CAN MY CHIP BEHAVE LIKE MY BRAIN?
OR
RECONFIGURABLE MIXED SIGNAL NEUROMORPHIC ARCHITECTURES

A Thesis
Presented to
The Academic Faculty

by

Suma George

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

Georgia Institute of Technology
May 2015

CAN MY CHIP BEHAVE LIKE MY BRAIN?
OR
RECONFIGURABLE MIXED SIGNAL NEUROMORPHIC ARCHITECTURES

Approved by:

Professor Jennifer Hasler, Advisor
School of Electrical and Computer Engineering
Georgia Institute of Technology

Professor Hua Wang
School of Electrical and Computer Engineering
Georgia Institute of Technology

Professor David Anderson
School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Michael Pfeiffer
Institute of Neuroinformatics
University of Zurich and ETH Zurich

Professor Omer Inan
School of Electrical and Computer Engineering
Georgia Institute of Technology

Date Approved: 6 April 2015

To my parents,

Suju and Ami

Because You're mine I walk the Line!

ACKNOWLEDGEMENTS

I want to thank all the people who make my life wonderful. There are many people who have been with me throughout my journey through graduate school and it has been much easier because of them. First and foremost I would like to thank my advisor Jennifer Hasler who has been my guide, teacher, friend and even therapist at times. I have always admired her technical prowess and the breadth of her knowledge. She has also taught me many life lessons on how to manage time and stress. I will be forever grateful for all your guidance. It has truly helped me grow both professionally and personally. I would also like to thank my committe members. David Anderson, who has been on my committe since my MS Thesis days. Omer Inan, and Hua Wang for all the wonderful feedback and support. Michael Pfeiffer for all your technical feedback both for the thesis and at the neuromorphic workshops at Capocaccia and Telluride.

I would like to thank my dearest parents. Mom and Dad you have always been my source of unconditional love and support. Mom, for helping me believe nothing is impossible if I set my mind to it. Dad, for teaching me the value of hard work and diligence. Your prayers and blessings have helped me achieve and become all that I am today. Thank you with all my heart! Suju, my elder sister for always blindly believing in me and egging me on. Ami, my younger brother for always helping me 'keep it real'! This journey would be incomplete without you. Aneesh Jiju, Annie and Joey my nieces, for keeping me on my toes! Leela aunty and Matchu for all their help. Fr. Matthew and Joy Uncle for being my guardian angels in the US. Joy uncle for all wonderful Halloween breaks at Disney. I'm truly blessed to have such a wonderful family.

Friends are the family we choose and I have been blessed with some really great people as friends. Smriti Chopra for being my sister in arms while we both pursued our PhDs. Through all my ups and downs you have been a constant, helping me navigate the craziness and celebrate the victories. It would not be any fun without you! Sudipto Das, for being chirpy, comical, and cynical. I know it is hard for one person to be all that, but you helped me see the lighter side of difficult moments. I'm very grateful to count you as one of my best friends. Michelle Collins for being the younger sister I never had. You are one of the sweetest people I know. I admire your integrity and strength. I'm cheering to see you succeed! Aishwarya Sarath, you always believed in me and would hear me out. You are truly cherished. Koshu, for being my strength and heart. You are an angel! KD, for being my rock and hearing me through countless sob stories. Aman Praji for being such a big support for me throughout, lucky to have a brother like you. Also would like to thank Amritha Arakali and Anjali Ashok for being awesome roommates and the countless karaoke nights. Samina Jamil, for your poetry and wonderful company.

Labmates are not just friends but an extension to your family as well, who help you keep it all going smoothly. I had the unique opportunity of being in the middle of the transition of old and new Integrated Computational Electronics (ICE) members. Richie Wunderlich for being my mentor, friend and counselor. Stephen Nease for being a great friend, I learned so much from you. Scott Koziol, for your positive attitude and advice and being an excellent collaborator. Shubha Ramakrishnan for being a guide, friend and an awesome chef feeding me Sambhar and other south Indian delicacies. Farhan Adil, for being a guide and a good friend. Craig Schlottmann for being extremely helpful and patient. Stephen Brink for the awesome technical discussions and advice. Arindam Basu, for getting me interested in neuromorphic circuits. As for new icicles Michelle Collins, Sihwan Kim and Sahil Shah, thanks for all the support while testing the ICs. I have really enjoyed our discussions both

technical and not. Sahil, keep laughing- it is refreshing to see you have a rather sensitive funny bone! Alex Cardwell, for being my punching bag, friend, and correcting my grammar. I have enjoyed our discussions on wide range of topics from religion, philosophy to neuromorphic systems. Andrew Freedman, your perseverance has been a stellar example for me. I wish all the Icicles past and present all the very best! Also would like to thank Dean Walker, my manager at Blackberry. You are one of the most amazing people I have met. I have learned a lot from you. Sudha Jha ma'am, you have been an inspiration for me throughout. Thank You all for being part of my journey!

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

'*Can My Chip Behave Like My Brain?* ' As a young graduate student at Georgia Tech looking at Prof. Hasler's research, this was the first query I had. Coming from a Very Large Scale Integration (VLSI) background, it was very exciting for me to see biophysical models of neurons using CMOS transistors! My research goal thereon has been to investigate how to build efficient neuromorphic systems using mixed signal reconfigurable architectures. Many decades ago, Carver Mead established the foundations of Neuromorphic Systems. Neuromorphic systems are analog circuits that emulate biology.They utilize sub-threshold dynamics of CMOS transistors to mimic biology. The objectives are to emulate biological processes, and also build useful applications using these bio-inspired circuits.

In this research we will learn how we can achieve this by using reconfigurable hardware like field programmable analog arrays(FPAA). FPAAs enable configuring/prototyping different systems on a unified platform. As digital systems saturate in terms of power efficiency, this alternate approach has the potential to improve computational efficiency by about eight orders of magnitude. These systems include analog, digital and neuromorphic elements as building blocks; an amalgamation of all of the above results in a very powerful processing machine. These systems can then be used to implement complex algorithms like Artificial Neural Networks, Winner-Take-All (WTA) and word-spotting to build ultra low-power applications.

This body of work is divided into three main parts. First we will discuss reconfigurable systems and talk about the latest FPAA System-on-a-chip (SoC) built. We will discuss some mixed signal, analog and digital examples as well as demonstrate a command word classifier. Second, we will discuss the VLSI CAD tools developed to make system design feasible on these SoCs. Third, we will talk about neuromorphic

architectures and applications one can build using these systems. We will go over bio-inspired modeling of dendrites and how along with other bio-physically based models of the soma, synapse and channels we can make a neuron block. We will discuss how combining all these approaches enables us to build efficient low power systems.

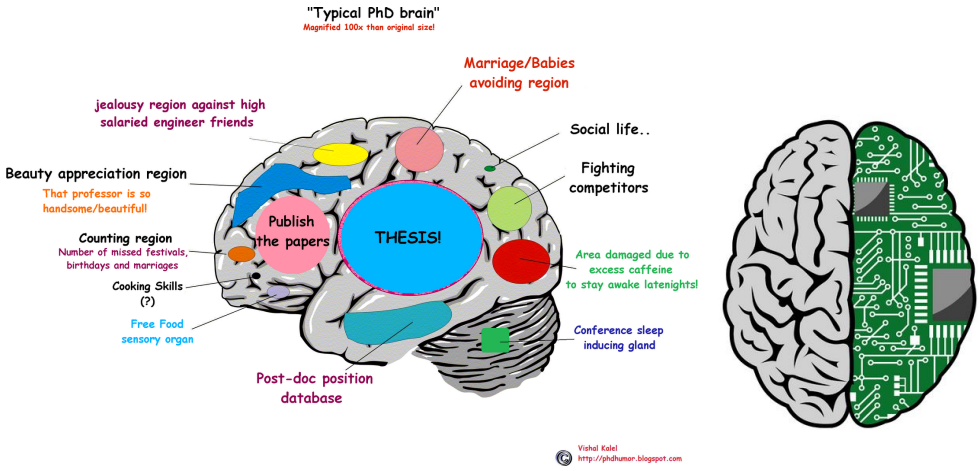
Why is this important? Modern day technology relies heavily on silicon devices to do computation. These systems have however hit an energy efficiency wall and hence we need to look at different solutions that will help break this efficiency barrier. This can be done by using reconfigurable and programmable analog solutions and we can further this approach by using bio-inspired systems. Together with existing mixed signal systems and neuromorphic models, we can build ultra efficient low power systems. Thus, along with *in vivo* studies, *in silico* studies are also very important. Building a neuromorphic supercomputer is within our grasp, and is a grand challenge of our times.

CHAPTER I

RECONFIGURABLE MIXED SIGNAL NEUROMORPHIC ARCHITECTURES

"The brain is a monstrous, beautiful mess. Its billions of nerve cells called neurons lie in a tangled web that displays cognitive powers far exceeding any of the silicon machines we have built to mimic it."

William F. Allman [99]



(a) (b)

Figure 1: (a) Typical PhD student's 'brain' (b) Efficient System On Chip(SoC) that can perform complex tasks like the brain aka "what we want"!

The human brain, though studied a lot, is still a mystery when it comes to its functioning. Scientists are yet to determine what the intricate relationships and functions among various parts of the brain are. Merely knowing all the components of the brain doesn't lead to a sound understanding of how they interact with each other [103]. Numerous comparisons have been made between the brain and a digital computer; the biggest similarity being they both process information. What sets the brain's neural

networks apart is a very high computational efficiency, robustness and the ability to solve structured as well as ill-structured problems. The digital computer has advantages of being very precise for well-structured problems. However, most real-world problems are not structured in nature. Therefore, even though a lot of progress has been made to develop systems that tackle real world problems, for example IBM Watson; we haven't yet designed a system that can function or adapt like the human brain and do so with minimal power. One can joke about a young graduate student's brain as shown in Fig. 1(a) with our sole focus being our thesis but it is only natural to draw comparisons between Very Large Scale Integrated (VLSI) chips which are ubiquitous as shown in Fig. 1(b). In this thesis we will explore how to combine different areas under VLSI system design and use it to build computationally efficient machines. Thus a more technical title for this thesis would be 'Reconfigurable Mixed Signal Neuromorphic Architectures'. VLSI systems pioneered by Carver Mead and Lynn Conway has spawned into multiple areas as shown in Fig. 2. This thesis will cover discussion about mixed signal systems reconfigurable systems, neuromorphic systems as well as CAD tools that help us design such complex systems.

Neuromorphic engineering is an area pioneered by Carver Mead in the late 1980s which endeavors to do this. Silicon devices operating in the sub-threshold regime and biological structures share similar physical principles of operation. This implies that silicon devices can be used to *emulate* biological systems. The consequences of this statement are two-fold. First, we can use Neuromorphic circuits to emulate biological systems and second, we can use these systems to perform novel computation. Neuromorphic engineering though considered a non-traditional approach, has a lot of potential to look into real world problems as well as model biology. I have focused my research on building systems that leverage digital, analog and bio-inspired circuits. The goal is to build a powerful prototype for a neuromorphic processor. These low-power reconfigurable systems, can be used to solve different problems like image

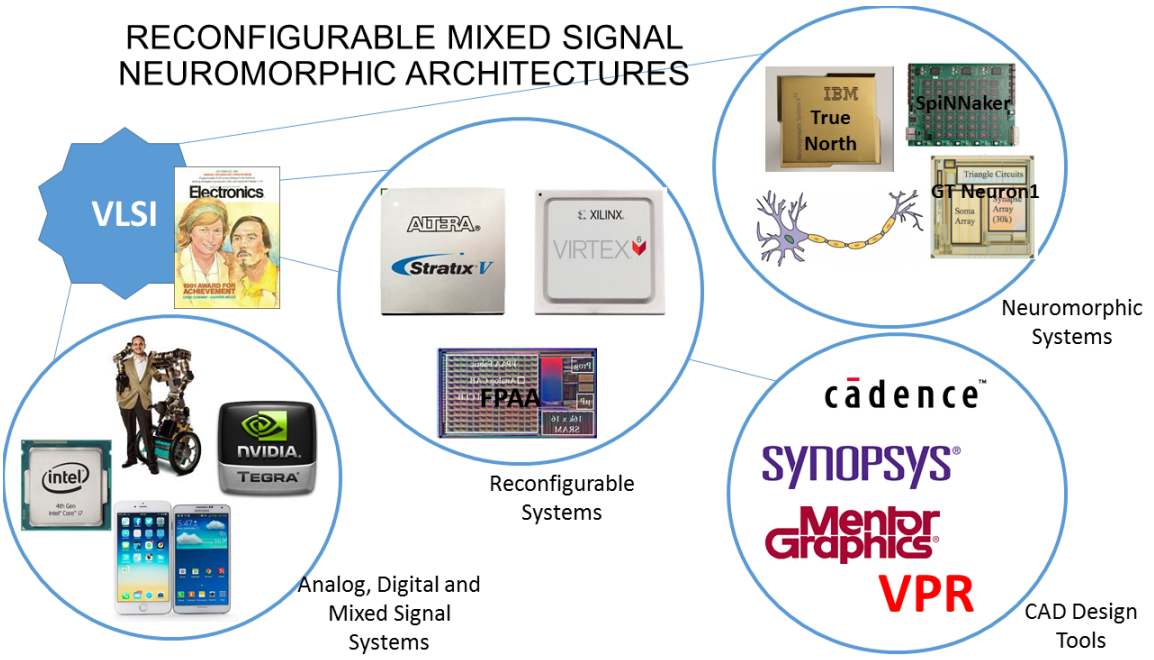


Figure 2: VLSI systems were pioneered by the Mead-Conway approach. Its defining Characteristics were top-down design, stress on system-level concepts, merging separate disciplines to create a new, simplified methodology, Present a small set of key concepts from a range of topics, to carry along the least amount of mental baggage as well as focus on Starting with education. Now VLSI systems are ubiquitous. This revolution spawned many different areas. You have analog, digital and mixed mode system chips driving computation on our smartphones, electronic gadgets, robots etc with industry leaders like Intel, Qualcomm, Nvidia, Apple, Samsung to name a few. We have reconfigurable systems like Field Programmable Gate Arrays (FPGAs) with market leaders like Xilinx, Altera etc. and FPAAs [153–158] that enable quick prototyping of digital and analog systems respectively. We have neuromorphic systems which propose novel physical algorithms that can be used to build efficient machines. We have digital implementations like the IBM True North, University of Manchester's SpiNNaker, Georgia Tech's Neuron ICs [164,165]. Last but not the least we have CAD design tools that actually make the implementation of systems possible on hardware and make design easier. Prominent industry leaders are Cadence, Synopsys, Mentor Graphics as well as open source tool efforts like Virtual Place and Route(VPR) [121].

processing, speech processing applications, prototyping and build simple circuits in the classroom. A technology evolution roadmap for neuromorphic engineers has been proposed by Hasler et al. as seen in Fig. 3, to get the same foresight that IC designers gained from Moore's law many years ago. Scaling of energy efficiency, performance,

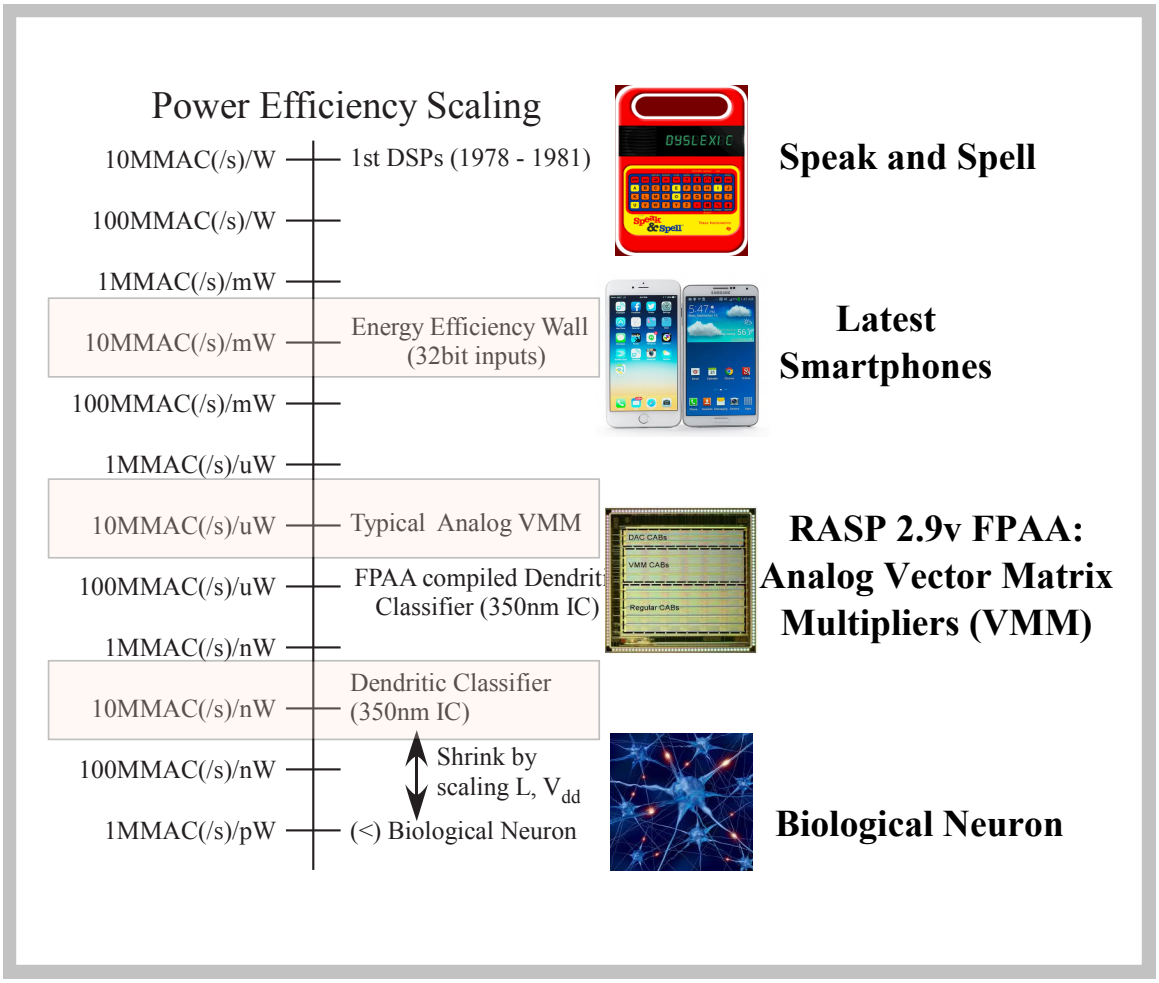
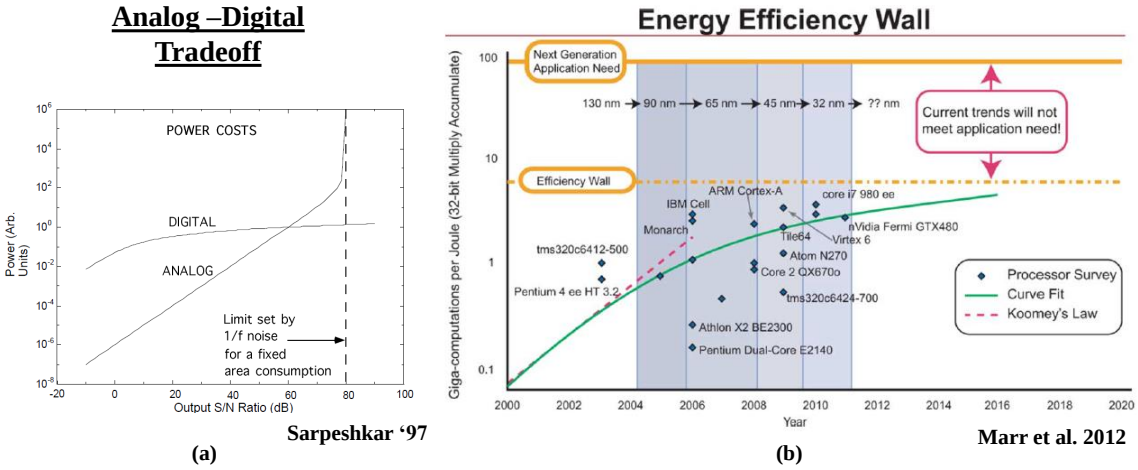


Figure 3: Power efficiency scaling has hit an efficiency wall in recent times with state of art digital processors as shown in a survey in 2012 in [16, 101]. Analog as well as bio-inspired solutions can help us further scale to ultra low power systems.

and size is discussed as well as how the implementation and application space of neuromorphic systems is expected to evolve over time [101]. This work endeavors to be a step in the direction of building such large-scale neuromorphic systems.

My research goal is to build computationally efficient bio-inspired systems and applications. As an engineer, I believe building bio-inspired systems gives me a unique outlook to not just understand the brain, but also utilize my findings to build bio-inspired systems for real world applications. In the process of doing so I cover multiple areas like neuromorphic systems, reconfigurable architectures, low power system design, mixed signal CAD tools, and application of these technologies for speech



Solve the parts of the problem in the domain most efficient for that problem. Reconfigurable systems enable that.

Figure 4: Limitations when using Analog vs Digital Systems. (a) Here is a plot demonstrating the tradeoff in terms of energy and SNR. We want to perform computation in the most efficient domains depending on our requirements. We can leverage analog processing to enable lower power systems while using digital systems when we need more precision. We build reconfigurable architectures to enable this flexibility. (b) Survey done by Marr et al. [16] demonstrating an enery efficiency wall for current state of the art digital processors.

processing, pattern recognition and robotics.

1.1 Neuromorphic Systems

Neuromorphic systems are analog circuits that utilize subthreshold dynamics of CMOS transistors to mimic biology. The objective is not just to simulate the human brain, but also to build useful applications using this knowledge for speech recognition, image processing, and robotics. As digital systems saturate in terms of power efficiency as shown in Fig. 4, this alternate approach becomes more attractive.

Neuromorphic hardware models the behavior of biological neural systems to enable efficient computational modeling. It leads to a significant reduction in size and power compared to the traditional approaches of modeling based on numerical integration on a digital computer.

1.1.1 Dendritic Computation

I focus my research on a much smaller yet important component of the nervous system- dendrites. Dendrites are essentially tree-like structures that connect neurons. Dendritic computation is often ignored and a point neuron model is typically adopted. However, studies show dendrites perform operations such as nonlinear filtering, spatial and temporal summation of synaptic inputs, coincidence detection, synaptic scaling and sequence detection. I have demonstrated that by exploiting the directional selectivity and coincidence detection properties of dendrites, we can implement a word-spotting network that can be used in many classifier applications using our VLSI chip. The word-spotting network is similar to a Hidden Markov Model (HMM) classifier that is often used in speech and pattern recognition.

1.1.2 Neuromorphic Integrated Circuits (ICs)

I was the co-architect in building a reconfigurable neuromorphic IC with neurons and a mixed signal fabric to further study these architectures. I developed an efficient and scalable hardware system for studying dendritic computation in large scale networks with programmable learning synapses and dendrites that support arbitrarily branched dendrites. This architecture also includes active channels in the dendrite for non linear filtering. Effectively, it is a multilayer neural network within each neuron. This system is a way to study and gain insight into building larger computationally efficient systems in the future.

1.1.3 Modeling the Brain

The ultimate goal is to build a system that matches or exceeds the complexity of the human brain. It is advantageous to build smaller applications emulating functionality of basic elements using silicon models of the neurons, synapses, and dendrites to build networks. Learning can be implemented on these networks using floating gate (FG) transistor technology, which can be used as a memory element that simulates learning.

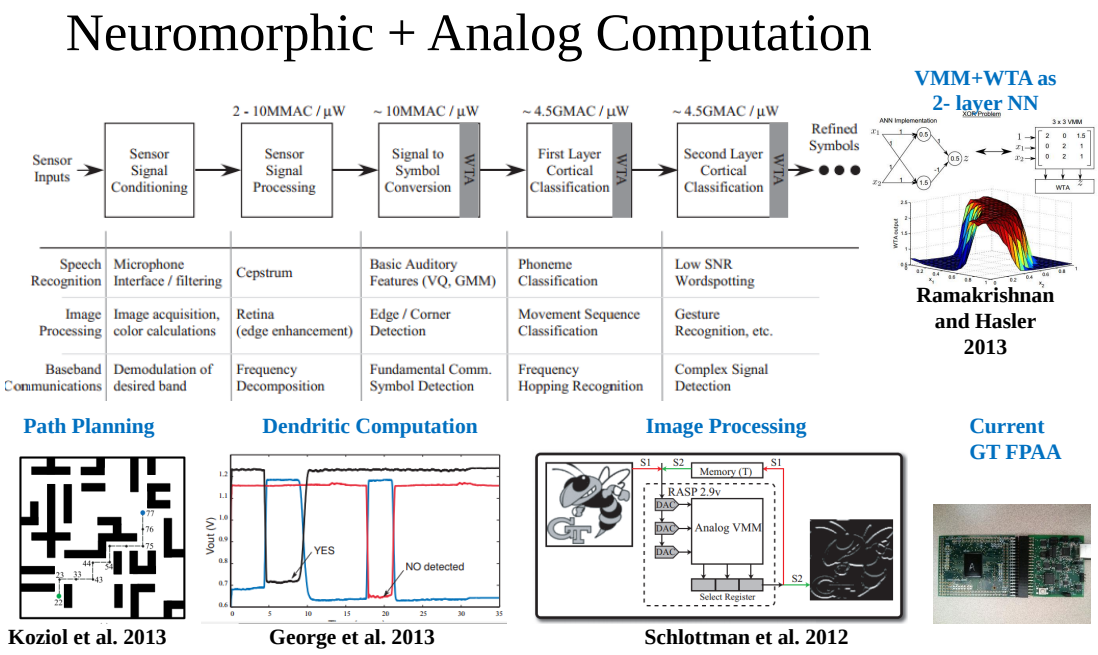


Figure 5: Applications like image processing, path planning for robots, wordspotting using dendritic classifier and a VMM+WTA circuit as a universal approximator have been demonstrated using neuromorphic and analog solutions on FPAAs.

Modern day technology relies heavily on silicon devices to do computation. However, there is a critical need to invest in silicon to build bio-inspired systems. With existing mixed signal systems and neuromorphic models, we can build ultra efficient low power systems. Thus, *in silico* studies are very important as are *in vivo* studies. Building a neuromorphic supercomputer, or a "Silicon Brain" is within our grasp, and is a grand challenge of the twenty-first century.

1.1.4 Applications of Neuromorphic and Analog Systems

One of the goals of my research is to not only to mimic biology in silicon, but also utilize these bio-inspired systems to solve real world problems like speech classification, pattern recognition, robotics etc. as shown in Fig. 5 I have demonstrated a YES/NO classifier using dendrites and a Winner Take All (WTA) circuit using our VLSI chip. I believe that by using this dendrite based neuromorphic classifier and other front end techniques, I can build an effective audio recognition system which

can be used for a wide variety of applications in speech/audio processing particularly for phoneme recognition. We will discuss more about this model in chapters 3 and 4.

1.2 Reconfigurable Mixed Signal Architectures

The backbone of my research that enables building such systems is reconfigurable hardware and a software CAD toolset. Reconfigurable hardware for digital computation is the norm these days, namely the Field Programmable Gate Arrays (FPGAs). Similarly for analog solutions, we have Field Programmable Analog Arrays (FPAAs). In such a platform, analog components are embedded in a switch fabric that enables arbitrary connections between them. We use FG as the switch element, as this adds the properties of non-volatility and compactness. Combining both analog and digital components into a FG switch fabric, we can leverage the best of both worlds.

1.2.1 Low Power IC Design:

I have been instrumental in developing and testing a new generation of Reconfigurable Analog Signal Processor (RASP) 3.0 family of SoCs designed by our research group, fabricated in the 350 nm technology. This design effort addressed a lot of the interfacing questions and made our systems more compact. The SoC has a processor, memory, ADCs, DACs and other peripherals on chip. The neural IC is a variation of this as it contains not just analog and digital blocks, but neuron blocks as well. The global interconnect between all tile elements is FPGA-style manhattan routing. These are very powerful SoCs that I plan to use. I also hope for students to learn IC design by building chips through MOSIS. I hope to build RASP peripherals for these chips that can specialize as sensor blocks that interface with the existing IC.

1.2.2 Hardware Software CoDesign/ CAD tools:

I was instrumental in developing a new CoDesign environment $x2c$ for simulating and programming reconfigurable FG based mixed signal SoCs. These SoCs consist

of an integrated processor, I/O peripherals, and a FPAA comprised of analog and digital components. This novel open source tool platform empowers the user to seamlessly CoDesign low power analog and digital systems in a single environment. This approach integrates multiple open source tools to develop a coherent user friendly design flow. Scilab is the graphical front end for system level block design, which invokes Verilog to Routing (VTR)/ Versatile Place and Route (VPR) tools.$vpr2swc$-VPR to switches tool handles analog component packing, integrates the system, then generates switches to program and test the IC. We demonstrated several mixed signal examples, as well as how to perform useful computation using the routing fabric. This is a very powerful open source platform that will be open to a wider audience post publication and will be very effective for teaching in the classroom. The tool can be extended to any new family of heterogeneous ICs.

1.3 Overview

In chapter 2, I will present an System-on-a-chip (SoC) that integrates divergent concepts from previous multiple large-scale FPAA designs along with low-power digital computation and interface circuitry (i.e. DACs, ADCs). This unified structure enables a wide range of a system-on-a-chip computing options that can be optimized for a wide range of parameters (i.e. Power); the resulting IC architecture is the most sophisticated FPAA device built to date.

In chapter 3, I will talk about the CAD synthesis tool $vpr2swcs$ for targeting floating gate (FG) based mixed-signal SoCs of the RASP 3.0 family. These SoCs consist of a digital processor, Field Programmable Analog Array (FPAA) fabric, DACs, ADCs and peripherals. The tool is used for building parametric FPAA architectures that consist of both digital and analog blocks. I will discuss here the modifications, challenges and novel solutions proposed while doing mixed signal system design. Mixed signal examples will be demonstrated. Also we will see how the routing fabric can be

leveraged for computation.

In chapter 4, An Analog-Digital Hardware-Software CoDesign environment for simulating and programming reconfigurable systems will be presented. For this thesis, we will focus on a large-signal SoC Field Programmable Analog Array (FPAA) comprised of analog and digital components, consist of an integrated processor with I/O peripherals, and based on Floating-Gate (FG) devices and circuits, although the approaches can be extended to other platforms. The open-source tool platform, integrating multiple open source codes, empowers the user to do seamless low-power analog-digital CoDesign in a single environment that generates and implements high-level simulation and experimental measurement of the resulting hardware system. The tool flow will be demonstrated with multiple mixed signal examples through this configurable system.

In chapter 5, we will foray into the world of neuromorphic systems especially dendrites. Many decades ago, Wilfrid Rall and others laid the foundations for mathematical modeling of dendrites using cable theory. With reconfigurable analog architectures, we are now able to accurately program different circuit architectures to emulate dendrites. Our work has shown that these circuits accurately reproduce results predicted from cable theory when inputs to the system are small. For large inputs, interesting nonlinear effects begin to take hold.

In chapter 6, we will talk about how a network of dendrites can be used to build the state decoding block of a wordspotter similar to a Hidden Markov Model (HMM) classifier structure. Simulation and experimental data will be presented for a single line dendrite and also experimental results for a dendrite-based classifier structure. This work builds on previously demonstrated building blocks of a neural network: the channel, synapses and dendrites using CMOS circuits. These structures can be used for speech and pattern recognition. The advantage of such a structure over digital systems is ultra low power consumption.

In chapter 7, we will discuss how to build neuromorphic systems to be able to mimic biology and also solve more complex problems.To this end, we design neuromorphic chips using biologically inspired circuits. We build a neuron chip embedded in FPGA style routing architecture which models dendrites using transistors as discussed in previous chapters. I will discuss the $Neuron2$ chip in this chapter as well as the RASP 3.0N SoC, which belongs to the RASP 3.0 family but in addition to analog and digital components has neuron blocks as well.

In chapter 8, I will summarize the work done so far as well as future directions for this research which I hope to pursue.

CHAPTER II

RECONFIGURABLE SOC: RASP 3.0

Reconfigurable Analog Signal Processor (RASP) 3.0 is an integrated Ultra-Low Power System-On-Chip (SOC) enabling configurable and programmable analog and digital computation and interfacing. The potential of energy efficiency improvement over current computing approaches (i.e. a factor of 1000 or better) [34, 35], as well as the saturation of energy efficiency in digital computation [36], puts this IC at an important industrial pain point for the embedded systems industry for a range of applications such as acoustic, vision, communications, and small robotics. We see this IC as both an analog–digital computational device as typical expected from a microprocessor (μP), as well as enabling analog and digital interfacing and control of an embedded system. We demonstrate this IC in a 350nm CMOS process; such approaches could be possible in scaled down IC processes as well.

This SoC integrates early concepts of rapid reconfigurable analog computation [37], along with early demonstration of configurable fabric of interdigitated analog and digital computing blocks [38], along with resulting μP (open-source MSP430 [39]) based computing and control, to address a wide range of ultra-low power embedded system computational needs. This large-scale Field-programmable analog array (FPAA) still enables analog computational energy efficiencies of 1000X as well as area efficiencies of 100X over digital solutions. The following sections describe this FPAA IC architecture, basic analog and digital computational approaches, capacitance, timing, and rapid reconfigurability of the configurable routing fabric, implementation of data converters in the mixed mode fabric, computation and classification utilizing the routing fabric as part of the computation, and system examples. The SoC is

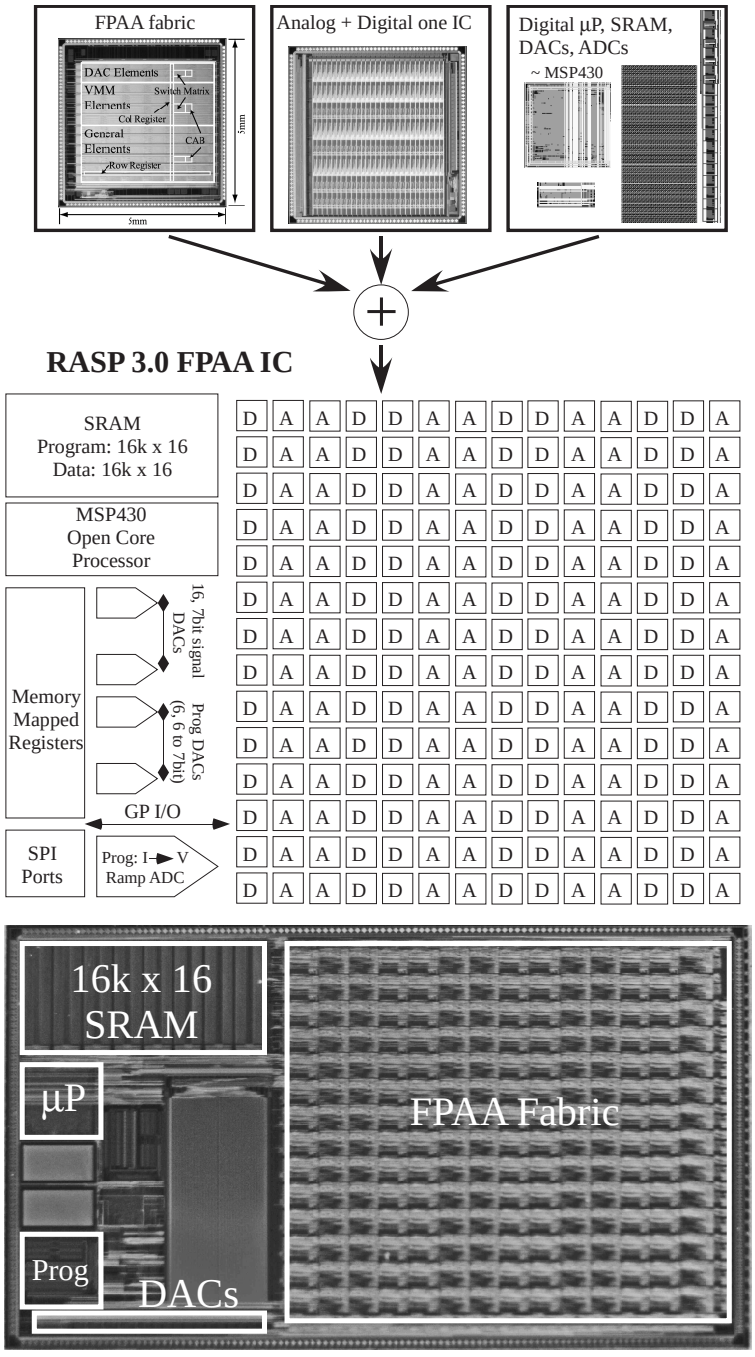


Figure 6: The RASP 3.0 integrates divergent concepts from previous multiple FPAA designs [37, 38, 40] along with low-power digital computation and interface circuitry (i.e. DACs, ADCs). This unified structure enables a wide range of a system-on-a-chip computing options that can be optimized for a wide range of parameters (i.e. Power); the resulting IC architecture is the most sophisticated FPAA device built to date. We show the die photo of the 12mm x 7mm FPAA device fabricated in a 350nm standard CMOS process.

illustrated in Fig. 6

2.1 Architecture Description of the FPAA SOC IC

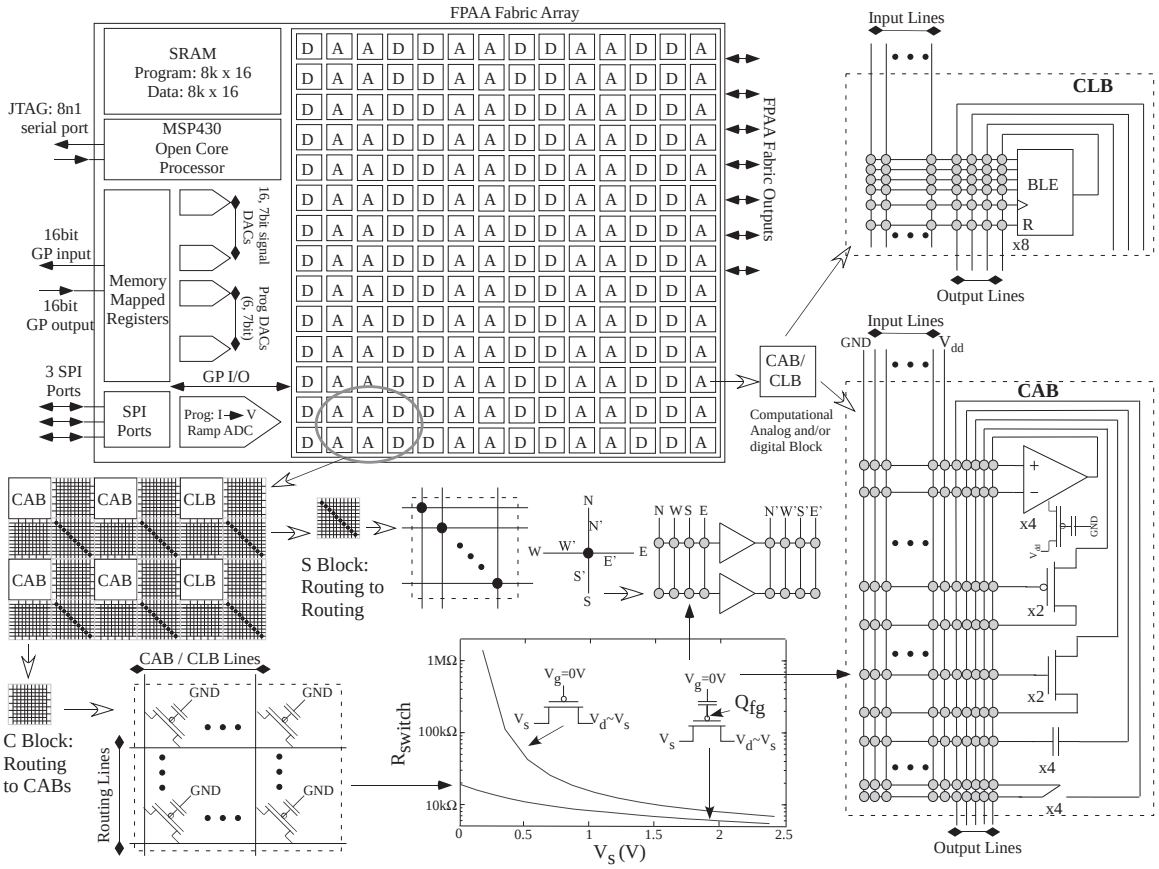


Figure 7: RASP 3.0 functional block diagram illustrating the resulting computational blocks and resulting routing architecture. We start showing the block diagram of the FPAA IC. The infrastructure control includes a μP developed from an open-core MSP 430 processor [39], as well as on chip structures include the on-chip DACs, current-to-voltage conversion, and voltage measurement, to program each Floating-Gate (FG) device. This configurable fabric device utilizes a manhattan geometry architecture to both effectively integrate analog (A) and digital (D) components, as well as build a compiler tool-friendly hardware platform. The floating-gate switches in the Connection (C) Blocks, the Switch Blocks (S), and the local routing are a single pFET Floating-Gate (FG) transistor programmed to be a closed switch over the entire fabric signal swing of 0 to 2.5V [69]. The Computational Analog Blocks (CAB) and Computational Logic Blocks (CLB) are similar to previous approaches [38]. Eight, 4 input Boolean Logic Element (BLE) lookup tables with a latch comprise the CLB blocks.

Figure 7 shows the block diagram for the RASP 3.0 FPAA IC based on a manhattan FPAA architecture, including the array of computation blocks and routing,

composed of Connection (C) and Switch (S) blocks. The computation is a combination of the components in the Computational Analog Blocks (CAB) and Computational Logic Blocks (CLB), as well as utilizing the devices in the routing architectures that are programmed to non-binary levels. We use data-flow architectures for power-efficient computing to merge as much as possible computation and memory, minimizing the amount of external memory access required.

The architecture is based on Floating-Gate (FG) device, circuit, and system techniques. The programming approach, based on previous all-digital infrastructure concepts [41], is fully integrated on-chip. The μP and other control infrastructure allow all programming of FG devices on-chip by simply downloading the list of switches to be programmed. The code for programming is eliminated when the processor is executing, efficiently utilizing the on-chip SRAM capabilities. The external system, through a serial port interface, first loads the programming code, and then executes the programming code on the downloaded data. The details of the FG programming approach will be explained in further sections of this chapter.

The processor is able to send information to and from the array through memory mapped I/O special purpose peripherals. These peripherals include ADCs and DACs, allowing measurements to be performed on chip, with the data taken by and stored in the processor. There are 16 memory mapped 7bit signal DACs for the architecture, as well as additional DACs / ADC for the FG programming. The processor supplements the processing power of the digital portion of the system and increase overall implementation flexibility; portions of a problem can be mapped to reconfigurable analog, reconfigurable digital, or a general purpose digital processor.

This FPAA SOC, with a highly programmable, fine grain fabric enabling signal-processing approaches, practically requires a toolset for system design in a reasonable design timeframe. Using manhattan geometry enables the use and direct modification of open-source tools, like VPR [42], for the place and route approaches, a huge

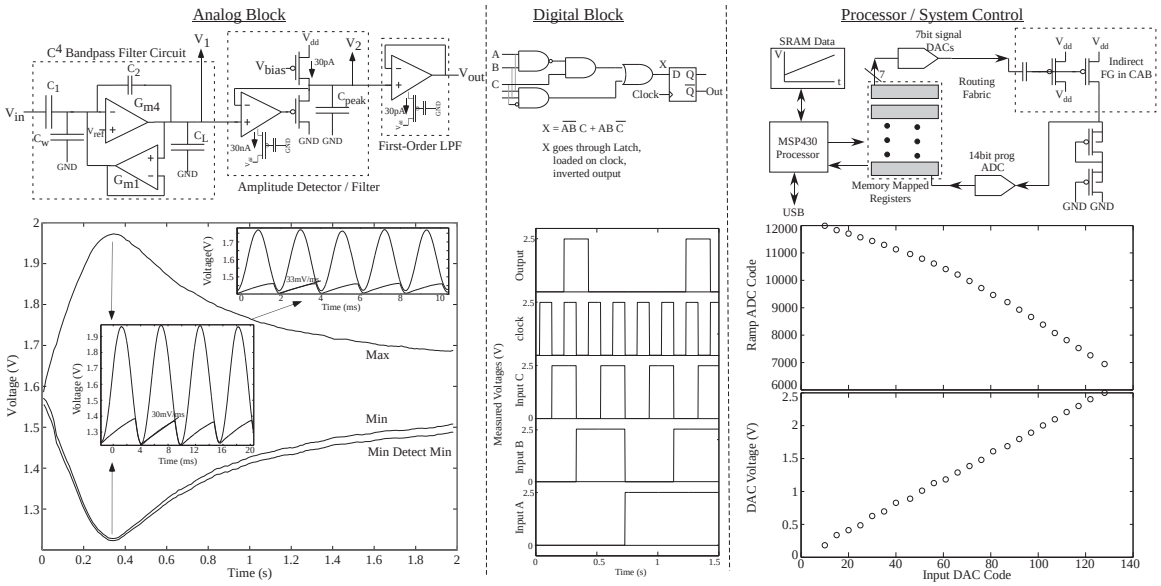


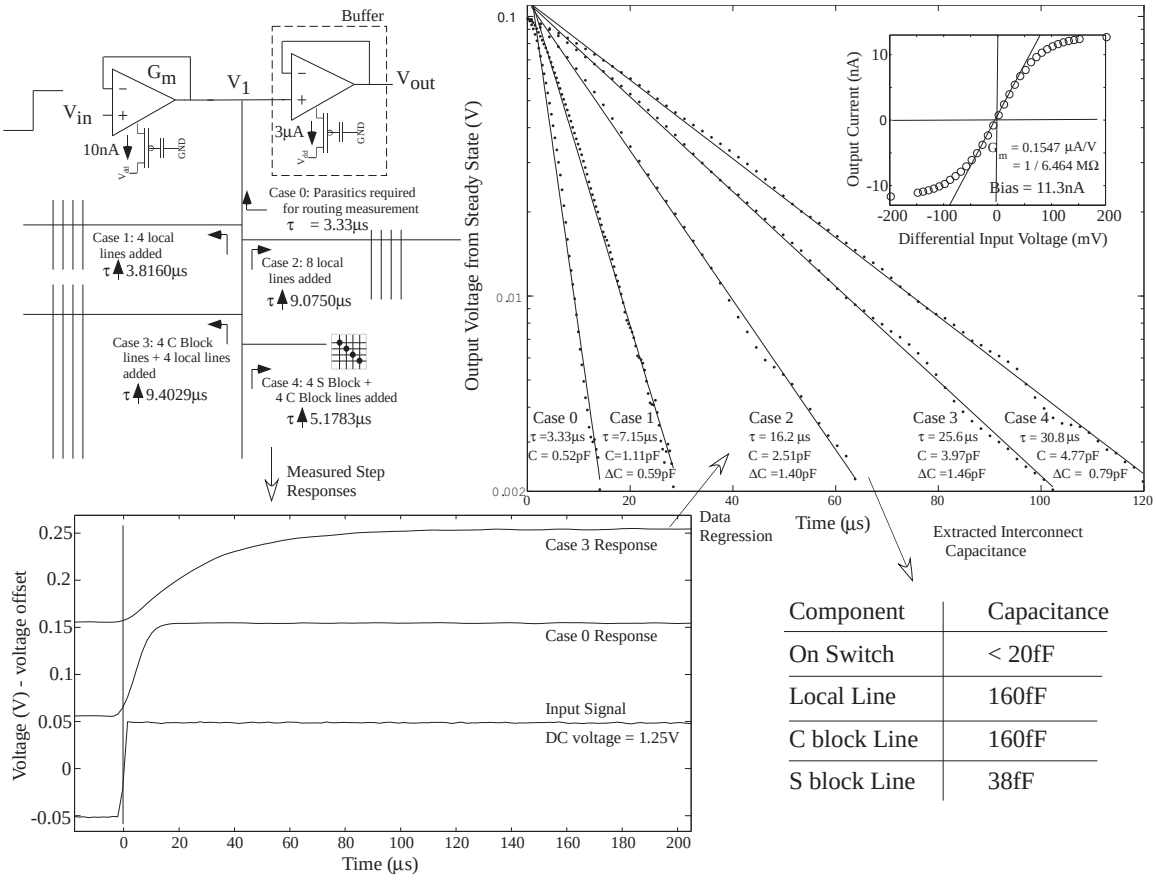
Figure 8: Approach for the RASP 3.0 FPAA IC enables integration of Analog and Digital Blocks in the routing fabric, as well as resulting standard digital computation (i.e. uP) and infrastructure. **Analog Blocks:** Compilation of single signal processing chain, including a second-order bandpass filter, amplitude detection, and smoothing of the output signal. **Digital Blocks:** We can compile digital blocks using the look-up tables in the CLB, the resulting latches in the CLB, as well as the routing fabric. We demonstrate basic capability in a single BLE element in a CLB for a simple combinatorial function. **Digital Computation / Infrastructure:** The μP design is an open-core MSP 430 processor with on-chip structures for 7-bit signal DACs, a ramp ADC, memory mapped General Purpose (GP) IO and related components. We show a measurement through a FG transistor in a CAB utilizing the processor, signal DACs and memory mapped register. We often instrument similar loops for instrumenting and measuring analog and digital blocks; in general we utilize all of these capabilities as part of the FPAA computation.

improvement over previous home grown tools [43] in both dense (system-level) compilation as well as optimizing routing infrastructure for digital and analog constraints. Our high level toolframework is recently built in Scilab / Xcos based on our earlier tool development in MATLAB / Simulink [44–46]. The graphical (i.e. Simulink) based compilation and macromodeling tool enables the FPAA to be an embedded system where the user has control over the resulting analog-digital co-design problem. The rest of the user control is through GUI interfaces. The detailed discussion of the recent tool flow is followed by this chapter. Higher-level tools also enable the use of these systems in educational experiences [47, 48], which will be essential to educating engineers to *design* for system applications.

Figure 8 shows our SOC FPAA approach enables integrated analog interfacing and computation,with digital blocks, both FPGA and μP blocks. We show the compilation of auditory processing chain for subband signal detection. Where possible, one wants to compile key blocks into a single CAB to minimize parasitic capacitances as well as minimize global routing requirements. The analog computation, utilizing significant innovations enabling integration of previous heterogeneous concepts [37,38,40], would seem familiar given our earlier FPAA designs. What is unique is the integration of digital low-power programmable and configurable FPGA fabric, first attempted in [38], to fully enable the routing of analog and digital signals through a continuous fabric. Further, we integrate these capabilities with an on-chip μP component and a range of digital communication ports (i.e. SPI ports), completing the picture that this FPAA device is a SoC computing device, not just a device for analog processing. Further, the interaction of analog computation, digital FPGA like components, and a μP infrastructure integrated together creates, in general, a significant co-design problem between these three domains, requiring significant innovations in design tools. Presenting our revised design tools is beyond the scope of this discussion, which is an entire discussion in its own right.

2.2 SOC FPAA Routing Fabric Characterization and Computation

From a classical FPGA approach, one considers the capability of the device to be in its components (CLBs, specialized blocks), and the routing fabric is simply a capability to interconnect these components. In such an approach, we want to minimize the effect of the routing fabric that from a circuit perspective is dead weight that can only degrade the circuit. That approach requires minimizing the amount of switches, each of which add resistance, as well as minimize the resulting capacitance of the routing. The routing infrastructure can effectively be modeled as a distributed RC line. The architecture choices look at the relationship of the resulting switch resistances, or



Component	Capacitance
On Switch	< 20fF
Local Line	160fF
C block Line	160fF
S block Line	38fF

Figure 9: Experimental measurements for characterizing the capacitances of the routing fabric. These FPAAs enable programming experiments that characterize the fundamental properties of the configurable fabric. We first measure the current-voltage relationship for a specific OTA device, shown in the inset, to exactly find the resulting Gm (0.1547μA/V) of the device. Then we use that exact OTA with the same programmed current to measure the time-constant of the step response (on a 1.2V dc for the 2.5V supply) for different (additive) routing combinations. In the presented measurement, we measure the resulting step responses, and from that we can linearly curve fit to the time constant after removing the effect of the steady state voltage. The resulting measurements give a measurement of the resulting routing capacitance, as well as enables, through the routing fabric, a range of tunable capacitor blocks. We summarize multiple measurement configurations for our values of routing capacitances.

other circuit uses of the FG switch devices, as a function of the number of CAB inputs and number of tracks, as well as the typical number of switches needed for a connection.

This FPAA structure enables directly characterizing the resulting capacitance; coupled with the resistance of an on-switch (5-10kΩ programmed at the maximum conductance point) we can directly predict delays along each of these lines. Figure

9 illustrates we can compile circuits to characterize precisely the behavior of these circuits, including load capacitance of the fabric itself. Every experiment is same voltage bias, so expect that p-n junction capacitances would be similar through this experiment. Precise measurement of routing capacitances enable tuning, through programming switches, for precise capacitances where needed for matching. Matching of capacitances and programmability of current sources by FG techniques dramatically reduces the effect of mismatch in small cell sizes.

But our approach further moves away from the classical FPGA approach, in a radical perspective that, because we can program FG devices to analog levels, our routing fabric is no longer dead weight, as we first hypothesized previously [49] and fully implemented in our SOC FPAA.

We begin by describing one aspect of our routing fabric used for computation tied with capability for rapid reconfigurability. Figure 10 shows additional routing structure enabling rapid reconfigurability in the FPAA fabric. Essential to analog structures, typically data flow, and want to configure the computation to optimally minimize the amount of intermediate data storage. Intermediate data storage often the largest power and complexity cost for a system development. We have developed rapid reconfigurability in the fabric such that we can change between programmed aspects in a single clock cycle or asynchronous request–acknowledge loop. SOC FPAA shift register control signals are controlled by locally routed signals in the fabric controlling the clock (CK) and data signals. Data stored in the FG fabric would be as optimal as data stored in an off-chip nonvolatile memory without the complexity of loading the resulting computation. We also see the first illustration of using the routing fabric elements, this time as a bank of parallel current sources, as well as a cascading transistor. One easily sees an Arbitrary Waveform Generator that could be compiled into the fabric; the circuit also becomes the non-volatile memory for the

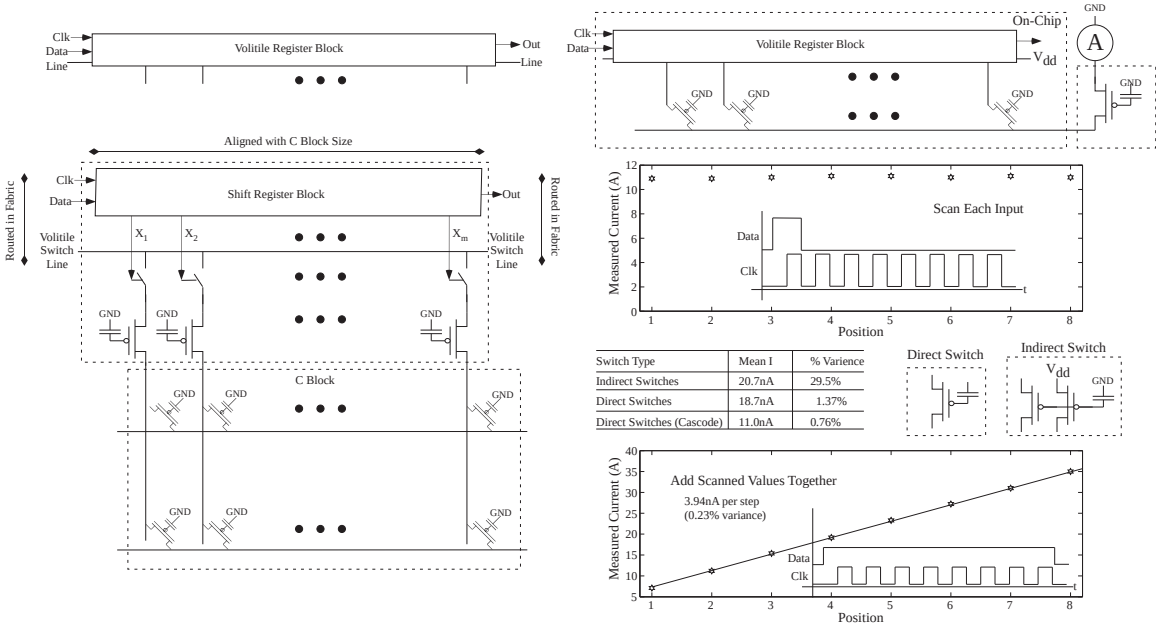


Figure 10: Additional aspects of our FPAA Routing Fabric. We include a set of T-gate based switch elements in the routing fabric to enable rapid reconfigurability. These switches are accessed through a shift register that enables rapid change of configuration on a single clock cycle; different lines of the resulting C block and/or local routing store the different configurations. We represent the resulting switches, resulting shift register, and switches connecting the block to the routing fabric as a single volatile routing block. We illustrated this capability utilizing routing elements programmed as precision current source elements, both using an input to the shift register input to scan through the individual signals, as well as using an input to the shift register to accumulate the resulting outputs through the individual signals. It is straight-forward to imagine a range of arbitrary waveform generation based on patterns stored in routing fabric. This measurement gives a metric of programming accuracy in operational mode. The accuracy for these switches ranged 0.2 to 0.76 percent for programmed subthreshold currents for uncorrected FG values; the resulting accuracy can be improved after such an initial measurement. Further, some switches in the routing fabric use only a single pFET transistor (Direct Switches), while some use two pFET transistors (Indirect Switches), where one device is used for computation and one device is used for programming. The Indirect switches show characteristically higher mismatch for uncorrected FG programming due to the threshold voltage mismatch of the two pFET devices. GND is signal GND; we bias the gate terminal for the FG devices at 0.6V.

function, eliminating outside memory and resulting complexity and energy requirements. The measurements show the accuracy of the FG transistor programming, either in the FG voltage or resulting channel current. We notice accuracy tighter than 1 percent in sub threshold, relating to less than 250μV variation.

The difference between directly programmed and indirectly programmed floating-gates is whether or not current measurements are made on the circuit transistor or

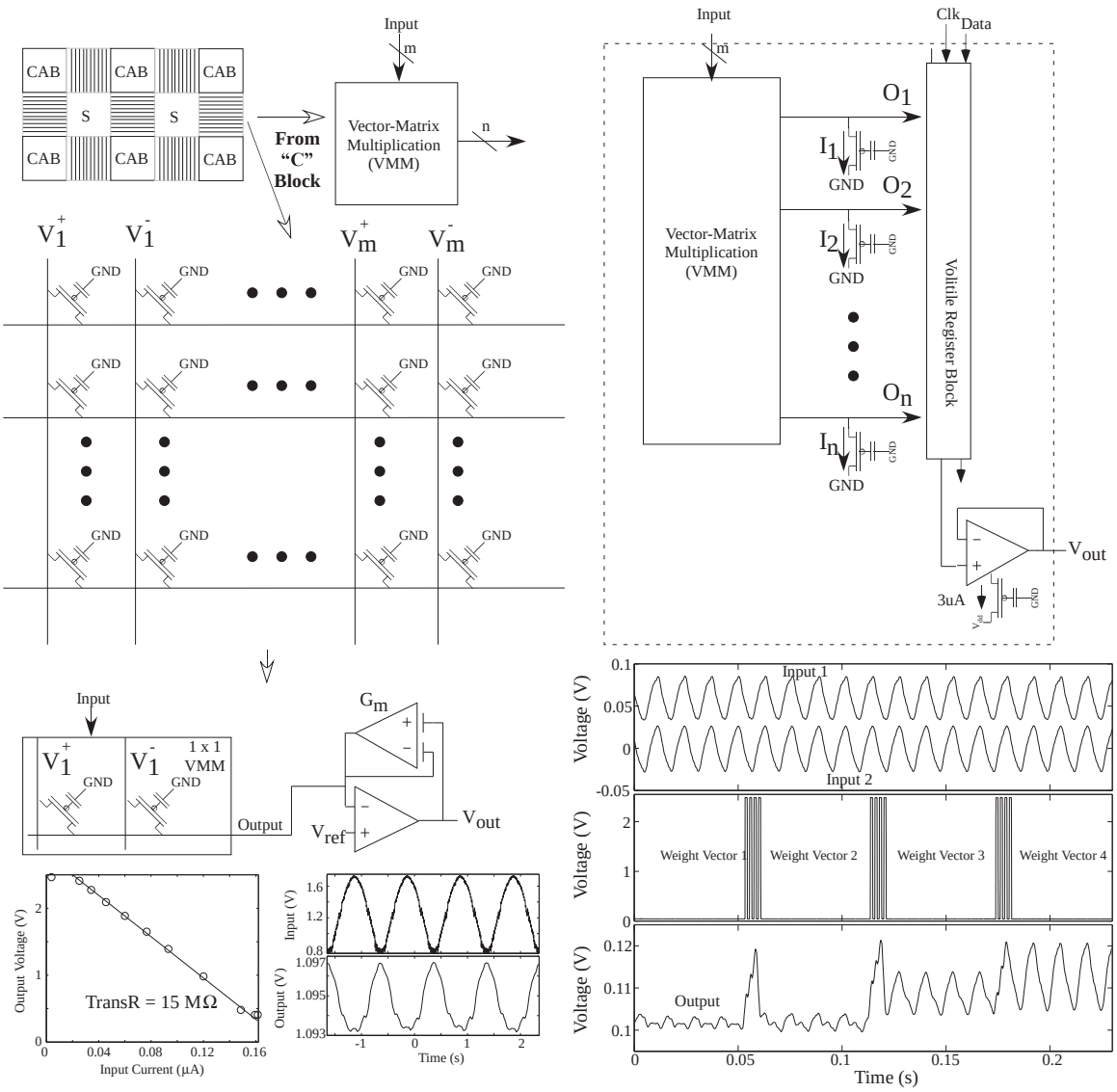


Figure 11: Vector-Matrix Multiplication (VMM) as a computational block instantiated in C Block routing fabric. The C Block forms a natural crossbar network typical for a VMM computation. We show the data for a single VMM element through routing fabric to illustrate the basic behavior; two pFET transistors are required for source-input 4-quadrant multiplication. We independently measure the resulting transresistance as 15MΩ. Further, we show an application of VMM integrated with the volatile switch register block to enable rapid (single-clock) switching between weight vectors.

the injection transistor during the programming algorithm. In the direct case, both the circuit and injection transistor are the same transistor. In the indirect case, they are two seperate transistors. The indirect FG device leads to a more efficient switch (fewer parasitics), but one must account for the V_{T0} mismatch between the two pFET devices. The direct FG device programs, measures, and computes through the same

device, eliminating any V_{T0} mismatch , requires additional transmission gates in the signal path for programming.

Computing Vector-Matrix Multiplication (VMM) solidifies the radical use of routing fabric as a computational element. Figure 11 shows implementation of a VMM in the routing fabric of our FPAA structure. We implement this functionality either in the C block or in the local CAB / CLB routing fabric, being that both structures are naturally crossbar arrays. Longer discussion on VMMs in early FPAA routing fabric is described elsewhere [50]. Further, we are effectively computing through our memory device, effectively the EEPROM storage of our values, directly in routing fabric, enabling the VMM computation; other approaches, including traditional FPGA approaches, require additional memory elsewhere from the resulting computations required. Further, we show integrated VMM and rapid reconfigurabiliity enabling switching between metrics in the FPAA architecture. This feature further enables data flow architectures to do a particular computation right when data arrives, reducing the need for short-term storage.

2.3 Representative Circuit and Signal Processing Components in the SOC FPAA

After considering the basic computation of the key components as well as the behavior and computation of the routing fabric, we move to looking at the behavior for some basic mixed-signal processing circuits compiled and experimentally measured in this system. The circuits illustrate some of the analog-digital co-design in these approaches.

Our first example is compiling two basic ADC devices in the routing fabric. Figure 12 shows circuit compilation at the analog–digital boundary through compilation of multiple forms of ADCs as an example of integration of the capabilities. Being able to compile an ADC, and the particularly needed ADC both allows for optimal power computation, heavy IP block reuse, as well as allowing the system lines to be blurred

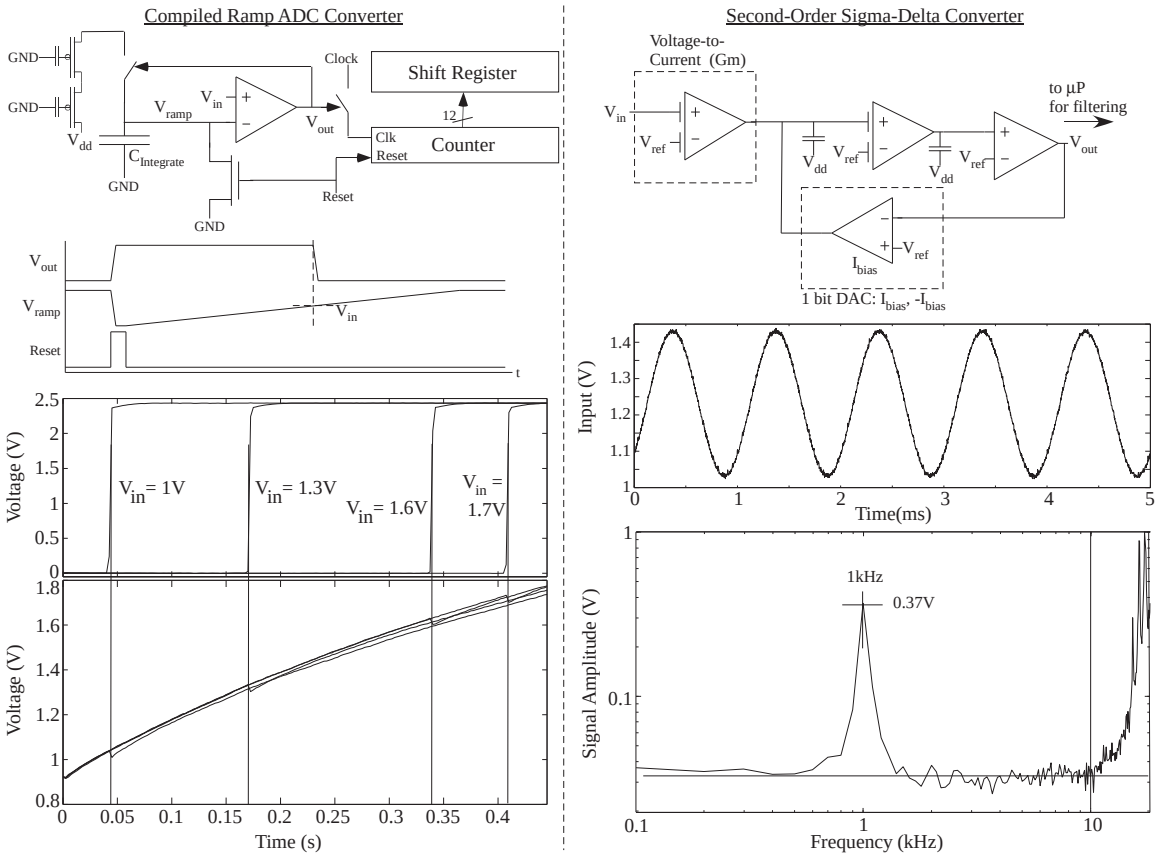


Figure 12: Two compiled Analog-to-Digital Converters (ADC) with experimental results. These approaches show the mixed signal structure compiled on our analog and digital enabled routing fabric. One example shows a compiled Ramp ADC converter. The second example shows a compiled second-order Sigma-Delta (Σ - Δ) converter.

between analog and digital for more effective approaches of classifying raw analog data. The design of the routing fabric was not a block of analog components and a block of digital components with hard-build data converters in between, but rather a mixed fabric to explicitly allow the lines to be blurred as the application requires.

Our second example is a basic FPAA classifier using a single Layer VMM + Winner-Take-All (WTA) circuit as a non-ADC conversion between analog and digital signals. Figure 13 shows a one-layer classifier approach based on the combination of a VMM and a k-winner Winner-Take-All (WTA) circuit [51], that elegantly compiles into routing fabric [52]. The one layer architecture can perform standard one-layer hyperplane classifiers, while also performing tasks considered impossible for

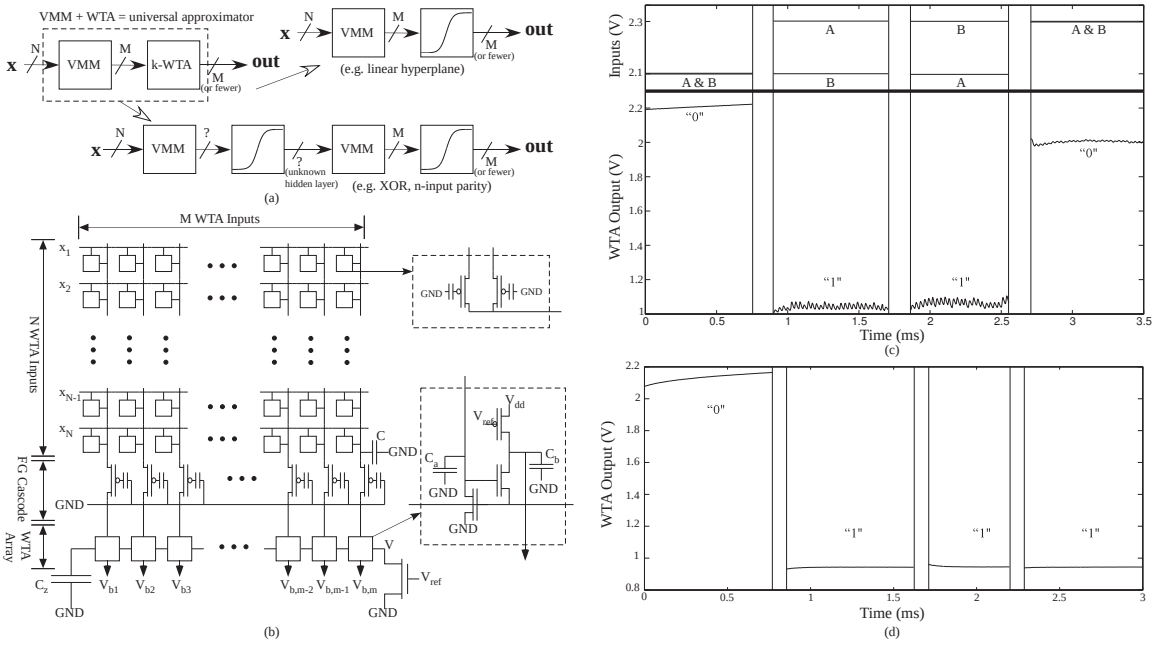


Figure 13: Instantiated FPAA classifier block based on a combination of VMM with a Winner-Take-All (WTA) block enabling a compiled one-layer universal approximator that efficiently compiles into an FPAA device. We show the circuit block for the one-layer VMM + WTA classifier block for N inputs and M outputs. The WTA circuit is operated as a single winner circuit, or as a k-WTA circuit, where upto K winners are possible if their metric is above a basic threshold, as originally described in [12]. We show an example of a single hyperplane classification, as well as an example of an n-input parity classification, experimentally verifying the universal approximator approach. (*need a block diagram of VMM +WTA type block)*

typical one-layer Neural Network architectures (i.e. XOR). Figure 13 shows experimental measurements for both of these cases, both an XOR function, as well as a linear approximator function. The result experimentally verifies that this one-layer VMM+WTA architecture, compiled on this RASP 3.0 FPAA structure is a universal approximator.

2.4 Representative System Application in the SOC FPAA

Following the example of signal processing circuits compiled in the SOC FPAA, we move in this section to discussing two representative applications compiled and measured on this SOC FPAA. The goal of this section is showing two possible application opportunities; we expect the wider range of applications for sound / acoustics / speech applications, image processing and vision sensors, robotics applications, and wireless

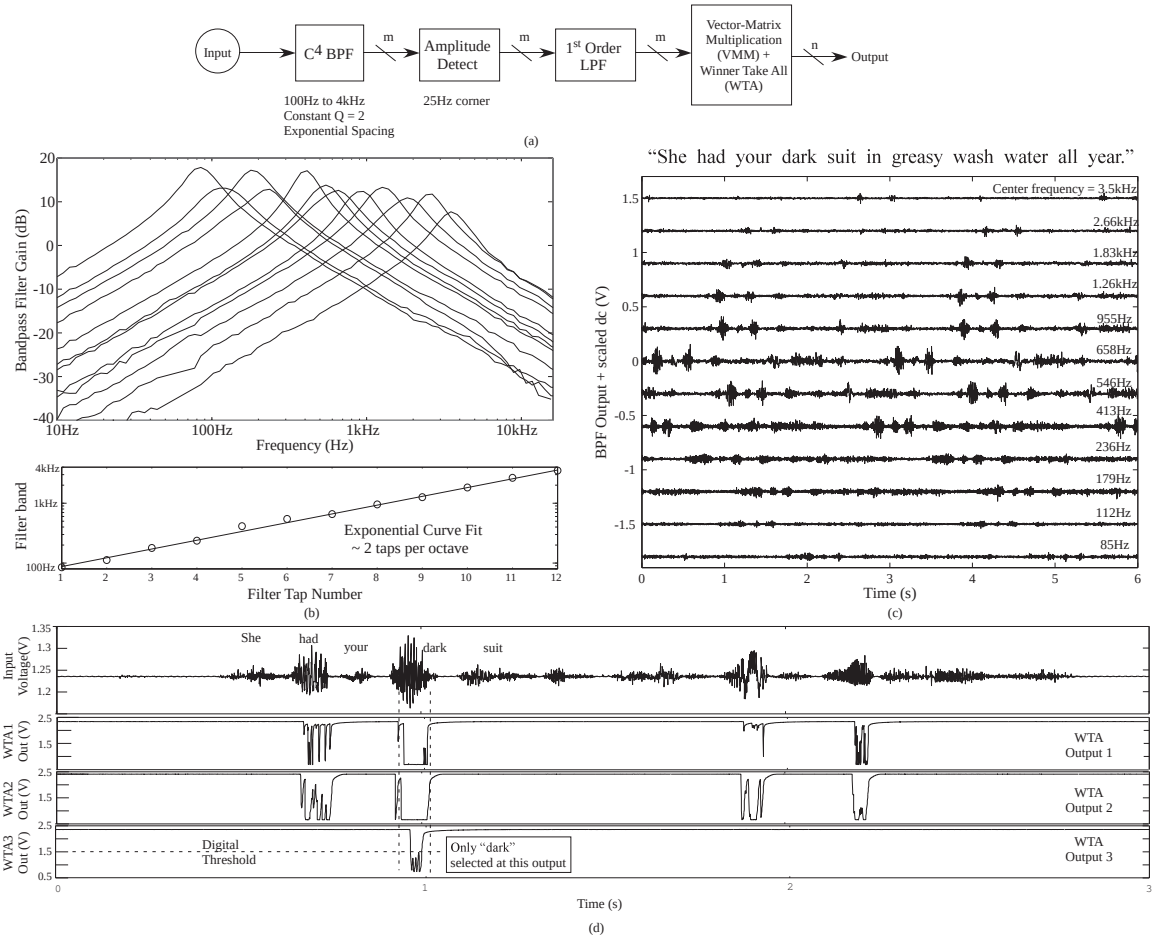


Figure 14: Analog auditory word classification application, compiled into the RASP 3.0, showing the experimental waveforms from the IC. (a) Block diagram for the classifier algorithm, in a similar representation used for our tool framework. (b) We use BPF center frequencies that are scaled evenly on a log-frequency scale between 100Hz and 4kHz with a constant Q filters (Q=2). (c) We show BandPass Filter (BPF) outputs and Amplitude Detection for a single phrase from the TIMIT database. (d) Classification of word and components for a TIMIT waveform. We use a k-WTA with three outputs to detect the ford "dark" in the resulting phrase.

communication applications, will be the subject of many future research where each system itself is a significant circuit and system design that can be experimentally implemented in this SOC FPAA IC.

We show an example application of auditory / speech classification looking at detecting a command word in a sentence. Figure 14 shows the first application example of an auditory classifier structure for a limited phrase, like a command word, that can be classified through features in the averaged signal spectrum. We start using a

continuous-time spectrum decomposition using a bank of constant Q filters, using a bank of amplitude detection and filtering operations, and then using a VMM+WTA classifier block to classify each of the resulting spectrum into simple symbols. In a more complex speech recognition system, we might have the spectrum correspond to phonemes or part of phonemes and build up the temporal representations using temporal classification (i.e. HMM classification) to word spot the resulting phonemes, syllables and words. In a simple command word application, we only need to distinguish between a few simple symbols, directly computed as a state machine on the MSP430 processor; a next level of computation, say as a simple Viterbi decoder, could be directly implemented on the MSP430 processor as well.

2.5 Summary Discussion and Comparisons

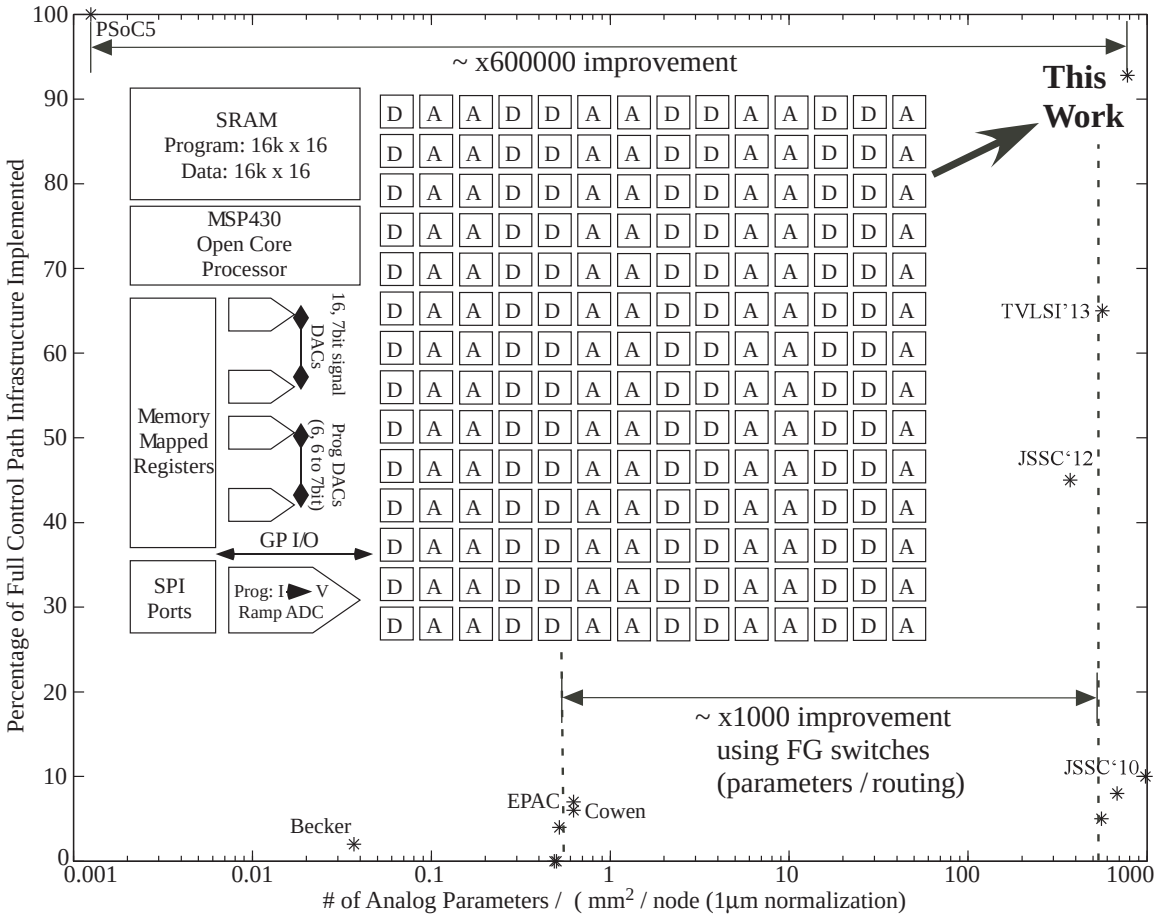
We presented an IC that integrates divergent concepts from previous multiple FPAA designs along with low-power digital computation and interface circuitry (i.e. DACs, ADCs). We showed through discussion and measured data that this unified structure enables a wide range of SoC computing options that can be optimized for a wide range of parameters, showing the most sophisticated FPAA capability built to date; we hope that the success of this IC inspires additional devices build in the near future. Figure 15 shows the table of parameters for the resulting SOC FPAA. Largest signal processing functions shown to date [37, 38, 65], where each are only taking a small percentage of the available IC.

Using data from generations of FPAA devices, built at GT and elsewhere, we plot various FPAA devices showing the (Percentage of Control Path implemented) vs. Analog Parameter Density. Figure 15 shows two key metrics for FPAA approaches based on a wide range of published FPAA devices [37,38,40,58–69]. We define analog parameter density as the number of programmable parameters per mm^2, normalized

to a 1μm CMOS node. Analog parameter density determines critically the IC computation complexity, particularly when using routing as computation. Figure 15 shows FG based FPAAs enable $\approx$ 1000 improvement in parameter density, enabling orders of magnitude potential computation on a single device; alternatives to FG devices require a DAC at every node or similar dynamic techniques.

One could imagine a second metric of maximum measured frequency, normalized to $1\mu m$ process. The result is very predictable and we find maximum analog frequency response being directly related to process technology; we have compared FPAA devices from 1μ CMOS to 40nm CMOS. Detailed discussions about frequency scaling will be discussed in a further discussion and demonstration of FPAA scaling and is beyond the scope of this discussion.

For an SOC FPAA device, we would want to maximize both metrics, so that we have a large number of programmable parameters, and resulting computation, as well as having the infrastructure to get data communicated to these processing devices. We develop the second metric to describe the the amount of control flow (mostly digital) relative to the amount of analog and digital data flow capability. Practically, the ability to get data to all of the processors can be a primary limitation for a range of application spaces, such as image processing, where data does not always arrive in the desired order for the computation. Recent RASP based FPAA designs [37,38] have started to focus on improving this second metric while not losing the analog parameter density efficiency. The presented SOC FPAA device maximizes both metrics, being nearly a factor of 500 improvement in area efficiency as typical of other analog FPAA devices, but with high utilization of the resulting computational resources; the closest high utilization structure (i,e, like PSoC5) is nearly a 300,000 factor improvement. This work resulted in the journal [126].



Parameter	Value	Parameter	Value
Number of CABs	98	Number of CLBs	98
On Chip μP	Open Source MSP430	μP clock frequency	0 - 50MHz
C block Line Cap.	160fF	S Block Line Cap.	160fF
V_{dd} (analog)	2.5V	V_{dd} (digital)	2.5V, 3.3V
V_{dd} Injection	6.0V	V_{dd} Tunneling	12V
Program Memory	16k x 16	Data Memory	16k x 16
CMOS Process	Standard 350nm	Die Size	12mm x 7mm
General Digital I/O	16 (in), 16(out)	SPI ports	5
General Analog I/O	125	Analog Parameters	359,014

Figure 15: Using data from generations of FPAA devices, built at GT and elsewhere, we plot various FPAA devices showing the Percentage of Control Path implemented versus Analog Parameter Density. Recent FPAA ICs, like the dynamically reconfigurable FPAA or FPAADD device begin to effectively maximize both parameters. Analog Parameter Density is the number of analog parameters per mm^2, normalized to a 1μm process (or analog parameter density). Analog parameters directly sets the complexity possible by the particular FPAA device. Further, we include a table of relevant parameters for our SOC FPAA device.

CHAPTER III

CAD SYNTHESIS TOOLS FOR HETEROGENEOUS SOCS

Field Programmable Gate Arrays (FPGAs) have evolved a lot over the past twenty years and have been rapidly adopted in industry, academia as well as by end users worldwide for a variety of applications. This has been possible due to powerful CAD tools for architectural exploration, CAD algorithm research and open source efforts [121, 148–150]. On the other hand, reconfigurable analog technologies have been lagging behind because of lack of such a rich toolset. Our endeavor is to build an integrated CAD tool framework that is a trailblazer for analog tool solutions for Field Programmable Analog Arrays (FPAAs) for shorter design turnaround times as shown in Fig. 16. Typically analog design is considered niche. We believe this will revolutionize analog design as we know it and enable a wider group of people to test analog, digital and mixed signal designs. Our tool suite **x2c** -Xcos to Chip, generates and integrates these tools to program and test an FPAA SoC. It enables fast and accurate co-simulations in both hardware and software. This methodology empowers the user to do seamless low power analog, digital and mixed signal design in a single environment from a graphical frontend to a switch list to target the SoC and test design.

In this chapter, I present a new synthesis, place and route tool called **vpr2swcs** which is a part of this tool suite and converts a netlist to an object file needed to program the FPAA SoC. These SoCs consist of a digital processor, an FPAA consisting of both digital and analog blocks in a reconfigurable switch fabric, DACs, ADCs and peripherals. This approach is novel as it enables, analog, digital and assembly codesign in the same environment. It also allows advanced synthesis by converting

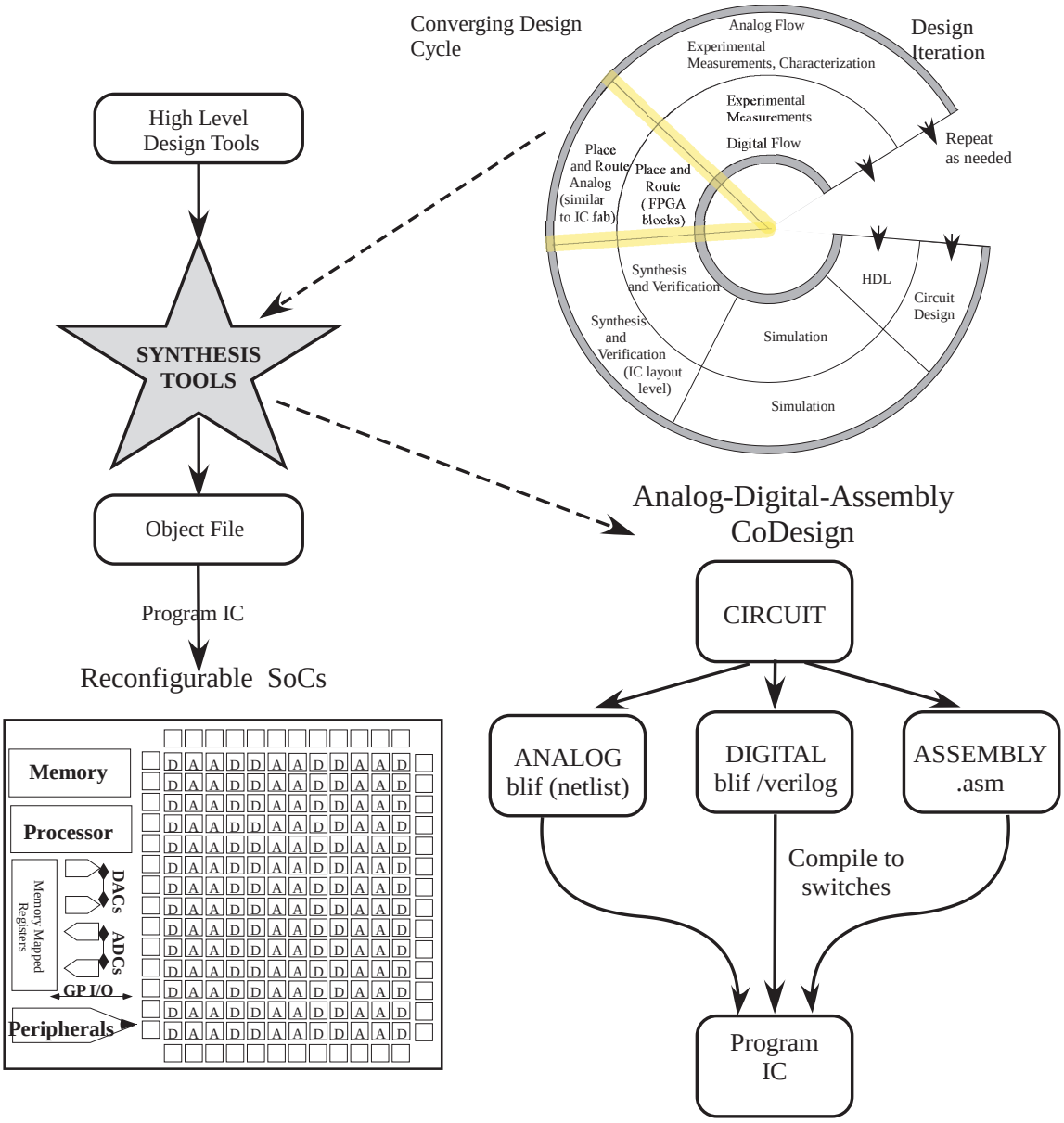


Figure 16: Typical iterative flow for CAD tools to design Integrated Circuits is illustrated here. Consider one such cycle, showing the design, simulation, synthesis, and testing for the analog and digital flows of the design. Reconfigurable ICs enable rapid turnaround time for testing. We have developed a software tool suite **x2c**, enabling design, simulation, verification, and experimental testing for configurable heterogeneous SoCs in a single environment.

a netlist to a programmed system on the hardware. Our approach embeds existing Virtual Place and Route (VPR) tool typically used for FPGAs in the modular python based tool *vpr2swcs* to develop a coherent user friendly design flow for heterogeneous

architectures. Though VPR treats analog blocks as a blackbox, *vpr2swcs* contains detailed descriptions of analog blocks. It effectively combines digital solutions from VPR along with custom techniques for analog circuits.

In this chapter, the focus is on the synthesis, place and route aspect of these heterogeneous SoCs and the tool called *vpr*2*swcs* that was developed to target FPAA SoCs. Previously a tool called Generic Reconfigurable Architecture Specification and Programming Environment or GRASPER was used for FPAA ICs which were based on a more crossbar routing structure [151]. This tool is a significant improvement over the previous tool. It can now be used for exploring heterogeneous architectures as well as for scaling systems. One particular novel piece of *vpr*2*swcs* is the ability to build useful computation out of routing resources. The synthesis and place and route of circuits containing Vector Matrix Multiplier (VMMs) [152] built out of floating-gate switches will also be highlighted in this chapter.

3.1 CAD Tools for Reconfigurable Hardware: Overview

Reconfigurable hardware for digital computation i.e. FPGAs are the norm these days. Similarly for analog solutions, we have FPAAs. The FPAA SoC is a floating-gate based, reconfigurable, fine-grained, mixed signal array with integrated processor, memory, and I/O peripherals. The hardware is tailored to explore and implement single chip solutions to mixed-signal problems in a codesign approach that flexibly varies how much of the problem is solved in software, digital or analog circuits. To facilitate the use of this hardware, **x2c** tool suite was created that starts with Scilab Xcos blocks and verilog , and runs all the way through synthesis, place and route, and programming of the hardware. In such a platform, analog components are embedded in a switch fabric which enable arbitrary connections between them. We use floating-gates(FG) as the switch element, as this adds the properties of non-volatility and compactness. Now, if we combine both analog and digital components into a FG

switch fabric, we can leverage the best of both worlds. Many FPAA structures have been built over the years such as [153–158].

There is a real need to develop an automated design flow for analog circuits, especially for non-analog designers who find the abstracted digital modeling much easier. Previously a tool called GRASPER [151] was developed to place and route for FPAAs. This tool for designed specifically for just complex analog blocks and didn't handle any digital circuits. It took a SPICE netlist as an input and generated switches to program the FPAA. The placement algorithm used was Modified Hyper-edge Coarsening (MHEC) order of cells and for routing labeling and backtracing techniques were used. However there were limitations while using the tool in terms of specifying a different architecture, digital elements, optimization and fixing placement of devices. The high level graphical interface for this tool was developed in MATLAB Simulink [159]. However, MATLAB not being an open-source software, it limited the outreach of this toolset. It was thus important to move to more flexible tool infrastructure that could support different architectures and enable quick testing and prototyping. Therefore, we decided to move towards using the open-source VTR/VPR tools as a code base. Though VTR/VPR is typically used for FPGAs, it supports heterogeneous architectures and is scalable for larger designs. However built for FPGAs it didn't meet all our requirements. Thus we developed *vpr*2*swcs* to specify analog circuits, utilize global and local routing resources for computation, as well as create an accurate switch map for a given FPAA chip. The new tool *vpr*2*swcs* is faster, more flexible, powerful, and easier to use than before, as well as relying only on a completely open source code base. This should enable a wider community to be able to use these tools for their FPAA solutions.

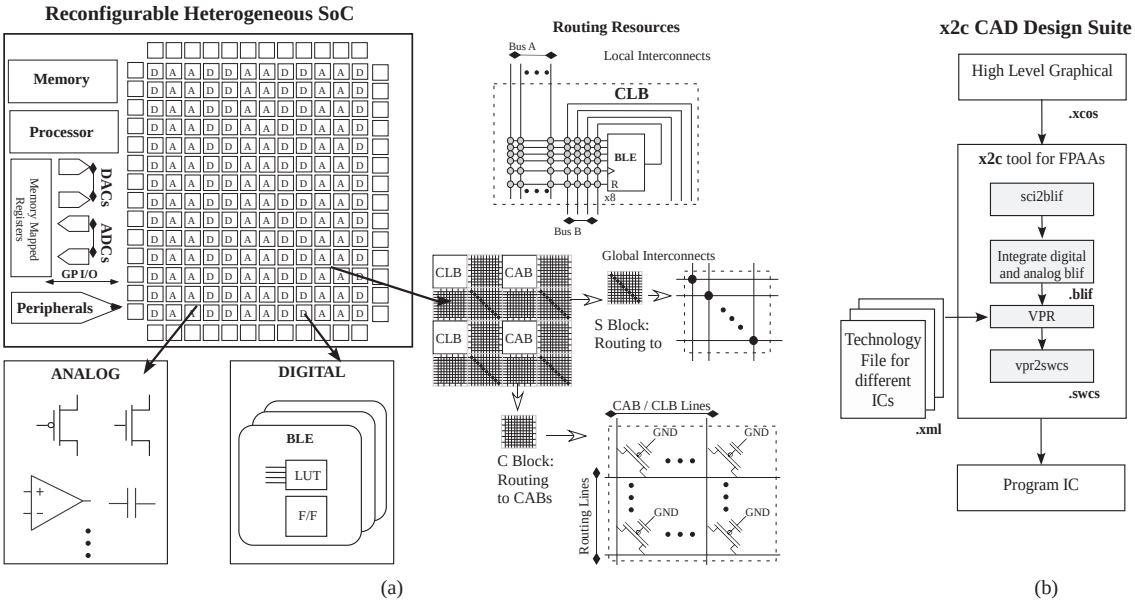


Figure 17: **x2c** Design Suite for FPAAs (a) Typical FPAA SoCs structure with Configurable Analog Blocks (CABs), digital Configurable Logic Blocks (CLBs), and global and local interconnects. (b) $x2c$ top-down tool design flow diagram. $x2c$ combines open-source software like Scilab Xcos, VPR/VTR and our open source software tools $sci2blif$ and $vpr2swcs$, to create a software suite to program and test FPAA SoCs.

3.2 x2c: Design Suite on FPAA SoCs

The latest FPAA SoCs are complex chips with an on-chip processor, data converters, FPAA fabric, routing resources and peripherals as shown in Fig. 16(a). $x2c$ or 'xcos to chip' is the open-source CAD tool software suite we have developed to build a tool to design mixed-mode circuits on FPAAs as shown in Fig. 17(b). The tools enables analog-digital-assembly level CoDesign as depicted in Fig. 16. The tool integrates different tools like $sci2blif$, VPR/VTR and $vpr2swcs$. It enables hardware software codesign on FPAA SoCs. An illustration of our SoCs are shown in Fig. 17(a). The whole software is setup in a Ubuntu Virtual-Machine, which makes setup and distribution very easy.

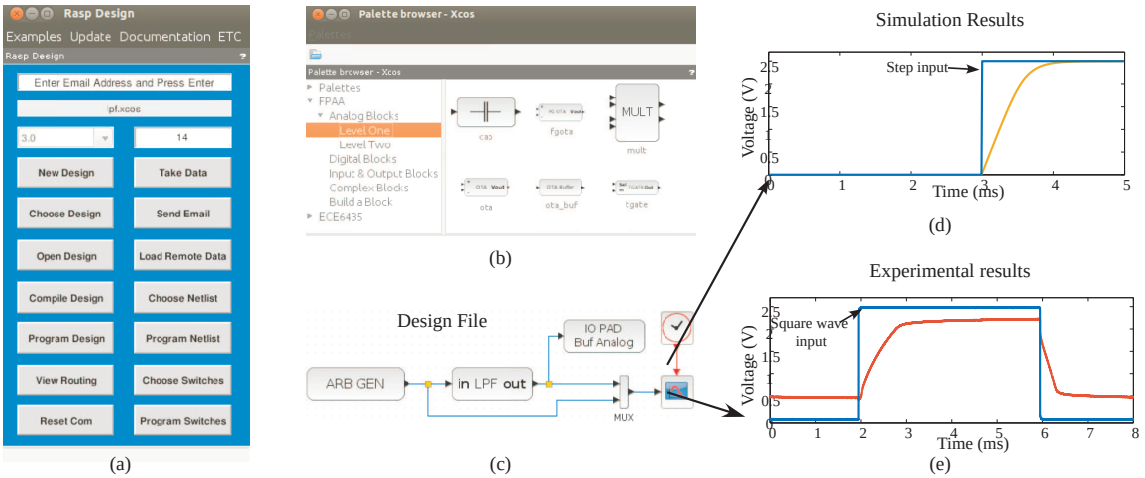


Figure 18: A system example of the tools to demonstrate Hardware-Sofware Codesign (a) FPAA Tools GUI which is the primary user-interface. It has various options the user can choose from including example designs. (b) Snapshot of the Xcos palette for FPAA blocks. There are four sections, namely the Analog, Digital, Input/Output and Complex Blocks. The Analog, Digital and I/O blocks consists of basic elements in different tiles of a chip. Complex blocks are pre-defined circuit blocks like C^4 Band-Pass Filters, peak detectors, VMMs etc whose parameters are programmable. (c) We show an XCOS diagram for a complex block, Low Pass Filter (LPF). The user can use the same file to simulate the block or synthesize the system on the SoC. (d) Simulation results for LPF block for a step function (e) Experimental Results for the LPF block for a square wave input.

3.2.1 *sci2blif*: Tool for XCOS to BLIF

The Xcos file has the information of the circuit and parameters to be compiled and programmed to the FPAA hardware. *sci2blif* is a tool we developed to convert block level information to a netlist in the BLIF format. This creates a netlist that the *vpr2swcs* tool can then use to place, route and program system on the chip. *sci2blif* can combine verilog, block level design, assembly code to create an integrated netlist that *vpr2swcs* then uses for further processing. FPAA Tools is the graphical interface we have developed for our tools as shown in 18 (a). The tool was designed in open-source software Scilab and using Scilab Xcos to make user-defined blocks and libraries. A detailed discussion of *sci2blif* will be presented in chapter 4.

In the following section, I will talk more about the tool *vpr2swcs* which helps

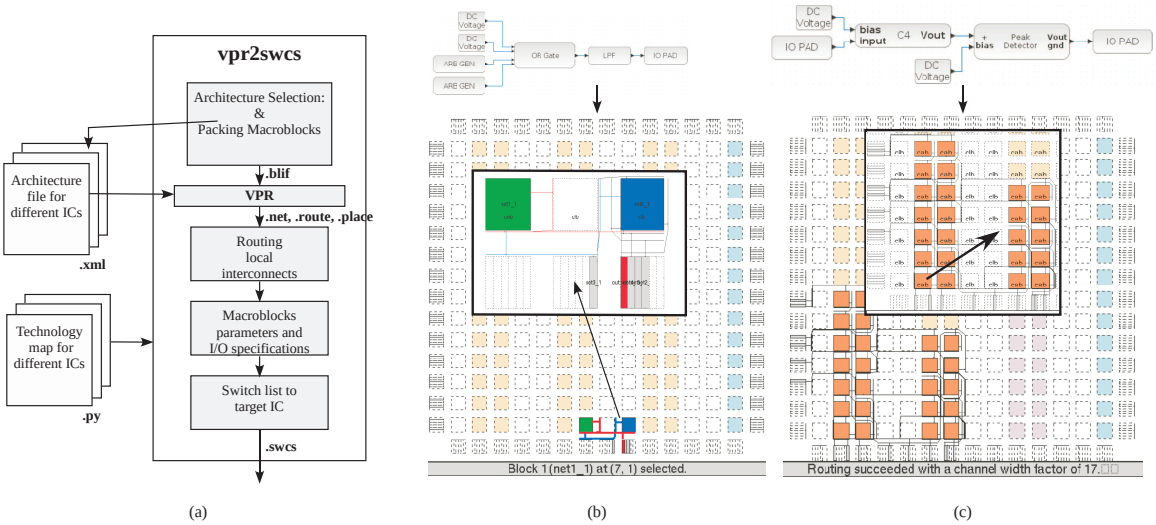


Figure 19: *vpr2swcs* synthesis tool flow (a) A flow diagram of *vpr2swcs* which converts the BLIF (Berkeley Logic Interchange Format) to a switch list. The tool uses VPR tools for place and global routing of elements. *vpr2swcs* takes both the architecture file and technology map as inputs along with the output of VPR. It handles the packing of macroblocks which are complex analog blocks, local interconnect routing as well specialized handling which is chip dependent. (b) Mixed-mode system example consisting of an OR gate and low-pass filter with different I/O blocks. Corresponding routing in VPR is shown. (c) A vectorized system using a C^4 band-pass filters and peak detector blocks is shown in Xcos and VPR. A vectorized block signifies multiple copies of the same block. The example here shows a block representing 16 such filter-banks.

target the FPAA SoCs.

3.3 vpr2swcs: Targeting Heterogenous SoCs

When building the tool set for the new generation FPAAs, it was our intention to build a software base that others can actively contribute to. Thus, we decided to use VPR as an open-source code base for our tools [121]. *vpr2swcs* then enables us to further define analog circuits which VPR just considers as a blackbox as well as add new features like reusing routing resources, building optimized analog blocks called macroblocks, switch architecture files, map switches for an FPAA SoC, define peripheral maps and generate switch list to target mixed signal FPAAs. The architecture files have been modified to accommodate special sub-circuits that are part of

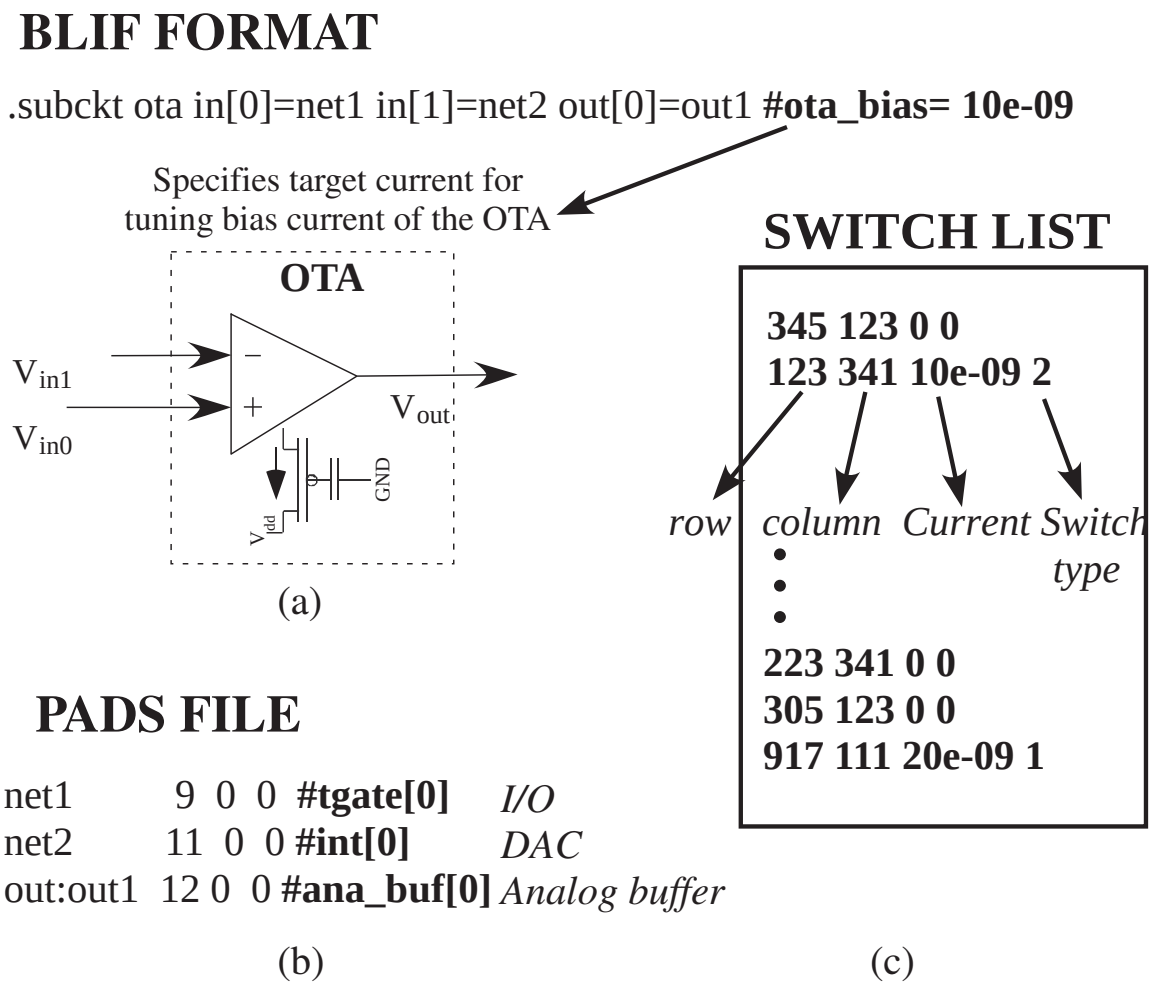


Figure 20: Configuration settings in $x2c$. One can specify configuration values in the BLIF file that we generate. This is then used by $vpr2swcs$ to generate a switch list accordingly. (a) BLIF format description of an Operational Transconductance Amplifier (OTA) with a programmbale bias current. $vpr2swcs$ parses the comment to correctly associate the bias current value with the OTA block, as well as handling the local interconnect. (b) Illustration of a typical pads file for the chip. Depending on the technology map of an SoC, different I/Os can be simple I/Os or DACs/ ADCs or buffered I/Os or just simple internal nets. $vpr2swcs$ enables I/O blocks to be configurable as well. (c) A typical switch list output that is used to target FPAA SoCs.

the analog CABs and aren't native to a typical FPGA. The circuits are expressed in the Berkeley Logic Interface Format(BLIF), which is essentially a netlist. The tool packs the analog/digital components into CABs/CLBs. Once packed, VPR places these depending on I/O pin placement and routes the global signals between them.

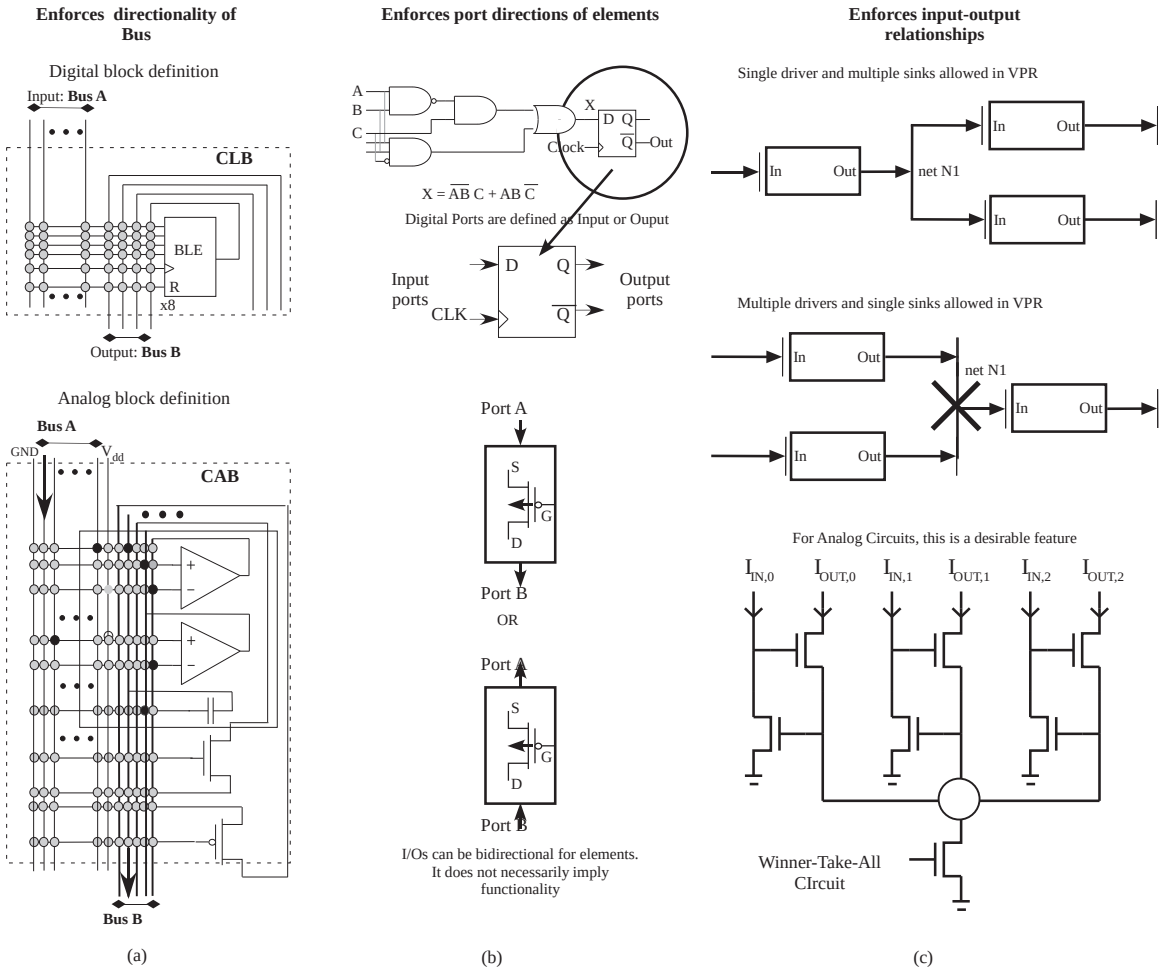


Figure 21: Challenges one faces when using VPR (a) Bi-directionality of local interconnects as well as I/Os of block elements is a desirable feature for analog circuits which VPR does not support. For digital elements the bus direction is specified but for analog CABs, one just considers the buses to be bidirectional. (b) The way ports of elements is defined is also different. In a flip-flop for example the input and output of a flip-flop is explicitly defined. But in the case of a transistor for example, the source and drain can be switched and either can be used as input or output. This isn't supported in VPR. Flexibility of defining ports is also a desirable feature. (c) Multiple drivers for a single net is also not supported in VPR. While this is a sanity check for digital circuits, functionality wise it is required for analog circuits. Here we show the example of a Winner-Take-All (WTA) circuit where we have multiple drivers and a sink which is a very useful analog circuit. This type of a circuit can't be specified in VPR.

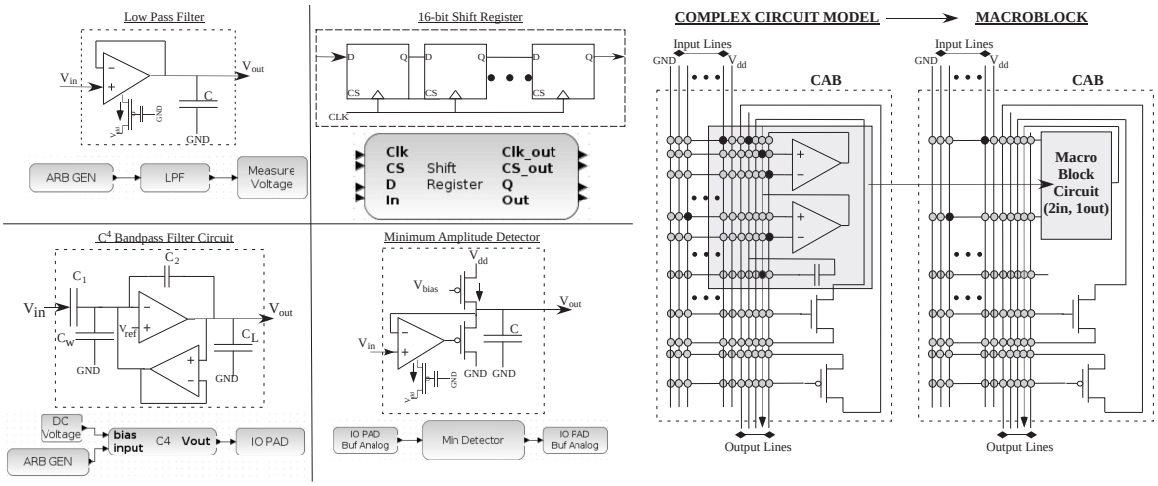
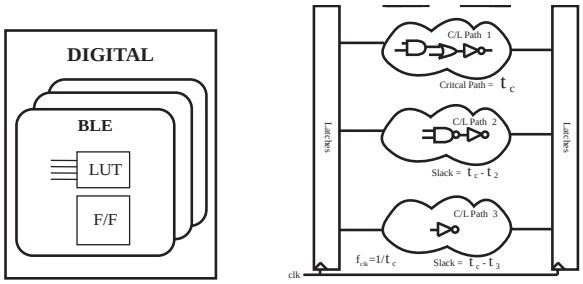


Figure 22: Macroblocks are complex blocks that are built from basic elements in the Analog and Digital Tiles. For simplicity, these have been encapsulated as a single block in the palette. We define these analog tiles as blackboxes in VPR such that all elements remain in a Analog/Digital Tile. We show a few examples: LPF, Minimum Amplitude Detector, C^4 Band-Pass Filter, and the Shift-Register Block.

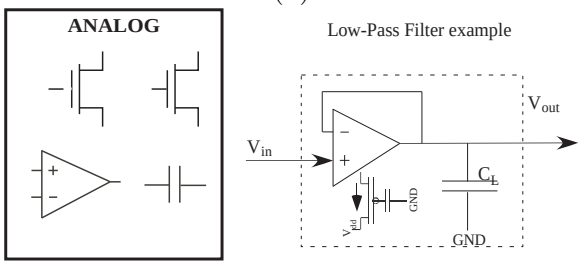
*vpr*2*swcs* then utilizes the global routing information from VPR and further builds the local interconnect switch map which is specialized along with special handling of I/Os to add to a switch list which is then used to target the SoC.

3.3.1 VPR

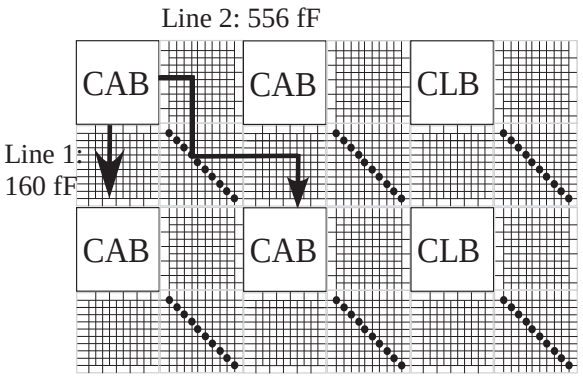
The VPR tool was designed to be a platform for simulation based FPGA place and route experiments. It was built to be an open-source academic platform for analyzing the efficacy of place and route algorithms in the mapping of benchmark circuits to FPGA architectures, the effects that varying the FPGA architecture has on the solution space, and the circuit performance of any routing solution. Being open source, algorithms, architectures, cost metrics, and benchmarks are easily swapped for large parametric and statistical experiments. At its heart, VPR is simply a wrapper for applying off-the-shelf placement and routing algorithms to the problem of implementing a target circuit graph out of some subgraph of a target architecture. In this sense, it can target a limitless variety of FPGA architectures. The VPR toolchain can be interrupted and any custom architecture graph could be inserted. However, the tool



(a)



(b)



(c)

Timing driven optimization:
Delay is calculated for the critical path.

As $Capcitance \downarrow, Delay \downarrow$

Delay, $\tau \propto Capacitance$

However,
$Noise \downarrow$, as $Capacitance \uparrow$

Thus, $SNR \uparrow$, as $Capacitance \uparrow$

Component	Capacitance
On Switch	< 20fF
Local Line	160fF
C block Line	160fF
S block Line	38fF

(d)

Figure 23: Optimization techniques for Heterogeneous Systems (a) Representation of Digital blocks. The main criteria for optimization is timing-driven and connection-driven. (b) One cannot always optimize for timing delay or capacitance for analog circuits as this can be a useful parameter. For example, consider the low-pass filter. Here, τ is a parameter that the user wants to modulate. (c) An example of different routing capacitances for path chosen. (d) A table of routing capacitance as observed on the FPAA chip fabric in $350nm$ CMOS.

does implement a parametric architectural graph generator. This generator provided a quick interface to building parametric FPGA architectures that were a subset of the architecture space it could target. That space was limited to Manhattan style architectures. The fabric was a linear array of tiles comprising complex logic blocks and global interconnect where the complex logic blocks contained the computational

devices (LUTs and FFs) and some reconfigurable wiring called the local interconnect.

Local interconnect is used to wire devices together that have been clustered together into these complex blocks, and then those groupings are wired together at a higher level using the global interconnect. In this manner, some level of hierarchy is applied to the global place and route problem.

Since we have built a lot of variations of reconfigurable chips, it is nice to have a sort of unified code base for utilizing these chips that leverages as much code reuse as possible. Therefore, we have adopted the Manhattan-style routing fabric which can then be easily targeted by VPRs architectural building language.

3.3.2 Challenges to make VPR work for Heterogeneous systems

The big challenge for making VPR work for heterogeneous ICs is defining Analog blocks and how to modify the basic rules of digital design for analog design place and route. We list some of these challenges in Fig. 21. One issue is having multiple drivers for a single net. While this is a sanity check for digital circuits, this doesn't quite work for analog circuits where this is fairly common. Take for example current-summation in a Winner-Take-All circuit as shown in Fig. 21 (c).

Another issue is not having bidirectional input/outputs. This is an issue when talking about local routing as well as for analog elements. This is illustrated in Fig. 21 (a-b). Take the example of a CMOS MOSFET, where the source and drain terminals are equivalent and interchangeable and one doesn't want to define it explicitly as input or output. In our system design all our I/Os were bidirectional mostly and handling that aspect in code was tricky. Now analog blocks are considered as blackboxes in the VPR tool. So, to circumvent this problem, I came up with idea of tailored macroblocks which I will define next. Now VPR just considers these macroblocks as new blackboxed element. We restrict the reuse of elements used to make the macroblock by restricting interconnect specification.

3.3.3 Macroblocks : Encapsulating complex circuits

Macroblocks is just a concept of encapsulating complex circuits using elements inside a single tile to create a single block. Some macroblocks are hard-coded to define some common circuit configurations. The connections within the CAB are pre-optimized and VPR now only handles global placement and routing. This helps us leverage some inherent circuit knowledge not apparent in the tool definition. We show some examples in Fig. 22. *vpr*2*swcs* utilizes parameters specified as comments in the BLIF file to now set parameters to configure these circuits.

3.3.4 vpr2swcs design flow

The *vpr*2*swcs* design flow is illustrated in Fig. 19. We will detail some key aspects below.

Architecture selection: The tool initially parses the BLIF file to choose the architecture file and configuration settings to use. Also any configuration settings to be passed to VPR are set.

Packing Macroblocks: The tool determines the packing of complex macroblocks in CABs/CLBs along with specifying the parameters related to these. Since these blocks are configurable/programmable in nature, the tools needs to handle setting the appropriate parameters correctly. To this end, we have modified the BLIF format to include comments after a sub circuit description that enables one to configure the circuit. This is illustrated in Fig. 20(a)where the bias current of an Operational Transconductance Amplifier is specified along with its model definition. Even for specialized macroblocks, it helps set the configuration of the block.

Routing local interconnects: Though VPR handles the placement of elements inside the CABs/CLBs, *vpr*2*swcs* tool handles local interconnect routing for CAB elements. This is because the interconnect defined on chip is highly complex and bidirectional. This feature is customizable and hence can be swapped depending on

the technology of the chip. Hence, much like the architecture file that VPR uses, a detailed chip dependent mapping of local interconnects can be made easily. This helps make the system very flexible and increases the degree of freedom while using buses. Also, for the purpose of maintaining hierarchy with global interconnects and the function of bi-directionality, we specified a bus as inputs to CABs as both input and output, even though the switches for both would map to the same bus.

Handling I/Os: Since our system is a mixed-signal system, we have many different options for I/Os. For each I/O pad on the chip there are multiple options like it being an analog buffered I/O, digital buffered I/O or just an unbuffered I/O. Besides these, I/O lines can also be re-routed back into the fabric or are connected to the DAC/ADC peripheral blocks. To handle such a complex set of conditions we define a single I/O block as multiple blocks each with a unique property which is again specified in the pads file but as a comment. Our tool *vpr*2*swcs* then parses this in the pads file and generates the correct switch configuration for that operation. The pads file configuration is illustrated in Fig. 20(b)

Parameters for switches: Various complex block or elements have some special parameters that need to be set. For example in digital LUTs, the conditions that need to be satisfied or in the case of OTAs, the bias condition for a floating-gate transistor as shown in Fig. 20(c). The tool helps to handle these and specify different conditions.

Generating switch lists: Our configurable architecture supports switches in three modes namely, ON, OFF or intermediate levels. This implies that the floating-gate switches can be used as ON/OFF switches as well as targeted to a specific current value.

3.3.5 Efficiency question for routing

For digital circuits few factors are considered while routing namely timing driven routing or custom routing. For timing-driven routing, the capacitive load is a major factor considered. Delay,

$$\tau \propto Capacitance \tag{1}$$

In synchronous digital design the delay of the critical path is used to set the clock frequency in order to guarantee that the propagation delays of all paths will satisfy the setup and hold times of the latches used to synchronize the data. For analog circuits however that is not always the criteria one might consider. For analog circuits one must consider parameters like delay, noise, Signal-to-Noise-ratio (SNR) etc. while designing. Now,

$$Noise \propto 1/Capacitance \tag{2}$$

SNR of the system is lower if we decrease capacitance.

$$SNR \propto Capacitance \tag{3}$$

Hence there needs to be trade-off while optimizing a circuit rather than just decreasing capacitance. This is another challenge which we handle in a more custom manner right now but also hope to integrate into the tools. We illustrate this issue in Fig. 23.

3.4 Routing resources for computation

VMMs are very efficient blocks for computation and can be used in image and speech processing algorithms [160, 162]. Unused Routing resources can be leveraged to build VMMs [163]. We were able to leverage local routing interconnects to build VMMs as shown in Fig. 24(a)-(b). This was done by manipulating the architectural file as well as the technology map specified for a chip to block the local interconnect. *vpr*2*swcs* can then selectively block certain columns in an architecture as VMM blocks. A high

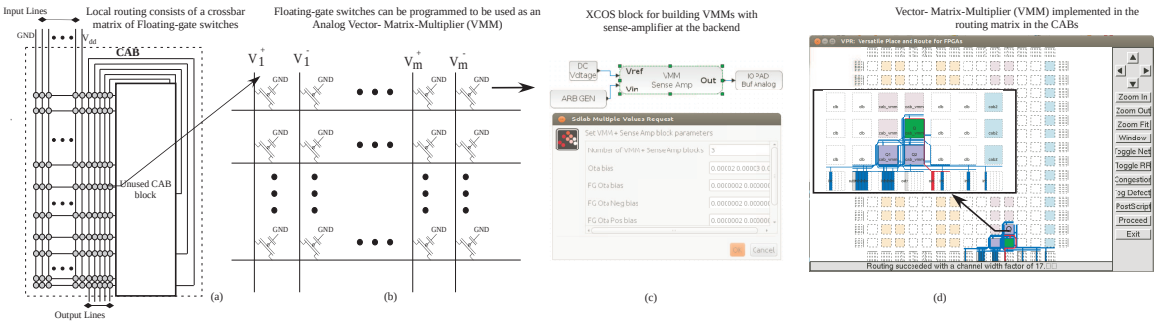


Figure 24: A novel feature of this hardware/software system is the ability to build useful computation out of routing resources. The synthesis, place, and routing of circuits containing VMMs built out of floating gate switches is highlighted here. (a) Local interconnect routing resources inside an Analog CAB are utilized to build VMMs. The size of basic VMM block depends on the size of the interconnect. Multiple VMM blocks can then be tiled to build bigger mXn VMMs. (b) Illustration of how a crossbar switch matrix can be used to build VMMs. (c) Block Diagram and parameters for a VMM block that can be set by the user. (d) When using VMMs, $vpr2swcs$ chooses a new architecture file and technology map that isolates some columns of the FPAA for use as VMMs. Thus, one can build these structures while still building other designs on the chip. We show the routing view for this new configuration type.

level Xcos block was developed as seen in Fig. 24(c) which could then be compiled on to chip as shown in the VPR routing output in Fig. 24(d).

3.5 System Example: Speech Classifier

I now present a system example using the circuit blocks we have presented so far. Let us look at the mixed-mode example of a speech classifier. the use of VMMs as a computing element is demonstrated here. The speech classifier circuit detects speech in an input signal. It is a good example of how we can encapsulate a large system into a simple block as shown in Fig. 26(a)-(b). We combine 12 filterbanks for speech over different bandwidths which is then followed by a 12x4 VMM and $4-input$ WTA processing stage and a digital shift register block. As one can see in Fig. 26(c) routing is much more complex and spread out. We show experimental data in Fig. 26(d) as recorded for the speech classifier synthesized on chip .

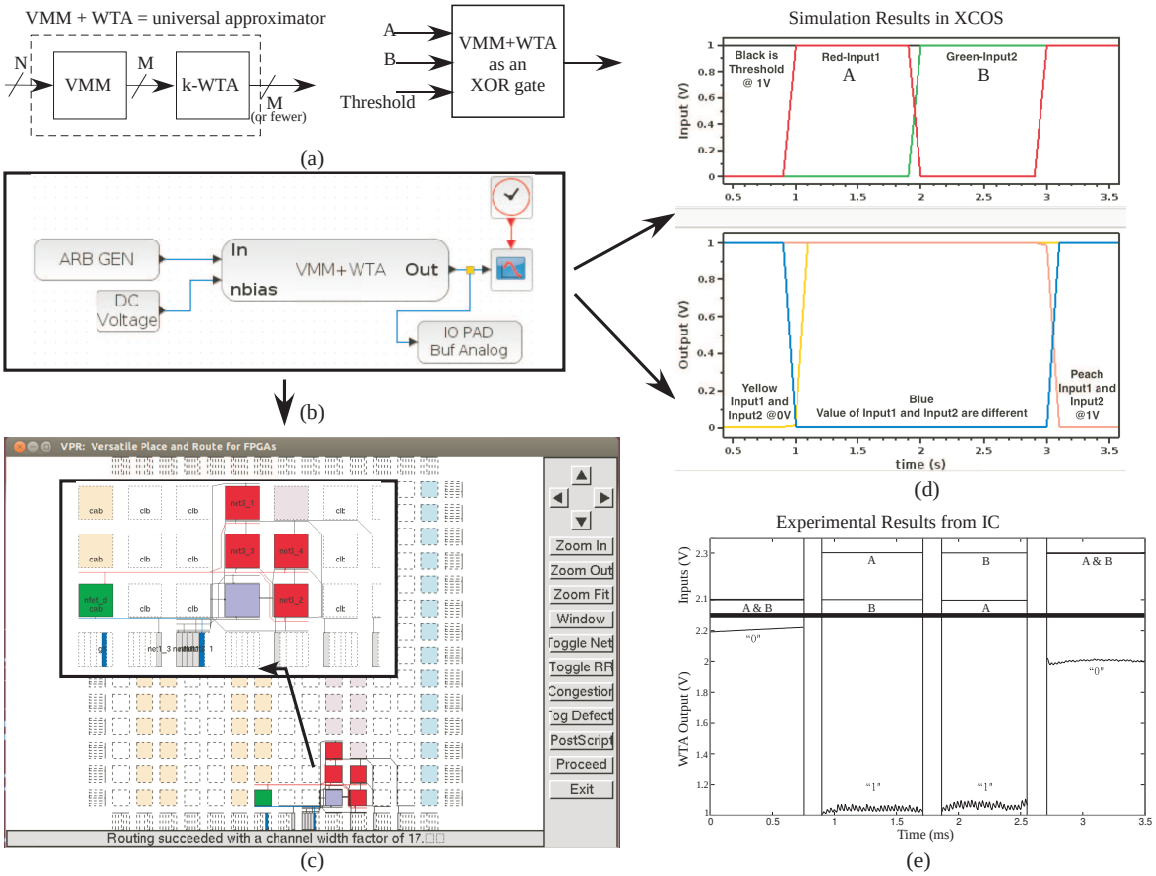


Figure 25: VMM+WTA or Universal Approximator system example: (a)The VMM+WTA has been shown to act as a universal approximator [160]. Here we present the circuit as a two-input Exclusive OR (EXOR) gate. The VMM matrix so chosen is a 3X3 matrix as shown in [160]. (b) The XCOS block diagram of the system. This Xcos file is used for simulation as well as experimental testing. (c) Routing as viewed in the VPR tool. (d) Simulation results of two-input EXOR gate. (e) Experimental results of two-input EXOR as recorded from the FPAA.

3.6 Conclusions and Future Directions

We presented a new synthesis tool *vpr2swcs* which is an open-source tool environment for mixed signal design using FPAA SoCs. The tool enables rapid prototyping on FPAA SoCs in the same integrated design tool framework. The toolset is open source (after publication) framework to target and explore of new FPAA architectures and encourage collaboration. We believe it will empower a wider community to do analog and digital system design. We have demonstrated some mixed signal design examples. There are no benchmarks specified for this tool, simply because such benchmarks are

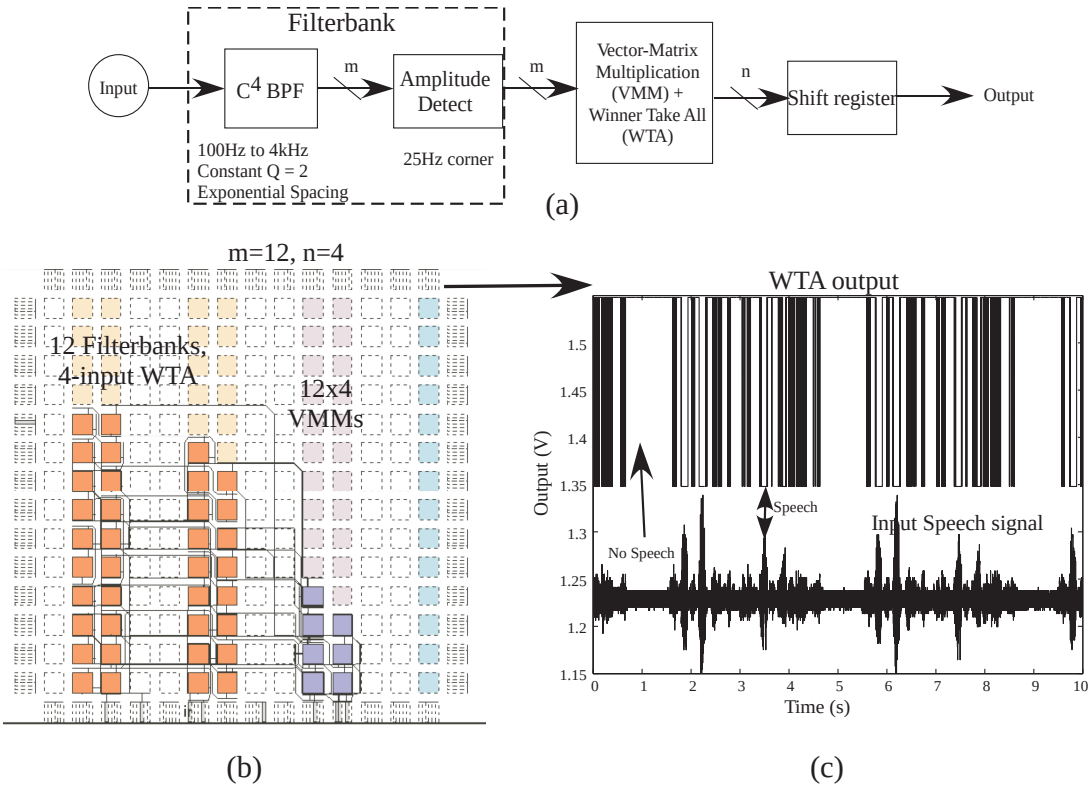


Figure 26: Speech Classifier System example: We demonstrate a classifier which can detect speech in a signal. (a) Block Diagram of a speech classifier system. It consists of a vectorized filterbank frontend that isolates frequencies in the speech spectrum. The signals are then input into a VMM+WTA circuit, which processes the signal to determine if speech was detected. (b) Routing the system using *vpr*2*swcs* for a speech system with twelve filterbank, 12X4 VMM and a four input WTA. (c) Experimental Data recorded from the FPAA for the speech classifier.

yet to be set for reconfigurable mixed signal systems. This work is an initial step in that direction.

CHAPTER IV

HARDWARE SOFTWARE CODESIGN

We present a unified tool framework for Analog–Digital Hardware-Software CoDesign, enabling the user to manipulate design choices (i.e. power, area) involving mixed-signal computation and signal processing. Digital Hardware-Software CoDesign is an established, although unsolved, discipline (e.g. [115]); incorporating analog computation and signal processing adds a new dimension to codesign. One typically assumes all computation is done in programmable digital hardware. The wide demonstration of programmable and configurable analog signal processing and computation [71] opens up an additional range of design choices, but requires user friendly design tools to enable system design without requiring understanding of analog circuit components.

We present our tool framework integrates a high-level design environment built in Scilab and Xcos (an open-source platform for MATLAB and Simulink, respectively), with a compilation tool, **x2c**, from the design environment to configurable and programmable configurable hardware. Figure 27 illustrates that although going from an application on a mixed-mode computing system may seem intractable, going through our Xcos tool framework compiled down to the system through **x2c** provides a potential method of solution. The approach is focused to enable system designers to integrate useful systems, while still enabling circuit experts to continue to develop creative and reusable designs within the same tool flow. Our example mixed-signal processing environment uses Large-Scale Field Programmable Analog Arrays (FPAA).

This open-source tool platform integrates existing open-source tools with software

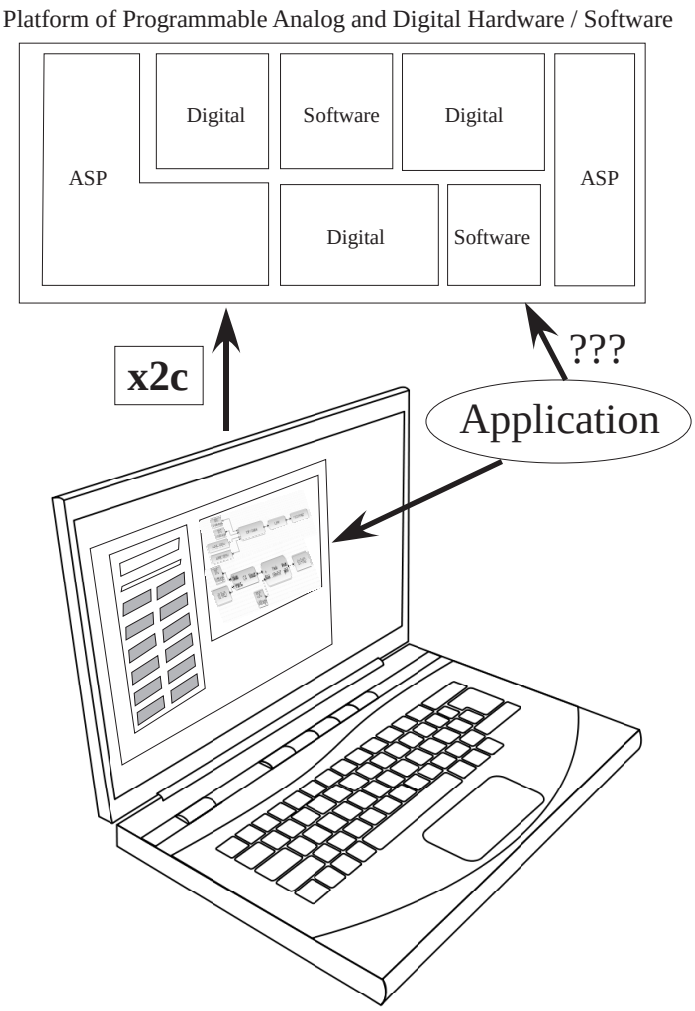


Figure 27: The translation from an application to a heterogeneous set of digital hardware + software resources is a known field of study; The translation from an application to a heterogeneous set of analog and digital hardware + software resources is question that is barely even considered. The focus of this chapter is to describe a set of software tools to encapsulate a range of potential application solutions, written in SciLab / Xcos, that enable a range of system design choices to be investigated by the designer. These tools enable high-level simulation as well as enable compilation to physical hardware through a tool **x2c**. The toolset will be publicly available upon publication.

we have developed to build an integrated environment to simulate, and experimentally test designs on the FPAA SoCs. Section II overviews the Analog–Digital design tool. Section III describes tool integration with an experimental FPAA platform. Section IV describes our methodology for implementing the toolset, including the approach for macromodel system simulation corresponding to measurements, and the approach of translating from the Xcos description to net list descriptions for hardware compilation.

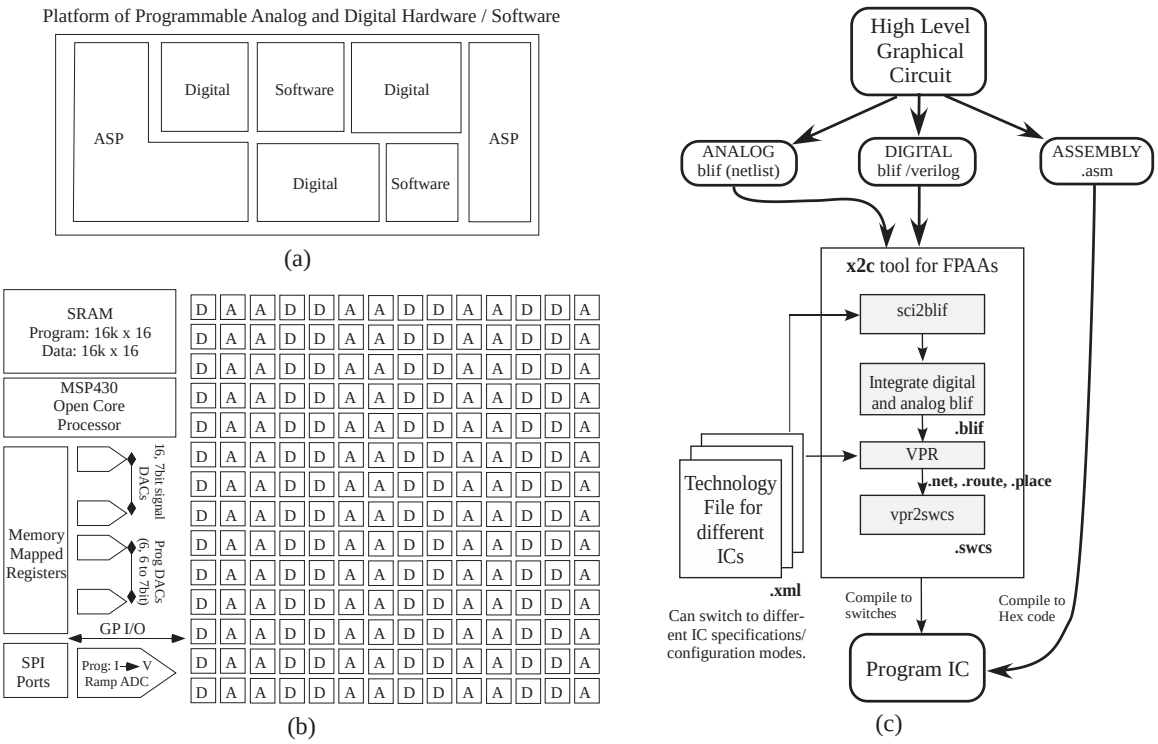


Figure 28: Tool Design Overview to handle programming this mixed platform of programmable analog and digital hardware and software, such as the block in (a). For this discussion, we will utilize large-scale Field Programmable Analog Arrays (FPAA), such as a recent FPAA shown in (b), for our starting discussion. These approaches are not limited to this particular system, and could be any particular system. In (c), we show an overview of our Top-down design tool flow for targeting such an FPAA device. The graphical high level tool uses a palette for available blocks that compile down to a combination of digital and analog hardware blocks, as well as software blocks on the resulting processor. The tool framework $x2c$, converts Xcos design to switches to program the SoC as well as for software simulation. $x2c$ combines open-source software like Scilab Xcos, VPR/VTR and our custom software $sci2blif$ and $vpr2swcs$ to create a software suite to program and test FPAA SoCs.

Section V describes some larger FPAA system examples. Section VI summarizes our chapter, as well as discusses strategies for Analog–Digital Co-Design. The toolset will be publicly available upon publication of this chapter.

4.1 Analog–Digital Design Tool Overview

Figure 28 shows our toolset for the translation from an application to a heterogeneous set of analog and digital hardware + software resources, such as the representative case in Fig. 28a, as well as our specific FPAA IC in Fig. 28b.

Figure 28c shows the block diagram of the resulting tool flow used for our infrastructure, used from IC experimental results as well as to simulate a given circuit before testing on FPAA hardware. Xcos system is built to enable macro model simulation of the resulting physical system. **x2c** converts Xcos design to switches to program the hardware system, made up of $sci2blif$ that converts Xcos to modified BLIF (Berkeley Logic Interface Format), and $vpr2swcs$ that converts BLIF to a programmable switch list as code around modified open-source Virtual Place and Route(VPR) tool [121], a tool originally designed for basic FPGA place and route algorithms. The particular system to be targeted, in this case a particular FPAA device, is defined by its resulting technology file for **x2c** tool use. A detailed discussion of $vpr2swcs$ place and route tools will be published elsewhere as it is beyond the scope and length of this chapter.

Figure 29 shows the graphical interface and the palette/library for different blocks in the tool. The tool is encapsulated in a single, open-source Ubuntu Virtual-Machine, with a single desktop button to launch the entire scilab tool framework. Xcos gives the user the ability to create, model, and simulate analog and digital designs. The Xcos editor is standard blocks that are compartmentalized into classes or palettes that range from mathematical operations to digital signal processing. The editor allows the internal simulator to utilize the functionality of each block to compute the final answer. Our tool structure took advantage of user-defined blocks and palettes that can interact with Scilab inherent blocks.

Our Xcos [125] tool uses user-defined blocks and libraries. When the user opens the Xcos editor, a palette browser is displayed, as shown in 29(b). The browser lists Scilab's collection of palettes as well as user defined palettes. One selects from a palette of available blocks to build the resulting system, which can be composed of a mixture of analog (BLIF), digital (verilog), and software (assembly language) components. The Palette for FPAA Tools contains sub-palettes for blocks that are

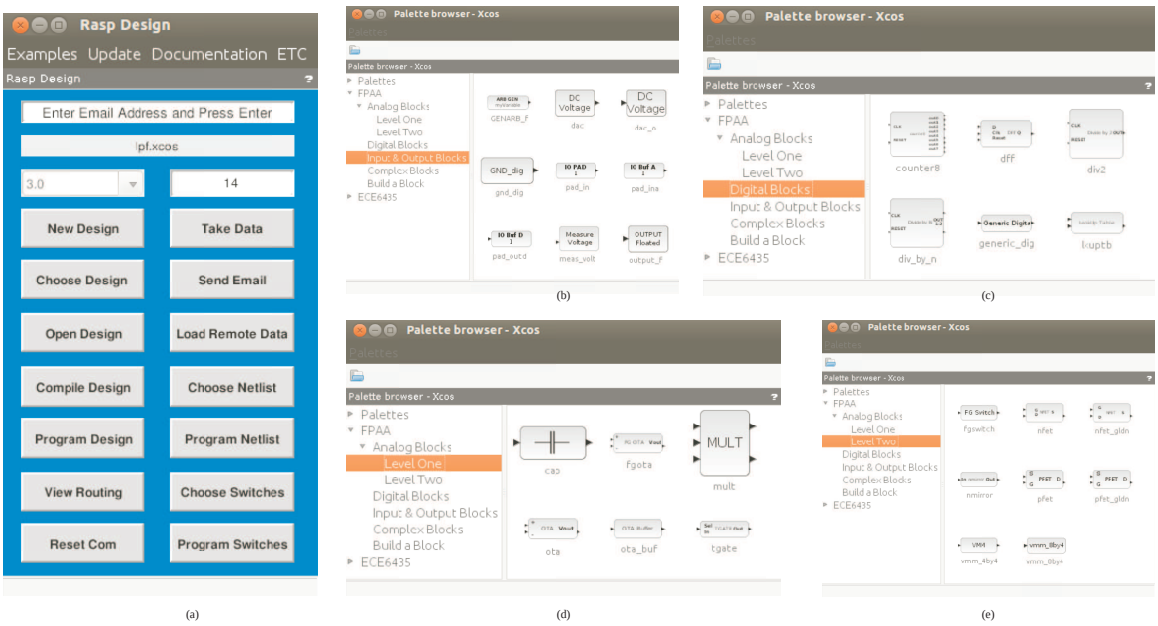


Figure 29: Graphical Tool Interface for $x2c$. (a) The user chooses basic design options through the FPAA Tools GUI, which starts running when the Scilab tools are started in the distributed Ubuntu Virtual Machine (VM). (b) Snapshot of the Xcos palette for FPAA blocks for I/O blocks. There are four sections, namely the Analog, Digital, Input/Output and Complex Blocks; the Analog, Digital and I/O blocks consists of basic elements in different tiles of a chip. Complex blocks are pre-defined circuit blocks called 'macroblocks' which we introduced in the previous chapter. (c) Snapshot of the Xcos palette for FPAA blocks for digital blocks. The user sets parameters for simulation or for compiling into IC. (d) Snapshot of the Xcos palette for FPAA blocks for Level 1 analog blocks. (e) Snapshot of the Xcos palette for FPAA blocks for Level 2 analog blocks.

classified as analog, digital, inputs/outputs, and complex blocks. The Input/Output palette contains typical circuits like DACs, Arbitrary Waveform Generator, IO pads, and Voltage Measurement. The digital palette contains typical circuits like a D-F/F and a clock divider. A few examples of complex blocks are a LPF, a Sigma-Delta ADC, and a VMM+WTA .

The blocks and their information are stored in a Scilab data structure that can be accessed in two different files, a block interfacing function and a block computational function. The interfacing function sets up the fields of the dialog box associated with each block to retrieve user parameters and set default values, as well as defines the size of the block and the number of inputs and outputs. Consistency checks to let the user know if the values they entered into the dialog box are valid implementation.

Each block has two specific files that dictate its appearance and its performance within Xcos. The computational function encourages model customization, while the interfacing function supports built-in data checking, variable inputs/outputs, and default parameters. The interfacing and computational functions are heavily coupled by the Scilab structure for a block. Thus, the parameters retrieved from a dialog box displayed to users are accessible to compute the output of blocks during simulation.

We follow previous representation [124] of analog blocks into Level 1 and Level 2 blocks. Level 1 blocks abstract away intricacies for system designers, such as the inputs and outputs are vectorized, voltage-mode signals. Level 2 blocks allow for general circuit design, typically at representation of CAB elements (i.e. caps, FG OTAs, and Tgates). Each block uses vector signals and resulting vector-based block computation, where each block may represent potentially N virtual blocks (or more) to either be compiled to silicon or simulated. The lines/links that connect the blocks together are essentially layered buses. Each link, and resulting block element, allows for vectorized signals consistent with level = 1 definition [124], empowering the user to develop systems with just the necessary number of blocks.

4.2 Integrating Analog–Digital Design Tool with an FPAA Platform

Figure 39 shows further details of the structure and testing of the FPAA devices. When one thinks of testing a mixed mode IC, one does not visualize, a full system IC requiring only simple interfacing to the outside world through USB or say SPI ports, appearing to be a standard peripheral to a typical device. Figure 39 shows a typical FPAA block diagram that enables both analog and digital components in its manhattan routing fabric, and most approaches for FPAA devices can be fit into this resulting framework [153–158]. The blocks with simply digital components are called in Computational Logic Blocks (CLB), and the blocks with analog and digital components are called Computational Analog Blocks (CAB). Figure 28 shows the block diagram

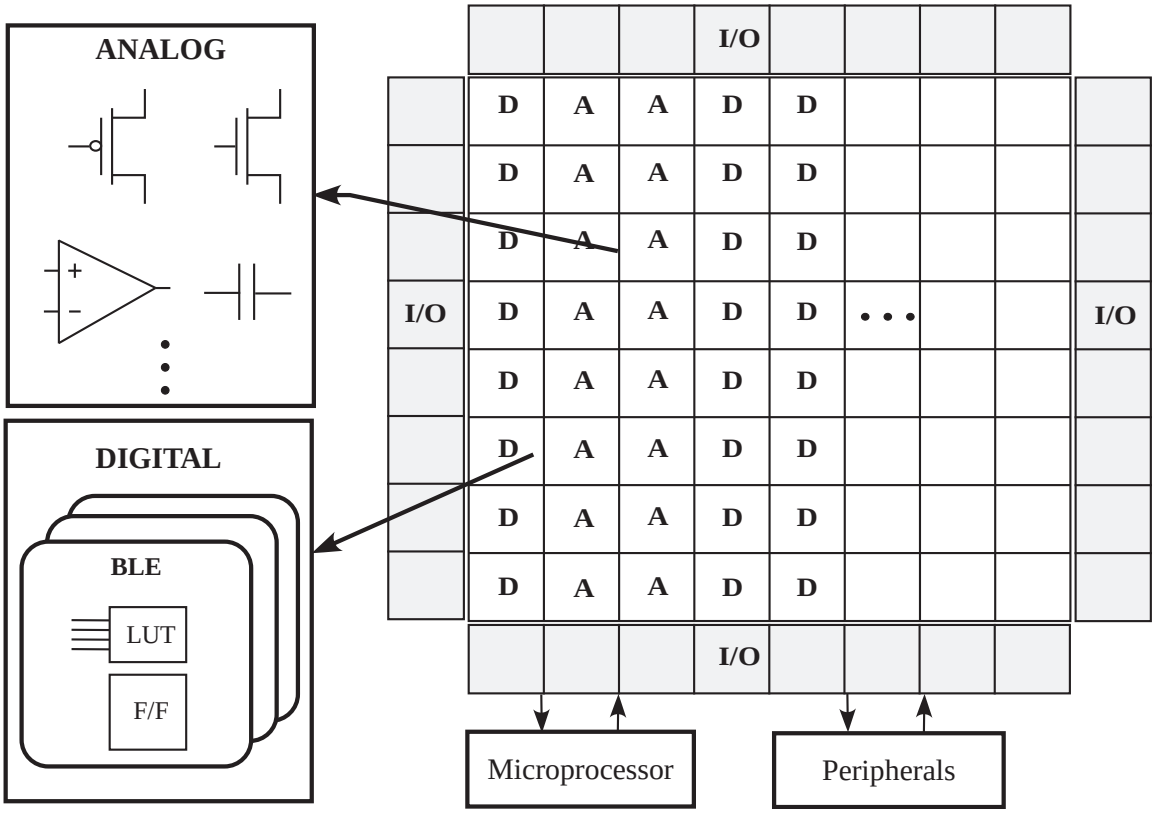


Figure 30: Illustration of the structure of FPAA devices. The particular FPAA, or system of FPAAs and/or other components are defined by their different technology files, chosen by the user. Such an array could be a combination of components with analog (i.e. Computational Analog Block, or CAB) or digital components (i.e. Computational Logic Block, or CLB) that are connected through a switch matrix, as well as a range of I/O components and special additional components.

of the SoC FPAA used in this chapter for experimental measurements. This SoC FPAA enables nonvolatile digital and analog programmability through Floating-Gate (FG) devices, both in the routing fabric, but also for parameters for the computing elements.

Figure 31 shows description of component blocks in the library (all level=1 blocks) and their resulting circuit schematics. Although a circuit expert gains tremendous insight to the particular circuit being compiled and used on the IC, most system designers are satisfied with getting the desired functional behavior, with minimal nonidealities from the circuit. The result is a rich set of analog and digital blocks, similar to FPGAs when using graphical design tools (i.e. [82]), that can be expanded

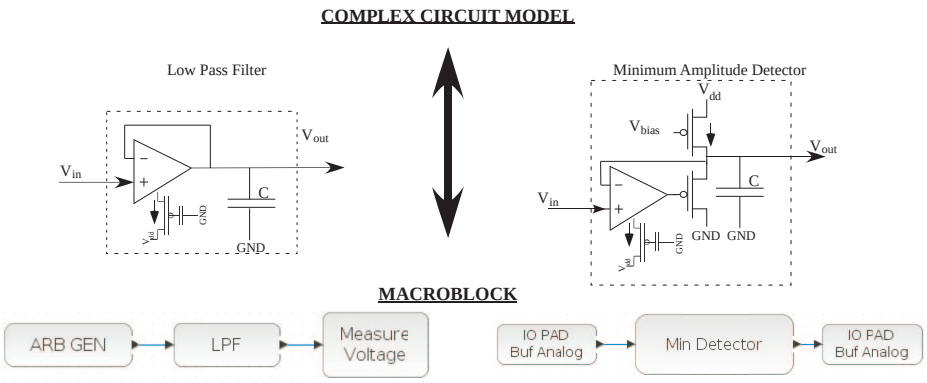


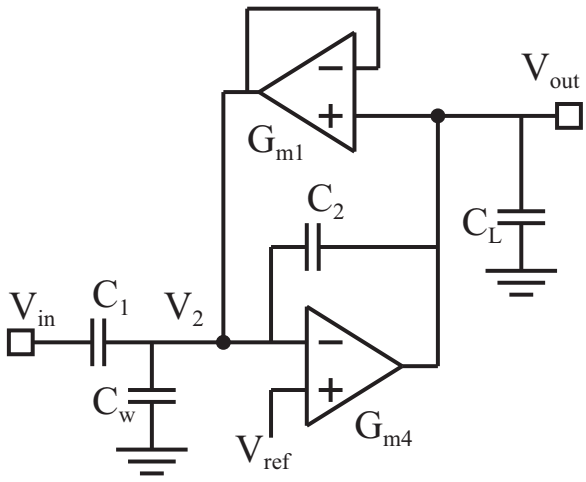
Figure 31:
The tool allows us to draw in block diagrams for mixed-mode computation; each of these cases the inputs could be a scalar or a vector; In each case, we often want to encapsulate the knowledge of the designer as much as possible in the resulting design. For example, the original analog designer might want a group of circuits all in a single CAB; we use a *macro block* to encapsulated as a single block in a CAB that are built from basic elements in the Analog and Digital Tiles, using separate blackboxes in VPR to keep all elements in the same Tile. (a) We show an example of a few low-level circuit components and their block diagram, as well as some of their testing circuits, that includes analog and digital components. We show a LPF and a Minimum Amplitude Detector circuit. (b) A rich palette of macroblocks that can be generated using the core elements of a chip.

and grown as needed.

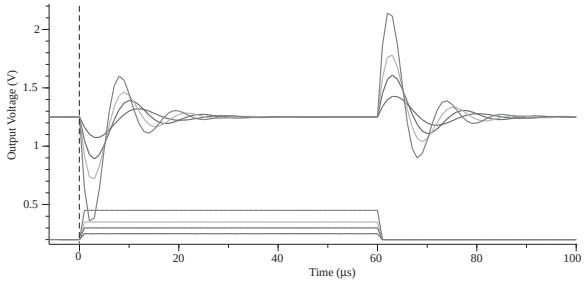
Figure 31 also shows typical components in a CAB, with its typical routing infrastructure of input and output port lines to the rest of the manhattan geometry routing fabric. We see the detailed routing to compile the C^4 bandpass filter circuit. We define a *macroblock* as encapsulating complex circuits using elements inside a single tile to create a single block, enabling more efficient high-level routing. The connections within the CAB are pre-optimized and the tool now only handles placement and global routing. The approach enables us to encapsulate much of the objectives of the circuit designer, often started at a level=2 block, as it becomes a full level=1 block.

Further, the use of FG devices for switches effectively embeds analog components into the routing fabric as well as enabling connections on the resulting lines [83], enabling often far more computation in just the routing fabric compared to the CAB or CLB elements. Any tool development for these FPAA SoC must be able to handle these opportunities; almost all configurable systems have some similar opportunities

that must be encoded in the system's technology file.



(a)



(b)

Modify ODEs of the form
$\frac{d\mathbf{V}}{dt} = \mathbf{f}(\mathbf{V}, \mathbf{V}_{in}, \frac{d\mathbf{V}_{in}}{dt})$

Modify ODEs to have
no input derivatives
$\frac{d\mathbf{V}}{dt} = \mathbf{f}(\mathbf{V}, \mathbf{V}_{in})$

$$C_{eq}\frac{dV_1}{dt} = C_1\frac{C_L+C_2}{C_2}\frac{dV_{in}}{dt} + \frac{C_L+C_2}{C_2}I_2 + I_1$$
$$C_{eq}\frac{dV_{out}}{dt} = C_1\frac{dV_{in}}{dt} \frac{C_1+C_2+C_w}{C_2}I_1 + I_2$$
$$I_2 = I_{bias2}\tanh\left(\frac{V_{out}-V1}{V_L}\right)$$
$$I_1 = I_{bias1}\tanh\left(-\frac{\kappa V_1}{2U_T}\right)$$

$$(C_1 + C_2 + C_w)\frac{dV_1}{dt} = C_1\frac{dV_{in}}{dt} + C_2\frac{dV_{out}}{dt} + I_{bias2}\tanh\left(\frac{V_{out}-V1}{V_L}\right)$$
$$(C_L + C_2)\frac{dV_{out}}{dt} = C_2\frac{dV_1}{dt} + I_{bias1}\tanh\left(-\frac{\kappa V_1}{2U_T}\right)$$

$$C_{eq}\frac{dV_1}{dt} = \frac{C_L+C_2}{C_2}I_2 + I_1$$
$$C_{eq}\frac{dV_{out}}{dt} = I_2 + \frac{C_1+C_2+C_w}{C_2}I_1$$
$$I_2 = I_{bias2}\tanh\left(\frac{V_{outa}-V2+(\beta-\alpha)V_{in}}{V_L}\right)$$
$$I_1 = I_{bias1}\tanh\left(-\frac{\kappa(V_2+\alpha V_{in})}{2U_T}\right)$$

Figure 32: Approach to building a level=1 macro model for the C^4 filter that corresponds closely to measured experimental data. (a) Circuit diagram for a C^4 bandpass filter. (b) Simulation of a step response for the C^4 bandpass filter. (c) Starting equations from the circuit in (a). (d) Modification of the equations into the 1^{st} form. (e) Modification of the equations into the final Xcos ODE formulation.

4.3 Methodology for Implementing the Tool Set

In this section we dive deeper into the key aspects of the high-level tool infrastructure.

Our system requires that we can use the same Xcos block diagram structure to both

simulate and compile to hardware. Our approaches in this section follow the level=1 definition [124], as defined elsewhere and first fully implemented in this work.

For level=2 cases, the compilation to BLIF / veriog follows the same path as level=1, but the simulation environment requires a far more complex simulation environment. Simulation in level=2 requires compiling the net list into a SPICE, direct measurement after compilation, or developing a simulation framework using Modelica with Scilab; this last topic will be the topic of future discussions.

In the following sections, we will address, in turn, the aspects required for level=1 macromodeled simulation, and then, the aspects required for *sci2blif* which converts the Scilab structure into a format ready for place and route compilation.

4.3.1 Macromodel Simulation

A typical design flow includes simulating a designs functionality, analyzing the results, and iterating to a good solution, before proceeding to hardware synthesis, often because of the constraints of accessing such a hardware system. At first, one might ask if physical hardware is directly available (and portable), why not always go directly to circuit measurement, where we get precisely our results in real time. Even in cases where hardware is available, it is often useful to have one simulation case, say for DC values and a reference simulation, to compare with experimental measurements.

The amount of simulation one might do before compiling a circuit will depend on compile time (longer compile time, more simulation), accessibility to FPAA hardware, whether in person or remote, user inexperience (more inexperienced, longer simulation time) as well as number of potential debugging points required.

We focus the simulation on as fast a simulation model as possible that gives accurate results. Further, different from a SPICE model, we do not need to make a model every possible transistor configuration and situation, but for a given macro model, we have precisely one specific case related to a particular hardware device,

greatly simplifying the resulting computations. We want a simulation accurate enough compared to the real data, but without much computational complexity. Scilab, like MATLAB, optimizes for vector operations; our vectorization of blocks preserves this functionality, as well as results in the fastest numerical simulation possible.

The analog system modeling requires using Ordinary Differential Equations (ODE), potentially in combination with algebraic equations, that capture the nonlinearities of a circuit in continuous time are used in the computational function. Scilab also enables discrete time modeling as well as modeling for clocked systems, which follows a similar approach. In the SoC FPAA, .every connection point will have some capacitance, resulting in dynamics where the resulting capacitor voltages would often be the state variables. The required form for Xcos /Scilab for ODE simulation is the standard form of

$$\frac{d\mathbf{V}}{dt} = \mathbf{f}(\mathbf{V}, \mathbf{V}_{in}) \tag{4}$$

where $\mathbf{V}$ is the vector of state variables (i.e. voltages), and $\mathbf{V}_{in}$ is the vector of system inputs. The resulting ODE definition is put into the computation function code using this functional form.

Figure 32 shows an example to formulate a physically realistic model for a C^4 bandpass filter. A particular system will require reformatting these vectors from typical circuit analysis. We show an example of a C^4 bandpass filter; this block was shown to be useful elsewhere, but not modeled at circuit (just linear transfer function). Nonlinearities to be modeled accurately (as seen by the $tanh()$ function) to enable a system designer to minimize the effect where needed, as well as to empower a designer to utilize nonlinearities when desired. Figure 32 shows the resulting three iterations of mathematics required to get the physical equations to their proper Xcos simulation form, as well as resulting Xcos simulation data that corresponds closely to experimental data.

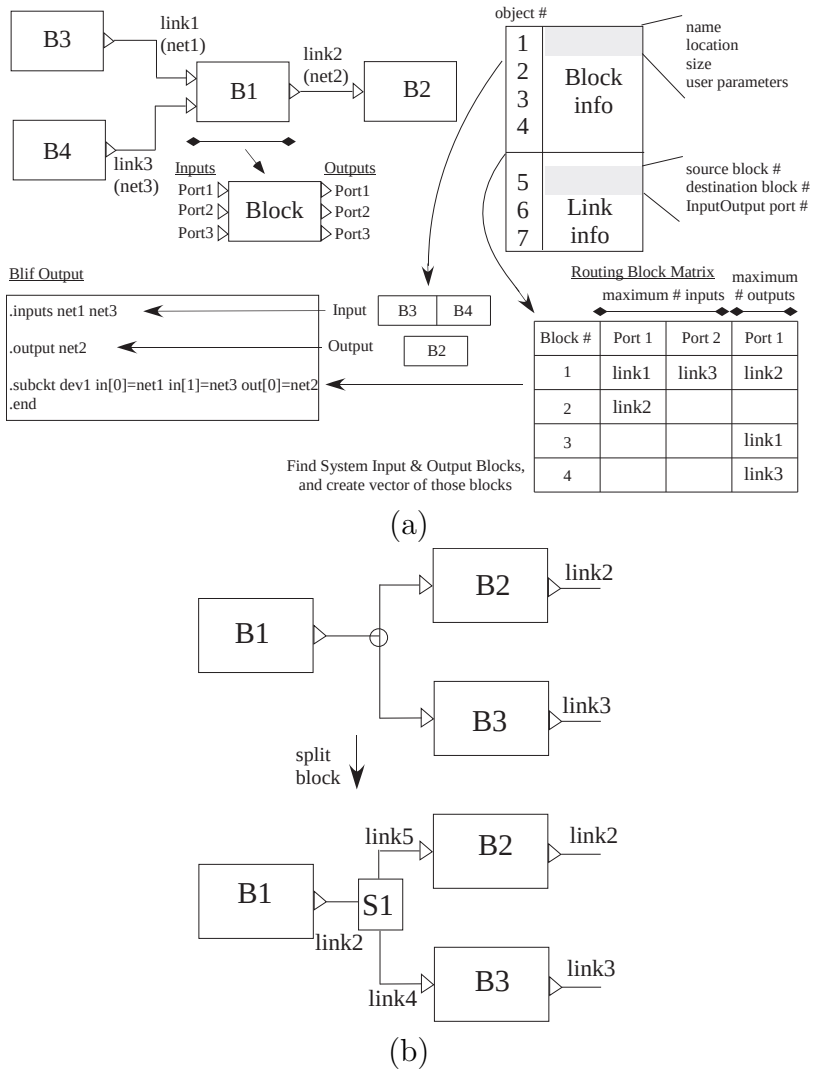


Figure 33: *sci2blif* fundamentals: Xcos model to blif / verily net list to put into VPR. (a) The data structure for a single set of blocks is an array with the block information, as well as link information. Blocks are enumerated by when they are created in Xcos; links are enumerated by where they are located on the block. We also show the transformation of this data structure to blif / VPR representation for VPR. (c) The resulting data structure of the Xcos network only allows for a single input and output for a particular link; therefore we need to have additional blocks included to handle when we have a single output going to multiple inputs.

4.3.2 *sci2blif*: Xcos to VPR

When user presses *Compile Design*, Scilab calls up Sci2blif that uses the Xcos representation to create a circuit format in blif / verilog for the place and route tools to create a switch list, as well as gathers the resulting assembly language modules. Analog blocks are directly converted into a Blif format. Digital blocks first have to call VTR before merging the resulting blif file. The switch list represents the low-level hardware description (i.e. switches to be programmed).

Figure 33 illustrates the approach in converting from Xcos visual representation to blif files for the analog components; the digital approaches are similar, although typically easier. Scilab saves the graph as a data structure shown in Figure 33a, described as

```
data = scs_m
```

that describes the Xcos file contents. The block objects are listed first followed by then link objects and then they are listed by link numbers.

The approach is converted in three passes over the data structure. The first pass parses *data* over the blocks portion to determine the number of blocks that are compiled to CAB/CLB, input blocks, and output blocks. The input and output blocks object numbers are saved in two separate vectors (a and b, respectively). We define B as the number of blocks, I as the number of inputs, and O as the number of outputs. Finally, the *data* object is represented as a matrix, G, of size [(B + I + O) x B] that contains the net numbers corresponding to each of the blocks to be compiled.

The second pass parses *data* to determine which block's input or output port is connected to another block's and input OR output port. Each link is represented by two values the source and destination in *data*. The information provided is block

number, port number (ports on blocks are numbered top-down for inputs and outputs), and if port is an input or output. The net number is placed in the matrix mentioned above.

The third pass parses *data* to generate resulting blif statements for compilation. The input and output vectors and the matrix are used to put the nets of inputs and outputs at the beginning of the blif file. Then one by one the command for each block is put in... the net numbers are retrieved from the matrix using the block number.

Figure 33c shows when users connect an output of a block to at least two inputs, an extra small block is inserted into the Xcos internal representation, increasing the number of blocks and links, that is removed before generating blif file.

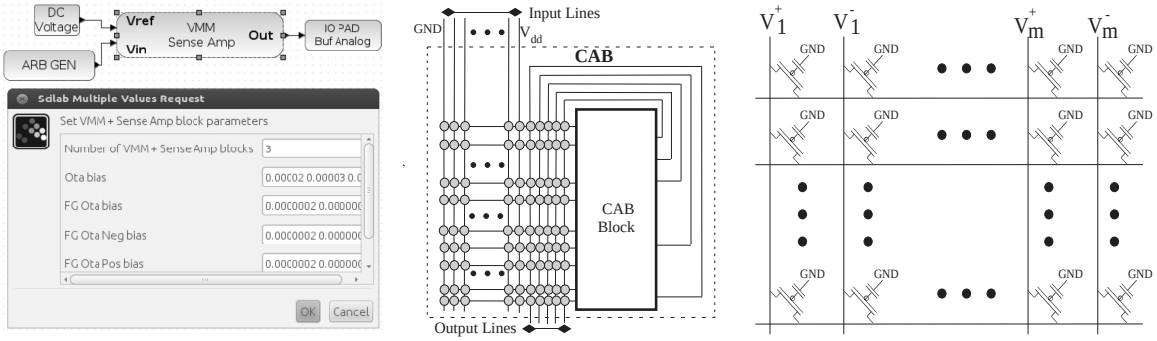


Figure 34: A key feature to these tools is the ability to build useful computation out of routing resources; for example, the synthesis, place, and routing of circuits containing Vector-Matrix Multiplication (VMM) built out of routing (i.e. floating-gate) switches. (a) Block Diagram and parameters for a VMM block. (b) Vector-Matrix Multiplier can be built from a crossbar switch matrix. (c) Local interconnect routing resources inside an analog tile.

4.4 System Examples

In this section we will investigate compilation of more complex system examples particularly using the routing components as computational elements. In our SoC FPAA, using FG switches enables analog computing devices when we program a switch to an analog value. For example, a crossbar array of these analog programmed FG switches could compute an Analog Vector-Matrix Multiplication (VMM), common

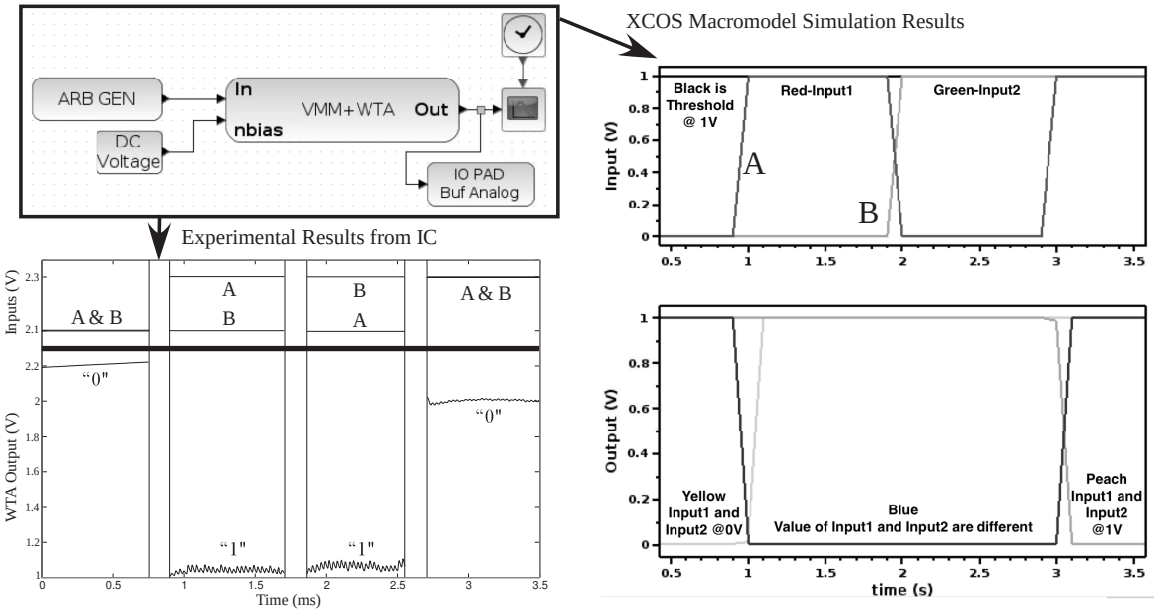


Figure 35: A system example showing a basic circuit classifier built from a VMM + a Winner-Take-All (WTA) block. The three input vectorized system, with a 3X3 VMM, is configured as a two-input EXOR gate; the resulting Xcos system block diagram is used for high-level simulation as well as experimental results, which agree closely in their experimental results.

in all signal processing operations, entirely in routing fabric. The inclusion of such capabilities requires additional sophistication at all levels of the tool flow.

Figure 34 shows an Xcos vectorized block diagram to implement a VMM structure, as well as its implementation using local routing crossbar array, as well as circuit representation, for a VMM computation. This circuit requires a current-to-voltage conversion, in this case a transimpedance amplifier of two OTA devices, to be consistent with level=1 requirements. The resulting block has a dialogue box for key parameters.

Figure 35 shows a VMM+ Winner-Take-all (WTA) circuit used as a classifier circuit, including the vectorized test system as well as macromodelled simulation data and experimentally measured data from the same vectorized test system. The VMM + WTA classifier both elegantly compiles into FPAA devices using he VMM through FG routing fabric, but further, the classifier combination using a k-WTA is a universal approximator in a single classifier layer [160]. We use a $3x3$ VMM matrix

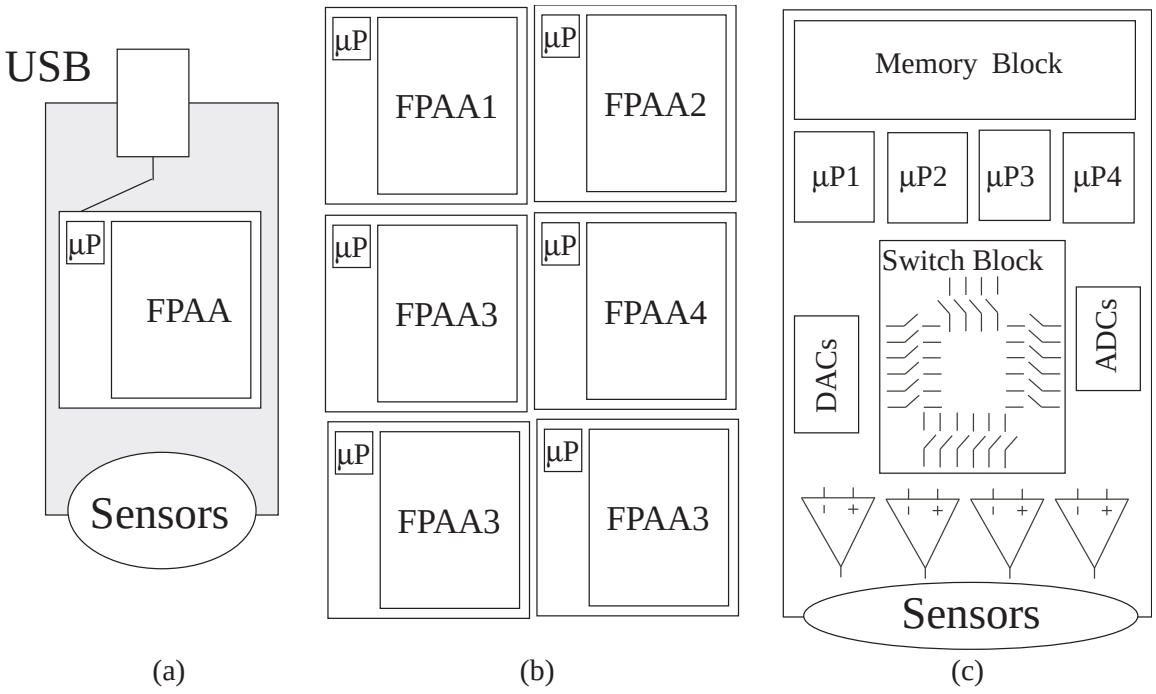


Figure 36: Possible approaches for mixed-mode computing systems. Implementation could be a (a) single FPAA device, (b) a board of FPAA devices, or even (c) a board with no FPAA devices but with programmable parameters and topology for a resulting board encoded in the resulting technology file.

to demonstrate an example of a 2-input classifier consistent with XOR functionality; For WTA circuits, a low output voltage signifies a winner.

4.5 Summary, Comparisons, and Approaches for Analog–Digital Co-Design

We presented an Analog–Digital Software-Hardware CoDesign environment and focused examples as mixed-signal design using FPAA SoCs. The tool simulates designs as well as enables experimental measurements after compiling to SoCs in the same integrated design tool framework. The toolset is open source (after publication) setup as an Ubuntu virtual machine enabling straight-forward user setup as well as open to contributions from third party users empowering a wider community to do analog and digital system design. Digital co-design questions pose issues for systems of mixed

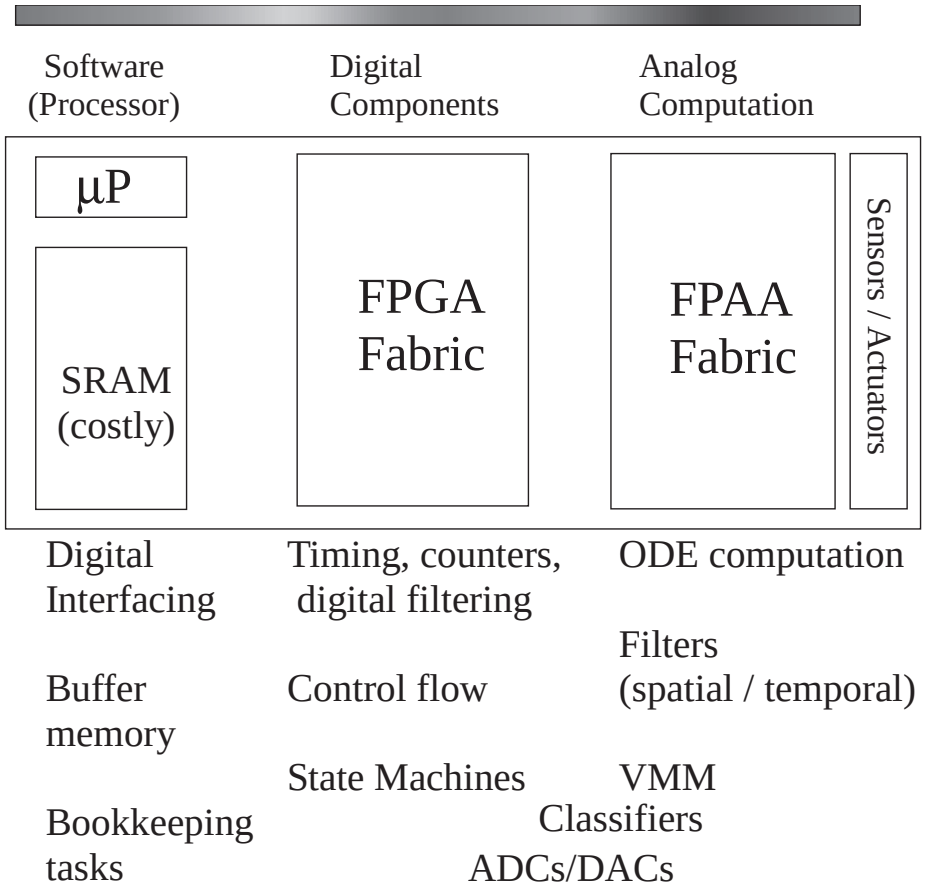


Figure 37: We show a heuristic guide in the analog–digital hardware-software system CoDesign for such computing systems.

hardware (i.e. FPGA) and software (i.e. code running on processor(s)) on the particular partitioning of the resulting computational system based on metrics of power, area, time to market, etc. The recent including of programmable and configurable analog computation allows this community to revisit fundamentally these tradeoff and issues already a vibrant field.

The need for large-scale design tools for SoC FPAA devices was the practical driver to create our toolset, although the approach is entirely extendable to a wide range of analog–digital programmable–configurable systems. We have also developed a high level software in Scilab (open-source MATLAB version) that converts high-level block description by the user to *.blif* format, which acts as an input to our modified

VPR tool [121], including our code *vpr*2*swcs*, as well as modified architecture files for analog–digital SoC. Our tool, *sci*2*bli*, translates a block in scilab to .blif format. The resulting tool builds an analog, as well as mixed-signal library of components, enabling users and future researchers of the basic analog operations / computations that are possible.

We built this entire toolset as an open-source configuration to explicitly enable a wider user community for these approaches. We package the entire tool flow, from Scilab/Xcos, device library files, through *sci*2*blif*, *vpr*2*swcs*, and modified VPR tools into a single Ubuntu 12.04 Virtual Machine (VM) that encapsulates the entire toolset that simply requires pressing one button to bring up the entire graphical working toolset. The approach allows us to fully distribute this toolset (which we will as soon as it is published), both for classroom use, research groups, as well as interested users; we hope this approach both encourages a user community around these tools as well as a community to further improve the toolset. Further, the need to incorporate modified VPR tool, which prefers to live under a unix environment, along with the significant modifications from our previous tools developed in Simulink [124], including modifications to enable Manhattan geometries and on-chip μProcessor, gave the positive pressure to move our tool framework to this open source Scilab/Xcos environment and fully implement the proposed level=1,2 concepts previously described.

We see this toolset as expanding out the graphical design approach for analog–digital computational systems, from what we see for relatively well established FPGA design tools for Simulink tools [82] developed to work with Xlinix [94] and Altera FPGA [95,96] devices. Simulink provides the framework to input into Xlinix / Altera compilation tools, completely abstracting away the details from the user. but in an approach for analog–digital (or simply analog or digital) application spaces. Currently the Simulink approach is the dominant graphical FPGA approach, allowing both standard simulink blocks to compile to verilog blocks as well as support for specific

blocks. although one finds a few other open-source approaches [125] approaches.

These approaches are also consistent with the world of hardware-software co-design (i.e. [115]), where almost all of the work focuses on digital hardware-software codesign (for example [91–93]). Rarely, we see directions towards analog approaches [89, 90], but usually more theoretical in nature without experimental hardware available, and therefore lacking momentum required to make into a system usable for even a class audience.

A remaining question is understanding the conceptual framework to guide the designer in these analog–digital co-design problems, a question we expect will be the subject of many future discussions and tool developments. Figure 36 shows we could be designing with a single FPAA IC, multiple Ics, rack of boards, etc. as well as a set of other components that have some programmability. The high-level graphical tools enables a user to be able to try different approaches to optimize the system performance, allowing consideration of trade-offs of power, system utilization, time to market, etc.

Figure 36d illustrates initial starting guidelines for the problem partition. In particular, one will typically want the heavy computation as physical computation blocks near the sensor where the data originates or transmits, ideally in a data flow approach to minimize the amount of short-term storage elements (power and area issues). Moving heavy processing to the more analog approaches also tends to have less impact on line and substrate coupling, a significant issue for mostly digital computing systems. Often the line between digital and analog computation is blurred, for example for data-converters or their more general concepts, analog classifier blocks that typically have digital outputs. We expect the digital processor will be invaluable for bookkeeping functions, including interfacing, memory buffering, and related computations, as well as serial computations that are just better understood at the time of design compared to other approaches.

This chapter discusses a novel remote test system, enabled by configurable analog–digital ICs to create a simple interface for a wide range of experiments; We have seen a wide range of previous remote test systems that have to spend considerable time developing their hand-tailored configurable system [108–113]. Figure 38 remote test system requires no additional setup, other than the experimental FPAA system, other than simple email handling, which is available over almost all network systems without affecting the network. Independent of the distance, the system enables users around the globe we have an internet connection sufficient to send and receive email. We see the opportunities both in academic as well as research and industrial applications. This approach minimizes computer support setup and maintenance, relieving the pressure overworked computer support staff, particularly in cost-conscious academic environments, trying to keep pace to maintain a larger number of computing systems.

An analog–digital programmable and configurable IC system, enabled by large-scale Field Programmable Analog Array (FPAA) device(s), requiring a single digital external digital interface opens opportunities for a simple remote test infrastructure. This approach gives a simple digital peripheral using a standard interface (i.e. USB), enabling a small Internet of Things (IoT) block interfaced through an email system to an open-source design / control tool. The resulting controlling device, whether it be directly connected through this digital port (i.e. phone or tablet) or through a network, can be a potentially simple OS enabling all features on the resulting system, including sensory / actuation devices connected to this remote platform.

We present an integrated remote testing system requiring minimal IT overhead. First, we will overview the FPAA devices and Baseline Tool Framework so that our discussion is self-contained. Second, we will present a Low-pass Filter as the overview example for the Remote System. Next, we will present the range of user interfacing, expanding the remote user application as well as measurement capability.

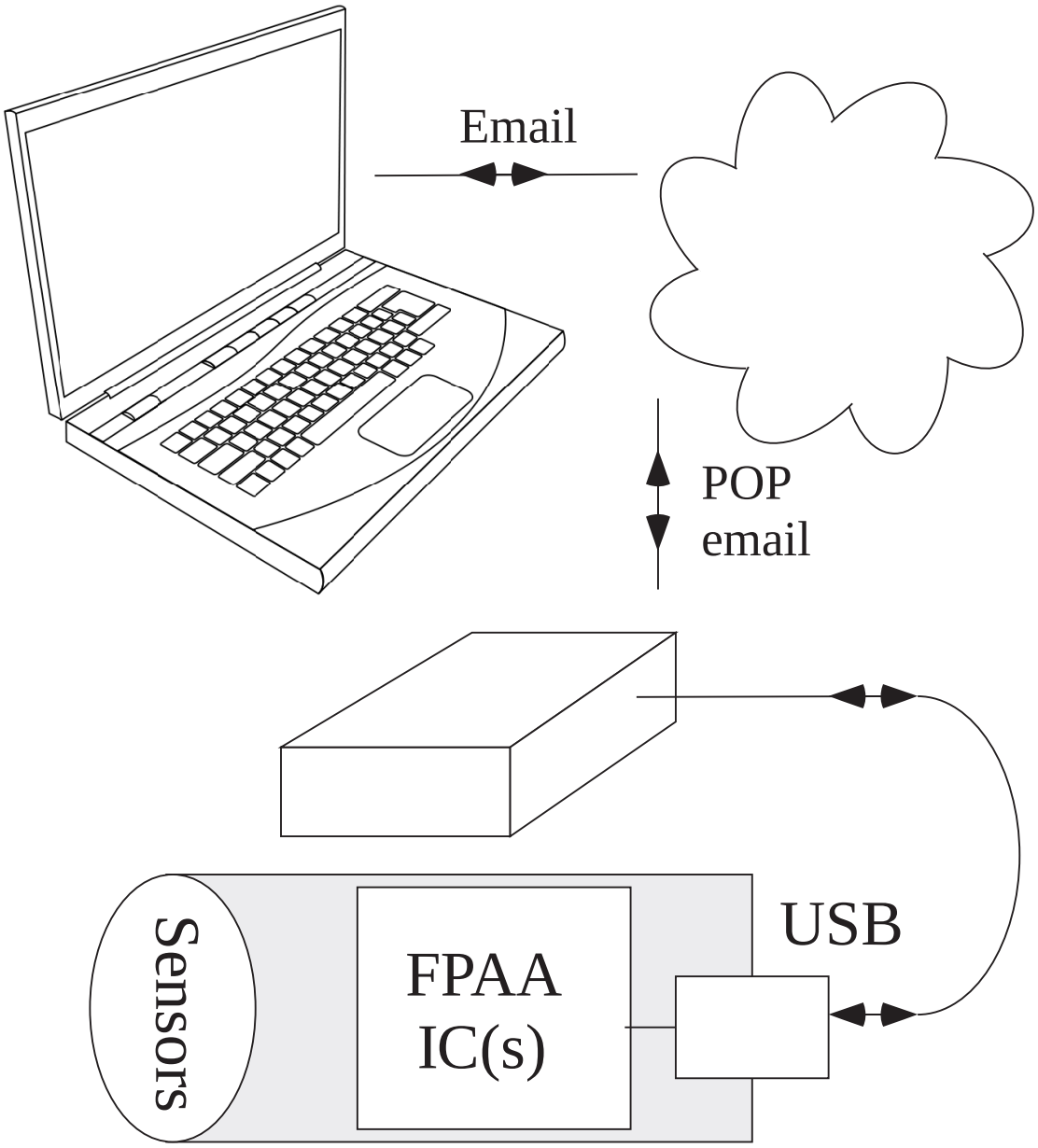


Figure 38: We present a remote test system based on FPAA devices that can be used within our current framework of high-level, open-source Xcos/Scilab tools. With a single button click in the graphical tool, the system will email the resulting targeting code for the FPAA device to a server location, to be picked up by the remote system, that compiles, runs, and then emails back the target results.

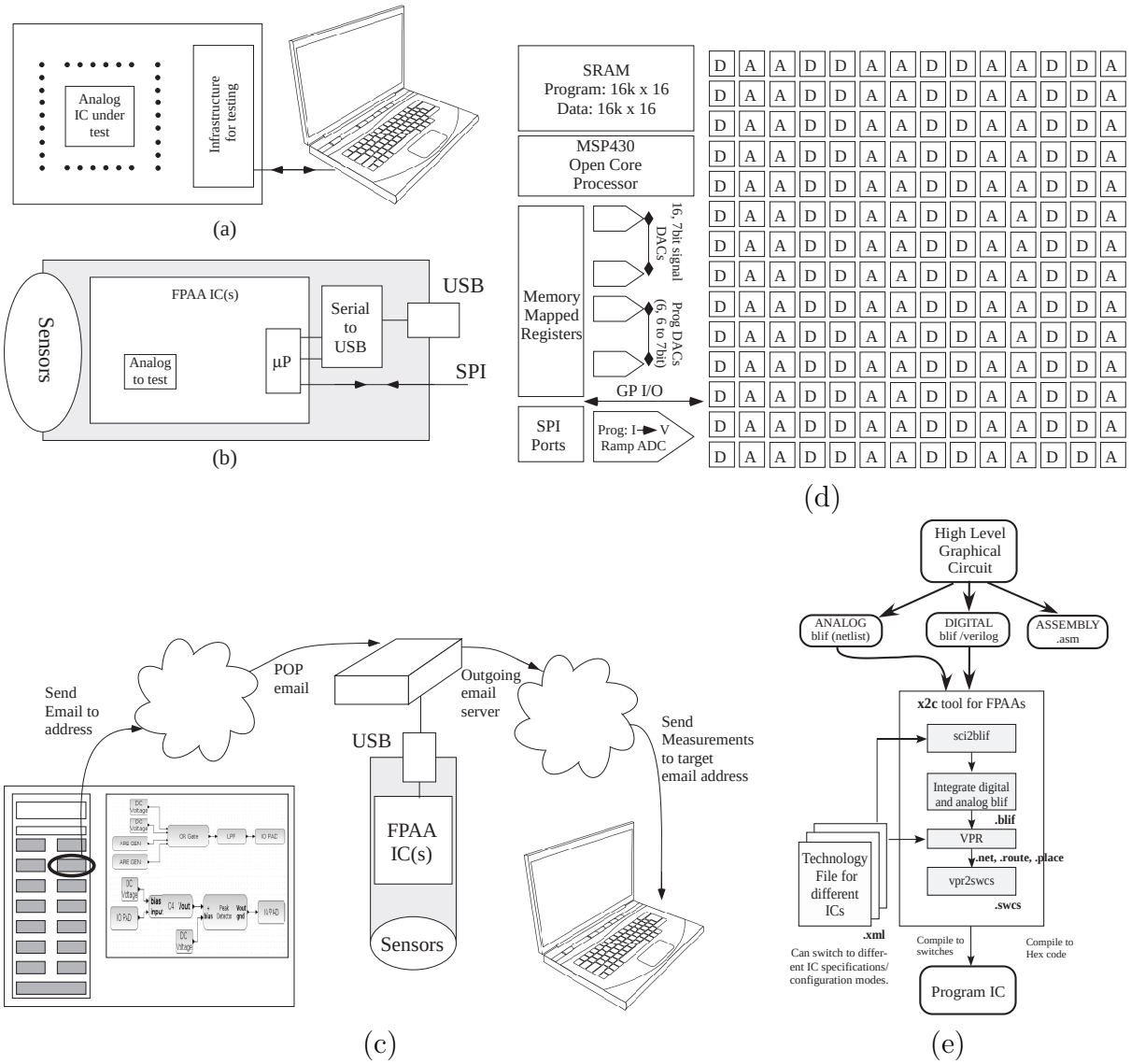


Figure 39: System perspective using a remote test system to utilize mixed-signal configurable systems. (a) Classical approach when considering analog IC design and testing that we have an analog component under test along with all of the required board (or bench) infrastructure required. (b) For these mixed signal ICs, the IC is an entire system, acting as a peripheral through a USB port and/or optional SPI port. Also, the resulting analog to test, if making a complete parallel to the system in (a) is a small part of the overall available computing infrastructure. (c) Detailed flow for the remote test system implementation. The design toolset in scilab / Xcos allows the user to "send email" in addition to "program FPAA". (d) Illustration of the structure of FPAA SoCs. The particular FPAA, or system of FPAAs and/or other components are defined by their different technology files, chosen by the user. (e) High level block diagram of the FPAA Xcos toolset, including the high level design environment as well as the compiler, **x2c**.

CHAPTER V

MODELING VOLTAGE-MODE CMOS DENDRITES

Neuromorphic engineering has garnered ever-increasing interest ever since Carver Mead's early explorations of the field [128]. Neuromorphic engineers claim that transistors can be used to emulate biological processes. Silicon devices and biological structures operate based on similar physical principles, so it is possible to make circuits which share many of the computational properties of neurobiological systems. There are two consequences of this statement: neuromorphic circuits can be used to natively simulate biological systems, and they can also be used to perform bio-inspired computation. This chapter explores how neuromorphic technology can be applied towards emulation of dendritic behavior.

Computational neuroscientists use mathematical models implemented on digital computers to simulate biological processes. While these are powerful tools, their effectiveness is decreased as simulation sizes grow. However, neuromorphic engineering promises a different paradigm: simulation through the physics common to silicon technology and biological systems. This allows for real-time emulation of dense biological systems, rather than a slower and less efficient numerical simulation.

The dendrite is a highly-branched conductive medium that connects neuron's synapses to its soma as shown in Fig. 40. They perform linear (and sometimes nonlinear) summations of input currents, directional selectivity and coincidence detection. Previously believed to have no computational value, they were modeled as wires. However, studies have demonstrated otherwise [14, 18, 20]. In order to begin to take advantage of this computation, we have verified that some of the most basic properties of dendrites can be observed using analog CMOS circuit models.

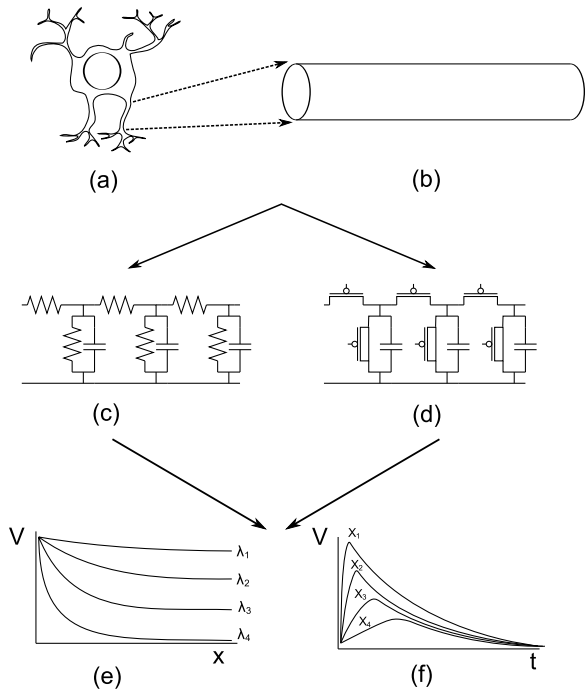


Figure 40: (a) Dendrites are the structures which connect synapses to the cell body. (b) Neuroscientists typically model these structures as passive linear cables. (c) The classical model for this linear cable is an equivalent RC delay line. (d) An alternative model for the linear cable is a network of aVLSI elements, primarily MOSFETs and capacitors. (e) The steady-state behavior of both models is expected to be an exponential decay in voltage, where the amount of decay depends on physical parameters. (f) The dynamic behavior of both models is expected to be exponential decay in space and a delay in time.

5.1 The Silicon Channel

Neuromorphic engineering begins with the principle that the transistor acts as a biological analog. Carver Mead recognized that this is true because both silicon and biological channels behave according to the same natural principle. The channel of a transistor operated in its subthreshold regime is governed by the diffusion equation, as are many biological processes [128].

The channel of a transistor is a region of silicon that separates the drain from the source (see Fig. 41a). This area forms an energy barrier to charge carriers at the source and at the drain. The number of charge carriers at the source or drain end of the channel is determined by the size of this barrier, which is modulated by the

difference between the gate voltage and the source or drain voltage. Since the source is operated at a higher potential than the drain in the P-channel device, the barrier at the source end of the channel is lower, so there are more charge carriers at the source end of the channel than at the drain end. Therefore we have a gradient of charge carriers from the source end of the channel to the drain end. This is illustrated in Fig. 41c. This means that carriers must diffuse from the source to the drain according to the diffusion equation from [128]:

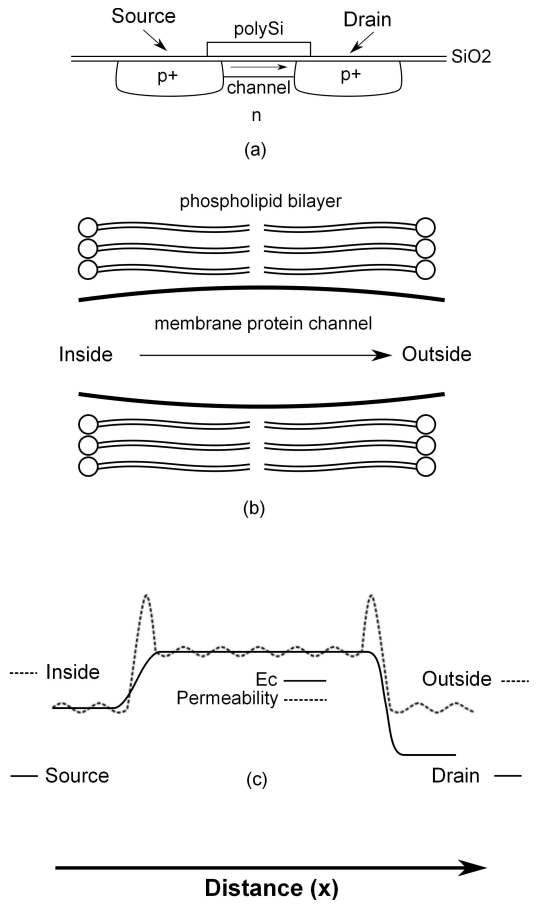


Figure 41: (a) The physical structure of a MOSFET consists of polysilicon, silicon dioxide, and doped n-type silicon. A channel is formed between the source and the drain. (b) The physical structure of a biological channel consists of an insulating phospholipid bilayer and a protein which stretches across the barrier. The protein is the channel in this case. (c) The band diagram of silicon (solid line) has a similar shape to the classical model of membrane permeability proposed by Danielli [135] (dashed line). In both cases, carriers must overcome energy barriers in order to travel from one side of the device to the other.

$$v_{diffusion} = -D\frac{1}{N}\frac{dN}{dh} \tag{5}$$

where $v_{diffusion}$ is the velocity of carriers, D is the diffusion constant, N is the number of charge carriers per unit volume, and h is distance. When the diffusion equation is applied in the case of a gradient of charge carriers from the source to the drain of a pFET channel, the current is given in [136] as:

$$\begin{aligned} I &= I_0 e^{\kappa(V_{dd}-V_g)/U_T}\left(e^{-(V_{dd}-V_s)/U_T} - e^{-(V_{dd}-V_d/U_T)}\right) \\ &= I_0' e^{-\kappa V_g/U_T}\left(e^{V_s/U_T} - e^{V_d/U_T}\right) \end{aligned} \tag{6}$$

V_{dd} is the well potential of the pFET, V_g is the gate voltage, V_s is the source voltage, and V_d is the drain voltage, all referenced to ground. I_0 is a collection of physical constants which is intuitively the saturation current when $V_g = V_s = V_{dd}$. κ is a measure of how well the gate voltage modulates the potential at the channel's surface. U_T is the thermal voltage (typically around 26 mV at room temperature). To simplify the nomenclature, we can reference the terminal voltages to V_{dd}, in which case $I_0' = I_0$. To reference everything to ground, we let $I_0' = I_0 e^{\kappa V_{dd}/U_T} e^{-V_{dd}/U_T}$.

The idea of overcoming energy barriers to produce current is also seen in biological channels. In Fig. 41b, we show the structure of a channel embedded in a membrane. Fig. 41c shows how both biological and silicon channels generate barriers to current, where the barrier is shown as a change in membrane permeability in the case of biological channels and a change in potential energy in the case of silicon channels.

5.2 Implementing the Linear Cable Model with Analog CMOS Circuits

Historically, dendrites have been modeled as linear cables. Their structure consists of a conductive solution that allows current to flow from the synapse to the cell body; a phospholipid bilayer which separates the membrane potential from the external

potential; and ion channels which allow small amounts of current to leak across the membrane. Wilfrid Rall adapted the mathematics originally developed to model core conductor cables and applied it to dendrites [140]. We wish to demonstrate that the behavior of a CMOS dendrite with pFET channels reduces to Rall's mathematical model when operated with small-signal inputs.

Our basic thesis is shown in Fig. 42. We begin with the biological dendrite and model both the conductive medium and the leak channel using a silicon channel. We also provide a bias current to set the resting membrane potential, V_{rest}. We then assume small signals are applied as inputs, and our circuit reduces to a linear model.

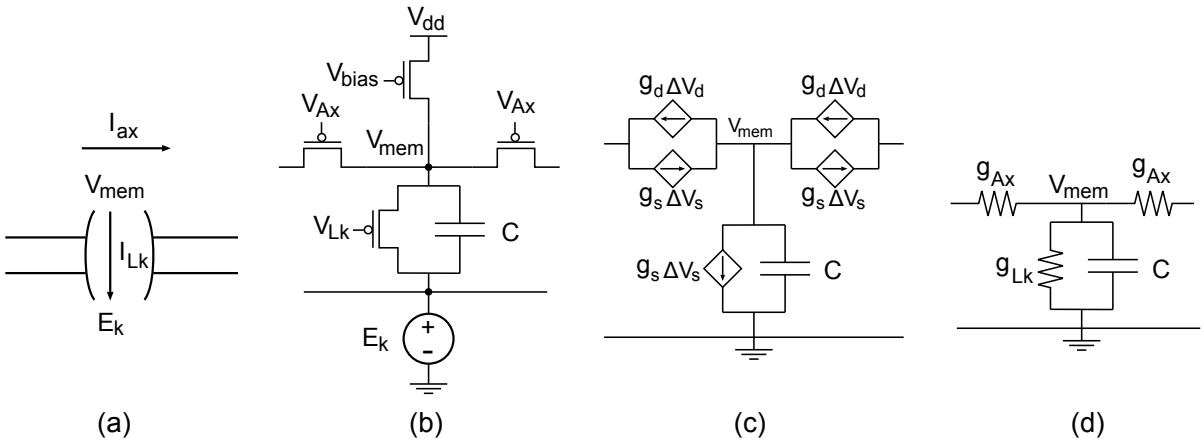


Figure 42: Various models of a dendrite. A biological dendrite is modeled as a conductive cylinder surrounded by an insulating layer. A cross section of this model is shown in (a), where I_{ax} represents the current flowing along the axial direction of the dendrite, I_{Lk} represents current from the dendrite to extracellular fluid through a leak channel, and the internal and external potentials are V_{mem} and E_k, respectively. When we translate channels into transistors, we get the model shown in (b), where both the axial and leakage current flow through transistors. The external voltage is set by a voltage source E_k, and V_{mem} is set by the bias structure. When we linearize the transistor model, the result is shown in (c) and (d). Current sources can be reduced simply to small-signal conductances.

The simplest model neuroscientists use to describe the function of dendrites is known as the Linear Cable Model. The dendrite is treated as a conductive core surrounded by an insulating layer. The core is modeled as a long piece of resistive

material, which can be discretized into many incremental resistances R_{Ax}. The insulating layer is a phospholipid bilayer, and it is modeled as a capacitance C because it separates the internal membrane potential from the extracellular potential. However, there is leakage current from the intracellular solution to the outside of the cell, so a leakage resistance R_{Lk} is also included in the model.

Koch gives a simple derivation of the mathematical cable model for this circuit in [134]. If one writes down Kirchhoff's Current Law (KCL) at the nodes V_{mem} and uses Ohm's Law $V = IR$ and the capacitor equation $I = C\frac{dV}{dt}$, then the following differential equation describes the system:

$$\lambda^2 \frac{\partial^2 V_{mem}}{\partial x^2} = \tau \frac{\partial V_{mem}}{\partial t} + V_{mem} - R_m I_{inj} \tag{7}$$

where I_{inj} is current injected into the dendrite, $\tau = R_{Lk}C$ and $\lambda = \sqrt{\frac{R_{Lk}}{R_{Ax}}}$. τ and λ are called the time constant and the space constant. Intuitively, τ determines how voltages along the dendrite change with time, and λ determines how voltages change with distance down the dendrite. If we only care about the steady-state solution, we can set the differential with respect to time equal to zero. This results in a solution for the steady-state behavior given in Eq. 8.

$$V(x) = V_0 e^{-|x|/\lambda} \tag{8}$$

Our goal is to replace the resistances in the linear cable model with silicon channels. The most intuitive way to do this is to simply replace each resistance with a single pFET. The axial resistances are replaced with a pFET whose gate is set at a fixed potential, V_{Ax}. Similarly, the membrane resistances are replaced with pFETs whose gates are set at a fixed potential V_{Lk}. On an intuitive level, the conductance of the pFETs is set by their gate voltage. We will need to bias the dendrite at a fixed membrane potential, so a transistor which provides a DC bias current is inserted into each node of the dendrite. It has a gate voltage V_{bias}, and it sets the DC point V_{mem}.

The final piece of the dendrite to consider is the capacitance. It is a fact of analog circuits that every node has some capacitance associated with it. So we do not have to place an explicit capacitance at each node to simulate a dendrite. If we so desire, the FPAA has the ability to compile 500 fF capacitances into the nodes. The final circuit is as shown in Fig. 42b.

In order to model an equivalence to the linear cable model, we can simplify the full circuit into a linear one. Each transistor is replaced with a small-signal, linearized model. To do this, we take partial derivatives of the current equation for a pFET as formulated in Eq. 6.

Linear Model of Axial FET

In the operation of the circuit, we will leave the gate fixed at a DC bias, so we can simplify Eq. 6 by incorporating the gate voltage term into $I_{bias} = I_0' e^{-\kappa V_g/U_T}$. Therefore, the current through the axial and leakage pFETs can be expressed as follows:

$$I = I_{bias}\left(e^{V_s/U_T} - e^{V_d/U_T}\right)$$

Traditionally, we form a linear model for this device by taking the partial derivative of the current with respect to a changing terminal voltage. Since a signal is traveling in the axial direction of our dendrite, both the source and the drain of the axial FET are changing. We model this with two current sources in parallel pointing in opposite directions, with the values $g_s \Delta V_s$ and $g_d \Delta V_d$. Ignoring channel length modulation, the values for g_s and g_d are given in [136] as:

$$g_s = \frac{\partial I_{Ax}}{\partial V_s} = \frac{I_{bias}}{U_T} e^{V_s/U_T}$$

$$g_d = -\frac{\partial I_{Ax}}{\partial V_d} = \frac{I_{bias}}{U_T} e^{V_d/U_T}$$

Note that, at rest, the dendrite will be biased such that all source and drain nodes of the axial pFETs will be at the same rest potential, V_{rest}. This means that $g_s = g_d$. We can combine the two current sources into one source with the value

$$\begin{aligned} I &= g_{Ax}\Delta V_s - g_{Ax}\Delta V_d \\ &= g_{Ax}\left(\Delta V_s - \Delta V_d\right) \\ &= g_{Ax}\Delta V_{sd} \end{aligned}$$

So this is simply a small-signal conductance,

$$g_{Ax} = \frac{I_{bias_{Ax}}}{U_T} e^{V_{mem}/U_T} \tag{9}$$

Linear Model of Leakage FET Modeling the leakage transistor is much easier. Both the gate and the drain are fixed to DC voltages. So any change in voltage across the device is completely due to a change in the source. Therefore, the small-signal conductance of the leakage FET is just the source conductance, as given above:

$$g_{Lk} = \frac{I_{bias_{Lk}}}{U_T} e^{V_{mem}/U_T} \tag{10}$$

Deriving the Space and Time Constants

The space constant is the parameter λ in the linear cable equation which describes how voltage in the dendrite decays with position along the dendrite. It is related to the ratio of the axial and leakage conductances. Now that we have linearized our model, we can define a space constant λ by taking the ratio of our conductances:

$$\lambda \equiv \left(\frac{R_{Lk}}{R_{Ax}}\right)^{1/2} = \left(\frac{g_{Ax}}{g_{Lk}}\right)^{1/2} = \left(\frac{I_{bias_{Ax}}}{I_{biasLk}}\right)^{1/2}$$
$$= e^{\frac{\kappa(V_{Lk}-V_{Ax})}{2U_T}} \tag{11}$$

Figure 43 verifies this expression experimentally using the FPAA. We measured how the conductance of a pFET changes as a function of its DC gate potential. To relate this back to Eq. 11, we measure a reference conductance and see how changing the gate voltage affects the square root of the ratio of the new conductance to the reference.

The time constant τ describes how voltages decay with time. It is defined as the product of the leakage resistance and the capacitance, or

$$\tau = \frac{C}{g_{Lk}} = \frac{CU_T}{I_{bias_{Lk}}} e^{-V_{mem}/U_T} \tag{12}$$

Sources of Error: The above expressions hinge on perfect matching among all pFET devices. This unfortunately is rarely achieved. We measured the values of κ and I_0 for a sample of 15 pFET CABs in the FPAA and measured the statistical variation for these two parameters. This information is shown below:

	μ	σ
κ	0.8393	0.0021
I_0	4.5740 fA	0.77549 fA

The above analysis assumes the system is processing "small" signals. We can no longer assume that the linear models behave if they are perturbed far from the DC bias. We limited inputs to the system such that the source nodes of the vertical pFETs never changed by more than 25 mV.

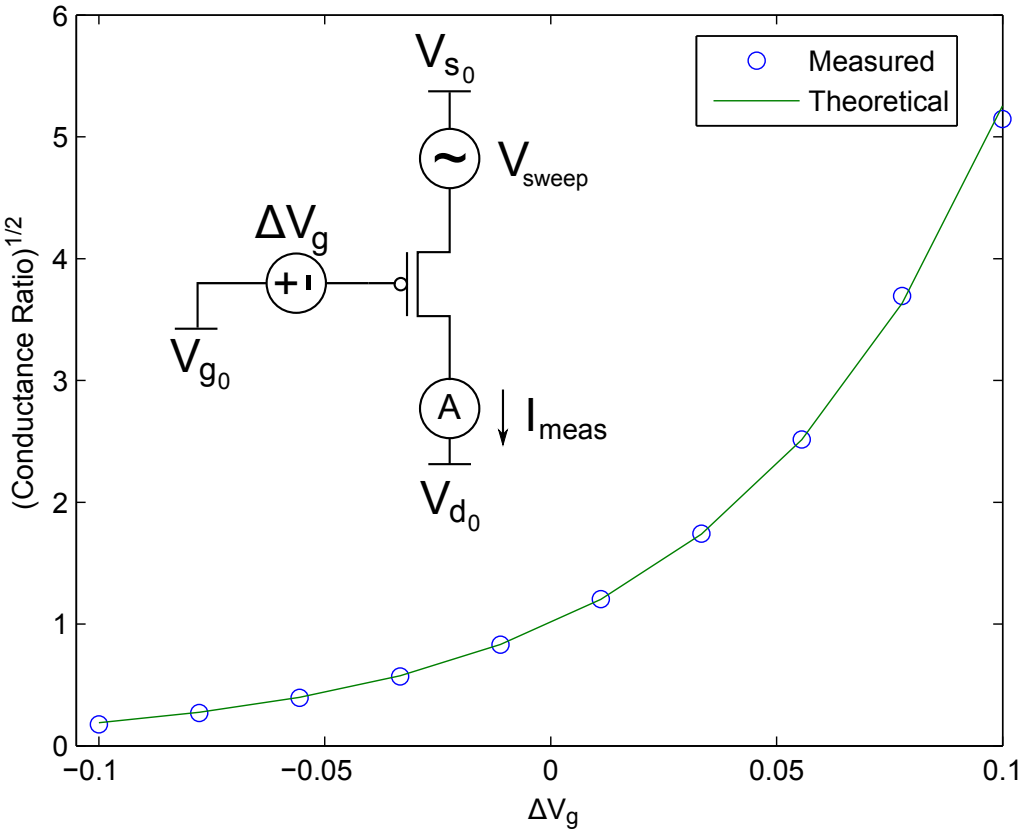


Figure 43: Demonstration that the ratio of source conductances is a function of the difference between gate voltages. We took a CAB pFET and measured a reference source conductance by fixing the DC potential at all of its terminals (V_{s_0}, V_{g_0}, and V_{g_0}), and measuring the DC current. We then swept its source voltage through a very small range (V_{sweep}) and measured the change in current. The reference conductance was the slope of change in current with respect to change in source voltage. We performed this same experiment for ten different values of the gate voltage (V_{g_0} - ΔV_g). We then plotted the square root of the ratio of source conductances as a function of the gate voltage. We used the difference in gate voltages to create a theoretical value of the conductance ratio from Eq. 11, and the two match very closely.

5.3 Demonstrating Equivalence to the Linear Cable Model

We now wish to demonstrate that our voltage-mode circuit retains many of the behaviors of a passive dendrite. We set up our cable using the system shown in Fig. 44 (a).

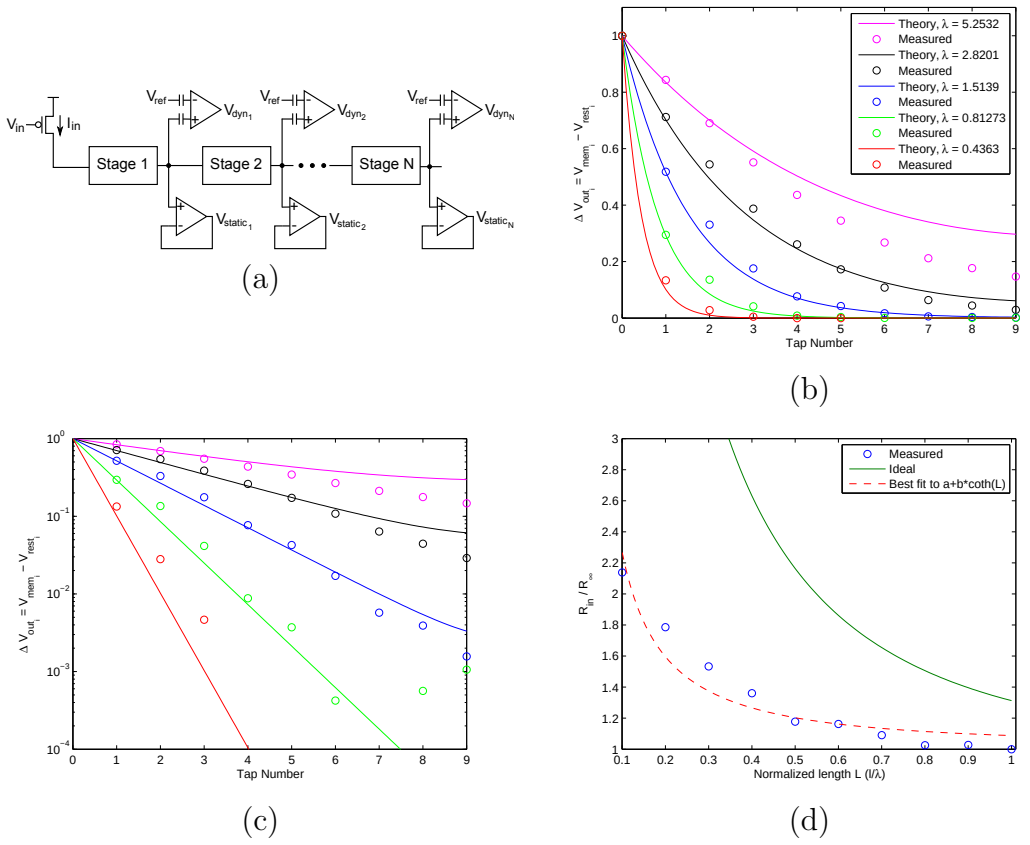


Figure 44: (a) Schematic for taking measurements from the cable. Each block representing a stage consists of one bias, axial, and leakage transistor as shown in Fig. 42b. (b) and (c) Steady-state decay of dendrite voltage. For five different values of λ, a ten-stage dendrite was biased at DC such that the V_{mem} nodes were about 20-50 mV above $E_k = 1$. Then a small DC current was injected into the first node. We then measured $\Delta V_i = V_{mem_i} - V_{rest_i}$ for every node in the dendrite. Then ΔV_i was normalized. The dots are experimental measurements, and the lines represent how the voltages should decay if λ matches the theoretical value perfectly. The linear plot gives an intuitive physical feel for how the dendrite behaves, while the logarithmic plot demonstrates how these are approximately exponential responses. The log plot also shows how error in slope accumulates.(d) This plot shows how input resistance changes as dendrite length is increased. A fixed input current was injected into Node 1 of the diffusor, and the membrane voltage at that node was measured before and after injection. We then calculated the difference between these two ($V_{delta} = V_{mem} - V_{rest}$). This was done for many different dendrite lengths. To calculate R_{in}/R_{∞}, we divided all values of V_{delta} by the value for L = 1. Since the injected current was the same for all tests, the ratio of resistances is therefore the ratio of the voltage responses. The response did not follow the quantitatively predicted curve, but it does demonstrate qualitative behavior similar to what we expect, as shown by the dashed curve, which is a curve fit to a + b*coth(L)

5.3.1 Steady-State Experiments

The first test to perform is a steady-state analysis. In our experiment, we compiled a 10-stage dendrite onto the FPAA. We set $E_k = 1V$ and biased the membrane voltage to around 20 mV above E_k. Due to mismatch among the bias transistors and leak transistors, not all membrane voltages were exactly the same, and they could vary by as much as tens of mV. We attempted to compensate for some of the mismatch by an iterative process of measuring and changing the bias voltages on the gates of the I_{bias} transistors, but this did not remove all of the mismatch. Since this is a dendrite of finite length, the steady-state solution takes on a slightly different form than that given earlier. From [134], the solution is

$$V(X) = V_0 \frac{\cosh(L - X)}{\cosh(L)} \tag{13}$$

where X = x/λ and L = l/λ. For this experiment, we defined the steady-state voltage of a particular node as the difference between its measured rest voltage and its voltage after applying an input. The results for this dendrite are given in Fig. 44 b,c.

The input resistance of a semi-infinite, sealed-end cable is also well-known. Its expression is given in [134] as

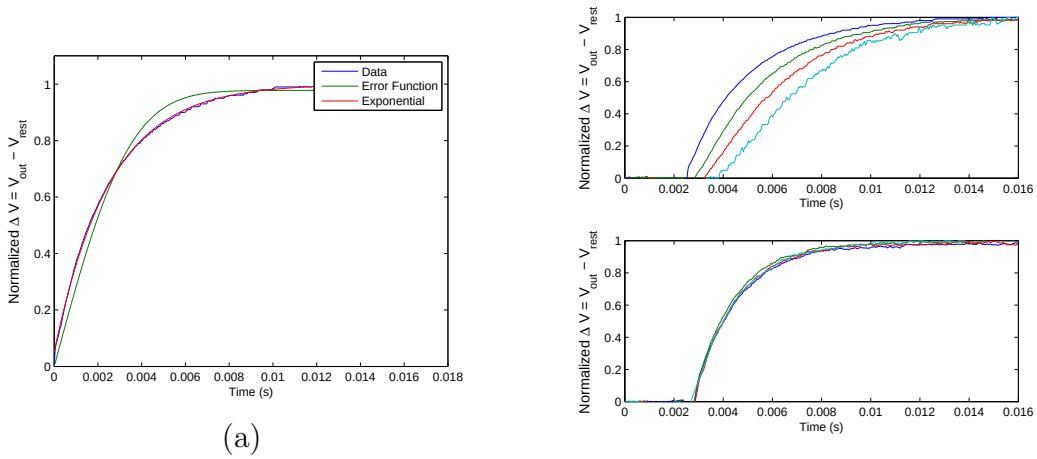
$$R_{in} = R_\infty \coth(L) \tag{14}$$

As L increases, R_{in} approaches R_∞. To test whether our dendrite follows this model, we applied a step input current of I_0 to our dendrite and varied the value of λ. For a fixed input current but variable dendrite length, we can predict what the voltage should be at various points along the dendrite. Our results are shown in Fig. 44(d).

Our theoretical results do not perfectly match the data, and there are a few possible reasons for this. Probably the largest contributor to the problem is biasing the dendrite correctly. For the experiments in Fig. 44 b,c, the resting membrane

potentials were as much as 30 mV away from each other. The ratio of small-signal conductances is $e^{(\Delta V/U_T)}$, so this means that the ratio of two ideally matched conductances could be as high as 3.32. It should also be noted that κ changes with the source voltage, so a 30 mV mismatch in source voltage could also affect κ.

5.3.2 Dynamic Experiments



(a)

(b)

Figure 45: (a) Step response for the first node of a diffusor, along with the best fit to the error function and an exponential function. The step response was obtained by setting the value of V_{ref} on the first node's floating-gate OTA such that V_{dyn_1} was midrail. Then the input current was pulsed, and the waveform was captured. We experimentally determined how much to pulse V_{in} by alternatively pulsing it, measuring how much the first node's voltage changed, and adjusting the gate until the first node's voltage changed by less than U_T, or 25 mV. We chose this value since the FETs would leave saturation if the source voltage changed by much more. We normalized the result by subtracting the DC offset and dividing by the maximum value reached. Linear cable theory predicts that the error function will be a closer fit than the exponential, but the data for our system mirrors an exponential response much more closely. It is possible that our step size was greater than needed to keep all devices in their linear regimes. (b) Step responses for four taps of the dendrite were taken for two different values of λ. For a small value of λ, the velocity of propagation is small, so one can see delays of the response as they travel down the dendrite. For higher values of λ, the velocity of propagation is very fast, so very little delay can be seen. Fig. 46 shows parasitic transients not visible in this figure.

Cable theory provides us with a prediction for what the shape of the step response should look like at the site of current injection. The form is given in [134] as

$$V_{Step}(0,T) = \frac{I_0 R_\infty}{2} erf(\sqrt{T}) \tag{15}$$

We have plotted a representative step response for x=0 along with a best-fit line to this theoretical function in Fig. 45a.

Since the cable model is basically an RC network, we expect to see delay down the line. The propagation velocity of a step input down the line is given in [134] as

$$v = 2\frac{\lambda}{\tau} \tag{16}$$

This means that we can increase the delay down the line (decrease the velocity of propagation) by decreasing λ or increasing τ. In our experiment, we changed λ and looked at how the velocity of propagation was affected. The results are shown in Fig. 45b.

In both the steady-state and dynamic experiments, we have seen a trend in our results. Namely, they agree with cable theory qualitatively but do not match it precisely, quantitatively. We do not expect these nonidealities to affect usability of the dendrites greatly. This is because we believe the computation in dendrites is not governed by precise tuning of every parameter. Neural computation is inherently different from the von-Neumann architectures in which precision is key. They exhibit high levels of stochastic behavior, redundancy, and recurrent connections. Rather, for us it was more more important to see that the basic dendritic properties can be varied over a wide range, allowing gross tuning of parameters.

5.3.3 Effects of a Reconfigurable Testbed

A reality of working in a reconfigurable environment is that parasitics can cause nonidealities to crop up when experiments are run. Fig. 46 demonstrates this. To apply an input current to our system, the gate of a pFET is pulsed low. This pulse can capacitively couple both into the system and into the instrumentation measuring

the system's response. The amount of coupling depends on how the system is routed, so certain care should be made to ensure that system components are routed to minimize such effects. For instance, the routing lines for the voltage measurement circuitry should not be physically close to the digital pulse on the gate of the input current source. Additionally, a cascode should be used on the input current source.

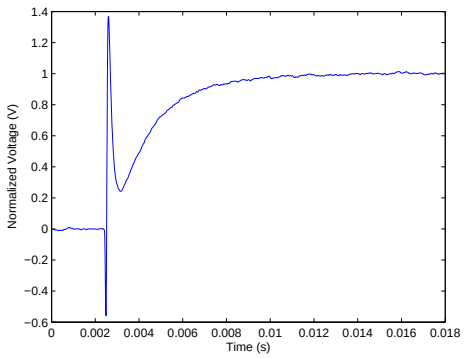


Figure 46: Two parasitic effects seen at once for one particular step response. When the gate of the pFET is pulsed down, some of that voltage change is coupled into the input node of the dendrite, and therefore initially the voltage at the membrane decreases. This change can be seen propagating along the system. For this step response, we also see a spike upwards. This is likely due to capacitive coupling into the instrumentation amplifier (a floating-gate input OTA), because this change is not seen propagating down the dendrite in other plots.

5.4 Simulink Model for simulating CMOS dendrites and FPAA configuration

For DSP and neuromorphic engineers with little or no hardware experience, it is beneficial to have a software tool that can provide an easy interface with the hardware. Matlab Simulink allows users to add new blocks with user-defined functionality providing the user an interactive graphical interface. DSP engineers are familiar with this tool to a large extent. Keeping this in mind, we developed a Simulink model for dendrites. The Dendrite Simulink block provides users with a block-level interface. Sim2spice [141] is the compiling tool we used to convert the block-level implementation to a Spice netlist. The GRASPER tool [142] is then used to configure the FPAA

and the RAT tool [143] is used to view and edit the routing.

5.4.1 Dendrite Simulink Block

The dendrite Simulink block is defined by level-2 M file S-funcions and corresponding netlist elements. The elements used to model the block are the CAB elements on the FPAA. The input parameters for the block are configurable. The Simulink block can be used to run a behavioral simulation of the CMOS dendrites and also generate a Spice netlist to configure the FPAA. It consists of mainly four files :

1. S-function Simulink block: Consists of the physical dendrite block with its inputs, outputs and other input parameters that need to be defined. It is the user-interface block as illustrated in Fig. 47a. The input parameters that the user can specify are given in Fig 47b.

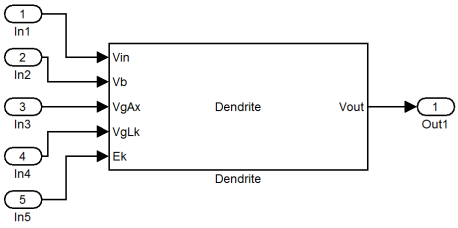
2. Matlab(.m) build script: Builds the spice netlist for the block

3. Description file(.desc): Defines list of paramters needed by the parser

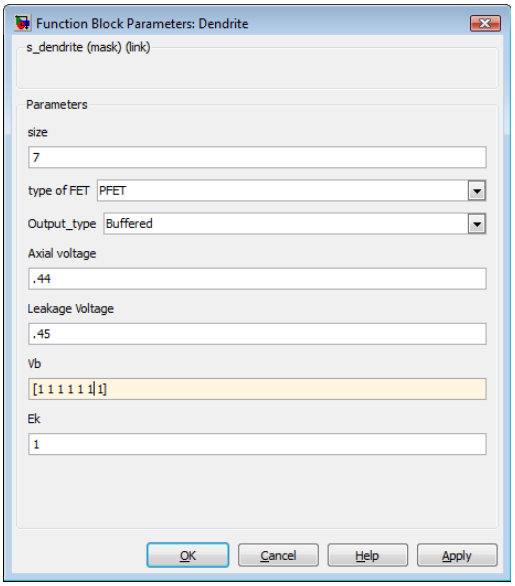
4. Simulink(.m) behavior file: Simulates dendrites in Simulink using the mathematical model based on the device physics of the silicon devices

5.4.2 Behavioral modeling

The Simulink block simulates the behavioral characteristics of the dendrite structure given inputs.This provides the user an insight to the working of the dendritic circuit when implemented using the FPAA. The MOSFET parameters used are based on the MOSFETS present on the FPAA. It is characterized by coupled ordinary differential equations (ODE) and solved using the ode solver ode-45. The model has been tested for both static as well as time-varying inputs and has given reasonable results. We present below a detailed analysis of the mathematical model used, based on the device physics of silicon.



(a)



(b)

Figure 47: (a)Dendrite Simulink Block. This is a fully connected block with five inputs, which are the biasing voltages required for the dendritic line and the output port which denotes the voltage at every tap. (b) Block parameter window for the Dendrite Simulink Block. The window asks users to specify input parameters needed for the block. The user is asked to specify the number of stages, the type of FET used (PFET/FG-PFET), if the output should be buffered or not, and the biasing voltages required for the circuit.

Consider a dendritic line as given in Figure 42, with n number of nodes. Current is injected only at the first node and the axial and leakage conductances are the same throughout.

Applying KCL at node 1, the injected current and the bias current are the sum of the axial and the leakage currents. The leakage current comprises of the current through the leakage capacitor and the leakage transistor.

Applying KCL at node 2; the current through the first axial conductance equals the current through the second axial conductance, the leakage conductance, and the leakage capacitance.

Generalizing the ODE for all nodes except the boundary cases, the voltage at the

n^{th} node is given by

$$\begin{aligned} C\frac{dV_n}{dt} &= I_{inj} + k_1(e^{V_{n-1}/U_T} - e^{V_n/U_T}) \\ &\quad - k_1(e^{V_n/U_T} - e^{V_{n+1}/U_T}) \\ &\quad - k_2(e^{V_n/U_T} - e^{E_k/U_T}) + I_{bias} \end{aligned} \tag{17}$$

where, C is the leakage capacitance and

$$k_1 = I_0 e^{((\kappa-1)V_{dd} - \kappa V_{Ax_n})/U_T} \tag{18}$$

$$k_2 = I_0 e^{((\kappa-1)V_{dd} - \kappa V_{Lk_n})/U_T} \tag{19}$$

This is a general expression for k_1 and k_2, however in our experiments V_{Ax} and V_{Lk} are the same throughout. The boundary conditions are as follows, At the first node,

$$\begin{aligned} C\frac{dV_n}{dt} &= I_{inj} - k_1(e^{V_1/U_T} - e^{V_2/U_T}) \\ &\quad - k_2(e^{V_1/U_T} - e^{E_k/U_T}) + I_{bias} \end{aligned} \tag{20}$$

At the last node,

$$\begin{aligned} C\frac{dV_n}{dt} &= I_{inj} + k_1(e^{V_{n-1}/U_T} - e^{V_n/U_T}) \\ &\quad - k_2(e^{V_n/U_T} - e^{E_k/U_T}) + I_{bias} \end{aligned} \tag{21}$$

Writing the equations in vector form is useful as it reduces the time required for Matlab computation. We define all the constants in the equations based on the MOSFETS used on the FPAA (κ, I_0,C) along with the input parameters as defined for the block (V_{Lk},V_{Ax},E_k).

We can re-write these equation in vector form ,

$$\begin{aligned} \frac{d\vec{V}}{dt} = \frac{1}{C}(a_1.I_{inj} &+ k_1(e^{a_2.\vec{V}/U_T} - e^{a_3.\vec{V}/U_T}) \\ &+ k_1(e^{a_4.\vec{V}/U_T} - e^{a_5.\vec{V}/U_T}) \\ &+ k_2(e^{a_6.\vec{V}/U_T} - e^{E_k/U_T}) + I_{bias}) \end{aligned} \tag{22}$$

where

$$\vec{V} = \begin{bmatrix} V_1 & V_2 & V_3 & \dots & V_n \end{bmatrix}$$

a_1,a_2,a_3, a_4, a_5 and a_6 are constant matrices whose size is dependent on the number of stages of the dendrite.

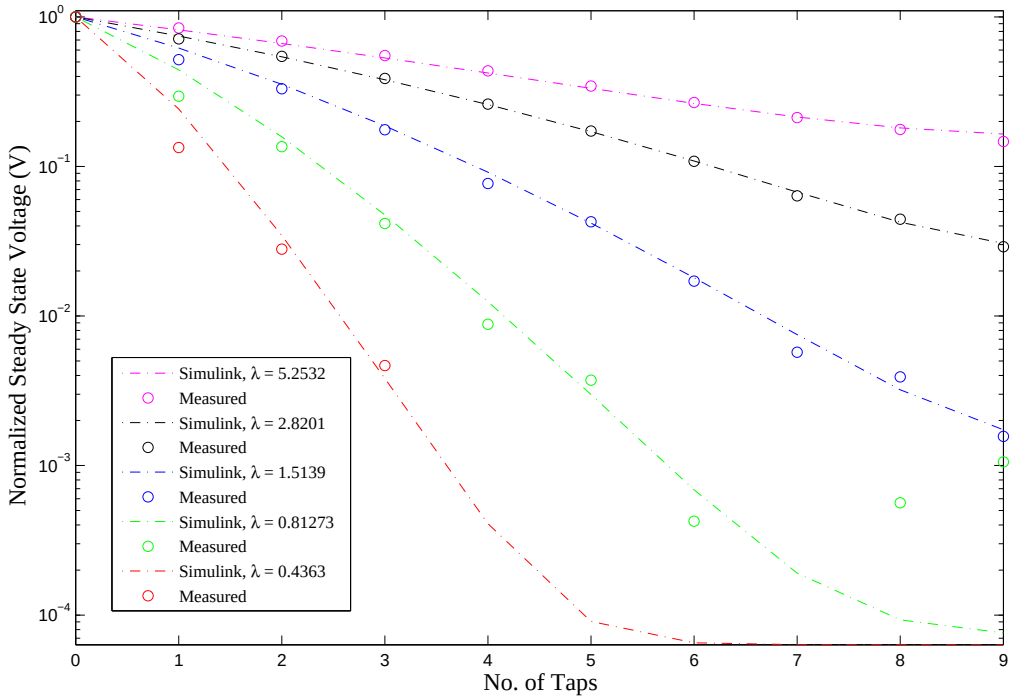


Figure 48: Comparing Simulation results to data obtained using the FPAA. After modifying the I_0 parameter, injected current, and node capacitance, the two normalized curves have similar qualitative behavior.

Results

We simulated a 10-stage dendrite using the Simulink Dendrite block. The nodes are biased at 1.02 V and a current is injected into the first node. The parameters used for the axial and leakage transistors are $\kappa = 0.8464$ and $I_0 = 0.05fA$. For the bias transistors, $\kappa = 0.72$ and $I_0 = 0.45fA$. The node capacitance was 70 pF, and the injected current was 5 pA.

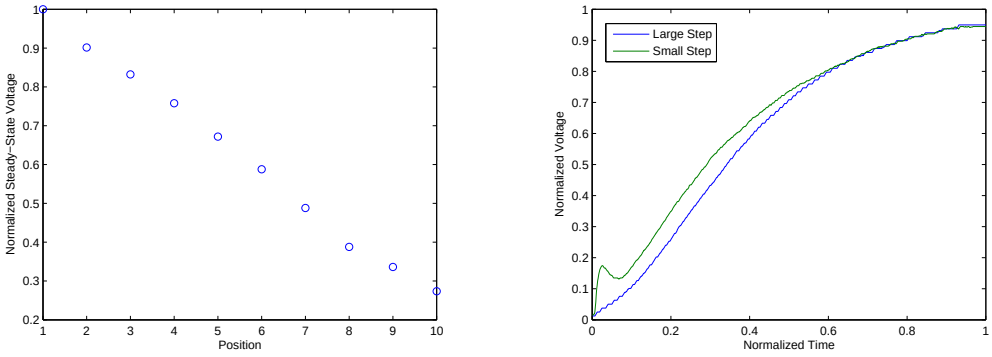
These simulation settings differ from our steady-state experiment in three ways. The input current is different from experiment, the node capacitance is higher than in

experiment, and I_0 differs from the experimental I_0 by one or two orders of magnitude. We believe the higher capacitance was needed in order to allow the simulation to reach its steady-state results more quickly.

Once the above parameters have been changed for best agreement, the average error between the normalized data and the simulation is 16.8%.

5.5 Nonlinear Behavior of Dendrites

Most of this chapter has concerned the behavior of the dendritic circuit operated in its linear regime. When the input current becomes large, however, the qualitative behavior of the circuit changes, and nonlinear effects begin to take hold. Typically, a difference between drain and source of about $4U_T$, or 100 mV is typically considered the nonlinear regime of the dendrite. In order to get a qualitative understanding of the nonlinear effects, we will analyze one "section" of dendrite, shown in Fig. 50.



(a) (b)

Figure 49: (a) When the steady-state response of a 10-stage dendrite is measured with a large input current (causing a change of about 200 mV at the first node), the response is a linear degradation in voltage. (b) Comparing shapes of small step and large step response. The step response was normalized in voltage by dividing by the steady-state value, and time was normalized by finding the point at which the voltage rises to 95% of its steady-state value. The initial response of the small step is more of an RC response, while the large step shows a sigmoidal behavior. See Fig. 46 for a discussion of the transient at the beginning of the small step.

5.5.1 Math Modeling

Applying KCL and the current equations for a capacitor and a saturated transistor,

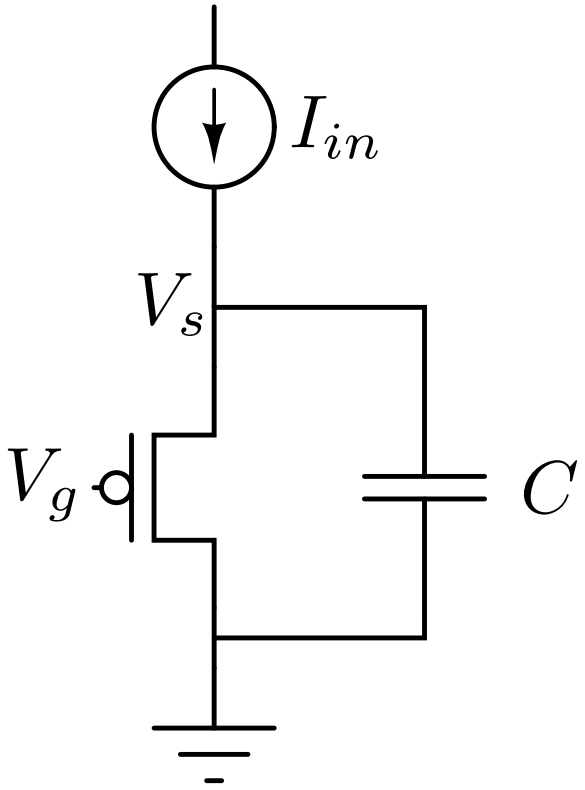


Figure 50: Illustration of nonlinear dynamics in dendrite circuit. A large-signal input current is sent into a node which sees a transistor and capacitor in parallel.

$$\frac{dV_s}{dt} = \frac{I_{in}}{C} - \frac{I_{bias}}{C} e^{V_s/U_T} \tag{23}$$

We can use Eq. 23 to plot a phase portrait. The basic shape is a negative exponential with a vertical offset, shown in Fig. 51.

This portrait gives us quantitative and qualitative information about our circuit's voltage response to an input current. First, it gives us the voltage where we expect V_s to settle:

$$V_s = U_T \ln \frac{I_{in}}{I_{bias}} \tag{24}$$

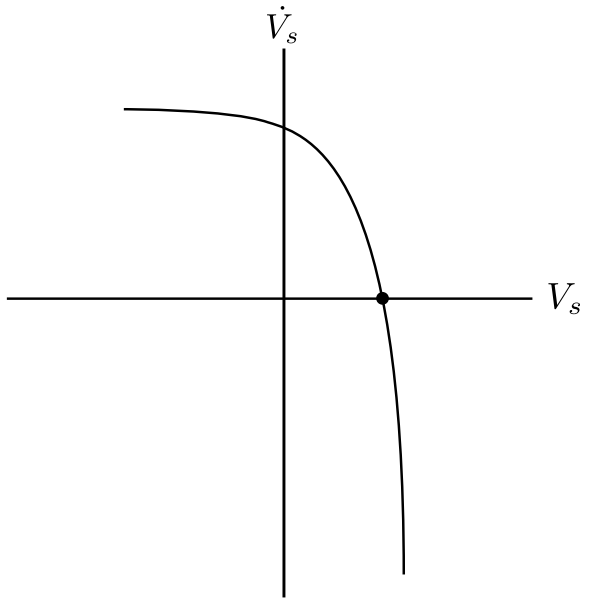


Figure 51: Illustration of the phase portrait resulting from the circuit in Fig. 50. The input current moves the line vertically, which changes the qualitative behavior of the system.

Second, the picture tells us that we will get small time constants for large values of I_{in}. Note from Eq. 23 that the vertical offset of this plot is determined by the value of I_{in}. As I_{in} increases, the plot is shifted up, and the rate at which V_s changes for a given value of V_s will be increased, thus decreasing the time constant. It is also important to point out that the slope of the actual phase portrait is much steeper than what we drew in Fig. 51. This means that a shift up in the plot won't affect the steady-state value of V_s as much as it will affect the time constant.

5.5.2 Demonstration of Impact on Dendrite Circuit Behavior

If we apply a large enough input current such that the membrane voltage changes by more than 100 mV, we can measure the effects of nonlinear input currents on the dendrite.

Our first experiment was to see how the steady-state voltage decays, as shown in 49a. The result is that the voltage decays linearly with space. This is a desirable effect, since it is essentially a compression operation. Recall that, for small inputs,

the steady-state voltage decayed exponentially. If this trend were to continue for large inputs, the dynamic range of available voltages would be severely limited. However, for a large input, the FETs are no longer operating as resistors; they are in saturation, so we merely require linear changes in voltage to achieve exponential changes in current. Therefore the dendrite is using nonlinearity to increase its dynamic range.

Our second experiment is to see how the shape of the step response changes with an increase in input current. We can rewrite Eq. 25 in the current domain. Defining $I_1 = I_{bias} e^{V_s/U_T}$, we can differentiate with respect to time to get $\dot{I}_1 = I_1/U_T \dot{V}_s$. Substituting into Eq. 23,

$$\begin{aligned} I_{in} &= \frac{CU_T}{I_1}\frac{\partial I_1}{\partial t} + I_1 \\ \frac{\partial I_1}{\partial t} &= \frac{I_1}{CU_T}\left(I_{in} - I_1\right) \end{aligned} \tag{25}$$

When Eq. 25 is solved, it behaves like a tanh function, so we expect the shape of our dendrite's step response to be sigmoidal for large current steps. Our results in Fig. 49 bear this out.

5.6 Implementing Dendrites in Large Reconfigurable Systems

Recall that the FPAA connects analog components together using a matrix of floating-gate pFET switches. These FG pFETs can be used as regular transistors, as well, so they can be connected to form floating-gate diffusors. Rather than explicitly apply a gate voltage to the horizontal and vertical transistors, they can be programmed with varying levels of charge, which effectively acts like an applied voltage. The switch matrix must by design be an extremely dense array of switches, so we can make very large dendrites as inputs into neurons.

Difficulties of Floating-Gate Diffusors

Modeling floating-gate denditic circuits is more complicated than with regular FETs.

The reason for this is that the capacitive coupling from the source and drain to the floating-gate is more pronounced than with regular pFETs. In order to design a floating-gate dendrite, therefore, an extra step of characterizing these coupling ratios is necessary. If we desire more complicated behavior by programming different values of the floating-gate voltage for different sections of the dendrite (i.e. changing the dendrite's diameter), we will need to take these coupling ratios into account when determining to what voltage we want to program the floating gate. We need to know coupling ratios because floating-gate transistors are programmed with their terminal voltages at one potential (in "program mode"), and after programming their terminal potentials undergo a change (in "run mode," when the circuit is operating).

Another important nonideality in floating-gate systems which requires characterization is the fact that the transistor which is programmed differs from the transistor which is actually placed in the circuit. This scheme is known as indirect programming, and any differences between the programmed and in-circuit transistor will affect the circuit's performance. Methods to characterize these effects are discussed in [144].

Benefits of Floating-Gate Diffusors

The most exciting aspect of dendritic circuits is that they can be made in an extremely compact manner. As we stated above, the switch matrix of the RASP 2.8a FPAA is completely made up of floating-gate switches. So there is potential to make huge arrays of dendrites using the switch matrix. Since the purpose of the array is to interconnect components, it makes sense that dendrites be used to send signals from one compiled structure to another. Partitioning of the switch matrix allows for a large number of dendrites to be created.

We can estimate how large these dendrites can be based on the FPAA routing structure. Each CAB has an associated floating-gate switch matrix. Some rows and columns are global, meaning they have connectivity among multiple CABs. We will only consider local rows and columns which do not connect beyond a CAB.

In addition, the columns have semi-local connections to their nearest vertical and horizontal neighbors, so we assume that half of those columns are available per CAB. The equivalent number of useful columns per CAB is 14. The rows are hard-wired to CAB elements, so the number of usable rows is reduced to ensure no CAB devices are turned on. For CAB types 1 and 2, the number of available rows is 24 and 34, respectively. If we make a dendrite using the switch matrix, each row connects to one vertical transistor, and each column connects to two horizontal transistors. The size of dendrites in CAB type 1 is limited by its number of rows, while CAB type 2 is limited by columns. Therefore, we estimate that CAB types 1 and 2 can implement dendrites of approximately 24 and 28 stages, respectively. Based on the numbers of these CABs in the FPAA, we can theoretically make 28 dendrites of length 24 and 4 dendrites of length 28. We can then use the global routing to chain some of these together.

It is also important to point out that neural systems are inherently imprecise. Real synapses are very unreliable, and no two dendritic structures are the same. So the disadvantages listed above are not necessarily detriments. Some amount of variability from dendrite-to-dendrite caused by floating-gate transistor mismatch could be seen as a good thing. In fact, the inability to precisely model the behavior could be an asset, for it requires designers to get an intuitive feel for what parameters work well for a given system. The chapter discussed here resulted in a journal paper [19].

CHAPTER VI

DENDRITIC COMPUTATION

Dendrites are highly branched tree like structures that connect neuron's synapses to the soma. They were previously believed to act just like wires and have little or no computational value. However, studies show that dendrites are computational subunits that perform some inherent processing that contributes to overall neural computation [8, 11, 14, 19, 20, 133]. It is thus interesting to explore computational models that can be built using dendrites as a unit.

6.1 Dendrites for Wordspotting

It has been shown that dendrites can perform computations similar to an HMM branch [4, 8] which can be used for wordspotting. Wordspotting is the detection of small set of words in unconstrained speech [13]. The interlink between Neuroscience, CMOS transistors and HMMs is shown in Fig. 52(a).

A typical Wordspotting system has at least three stages: Feature generation, Probability Generation (Signal to symbol conversion) and the State Decoding (classification) stage, which determines the word detected. Fig. 52(b) shows the general block diagram for a classification system. In the specific example of speech recognizer, the sensor would be a microphone stage. The first stage has interface circuitry to acquire the signal from the microphone as well as initial signal processing. This processing may include signal conditioning and filtering, frequency decomposition as well as signal enhancement.

Fig. 53 shows the FPAA as a prototyping device for audio signal processing applications. Our approach to audio processing includes a range of signal processing

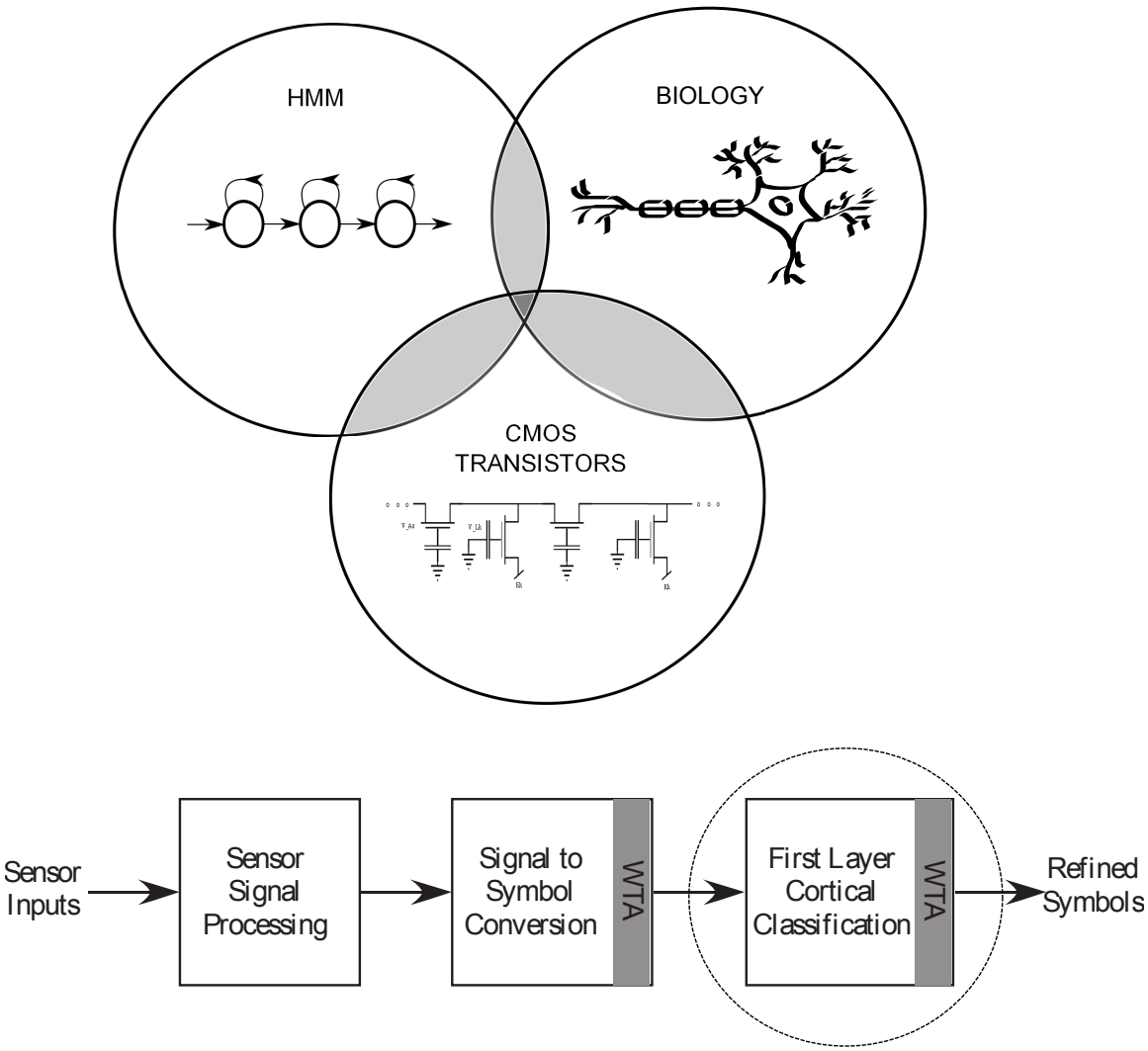


Figure 52: (a) The Venn Diagram depicts the interlinks between the fields of neurobiology, HMM structures and CMOS transistors. We have demonstrated in the past how we can build reconfigurable dendrites using programmable analog techniques. We have also shown how such a dendritic network can be used to build an HMM classifier which is typically used for speech recognition systems. (b) Block Diagram for a Speech/Pattern Recognition system with respect to biology. In a typical speech recognition system, we have an auditory front-end processing block, a signal to symbol conversion block and a state decoding block for classification. We have implemented the state decoding block using dendritic branches, WTA and supporting circuitry for wordspotting. It is the classification stage before which symbols have been generated for a word.

algorithms, that fit into the pathway between speech production (source) and perception (human ear). These algorithms are implemented by non-linear processing of subbanded speech signals for applications such as noise suppression or hearing compensation, by proper choice of the non-linearity. In addition, the outputs of the non-linear

processor can be taken at each sub-band, for speech detection instead of recombining to generate a perceptible signal for the human ear. Using this general framework, a variety of non-linear processing can result in applications in speech classifiers and hearing aid blocks. Here, we focus on the application of speech enhancement by noise-suppression, targeting word recognition in noisy environments. Detailed experimental results for a noise suppression application are discussed in [22, 23], where the speech-enhanced sub-band signals are recombined together. For a speech recognizer, we use the enhanced sub-band signals directly to extract basic auditory features.

The second stage of the speech classifier consists of the probability generation stage that detects basic auditory features and supplies input probabilities to the state decoding stage. These enhanced sub-band signals undergo first-level information refinement in the probability generation stage, resulting in a sparse "symbol" or "event" representation. This stage maybe implemented as an Artificial Neural network (ANN), Gaussian Mixture model (GMM) or a VMM+WTA classifier. A typical 2-layer NN has synaptic inputs represented by the VMM and the sigmoid modeling the soma of a point-neuron. Alternatively, we can have synaptic computation followed by a competitive network modeled by the WTA.

We show in [160] that a single-stage VMM+WTA classifier can be used as a universal approximator, in contrast to an ANN implementation which requires two layers to implement a non-linear decision boundary. Fig. 54 shows the comparison in circuit complexity of a two-layer ANN and a VMM+WTA classifier. A 1-layer NN requires the computation of a Vector-Matrix Multiply (VMM) + neuron. The addition of various weighted inputs is achieved through Kirchoff's Current Law (KCL) at the soma node, adding all currents. The computation at the neuron is governed by the choice of complexity in the model. Usually, for moderate size of the network, the synaptic computation dominates the neuron computation. The sigmoidal threshold block for the soma nonlinearity in a NN can be implemented in voltage mode by converting

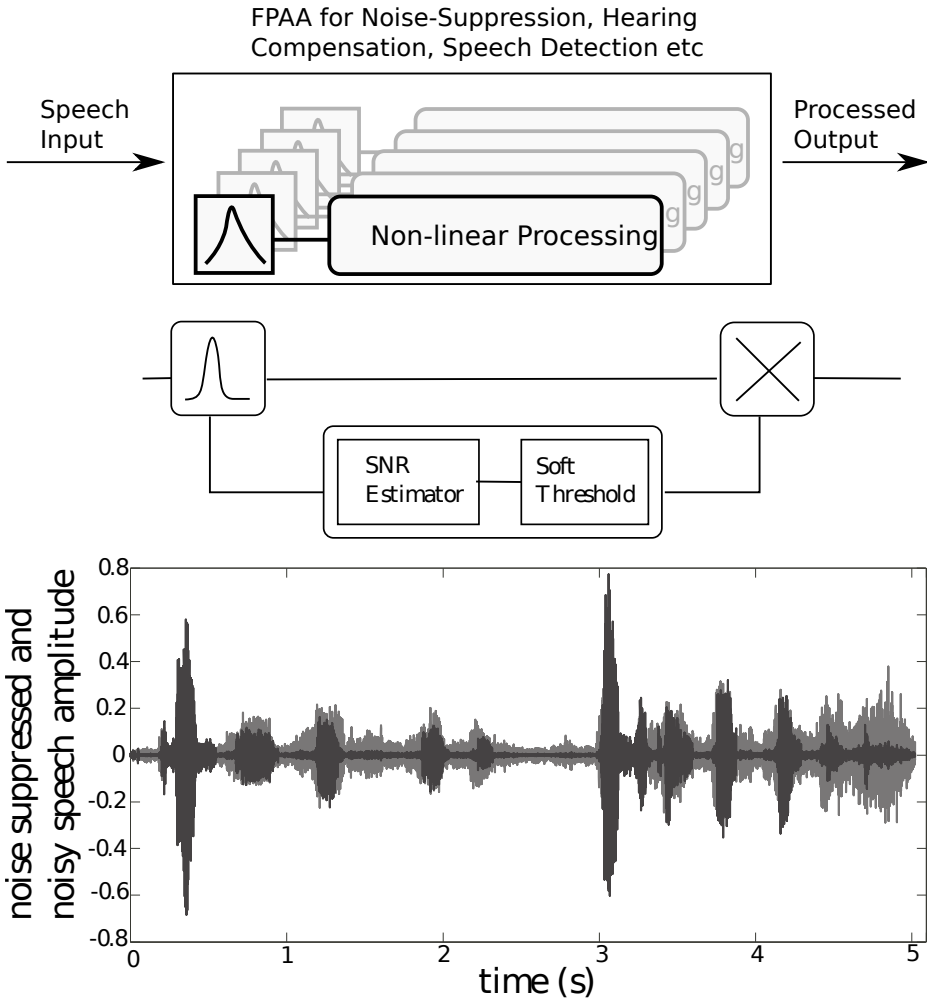


Figure 53: **High level overview:** The FPAA can be used for a variety of audio processing applications using the signal framework described. The first stage is a frequency decomposition stage followed by a non-linear processing block. The non-linear circuit can be used to implement the SNR estimator and a soft-threshold, which sets the gain in each sub-band. The gain control is implemented using a multiplier. Transient results from a MATLAB simulation of a 4 channel system is plotted. The noisy speech is gray, while the processed speech is in black.

the current output from the VMM into voltage and using a voltage-mode threshold block, or in current mode with an $arcsinh(.)$ block. All of these implementations require more transistors per neuron compared to a WTA, which requires as few as 2 transistors per neuron.

The VMM+WTA classifier topology has the advantage of being highly dense and low power. Each multiply is performed by one single transistor that stores the weight

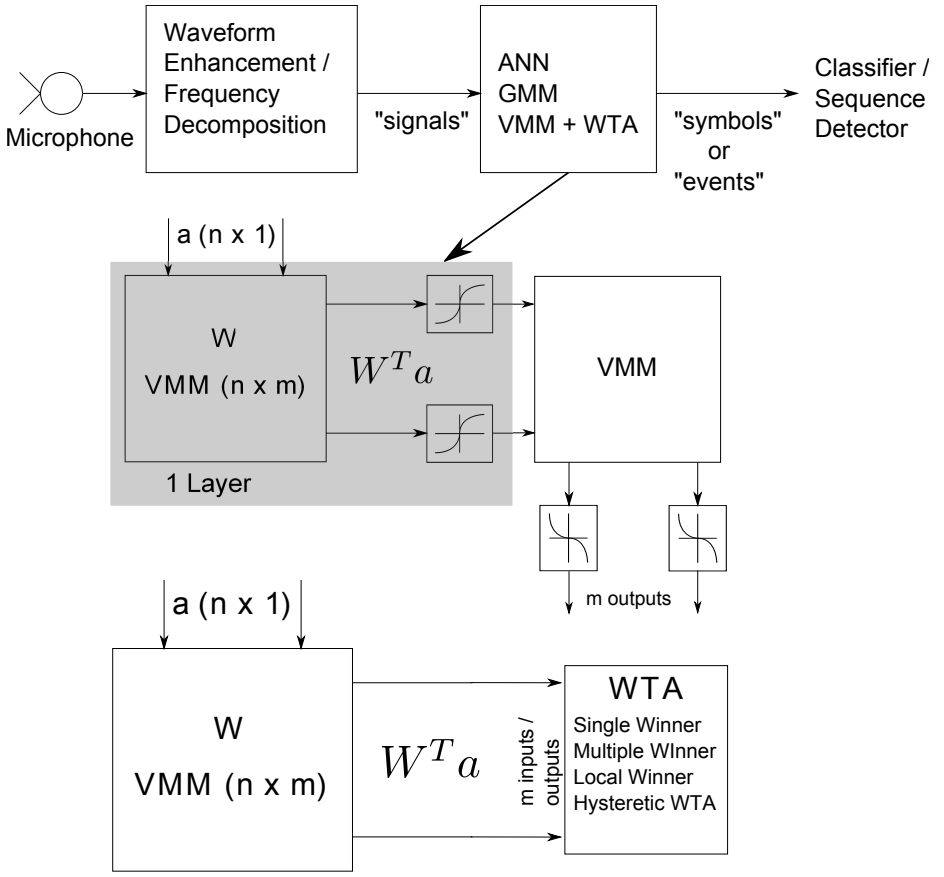


Figure 54: **Basic auditory feature extraction and probability generation stage:** The speech input undergoes frequency decomposition or enhancement resulting in sub-band signals. The probability generation block can be implemented using an ANN, GMM or the VMM+WTA classifier. The circuit complexity is halved by using a VMM+WTA classifier.

as well, and each WTA unit has only 2 transistors, providing very high circuit density. Custom analog VMMs have been shown to be 1000X more power efficient than commercial digital implementations [162]. The non-volatile weights for the multiplier can be programmed allowing flexibility. The transistors performing multiplication are biased in deep sub-threshold regime of operation, resulting in high computing efficiency. We combine these advantages of VMMs with the reconfigurability offered by FPAA platforms to develop simple classifier structures.

In this chapter, we demonstrate the state decoding stage of a simple YES/NO wordspotter. We have implemented an HMM classifier using biophysically based

CMOS dendrites for state decoding. For all experimental results in this chapter, it is assumed that we have the outputs of the feature and probability generation stages.

We shall describe an HMM classifier model and its programmable IC implementation using CMOS dendrites. The first part of this chapter describes the similarity between a single dendritic branch and HMM branch, in addition to exemplifying its usage to compute a metric for classification. An HMM classifier is modeled comprising of these dendritic branches, a Winner-Take-All (WTA) circuit and other supporting circuitry. Subsequently, the computational efficiency of this implementation in comparison to biological and digital systems is discussed. Intriguingly, this research substantiates the propensity of computational power that biological dendrites encompass, allowing speculation of several interesting possibilities and impacts on neuroscience. It is in some ways a virtual visit into the dendritic tree as was suggested by Segev et al [29]. This chapter further explores the interlinks between neurobiology, Hidden Markov Models and CMOS transistors based on which we can postulate that a large group of cortical cells function in a similar fashion as an HMM network [4,19]. Section II describes the similarities between a dendrite branch and an HMM branch. We discuss the similarities between a simulated HMM branch and experimental results using a CMOS dendrite branch. In Section III, we discuss the single CMOS dendrite in detail. We will present experimental results for the line for different parameters. We also discuss the simulation model that we have developed and the similar results seen. In section IV, we discuss the Analog HMM classifier implementation. We discuss the experimental results for a YES/NO wordspotter for different sequences. In section V, we discuss the tools that made the implementation of this classifier structure possible. In section VI, we will discuss the computational efficiency of the system as compared to digital and biological systems. In the final section we will summarize the results and discuss the future possibilities.

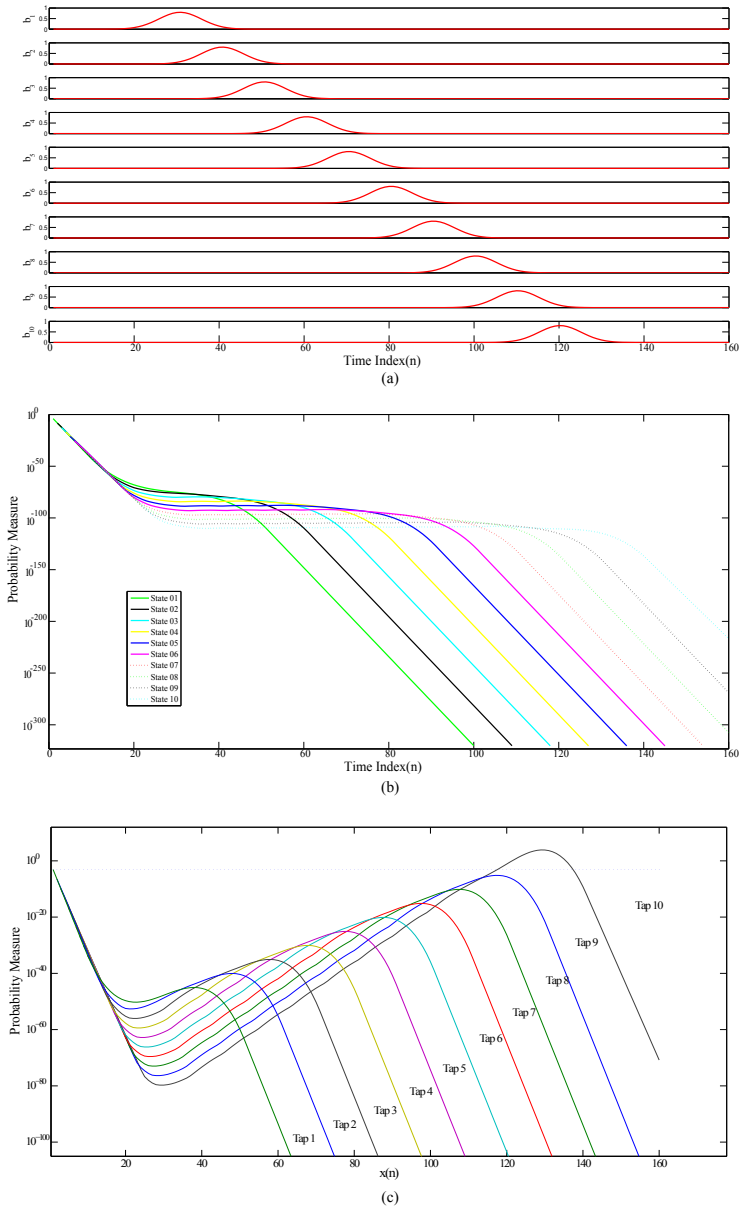


Figure 55: Simulation results for an HMM state machine based on a Mathematical HMM model built using MATLAB (a) Input probability distribution of different symbols varying with time. (b) Likelihood outputs of all the states on a logarithmic scale. (c) Normalized likelihood outputs of all the states. The outputs were normalized by multiplying them with an exponential function of the form $exp(n/\tau)$

6.2 Dendritic computation and the HMM branch

For a typical HMM used for speech recognition, the update rule is given by:

$$\phi_i[n] = b_i[n]((1 - a_i)\phi_i[n-1] + a_{i-1}\phi_{i-1}[n-1]) \tag{26}$$

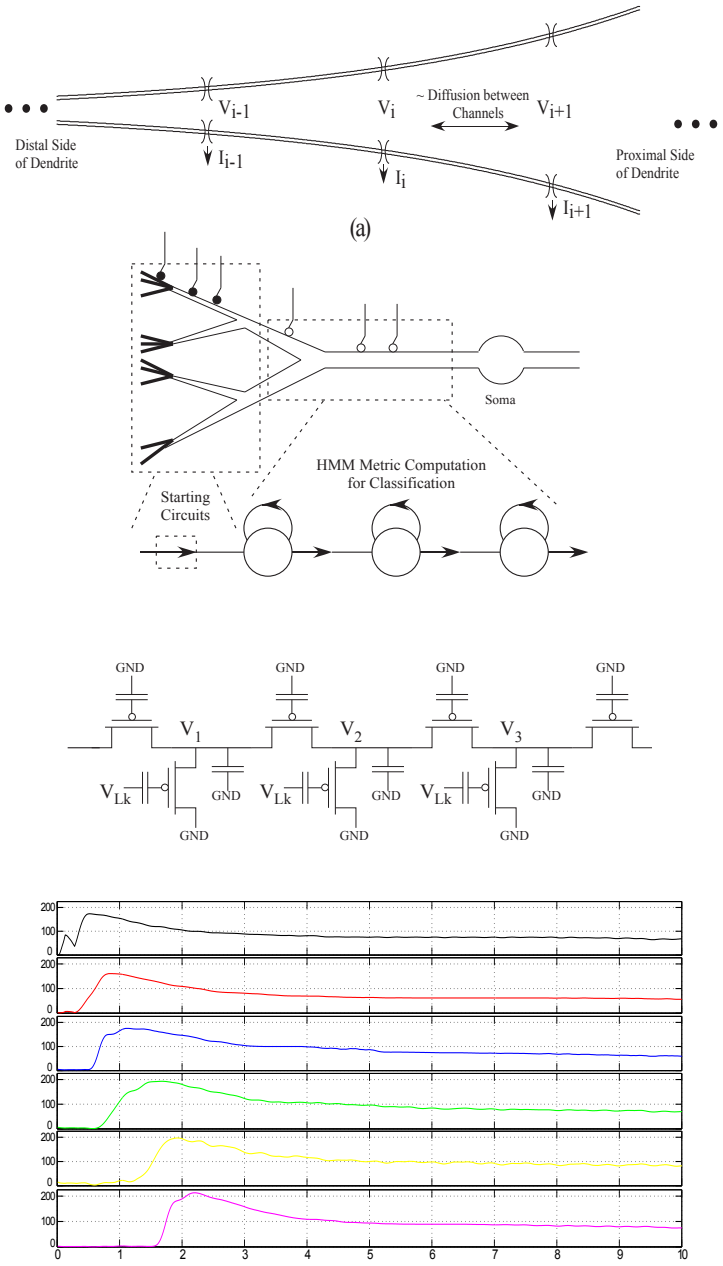


Figure 56: CMOS implementation for a dendritic branch and experimental results. (a) Dendrite with increasing diameter as typically seen in pyramidal cells. We refer this increasing diameter as 'taper'. (b) Co-relation between an HMM branch and a CMOS dendrite branch with 'taper'. (c) Resulting IC implementation using programmable analog floating-gate pFETs. For the CMOS dendrite the 'taper' is modeled by increasing the axial conductance. (d) Experimental results showing the outputs from each tap of the CMOS dendrite. These outputs are equivalent to likelihood outputs from the HMM states. The output doesn't decay completely but attains a new dc level. Note that we did not do normalization explicitly for the outputs of the dendrite as the decay is not as sharp as seen in HMMs. All taps are set initially to have the same membrane voltage V_{mem}.

The probability distribution $b_i[n]$, represents the estimate of a symbol (short segment of speech/phoneme) produced by a state i in frame n. $\phi_i[n]$ represents the likelihood that a particular state, was the end-state in a path of states that models the input signals [12] as shown in (26). a_i is the transition probability from one state to another. In a typical speech recognition model, the states would be phonemes/words and the output would represent the audio signal produced by the subject. The features of the audio signal tend to vary for different subjects. The goal of this classifier model is to correctly classify a sequence of symbols with some tolerance. For an HMM state machine for speech recognition using CMOS dendrites, the inputs $b_i[n]$ can be modeled as Gaussian inputs as shown in Fig. 55a, which is typical for $b_i[n]$ for speech signals with an exponential rise-time and fall-time. In Fig. 55b, the likelihood outputs for each state shows a very a sharp decay and has a very high dynamic range. To limit this range, we normalize this output with an exponential function. It can be observed that the normalized likelihood is similar to an EPSP signal with an asymmetric rise and fall time. For a single n-stage dendritic line with 'taper', if we applied sequential EPSP inputs at subsequent nodes, the output observed at the end of the line is as shown in Fig. 56d. A 'taper' signifies the changing diameter through the length of a dendrite. It represents the normalized likelihood outputs of an HMM classifier. The Gaussian inputs for the HMM model can be modeled using synaptic currents for a dendrite which is also typical for biological systems. $b_i(t)$ is thus represented as the synaptic current into each node. The output voltage of each tap of the dendrite represents the likelihood $\phi_i(t)$ of an HMM state. This can be linearly-encoded or log-encoded depending on the region of operation. For the dendritic system, no normalization is done as the decay is not as sharp as seen in the HMM branch for a wide dynamic range.

For a continuous-time version of (26), the update rule is given by,

$$\phi_i(t) = b_i(t)((1 - a_i)\phi_i(t - \tau) + a_{i-1}\phi_{i-1}(t - \tau)) \tag{27}$$

where, $b_i(t)$ is the input probability of symbol in state i and $\phi_i(t)$ is the likelihood of a state i at time t, τ is the time index between two consecutive time indexes and a_i is the transition probability between adjacent states. Even though the state sequence is implied, one cannot assume a definitive observation of transition between the states. This is the reason why it is called *Hidden* Markov Model although the state sequence has a Markovian structure [10]. Continuous-time HMMs can be represented as a continuous-time wave-propagating PDE as given in (28) [6].

$$\underbrace{\tau\frac{\partial\varphi\left(x,t\right)}{\partial t}}_{\substack{\text{state}\\\text{element}}}+\underbrace{\left(\frac{1}{b\left(x,t\right)}-1\right)\varphi\left(x,t\right)}_{\substack{\text{decay}\\\text{term}}}+\underbrace{a\left(x\right)\Delta\frac{\partial\varphi\left(x,t\right)}{\partial x}}_{\substack{\text{wave}\\\text{propagation}}}=0 \tag{28}$$

where, Δ is the distance between two state nodes. This can be compared to analog diffuser circuits. Also, an HMM branch and a dendrite branch have similar looking topologies and similar wave-propagating properties. The HMM state machine used, as shown in Fig. 55a, is a left-to-right model. Studies have shown that a biological dendrite also does not have a constant diameter [18]. Its diameter at the distal end is smaller as compared to the proximal end as shown in Fig. 56a and Fig. 56b [3]. Thus, for a similar CMOS dendritic line that is uni-directional, we would expect the axial conductances of the line to increase from left-to-right as shown in Fig. 56c. This is the case of a dendrite with 'taper'. Such a topology ensures that the current flow is uni-directional. This also favors coincidence detection in the dendrite. We can compare the continuous-time HMM to an RC delay line with 'taper'. For this let us analyze the behavior of an RC delay line with and without taper.

6.2.1 RC delay line without taper

The classical RC delay line is reviewed in Mead's text [103]. Fig.57 shows the topology. Kirchoff's Current Law (KCL) can be used to derive a differential equation for this circuit, given by (29), where G is conductance.

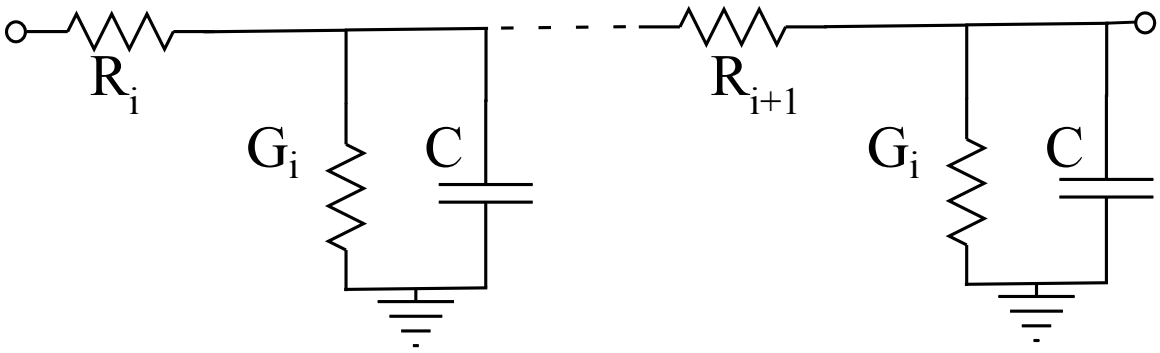


Figure 57: RC delay line representing a dendrite. The Rs represent the axial resistances, the Gs represent the leakage conductances and C is the membrane capacitance.

$$\begin{aligned} I_i(t) = C_i \tfrac{dV_i(t)}{dt} + V_i(t) G_i + \tfrac{[V_i(t) - V_{i+1}(t)]}{R_{i-1}} \\ + \tfrac{[V_i(t) - V_{i-1}(t)]}{R_i} \end{aligned} \tag{29}$$

Assuming the horizontal resistances are equal as given in (30) allows one to simplify (29) to (31):

$$R_i = R_{i-1} = R_x \tag{30}$$

$$\begin{aligned} I_i(t) = C_i \tfrac{dV_i(t)}{dt} + V_i(t) G_i \\ + \tfrac{1}{R_x} [2V_i(t) - V_{i+1}(t) - V_{i-1}(t)] \end{aligned} \tag{31}$$

Assuming there are many nodes allows one to perform the following change of notation from discrete nodes to continuous nodes:

$$V_i(t) = V(x,t) \tag{32}$$

$$V_{i+1}(t) = V(x + \Delta_x, t) \tag{33}$$

$$V_{i-1}(t) = V(x - \Delta_x, t) \tag{34}$$

Assuming that Δ_x represents a "position delta" one may use the Taylor series to describe the continuous nodes in terms of Δ_x, (35), (36).

$$V(x + \Delta_x, t) = V(x,t) + \Delta_x \frac{dV(x,t)}{dx} + \frac{1}{2} (\Delta_x)^2 \frac{d^2V(x,t)}{dx^2} + \cdots \tag{35}$$

$$V(x-\Delta_x,t) = V(x,t) - \Delta_x \frac{dV(x,t)}{dx} + \frac{1}{2}(\Delta_x)^2 \frac{d^2V(x,t)}{dx^2} + \cdots \tag{36}$$

Substituting (35) and (36) into (31) and simplifying, yields (37), the generalized PDE describing the RC delay line diffusor.

$$I_i(t) R_x = R_x C_i \frac{dV_i(t)}{dt} + R_x G_i V_i(t) - (\Delta_x)^2 \frac{d^2V(x,t)}{dx^2} \tag{37}$$

If one assumes no input current at the top of each node $I_i = 0$, then one can put the diffusor circuit into a form similar to the continuous time HMM equation as given in (38).

$$\underbrace{R_x C_i \frac{dV(x,t)}{dt}}_{\substack{state \\ element}} + \underbrace{R_x G_i V(x,t)}_{\substack{decay \\ term}} - \underbrace{(\Delta_x)^2 \frac{d^2V(x,t)}{dx^2}}_{\substack{diffusion \\ term}} = 0 \tag{38}$$

The impulse response of such a system is a Gaussian decaying function over time. In this case, diffusion is the dominant behavior of the system.

6.2.2 RC delay line with taper

Assuming that HMM will always propagate to the next state and there is no probability that it will remain in its current state leads to the assumption as given in (39) which can be substituted in (28):

$$a(x) = 1 \tag{39}$$

For a dendrite circuit with taper, axial conductances are NOT equal and increase towards the right. Using this assumption, (29) simplifies to (40):

$$I_i(t) = C_i \frac{dV_i(t)}{dt} + V_i(t)\left[G_i + \frac{1}{R_i}\right] - \frac{V_{i-1}(t)}{R_i} \tag{40}$$

Substituting the Taylor series expansions of ((35)) and ((36)) into the above we

get:

$$I_i(t) = C_i \frac{dV(x,t)}{dt} + V(x,t)\left[G_i + \frac{1}{R_i}\right] - \frac{1}{R_i}\begin{bmatrix} V(x,t) \\ -\Delta_x \frac{dV(x,t)}{dx} \\ +\frac{1}{2}(\Delta_x)^2 \frac{d^2V(x,t)}{dx^2} \end{bmatrix} \tag{41}$$

Assuming that

$$\Delta x \ll 1 \tag{42}$$

we can neglect higher order terms of the Taylor series.

$$(\Delta_x)^2 \approx 0 \tag{43}$$

We can see in (41) that there is still some diffusion that can be seen in the line. It is however negligible as the wave propagation term is more dominant. Re-arranging terms and assuming no input current we get:

$$0 = \underbrace{R_i C_i \frac{dV(x,t)}{dt}}_{\substack{state \\ element}} + \underbrace{V(x,t)\left[G_i R_i - 1\right]}_{\substack{decay \\ term}} + \underbrace{\Delta_x \frac{dV(x,t)}{dx}}_{\substack{wave \\ propagation}} \tag{44}$$

Table 1 closely examines the similarities between a RC delay line and an HMM PDE.

Table 1: Comparing HMM PDE and RC Delay Line Terms w/Assumptions

Element Description	HMM PDE	RC Delay Line
Recursion Variable	$\varphi(x,t)$	$V(x,t)$
State Element Coefficient	τ	$R_i C_i$
Decay Term Coefficient	$\frac{1}{b(x,t)} - 1$	$G_i R_i - 1$
Wave Propagation/Diffusion Term	$K\frac{\partial \varphi(x,t)}{\partial x}$	$K\frac{dV(x,t)}{dx}$

In Fig. 59, we have studied the trends that one would observe collectively for different parameters. The output metric here is the difference of amplitude of last node when all inputs are present and when only the last input is present. We observed that as we increased the timing difference between various inputs, the final metric of

the line decreased as seen in Fig. 59b. We simulated the dendritic branch to observe the effects a wide range of time delays between inputs as shown in Fig. 59c. We observed that the output metric decreased as we increased the delay between the inputs for a line. And for the cases where we reversed the sequence, the amplitude was very close to zero. This clearly demonstrates that if the sequence of the inputs is not in succession, there will be no word detection. Also, the output metric decreases as the delay between the inputs increases.

6.3 CMOS Dendrite

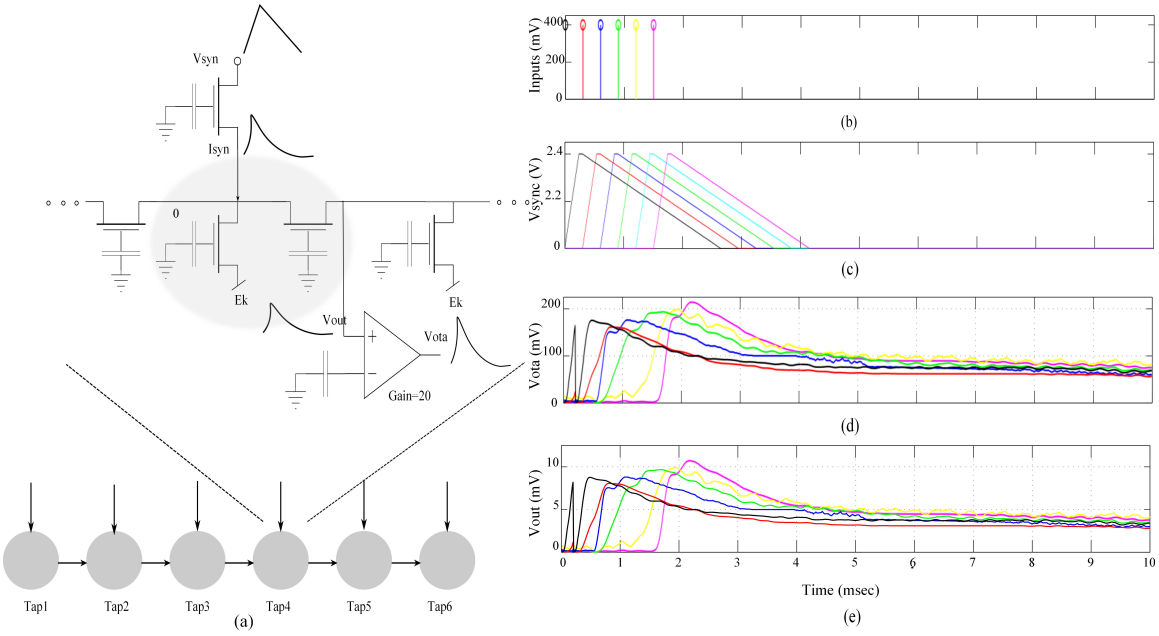


Figure 58: (a) Detailed diagram for a single dendritic line (b) The representation of input voltage on the source of the transistor representing the input synapses (c) The asymmetric triangular input voltages V_{syn} on the source of the transistor representing the input synapses. I_{syn}, the input synapse currents into each of the different nodes is proportional to V_{syn} (d) V_{ota}, the output of FG-OTA which has a gain of approximately 20. (e) V_{out}, the output voltage at each node.

Fig. 60 shows a complete overview of how CMOS dendrites are modeled and also the experimental results for a 6-compartment CMOS dendrite. We implemented a single 6-compartment dendrite. Each compartment consisted of 3 FG pFETS for the axial conductance, the leakage conductance and the synaptic input respectively.The

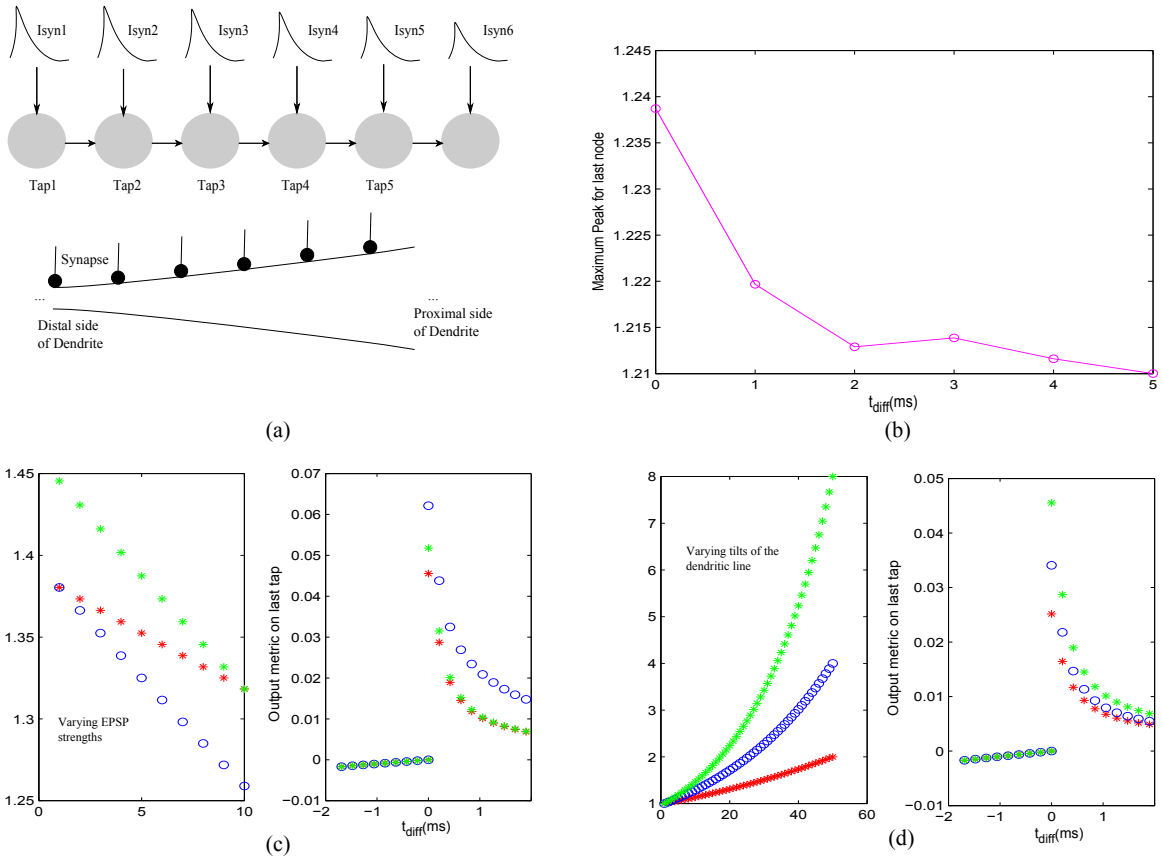


Figure 59: We varied the input sequence with respect to the time difference between signals. The output metric in this case is the difference between the output of the dendrite when all signals were present and output of the dendrite when only the last input was present. (a) Diagram depicting the decreasing EPSP inputs into a single CMOS dendrite line. (b)Experimental data showing change in peak to peak amplitude for a dendrite as the EPSP inputs into each of the nodes decrease down the line. (c) Change in amplitude of the output with respect to increasing difference in the EPSP amplitudes as we progress from left to right down the line. t_{diff} implies the time delay between inputs. As we increase the time delay the output metric reduces. Negative t_{diff} implies a reversed sequence of inputs, where the output metric is zero. (d) Change in amplitude of the output with respect to increasing difference in the taper of the dendrite. In this experiment, the diameter of the dendrite was increased as we progress from left to right down the line. t_{diff} implies the time delay between inputs. As we increase the time delay the output metric reduces. Negative t_{diff} implies a reversed sequence of inputs, where the output metric is zero.

inputs to the dendrite are synaptic currents. In biological systems, synaptic inputs can be excitatory and inhibitory in nature. However, in our experiments we use excitatory synapses as a majority of contacts on a pyramidal cell are excitatory in

nature. The dendrite does not have a constant diameter. This implies that for a CMOS dendrite, the conductance of the dendrite increases towards the soma i.e. from left to right [3]. We will hereon call it 'taper'. Also the EPSP strengths near the distal end are larger than the EPSP strengths near the proximal end. Evidence for the same has been shown in biology [18]. It was observed that as the difference in amplitude was increased, the amplitude of the output reduced. To test the behavior of dendrites, we varied three parameters namely: 'taper', delay between inputs and the EPSP strengths of the synaptic inputs. In terms of 'taper', two approaches were tested. One without 'taper' and the second with increasing 'taper'.

6.4 Dendrites: Behavioral Modeling

We developed a Simulink block to simulate the behavioral characteristics of the CMOS dendrite structure using the given inputs. This provides the user an insight to the working of the dendritic circuit when implemented using the FPAA. The MOSFET parameters used are based on the MOSFETS present on the FPAA. It is characterized by coupled ordinary differential equations (ODE) and solved using the ODE solver ode-45. The model has been tested for both static as well as time-varying inputs and has given reasonable results. I present below the mathematical model used, based on the device physics of silicon.

In Fig. 59, we have studied the trends that one would observe collectively for different parameters of a dendrite. The output metric here is the difference of amplitude of last node when all inputs are present and when only the last input is present. We observed that as we increased the timing difference between various inputs, the final metric of the line decreased as seen in Fig. 59b. We simulated the dendritic branch to observe the effects a wide range of time delays between inputs as shown in Fig. 59c. We observed that the output metric decreased as we increased the delay between the inputs for a line. Here one observes coincidence detection and nonlinear summation.

And for the cases where we reversed the sequence, the amplitude was very close to zero. This shows directional selectivity.

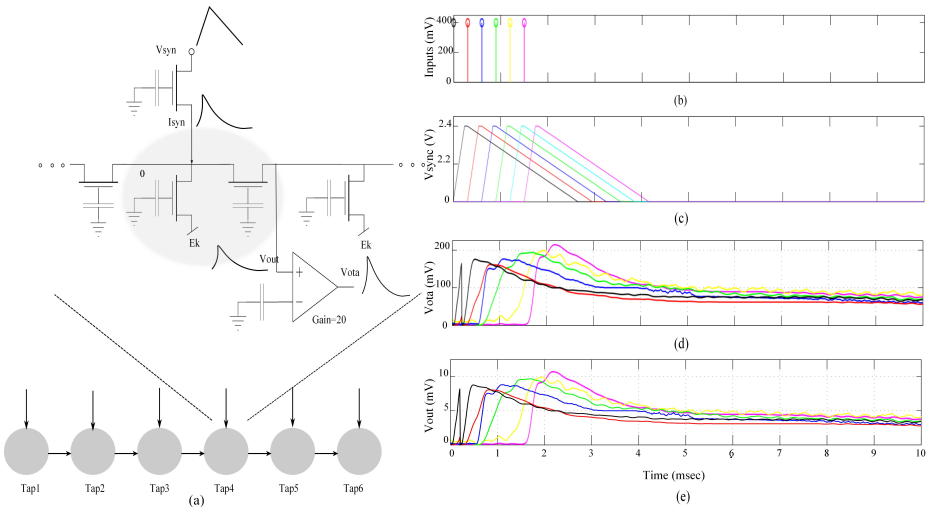


Figure 60: System overview for a dendrite branch. (a) Detailed diagram for a single dendritic line which is equivalent to an HMM branch (b) The representation of input voltage on the source of the transistor representing the input synapses (c) The asymmetric triangular input voltages V_{syn} on the source of the transistor representing the input synapses. I_{syn}, the input synapse currents into each of the different nodes is proportional to V_{syn} (d) V_{ota}, the output of FG-OTA which has a gain of approximately 20. (e) V_{out}, the output voltage at each node.

6.5 Single Line CMOS dendrite

Since dendrites have computational significance, it is interesting to explore computational models that can be built using dendrites or a network of dendrites. One such application is classification in speech recognition. We have already discussed the similarities between an HMM branch and a dendritic branch. To test this hypotheses, we implemented a single dendritic branch with spatially temporal synaptic inputs. We compared a single CMOS dendritic branch implemented on a reconfigurable analog platform and a MATLAB Simulink simulation model based on the device physics of CMOS transistors. Fig. 60 shows a complete overview of how CMOS dendrites are modeled and also the experimental results for a 6-compartment CMOS dendrite. The inputs to the dendrite are synaptic currents. In biological systems, synaptic inputs

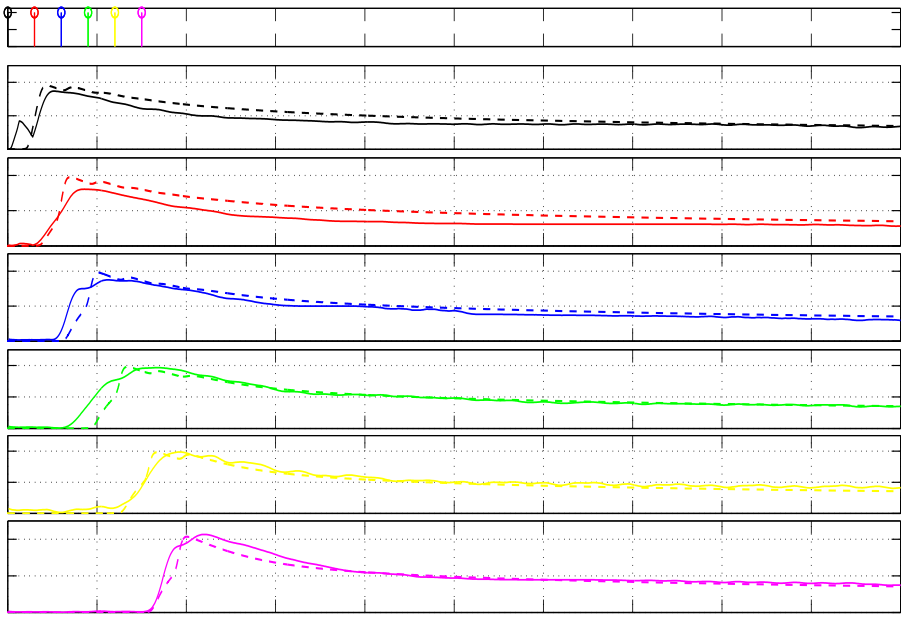


Figure 61: Simulation Data vs. experimental data comparison. The dotted lines depict the simulation data and the solid lines are the experimental data. The parameters for simulation data are $V_{Leak} = 0.5V$, $V_{axial} = 0.5V$, $\kappa = 0.84$, $I_0 = 0.1fA$, $C = 1.3pF$, $E_k = 1V$, $V_{dd} = 2.4V$

can be excitatory and inhibitory in nature. However, majority of contacts on a pyramidal cell are excitatory in nature. As discussed before the dendrite does not have a constant diameter. This implies that for a CMOS dendrite, the conductance of the dendrite increases towards the soma i.e. from left to right [3]. The inputs will also decrease in amplitude as conductance increases. This ensures that an input closer to the soma does not have a larger effect than inputs farther away. This indicates decreasing synaptic strengths of inputs down the dendritic line. This has been observed previously in biological dendrites [18]. Thus, we also varied the synaptic strengths of inputs in our experiments. We implemented the single dendritic line both as a CMOS circuit model and a MATLAB Simulink simulation model. We found that the comparison of our experimental and simulation results were fairly close. This is demonstrated in Fig. 61.

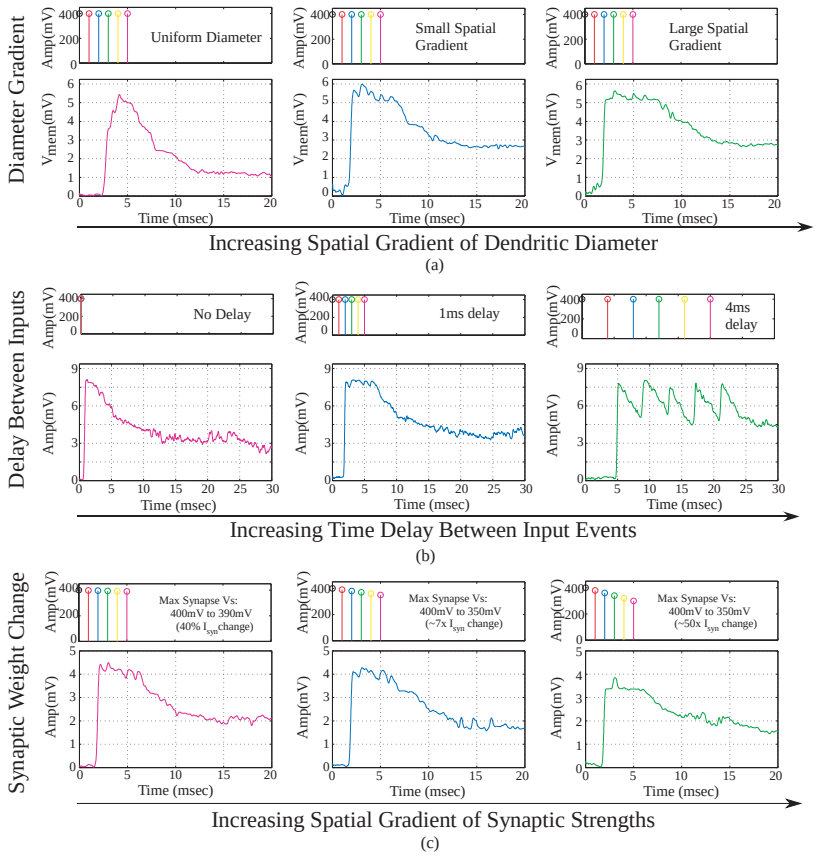


Figure 62: Experimental results for a single branch 6-tap dendrite for different parameters. The three main parameters that govern the output of a dendrite are, namely the taper of the line, the spatial-temporal characteristics of the synaptic inputs and the strength of the synaptic inputs. All results are from the last tap of the dendrite. (a) Metric changed is the taper of the dendrite. For subsequent figures, the taper is increased from no taper to a larger taper. The diameter of the dendrite increases down the line which is achieved by increasing the conductances of the axial transistors from left to right(b)Metric changed is the delay between EPSP inputs into each of the taps of the dendrite. In the first case we have zero time unit delay, 10 time units delay ($2ms$) for second and 20 time units delay ($4ms$)for the third diagram in the sequence. One time unit=$0.2ms$ (c)Metric changes is the difference between the EPSP strengths of the input signals. In the first case, the difference is 10 mV, 50 mV for the second and 100 mV for the third case. As can be seen in the graph we can see decreasing amplitude as the difference in EPSP strengths increases

6.5.1 Inputs to the PFET source

The input probabilities $b_i(t)$ are represented as log-compressed voltage signal at the dendrite node. To generate EPSP input currents into each of the dendritic nodes, we input an asymmetric triangular wave voltage at the source of the pFET FG-FETs.

This generates typical EPSP signals, which have a faster rising time and a slower fall time. By varying the magnitude of the triangular waves we were able to control the input current into each of the nodes of the dendrite. This can be seen in Fig. 60c. The current of a transistor is exponentially proportional to its source voltage V_S.

$$I_{syn} = I_0 e^{\kappa(V_S - V_G)/U_T}(e^{-(V_S - V_D)/U_T} - 1) \tag{45}$$

where, $V_S = V_{dd}$. This enables us to generate EPSP-like inputs for the CMOS dendrite. All input representations shown thus are voltage inputs on the source of the transistor, that acts as synapse at every node of the dendrite.

6.5.2 Single line dendrite results

We implemented a single 6-compartment dendrite. Each compartment consisted of 3 FG pFETS for the axial conductance, the leakage conductance and the synaptic input respectively. We present experimental results for the same. To test the behavior of dendrites for a typical speech model, we varied three parameters namely: 'taper', delay between inputs and the EPSP strengths of the synaptic inputs. In terms of 'taper', two approaches were tested. One without 'taper' and the second with increasing 'taper'. Results are shown in Fig. 62a. We observed that by using 'taper' we could ensure that the input current would transmit more in one direction of the dendritic cable. To achieve this we increased the axial conductance of the cable down the line, such that maximum current tends to flow to the end of the cable. At every node of the dendrites we input EPSP currents in a sequence. This is similar to a speech processing model, where all the phonemes/words are in a sequence and based on the sequence we classify the word/phoneme. We then varied the delay between the input EPSP signals as seen in Fig. 62b. It was observed that as the delay between the inputs increases, the amplitude of the output decreases. This implies that as outputs are spaced farther apart, there is less coincidence detection. The third parameter varied was the strength of the EPSP inputs, with the difference in EPSP strengths

of the first node and the last node increasing for subsequent plots as seen in Fig. 62c. The EPSP strengths near the distal end are larger than the EPSP strengths near the proximal end. Evidence for the same has been shown in biology [18]. It was observed that as the difference in amplitude was increased, the amplitude of the output reduced. The study of the variation of these parameters showed the robustness that such a system would demonstrate in terms of speech signals. The difference in delay, models the different time delays between voice signals when a word is spoken by different subjects. The difference in EPSP strengths ensures that the impact of all the inputs to the classifier on the output will be similar for classification and not dominated by just the last stage.

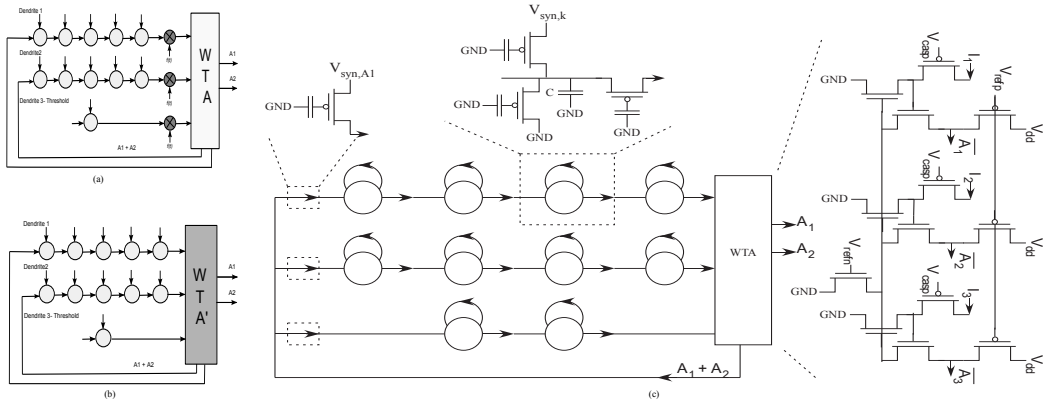


Figure 63: (a) The classifier structure with the normalization factor multiplied, $f(t) = e^{t/\tau}$. (b) The classifier structure after normalization. This figure demonstrates that the normalization is inherent in the system. (c) Detailed structure of the HMM classifier using reconfigurable floating-gate devices. There are three main structures here : The dendrite branches, the Winner-Take-All circuit and the supporting circuitry. The dendrite branch consists of a 5-stage dendrite for both the branches representing the words YES and NO; and a single stage dendrite to set the the threshold current. The dendrites have synaptic inputs at each node, which represent the phonemes of the word to be detected. When the output of a dendrite exceeds the threshold limit i.e. if a YES/NO is detected, the threshold loses. The supporting circuitry consists of a Vector-Matrix Multiplier (VMM) building block which acts as a reset after a word is spotted [162].

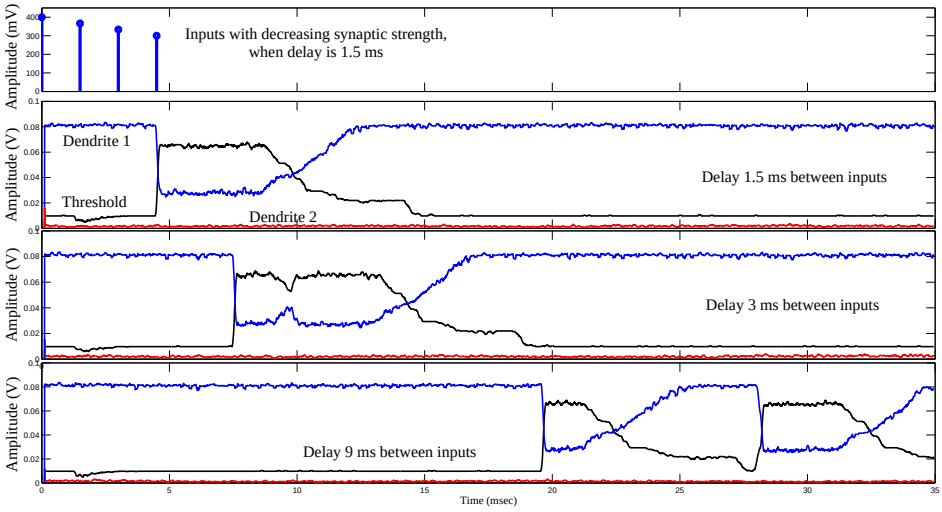


Figure 64: Experimental results for the classifier system with different timing delays for inputs and varying synaptic strength. It demonstrates the effect of different timing in sequences. This implementation is for a classifier system with 4-tap dendrites, 3-input WTA and supporting circuitry. The WTA outputs for varying delays between inputs is shown. The delay between subsequent inputs is $1.5ms$, $3ms$ and $9ms$ respectively. Also the inputs have different EPSP current strengths.

6.5.3 Dendrite on the routing fabric

We used floating gate pFET switches to build the network of dendrites. This would also enable us to build denser networks as we scale the system. In our current system implementation for a single dendrite, we implemented 5 dendritic compartments, with each compartment consisting of 3 floating gate transistors. The most exciting aspect of implementing dendritic circuits using floating-gates is, that we can do so in a very compact manner. As stated above, the switch matrix of the RASP 2.8a FPAA is completely made up of about 50,000 floating-gate elements. Thus huge arrays of dendrites can be made using the switch matrix. Its inherent function is to interconnect components, which is similar to the function of dendrites that are used to transmit signals from one structure to another. Modeling dendritic circuits using floating gates, however has a few complications. The reason being the capacitive coupling from source and drain to the floating gate is more pronounced than regular pFETs [19]. Characterizing this capacitive coupling between the source and the drain

is important if precision is desired. Another non-ideality that arises due to indirect programming is the mismatch between the transistor that is 'programmed' versus the transistor that is actually used in the circuit. However, recently methods have been developed to characterize this mismatch [30].

Nevertheless, floating-gates enable building very compact circuits. This enables the building of larger systems like HMM classifiers using CMOS dendrites. The advantage being that not only could we individually program the FG-FETs for varying levels of charge to obtain taper easily but also could build a denser network. This would be useful for building larger systems. Also one must also take into account that neural systems are known to be inherently imprecise. Dendritic structures are not always similar and synapses are very unreliable. So one can say that this floating-gate mismatch is similar to dendrite-to-dendrite variability [19].

6.5.4 Simulating CMOS dendrites

The Simulink block as discussed in the previous chapter mainly serves two purposes. First, it converts the block-level Simulink model into a spice netlist which can be implemented on the FPAA. Secondly, it can also be used to run a behavioral simulation of the circuit.

Dendrite Simulink Block

The Simulink block simulates the behavioral characteristics of the dendrite structure given input/s. This provides the user an insight to the working of the dendritic circuit when implemented using the FPAA. The MOSFET parameters used are based on the MOSFETS present on the FPAA. It is characterized by coupled ordinary differential equations (ODE) and solved using the ode solver ode-45. The model has been tested for both static as well as time-varying inputs and has given reasonable results. For this chapter we have used EPSP signals as inputs for the block. Consider a dendritic line as given in Fig. 56c, with n number of nodes. The voltage at each node can be

calculated using the following coupled ODE [19],

$$\begin{aligned}\frac{d\vec{V}}{dt} = \frac{1}{C}(a_1 \cdot I_{inj} &+ k_1(e^{a_2\cdot\vec{V}/U_T} - e^{a_3\cdot\vec{V}/U_T}) \\ &+ k_1(e^{a_4\cdot\vec{V}/U_T} - e^{a_5\cdot\vec{V}/U_T}) \\ &+ k_2(e^{a_6\cdot\vec{V}/U_T} - e^{E_k/U_T}))\end{aligned} \tag{46}$$

For taper, we changed the parameters k_1 as it is proportional to axial conductances.

6.6 Analog Classifier for Word-spotting

We will now discuss the complete classifier structure. We have built a simple YES/NO HMM classifier using dendrite branches, a Winner-Take-All (WTA) circuit and supporting circuitry. We will simplify the modeling of a group of neuron somas and the inhibitory inter-neurons as a basic WTA block, with one winning element. We can consider the winning WTA output, when it transitions to a winning response as an equivalent of an output event (or action potential). To build this network, we made a model of a dendrite, initially a single piece of cable with branch points, where the conductance of the line gets larger towards the soma end, and the inputs are excitatory synaptic inputs. For classification, we focus on the ability for dendritic trees to be able to compute useful metrics of confidence of a particular symbol occurring at a particular time. This confidence metric will not only be a metric of the strength of the inputs, but also will capture the coincidence of the timing of the inputs. We would expect to get a higher metric if the $1st$, $2nd$, and $3rd$, inputs arrived in sequence, whereas we would expect a lower metric for the $3rd$, $2nd$, and $1st$ inputs arrived in sequence. This type of metric building is typical of HMM type networks. Simple example being if the word 'Y' 'E' 'S' were detected in a sequence as opposed to 'S' 'E' 'Y'. This is demonstrated by the simulation results as shown in Fig. 59, where when the input sequence is reversed the output metric is zero. The output metric is defined as the difference in output of last node when all inputs are present and when only the last input is present.

The network we built has two desired winning symbols, 'YES' and 'NO'. Each symbol is represented by one or more states that indicate if a valid metric has been classified. Only the winning states would be seen as useful outputs. The useful outputs feed back to the input dendrite lines, and in effect reset the metric computations. This is implemented using a Vector-Matrix-Multiplier block [162]. The system block diagram is as shown in Fig. 63. Each of the dendritic lines for the desired winning symbols has 5 states (dendritic compartments), where the inputs to the dendritic line represent typical excitatory synaptic inputs.

Synaptic inputs model symbol probability

In speech/pattern recognition, signal statistics/features are the inputs to the HMM state decoder. It generates the probability of the occurrence of any of the speech symbols. These signals when grouped, generate a larger set of symbols like phonemes or words [12]. We assume we have these input probabilities to begin with, as inputs to the classifier structure. We have taken inspiration from Lazzaro's Analog wordspotter for classification. However, we use a different normalization technique to eliminate the decay as shown in Fig. 55c. We can draw comparisons for such a system to a biological dendrite with synaptic inputs. We have modeled the input signals as excitatory synaptic currents. The synaptic current is given by :

$$I_{syn} \propto te^{-t/t_{peak}} \tag{47}$$

For a continuous cable,

$$\tau \frac{dV(x,t)}{dt} + V(x,t) = \lambda^2(x) \frac{d^2V(x,t)}{dx^2} + R(x)I_{input} \tag{48}$$

Considering that $exp(t/\tau)$ is the normalizing factor we have,

$$V(x,t) = V_1(x,t)e^{t/\tau} \tag{49}$$

where,

$$\tau \frac{dV_1}{dt} + V_1(x,t) = \lambda^2(x) \frac{d^2V_1}{dx^2} + R(x)I_{input}e^{-t/\tau} \tag{50}$$

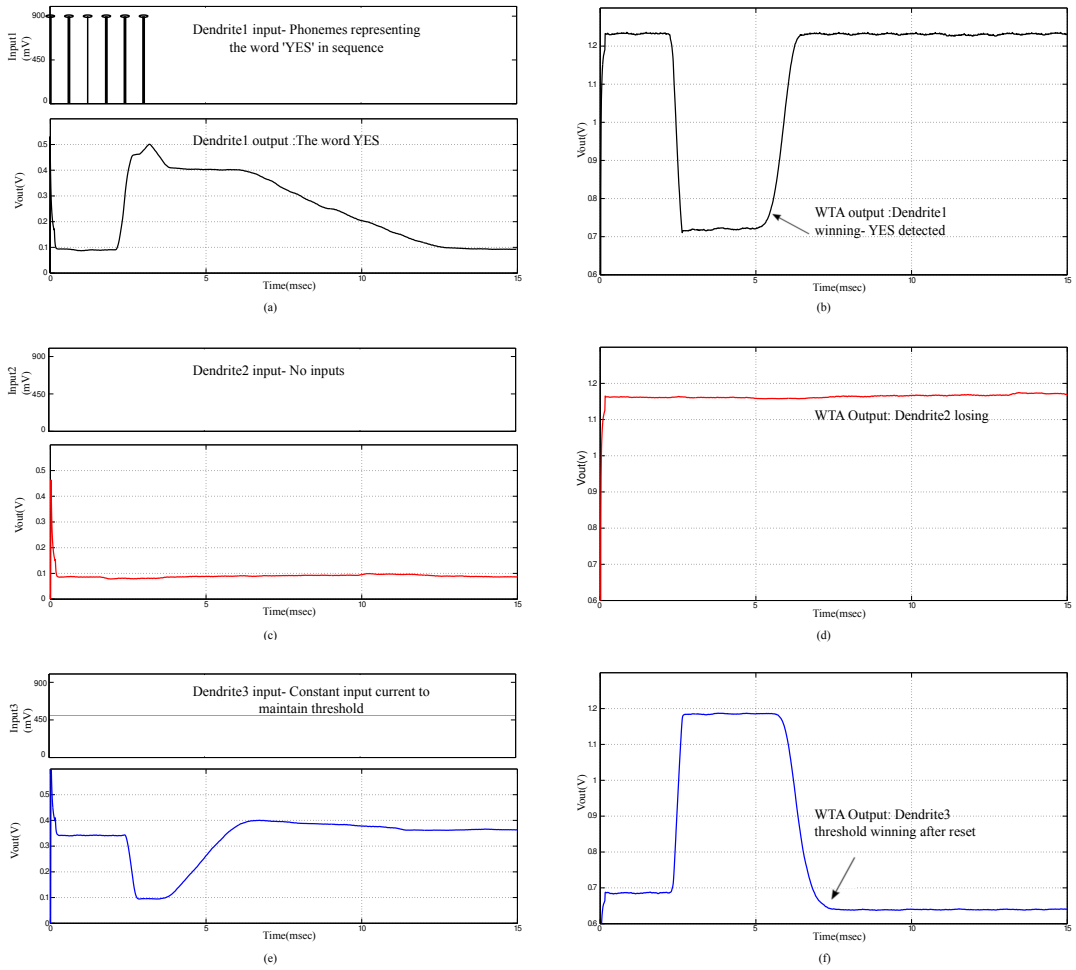


Figure 65: Experimental results for the YES/NO classifier system. The results shown are for the case when a YES is detected by the system (a) Synaptic inputs at the nodes of the first dendrite and the line output for the first dendrite. Here we assume we have the input probability estimate for the phonemes(symbols) for the word YES. (b) Corresponding WTA output for first dendrite. A low value signifies that it is winning. (c) The synaptic input and output for the second dendrite. (d) Corresponding WTA output for the second dendrite. (e) The line output for the third dendrite. (f) Corresponding WTA output of the third dendrite. The third dendrite acts as a threshold parameter. The amplitude of the word detected on a particular line needs to be higher than the threshold to win

$V_1(x,t)$ is the system output before normalization. From (47) and (50), we see that the input is similar to a synaptic current. Thus the inputs for the classifier using dendrites can be modeled as synaptic currents. This is represented in Fig. 63a and Fig. 63b. The derivation has two implications. First, we can use EPSP inputs to represent the input probabilities for phonemes. Second the system inherently

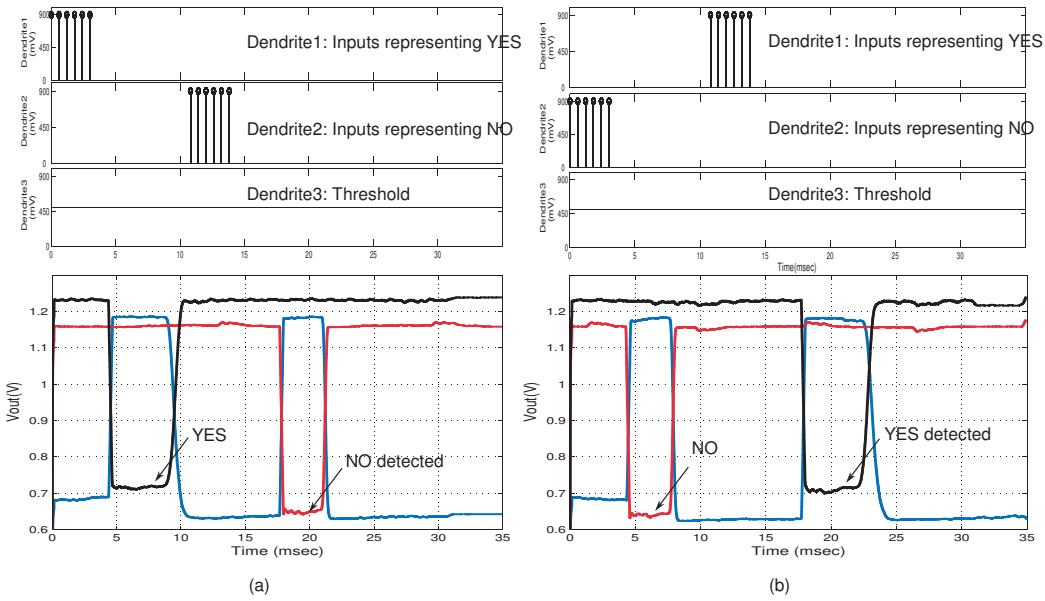


Figure 66: Experimental results for the classifier system when a sequence of words is detected. (a) First dendrite wins when the word YES is detected and the second dendrite wins when the word NO is detected. The WTA inputs and outputs are shown. (b) Second dendrite wins when the word NO is detected and first dendrite wins when YES in detected.

normalizes the outputs. In Fig. 65, the input to dendrite-1 signifies the phonemes of the word 'YES'. The inputs used were EPSP inputs that are similar to probability inputs $b_i(t)$ that in a typical HMM classification structure would be generated by a probability estimation block. There is no input into dendrite-2 which signifies that phonemes of 'NO' weren't detected. The threshold dendrite, dendrite-3 sets the threshold level. The WTA circuit determines the winner amongst the three dendritic lines. It is observed that when 'YES' is detected, dendrite-1 wins. This happens when coincidence detection is observed at the output of dendrite-1. The winning line signifies the word that is classified. It is only when all the inputs are in sequence and cross the given threshold that the dendrite line wins. In Fig. 65 we demonstrate the classification of the word 'YES'. The feedback from the WTA acts as a reset function for the dendrites, as after a word has been classified the threshold dendrite wins again. In Fig. 66, the classification of words 'YES' and 'NO' in a sequence is demonstrated. In Fig. 64 we show the effect of timing and variation of EPSP strengths for input

sequences.

The winning output of the WTA is akin to an action potential. In terms of classification too, the WTA output signifies if a 'word' has been detected. Our results have demonstrated that, such a system looks similar to an HMM state machine for a word/pattern. We can postulate from these experimental results that there are some similarities in computation done by HMM networks and a network of dendrites. The results are shown in Fig. 65 for a single word and for continuous detection of words in Fig. 66. We have demonstrated a biological model, built using circuits that is much closer than the implementation of any HMM network to date. Thus we have shown that an HMM classifier is possible using dendrites, and we have made a clearly neuromorphic connection to computation, a computation more rich than previously expected by dendritic structures.

6.7 Reconfigurable platform to build Neuromorphic circuits

I will give a brief overview of the experimental setup used for the study. We used the FPAA, RASP 2.8a for all experimental data and the software tool MATLAB Simulink and $sim2spice$ script to build the dendrite simulation block. All the data presented in this chapter comes from a reconfigurable hardware platform. It allowed us to build multiple complex circuits. The specific chip used from the family of RASP chips for this research work is RASP 2.8a [153]. It is a powerful and reconfigurable analog computing platform that can be used to build neuromorphic models. The CAB consists of groups of analog elements which include nFETs, pFETs, Operational Transconductance Amplifiers, capacitors, Gilbert multipliers, among others. These act as the computational elements which together can form complex sub-circuits that can be used to build analog computational systems. The interconnection of the CAB components is achieved by the switch matrix. It essentially consists of floating-gate(FG) pFETs. These $50,000$ programmable elements can be used not

Table 2: Comparing computational efficiency of Digital, Analog and Biological systems

Computing Type	Computational Efficiency
Digital (DSP)	$<$ 10MMAC/mW [21]
Analog SP (VMM)	10MMAC/μW [16, 28]
Analog(wordspotter)	$>$ 10MMAC/μW
Neural Process	$>$ 10MMAC/pW

only as programmable interconnects for routing but also as adaptive computational elements. The switch matrix allows for both local routing between CAB elements as well as global routing. Last but not the least, it has the programmer block, which selectively accesses a floating-gate device on the chip and through tunneling and injection tune it on, off or operational in between. This is not only an efficient routing scheme but can enable implementation of dense systems.

6.8 Classifier:Computational efficiency

A major advantage that analog systems have over digital systems is computational efficiency. The unit used to compare computational efficiency is Multiply ACcumulates (MAC) per second. The energy efficiency at a given node of the system, depends on the bias currents, supply voltage and also the node capacitance. We know that the node capacitance C is the product of conductance and the time constant τ. Now the bias current I_{bias} for a dendrite node is given by,

$$I_{bias} = (V_{rest} - E_k)\frac{C}{\tau} \tag{51}$$

where, V_{rest} is the resting potential, E_k signifies the voltage of a potassium channel and G is the axial conductance. Also, power is the product of voltage across the node and current into the node. Now for a single node of an HMM classifier, we have 2 MAC/sample. Assuming $\tau \sim delay$, which at a given node is approximately $1ms$. Thus,

$$Energy/MAC = \frac{1}{2}V_{dd}(V_{rest} - E_k)C \tag{52}$$

We have compared the computational efficiency of digital, analog and biological systems as shown in Table 2. Now for a wordspotting passive dendritic structure, we have 2 MAC/node. Typical dendrite would have over 1000 state variable equivalents in its continuous structure. For a particular neuron time constant τ, we would want to have multiple samples for proper operation. For this discussion, let's assume an effective discrete time sample rate 5 times more than τ. Let us choose $\tau = 1ms$ for this discussion. Thus, we have each tree computing 10 MMAC for an HMM computation. For biological systems, say the brain has $1T$ neurons and total power consumption of about 20 W. Thus the power consumption is 20 pW/neuron. In a passive dendritic structure, the computational efficiency is 10 MMAC /neuron. Thus the computational efficiency of biological systems works out to be 0.5 MMAC/pW. Also from the equation it is evident that a major factor contributing to energy efficiency is node capacitance. Currently the node capacitance on the chip we used was $1pF$. If we further scale down the process used, this number will also reduce. This effectively means higher computational efficiency. A decrease to $10fF$ itself will give us an improvement of 2 orders of magnitude.

6.9 Conclusion

We have demonstrated a low power dendritic computational classifier model to implement the state decoding block of a YES/NO wordspotter. We have also found that this implementation is computationally efficient. We have demonstrated a single dendritic line with 6 compartments, with each compartment having a single synaptic input current. We have seen the behavior of a single dendrite line by varying three parameters, namely, the 'taper', the delay between inputs and the strength of the EPSP input currents. The effects of taper which enabled coincidence detection were studied. We have also seen the functioning of the WTA block with dendritic inputs and the how feedback helps initiate the reset after a word/phoneme is detected. We

also build a Simulink dendritic model and simulated the output for time-varying inputs to compare with experimental data. This demonstrated how such a network would behave if inputs were in a sequence or if they were reversed.

The broader impact of such a system is two-fold. First, this system is an example of a computational model using bio-inspired circuits. Secondly the system proposes a computationally efficient solution for speech-recognition systems using analog VLSI systems. As we scale down the process, we can get more efficient and denser systems. We can also address how synaptic learning can be implemented and classification systems be trained. We can also model the input synapses as NMDA synapses to get a more multiplicative effect. In NMDA synapses, the synaptic strength is proportional to the membrane voltage. It couples the membrane potential to the cellular output. This could lead to a more robust system and is also closer to how biological systems are modeled. Also, we have modeled passive dendrites in this chapter. It would be interesting to see how the system behaves when we add active channels. We currently have systems built that will enable us to further explore this discussion which is beyond the scope of this chapter. There is a lot of scope for discussing how to build larger systems using this architecture. We can use spiking WTA networks for a larger dictionary of words. It is evident from the computational efficiency discussions, that clearly analog systems are a better choice for higher computational efficiency and lower costs. This calls for greater effort to build such systems. Reconfigurable/programmable analog systems open a wide range of possibilities in demonstrating biological processing and also for signal processing problems. There is great potential in other areas as image processing and communication networks as well. These systems will not only enhance our understanding of biological processes but also will help us design more computationally efficient systems.

CHAPTER VII

BUILDING RECONFIGURABLE NEUROMORPHIC SYSTEMS

Modern day technology relies heavily on silicon devices to do computation. However, digital processors are now approaching an efficiency wall as illustrated in Fig. 67. Hence, there is a critical need to invest in alternate approaches such as neuromorphic systems. The brain is a really efficient system when it comes to classifying images and sounds. In the brain, computation and memory are not separate as is typical in most Very Large Scale Integration (VLSI) chips, that have a Von-Neumann architecture. Neurons have synapses that change their weight based on learning. We propose using floating-gate(FG) transistors in a similar fashion, where the gate terminal charge is modulated depending on the inputs. Thus, we achieve computing in memory similar to biology. Together with existing mixed-signal systems and neuromorphic models, we can build ultra efficient low-power systems. Building a neuromorphic supercomputer aka the Silicon Brain is within our grasp, and is a grand challenge of our times.

To be able to mimic biology and also solve more complex problems there is a need to build system solutions. To this end, we design neuromorphic chips using biologically inspired circuits. In our research lab two approaches were followed. One using the all-to-all connectivity point-neuron Neuron1 chip and the FPGA style architecture of the Neuron2 chip which also models closely the dendrites in the fabric. I will discuss the Neuron2 chip in this chapter as it is relevant to this discussion.

Neuromorphic hardware models the behavior of biological neural systems to enable efficient computational modeling. It leads to significant reduction in size and power, compared to the traditional approaches of modeling based on numerical integration

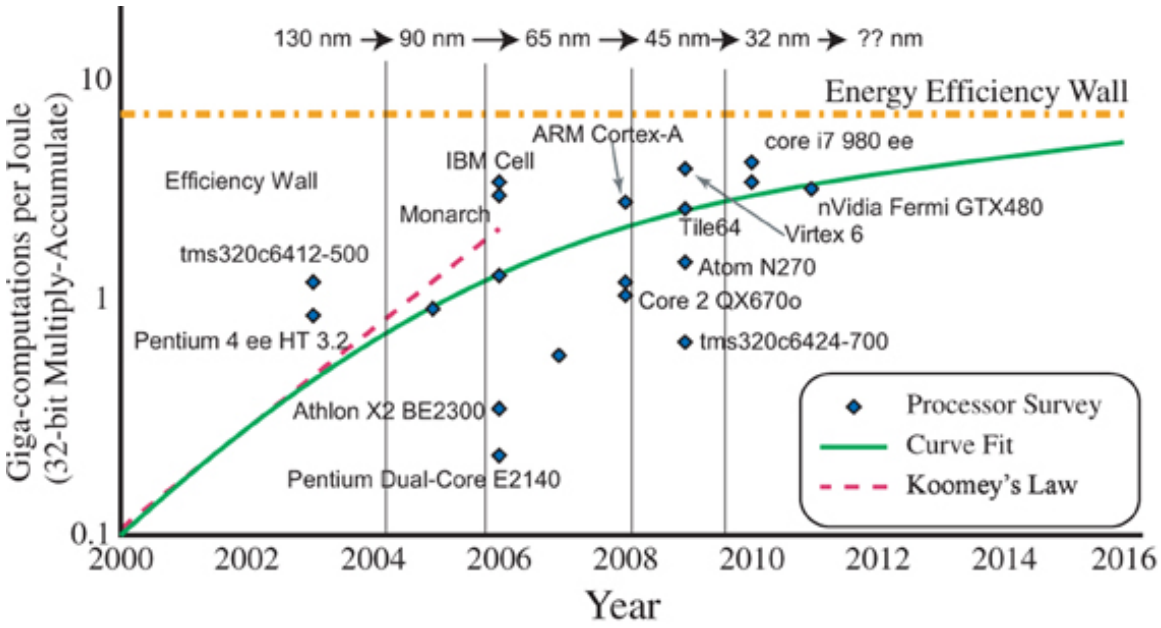


Figure 67: A study of different modern digital processors shows saturation in terms of computational efficiency [16]. This can be addressed by moving towards analog/neuromorphic solutions.

on a digital computer. Dendritic computation is often ignored and a point-neuron model is typically adopted in such approaches. However, studies show dendrites perform operations such as non-linear filtering, spatial and temporal summation of synaptic inputs, coincidence detection, synaptic scaling, and sequence detection [14]. We present an efficient and scalable hardware system for studying dendritic computation in large-scale networks with programmable learning synapses and dendrites that support arbitrary branched dendrites. We also include active channels in the dendrite for non-linear filtering. Effectively, we implement a multi-layer neural network within each neuron, which results in very powerful computation as shown in Fig. 68.

7.1 Neuron2 chip

The neuron chip was built keeping in mind architectural changes that were necessary to build large-scale neuromorphic systems and was a first step in that directions. This chip consisted of an array of neurons with plastic synapses and programmable

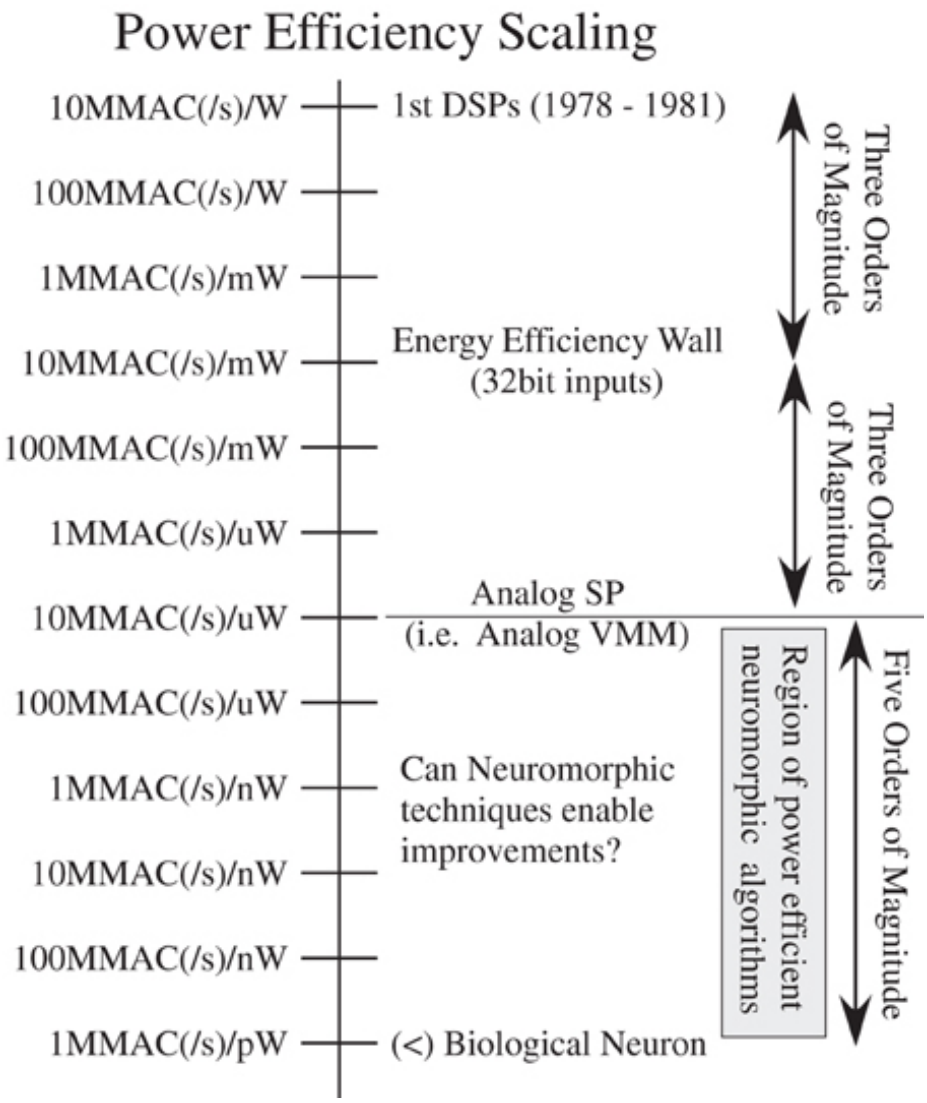


Figure 68: Comparison of current silicon systems is illustrated here. One can see that neuromorphic approaches are far more efficient than current digital/analog solutions. Efficiency is stated in terms of Million Multiply Accumulates/Watt

dendrites. This architecture used FPGA-style routing for intra-chip routing and Address Even Representation (AER) developed by Mahowald and Silvotti for inter-chip routing. It was designed to contend with traditional digital solutions as a reconfigurable and programmable alternative. The main drawback of FPGA's is high power consumption, size and interconnect parasitics. If we compare reconfigurable digital solutions with biology, we see that in biology neurons mainly rely on nearest-neighbor connections. Thus an all to all connectivity matrix for all logic elements wouldn't be

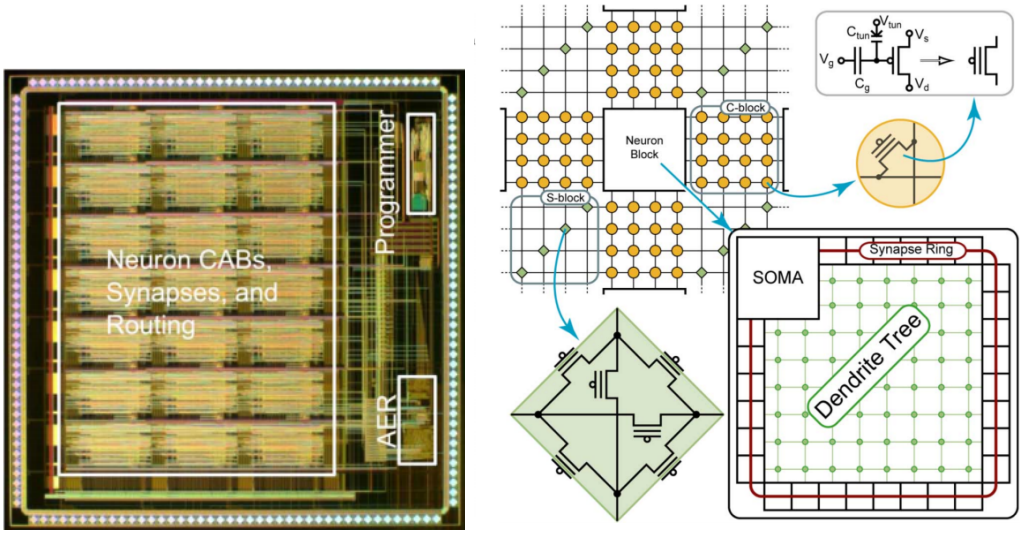


Figure 69: (a) Neuron2 chip die photo (b) Neuron CAB architecture: The Neuron chip consists of neuron cells embedded in a typical FPGA routing fabric. The Neuron I/O interface with the routing at the C-Blocks through programmable FG switches. The tracks are routed at the S-Blocks, where each node consists of 6 switches. The neuron cell has synaptic inputs, programmable dendrites with active channels that aggregate inputs into the soma block.

an ideal solution. The neurons have rich dynamics and several state variables. They communicate using action potentials and have a low event rate. It was endeavored to replicate this in the chip.

The chip consists of 21 programmable neurons which consists of synapses, 2D dendritic structure with active channel and a soft winner-take-all (WTA) network. The neuron inputs and output can be routed on the programmable routing fabric. Some of these routing lines are also connected to the AER for off-chip communication. In Fig. 69 you can see an illustration of how a single Computational Analog Block(CAB) is built.

7.2 Dendritic Modeling and Computation

As discussed before we have modeled dendrites based on the linear cable theory as postulated by Wilfred Rall. The 2D dendrite architecture supports arbitrary arborization which enable the study of some interesting structures. Properties like directional selectivity, sub-linear temporal summation and non-linearity were studied. Also these properties were employed to study a classifier structure using dendrites to re-create results we had demonstrated previously.

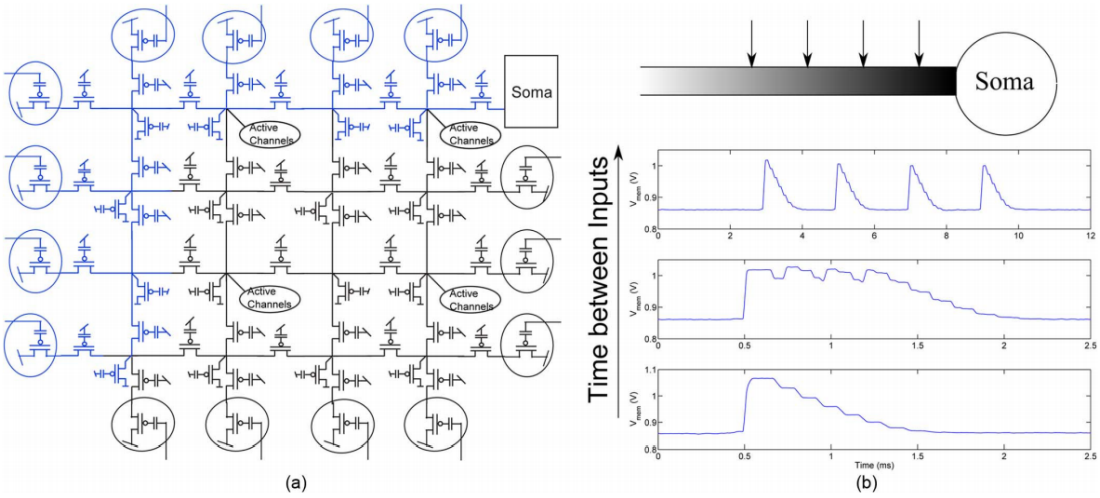


Figure 70: (a) $2D$ Dendritic Structure as implemented on the chip (b) Coincidence detection results

7.3 *RASP* 3.0N

The RASP 3.0N is the first neuromorphic SoC built with an FPGA architecture. It has specialized neuron blocks as well as digital and analog blocks as in RASP 3.0N. The analog CABs have specialized macroblocks. Macroblocks are circuits that are very useful when computing like the $C4$ filterbank, envelop detectors, WTA circuits etc. It also has an AER block for easy communication between neuron blocks along with $16K$ memory dedicated to it. The neuron blocks consists the neuron model discussed before along with a $2D$ dendritic network with different kinds of synapse

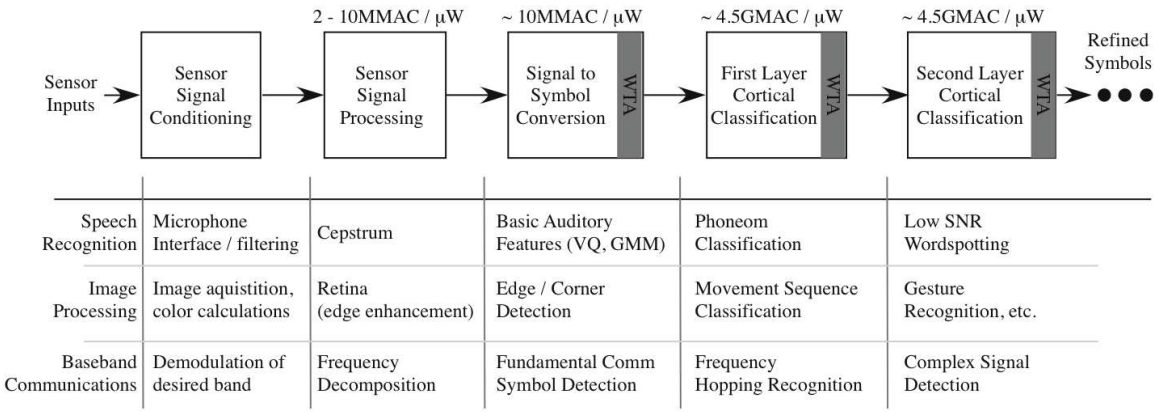


Figure 71: Different applications using the Pattern Recognition system based on biology. It has application in speech and image processing and in communication systems. The state decoder in this paper is one block that is part of the whole system level design that we plan to build.

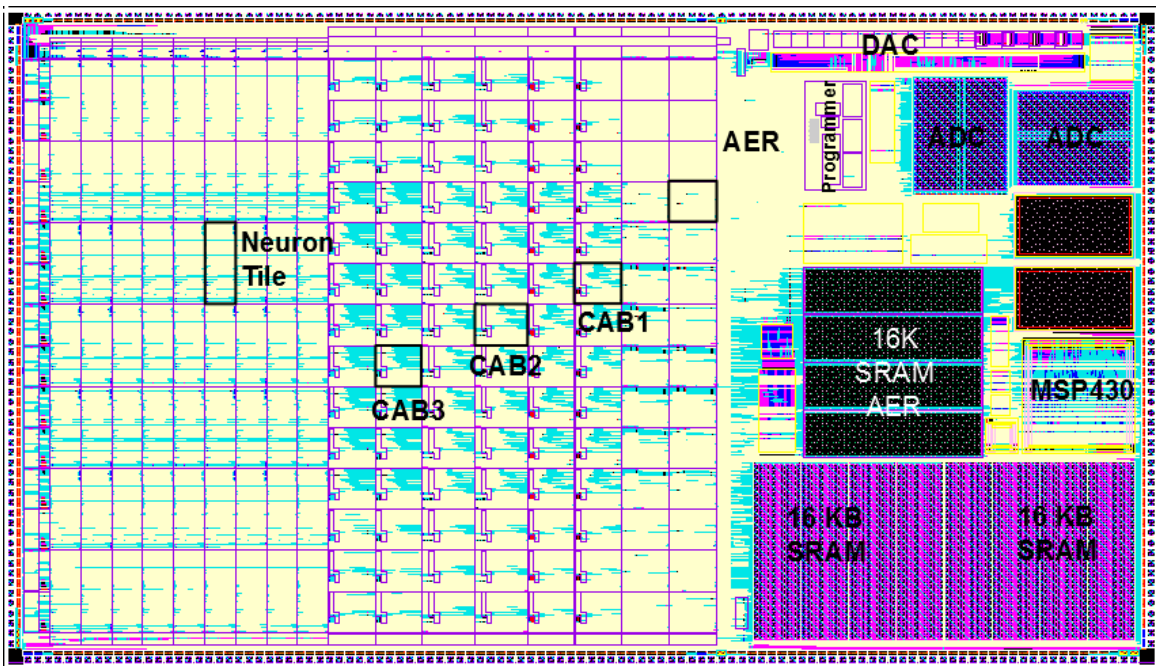


Figure 72: RASP 3.0*N* chip layout

structures as shown in Fig. 75. The RASP 3.0N core has 28 Digital tiles, 84 Analog tiles (each having specialized blocks), and 63 Neuron tiles. The basic tile is depicted in Fig. 75. Each tile consists of the global interconnect and the CABs. The global interconnect consists of the C-Block that makes connections from CABs to the interconnect and the S-Block (switch block) that is used for routing. Within the

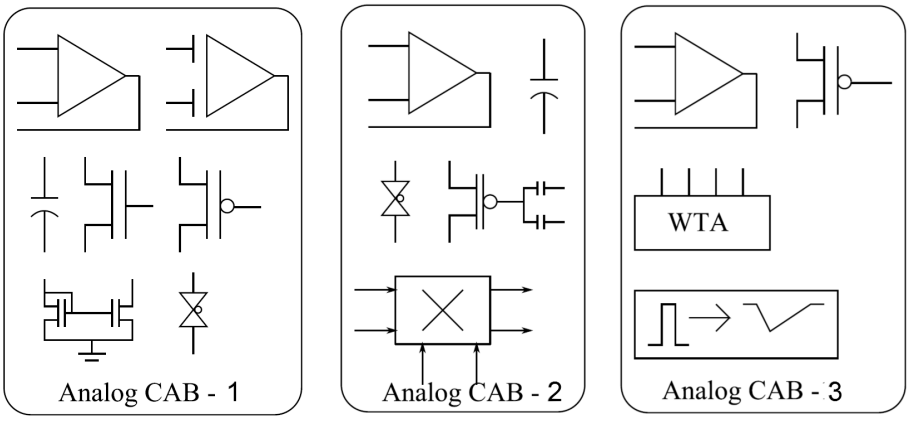


Figure 73: Different CAB types in RASP 3.0*N*. CAB1 and CAB2 are similar to RASP 3.0. CAB3 is a specialized CBA with WTAs, waveform shaping block, FETs and OTAs.

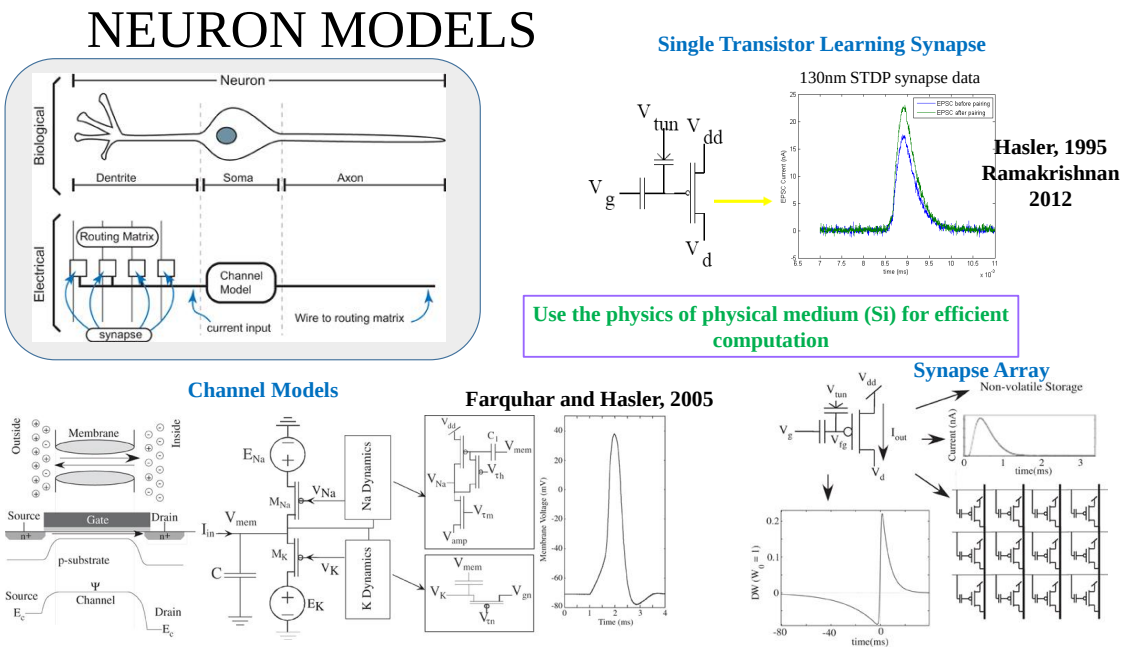


Figure 74: RASP 3.0*N* Neuron CAB :This diagram illustrates the how the different parts of the neuron are biophysically modeled. We use the channel models developed by Farquhar et al. [167] as well as a single floating gate as a learning synapse.

CAB, there exists the local interconnect, which allows all-to-all connectivity between the components. The array was designed such that the analog and digital CABs have 24 I/O. This choice reflects a trade-off between the number of I/O and the size of the local interconnect within each block. An increased local interconnect

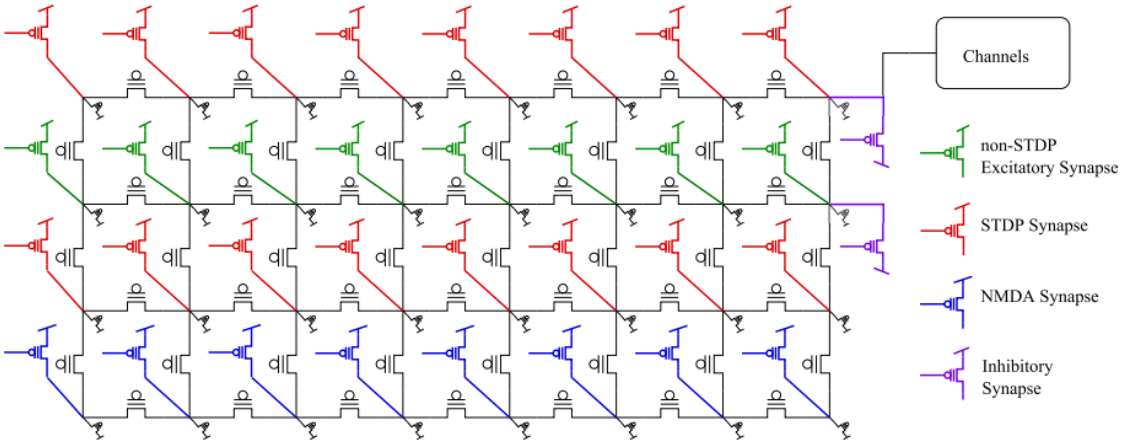


Figure 75: RASP 3.0N Neuron CAB :This diagram illustrates the dendrite connectivity with different types of synapses feeding into the channels and then to the soma. This was a change as compared to the previous Neuron2 chip

Table 3: Comparison of RASP chips

RASP	3.0	3.0N
Parameter	Value	Value
Digital CLBs	98	28
Analog CAB1	84	28
Analog CAB2	14	28
Analog CAB3	–	28
Neuron CABs	–	63

also increases routing parasitics. Each Digital CAB consists of 8 BLEs and local interconnect. The BLE itself comprises of a four-input Look-up Table (LUT) and a D-F/F whose inputs and clocks are routable. The analog and digital tiles have general purpose I/O blocks terminating the tiles, which allow analog or digital signals in/out of the tiles. The I/O blocks terminating the neuron tiles are basic I/O which connect the global interconnect to the AER in/out blocks. At the bottom of the array is the C^4 BPF I/O block, which consists of programmable filterbanks. The C^4 I/O block is reconfigurable, since its inputs may be from an external microphone or from the array itself. The outputs from the filterbank can be routed into the array for further processing, or be routed out to pads. The neuron CAB itself consists of

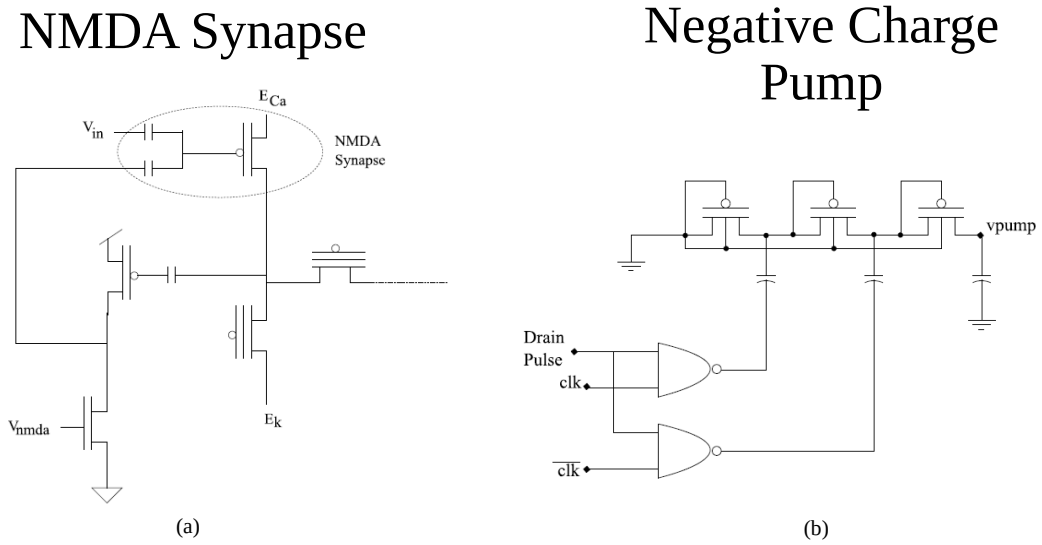


Figure 76: New components added on the RASP 3.0*N* chip. (a) NMDA Synapse model as implemented on the chip. This synapse is modeled using a common-source amplifier whose gain is set by the ratio of the input capacitance of the floating gate to the overlap capacitance is used to amplify the local dendritic potential before feeding it back to the second control gate on the synapse. The inverting characteristic of the common-source amplifier is desirable, since the synapse is implemented using a pFET device [166].(b) **Negative Charge Pump**: Integrated negative charge pump on chip to eliminate use of noisy high-voltage supplies and allow run-time injection for synaptic learning in the STDP synapses [166].

biophysically based models as shown in Fig. 74 for the channels, synapse and the dendritic network which equivalent to the local interconnect with different synaptic inputs is shown in Fig. 75. This architecture differs from Neuron2 where synaptic inputs were only on the periphery of the dendrites. This is very useful especially when one wants to build the dendritic classifier. Also, the synapses are of four types: 16 Spike-Timing Dependant Plasticity (STDP),8 Non-STDP excitatory, 8 N-Methyl D-Aspartate (NMDA) and 2 inhibitory synapse. Typical synapse structure is shown in Fig. 74. For STDP synapses learning is done by using injection and tunneling mechanisms to change the 'weight' of the synapse which is represented by a floating-gate. The NMDA synapse as shown in Fig. 76 was a new feature on this chip. NMDA

receptors in synapses are thought to play an important role in synaptic plasticity. A sufficient pre-synaptic excitation causes NMDA receptors to be activated and increase synaptic efficacy. Studies on dendritic trees have also revealed that NMDA synapses are a key component for obtaining robust directional selectivity which further help give us the super-linear effect we desire while building the dendritic classifier. Also, in order to avoid setting different injection voltages as needed for the learning synapses, we included a negative charge pump as shown in Fig. 76 which enables all elements in the Neuron CAB to also operate at 2.5V supply. The negative charge pump provides sufficient field across the drain and source terminals to cause injection to support learning in the synapses.

The choice of FPGA-style manhattan geometry for global routing allows us to extend the $x2c$ design suite to include the RASP $3.0N$ chip. A major drawback while testing the Neuron2 was lack of tools. This chip is still to be tested but with the new tool framework, we know this will be an easier task.

CHAPTER VIII

CONCLUSION

The objective of my research has been to build reconfigurable mixed-signal and neuromorphic systems to implement low-power solutions. I demonstrated the first computationally efficient classifier structure using bio-inspired CMOS dendrites, which can be used for speech/pattern classification. I was instrumental in developing the first mixed signal FPAA SoCs and the software CAD tools for this system. These reconfigurable chips can be used for a variety of applications like speech and image processing. I presented an Analog-Digital Hardware-Software CoDesign environment $x2c$ for simulating and programming reconfigurable systems. I also talked about the CAD synthesis tool $vpr2swcs$ used for targeting FG based mixed signal SoCs of the RASP 3.0 family.

8.1 Summary of Research so far

In chapter 1, I discussed why it is important to look at neuromorphic solutions as one of the solutions to study the brain as well as get inspired by it to build new technology. The objective is to build systems that leverage digital, analog and bio-inspired circuits. A Neuromorphic engineer's thesis is that silicon emulates biology [19]. These low-power reconfigurable systems can be used for image processing, speech processing applications, prototyping, education and also studying networks of neurons. This work endeavors to be a step in the direction of building such large-scale neuromorphic systems.

In chapter 2, I presented an IC that integrates divergent concepts from previous multiple large-scale FPAA designs along with low-power digital computation and interface circuitry (i.e. DACs, ADCs). This unified structure enables a wide range

of a system-on-a-chip computing options that can be optimized for a wide range of parameters (i.e. Power); the resulting IC architecture is the most sophisticated FPAA device built to date.

In chapter 3, we discussed the synthesis tool $vpr2swcs$. The tool was used to build parametric FPAA architectures that consists of both digital and analog blocks. The modifications, challenges and novel solutions implemented while doing mixed signal system design were discussed.

In chapter 4, I presented an Analog-Digital Hardware-Software CoDesign $x2c$ environment for simulating and programming reconfigurable systems as well as $sci2blif$. The tool flow was demonstrated with multiple mixed signal examples through this configurable system.

In chapter 5, we forayed into the world of neuromoprhic systems especially dendrites. We were able to accurately program different circuit architectures to emulate dendrites. It was demonstrated that these circuits accurately reproduce results predicted from cable theory when inputs to the system are small.

In chapter 6, we talked about how a network of dendrites can be used to build the state decoding block of a wordspotter similar to a Hidden Markov Model (HMM) classifier structure. We talked about how these structures can be used for speech and pattern recognition. The advantage of such a structure over digital systems is ultra low power consumption.

In chapter 7, we discussed why neuromorphic systems are more efficient and can be used to solve more complex problems. We discussed neuromorphic chips that were developed using biologically inspired circuits. The neuron chip shown was embedded in FPGA style routing architecture with a CAB that modeled dendrites, neurons, synapse, active channels using transistors, .

8.2 Summary of Work Completed

Significant work has been accomplished towards the goal of design reconfigurable high performance mixed-signal FPAAs with the guidance of Prof.Jennifer Hasler and the help of a lot of members of the Integrated Computation Electronics (ICE) Lab. The details of the work I have done are listed below.

- Key part of the RASP 3.0 design and layout team with Richard Wunderlich, Farhan Adil and Stephen Nease.
- Led the testing effort for 3.0 chip along with Sihwan Kim, Michelle Collins, Sahil Shah and Farhan Adil. This system was then presented and speech classifier demonstrated at ISSC 2015.
- Designer for FPAA synthesis tool *vpr*2*swcs* with Richard Wunderlich. Added specialized macroblocks to the architecture to enable use of dedicated blocks. Helped first few circuit examples using the tools.
- Higher level tools infrastructure called sci2blif developed along with Michelle Collins which helps translate high level blocks in xcos to BLIF format.
- Detailed digital simulations of MSP430 processor with peripherals and memory. Completed Back-annotated delay simulations to verify operation of the processor with memory interfaces. Designed and implemented a memory controller for the same.
- Preliminary testing on the RASP3.0 along with Sihwan Kim for programming algorithm for FGs.
- Dendrite Modeling: Worked actively with Stephen Nease to build bio-physically modeled dendrites. Developed a mathematical model and implemented it in MATLAB SIMULINK to be able to simulate n-tap dendrites. The simulation

model was found to be closely matching the results from the hardware. It was also pivotal for future design choices.

- Demonstrated first Wordspotting application using dendrites. This demo was also selected for presentation at BIOCAS 2011. Analyzed dendrite line results and implemented THE classifier structure on the RASP 2.8a. Mathematical proofs were done along with Scott Koziol.

- Designed the AER module layout for Neuron2 chip. Lead designers on this chip were Shubha Ramakrishnan and Richard Wunderlich. Helped with implementing classifier using dendrites on the chip.

- Co-architect on the Neuron 3.0 design and layout along with Shubha Ramakrishnan. Richard Wunderlich, Stephen Nease, Farhan Adil also contributed to this chip.

- Led Design layout for 40nm SRAM 16K bank along with Richard Wunderlich and Sihwan Kim. Completed digital simulation for 40nm processor and memory peripherals.

- Design layout for standard cells, peripherals and small blocks in 40nm along with Richard Wunderlich, Michelle Collins and Sihwan Kim.

- Helped in the design process for the new FPAA class board. Lead designer was Michelle Collins.

- Co-taught ECE 6435 graduate course 'Neuromorphic Analog VLSI circuits' in Spring 2014 along with Prof. Hasler.

8.3 Vision going forward

I'm very excited to pursue the research path I have started on and know there is a lot of potential to take these systems further. Below I detail my vision for the broad areas

of neuromorphic systems and applications as well as building reconfigurable hardware.

8.3.1 Neuromorphic Systems

I plan to build a hybrid neuromorphic machine. This system would include analog, digital and neuromorphic elements; an amalgamation of all of the above results in a very powerful processing machine. Neuromorphic systems though considered non-traditional have a lot of potential to look into real world problems as well as model biology. My goal will be to build an efficient neuromorphic processor. I will focus my research on building systems that leverage both digital, analog and bio-inspired circuits. The goal being to build a powerful prototype for a neuromorphic processor.

Dendritic Computation: I will continue to research dendrites and build classifier using them for solving spatio-temporal problems. I hope to further investigate various configurations of the dendrite such as, passive dendrite cables, with NMDA synapses and active channels. I hope to investigate the classifier with the following configurations of the dendrite: passive dendrite cable, with NMDA synapses, with active channels, and with NMDA synapses and active channels. We believe that using this dendrite-based neuromorphic classifier and other front-end techniques, we can build an effective audio recognition system which can be used for a wide variety of applications in speech/audio processing.

RASP $3.0N$ **system:** The RASP $3.0N$ has some specialized types of synapse like the STDP, inhibitory, NMDA and non-STDP synapses. I plan to test these with the dendritic line. We believe that NMDA synapses have a multiplicative effect on the dendritic line thus making the HMM like dendrite line more robust. It will be interesting to study the effects of active channels on the dendrite line as well. The RASP $3.0N$ architecture allows us to easily do that.

Neuron Network implementation: There are some interesting networks like

Liquid State Machine with dendritically enhanced output(LSM-DER) [25] and the Synaptic Kernel Inverse(SKIM) model for pattern recognition [31] that I would like to test on the RASP 3.0N. The fundamental blocks of these systems are synapses, dendrites and WTA network.

In-silico Brain: The ultimate goal is to build a system that matches or exceeds the complexity of the human brain. It is advantageous to build smaller applications emulating functionality of basic elements using silicon models of the neurons, synapses, and dendrites to build networks. Learning can be implemented on these networks using floating gate (FG) transistor technology, which can be used as a memory element that simulates learning. Building a neuromorphic supercomputer is indeed a grand challenge of the twenty-first century.

8.3.2 Applications of Neuromorphic systems

One of the goals of my research is to not only to mimic biology in silicon, but also utilize these bio-inspired systems to solve real world problems like speech classification, pattern recognition, and robotics.

Speech Recognition: I have demonstrated a YES/NO classifier using dendrites and a Winner Take All (WTA) circuit using our VLSI chip. I believe that by using this dendrite based neuromorphic classifier and other front end techniques, I can build an effective audio recognition system which can be used for a wide variety of applications in speech/audio processing particularly for phoneme recognition.

Wearable Low Power Technology: Collecting biometrics from regular items like cellphones, clothing, glasses etc to compute useful information for the user in a very power efficient manner. I will focus on building low power systems that utilize subthreshold dynamics of CMOS transistors to do very efficient computation. The goal is to use these systems to compute basic day to day metrics that we need as we go about our tasks. These will be wearable devices that monitor sound, touch or even

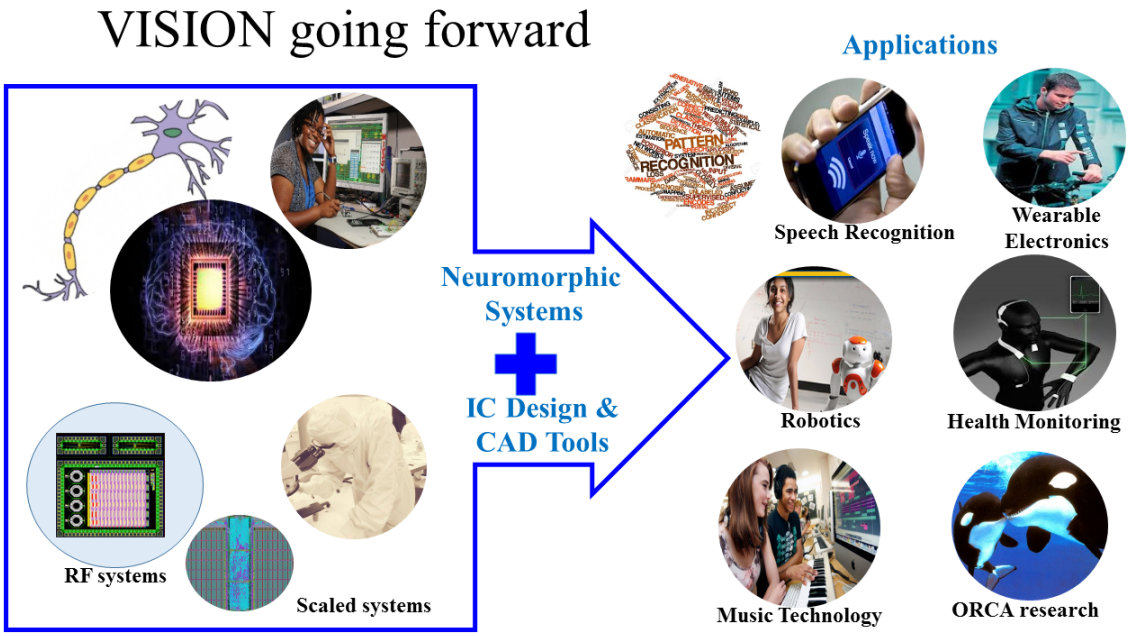


Figure 77: In the future I see myself continue to build and scale reconfigurable and neuromorphic systems to apply to a wide variety of applications like pattern recognition, health monitoring systems to name a few. Interesting tangents could be in music technology and orca research (build useful electronics for one of my favorite species).

body heat. For this we can use existing technology and use hardware to compute something meaningful with it.

For testing these systems I plan to use reconfigurable FPAA SoCs that I have helped build to test prototypes. One can test many interesting ideas on these devices. I also see this as an excellent avenue to collaborate with people already working on building sensors or front end signal detection, which my systems can then classify.

Building bio-applications: We have done some initial testing to help implement post-processing circuits for the wearable ballistocardiography system on the FPAA [9]. I plan to continue to develop such a system.

8.3.3 Reconfigurable Mixed Signal Architectures

The backbone of my research that enables building such systems has been reconfigurable hardware and a software CAD toolset. Using FG as the switch element, adds

the properties of non-volatility and compactness.

FPAA sensor ICs: I have been instrumental in developing and testing a new generation of Reconfigurable Analog Signal Processor (RASP) 3.0 family of SoCs designed by our research group, fabricated in the 350 nm technology. This design effort addressed a lot of the interfacing questions and made our systems more compact. The neural IC is a variation of this as it contains not just analog and digital blocks, but neuron blocks as well. These are very powerful SoCs that I plan to use as well as build ICs that can be used to interface with these SoCs. I also hope for students to learn IC design by building chips through MOSIS. I hope to build RASP peripherals for these chips that can specialize as sensor blocks that interface with the existing IC. // **Hardware Software Codesign/CAD tool:** I was also instrumental in developing a new CoDesign environment $x2c$ for simulating and programming reconfigurable FG based mixed signal SoCs. These SoCs consist of an integrated processor, I/O peripherals, and a Field Programmable Analog Array (FPAA) comprised of analog and digital components. This novel open source tool platform empowers the user to seamlessly CoDesign low power analog and digital systems in a single environment. This approach integrates multiple open source tools to develop a coherent user friendly design flow. Scilab is the graphical front end for system level block design, which invokes Verilog to Routing (VTR)/ Versatile Place and Route (VPR) tools. Custom software generates and integrates these tools to program and test the IC. We demonstrated several mixed signal examples, as well as how to perform useful computation using the routing fabric. This is a very powerful open source platform that is now open to a wider audience and will be very effective for teaching in the classroom. The tool can be extended to any new family of ICs that I build in my future research group. // **Scaling FG devices:** We had the opportunity to build an FPAA on TSMC 40nm process with a heterogeneous fabric with an RF frontend, RF CABs and Baseband CABs/CLBs. I helped build the digital infrastructure of the chip such as the SRAM,

processor and peripheral blocks. We got some very promising initial results and the floating gates have been shown to retain charge on this process node. However, the processor testing revealed some errors due to which we could not test the chip and it requires fixes. However, it was a good proof of concept for floating-gates devices on scaling the process node and I hope to utilize this experience in the future.

In conclusion, though the first question that came to my mind as a young graduate student '*Can My Chip Behave Like My Brain?*' seemed very naive, we have made some inroads towards figuring out the problem. Yes, we still have a lot more to discover and innovate but I'm excited about the path ahead full of myriad possibilities. Or on a lighter note as my good friend Alex remarked, "*I'm not sure if my chip can behave like my brain, but I know my brain can behave like a chip!*".

REFERENCES

[1] Allman, W. F., *Apprentices of Wonder: Inside the Neural Network Revolution*. Bantam Books, 1989.

[2] Basu, A., Brink, S., Schlottmann, C., Ramakrishnan, S., Petre, C., Koziol, S., Baskaya, F., Twigg, C., and Hasler, P., "A Floating-Gate-Based Field Programmable Analog Array," *IEEE Journal of Solid-State Circuits*, vol. 45, pp. 1781–1794, 2010.

[3] Farquhar, E., Abramson, D., and P.Hasler, "A Reconfigurable Bidirectional Active 2 Dimensional Dendrite Model," *IEEE International Symposium on Circuits and Systems*, vol. 1, pp. 313–316, 2004.

[4] George, S. and Hasler, P., "Hmm classifer using biophysically based cmos dendrites for wordspotting," *BIOCAS*, 2011.

[5] Hasler, J. and Marr, B., "Finding a roadmap to achieve large neuromorphic hardware systems," *Frontiers in neuroscience*, vol. 7, 2013.

[6] Hasler, P., Smith, P., Anderson, D., and Farquhar, E., "A Neuromorphic IC Connection Between Cortical Dendritic Processing and HMM Classification," *IEEE 11th Digital Signal Processing and 2nd Signal Processing Education Workshop*, pp. 334–337, 2004.

[7] Hasler, P., *Foundations of Learning in analog VLSI*. PhD thesis, California Institute of Technology, 1997.

[8] Hasler, P., Koziol, S., Farquhar, E., and Basu, A., "Transistor Channel Dendrites implementing HMM classifiers," *Circuits and Systems, 2007. ISCAS 2007. IEEE International Symposium on*, vol. 1, pp. 3359 – 3362, 2007.

[9] Inan, O. T., "Recent advances in cardiovascular monitoring using ballistocardiography," in *Engineering in Medicine and Biology Society (EMBC), 2012 Annual International Conference of the IEEE*, pp. 5038–5041, IEEE, 2012.

[10] Juang, B. H. and Rabiner, L. R., "Hidden markov models for speech recognition," *Technometrics*, vol. 33, pp. 251–272, 1991.

[11] Koch, C., *Biophysics of Computation*. New York, NY: Oxford University Press, 1999.

[12] Lazzaro, J., Wawrzynek, J., and Lippmann, R., "A micropower Analog VLSI HMM State Decoder for Wordspotting," *Advances in Neural Information Processing Systems 9*, vol. M. C. Mozer, M. I. Jordan, and T. Petsche, Eds. Cambridge, Massachusetts: MIT Press, pp. 727–733, 1996.

[13] Lippmann, R. P., C. E. I. and Jankowski, C. R., "Wordspotter training using figure-of-merit back-propagation," *Proceedings International Conference on Acoustics, Speech, and Signal Processing*, vol. 1, pp. 389–392, 1994.

[14] London, M. and Hausser, M., "Dendritic computation," *Annual Review of Neuroscience*, vol. 28, pp. 503–532, July 2005.

[15] Luu, J., Goeders, J., Wainberg, M., Somerville, A., Yu, T., Nasartschuk, K., Nasr, M., Wang, S., Liu, T., Ahmed, N., Kent, K. B., Anderson, J., Rose, J., and Betz, V., "VTR 7.0: Next Generation Architecture and CAD System for FPGAs," vol. 7, pp. 6:1–6:30, June 2014.

[16] Marr, H. B., Degnan, B., Hasler, P., and Anderson, D., "Minimization of energy per op in an asynchronous pipeline above and below threshold," in *IEEE Trans. on VLSI*, Accepted 2011.

[17] Mead, C., *Analog VLSI and Neural Systems*. Reading, MA: Addison-Wesley, 1989.

[18] Mel, B. W., "What the synapse tells the neuron," *Science*, vol. 295, p. 1845, 2002.

[19] Nease, S., George, S., Halser, P., and Koziol, S., "Modeling and implementation of Voltage-Mode CMOS dendrites on a reconfigurable analog platform," *IEEE Transactions on Biomedical Circuits and Systems*, accepted for publication.

[20] Polsky, A., Mel, B. W., and Schiller, J., "Computational subunits in thin dendrites of pyramidal cells," *Nature Neuroscience*, vol. 7, pp. 621–627, 2004.

[21] R. Chawla, A. Bandyopadhyay, V. S. P. H., "A 531 nw/mhz, 128 x 32 current-mode vector matrix multiplier with over 2 decades of linear range," in *IEEE Conference on Custom Integrated Circuits , pp. 29-4-1-29-4-4.*, October 2004.

[22] Ramakrishnan, S., Basu, A., Chiu, L. K., Hasler, P., and Anderson, D., "Speech processing on a reconfigurable analog platform," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, p. In Press, 2012.

[23] Ramakrishnan, S., Basu, A., Chiu, L. K., Hasler, P., Anderson, D., and Brink, S., "Speech processing on a reconfigurable analog platform," in *Subthreshold Microelectronics Conference (SubVT), 2012 IEEE*, pp. 1 –3, oct. 2012.

[24] Ramakrishnan, S. and Hasler, P., "The vmm and wta as an analog classifier," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, p. Accepted, 2012.

[25] Roy, S., Basu, A., and Hussain, S., "Hardware efficient, neuromorphic dendritically enhanced readout for liquid state machines," in *Biomedical Circuits and Systems Conference (BioCAS), 2013 IEEE*, pp. 302–305, IEEE, 2013.

[26] Schlottmann, C. and Hasler, P., "A highly dense, low power, programmable analog vector-matrix multiplier: The fpaa implementation," *IEEE JetCAS*, vol. In Print, 2011.

[27] Schlottmann, C., Petre, C., and Hasler, P., "A High-Level Simulink-Based Tool for FPAA Configuration," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. Issue:99, pp. 1–1, 2010.

[28] Schlottmann, C. R. and Hasler, P. E., "'a highly dense, low power, programmable analog vector-matrix multiplier: The fpaa implementation," *IEEE JetCAS*, 2011.

[29] Segev, I. and London, M., "Untangling dendrites with quantitative models," *Science*, vol. 290, p. 744, 2000.

[30] Shapero, S. and Hasler, P., "Precise programming and mismatch compensation for low power analog computation on an FPAA," *IEEE Transactions on Circuits and Systems I*, vol. 1, p. 1, submitted for review.

[31] Tapson, J. C., Cohen, G. K., Afshar, S., Stiefel, K. M., Buskila, Y., Wang, R. M., Hamilton, T. J., and van Schaik, A., "Synthesis of neural networks for spatio-temporal spike pattern recognition and processing," *Frontiers in neuroscience*, vol. 7, 2013.

[32] Wang, Y. and Liu, S.-C., "Input evoked nonlinearities in silicon dendritic circuits," *IEEE International Symposium on Circuits and Systems*, vol. 1, pp. 2894 – 2897, 2009.

[33] Wunderlich, R. B., Adil, F., and Hasler, P., "Floating gate-based field programmable mixed-signal array," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 21, no. 8, pp. 1496–1505, 2013.

[34] C. Mead, "Neuromorphic electronic systems", *IEEE Proceedings*, vol. 78, no. 10, 1990, pp. 1629-1636.

[35] J. Hasler and B. Marr, "Finding a roadmap to achieve large neuromorphic hardware systems," *Frontiers in Neuromorphic Engineering*, September 2013. pp. 1-29. doi:10.3389/fnins.2013.00118.

[36] H. B. Marr, B. Degnan, P. Hasler, and D. Anderson, "Scaling Energy Per Operation via an Asynchronous Pipeline," *IEEE Trans. on VLSI*, Vol. 21, No. 1, January 2013, pp. 147-151.

[37] C. Schlottmann, S. Shapero, S. Nease, and P. Hasler, "A Digitally-Enhanced Reconfigurable Analog Platform for Low-Power Signal Processing," *IEEE Journal of Solid State Circuits*, September 2012, vol. 47, no. 10, pp. 2174-2184

[38] R. Wunderlich, F. Adil, and P. Hasler, "A Floating Gate Based Field Programmable Mixed-Signal Array," *IEEE Transactions on VLSI*, vol. 21, no. 8, 2013, pp. 1496-1505.

[39] OpenMSP430 Project: open core MSP430. http://opencores.org/project,openmsp430

[40] A. Basu, S. Brink, C. Schlottmann, S. Ramakrishnan, C. Petre, S. Koziol, F. Baskaya, C. M. Twigg, and P. Hasler, "Floating-gate based Field Programmable

Analog Array," *IEEE Journal of Solid State Circuits*, vol. 45, no. 9, September 2010, pp. 1781-1794.

[41] A. Basu and P. E. Hasler, "A Fully Integrated Architecture for Fast and Accurate Programming of Floating Gates over Six decades of Current," *IEEE Transactions on VLSI*, June 2010.

[42] J. Luu, J. Goeders, M. Wainberg, A. Somerville, T. Yu, K. Nasartschuk, M. Nasr, S. Wang, T. Liu, N. Ahmed, K. B. Kent, J. Anderson, J. Rose and V. Betz "VTR 7.0: Next Generation Architecture and CAD System for FPGAs," *ACM TRETS*, Vol. 7, No. 2, June 2014, pp. 6:1 - 6:30.

[43] Baskaya, F. and Anderson, D.V. and Hasler, P. and Lim, S.K., "A generic reconfigurable array specification and programming environment (GRASPER)", *Europen Conference on Circuit Theory and Design*, 2009.

[44] C. R. Schlottmann, C. Petre, and P. E. Hasler, "Simulink Framework for Design to and Automated Conversion on Large-Scale FPAA Devices," *IEEE Transactions on VLSI*, February 2011.

[45] C. R. Schlottmann and J. Hasler, , "High-Level Modeling of Analog Computational Elements for Signal Processing Applications," *IEEE Transactions on VLSI Systems*, 2014.

[46] S. Koziol, C. Schlottmann, A. Basu, S. Brink, C. Petre, S. Ramakrishnan, P. Hasler "Hardware and Software Infrastructure for a Family of Floating-Gate FPAAs," *IEEE IEEE International Symposium on Circuits and Systems*, June 2010. Winner of the best demonstration paper award.

[47] C. Twigg and P. Hasler, "Incorporating Large-Scale FPAAs Into Analog Design and Test Courses," *IEEE Transactions on Education*, Vol. 51, No. 3, 2008, pp. 319-324.

[48] P. Hasler, C. Schlottmann, S. Koziol, S. Ramakrishnan, S. Brink, and A. Basu, "FPAA chips and tools as the center of an Design-Based Analog Systems Education," *IEEE MSE*, San Deigo, June 2011.

[49] C. Twigg, J. Gray, and P. Hasler, "Programmable Floating-gate FPAA switches are not dead weight," *International Symposium on Circuits and Systems*, May 2007, pp. 169-72.

[50] C. Schlottmann, and P. Hasler, "A highly dense, low power, programmable analog vector-matrix multiplier: the FPAA implementation," *IEEE Journal of Emerging CAS*, vol. 1, 2012, 403411.

[51] J. Lazzaro, S. Ryckebusch, M. A. Mahowald and C. A. Mead, Winner-take-all networks of O(N) complexity, *in Advances in Neural Information Processing Systems 1*, Morgan Kaufmann Publishers, CA, 1989.

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

[53] S. Ramakrishnan, A. Basu, L.K. Chiu, J. Hasler, D. Anderson, and S. Brink, "Speech processing on a reconfigurable analog platform," *IEEE VLSI Systems*, vol 22, no. 2, Feb. 2014, 430-433.

[54] L. Itti, C. Koch, E. Niebur, A Model of Saliency-Based Visual Attention for Rapid Scene Analysis, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 20, No. 11, pp. 1254-1259, Nov 1998.

[55] L. Itti, and C.Koch, "Computational Modeling of Visual Attention," *Nature Neuroscience*, vol. 2, March 2001, pp. 1.

[56] J. Hasler, "Physics based computing enabling energy efficiency past Moore's law," *IEEE GlobalSIP*, 2013, pp. 679 - 682

[57] S.Kim, J. Hasler and S. George "Integrated Floating-Gate Programming Environment for System-Level ICs," *IEEE TVLSI*, 2015, Submitted

[58] M. A. Sivilotti, Wiring Considerations in Analog VLSI Systems, With Application to Field-Programmable Networks (VLSI), Ph.D., California Institute of Technology, Pasadena, CA, 1991.

[59] Pankiewicz, B.; Wojcikowski, M.; Szczepanski, S.; Yichuang Sun, A field programmable analog array for CMOS continuous-time OTA-C filter applications, IEEE Journal of Solid-State Circuits, vol. 37, no. 2, Feb 2002, pp. 125-126.

[60] E.K.F Lee amd Gulak, P.G., Field programmable analogue array based on MOSFET transconductors, *Electronics Letters*, vol. 28, no. 1, 1992, pp. 28 29.

[61] C.A. Looby and C. Lyden, A CMOS Continuous-Time Field Programmable Analog Array, *FPGA*, 1997.

[62] Vincent Gaudet and Glenn Gulak. 10 MHz Field Programmable Analog Array Prototype Based on CMOS Current Conveyors, *Micronet*, 1999.

[63] T. Hall, Twigg, P. Hasler, and D. V. Anderson, Application performance of elements built in a Floating-gate FPAA, International Symposium on Circuits and Systems, May 2004.

[64] Joachim Becker, Yiannos Manoli, "A Continuous-Time Field Programmable Analog Array (FPAA) consisting of Digitally Reconfigurable Gm-Cells," *ISCAS* 2004.

[65] G. Cowan, R. Melville, and Y. Tsividis, "A VLSI analog computer/math coprocessor for a digital computer, *IEEE ISSCC*, pp. 82 83, 2005.

[66] C. Twigg and P. Hasler, "A Large-Scale Reconfigurable Analog Signal Processor (RASP)," *Custom Integrated Circuits Conference*, 2006.

[67] PSoC5 Data Sheet, Cyprus Semi, 2011.

[68] P. Lajevardi, A. P. Chandrakasan, and H.-S. Lee, "Zero-Crossing Detector Based Reconfigurable Analog System," *IEEE Journal of Solid State Circuits*, vol. 46, no. 11, Nov. 2011, pp. 2478-2487.

[69] S. Brink, J. Hasler, and R. Wunderlich, " Adaptive Floating-Gate Circuit Enabled Large-Scale FPAA," IEEE VLSI Systems, 2014.

[70] W. Wolf, "Hardware-software co-design of embedded systems," *Proceedings of the IEEE*, 1994, 967–989.

[71] C. Schlottmann, S. Shapero, S. Nease, and P. Hasler, A Digitally- Enhanced Reconfigurable Analog Platform for Low-Power Signal Processing, *IEEE Journal of Solid State Circuits*, vol. 47, no. 10, 2012, pp. 2174-2184.

[72] J. Becker and Y. Manoli. A continuous-time field programmable analog array (FPAA) consisting of digitally reconfigurable G M-cells. *IEEE ISCAS*, vol. 1, pages I–1092. IEEE, 2004.

[73] A. Basu, S. Brink, C. Schlottmann, S. Ramakrishnan, C. Petre, S. Koziol, F. Baskaya, C. Twigg, and P. Hasler. A Floating-Gate-Based Field Programmable Analog Array. *IEEE JSSC*, 45:1781–1794, 2010.

[74] G. Cowan, R. Melville, and Y. Tsividis. *A VLSI analog computer/math co-processor for a digital computer*. Columbia University, 2005.

[75] R. B. Wunderlich, F. Adil, and P. Hasler. Floating gate-based field programmable mixed-signal array. *IEEE TVLSI*, vol. 21, no. 8, 2013, pp. 1496–1505.

[76] P. Lajevardi, A. P. Chandrakasan, and H.-S. Lee. Zero-crossing detector based reconfigurable analog system. *JSSC*, 46(11):2478–2487, 2011.

[77] J. Luu, J. Goeders, M. Wainberg, A. Somerville, T. Yu, K. Nasartschuk, M. Nasr, S. Wang, T. Liu, N. Ahmed, K. B. Kent, J. Anderson, J. Rose, and V. Betz. VTR 7.0: Next Generation Architecture and CAD System for FPGAs. volume 7, pages 6:1–6:30, June 2014.

[78] S. Ramakrishnan and P. Hasler. The VMM and WTA as an analog classifier. *IEEE Trans on VLSI*, 2013.

[79] B. Rumberg and D. W. Graham. Reconfiguration Costs in Analog Sensor Interfaces for Wireless Sensing Applications. In *Midwest CAS*, 2013 pages 321–324.

[80] C. R. Schlottmann and J. Hasler, High-Level Modeling of Analog Computational Elements for Signal Processing Applications *IEEE Trans on VLSI*, 2014.

[81] Scilab Enterprises. *Scilab: Free and Open Source software for numerical computation.* Scilab Enterprises, Orsay, France, 2012.

[82] http://it.mathworks.com/solutions/fpga-design/

[83] C. Twigg, J. Gray, and P. Hasler, "Programmable Floating-gate FPAA switches are not dead weight," *International Symposium on Circuits and Systems*, May 2007, pp. 169-72.

[84] G. DeMicheli and R. K. Gupta, "Hardware/Software Co-Design," *Proceedings of the IEEE*, Vol. 85, no. 3, 1997, pp. 349-365.

[85] R. Gupta, "Hardware Software Co-design: Tools for Architecting Systems-On-A-Chip," *Asia and South Pacific Design Automation Conference*, 1997, pp. 285-289

[86] G. Schirner , A. Gerstlauer , and R. Domer, "System-level Development of Embedded Software," *Asia and South Pacific Design Automation Conference*, 2010, pp. 903-909.

[87] R. K. Gupta and F. Brewer, "High-level synthesis: A retrospective," in P. Coussy and A. Morawiec, *High-Level Synthesis: From Algorithm to Digitial Circuit*, Springer Netherlands, Chapter 2, 2008, pp 13-28.

[88] L. Shang and N. K. Jha, "Hardware-Software Co-Synthesis of Low Power Real-Time Distributed Embedded Systems with Dynamically Reconfigurable FPGAs," *International Conference on VLSI Design (VLSID02)*, 2002.

[89] Digital Partitioning for Field-ProgrammableMixed Signal Systems* Sree Ganesan and Ranga Vemuri ARVLSI, 2001, pp. 172- 185

[90] Exploration-Based High-Level Synthesis of Linear Analog Systems Operating at Low/Medium Frequencies Alex Doboli, Member, IEEE, and Ranga Vemuri, Senior Member, IEEE 1556 - 1568 IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN OF INTEGRATED CIRCUITS AND SYSTEMS, VOL. 22, NO. 11, NOVEMBER 2003

[91] D. Rossi, C. Mucci, M. Pizzotti, L. Perugini, R. Canegallo, and R. Guerrieri, "Multicore Signal Processing Platform with Heterogeneous Configurable hardware accelerators," *IEEE Trans. on VLSI*, vol. 22, no. 9, 2014, pp. 1990-2003.

[92] Qian Zhao, Motoki Amagasaki, Masahiro Iida, Morihiro Kuga and Toshinori Sueyoshi, "An Automatic FPGA Design and Implementation Framework," DAC, 2013.

[93] M. Weinhardt, A. Krieger, T. Kinder, "A Framework for PC Applications with Portable and Scalable FPGA Accelerators ," DAC, 2013.

[94] Zynq: All Programmable SoC Architecture, 2012. http://www.xilinx.com/products/silicon-devices/soc/index.htm.

[95] SoC FPGAs: Integration to Reduce Power, Cost, and Board Size, 2012. http://www.altera.com/devices/processor/soc-fpga/proc-soc-fpga.html.

[96] http://www.altera.com/products/software/products/dsp/dsp-builder.html

[97] K. Huang1, S. Han2,3, K. Popovici3, L. Brisolara4, X. Guerin3, "Simulink-Based MPSoC Design Flow: Case Study of Motion-JPEG and H.264,"

[98] Lei Li1, Xiaolang Yan1, Soo-Ik Chae2, Luigi Carro4, Ahmed Amine Jerraya, DAC 2007, 39-42.

[99] Allman, W. F., *Apprentices of Wonder: Inside the Neural Network Revolution*. Bantam Books, 1989.

[100] Basu, A., Brink, S., Schlottmann, C., Ramakrishnan, S., Petre, C., Koziol, S., Baskaya, F., Twigg, C., and Hasler, P., "A Floating-Gate-Based Field Programmable Analog Array," *IEEE Journal of Solid-State Circuits*, vol. 45, pp. 1781–1794, 2010.

[101] Hasler, J. and Marr, B., "Finding a roadmap to achieve large neuromorphic hardware systems," *Frontiers in neuroscience*, vol. 7, 2013.

[102] Luu, J., Goeders, J., Wainberg, M., Somerville, A., Yu, T., Nasartschuk, K., Nasr, M., Wang, S., Liu, T., Ahmed, N., Kent, K. B., Anderson, J., Rose, J., and Betz, V., "VTR 7.0: Next Generation Architecture and CAD System for FPGAs," vol. 7, pp. 6:1–6:30, June 2014.

[103] Mead, C., *Analog VLSI and Neural Systems*. Reading, MA: Addison-Wesley, 1989.

[104] Ramakrishnan, S. and Hasler, P., "The vmm and wta as an analog classifier," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, p. Accepted, 2012.

[105] Schlottmann, C. and Hasler, P., "A highly dense, low power, programmable analog vector-matrix multiplier: The fpaa implementation," *IEEE JetCAS*, vol. In Print, 2011.

[106] Schlottmann, C., Petre, C., and Hasler, P., "A High-Level Simulink-Based Tool for FPAA Configuration," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. Issue:99, pp. 1–1, 2010.

[107] Wunderlich, R. B., Adil, F., and Hasler, P., "Floating gate-based field programmable mixed-signal array," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 21, no. 8, pp. 1496–1505, 2013.

[108] Ananda Maiti et.al, "Merging Remote Laboratories and Enquiry-based Learning for STEM Education," *IJOE*, 2014.

[109] V.J. Harward et. al, "The iLab Shared Architecture: AWeb Services Infrastructure to Build Communities of Internet Accessible Laboratories," *Proceedings of the IEEE*, 2008.

[110] D. Lowe et. al, "Evolving Remote Laboratory Architectures to Leverage Emerging Internet Technologies," *IEEE Transactions on Learning Technologies*, 2009.

[111] N. Suosa et. al, 'An Integrated Reusable Remote Laboratory to Complement Electronics Teaching', IEEE Transactions on learning technologies 2010

[112] M. A. Bochicchio et. al, "Hands-On Remote Labs: Collaborative Web Laboratories as a Case Study for IT Engineering Classes", *IEEE Transactions on Learning Technologies*, 2009

[113] M. Cooper et. al, "Remote Laboratories Extending Access to Science and Engineering Curricular", *IEEE Transactions on Learning Technologies*, 2009.

[114] M. Collins, J. Hasler, and S. George, "An Open-Source Toolset enabling Analog-Digital-Software Codesign," *Submitted to IEEE Transactions on VLSI*, December 2014.

[115] W. Wolf, "Hardware-software co-design of embedded systems," *Proceedings of the IEEE*, 1994, 967–989.

[116] J. Becker and Y. Manoli. A continuous-time field programmable analog array (FPAA) consisting of digitally reconfigurable G M-cells. *IEEE ISCAS*, vol. 1, pages I–1092. IEEE, 2004.

[117] A. Basu, S. Brink, C. Schlottmann, S. Ramakrishnan, C. Petre, S. Koziol, F. Baskaya, C. Twigg, and P. Hasler. A Floating-Gate-Based Field Programmable Analog Array. *IEEE JSSC*, 45:1781–1794, 2010.

[118] G. Cowan, R. Melville, and Y. Tsividis. *A VLSI analog computer/math co-processor for a digital computer*. Columbia University, 2005.

[119] R. B. Wunderlich, F. Adil, and P. Hasler. Floating gate-based field programmable mixed-signal array. *IEEE TVLSI*, vol. 21, no. 8, 2013, pp. 1496–1505.

[120] P. Lajevardi, A. P. Chandrakasan, and H.-S. Lee. Zero-crossing detector based reconfigurable analog system. *JSSC*, 46(11):2478–2487, 2011.

[121] J. Luu, J. Goeders, M. Wainberg, A. Somerville, T. Yu, K. Nasartschuk, M. Nasr, S. Wang, T. Liu, N. Ahmed, K. B. Kent, J. Anderson, J. Rose, and V. Betz. VTR 7.0: Next Generation Architecture and CAD System for FPGAs. volume 7, pages 6:1–6:30, June 2014.

[122] S. Ramakrishnan and P. Hasler. The VMM and WTA as an analog classifier. *IEEE Trans on VLSI*, 2013.

[123] B. Rumberg and D. W. Graham. Reconfiguration Costs in Analog Sensor Interfaces for Wireless Sensing Applications. In *Midwest CAS*, 2013 pages 321–324.

[124] C. R. Schlottmann and J. Hasler, High-Level Modeling of Analog Computational Elements for Signal Processing Applications *IEEE Trans on VLSI*, 2014.

[125] Scilab Enterprises. *Scilab: Free and Open Source software for numerical computation*. Scilab Enterprises, Orsay, France, 2012.

[126] S. George, S. Kim, S. Shah, J. Hasler, M. Collins, F. Adil, R. Wunderlich, S. Nease and S. Ramakrishnan "A Programmable and Configurable Mixed-Mode FPAA SOC," *Submitted to IEEE Journal of Solid State Circuits*, December 2014.

[127] S. George, R. Wunderlich and J. Hasler, "CAD Synthesis Tools for Heterogeneous SoCs " *Submitted to IEEE Transactions on CAD*, December 2014.

[128] C. Mead, *Analog VLSI and Neural Systems*. Reading, MA: Addison-Wesley, 1989.

[129] C. Koch and I. Segev, "The role of single neurons in information processing," *Nature Neuroscience*, vol. 3, pp. 1171–1177, Nov. 2000.

[130] M. London and M. Hausser, "Dendritic computation," *Annual Review of Neuroscience*, vol. 28, pp. 503–532, Jul. 2005.

[131] J. G. Elias, "Artificial dendritic trees," *Neural Computation*, vol. 5, pp. 648 – 663, 1993.

[132] C. Rasche and R. J. Douglas, "Forward- and backpropagation in a silicon dendrite," *IEEE Transactions on Neural Networks*, vol. 12, no. 2, pp. 386 – 393, Mar. 2001.

[133] Y. Wang and S.-C. Liu, "Input evoked nonlinearities in silicon dendritic circuits," *IEEE International Symposium on Circuits and Systems*, pp. 2894 – 2897, 2009.

[134] C. Koch, *Biophysics of Computation.* New York, NY: Oxford University Press, 1999.

[135] B. Hille, *Ion Channels of Excitable Membranes, Third Edition.* Sunderland, MA: Sinauer Associates, Inc, 2001.

[136] S.-C. Liu, J. Kramer, G. Indiveri, T. Delbruck, and R. Douglas, *Analog VLSI: Circuits and Principles.* Cambridge, MA: The MIT Press, 2002.

[137] A. Basu, S. Brink, C. Schlottmann, S. Ramakrishnan, C. Petre, S. Koziol, F. Baskaya, C. Twigg, and P. Hasler, "A floating-gate-based field programmable analog array," *IEEE Journal of Solid-State Circuits*, vol. 45, pp. 1781–1794, 2010.

[138] C. Twigg, J. Gray, and P. Hasler, "Programmable floating gate fpaa switches are not dead weight," *Circuits and Systems, 2007. ISCAS 2007. IEEE International Symposium on*, pp. 169–172, May 2007.

[139] E. K. F. Lee and W. L. Hui, "A novel switched-capacitor based field-programmable analog array architecture," *Analog Integrated Circuits and Signal Processing*, vol. 17, no. 1-2, pp. 35–50, 1998.

[140] I. Segev, J. Rinzel, and G. M. Shepherd, *The Theoretical Foundation of Dendritic Function: Selected Papers of Wilfrid Rall with Commentaries.* Cambridge, MA: The MIT Press, 1995.

[141] C. Petre, C. Schlottmann, and P. Hasler, "Automated conversion of simulink designs to analog hardware on an fpaa," *Circuits and Systems, 2008. ISCAS 2008. IEEE International Symposium on*, pp. 500–503, May 2008.

[142] F. Baskaya, D. Anderson, P. Hasler, and S. K. Lim, "A generic reconfigurable array specification and programming environment," *Circuit Theory and Design, 2009. ECCTD 2009. European Conference on*, pp. 619–622, Aug. 2009.

[143] S. Koziol, C. Schlottmann, A. Basu, S. Brink, C. Petre, B. Degnan, S. Ramakrishnan, P. Hasler, and A. Balavoine, "Hardware and software infrastructure for a family of floating-gate based fpaas," *Circuits and Systems (ISCAS), Proceedings of 2010 IEEE International Symposium on*, pp. 2794–2797, May 2010.

[144] S. Shapero and P. Hasler, "Precise programming and mismatch compensation for low power analog computation on an fpaa," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, submitted for review.

[145] T. Branco, B. A. Clark, and M. Hausser, "Dendritic discrimination of temporal input sequences in cortical neurons," *Science*, vol. 329, no. 5999, pp. 1671–1675, September 2010.

[146] A. Destexhe, "Dendrites do it in sequences," *Science*, vol. 329, no. 5999, pp. 1611–1612, September 2010.

[147] P. Hasler, S. Koziol, E. Farquhar, and A. Basu, "Transistor channel dendrites implementing hmm classifiers," *Circuits and Systems, 2007. ISCAS 2007. IEEE International Symposium on*, pp. 3359 – 3362, 2007.

[148] A. Sharma and S. Hauck, "Accelerating fpga routing using architecture-adaptive a* techniques," in *Field-Programmable Technology, 2005. Proceedings. 2005 IEEE International Conference on*. IEEE, 2005, pp. 225–232.

[149] C. Lavin, M. Padilla, J. Lamprecht, P. Lundrigan, B. Nelson, and B. Hutchings, "Rapidsmith: do-it-yourself cad tools for xilinx fpgas," in *Field Programmable Logic and Applications (FPL), 2011 International Conference on*. IEEE, 2011, pp. 349–355.

[150] N. Steiner, A. Wood, H. Shojaei, J. Couch, P. Athanas, and M. French, "Torc: towards an open-source tool flow," in *Proceedings of the 19th ACM/SIGDA international symposium on Field programmable gate arrays*. ACM, 2011, pp. 41–44.

[151] F. Baskaya, S. Reddy, S. K. Lim, and D. V. Anderson, "Placement for large-scale floating-gate field-programable analog arrays," *IEEE TVLSI*, vol. 14, no. 8, pp. 906–910, 2006.

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

[153] A. Basu, S. Brink, C. Schlottmann, S. Ramakrishnan, C. Petre, S. Koziol, F. Baskaya, C. Twigg, and P. Hasler, "A Floating-Gate-Based Field Programmable Analog Array," *IEEE Journal of Solid-State Circuits*, vol. 45, pp. 1781–1794, 2010.

[154] J. Becker and Y. Manoli, "A continuous-time field programmable analog array (FPAA) consisting of digitally reconfigurable G M-cells," in *Circuits and Systems, 2004. ISCAS'04. Proceedings of the 2004 International Symposium on*, vol. 1. IEEE, 2004, pp. I–1092.

[155] G. Cowan, R. Melville, and Y. Tsividis, *A VLSI analog computer/math coprocessor for a digital computer*. Columbia University, 2005.

[156] P. Lajevardi, A. P. Chandrakasan, and H.-S. Lee, "Zero-crossing detector based reconfigurable analog system," *Solid-State Circuits, IEEE Journal of*, vol. 46, no. 11, pp. 2478–2487, 2011.

[157] R. B. Wunderlich, F. Adil, and P. Hasler, "Floating gate-based field programmable mixed-signal array," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 21, no. 8, pp. 1496–1505, 2013.

[158] B. Rumberg and D. W. Graham, "Reconfiguration Costs in Analog Sensor Interfaces for Wireless Sensing Applications," in *Circuits and Systems (MWSCAS), 2013 IEEE 56th International Midwest Symposium on.* IEEE, 2013, pp. 321–324.

[159] C. Schlottmann, C. Petre, and P. Hasler, "A High-Level Simulink-Based Tool for FPAA Configuration," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. Issue:99, pp. 1–1, 2010.

[160] S. Ramakrishnan and P. Hasler, "The VMM and WTA as an analog classifier," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, p. Accepted, 2012.

[161] S. Ramakrishnan, P. Hasler, C. Gordon "Floating-gate Synapses with spike-time-dependent plasticity," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, p. Accepted, 2012.

[162] C. Schlottmann and P. Hasler, "A highly dense, low power, programmable analog vector-matrix multiplier: The fpaa implementation," *IEEE JetCAS*, vol. In Print, 2011.

[163] C. M. Twigg, J. D. Gray, and P. E. Hasler, "Programmable floating gate fpaa switches are not dead weight," in *Circuits and Systems, 2007. ISCAS 2007. IEEE International Symposium on.* IEEE, 2007, pp. 169–172.

[164] Brink, S., Nease, S., Hasler, P., Ramakrishnan, S., Wunderlich, R., Basu, A., and Degnan, B., "A learning-enabled neuron array ic based upon transistor channel models of biological phenomena," *Biomedical Circuits and Systems, IEEE Transactions on*, vol. 7, no. 1, pp. 71–81, 2013.

[165] Ramakrishnan, S., Wunderlich, R., Hasler, J., and George, S., "Neuron array with plastic synapses and programmable dendrites," *Biomedical Circuits and Systems, IEEE Transactions on*, vol. 7, pp. 631–642, 2013.

[166] Ramakrishnan, S., "A SYSTEM DESIGN APPROACH TO NEUROMORPHIC CLASSIFIERS," PhD thesis, Georgia Institute of Technology, 2013.

[167] Farquhar, E., and P.Hasler, "A bio-physically inspired silicon neuron," *IEEE TCAS I*, vol. 52, pp. 477-488, 2005.

VITA

Suma George completed her PhD and M.S. in Electrical and Computer Engineering at Georgia Institute of Technology in 2015 and 2011 respectively. She completed her B.Tech in Electronics and Communication Engineering at GGSIPU, New Delhi,India. Her research interests are in the areas of neuromorphic systems, reconfigurable architectures, system IC design, mixed signal CAD tools, and speech recognition applications. She also has industry experience, working at Blackberry designing new system architectures as well as being part of a startup nSys (later acquired by Synopsys) modeling 100/40 GHz ethernet systems. In her spare time, she is an avid vocalist, amateur guitarist, and loves to compose music. She is a merry lass from the historic city of New Delhi. Her favorite sweets are Gulab Jamun and Jalebis!