. Sheik, M. Coath, G. Indiveri, S. Denham, T. Wennekers, and E. Chicca, "Emergent auditory feature tuning in a real-time neuromorphic VLSI system," *Frontiers in Neuroscience*, vol. 6, no. 17, 2012.

[40] O. Richter, R. F. Reinhart, S. Nease, J. Steil, and E. Chicca, "Device mismatch in a neuromorphic system implements random features for regression," in *2015 IEEE Biomedical Circuits and Systems Conference (BioCAS)*, Oct. 2015, pp. 1–4.

[41] R. George and G. Indiveri, "Tunable device-mismatch effects for stochastic computation in analog/digital neuromorphic computing architectures," in *2016 IEEE International Conference on Electronics, Circuits and Systems (ICECS)*, Dec. 2016, pp. 77–80.

[42] S. Borkar, T. Karnik, S. Narendra, J. Tschanz, A. Keshavarzi, and V. De, "Parameter variations and impact on circuits and microarchitecture," in *Proceedings of the 40th annual Design Automation Conference*, ACM, 2003, pp. 338–342.

[43] S. Brink *et al.*, "A learning-enabled neuron array ic based upon transistor channel models of biological phenomena," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 7, no. 1, pp. 71–81, Feb. 2013.

[44] S. Kim, S. Shah, and J. Hasler, "Calibration of floating-gate SoC FPAA system," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 2017.

[45] E. Neftci and G. Indiveri, "A device mismatch compensation method for vlsi neural networks," in *Biomedical Circuits and Systems Conference (BioCAS), 2010 IEEE*, IEEE, 2010, pp. 262–265.

[46] E. Neftci, E. Chicca, G. Indiveri, and R. Douglas, "A systematic method for configuring vlsi networks of spiking neurons," *Neural Computation*, vol. 23, no. 10, pp. 2457–2497, 2011.

[47] L. Buhry, M. Pace, and S. Saıghi, "Global parameter estimation of an hodgkin-huxley formalism using membrane voltage recordings: Application to neuro-mimetic analog integrated circuits," *Neurocomput.*, vol. 81, pp. 75–85, Apr. 2012.

[48] L. Buhry, F. Grassia, A. Giremus, E. Grivel, S. Renaud, and S. Saighi, "Automated parameter estimation of the hodgkin-huxley model using the differential evolution algorithm: Application to neuromimetic analog integrated circuits," *Neural Computation*, vol. 23, no. 10, pp. 2599–2625, Oct. 2011.

[49] H. Chen, S. Saighi, L. Buhry, and S. Renaud, "Real-time simulation of biologically realistic stochastic neurons in vlsi," *IEEE Transactions on Neural Networks*, vol. 21, no. 9, pp. 1511–1517, Sep. 2010.

[50] T. J. Hamilton, S. Afshar, A. van Schaik, and J. Tapson, "Stochastic electronics: A neuro-inspired design paradigm for integrated circuits," *Proceedings of the IEEE*, vol. 102, no. 5, pp. 843–859, May 2014.

[51] K. Yue and A. C. Parker, "Noisy neuromorphic neurons with rpg on-chip noise source," in *2017 International Joint Conference on Neural Networks (IJCNN)*, May 2017, pp. 1225–1229.

[52] W. Maass, "Noise as a resource for computation and learning in networks of spiking neurons," *Proceedings of the IEEE*, vol. 102, no. 5, pp. 860–880, May 2014.

[53] S. Renaud *et al.*, "Pax: A mixed hardware/software simulation platform for spiking neural networks," *Neural Networks*, vol. 23, no. 7, pp. 905–916, 2010.

[54] T. Levi, N. Lewis, J. Tomas, and S. Renaud, "Application of IP-based analog platforms in the design of neuromimetic integrated circuits," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 31, no. 11, pp. 1629–1641, Nov. 2012.

[55] S. Yaghini Bonabi, H. Asgharian, S. Safari, and M. Nili Ahmadabadi, "Fpga implementation of a biological neural network based on the hodgkin-huxley neuron model," *Frontiers in Neuroscience*, vol. 8, p. 379, 2014.

[56] T. Levi, F. Khoyratee, S. Saïghi, and Y. Ikeuchi, "Digital implementation of hodgkin—huxley neuron model for neurological diseases studies," *Artif. Life Robot.*, vol. 23, no. 1, pp. 10–14, Mar. 2018.

[57] M. Ambroise, T. Levi, S. Joucla, B. Yvert, and S. Saighi, "Real-time biomimetic central pattern generators in an fpga for hybrid experiments," *Frontiers in Neuroscience*, vol. 7, p. 215, 2013.

[58] M. Lu, J.-L. Wang, J. Wen, and X.-W. Dong, "Implementation of hodgkin-huxley neuron model in fpgas," in *2016 Asia-Pacific International Symposium on Electromagnetic Compatibility (APEMC)*, vol. 01, May 2016, pp. 1115–1117.

[59] J. Hasler and H. B. Marr, "Finding a roadmap to achieve large neuromorphic hardware systems," *Frontiers in Neuroscience*, vol. 7, no. 118, 2013.

[60] J. Hasler, "Starting framework for analog numerical analysis for energy-efficient computing," *Journal of Low Power Electronics and Applications*, vol. 7, no. 3, p. 17, 2017.

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

[62] D. P. Foty, *MOSFET Modeling with SPICE: Principles and Practice*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1997, ISBN: 0-13-227935-5.

[63] Y. Cheng and C. Hu, *MOSFET Modeling & BSIM3 User's Guide*. Springer US, 2002.

[64] Y. P. Tsividis and K. Suyama, "MOSFET modeling for analog circuit cad: Problems and prospects," *IEEE Journal of Solid-State Circuits*, vol. 29, no. 3, pp. 210–216, Mar. 1994.

[65] I. Guerra-Gómez, T. Mcconaghy, and E. Tlelo-Cuautle, "Operating-point driven formulation for analog computer-aided design," *Analog Integr. Circuits Signal Process.*, vol. 74, no. 2, pp. 345–353, Feb. 2013.

[66] C. Schlottmann, C. Petre, and P. E. Hasler, "A high-level simulink-based tool for FPAA configuration," *IEEE Trans. VLSI Syst.*, vol. 20, no. 1, pp. 10–18, 2012.

[67] S. L. Campbell, J.-P. Chancelier, and R. Nikoukhah, *Modeling and Simulation in Scilab/Scicos*. Springer-Verlag New York, 2006.

[68] K. C. A. Lam and M. Zwolinski, "Circuit simulation using state space equations," in *Ph.D. Research in Microelectronics and Electronics (PRIME), 2013 9th Conference on*, Jun. 2013, pp. 177–180.

[69] M. Tiller, *Introduction to Physical Modeling with Modelica*. Springer US, 2001.

[70] C. C. Enz, F. Krummenacher, and E. A. Vittoz, "An analytical MOS transistor model valid in all regions of operation and dedicated to low-voltage and low-current applications," *Analog Integrated Circuits and Signal Processing*, vol. 8, no. 1, pp. 83–114, 1995.

[71] A. Low and P. Hasler, "Cadence-based simulation of floating-gate circuits using the EKV model," in *Circuits and Systems, 1999. 42nd Midwest Symposium on*, vol. 1, 1999, 141–144 vol. 1.

[72] Y. P. Tsividis, *Operation and Modelling of the MOS Transistor*. McGraw-Hill, 1987.

[73] K. M. Odame and P. E. Hasler, "Theory and design of OTA-C oscillators with native amplitude limiting," *IEEE Trans. on Circuits and Systems*, vol. 56-I, no. 1, pp. 40–50, 2009.

[74] R. Chawla, F. Adil, G. J. Serrano, and P. E. Hasler, "Programmable g_m- C filters using floating-gate operational transconductance amplifiers," *IEEE Trans. on Circuits and Systems*, vol. 54-I, no. 3, pp. 481–491, 2007.

[75] D. W. Graham, P. E. Hasler, R. Chawla, and P. D. Smith, "A low-power programmable bandpass filter section for higher order filter applications," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 54, no. 6, pp. 1165–1176, Jun. 2007.

[76] S. Ramakrishnan, A. Basu, L. K. Chiu, J. Hasler, D. Anderson, and S. Brink, "Speech processing on a reconfigurable analog platform," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 22, no. 2, pp. 430–433, Feb. 2014.

[77] C. Teague *et al.*, "Novel approaches to measure acoustic emissions as biomarkers for joint health assessment," in *Wearable and Implantable Body Sensor Networks (BSN), 2015 IEEE 12th International Conference on*, Jun. 2015, pp. 1–6.

[78] M. Etemadi, O. T. Inan, J. A. Heller, S. Hersek, L. Klein, and S. Roy, "A wearable patch to enable long-term monitoring of environmental, activity and hemodynamics variables," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 10, no. 2, pp. 280–288, Apr. 2016.

[79] S. Shah, J. Smith, J. Stowell, and J. B. Christen, "Biosensing platform on a flexible substrate," *Sensors and Actuators B: Chemical*, vol. 210, pp. 197–203, 2015.

[80] R. Pierret, *Semiconductor Device Fundamentals*. Addison-Wesley, 1996, ISBN: 9780131784598.

[81] J. Hasler, S. Kim, and F. Adil, "Scaling floating-gate devices predicting behavior for programmable and configurable circuits and systems," *J. Low Power Electron. Appl.*, 2016.

[82] B. Minch, *Ekv mos transistor model summary (http://madvlsi.olin.edu/circuits/handouts)*.

[83] B. P. Degnan, "Temperature robust programmable subthreshold circuits through a balanced force approach," Ph.D. dissertation, Georgia Institute of Technology, 2013.

[84] S. Shah, H. Toreyin, J. Hasler, and A. Natarajan, "Temperature sensitivity and compensation on a reconfigurable platform," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 26, no. 3, pp. 604–607, 2018.

[85] *Ngspice web site, "http://ngspice.sourceforge.net/"*.

[86] H. V. Paolo Nenzi, *Ngspice users manual*.

[87] *Ekv website, http://ekv.epfl.ch/*.

[88] A. Natarajan and J. Hasler, "Hodgkin–huxley neuron and fpaa dynamics," *IEEE transactions on biomedical circuits and systems*, vol. 12, no. 4, pp. 918–926, 2018.

[89] S. Brink *et al.*, "A learning-enabled neuron array ic based upon transistor channel models of biological phenomena," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 7, no. 1, pp. 71–81, Feb. 2013.

[90] G. Indiveri, E. Chicca, and R. Douglas, "A vlsi array of low-power spiking neurons and bistable synapses with spike-timing dependent plasticity," *IEEE Transactions on Neural Networks*, vol. 17, no. 1, pp. 211–221, Jan. 2006.

[91] J. Schemmel, A. Grubl, K. Meier, and E. Mueller, "Implementing synaptic plasticity in a vlsi spiking neural network model," in *The 2006 IEEE International Joint Conference on Neural Network Proceedings*, Jul. 2006, pp. 1–6.

[92] C. Koch, *Biophysics of Computation: Information Processing in Single Neurons (Computational Neuroscience Series)*. New York, NY, USA: Oxford University Press, Inc., 2004, ISBN: 0195181999.

[93] D. Purves, G. Augustine, and D. Fitzpatrick, *Neuroscience. 2nd edition. ; 2001.* Sunderland (MA): Sinauer Associates, 2001.

[94] S. George *et al.*, "A programmable and configurable mixed-mode fpaa soc," *IEEE Transactions on VLSI*, vol. 24, no. 6, pp. 2253–2261, 2016.

[95] J. Hasler and S. Shah, "Soc fpaa hardware implementation of a vmm+wta embedded learning classifier," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 2018.

[96] O. Rhodes *et al.*, "Spynnaker: A software package for running pynn simulations on spinnaker," *Frontiers in Neuroscience*, vol. 12, p. 816, 2018.

[97] A. Basu, S. Ramakrishnan, C. Petre, S. Koziol, S. Brink, and P. E. Hasler, "Neural dynamics in reconfigurable silicon," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 4, no. 5, pp. 311–319, Oct. 2010.

[98] M. Oster, R. Douglas, and S.-C. Liu, "Computation with spikes in a winner-take-all network," *Neural computation*, vol. 21, no. 9, pp. 2437–2465, 2009.

[99] S. Koziol, S. Brink, and J. Hasler, "A neuromorphic approach to path planning using a reconfigurable neuron array ic," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 22, no. 12, pp. 2724–2737, Dec. 2014.

[100] S. Sheik, E. Chicca, and G. Indiveri, "Exploiting device mismatch in neuromorphic vlsi systems to implement axonal delays," in *The 2012 International Joint Conference on Neural Networks (IJCNN)*, Jun. 2012, pp. 1–6.

[101] S. Still and M. W. Tilden, "Controller for a four-legged walking machine," 1998.

[102] A. Natarajan and J. Hasler, "Dynamics of hodgkin huxley neuron across chips implemented on a reconfigurable platform," in *2018 IEEE International Symposium on Circuits and Systems (ISCAS)*, May 2018, pp. 1–5.

[103] P. Allen and D. R. Holberg, *CMOS Analog Circuit Design*, 2nd, Ed. Oxford University Press, 2002.

[104] R. Rubino, P. S. Crovetti, and F. Musolino, "Fpga-based relaxation d/a converters with parasitics-induced error suppression and digital self-calibration," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 68, no. 6, pp. 2494–2507, 2021.

[105] S. Shah and J. Hasler, "Tuning of multiple parameters with a bist system," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 64, no. 7, pp. 1772–1780, 2017.

[106] C. S. Thakur, R. Wang, T. J. Hamilton, R. Etienne-Cummings, J. Tapson, and A. van Schaik, "An analogue neuromorphic co-processor that utilizes device mismatch for learning applications," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 65, no. 4, pp. 1174–1184, 2018.

[107] S. Sheik, E. Chicca, and G. Indiveri, "Exploiting device mismatch in neuromorphic vlsi systems to implement axonal delays," in *The 2012 International Joint Conference on Neural Networks (IJCNN)*, 2012, pp. 1–6.

[108] E. Kauderer-Abrams, A. Gilbert, A. Voelker, B. Benjamin, T. C. Stewart, and K. Boahen, "A population-level approach to temperature robustness in neuromorphic systems," in *2017 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2017, pp. 1–4.

[109] E. Neftci and G. Indiveri, "A device mismatch compensation method for vlsi neural networks," in *2010 Biomedical Circuits and Systems Conference (BioCAS)*, 2010, pp. 262–265.

[110] M. Paliy, S. Strangio, P. Ruiu, T. Rizzo, and G. Iannaccone, "Analog vector-matrix multiplier based on programmable current mirrors for neural network integrated circuits," *IEEE Access*, vol. 8, pp. 203 525–203 537, 2020.

[111] M. Paliy, T. Rizzo, P. Ruiu, S. Strangio, and G. Iannaccone, "Single-poly floating-gate memory cell options for analog neural networks," *Solid-State Electronics*, p. 108 062, 2021.

[112] P. Hasler and L. Akers, "Circuit implementation of trainable neural networks employing both supervised and unsupervised techniques," in *[Proceedings] 1992 IEEE International Symposium on Circuits and Systems*, vol. 3, 1992, 1565–1568 vol.3.

[113] M. Kucic, A. Low, P. Hasler, and J. Neff, "A programmable continuous-time floating-gate fourier processor," *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 48, no. 1, pp. 90–99, Jan. 2001.

[114] H. Töreyin and P. T. Bhatti, "A low-power asic signal processor for a vestibular prosthesis," *IEEE transactions on biomedical circuits and systems*, vol. 10, no. 3, pp. 768–778, 2016.

[115] J. Binas, D. Neil, G. Indiveri, S.-C. Liu, and M. Pfeiffer, "Precise deep neural network computation on imprecise low-power analog hardware," *arXiv preprint arXiv:1606.07786*, 2016.

[116] M. Bavandpour, M. R. Mahmoodi, and D. B. Strukov, "Energy-efficient time-domain vector-by-matrix multiplier for neurocomputing and beyond," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 66, no. 9, pp. 1512–1516, Sep. 2019.

[117] C. S. Thakur *et al.*, "Large-scale neuromorphic spiking array processors: A quest to mimic the brain," *Frontiers in Neuroscience*, vol. 12, p. 891, 2018.

[118] R. Chawla, A. Bandyopadhyay, V. Srinivasan, and P. Hasler, "A 531 nw/mhz, 128/spl times/32 current-mode programmable analog vector-matrix multiplier with over two decades of linearity," in *Proceedings of the IEEE 2004 Custom Integrated Circuits Conference (IEEE Cat. No.04CH37571)*, 2004, pp. 651–654.

[119] C. R. Schlottmann and P. E. Hasler, "A highly dense, low power, programmable analog vector-matrix multiplier: The fpaa implementation," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 1, no. 3, pp. 403–411, Sep. 2011.

[120] R. Genov and G. Cauwenberghs, "Charge-mode parallel architecture for vector-matrix multiplication," *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 48, no. 10, pp. 930–936, Oct. 2001.

[121] M. R. Mahmoodi and D. Strukov, "An ultra-low energy internally analog, externally digital vector-matrix multiplier based on nor flash memory technology," in *2018 55th ACM/ESDA/IEEE Design Automation Conference (DAC)*, Jun. 2018, pp. 1–6.

[122] S. Sahay, M. Bavandpour, M. R. Mahmoodi, and D. Strukov, "Energy-efficient moderate precision time-domain mixed-signal vector-by-matrix multiplier exploiting 1t-1r arrays," *IEEE Journal on Exploratory Solid-State Computational Devices and Circuits*, vol. 6, no. 1, pp. 18–26, 2020.

[123] P. Hasler and J. Dugger, "An analog floating-gate node for supervised learning," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 52, no. 5, pp. 834–845, 2005.

[124] S. Ramakrishnan and J. Hasler, "Vector-matrix multiply and winner-take-all as an analog classifier," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 22, no. 2, pp. 353–361, 2014.

[125] B. A. Minch, C. Diorio, P. Hasler, and C. A. Mead, "Translinear circuits using subthreshold floating-gate mos transistors," *Analog Integrated Circuits and Signal Processing*, vol. 9, no. 2, pp. 167–179, 1996.

[126] R. J. D'Angelo and S. R. Sonkusale, "A time-mode translinear principle for nonlinear analog computation," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 62, no. 9, pp. 2187–2195, 2015.

[127] D. W. Graham, E. Farquhar, B. Degnan, C. Gordon, and P. Hasler, "Indirect programming of floating-gate transistors," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 54, no. 5, pp. 951–963, May 2007.

[128] A. Basu *et al.*, "Rasp 2.8: A new generation of floating-gate based field programmable analog array," in *2008 IEEE Custom Integrated Circuits Conference*, Sep. 2008, pp. 213–216.

[129] C. M. Twigg and P. Hasler, "A large-scale reconfigurable analog signal processor (rasp) ic," in *IEEE Custom Integrated Circuits Conference 2006*, Sep. 2006, pp. 5–8.

[130] I. B. Cioc, I. Lita, D. A. Visan, and I. Bostan, "Fpaa implementation of signal processing circuits for radiation sensors," in *2009 32nd International Spring Seminar on Electronics Technology*, 2009, pp. 1–4.

[131] D. G. Moreno, A. A. D. Barrio, and G. B. Juan, "Simulating and deploying analog arithmetic circuits on fpaas," in *Proceedings of the 2020 Summer Simulation Conference*, ser. SummerSim '20, Virtual Event, Spain: Society for Computer Simulation International, 2020, ISBN: 9781713814290.

[132] S. Shah, H. Toreyin, J. Hasler, and A. Natarajan, "Models and techniques for temperature robust systems on a reconfigurable platform," *Journal of Low Power Electronics and Applications*, vol. 7, no. 3, p. 21, 2017.

[133] B. Ahuja, H. Vu, C. Laber, and W. Owen, "A very high precision 500-na cmos floating-gate analog voltage reference," *IEEE Journal of Solid-State Circuits*, vol. 40, no. 12, pp. 2364–2372, 2005.

[134] J. Hasler, "Starting framework for analog numerical analysis for energy efficient computing," *Journal of Low Power Electronics Applications*, vol. 7, no. 17, pp. 1–22, Jun. 2017.

[135] G. E. Golub and C. F. V. Loan, "Matrix computation," vol. 2, 1989.

[136] J. J. Dongarra, P. Luszczek, and A. Petitet, "The linpack benchmark: Past, present and future," *Concurrency and Computation: practice and experience*, vol. 15, no. 9, pp. 803–820, 2003.

[137] S. D. Conte and C. de Boor, *Elementary Numerical Analysis: An Algorithmic Approach*. McGraw Hill: New York, NY, USA, 1980.

[138] J. Hasler, "Analog architecture complexity theory empowering ultra-low power configurable analog and mixed mode soc systems," *Journal of Low Power Electronics Applications*, pp. 1–37, Jan. 2019.

[139] J. Liu, Y. V. Zakharov, and B. Weaver, "Architecture and fpga design of dichotomous coordinate descent algorithms," *IEEE Transactions CAS I*, vol. 56, no. 11, pp. 2425–2438, Nov. 2009.

[140] S. Brink, J. Hasler, and R. Wunderlich, "Adaptive floating-gate circuit enabled large-scale fpaa," *IEEE Transactions on VLSI*, vol. 22, no. 11, pp. 1–5, 2014.

[141] R. M. Walker, "An analogue computer for the solution of linear simulaneous equations," *Proceedings of the IRE – Waves and Electrons Section*, pp. 1467–1473, 1949.

[142] S. K. Mitra, "Electrical analog computing machine for solving linear equations and related problems," *Review of Scientific Instruments*, vol. 26, pp. 453–457, 1955.

[143] K. P. Lanneau and L. I. Griffin, "Analogue computer for solving simultaneous equations," *US Patent*, vol. 27, p. 2953, May 1959.

[144] Z. Sun, G. Pedretti, E. Ambrosi, A. Bricalli, W. Wang, and D. Ielmini, "Solving matrix equations in one step with cross-point resistive arrays," *Proceedings of the National Academy of Sciences*, vol. 116, no. 10, pp. 4123–4128, 2019.

[145] Z. Sun, G. Pedretti, and D. Ielmini, "Fast solution of linear systems with analog resistive switching memory (rram)," *IEEE ICRC*, pp. 1–5, 2019.

[146] Y. Xia, J. Wang, and D. L. Hung, "Recurrent neural networks for solving linear inequalities and equations," *IEEE Transactions CAS I*, vol. 46, no. 4, pp. 452–462, Apr. 1999.

[147] B. Ulmann and D. Killat, "Solving systems of linear equations on analog computers," *Kleinheubach Conference*, pp. 1–4, 2019.

[148] M. S. Ansari and S. A. Rahman, "A non-linear neural circuit for solving system of simultaneous linear equations," *IMPACT*, pp. 120–123, 2009.

[149] ——, "Mo-ota based recurrent neural network for solving simultaneous linear equations," in *International Conference on Multimedia, Signal Processing and Communication Technologies*, 2011, pp. 192–195.

[150] J. J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities," *Proceedings of NationalAcademy of Science*, vol. 79, 1982.

[151] ——, "Neurons with graded responses have collective computational properties like those of two- state neurons," *Proceedings of NationalAcademy of Science*, vol. 81, pp. 3088–3092, 1984.

[152] R. Umbehauen and A. Cichocki, *MOS Switched-Capacitor and Continuous-Time Integrated Circuits and Systems*. Springer-Verlag, 1989.

[153] A. Cichocki and R. Umbehauen, "Neural networks for solving systems of linear equations and related problems," *IEE*, vol. 39, p. 2, Mar. 1992.

[154] N. Guo *et al.*, *Energy-Efficient Hybrid Analog/Digital Approximate Computation in Continuous Time*. IEEE Journal of Solid State Circuits, 2016.

[155] Y. Huang, N. Guo, M. Seok, Y. Tsividis, and S. Sethumadhavan, "An analog accelerator for linear algebra," in *Proceedings of the 43rd International Symposium on Computer Architecture*, J. 1. . Seoul, Ed., 2016, pp. 570–582.

[156] H. B. Marr, B. Degnan, P. Hasler, and D. Anderson, *Minimization of Energy Per Op in an Asynchronous Pipeline Above and Below Threshold*. IEEE Trans. on VLSI, 2012.

[157] R. Chawla, A. Bandyopadhyay, V. Srinivasan, and P. Hasler, "A 531 nw/mhz, 128 x 32 current-mode programmable analog vector-matrix multiplier with over two decades of linearity," *IEEE Custom Integrated Circuits Conference*, pp. 651–654, Oct. 2004.

[158] C. S. and, "And hasler, "a highly dense, low power, programmable analog vector-matrix multiplier: The fpaa implementation,"" *IEEE Journal of Emerging CAS*, vol. 1, pp. 403–411, 2012.

[159] S. Koziol, R. Wunderlich, J. Hasler, and M. Stilman, "Single-objective path planning for autonomous robots using reconfigurable analog vlsi," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, pp. 1–14, 2017.

[160] R. Chawla, F. Adil, G. Serrano, and P. Hasler, "Programmable gm-c filters using floating-gate operational transconductance amplifiers," *IEEE Transactions CAS I*, vol. 54, no. 3, pp. 481–491, Mar. 2007.

[161] B. M. Hasler and C. Diorio, "An autozeroing floating-gate amplifier," *IEEE Transactions CAS II*, vol. 48, p. 1, Jan. 2001.

[162] S. Shah and J. Hasler, "Tuning of multiple parameters with a bist system," *IEEE Transactions CAS I*, vol. 64, no. 7, pp. 1772–1780, Jul. 2017.

[163] K. Odame and P. Hasler, "Theory and design of ota-c oscillators with native amplitude limiting," *IEEE Transactions CAS I*, vol. 56, no. 1, pp. 40–50, Jan. 2009.

[164] J. Hasler, "Circuit implementations teaching a junior level circuits course utilizing the soc fpaa, iscas 2018, florence, italy, may 2018," *p*, pp. 1–5,

[165] J. Hasler, S. Kim, and F. Adil, "Scaling floating-gate devices predicting behavior for programmable and configurable circuits and systems," *JLPEA*, vol. 6, no. 13, pp. 1–19, 2016.

[166] A. Natarajan and J. Hasler, "Using soc fpaa and integrated simulator for implementation of circuits and systems in education," in *2017 IEEE International Symposium on Circuits and Systems (ISCAS)*, IEEE, 2017, pp. 1–4.

[167] J. Hasler, A. Natarajan, S. Shah, and S. Kim, "SoC FPAA immersed junior level circuits course," in *2017 IEEE International Conference on Microelectronic Systems Education (MSE)*, IEEE, 2017, pp. 7–10.

[168] J. Hasler and A. Natarajan, "An open-source toolset for fpaa design,"

[169] A. Natarajan and J. Hasler, "Built-in self-test of vector matrix multipliers on a reconfigurable device," in *2020 IEEE International Symposium on Circuits and Systems (ISCAS)*, IEEE, 2020, pp. 1–5.

[170] A. Natarajan and J. Hasler, "Analog solutions of systems of linear equations on a configurable platform," in *2020 IEEE International Symposium on Circuits and Systems (ISCAS)*, IEEE, 2020, pp. 1–5.

[171] J. Hasler and A. Natarajan, "Continuous-time, configurable analog linear system solutions with transconductance amplifiers," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 68, no. 2, pp. 765–775, 2020.

[172] A. Natarajan and J. Hasler, "Implementation of synapses with hodgkin huxley neurons on the fpaa," in *2019 IEEE International Symposium on Circuits and Systems (ISCAS)*, May 2019, pp. 1–5.

[173] J. Hasler, "Defining analog standard cell libraries for mixed-signal computing enabled through educational directions," in *2020 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2020, pp. 1–5.

VITA

Aishwarya Natarajan was born in Trivandrum, India. She received the B.Tech degree in Electronics Engineering from University of Mumbai, India, in 2014 and M.S. degree from Georgia Institute of Technology, Atlanta in 2016. She received her Ph.D. degree in Electrical and Computer Engineering at Georgia Institute of Technology, Atlanta in 2021. She is a recipient of the Analog Devices Outstanding Student Designer award in recognition of excellence in contribution to IC Design. Her research interests include analog and mixed signal integrated circuits and systems design, neuromorphic computing and low-power bio-inspired circuits and systems.