Bio-Inspired Artificial Olfactory System

Ping-Chen Huang
Jan M. Rabaey

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2017-173
http://www2.eecs.berkeley.edu/Pubs/TechRpts/2017/EECS-2017-173.html

December 1, 2017

Bio-Inspired Artificial Olfactory System

by

Ping-Chen Huang

A dissertation submitted in partial satisfaction of the
requirements for the degree of
Doctor of Philosophy

in

Engineering - Electrical Engineering and Computer Sciences

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Jan Rabaey, Chair
Professor Kannan Ramchandran
Professor Bruno Olshausen

Fall 2015

Bio-Inspired Artificial Olfactory System

Abstract

Bio-Inspired Artificial Olfactory System

by

Ping-Chen Huang

Doctor of Philosophy in Engineering - Electrical Engineering and Computer Sciences

University of California, Berkeley

Professor Jan Rabaey, Chair

The escalating statistical variation of device behaviors in the nano-scale era has led to a search for alternative computational paradigms where randomness can be treated as opportunities rather than nuisance. This will allow computational systems to be built with circuits designed at the nominal case and have better access to the advantages of scaled technologies. Such computational paradigms may emerge as we explore the information processing in the biological sensory systems, which achieve unparalleled performance and energy efficiency with mediocre and unreliable components. Implementations of signal processing tasks such as feature extraction, learning, or recognition can especially benefit from bio-inspired computational models. In this thesis, we present a bio-inspired artificial olfactory system, which consists of an analog feature extraction front end that operates efficiently in the low-precision regime and a spike pattern classifier that exploits the randomness in circuits.

The analog front end implements a novel trainable feature extraction algorithm for metal-oxide gas sensor arrays. The algorithm extracts one composite feature of all analytes and transforms the sensor responses into concentration-invariant spike patterns. This composite feature is extracted by performing the gradient decent algorithm during training. This 6-channel analog frond end consumes 519 nW/channel in the training mode, and 463 nW/channel in the recognition mode.

The spike pattern classifier consists of a transformation of the input spikes into high-dimensional sparse vectors and a cortical memory model. The transformation is based on a random sampling scheme that can be efficiently performed with circuits exhibiting large parametric variations. Moreover, sparse representations allow fast and robust pattern storage and retrieval with associative memories such as the correlation matrix memory. Its realized today that hyper-dimensional computing architectures like this may be a perfect match to the emerging nano-scale devices. We show how this classifier can be densely and efficiently implemented in a 3-D CNFET-RRAM technology.

Dedicated to my parents.

Contents

List of Figures

List of Tables

Acknowledgments

Writing the acknoledgement reminds me of the time when I just arrived in Berkeley. Naive as I was, I didn't know what was ahead of me. I wouldn't be able to finish this thesis without the help and support from so many people along this long journey.

First, I would like to thank my research advisor, Prof. Jan Rabaey, for his constant support and guidance. From the time I was in a mist to find a way out, to the time I needed a lot of resources to get things done, he is always there and ready to help. His exceptional vision has shaped this thesis in a direction that I couldn't have imagined in the beginning. What has come out in the end is indeed a wonder to me.

I would also like to thank Prof. Bruno Olshausen, who led me into the field of neural computations. After taking his class, I made a significant progress. His passion has inspired me to dive into many interesting theories. That was the most enjoyable time in my PhD. I also have to thank Prof. Elad Alon and Prof. Kannan Ramchandran for serving in my qualifiying/thesis committee.

During the early stage of my research, I worked with Prof. Jajeet Roychowdhury and his student Prateek Bhansali. It was them who brought this research to biologically inspired computations and the olfactory system. I value the time of our weekly genuine brainstorming for alternative computing paradigms. Around the same time, I had chances to work with Prof. David Macii from Unversity of Trento and Prof. Lauri Koskinen from University of Turku. As a junior graduate student, I learned a lot from these two experienced researchers.

I have to thank William Mickelson and Allen Sussman from Prof. Alex Zettl's Lab to conduct gas sensor testings for us in their facilities. These sensors are tested with toxic gases. I couldn't imagine doing these experiments myself. With these measurement results, I was able to build models for gas sensors.

As for my first and last tapeout in Berkeley, it is an epic story. In the early phase, I received a lot of help from Ashkan Borna on block-level designs. Even he left Berkeley, he still spent time to discuss with me some design problems. In the later phase, I owe a lot of thanks to Jun-Chau Chien for unselfishly sharing his circuit design, tapeout, and testing experiences, and for brainstorming with me. I wouldn't be able to tapeout or test the chip successfully without his help. I would also like to thank Siva Thyagarajan for helping me on automating the simulation of large number of initial states. People whom I have consulted regarding the tapeout and testing also include Wen Li, Wenting Zhou, Yida Dhuan, Min-Han Hsieh, Nai-Chung Kuo, Tsung-Te Liu, Namsoeg Kim, Nathan Narevsky, Dan Yeager, and William Biederman. Thanks to Brian Richard for coordinating the tapeout shuttles and setting up the CAD tools. TSMC has kindly donated the silicon area. The working of this chip is largely attributed to their amazingly accurate models. I also thank Silitronics for performing the excellent wire bonding and SMT assembly, Wilson Chang for helping debug the scan chain, and Fred Burghardt for getting every test equipment ready.

In the later stage of my research, I had the chance to work with Prof. Subashih Mitra's and Prof. Philip Wong's groups from Stanford University and Pentti Kanerva from the Redwood

Neuroscience Institute. The sparkles from mixing these two fields contribute to Chapter 4 of this thesis and will be continued in the following years.

This research was funded by Giga-Scale Research Center (GSRC) and later by Systems On Nanoscale Information fabrICs (SONIC) Center. Both are parts of SRC STARnet Centers, sponsored by MARCO and DARPA.

I spent most of my graduate student life in the Berkeley Wireless Research Center (BWRC) and received a lot of support from Jan's group, BWRC faculty, and students. Special thanks to the BWRC staffs who had/have been taking take of our computing systems and daily administrative stuffs: Kevin Zimmerman, Bira Coelho, James Dunn, Greg Pearson, Tom Boot, Deirdre McAuliffe-Bauer, Olivia Nolan, Leslie Nishiyama, Sarah Jordan, and Candy Corpus. I also enjoyed the time with some students/researchers who always stayed up late in BWRC together and thank them for walking to my car in the middle of the night: Nai-Chung Kuo, Min-Han Hsieh, Wilson Chang, Bo Zhao, and Angie Wong.

Within the department, I also gained a lot of help from the EECS faculty, students, and staffs. Special thanks to Ruth Gjerde, and Shirley Salanio: other than taking care of the administrative work of our degree program professionally and efficiently, their kindness and patience make international students feel like home.

During my internship in the Kilby Lab of Texas Instrument, my manager Ali Kiaei had provided me a lot of freedom and support to help me finish testing and the thesis. I couldn't ask for a better manager.

Outside of the school, I thank the supports from Heart Chan practitioners. Although I started practicing Heart Chan meditation when I was young, I didn't notice its benefits until I picked it up again few years ago. It has helped cultivate the inner strength, and helped clear my mind to deal with research problems.

Finally, I have to thank my family for bearing me so long. Even during my mother's toughest time, they never asked me to quit and always encouraged me to keep going. No words can express my gratitude and love to them. Thanks to my boyfriend Shih-Yu Yeh, and Shu-Jen Lu. They are my best listeners and always on my side.

Thank you all so much!

Chapter 1

Introduction

We are nearing the end of a long, historically unique, and profoundly transformative trend in the device miniaturization of information technology. As devices shrink to molecular dimensions, current practices and assumptions will lead to increasingly inefficient systems. Emerging nano-devices have been actively sought as successors to the silicon CMOS field-effect transistors (FETs). Although they are not immediately ready to replace silicon FETs, some of their unique properties have opened a new door for alternative computing paradigms that could build much more efficient systems. In the first part of this chapter, the motivation to explore new architectures is given. Next, the promise of neuro-inspired computing and its return are discussed. Finally, the research goal is stated and the organization of this dissertation is provided.

1.1 The Need for New Computing Models

1.1.1 The End of Moore's Law[1]

At the end of Moore's Law, we are getting diminishing gains from technology scaling. Figure 1.1 shows the microprocessor trend over the past 40 years [1]. Even if transistor count keeps rising, the clock frequency leveled off a decade ago - the power has reached the limit. Therefore, as the number of cores increases exponentially, the improvement of single-thread performance is only incremental.

What makes technology scaling worse is the increasing variability. As technology scales from 45 nm to 12 nm, the threshold variation increases by almost four times [2]. Although the threshold voltage (and supply voltage) has stopped scaling down, this will still cause severe reliability issues. To ensure reliable operation, the overhead from adding more and more operational margins will soon offset the intrinsic advantage of scaling.

The other problem of scaling the planar CMOS technology is the on-chip interconnect, which has taken up 90% of volume in an IC chip today. Figure 1.2 shows the scaling trends

[1]To be more accurate, it's actually Denard's Law that is ending.

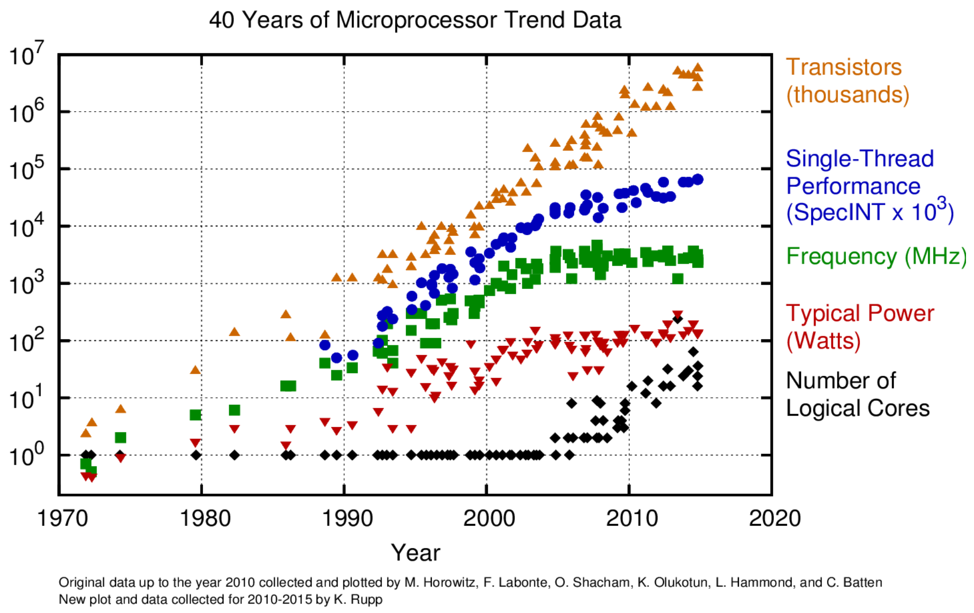


Figure 1.1: Microprocessor trend over the past 40 years. Adapted from [1].

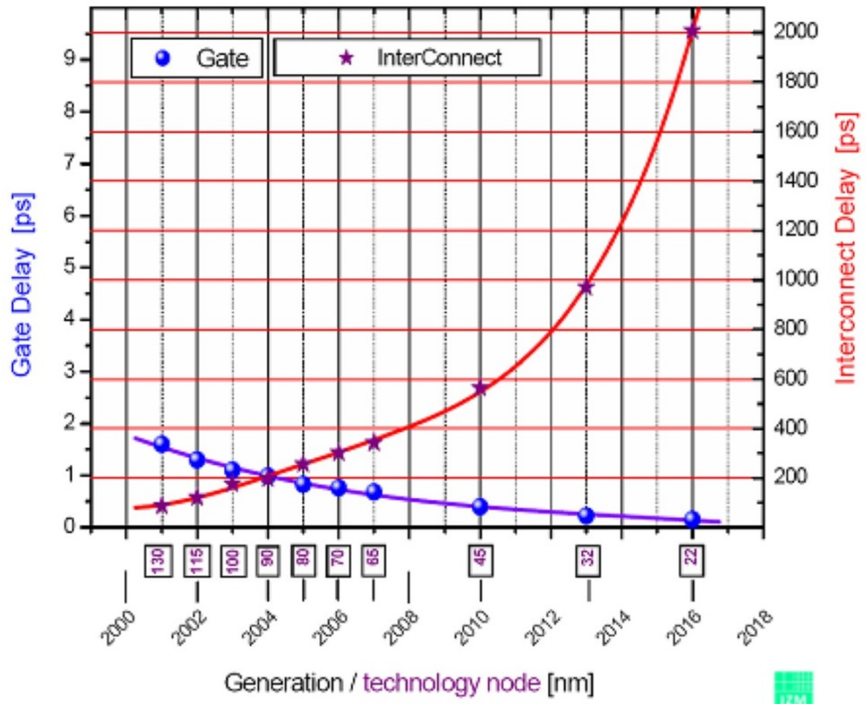


Figure 1.2: Scaling trends of gate and interconnect delays. Adapted from [3].

of gate and interconnect delays [3]. The interconnect RC delays are increasing exponentially with scaling. Therefore, interconnect is dominating device power consumption, performance and cost.

Eventually, the semiconductor industry will need a approach to supplement and extend CMOS beyond its fundamental scaling limit.

1.1.2 Emerging Nanotechnologies

Fundamental Scaling Limits of Charge-Based Logic Devices

How much further can the present CMOS-based technology be scaled? A theoretical analysis [4], [5] based on a two-well model for the source-gate-drain structure of an FET (Figure 1.3) derived the minimum barrier width a_{min} and height E_{bmin} such that a electron is either in the drain or source with 50% probability by thermionic emission or quantum mechanical tunneling. It is shown that $a_{min} = 1.5$ nm and $E_{bmin} = kTln2$, where k is the Boltzmann constant, and T is the absolute temperature. An additional constraint requiring minimization of energy dissipated in changing potential well sets $a_{min} = 5$ nm.

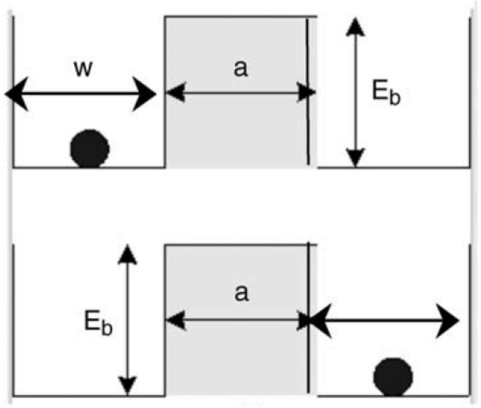


Figure 1.3: Idealized model of the potential barrier separating the source drain potential wells, where w is the well width, a is the barrier width, and E_b is the barrier energy. Adapted from [5].

Since high-volume manufacturing technology for the 16 nm node is currently being ramped to full production, the 5 nm generation may go into manufacturing in 7 years or in 2020. Consequently, scaling of CMOS-based information technology will face fundamental limits within the next 5-7 years or by 2020. This determines a need to have any new technology for extending CMOS in place by that time.

Emerging Logic Devices

For emerging logic devices, the quantitative assessment and benchmark by ITRS [6] and investigations between 2011-2013 [7] suggest that substantial progress has been made in demonstrating devices beyond conventional CMOS. Specifically, a bifurcation is observed that some devices appear to hold promise for higher speed operation than CMOS (e.g. Mott FET, carbon nanotube FET (CNFET)), while others might provide a route to lower power consumption (e.g. nanowire FET, negative capacitance ferroelectric). For example, an ideal CNFET inverter at 32-nm can have 13X of improvement in Energy-Delay-Product compared to a 32-nm CMOS inverter [8]. At 22-nm and 16-nm nodes, the advantage over silicon CMOS can improve to 20X [9]. This is due primarily to near ballistic electrical transport in CNTs. However, each emerging device still has challenges to overcome before any of them can be considered as an "heir" to the silicon CMOS FET. In CNFET technology, for example, the challenges remain controlling the percentage of metallic CNTs grown or deposited on substrate, CNT alignment and positioning, getting high density aligned CNTs [10], and the difficulty of addressing individual CNTs with source, drain, and gate contacts within a dense, interconnected circuit [7].

Fundamental Scaling Limit of Charge-based Memory

Current memory technologies (DRAM, SRAM, and flash) are based on storing electron charges in a storage node. States 0 and 1 are created by the presence or absence of electrons in the charge storage node. A theoretical analysis based on a barrier model (Figure 1.4) [11] shows that the minimum barrier height E_{bmin} to suppress thermionic emission and the minimum barrier width a_{min} to suppress tunneling are $E_{bmin} = 1.42$ eV, and $a_{min} \approx 5$ nm, for a retention time of 10 years[2].

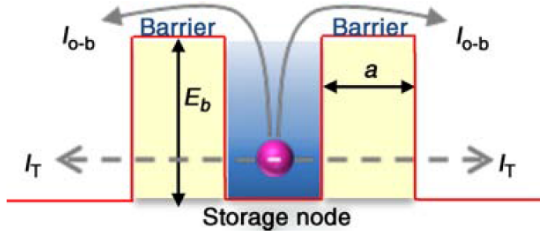


Figure 1.4: Barrier model of charge-based memory. Adapted from [11].

However, both DRAM and flash cannot reach this theoretical limit. From the Space-Action metric [12], DRAM achieves optimum performance in the 40-50 nm regime. As the size of the DRAM approaches the scaling limit, the access time will dramatically increase due to the serial resistance of the cell capacitor. Also, there is still no known manufacturable solution for DRAM beyond the 20 nm node [13].

The scalability of flash memory is limited by the gate stack - practical minimum size of a floating gate cell is roughly 10 nm. As of the time of this writing, production NAND

[2]This is for non-volatile memories. For smaller retention times, e.g., 50-100 ms, typical for DRAM, E_{bmin} and a_{min} can be smaller.

flash has been scaled to a critical dimension of 16 nm [14] and 3D Vertical NAND [15, 16] has entered commercial production. However, endurance and rention have become strongly degraded as flash tunnel oxides become thinner, leading to extensive error correction code (ECC) schemes and substantial redundant storage requirements [17].

In contrast, SRAM scalability has no obvious hard limiting factor to prevent it from reaching the theorectical limit [11]. However, a significant challenge for SRAM scaling is the increasing variability that will result in the failure of the cell. Moreover, minimum cell area cannot be maintained below 45 nm, if the minimum supply voltage and data retention voltage are to be minimized [18].

Emerging Memory Devices

According to the taxonomy of memory technologies from ITRS [13], emerging memory devices include ferroelectric memory and resistance switching memories (RRAM). Among RRAM, Redox RAM (ReRAM) technologies have advanced rapidly for the past few years. Overall ReRAM assessment is similar or better than existing CMOS-based nonvolatile memories [19]. This is due to its promising scalability, dense integration, and CMOS-compatibility. Some recent breakthroughs in ReRAM significantly enhanced its perceived potential, for example, a 32-Gb multi-layer NAND-scale array demonstration [20].

Scalability below 10 nm is predicted for all emerging memories. For ReRAM, it's operation and switching relies on the conductive filament in the metal-oxide, as shown in Figure 1.5 [21]. Since the size of the conductive filament is less than 10 nm, in principle it can scale down to sub-10 nm regime [22]. It's also reasonable to conclude that other emerging memories that rely on filamentary switching will scale to the same dimensions as standard filamentary ReRAM.

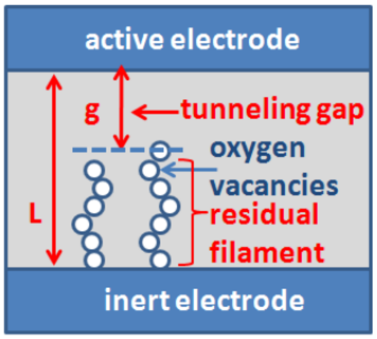


Figure 1.5: Conductive filament in the metal-oxide RRAM. Adapted from [21].

Except for ferroelectric memory, all of the emerging memory technologies are two terminal resistance switching devices. This makes them amenable to an efficient crossbar configuration. This is the most dense possible configuration, as each individual device only occupies an area of $4F^2$ [17], where F is the minimum feature size of the device. The other advantage of

ReRAM is that the fabrication temperature is CMOS back-end-of-the-line (BEOL) process compatible and no special material is required.

Generally, all emerging memories are scored low in the "operational reliability" category [17]. This is due to their variability in switching between the low-resistance state (LRS) and the high-resistance state (HRS) and also the lack of long term reliability.

However, the memresistive behavior [23] exhibited in the emerging memories have enabled efficient implementation of synaptic function that is very difficult to realize by devices in conventional technologies. This has enabled alternative computing paradigms that could be much more efficient than today's von Neumann-based systems.

The 3D Nano Promises and Challenges

As mentioned previously, on a deeply-scaled CMOS chip, an enormous amount of power and performance compromise are wasted on the interconnects. The greatest barrier holding today's processors from reaching higher computing speeds lies not in transistors but in the communication with memory. However, recent breakthroughs in monolithic 3D integration have demonstrated a visible new path ahead: from 3D integration of CNFET logics [24], 3D integration of CNFETs with silicon CMOS logics [25], to the 3D integration of CNFET/silicon CMOS logic and ReRAM memory [26]. The most important breakthrough is to interweave memory devices with computing devices. Figure 1.6 shows a four-layer prototype built by Stanford researchers. The bottom and top layers are logic transistors. Sandwiched between them are two layers of memory. The use of inter-layer vias (ILVs) to connect between various layer allows massive vertical interconnect density [26], maximizing the benefits of 3D integration. This has vastly reduced the "commuting time" between the processor and the memory, and will make the a chip much faster.

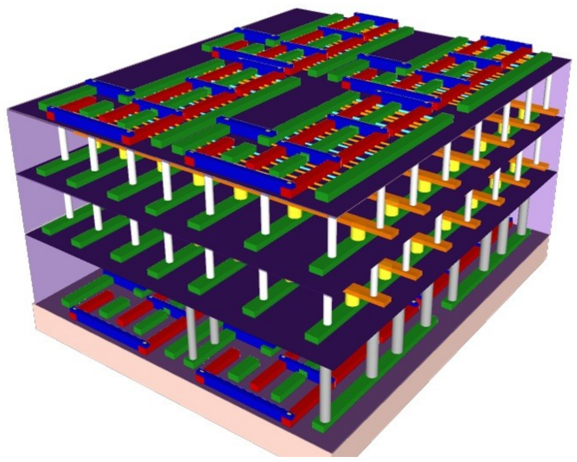


Figure 1.6: Monolithic 3D integration of logic and memory devices. Credit: M. Shulaker.

Although the 3D integration technology offers great density, the large variabilities and error/failure rates of the emerging logic and memory devices pose a great challenge for building a reliable system. Moreover, to reduce energy per operation and to minimize the heat flux, circuits built from such devices essentially must operate in a low signal-to-noise ratio (SNR) regime. A system requiring high SNR operation will simply fail, or will need overly designed circuits with large operational margins to work - this option will no longer be affordable as the system size scales up. Therefore, before device research comes up with solutions for making better and more reliable devices, we may need to seek solutions at the architectural-level: could there be an alternative computing model that can perform reliable computation efficiently with large number of devices exhibiting significant variations and failures? This has therefore led to a search for new computing models.

1.2 The Search for New Computing Models

Figure 1.7 illustrates the concept of traditional and alternative computing paradigms. In the traditional platform, the information processing architecture/algorithms presume its underlying computations are executed deterministically and reliably, whereas the devices today exhibit large variations. Therefore, the system level reliability is usually achieved by circuit/logic level reliability. As mentioned above, the cost of make each computing element precise and reliable will soon become unaffordable and offset the intrinsic advantages of scaled technologies. Therefore, the mapping is very inefficient.

In the alternative paradigms, we are looking for architecture/algorithms that expect the statistical natures in the computing elements, so that reliable computation can be achieved without the devices needing to behave precisely and reliably. Hence the circuits do not need to be over-designed and can be designed in the efficient regime. Therefore, the mapping is inherent efficient.

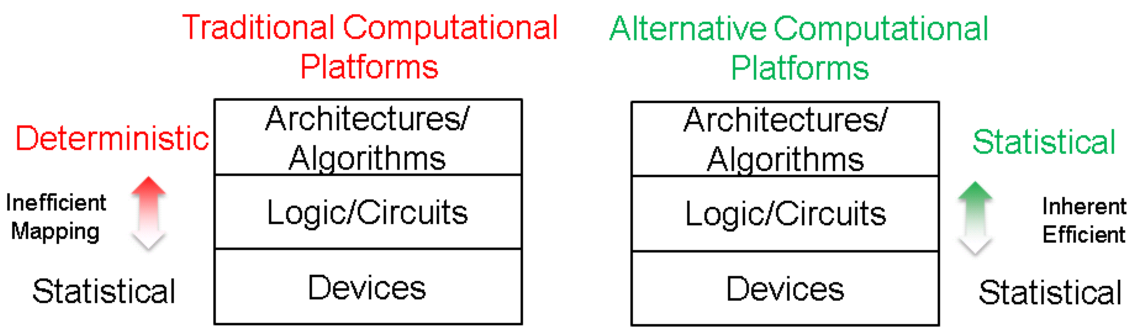


Figure 1.7: Traditional and alternative computing paradigms.

In fact, the natural world represents a gold mine of problem solving algorithms, yet this potential is largely untapped. Collective behaviors that operate without central control to regulate activity and growth are ubiquitous in nature. For example, neurons act collectively as networks to produce sensations, cognition and motions, while each individual neuron is prone to variations and failure. Many animal groups regulate their movement without a leader, such as bird flocks or fish schools. Social insects live in colonies, which forage, maintain their nests, and reproduce based on simple local cues and interactions. It is shown that three environmental constrains might have shaped the nature's evolution of collective behavior [27]: the patchiness of the resource, the operating costs of maintaining the interaction network that produces collective behavior, and the threat of rupture of the network.

Since a large network of today's nano-devices is very similar to a network of biological organisms, future nano technologies may need to be used in computing paradigms like those in nature, where functions emerge as the collective behavior of the devices interacting locally and does not require each individual to be precise and reliable. Among natural computations, neural computation is one of the most sophisticated and advanced. While it is not considered general purpose, the brain performs functions such as feature extraction, recognition, learning, and decision making very well. Therefore, neuro-inspired computing may lead to cheap and efficient processors for cognitive applications.

1.3 The Return of Neuro-Inspired Computing

1.3.1 The Recurring Waves of Neuro-Inspired Computing

Speaking of neuro-isnpired computing, it's not a new term. It has seen few times of booms and winters over the past several decades.

The first wave occurred between the 40's and the 60's. It began with McCullogh-Pitts' model (1943), Hebbian learning (1949), and then Frank Rosenblatt's perceptron (1958). Since a single-layer of perceptrons was shown to have severe limitations for solving complex problems in Minskey and Papert's book *Perceptrons* (1969), all the research funds was diverted to the field later called artificial intelligence (AI). Hereafter, virtually no research at all was done in connectionism for 10 years.

The second wave occurred between the 80's and the 90's. In 1982, John Hopfield introduced a form of neural network that can learn and retrieve patterns in a completely new way. Around the same time, David Rumelhard proposed a new method called back-propagation to train multi-layer perceptrons. These two discoveries revived the field of connectionism. During this period, Carver Mead recognized that neuro-inspired computing could be promising for building more efficient computing systems and had promoted *neuromorphic* systems [28]. However, the technology at that time was not able to make it an attractive alternative. There was no efficient and simple way to implement adaptive function either, making it difficult to build large learning networks.

The third wave is happening right now. It is because we are getting better understanding of some neural functions. Between late 80s and early this century, there have been many important theories developed, such as the Boltzman machine [29] (which later evolved into deep learning), Sparse Distributed Memory (SDM) [30] (later generalized to hyperdimensional computing [31]), and the Sparse Coding [32], etc. Also, the success of deep learning (e.g. Google Brain) has triggered a surge of efforts to build machines that can process data in much the same way the brain does [33]. In addition, the emerging nanotechnologies have enabled some neural functions and architectures that were previously very difficult or impossible to be built. Therefore, if anyone can figure out how the brain makes use of noisy neurons to compute and apply it to the emerging technologies, the information technology could be revolutionized. These are why there is significant investment today in combining the research of computational neuroscience and engineering.

1.3.2 Features of Neural Computation

The brain is a wonder of millions of years of evolution. Here are some of the important features of neural computation.

- Brains are *the most energy-efficient systems in the world*: The human brain performs more than 10^{16} FLOPS (floating-point operations per second) with only 20 W of power consumption [34]. This is 2-3 orders of magnitude more efficient than today's most energy-efficient silicon equivalence.

- *Robustness in the presence of component defects and variations*: For example, human can have complete visual color perception with incomplete deployment of color receptors [35]. Also even though neural responses are highly variable (with Fano factor $\sigma/\mu \approx 1$ [36]), the neural system can still perform reliable functions.

- *Amazing performance with mediocre components*: The neural systems do not have precise and excellent neural components. However, the use of collective computation by low to moderate-precision units can actually perform more complex and precise computations. For example, the auditory system can tell difference of time arrival within 10 μs with cells having time constant of 1 ms [34]. The olfactory system can discriminate 10^4-10^5 of odors with slight differences of chemical structures using olfactory receptors having broad reception ranges [37].

- *Seamless interaction of the analog world*: Each sensory modality uses specific kind of sensor to sense analog information from the world. The useful information or features are extracted using analog processing. It was point out that the use of physical functions in analog computation is the key to its energy efficiency [28].

- *3D integration of memory and computation*: Unlike a digital microprocessor where there is a centralized computing unit and a separate memory unit, the computation and memory are intimately integrated and distributed in the brain. Therefore, there is no 'von Neumann bottleneck' as in a microprocessor. The brain has closely coupled its algorithms with the implementation [38], unlike the standard engineering practices where algorithm design is mostly independent from how it will be implemented. Moreover, the fantastic 3D interconnect technology allows a fan-in and fan-out of 6,000 connections per neuron [34] compared to roughly 6 of a logic gate in the microprocessor. These might also be why the brain has achieved unparallel performances.

1.3.3 Opportunities of Neuro-Inspired Computing

Given the features of neural computation and the properties of current and emerging technologies, we can summarize the opportunities of neuro-inspired computing as follows.

- *To exploit the properties in neural computing*, including robustness to variations/failures and efficient low resolution processing through massively parallel computing, and to ensure efficient operation in the most informative regions of the input statistics through adaptation and feedback.
- *To efficiently realize some hard cognitive problems*, such as artificial olfaction, vision, classification, detection, and decision making, etc.
- *To leverage the opportunity while mitigating the challenges in the emerging nanotechnologies.* For example, hyper-dimensional computing paradigm [31] that is very difficult or even impossible in a 2D platform can be efficiently implemented in a 3D logic and memory integration platform while be very robust to device variations and failures.

1.4 Research Goal

The research goal of this thesis work is to develop an efficient neuro-inspired implementation for feature extraction and classification. As a demonstration, we have chosen the application to be artificial olfaction. First, it's because gas sensor signals are considered static and easier to deal with, compared with audio and video signals. Second, since so far there are very rare research efforts or commercial products of integrated signal processing front end for gas sensor arrays, we believe we can have good contributions to this field. Third, other than traditional applications in explosive detection, in food or pharmaceutical industry, it is predicted that there is an increasing market of integrated gas sensors in smart phones or handsets for health monitoring (e.g. breath analysis) and environmental monitoring [39]. Therefore, an integrated artificial olfactory system may find its application soon.

1.5 Thesis Outline

The thesis is organized as follows.

Chapter 2 provides an overview of the information processing in the sensory pathway, with an emphasis on the olfactory system. Based on the understanding of the biological sensory pathway, the architecture for the artificial olfactory system is proposed, which consists of an analog feature extraction front end and a spike pattern classifier. Finally an information-theoretic framework is developed to evaluate the energy efficiency of signal representations and to systematically optimize a design at both the architectural and circuit level.

Chapter 3 describes the design of the analog gas sensing front end. The analog front end implements a novel trainable feature extraction algorithm for metal-oxide gas sensor arrays. The algorithm extracts one composite feature of all analytes and transforms the sensor responses into concentration-invariant spike patterns. This composite feature is extracted by performing the gradient decent algorithm during training. This 6-channel analog frond end consumes 150nW/channel in the training mode, and 250nW/channel in the recognition mode.

Chapter 4 describes the spike pattern classifier consisting of a sparse transformation of the spike patterns and a cortical memory model. The transformation is based on a random sampling scheme that can be efficiently performed with circuits exhibiting large parametric variations. Moreover, sparse representations allow fast and robust pattern storage and retrieval with associative memories such as the correlation matrix memory. As mentioned in previous sections, hyper-dimensional computing architectures like this may be a perfect match to the emerging nanotechnologies. We show how this classifier can be densely and efficiently implemented in a 3-D CNFET-RRAM technology.

Chapter 5 concludes by summrizing the key contributions and suggesting directions of future research.

Chapter 2

System Overview

In this chapter, an overview on the biological olfactory pathway is given, including the sensing mechanism in the olfactory epithelium (OE), the combinatorial coding in the olfactory bulb (OB), and the overcomplete representation in the primary olfactory cortex (OC). Next, a signal processing architecture inspired by the olfactory pathway is proposed for artificial olfaction. Finally, an information-theoretic framework and a joint architectural and circuit level optimization methodology are provided to systematically determine the optimum architecture for efficiency.

2.1 The Biological Olfactory Pathway

Probably the oldest sensory system in the nature, the olfactory system concerns the sense of smell. Through the sense of smell, humans and other mammals can perceive a vast number and variety of chemicals in the external world. It is estimated that humans can sense 10^4 to 10^5 chemicals as distinct odors. Even a slight change in the chemical structure of an odorant can change its perceived odor. How do mammals detect so many different environmental chemicals? And how does the brain translate those chemicals into diverse odor perceptions?

Unlike visual and auditory systems, the mechanisms of mammalian olfaction remained a mystery until early this century. The discovery of the olfactory receptors and the combinatorial coding in the mammalian olfactory system by Richard Axel and Linda Buck was awarded the Nobel Prize in 2004. The first half of this section is primarily from the Nobel Lecture on their discovery [37]. The second half describes the computational principle behind the sparse and overcomplete representations in the primary cortex [32], [40], [41] and how such representation might be subsequently processed.

The signal processing in the olfactory system can be roughly divided into three stages, as shown in Figure 2.1. The first stage is in the **olfactory epithelium (OE)**, where odorants are initially detected by millions of olfactory sensory neurons. These neurons then transmit signals to the **olfactory bulb (OB)** of the brain, which then relays those signals to the **olfactory cortex (OC)**. From there, olfactory information is sent to a number of other brain areas, including higher cortical areas thought to be involved in odor discrimination, and deep limbic areas thought to mediate emotional and physiological effects of odors.

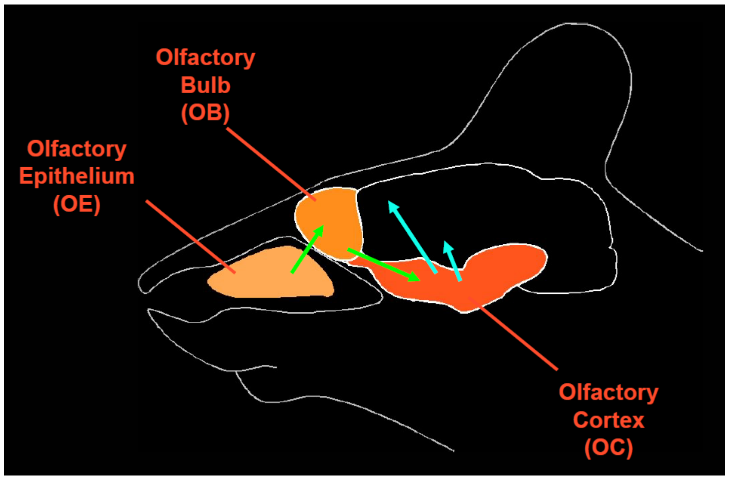


Figure 2.1: The olfactory pathway. Adapted from [37].

2.1.1 Sensing in the Olfactory Epithelium

The olfactory epithelium contains millions of *olfactory sensory neurons*, as shown in Figure 2.2. At the surface of the epithelium, each neuron extends cilia into the nasal lumen, allowing it to be in contact with odorants dissolved in the nasal mucus. Each neuron communicates with the brain via a single axon extending to the olfactory bulb.

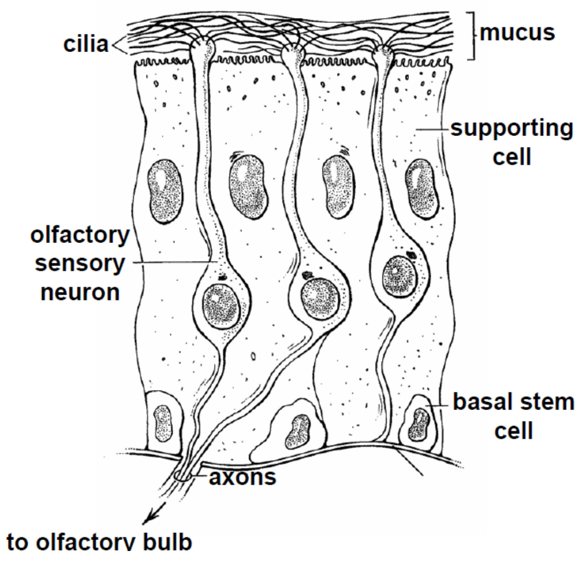


Figure 2.2: Structure of the olfactory epithelium. Adapted from [37].

Olfactory Receptors

How do these neurons detect odorants? In Axel and Buck's experiments, they discovered a family of G protein-coupled receptors (GPCRs) that are expressed exclusively by olfatory sensory neurons. These receptors, though related, vary extensively in their amino acid sequences. This is how the receptors interact with odorant with different structures. Figure 2.3 shows the structure of one olfactory receptor. Odorants can bind to the extracellular part of the receptor and activate the G protein, which will initiate subsequent messengers and trigger a change of gene expression within the nucleus.

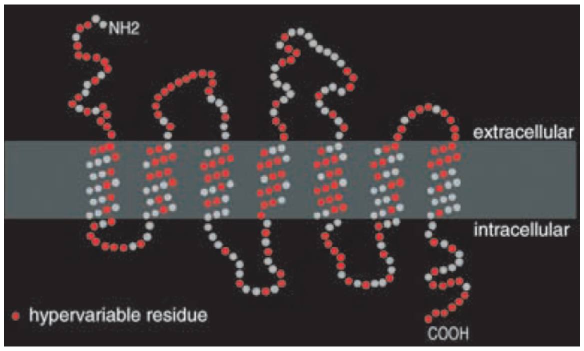


Figure 2.3: Topology of an odor receptor in the membrane. Individual amino acid residues are indicated by balls. Red balls indicate residues that vary extensively among odor receptors. Adapted from [37].

Later studies indicate that humans have about 350 different olfactory receptors and mice have about 1000. Each neuron can express only one olfactory receptor gene, and each receptor gene is expressed in several thousands of neurons. Neurons expressing the same receptor gene are distributed within one zone in the olfactory epithelium, as shown in Figure 2.4.

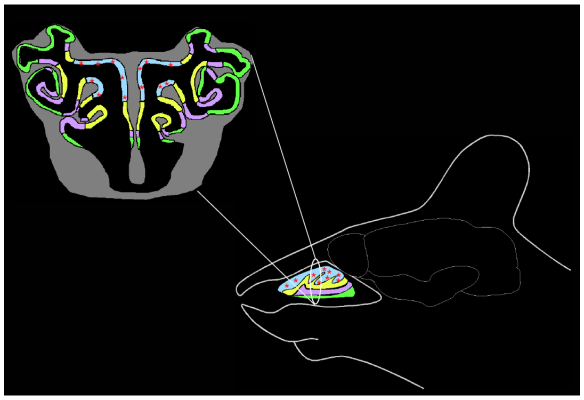


Figure 2.4: Expression zones in the olfactory epithelium. Zones expressing different receptor genes are marked in different colors. Adapted from [37].

The Combinatorial Coding

How do olfactory receptors respond to odorants? By examing the responses of different receptors to different odorants, it is concluded that odor receptors are used combinatorially to encode odor identities. As conceptually illustrated in Figure 2.5, each olfactory receptor can detect multiple odorants, and each odorant can activate multiple different receptors. It was shown that such combinatorial code can actually maximize the mutual information [42].

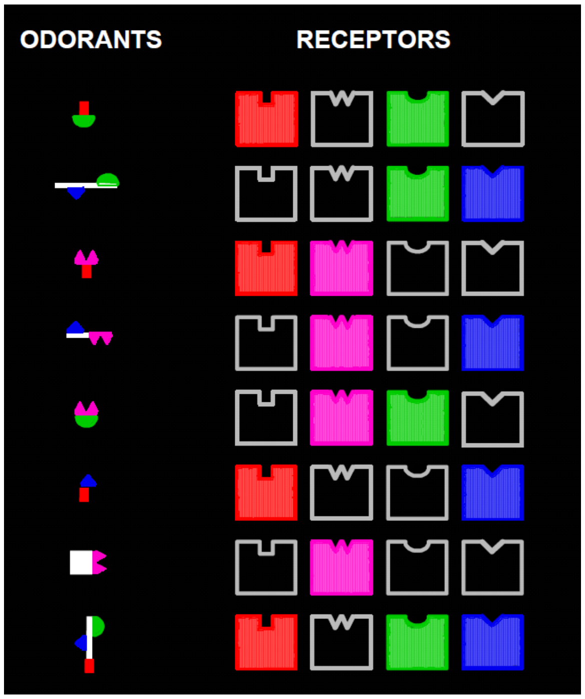


Figure 2.5: Combinatorial receptor codes for odors. Adapted from [37].

2.1.2 Signal Integration in the Olfactory Bulb

How does the brain translate an odorant's combinatorial receptor code into a perception? Each olfactory sensory neuron in the olfactory epithelium sends a single axon to a spherical structure called glomerulus in the olfactory bulb. Since there are only 2000 glomeruli while there are 10 million olfactory sensory neurons, there is a massive convergence in the olfactory bulb. Axons of thousands of olfactory sensory neurons expressing the same olfactory receptor converge in only 2-4 glomeruli dedicated to the same olfactory receptor, as shown in Figure 3.42. This allows a high degree of signal integration.

This further indicates that sensory information that is broadly organized and interspersed in different zones in the nose is transformed into a stereotyped sensory map in the bulb. Remarkably, this map is virtually identical in different individuals. This bulb map is likely to be important for stimulation of odor memories. Sensory neurons in the epithelium are short lived and are continuously replaced. However, the bulb map remains constant over time, assuring odorants can elicit distant memories.

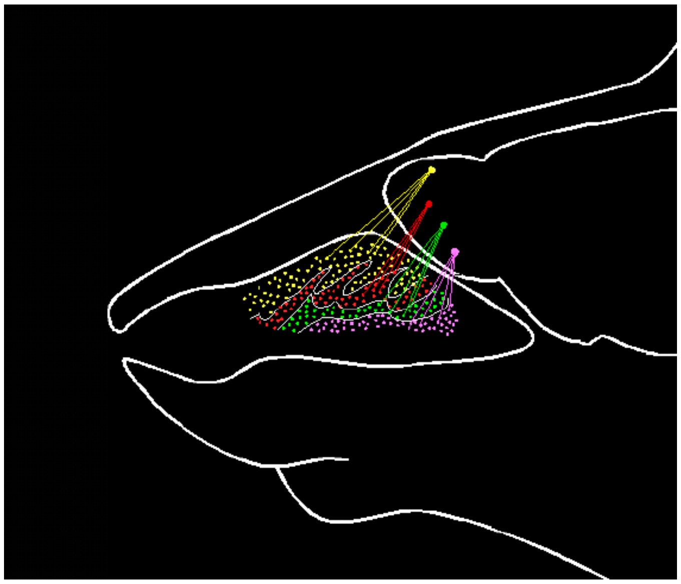


Figure 2.6: Convergence of signals from the OE to the OB. Adapted from [37].

Therefore, in the nose the code for an odor is a dispersed ensemble of neurons, each expressing one olfactory receptor. In the bulb, the code is a specific combination of glomeruli and have a similar spatial arrangement among different individuals. This is illustrated with two examples in Figure 2.7.

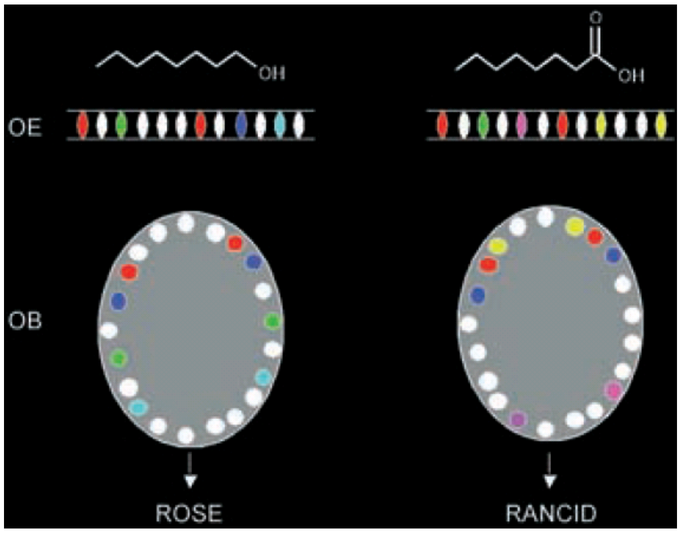


Figure 2.7: Odor coding in the OE and the OB. Adapted from [37].

2.1.3 Dimensionality Exanpsion in the Olfactory Cortex

What happens to this information at higher levels of the brain to ultimately generate odor perceptions?

Mitral cells in the bulb relays the signals from glomeruli to the olfactory cortex. The olfactory cortex is composed of a number of distinct anatomical areas. The largest area is called the piriform cortex (PC). In the 80's, it was shown that a tracer placed in one small region of this cortex would back-label mitral cells in many parts of the bulb [43]. This indicates that the organization of sensory information in the olfactory bulb could not be recapitualted in the cortex. How the information is organized in the olfactory cortex was a mystery.

By developing a genetic method to trace neural circuits, Buck's experiments found that signals from one type of receptor in the bulb project to multiple areas in the olfactory cortex, with a divergence ratio about 1:20 to 1:30, as shown in Figure 2.8.

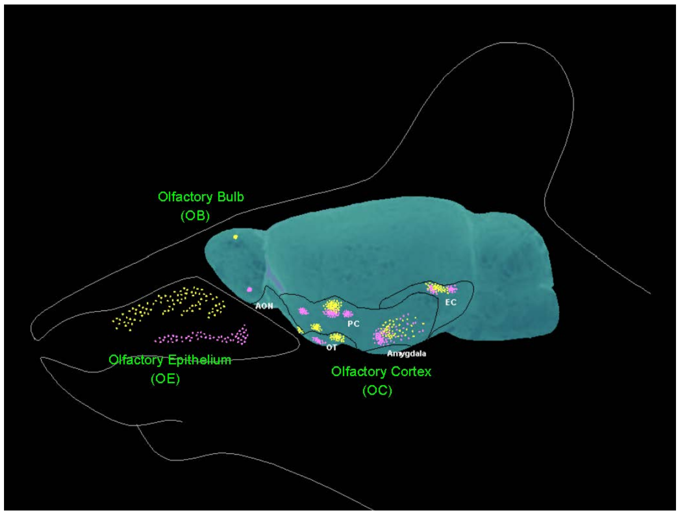


Figure 2.8: Stereotyped map in the olfactory cortex. Signals of the same type of olfactory receptor (marked with the same color) project to multiple spots in the cortex. Adapted from [37].

This map in the cortex is markedly different: Signals from different receptors are spatially segregated in different glomeruli in the OE and OB. In contrast, different receptor signals overlap extensively in the OC and single cortical neuron receives signals from a combination of glomeruli expressing different receptors. Moreover, it was shown that in rat primary olfactory cortex, odor-evoked firing activity is sparse and distributed across the cortical neuron population [44]. Therefore, after the convergence of signals from OE to OB, the combinatorial code in OB is transformed to an overcomplete and sparse representation in the OC. Figure 2.9 illustrates how signals are arranged in these three stages.

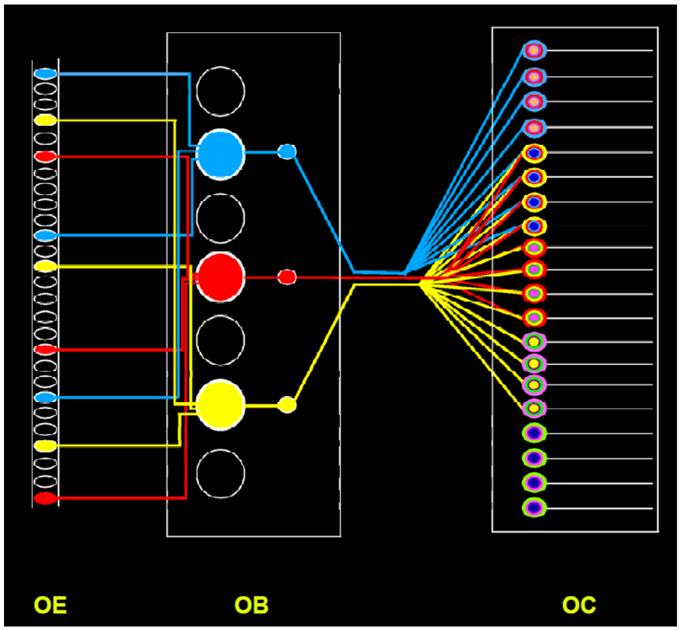


Figure 2.9: Signals of different receptors converge in the OB and then diverge into the OC. Adapted from [37].

Sparse Coding with Overcomplete Basis Functions

What is the significance of this dimensionality expansion? In fact, such dimensionality expansion also exists in the visual system: from the lateral geniculate nucleus (LGN) to the primary visual cortex (V1) [45]. Therefore, the sparse coding theory for V1 simple cells [32], [40], [41] may provide useful insight here. As will be explained shortly, such dimensionality expansion allows the neurons in the cortex to learn the higher-order structure of the inputs and represent the input with as few active neurons as possible.

It's postulated that one important goal of sensory coding is to transform the redundancy (statistical dependence) in the input information to more effective representations for later stages of processing [40]. If we consider a code in terms of an autoencoder network, the design of the code is to find the optimized encoder $\mathbf{W}$ and decoder $\mathbf{M}$ such that information loss is minimized:

$$\min_{\mathbf{W},\mathbf{M}} |\mathbf{x} - \hat{\mathbf{x}}|^2, \tag{2.1}$$

where $\mathbf{x}$ is the input information, and $\hat{\mathbf{x}}$ is the recovered information.

Figure 2.10 shows two coding strategies under two different constraints [46]. According to the information theory, the best code is the one that can preserve the input information with least number of bits. This is the compact code in Figure 2.10(a), where the constraint is imposed on the number of elements representing the information. This can be achieved by principle component analysis (PCA) or singular value decomposition (SVD). Since each output element represents a basis function in an orthonormal set, the input data is decorrelated at the output. It has been noted that Hebbian learning can find the principle components under the right conditions [47]. It was also shown that such redundancy reduction strategy is used in the early visual system [48]. However, the limitation is that it considers only the redundancy due to linear pairwise correlations of the input data and is not sufficient to account for the receptive field of neurons found in the cortex. Such representation also has many drawbacks: First, information can easily be lost due to a failure or malfunction of a neuron. Second, every output neuron has high activity for all inputs. Therefore, this code might be suitable in situations where only limited output dimensionality is available.

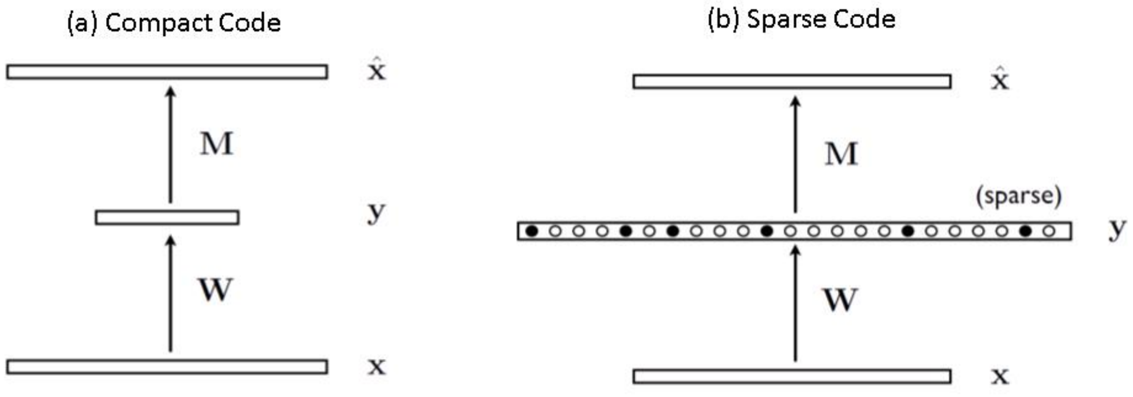


Figure 2.10: Two coding strategies in term of autoencoder networks. Adapted from [46].

The idea of sparse coding (Figure 2.10(b)) is based the assumption that the sensory system is organized to achieve as complete a representation of the input information as possible with minimum number of active neurons [49]. Not only the proportion but also the actual number of active neurons is reduced. Therefore, the constraint of the sparse code is not on the number of total neurons. Rather, it allows a much greater number of neurons to code, and constrains the number of active neurons.

The receptive field of simple cells in mammalian primary visual cortex is characterized as spatially localized, oriented and bandpass, similar to the basis functions in the wavelet transform. Also, the firing of neurons in V1 is sparse and distributed across all neurons. This has led to a conjecture that neurons in the primary visual cortex have adapted themselves to the higher-order structures in the input, such that the response histogram is highly non-Gaussian [40], as shown in Figure 2.11. This is further confirmed by the fact that convolving a Gabor filter with natural images yields highly non-Gaussian responses. And the Gabor function resulting in the highest kurtosis in the response histogram resembles the receptive fields found in V1.

But how do neurons in V1 arrive at such receptive fields from natural images? It was demonstrated that placing only two global objectives is sufficient to account for the principal

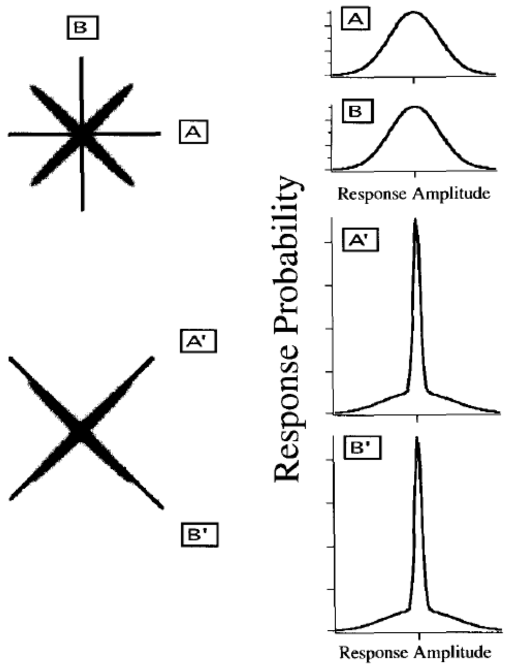


Figure 2.11: Projection pursuit. The left part shows the state space of a two-dimensional data set. The right part shows the distribution of the projection onto the axes. When the axes are aligned with the data structure, the response shows a high degree of kurtosis. Adapted from [40].

spatial properties of simple-cell receptive fields: (1) that information to be preserved, and (2) that the representation be sparse [32]. Figure 2.12 shows the resulting basis functions learned from natural images. It can be seen that except for some low-frequency basis functions, most of them are localized, oriented and bandpass.

As for why using an overcomplete basis - i.e. the number of code elements (basis functions) is greater than the effective dimensionality of the input space, it was shown that sparsification with an overcomplete basis will make each unit more selective to stimulus properties [41] and create more diversities of features in the basis. Several theories had proposed that natural images actually lie along a low-dimensional manifold embedded in a N-dimensional hyperspace [50], [51]. Therefore, it is suggested that the advantage of overcomplete coding is that the manifold becomes flattened out in the space defined by overcomplete basis, making it easier for higher cortical areas to learn structures in the data (i.e., the shape of the manifold) [52].

The other reason for overcomplete representations is that the higher-level cortices are hypothesized to compute with high-dimensional vectors [31]. As will be described in Chapter 4, the brain might have leveraged some subtle properties in the hyper-space such that its computation is highly robust to component variations and failures.

Not only in mammalian visual cortex, there are evidences of sparse coding in other sys-

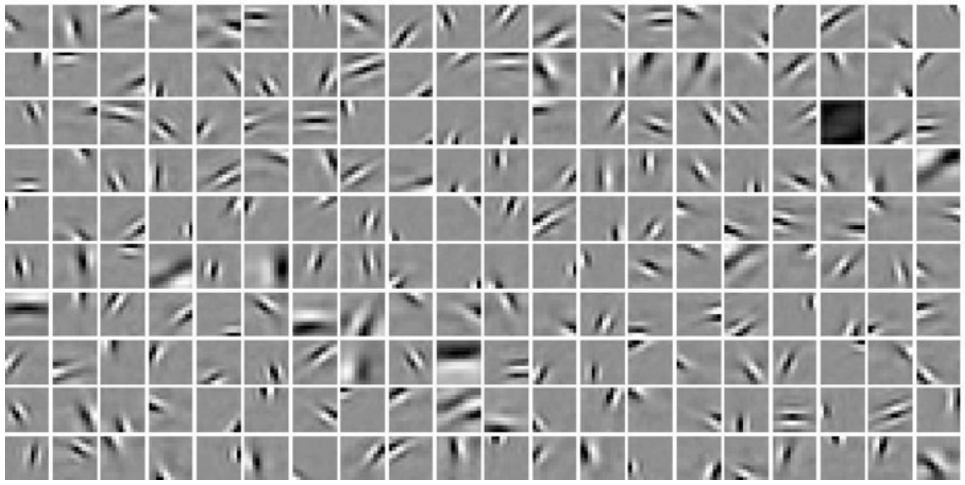


Figure 2.12: Resulting basis functions of sparse coding. Adapted from [32].

tems and species, such as in the mushroom body of locusts [53], auditory cortex in mice [54], hippocampus in rats and primate [55], etc. Although there is no direct evidence for sparse coding in mammalian olfactory cortex, this still provides useful insight for the design of a bio-inspired system.

2.1.4 Computations in Higher-Level Cortices

The computations in higher-level areas are still poorly understood. However, several important biologically plausible models have demonstrated alternative computing paradigms intertwining computation and memory, for example, the Hopfield network [56], the Sparse Distributed Memory for the the cerebellum [30], self-organizing maps, and Hierarchical Temporal Memory (HTM) [57], etc.

Recent years, more and more research efforts are devoted to probabilistic models, inference, and recurrent networks, such as Boltzman machines, deep learning, long-short term memory networks, etc. Some of them have been proved powerful in analyzing the big data. We are looking forward to the future advancement in computational neuroscience that may revolutionize the human information technology.

2.2 Architecture Overview

In light of the biological olfactory pathway, an architecture for artificial olfaction is proposed, as shown in Figure 2.13.

The analytes are first sensed by N gas sensors. Because most gas sensors are broadly responsive, an array of different sensor types must be used to produce unique signatures for different analytes. This is similar to the combinatorial coding in the olfactory epithelium and the olfactory bulb. The sensor output vector is denoted by $\mathbf{s}$ and has a dimension of N.

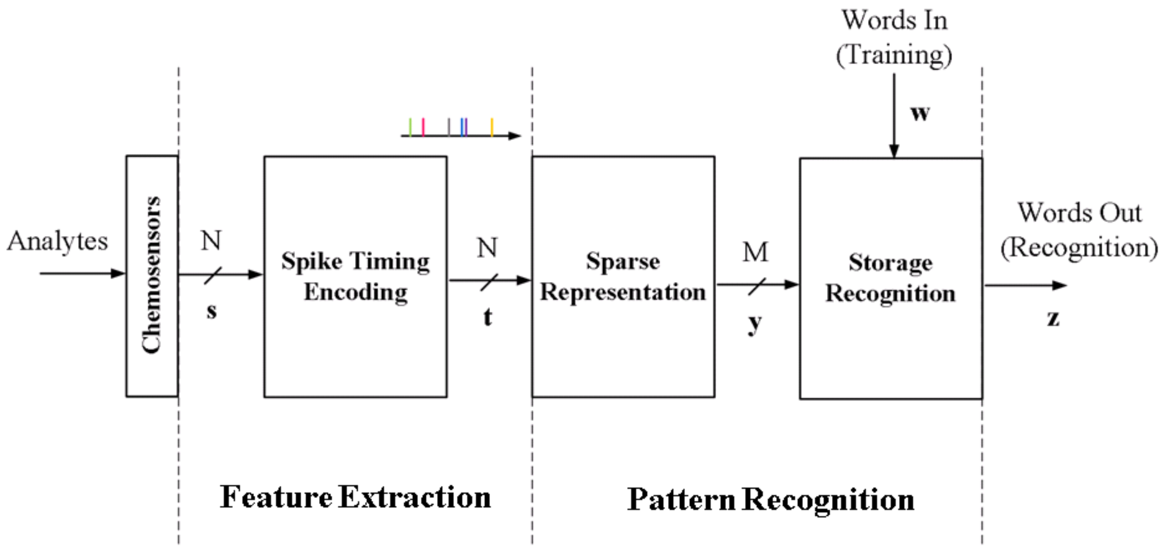


Figure 2.13: System architecture.

The first stage of processing is feature extraction. Instead of immediately digitizing the inputs for digital signal processing, analog primitives are used here to achieve high efficiency. As will be shown in Section 2.3, increasing the degrees of freedoms (sensors) allows analog circuits to operate in the low-power and low-precision regime while achieving the same output precision. This analog front end implements a novel trainable feature extraction algorithm that encodes the analyte identities into spike patterns while decoupling the concentration information to spike delays. It also has a learning mechanism to adapt its parameters to the input statistics. The output spike pattern is denoted as a N-dimensional vector $\mathbf{t}$. Chapter 3 describes the detail of this feature extraction front end and the chip testing results.

The second part is a spike pattern classifier, consisting of a transformation of the input spikes into sparse vectors and a cortical memory model. The transformation is based on a random sampling scheme that can be efficiently performed with circuits exhibiting large parametric variations. This allows the use of minimum-sized devices and operation in the low-voltage regime. As mentioned, sparse representations allow fast and robust pattern storage and retrieval with associative memories, such as the correlation matrix memory used here. Through an overcomplete basis, the spike patterns are transformed into a sparse vector $\mathbf{y}$ with dimension $M \gg N$. During the training phase, the correlation matrix memory takes the pattern $\mathbf{y}$ and its associated word $\mathbf{w}$ as inputs. During the recognition phase, the memory outputs the word $\mathbf{z}$ that is associated with the input pattern. Both $\mathbf{w}$ and $\mathbf{z}$ have an arbitrary dimension U. The algorithms and how it can be implemented in an 3D CNTFET-RRAM technology are described in Chapter 4.

2.3 An Information-Theoretic Framework for Joint Architectural and Circuit Level Optimization

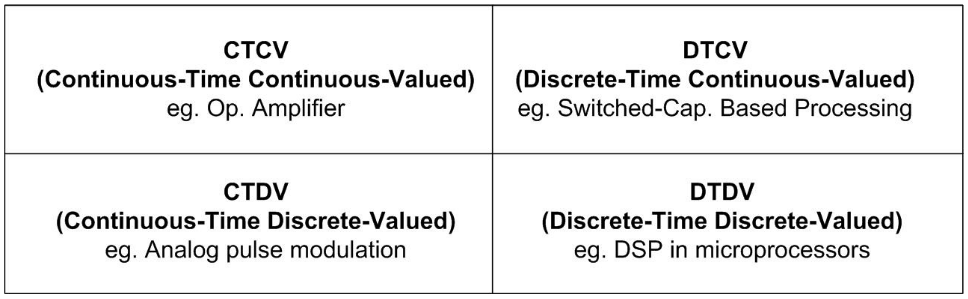
In exploring new computing paradigms, it is essential to have a certain comparison framework for evaluation. In this section, an information-theoretic framework is developed to compare signal representations and to systematically optimize the architectures.

In the sensory processing on biosensors, the nature of the input signal is analog. For processing tasks such as feature extraction or pattern recognition, a precise restitution of the information is not required. In most systems, the input signal is immediately digitized and all the processing is performed in the digital domain. However, it has long been recognized and demonstrated experimentally that low precision analog may be a more efficient way when certain extent of distortion is allowed [58, 59, 60, 61]. Hence a first fundamental question is: What is the most efficient signal representation for a given processing task? That is, when and where is it advantageous to use analog, digital, or other types of signal representations?

It is known that nervous systems do not behave like digital computers. Instead, the information of neural signals is encoded in the time domain without explicit quantization. Can such signaling yield better energy efficiency on silicon? In Section 2.3.1, we revisit the information-theoretic framework set forth by [58], [60] to compare the efficiency of different circuit-level signal representations. In this framework, the signal representation system is viewed as a noisy channel which transfers the information. The criterion to evaluate the goodness of a certain signal representation is the energy per transmitted information bit. We use this framework to investigate the efficiency of one particular continours-time discrete-valued (CTDV) representation and compare it to the digital approach. The results suggest its potential in the low precision regime, similar to the properties of continuous-time continuous-valued (CTCV) and discrete-time continuous-valued (DTCV) representations [58], [60]. In Section 2.3.2, we first present a problem of distinguishing fixed number of analytes using multiple sensors. Then a joint architectural and circuit level optimization is demonstrated on this problem to determine the optimum number of sensors.

2.3.1 Circuit-Level Comparison

Four types of signal representations have been identified [60], [61] as shown in Figure 2.14. In [60], it is shown that on-chip communication links based on CTCV and DTCV representations yield similar behaviors as compared to the discrete-time discrete-valued (DTDV) one: in the low-precision regime, the CTCV and DTCV systems are more efficient than the DTDV system; whereas in the high-precision regime, DTDV system is more efficient. As for the CTDV system, where information is encoded in the time domain rather in the amplitude, it is still less understood and is believed to be the way neurons encode information. Could it be an efficient scheme for on-chip signaling? In the following, we investigate its performance through a on-chip communication link model and evaluate it under a comparison framework.



CTCV (Continuous-Time Continuous-Valued) eg. Op. Amplifier	**DTCV (Discrete-Time Continuous-Valued)** eg. Switched-Cap. Based Processing
CTDV (Continuous-Time Discrete-Valued) eg. Analog pulse modulation	**DTDV (Discrete-Time Discrete-Valued)** eg. DSP in microprocessors

Figure 2.14: Four types of signal representations.

The Methodology

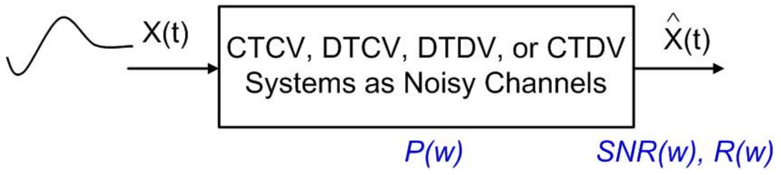
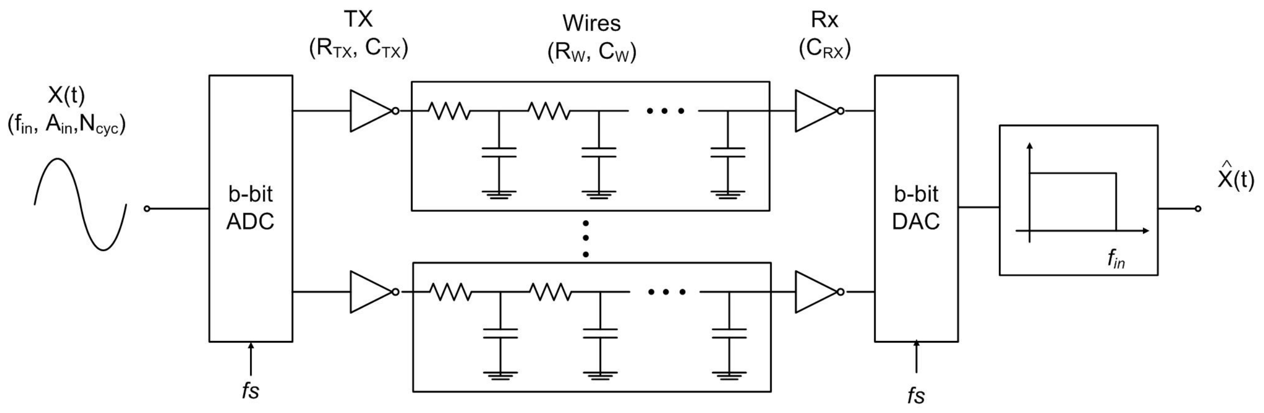


Figure 2.15: Different signaling systems as noisy channels.

First, on-chip communication links are treated as noisy channels [58], [60], [62]. To compare different signal representations, the same analog input $X(t)$ is fed into different systems as shown in Fig 2.15. Then the signal will be represented in different forms $\hat{X}(t)$. Secondly, the power dissipation P is related to different output SNR's (as a measure of the signal fidelity) by varying the design parameter w (eg. number of bits in the DTDV system or the mean firing rate in the CTDV system). The output information rate R is approximated with the rate-distortion function of Gaussian-distributed signals [63], given by $R = f_s log_2 SNR$ where f_s is the signal bandwidth.[1] Finally, the energy per information bit E_b is obtained by dividing the power dissipation in each system by the delivered information rate, i.e. $E_b = P/R$.

The methodology here is different from that in [60] in the evaluation of the power dissipation and the information rate. Here we evaluate the power dissipated in converting the input signal into different signal representations as well as the power dissipated on the interconnect, rather than the input power required to drive the interconnect. This is because different signaling schemes should be compared under the same input power. Second, the information rate is obtained with the rate-distortion function instead of the channel capacity. Although both have similar forms, they differ in the bandwidth used in the formulas and

[1]For signals that are not Gaussian-distributed, the rate-distortion function for the particular input amplitude distribution should be used. We use this approximation because of its simple and explicit relationship with the SNR. Although the absolute bit energies may be deviated, the relative bit energies between the DTDV and CTDV systems are still valid.



Input Signal	
Signal Frequency, fin	16 MHz
Signal Amplitude, Ain	0.5 V
Number of Cycles, Ncyc	51
A/D, D/A	
Number of Bits, b	3-16
Sampling Frequency, fs	200 MHz

Interconnects		
Wire 1mm/0.1um	Wire Res., Rw	400 Ohm
	Wire Cap., Cw	20 fF
Tx	Output Res., Rtx	120 Ohm
	Output Cap., Ctx	24 fF
Rx	Input Cap., Crx	20 fF

Noise Sources	
Clock Jitter, σjitter	33 ps
Voltage Noise on Clock, σv_clk	100 mV
Voltage Noises (PS,XT), σv	100 mV
DAC Noises, σv_dac	1 mV

Figure 2.16: Modeling of a on-chip digital communication link and the parameters in the model. The interconnect parameters are based on UMC's 90nm CMOS Technology.

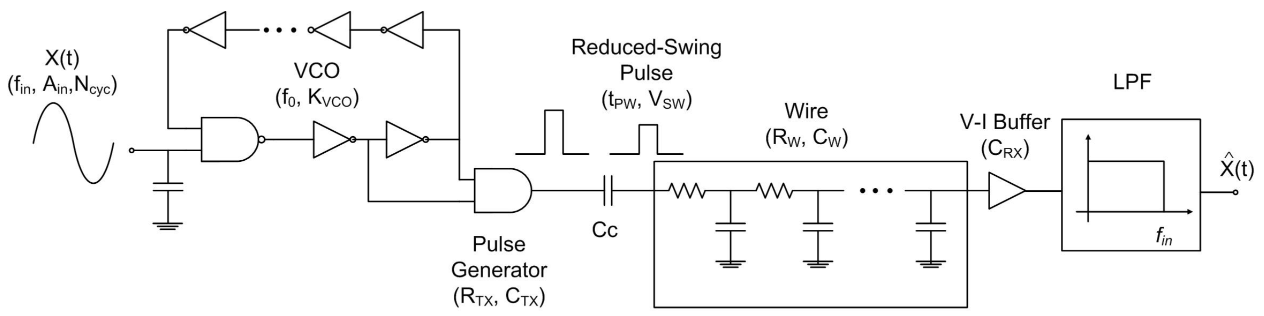
their meanings. The bandwidth in the channel capacity formula is the channel bandwidth, which is in the order of the GHz; whereas that of the signal can be ten to hundred times smaller.

Modeling On-Chip Communication Links

DTDV System The model shown in Figure 2.16 represents a on-chip digital link where an analog signal is sampled by a b-bit A/D converter, transmitted over b parallel wires and recovered by a b-bit DAC followed by a low-pass filter. The input signal used here is a sinusoid with a certain duration. The tables show the parameters used for simulations in *Mathwork Simulink*. They include the input signal, the A/D and D/A converters, the interconnect, and several possible noise sources affecting the converters and the interconnects. The precision of the signal can be increased by increasing b. The total energy consumption on the interconnect is obtained by measuring the total number of transitions on the interconnect. [2]

CTDV System We investigate a CTDV representation that encodes information in the pulse rate [64]. The pulse-rate modulated signal can be generated by driving with the input

[2]We ignored the power in A/D and D/A converters because we assume their power is a small fraction of the total power in a large digital system.



VCO	
Mean Frequency, f0	250-800 MHz
Sensitivity, Kvco	f0 Hz/V

Pulse	
Pulse Width, tpw	200 ps
Pulse height, Vsw	0.3-0.5 V
Coupling Cap., Cc	17-40 fF

Noise Sources	
VCO Jitter, σv1	33 ps
Voltage Noises (PS, XT), σv2	100 mV

Figure 2.17: Modeling of a pulse-rate modulated communication link and the parameters in the model.

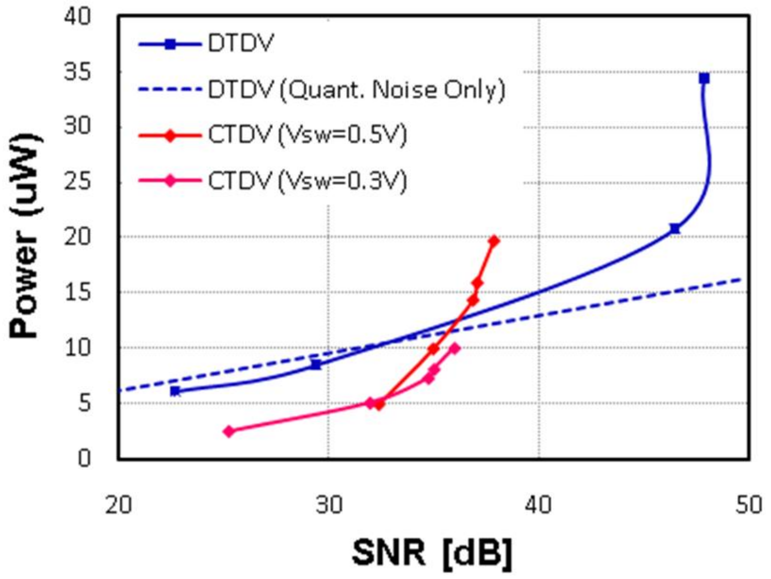
signal a VCO followed by a pulse generator, as shown in Figure 2.17. The pulses propagate through the same interconnect, and the signal is recovered by a low-pass filter. The pulse swing can be reduced by using a coupling capacitor C_c without significant degradation of the output SNR. The higher the mean frequency f_0 of the VCO, the higher the output SNR. Similarly, the total energy consumption is obtained by measuring the total number of transitions.

E$_b$ VS. SNR

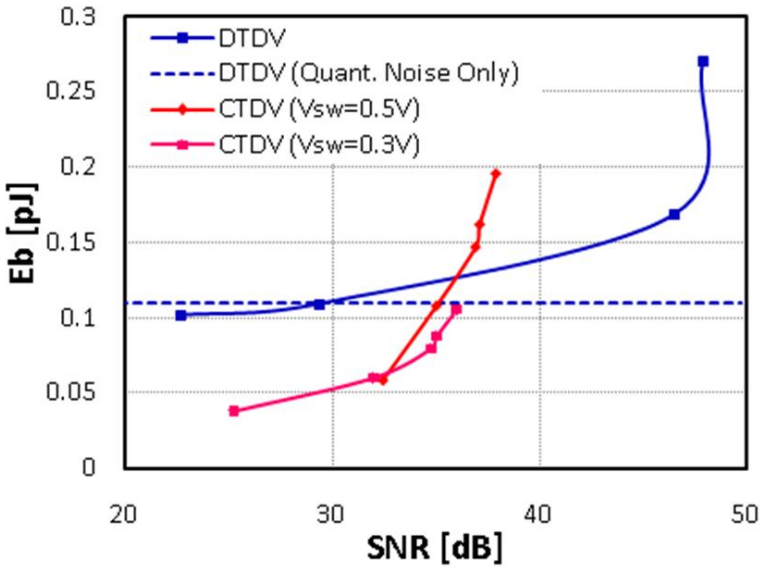
The simulation results are shown in Figure 2.18. The number of bits in DTDV system ranges from 3-16. The mean firing rate of the CTDV system ranges from 250-800 MHz. Figure 2.18a plots the power versus SNR, which is then divided by the effective information rate to obtain the bit energy versus SNR as shown in Figure 2.18b.

We can see that the pulse rate-modulated signal is more efficient in the low precision regime, while the digital signaling is more efficient in the high precision regime. This property similar to those of CTCV and DTCV representations [60, 61] suggests that CTDV representation may actually fall into the larger category of 'analog' representations. We suspect other analog pulse modulations also exhibit similar properties.

As a final note, even though the digital signaling is more efficient in high precision, its performance is still limited: the dotted blue line in Figure 2.18 is the ideal digital signaling where quantization noise is the only noise source. It can be seen that ideally the digital signaling should have constant E_b while in reality its precision is limited by other noise sources. Therefore, even with digital signaling, the cost to operate at the high-precision regime is still high.



(a) the power versus SNR



(b) the E_b versus SNR.

Figure 2.18: Simulation results for the DTDV and CTDV systems..

2.3.2 Joint Architectural and Circuit Level Optimization

The analysis above indicates the possible energy savings operating in the low-precision regime. To take advantage of this opportunity requires signal processing architectures that operate effectively in this regime. Feature extraction or pattern recognition are processing tasks that best fit this opportunity: Instead of precisely restituting the input information, only the useful information (e.g. the identity) needs to be extracted at the output. Therefore, the output precision can be just as much as needed to represent the useful information.

In this exploration, we choose a problem of distinguishing a fixed number of gas analytes using multiple sensors. Since the amount of output information that needs to be represented is fixed, the required precision per degree of freedom decreases as number of sensors increases. Therefore, the circuits can afford to be noisier and consume less power. In conjunction with the power-precision model of the computation circuits, the optimum number of sensors can be derived.

Concentration-Invariant Encoding

Here we consider using multiple carbon black/polymer composite sensors to detect gas analytes. Carbon black/polymer composite sensors are characterized as having simple linear responses (fractional resistance change $\Delta R/R$) with respect to gas concentration [65]. Similar to most gas sensors, each sensor responds to multiple analytes. Suppose there are N sensors, then the N-dimensional vector response of the sensor array for analyte k, $\mathbf{s}_k$, can be expressed as

$$\mathbf{s}_k = \mathbf{a}_k \cdot c \tag{2.2}$$

where $\mathbf{a}_k$ is the signature vector associated with each analyte and c is the gas concentration.

Figure 2.19 illustrates the responses for K analytes in an N-dimensional response space. Red and blue dots represent responses at two concentration levels c_H and c_L, respectively. The goal of the feature extraction task is to extract the signature vector $\mathbf{a}_k$ from the sensor response $\mathbf{s}_k$. A simple logarithmic transform can do so by separating the identity information from the concentration information:

$$log\mathbf{s}_k = log\mathbf{a}_k + logc \tag{2.3}$$

If $log\mathbf{s}_k$ can be linearly transformed to the delays of spikes $\mathbf{t}_k$, then the signature vector would be transformed into a unique spike pattern while the concentration information would appear as a delay in time, as shown in the right of Figure 2.19. These spikes patterns will then be used as features for subsequent storage or classification.

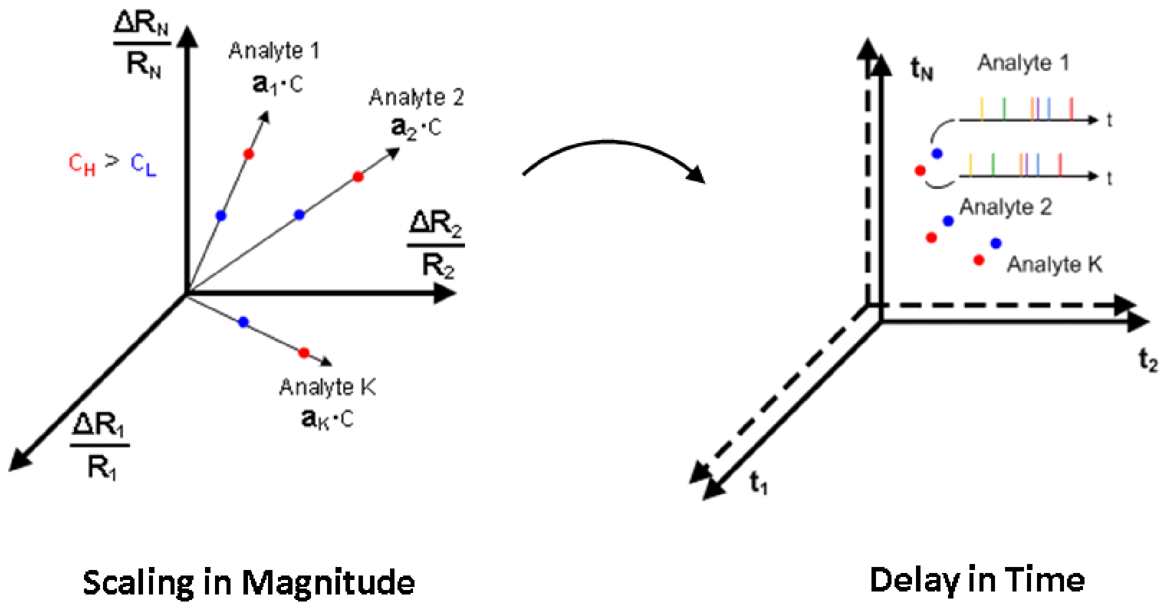


Figure 2.19: Concentration invariant encoding.

Minimum Required Precision Per Degree of Freedom

If the number of total interested analytes K are fixed, and the goal is to distinguish each analyte for certain amounts of concentration levels, the required precision per degree of freedom would decreases as the number of sensors (degree of freedoms) increases.

Figure 2.20 shows the simulated sensor responses after logarithmic transform using N=2 and N=3 sensors. The sensors are modeled with parameters from the measurements in [65]. There are total $K = 8$ analytes in this simulation. As the number of sensors N increases, the distance between analyte responses gets larger. Therefore, the minimum required precision of the logarithmic transform in each channel can be lower.

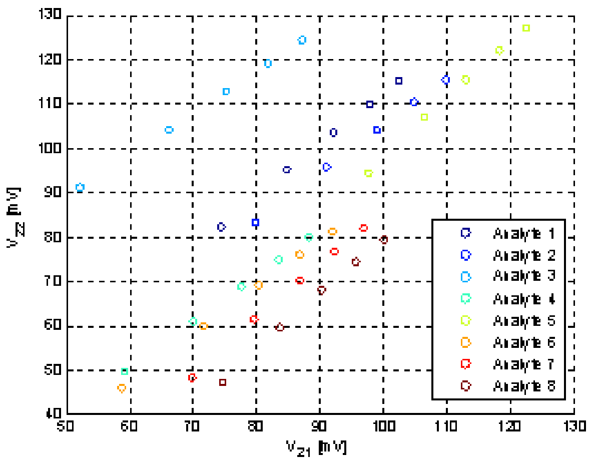
Figure 2.21 shows the minimum required precision of the logarithmic transform per channel as N increases from 2 to 7. The precision is defined as

$$log_2 \frac{V_{range}}{\sqrt[N]{d_{min}}} \text{ [bits]} \tag{2.4}$$

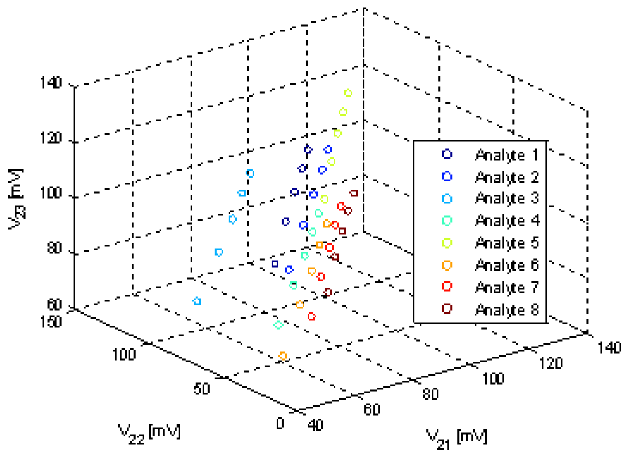
where V_{range} is the output voltage range of logarithmically transformed sensor response, and d_{min} is the minimum distance between any two dots in the log response space, calculated by the program.

Power-Precision Trade-off

In this exploration, we consider an analog logarithmic converter as shown in Figure 2.22a. In order to compare with a digital implementation, a digital lookup table for logarithmic



(a) N=2



(b) N=3

Figure 2.20: Analyte responses in the log space with different number of sensors (N).

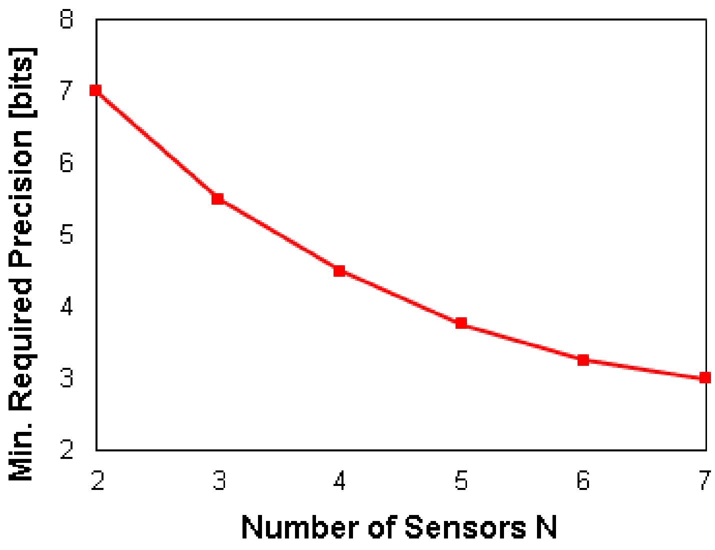
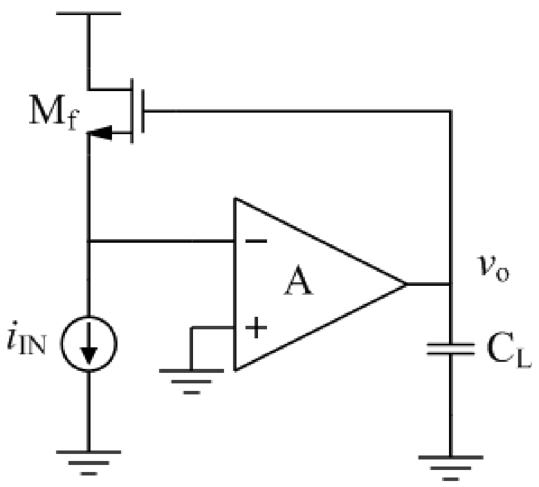
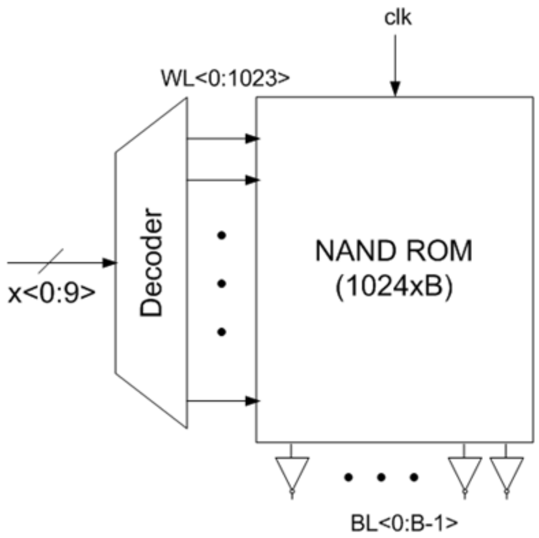


Figure 2.21: Minimum required precision per degree of freedom.

transform shown in Figure 2.22b is also considered.



(a) Analog logarithmic converter



(b) Digital lookup table (LUT)

Figure 2.22: Logarithmic transform circuits.

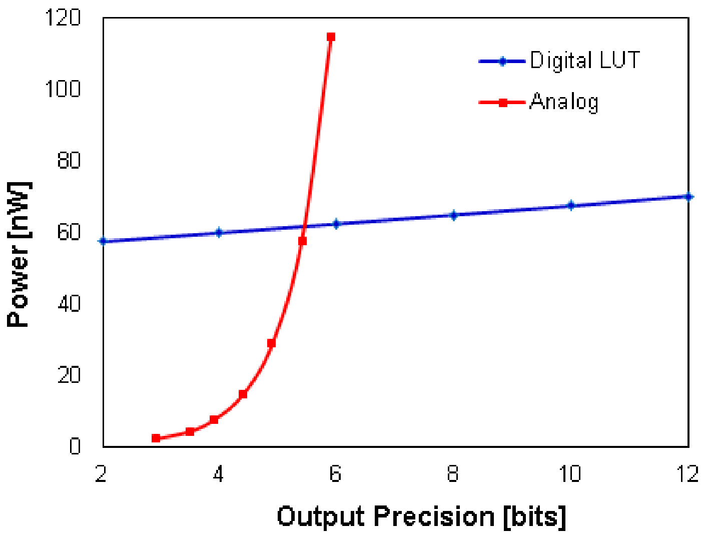


Figure 2.23: The power-precision trade-offs.

The power-precision trade-off in the analog logarithmic converter mainly comes from sizing up the transistors to lower the threshold variations while keeping the closed-loop bandwidth constant. The trade-off in the digital logarithmic lookup table simply comes from the adding more columns as output precision increases.

Both designs are supplied at 0.5 V and simulated with models from ST Microelectronics' 65 nm Low Power (LP) technology. The details of the simulation and power estimation are described in Appendix A. Figure A.8 shows the resulting power-precision profile of the analog and digital designs. It can be seen that analog implementation has a notable advantage in the low-moderate precision regime.

Optimization Methodology

Combining the required precision per degree of freedom (Figure 2.21) and the power-precision profile (Figure A.8), the required power per degree of freedom $P(N)$ as a function of number of sensors N can be obtained, as shown in Figure 2.24.

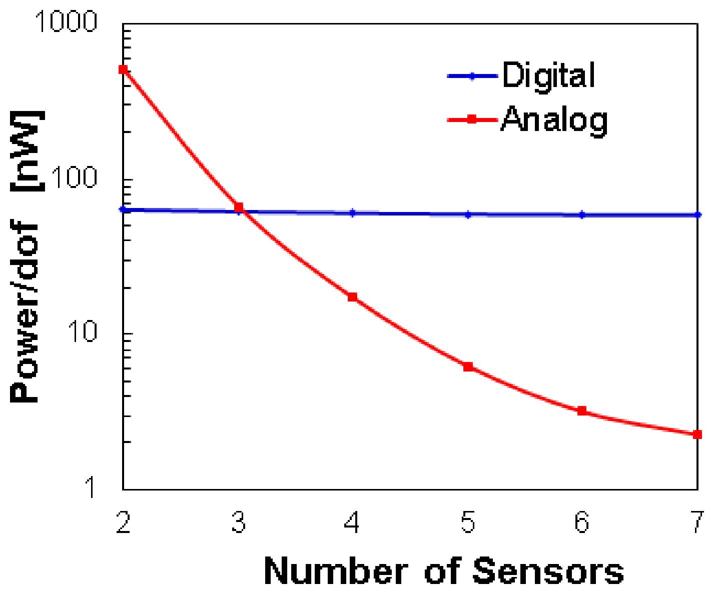


Figure 2.24: The required power per degree of freedom $P(N)$.

Therefore, the optimal number of sensors can be determined by the following objective:

$$\min_{N} N \cdot P(N) \tag{2.5}$$

Plotting $N \cdot P(N)$ versus N in Figure 2.25, we can see that increasing number of sensors benefits analog implementation. But the decrease of analog power gradually slows down around 6 to 7 sensors.

To summarize, we have demonstrated a joint architectural and circuit level optimization method to determine the optimum number of sensors for an analog design. The power savings with respect to a digital approach is also investigated. Although a specific feature extraction using logarithmic transform is analyzed here, the comparison framework and the optimization methodology can be applied in other contexts.

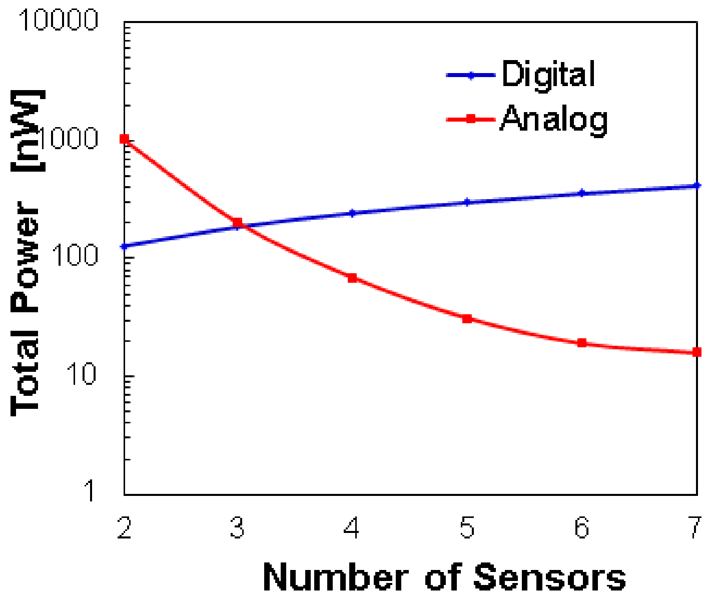


Figure 2.25: Total power consumption $N \cdot P(N)$.

Chapter 3

An Analog Gas Sensing Front End

As mentioned in previous two chapters, the biological systems interact with the analog world seamlessly. The useful information is extracted by extremely efficient analog signal processing in the early sensory pathways [34]. It is suggested that the neural systems have used the physics in analog computation to achieve this. In this chapter, we propose an analog signal processing algorithm for gas sensor arrays. This algorithm can be efficiently implemented in subthreshold analog circuits. Moreover, by leveraging the degree of freedom in multi-channel processing, the analog circuits are designed in the low-power and low-precision regime while achieving the desired output precision.

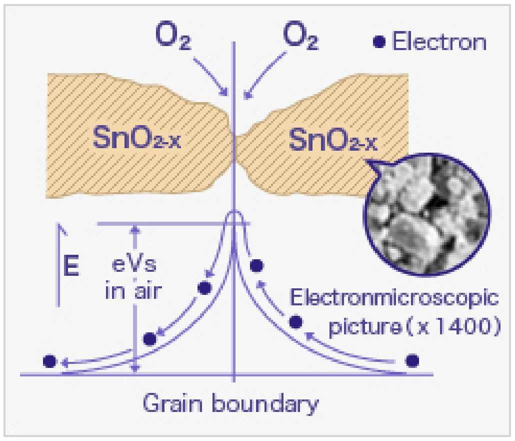
Among many gas sensing technologies, metal-oxide gas sensors are the most available one in the market. Due to its high-temperature operation, it used to consume large power on the heater. However, recent advances in integrating the sensor with MEMS micro-hotplate has significantly reduced the power and size [66], [67], [68], [69], making them more amenable for mobile or portable applications. Polymer-based gas sensors, such as those mentioned in Section 2.3 are very attractive, as they operate in room temperature, have simple linear response relationship with gas concentration, and can offer great number of varieties. However, they are extremely sensitive to environmental factors (e.g. humidity, temperature) and are still hard to be commercialized. Other gas sensors such as electrochemical and optical sensors can be made selective, but they suffer from limited life time and requiring large gas cells, respectively. Therefore, in this research, we chose the common metal-oxide sensors to work with, and developed a dedicated algorithm for an array of metal-oxide sensors. To the best of our knowledge, this is the first integrated trainable feature extraction front end that can simultaneously process parallel gas sensor signals.

In the first section this chapter, the sensing mechanism and response property of metal-oxide sensors are introduced. Next, the feature extraction and learning algorithms for an array of metal-oxide sensors are proposed. These algorithms are implemented with subthreshold analog circuits in CMOS technology. Finally, the testing results are described.

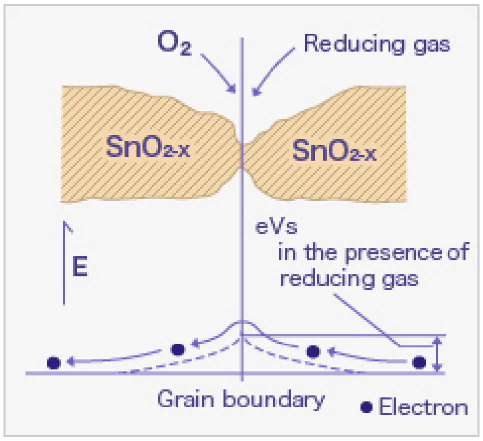
3.1 Metal-Oxide Gas Sensors

3.1.1 Sensing Mechanism

Metal oxides such as SnO_2, ZnO, Fe_2O_3, and WO_2 are intrisically n-type semiconductors. At temperatures of 200 to 500 °C, they respond to reducing gases such as H_2, CH_4, CO, C_2H_5, or H_2S by decreasing the resistance.



(a) In clean air



(b) In the presence of reducing gases

Figure 3.1: Potential barriers at the grain boundaries in metal-oxide semiconductors. Credit: FIGARO, Inc.

Figure 3.1 illustrates the sensing mechanism. In clean air (approximately 21% O_2), oxygen is adsorbed on the metal oxide surface. With its high electron affinity, adsorbed oxygen attracts free electrons inside the metal oxide, forming a potential barrier at the grain boundaries. This potential barrier prevents electron flow, causing high sensor resistance in clean air. When the sensor is exposed to combustible or reducing gases, the oxidation reaction of such gas with adsorbed oxygen occurs at the surface of metal oxide. As a result, the density of adsorbed oxygen on the metal oxide surface decreases, and the height of the potential barrier is reduced. Because more electrons can flow through the potential barrier of reduced height, the sensor resistance decreases. The reactions can be expressed as follows [70].

$$e + \frac{1}{2}O_2 \rightarrow O^- \tag{3.1}$$

$$R_{(g)} + O^- \rightarrow RO_{(g)} + e \tag{3.2}$$

where e is the electron from the conduction band of the oxide semiconductor, and $R_{(g)}$ is the reducing gas.

3.1.2 Sensor Response Characteristics

The most widely used semiconducting material is SnO_2, doped with small amounts of impurities and catalytic metal additives. By changing the choice of impurities and catalysts, many types of gas sensors can be developed. Typically, metal-oxide gas sensors exhibit relatively poor selectivity for gases and respond to many gases. Moreover, the response follows a power law within a certain range of gas concentration [71]. These can be seen in a typical response profile as shown in Figure 3.2 [72], where R_0 represents the sensor resistance in a reference condition, and R_S represents the sensor resistance in the presence of the gas at different concentrations. Therefore, their responses are usually modeled as:

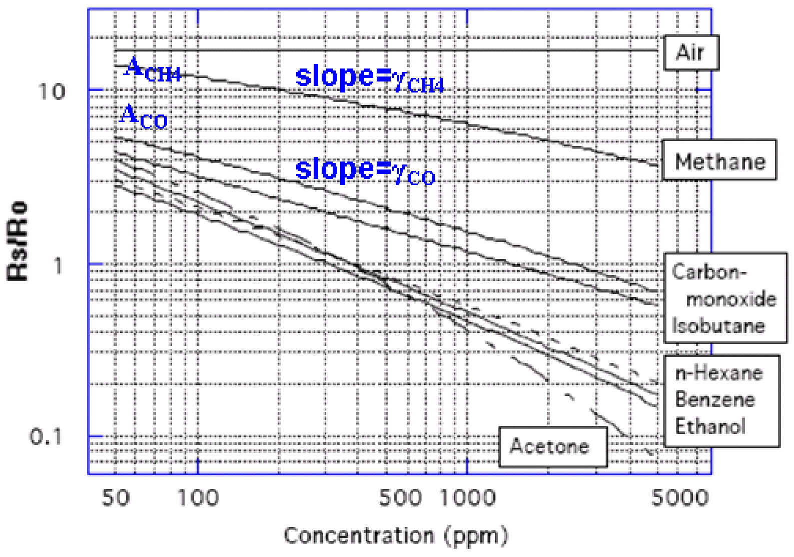


Figure 3.2: Response profile of FIGARO's TGS822 sensor for solvent vapor detection.

$$\frac{R_s}{R_0} = a[c]^{-\gamma} \tag{3.3}$$

The response of a sensor to an analyte is then characterized by a scaling factor a and an exponent γ. For example, for analyte p and analyte q, their responses from a sensor can be expressed as follows.

$$\left(\frac{R_s}{R_0}\right)_p = a_p[c]^{-\gamma_p} \tag{3.4}$$

$$\left(\frac{R_s}{R_0}\right)_q = a_q[c]^{-\gamma_q} \tag{3.5}$$

3.1.3 Arrayed Sensing

Since one sensor is not sufficient to distinguish different analytes, different types of sensors must be used in an array such that the array can produce a unique pattern for each analyte. Then the responses of the sensor array to analyte p and analyte q can be expressed as vector responses:

$$\mathbf{R}_p = \begin{bmatrix} \left(\frac{R_{S1}}{R_{01}}\right)_p \\ \left(\frac{R_{S2}}{R_{02}}\right)_p \\ \vdots \\ \left(\frac{R_{SN}}{R_{0N}}\right)_p \end{bmatrix} = \begin{bmatrix} a_{1p}c^{-\gamma_{1p}} \\ a_{2p}c^{-\gamma_{2p}} \\ \vdots \\ a_{Np}c^{-\gamma_{Np}} \end{bmatrix} \tag{3.6}$$

$$\mathbf{R}_q = \begin{bmatrix} \left(\frac{R_{S1}}{R_{01}}\right)_q \\ \left(\frac{R_{S2}}{R_{02}}\right)_q \\ \vdots \\ \left(\frac{R_{SN}}{R_{0N}}\right)_q \end{bmatrix} = \begin{bmatrix} a_{1q}c^{-\gamma_{1q}} \\ a_{2q}c^{-\gamma_{2q}} \\ \vdots \\ a_{Nq}c^{-\gamma_{Nq}} \end{bmatrix} \tag{3.7}$$

Figure 3.3 shows the response trajectories for five analytes using three gas sensors. We can see that as concentration changes, each analyte streaks a unique trajectory in this three dimensional space.

3.2 Algorithms

As shown in Equations 3.6 and 3.7, each analyte is characterized by a unique scaling vector $\mathbf{a} = [a_1, a_2, \cdots, a_N]$ and an exponent vector $\boldsymbol{\gamma} = [\gamma_1, \gamma_2, \cdots, \gamma_n]$, so the goal of the front end is to extract either the $\mathbf{a}$ vector or the $\boldsymbol{\gamma}$ vector, or even both vectors as the feature for each analyte. Most feature extraction algorithms require computing and storing all features for all analytes during learning, and require multiple times of processing during recognition. Here we propose an efficient feature extraction algorithm that requires computing and storing only one composite feature during learning, and requires only one processing during recognition.

3.2.1 Feature Extraction

The goal of this feature extraction front end is to get a single unique feature for all input vectors corresponding to the same analyte. Therefore, the design of the feature extraction algorithm is aimed to have a certain transformation such that input vectors on the same trajectory are transformed into one representation.

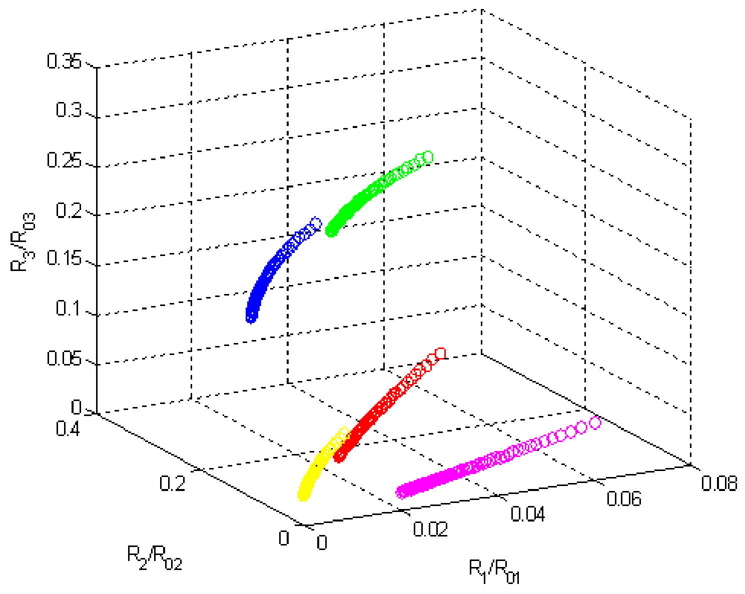


Figure 3.3: Response trajectories in the sensor response space.

First, it is reasonable to think that straight trajectories are easier to process. Since the gas sensor output responses follow a power law (Equation 3.3), the trajectories can be "streghtened" by a logarithmic transform. For example, for analyte p and analyte q, their transformed responses become

$$log\frac{1}{\mathbf{R}_p} = -\mathbf{a}_p + logc \cdot \boldsymbol{\gamma}_p \tag{3.8}$$

$$log\frac{1}{\mathbf{R}_q} = -\mathbf{a}_q + logc \cdot \boldsymbol{\gamma}_q \tag{3.9}$$

Figure 3.4 shows the logarithmically transformed response trajectories from Figure 3.3.

In the $1/log\mathbf{R}$ space, if we can find the vector $\mathbf{s}$ closest to all the extrapolated trajectories, i.e. the virtual intersection as marked in Figure 3.4, then projecting the trajectories onto the unit sphere centered at $\mathbf{s}$ will concentrate each trajectory in a small region on the unit sphere. The operation of projection is essentially a vector normalization:

$$\frac{log\frac{1}{\mathbf{R}} - \mathbf{s}}{||log\frac{1}{\mathbf{R}} - \mathbf{s}||} \tag{3.10}$$

Figure 3.5 shows the projections on the unit sphere.

If we have the freedom to configure the sensor types such that their responses to target analytes virtually intersect at $\mathbf{s}$ in the $1/log\mathbf{R}$ space, then each trajectory will be concentrated

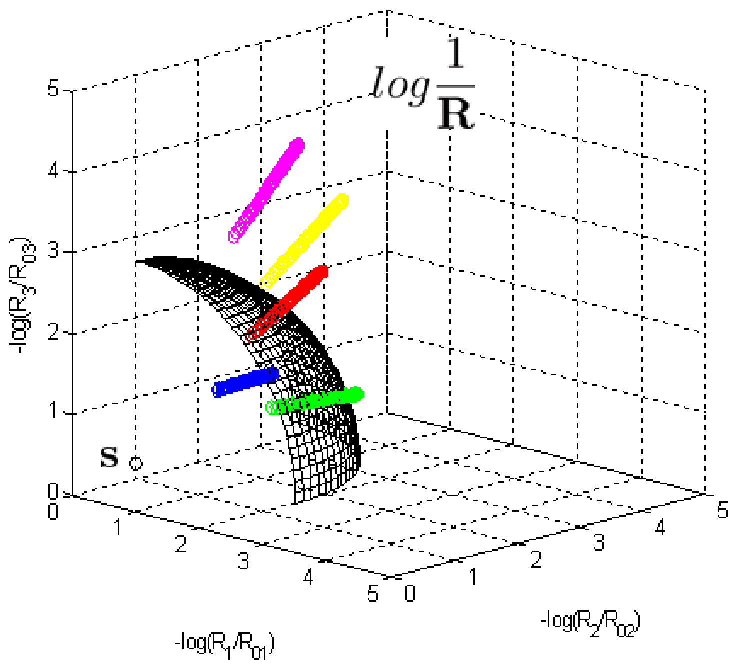


Figure 3.4: Sensor responses in the $1/log\mathbf{R}$ space.

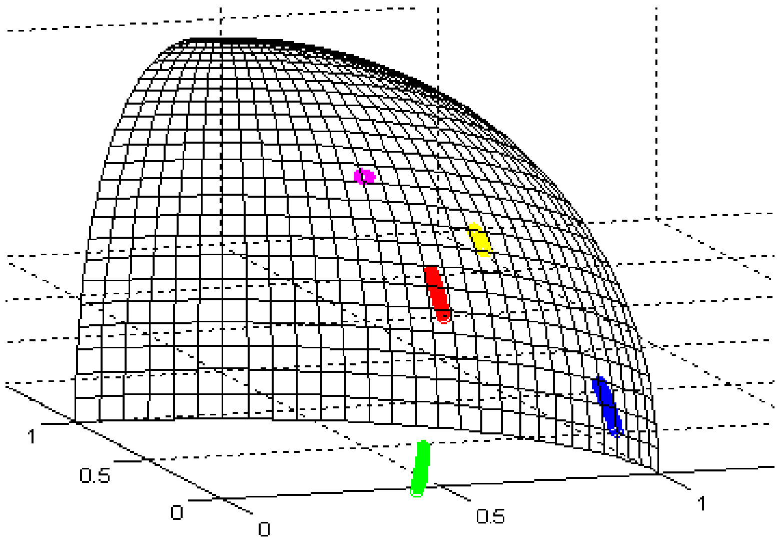


Figure 3.5: Projected trajectories on the unit sphere centered at $\mathbf{s}$.

at one single point, which is then the feature of the corresponding analyte. If not, the concentrated regions can be considered as noisy outputs. The subsequent pattern recognition can still distinguish them if the projected regions stay far away enough from each other. In next section, the learning of the optimum $\mathbf{s}$ from a set of analyte responses will be described. Here we will move on to discuss another way of extracting the feature while preserving the concentration information.

Concentration-Invariant Encoding

One of the key computational problems of olfaction is concentration invariance [73]. It was discovered that odors at different concentrations generate families (low-dimensional manifolds) of spatiotemporal representations [74], providing the neural substrate for concentration invariance. It was also suggested that logarithmic encoding is particularly useful in scale-invariant recognition problems [75], because it makes the relative timing of spikes independent of the intensity or scale.

In light of these, we may do something different than Equation 3.10. First, note that $(1/log\mathbf{R} - \mathbf{s})$ is actually proportional to $\boldsymbol{\gamma}$, assuming that all extrapolated trajectories intersect at $\mathbf{s}$. For example, for analyte p

$$\begin{aligned} log\frac{1}{\mathbf{R}_p} - \mathbf{s} &= -\mathbf{a}_p - \mathbf{s} + logc \cdot \boldsymbol{\gamma}_p & (3.11)\\ &= \lambda_p \cdot \boldsymbol{\gamma}_p + logc \cdot \boldsymbol{\gamma}_p & (3.12)\\ &= (\lambda_p + logc) \cdot \boldsymbol{\gamma}_p & (3.13)\end{aligned}$$

where λ_p is a scaler relating $(-\mathbf{a}_p - \mathbf{s})$ to $\boldsymbol{\gamma}_p$. The feature vector, $\boldsymbol{\gamma}_p$, can then be extracted by another logarithmic transform:

$$log(log\frac{1}{\mathbf{R}_p} - \mathbf{s}) = log\boldsymbol{\gamma}_p + log(\lambda_p + logc) \cdot \mathbf{1} \tag{3.14}$$

Similar to the encoding scheme in Section 2.3, if $log(log(1/\mathbf{R}_p) - \mathbf{s})$ can be converted to the timing of spikes $\mathbf{t}_p$, then the responses of each analyte will be transformed to a unique spike pattern, with its concentration information encoded in the delay of the spikes, as shown in Figure 3.6. This spike pattern is the feature that the subsequent classification unit needs to recognize.

To summarize, the feature extraction algorithm is

$$\mathbf{t} = log(log\frac{1}{\mathbf{R}} - \mathbf{s}) \tag{3.15}$$

3.2.2 Learning

As mentioned in the previous section, how to find $\mathbf{s}$ given a set of analyte responses? In this section, we show how to use the gradient decent algorithm to iteratively update $\mathbf{s}$, such that it will gradually approach the optimum solution each time an analyte is applied.

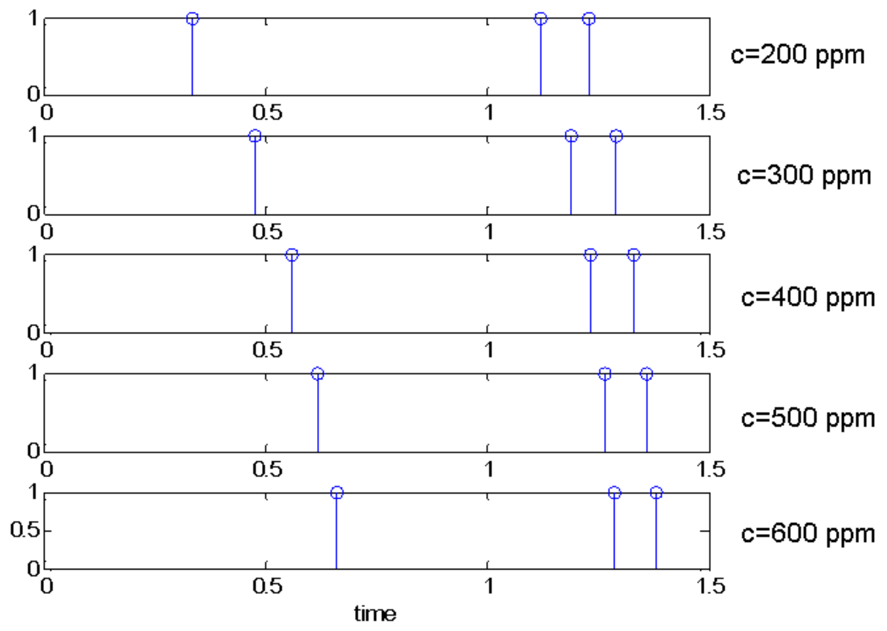


Figure 3.6: Analyte responses at different concentrations are transformed to a unique spike pattern with different delays in time.

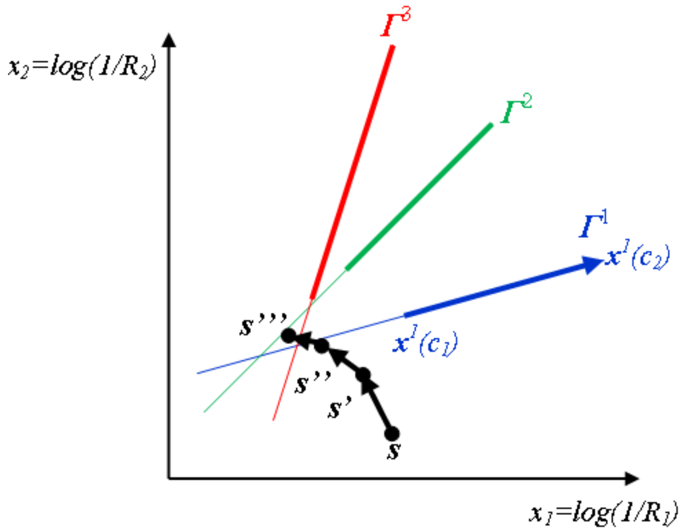


Figure 3.7: Updates of $\mathbf{s}$ in the $\mathbf{x} = log(1/\mathbf{R})$ space.

Let $\mathbf{x} = log(1/\mathbf{R})$. Figure 3.7 shows the response trajectories in the $\mathbf{x}$ space. For simplicity, this figure illustrates the case where the number of sensors is 2. However, this algorithm is for any number of sensors and the following derivation is generalized to N sensors.

During the learning phase, each analyte is applied at two concentration levels c_1 and c_2, yielding two responses $\mathbf{x}^k(c_1)$ and $\mathbf{x}^k(c_2)$, where k denotes the k-th analyte. $\mathbf{\Gamma}^k$ is the signature vector of the k-th analyte.

$$\begin{aligned} \mathbf{\Gamma}^k &= \mathbf{x}^k(c_2) - \mathbf{x}^k(c_1) && (3.16) \\ &= \boldsymbol{\gamma}_k \cdot log\frac{c_2}{c_1} && (3.17) \end{aligned}$$

When $\mathbf{s}$ is at this optimal location, $(\mathbf{x}^k - \mathbf{s})$ should align with $\mathbf{\Gamma}^k$. Therefore, the energy function to be minimized can be defined as the following.

$$E[\mathbf{s}] = \frac{1}{2}\sum_k \|\frac{\mathbf{\Gamma}^k}{\|\mathbf{\Gamma}^k\|} - \frac{\mathbf{x}^k - \mathbf{s}}{\|\mathbf{x}^k - \mathbf{s}\|}\|^2 \qquad (3.18)$$

Accordingly, the update rule is

$$\begin{aligned} \Delta\mathbf{s} &= -\eta\frac{\partial E}{\partial \mathbf{s}} && (3.19) \\ &\approx -\eta\sum_k \left(\frac{\mathbf{\Gamma}^k}{\|\mathbf{\Gamma}^k\|} - \frac{\mathbf{x}^k - \mathbf{s}}{\|\mathbf{x}^k - \mathbf{s}\|}\right) && (3.20) \end{aligned}$$

Since $\mathbf{s}$ is updated sequentially, the update for each trial is simply

$$\Delta\mathbf{s} \approx -\eta\left(\frac{\mathbf{\Gamma}^k}{\|\mathbf{\Gamma}^k\|} - \frac{\mathbf{x}^k - \mathbf{s}}{\|\mathbf{x}^k - \mathbf{s}\|}\right). \qquad (3.21)$$

Figure 3.8 shows that updating $\mathbf{s}$ according to the update rule (i.e. moving in the direction of the difference between the normalized $(\mathbf{x}^k - \mathbf{s})$ vector and the normalized $\mathbf{\Gamma}^k$vector) will lead to the optimum location.

3.3 CMOS Implementation

3.3.1 Architecture Overview

Figure 3.9 shows the gas sensing frontend architecture. It consists of a main signal path implementing the feature extraction algorithm and a learning unit. During the training phase, the learning block is activated to iteratively update the voltage vector $\mathbf{V_S}$, which will then be used in the main signal path to perform the feature extraction algorithm.

The input is a vector of sensor resistance $\mathbf{R_S}$. After a resistance-to-current conversion, the log converter relates the currents log-linearly to the output voltage vector $\mathbf{V_X}$. $\mathbf{V_X}$ is then

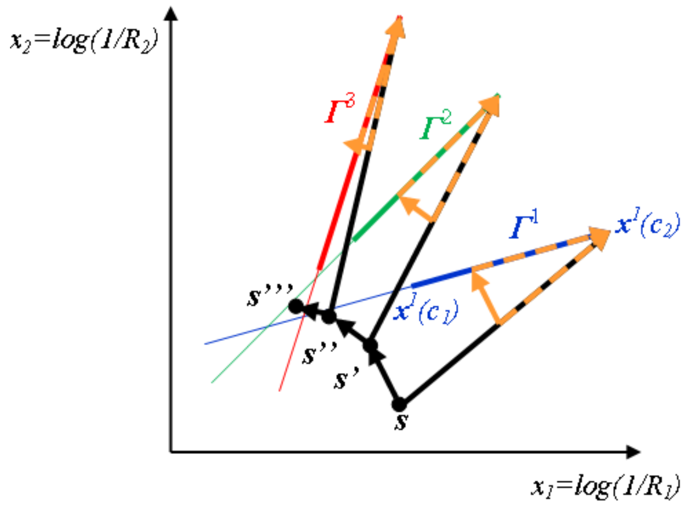
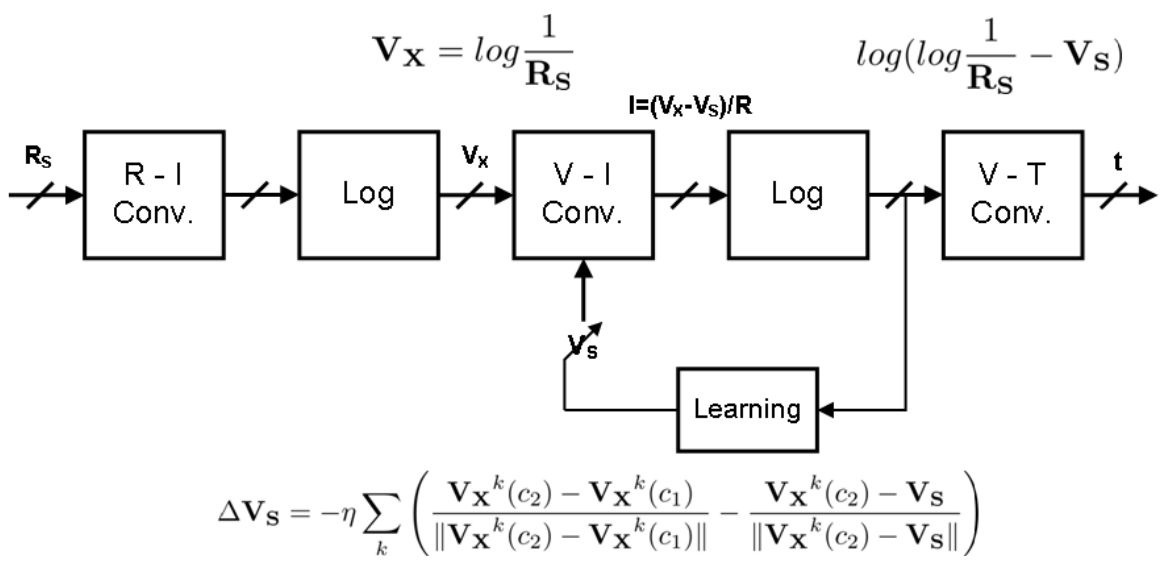


Figure 3.8: Updates of **s** according to the update rule.

subtracted by $\mathbf{V_S}$ in a voltage-to-current converter and undergoes the second logarithmic transform. The voltage-to-spike timing converter linearly relates the input voltage to the output spike delay. Therefore the timing vector of the output spikes **t** is proportional to $log(log\frac{1}{\mathbf{R_S}} - \mathbf{V_S})$.



$\mathbf{V_X} = log\frac{1}{\mathbf{R_S}}$
$log(log\frac{1}{\mathbf{R_S}} - \mathbf{V_S})$
I=(VX-VS)/R
RS
R - I Conv.
Log
VX
V - I Conv.
Log
V - T Conv.
t
VS
Learning
$\Delta\mathbf{V_S} = -\eta\sum_k\left(\frac{\mathbf{V_X}^k(c_2) - \mathbf{V_X}^k(c_1)}{\|\mathbf{V_X}^k(c_2) - \mathbf{V_X}^k(c_1)\|} - \frac{\mathbf{V_X}^k(c_2) - \mathbf{V_S}}{\|\mathbf{V_X}^k(c_2) - \mathbf{V_S}\|}\right)$

Figure 3.9: The analog gas sensing front end architecture.

In this design, a 6-channel frontend has been implemented. The functional block diagram is shown in Figure 3.10. There are 6 current inputs IIN⟨1 : 6⟩. These ports can be connected to real sensors. However, for verifying the chip functionality, we use current inputs to emulate sensor resistance changes. Each channel consists of two logarithmic converters, a voltage-to-current converter, and an integrate-and-fire (IF) circuit that converts voltage to spike timing. The blocks and ports that are marked blue are used only during learning, including a vector normalizer, and sample-and-hold (S/H) circuits. As will be detailed in Section 3.3.3, the response vector $\mathbf{V_X}^k(c_1)$ is sampled by 6 off-chip ADCs through ports VX_OUT$\langle 1:6\rangle$, stored in an off-chip memory, and fed back to ports VX_DAC $\langle 1:6\rangle$ for computations in Phase II of each learning trial. Similarly, the updated vector $\mathbf{V_S}$ is sampled through VS_OUT$\langle 1:6\rangle$ by another 6 off-chip ADCs, stored in the off-chip memory, and fed back to ports VS_OUT$\langle 1:6\rangle$ for computations in next learning trail.

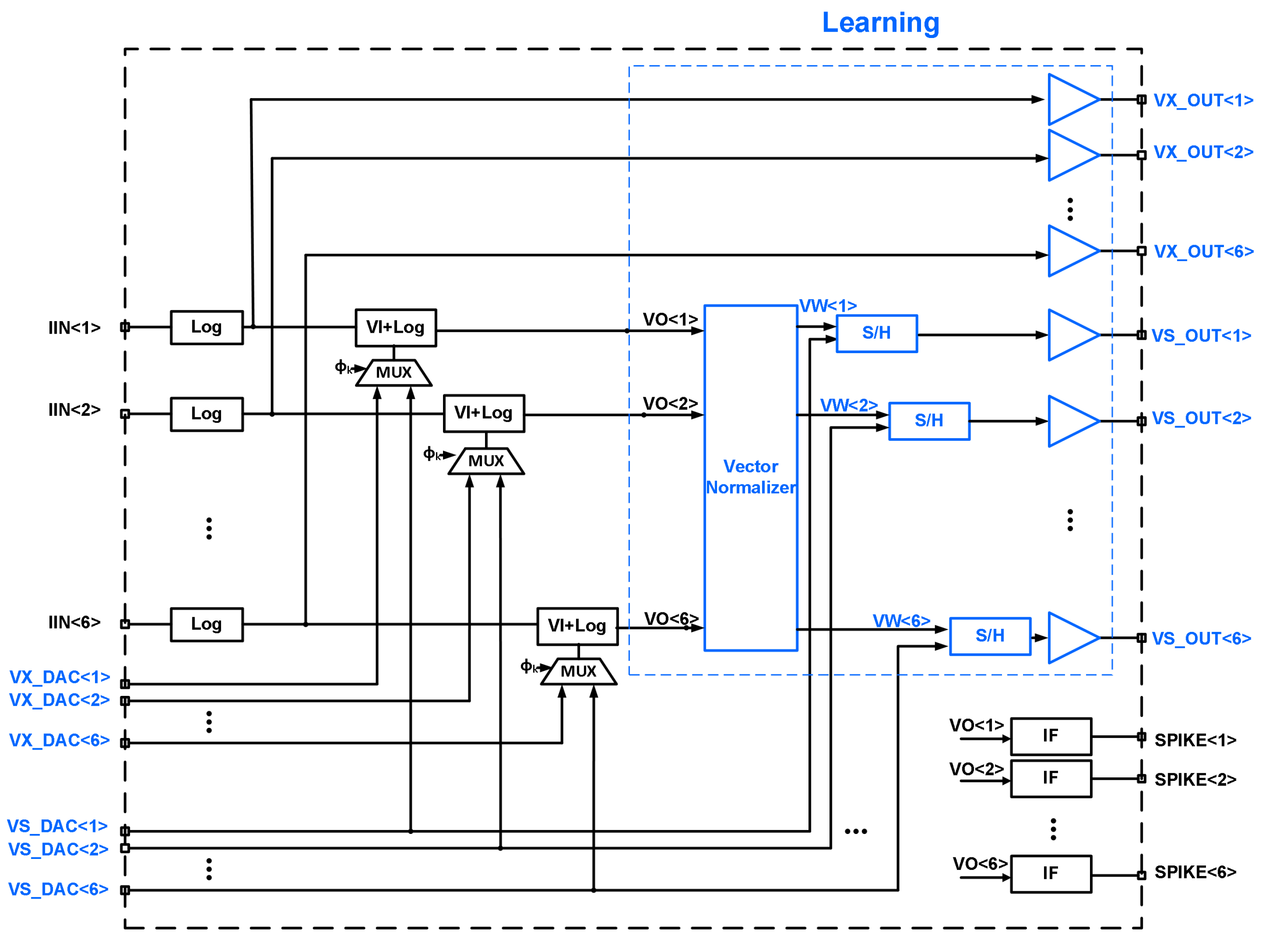


Figure 3.10: The functional block diagram.

This analog learning front end was implemented in TSMC 65-nm 1.2V Low Power (LP) technology. Figure 3.11 shows the layout. The entire die area is 1.7 mm×1.3 mm, with core area 0.65 mm×0.55 mm. The supply voltage is 0.6/0.65V[1].

[1]Most blocks are supplied at 0.6 V. However, some blocks have separate supplies to accommodate different process coreners and can be supplied up to 0.65 V.

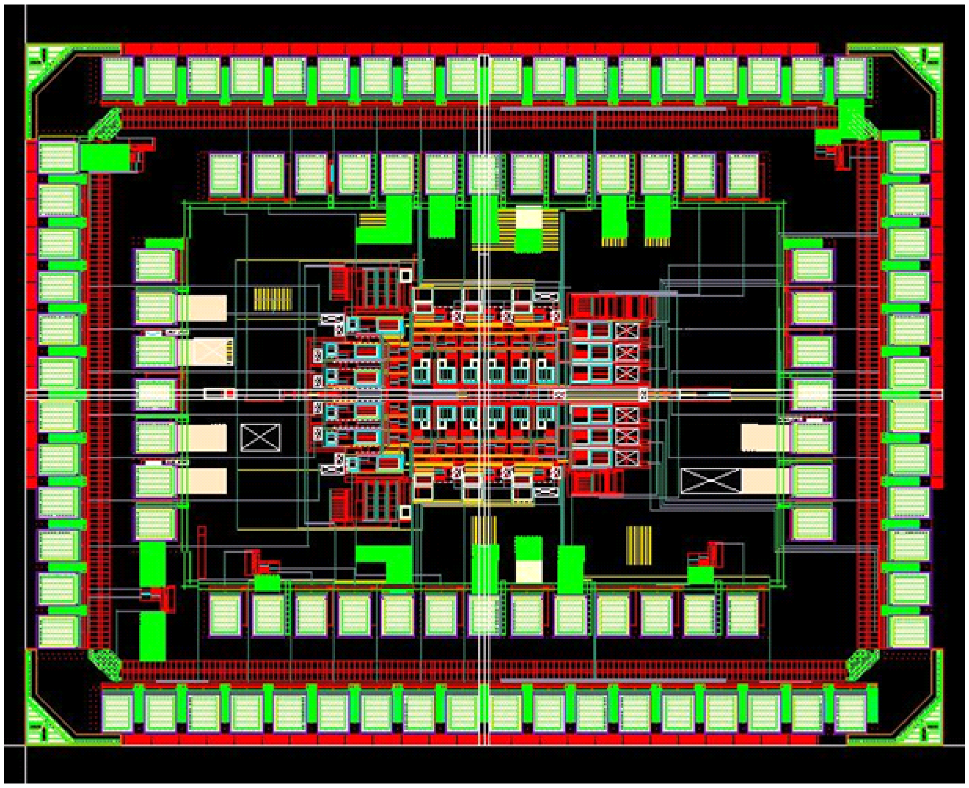


Figure 3.11: Layout of the analog gas sensing frontend.

3.3.2 Impelementation of the Feature Extraction Algorithm

Figure 3.12 shows the circuit implementation of the feature extraction algorithm for each channel. The op-amp in the input logarithmic converter fixes the input voltage at V_R. Therefore, the sensor resistance R_S is converted to current $I_R = V_R/R_S$. Transistor M3 is biased in the subthreshold, so V_X will be linear to $log(1/R_S)$. The op-amps in the V-I converters set the voltages on the right terminals of the resistors to be V_S. Therefore, the current flowing through the resistor in either one branch is $|V_X - V_S|/R$, and 0 in ther other branch. I_{XN} and I_{XP} are then mirrored and summed to flow through a stack of two transistors biased in the subthreshold. Hence the output voltage V_O will be linear to $log(log(V_R/R) - V_S)$ as desired. Finally the integrated-and-fire circuit converts V_O to the delay of the spike t through a linear relationship $t = C \cdot V_O/I$, where C is the integrating capacitance and I is a constant current.

The design and measurement results for each block will be described in the following subsections.

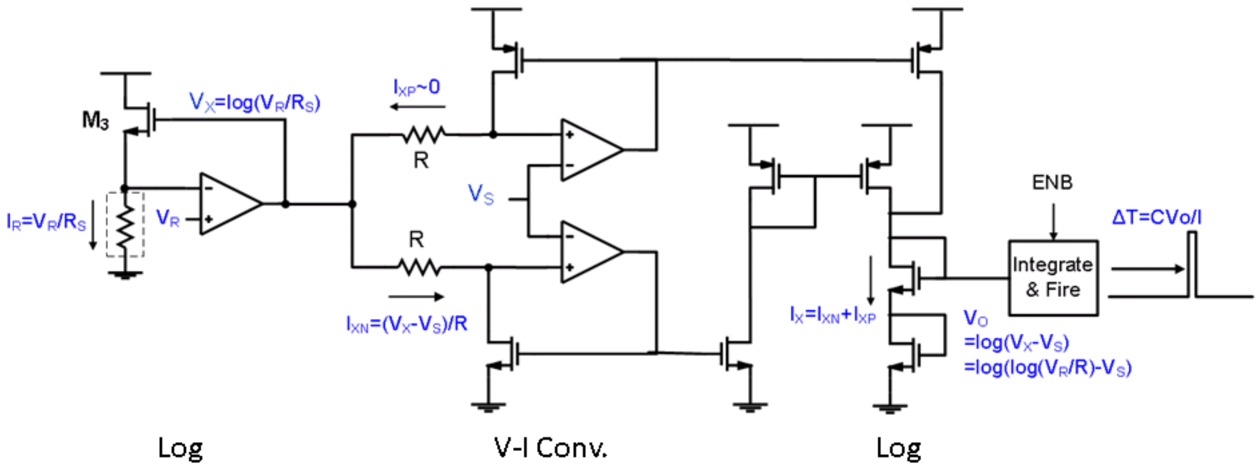


Figure 3.12: Implementation of the feature extraction algorithm.

Input Logarithmic Converter

Figure 3.13 shows the schematic of the input logarithmic converter. The input current ranges from 40 nA to 4 uA, and is ratioed down by 100 times with two stages of current mirrors. Since V_R is set at 0.2 V by the feedback loop, this emulates a sensor resistance range from 5 MΩ to 500 MΩ.

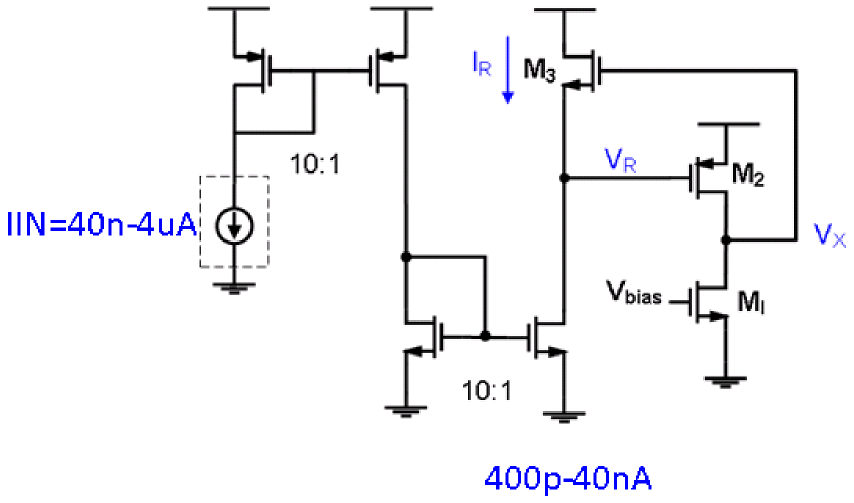


Figure 3.13: Schematic of the input logarithmic converter.

A common source amplifier (M_1 and M_2) instead of a two-input terminal op-amp is used to set V_R. The bias current is 100 nA from the 0.6-V supply. The loop gain is 25 dB and the loop bandwidth is 1.39 MHz. In the case where V_R is directly connected to sensor, this implements the resistance-to-current conversion $I_R = V_R/R_S$. Transistor M_3 is biased in the subthreshold, so its current can be modeled as [76]

$$I_R = I_0 \cdot exp\left(\frac{V_{GS3} - V_{TH3}}{nVt}\right) \tag{3.22}$$

where $V_{GS3} = V_X - V_R$, Vt is the thermal voltage, and both I_0 and n are empirical constants. Therefore,

$$V_X = V_R + V_{TH3} + nVt \cdot ln\left(\frac{I_R}{I_0}\right) \tag{3.23}$$

Since V_R, V_{TH3}, n, Vt, and I_0 are all constants, they can be lumped together. The input-output relationship then becomes

$$V_X = A + B \cdot ln(I_R) \tag{3.24}$$

or

$$V_X = A + B \cdot ln(\frac{V_R}{R_S}) \tag{3.25}$$

when the sensor is directly connected to V_R. Note that the variations in V_R, V_{TH3}, and I_0 between different channels do not matter because they can be captured by the learning algorithm. As for n and Vt, they are the same for all channels, as these two parameters depend only on process and temperature.

Figure 3.14 shows the simulated and measured V_X versus IIN, where V_X is measured from the output buffer. It can be seen that the input current is logarithmically linear to the output voltage.

V-I and Log Converters

Figure 3.15 shows the schematic of the V-I converter and the second log converter. The V-I converter consists of NMOS and PMOS branches for bidirectional current flow. Since current can only flow in either branch, the function of the V-I converter can be expressed as

$$I_{XN}, I_{XP} = \frac{|V_X - V_K|}{R} \tag{3.26}$$

To avoid the mirror pole in a current mirror loaded differential amplifier [77], a combination of a source follower and a common-gate amplifier is used for better loop stability. During the training phase, V_K is multiplexed between VX_DAC and VS_DAC to implement the update. After training, V_K is connected to VS_DAC. The input common-mode range of this amplifier is designed from 100 mV to 350 mV to cover the output range from the previous stage and the range of possible V_S. The input resistance is designed at 10 MΩ to achieve a loop gain of 41 dB. The loop bandwidth is around 480 kHz.

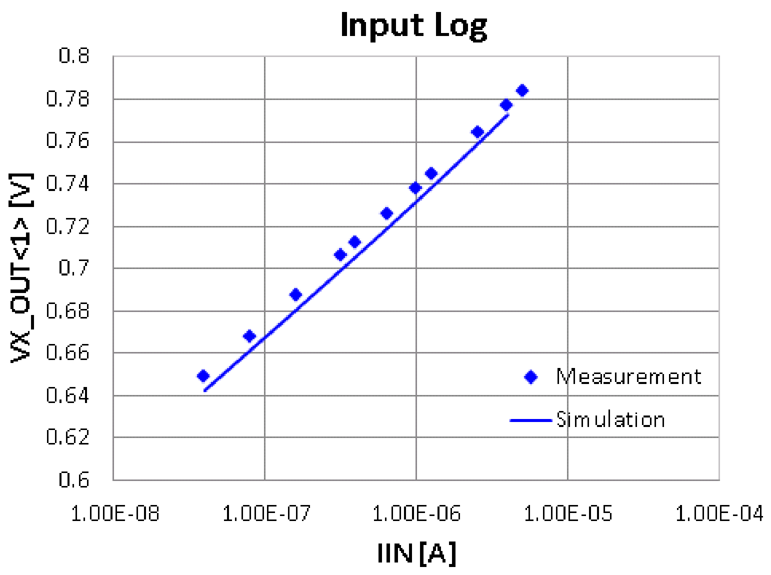


Figure 3.14: Simulated and measured output voltage versus input current for the input log converter.

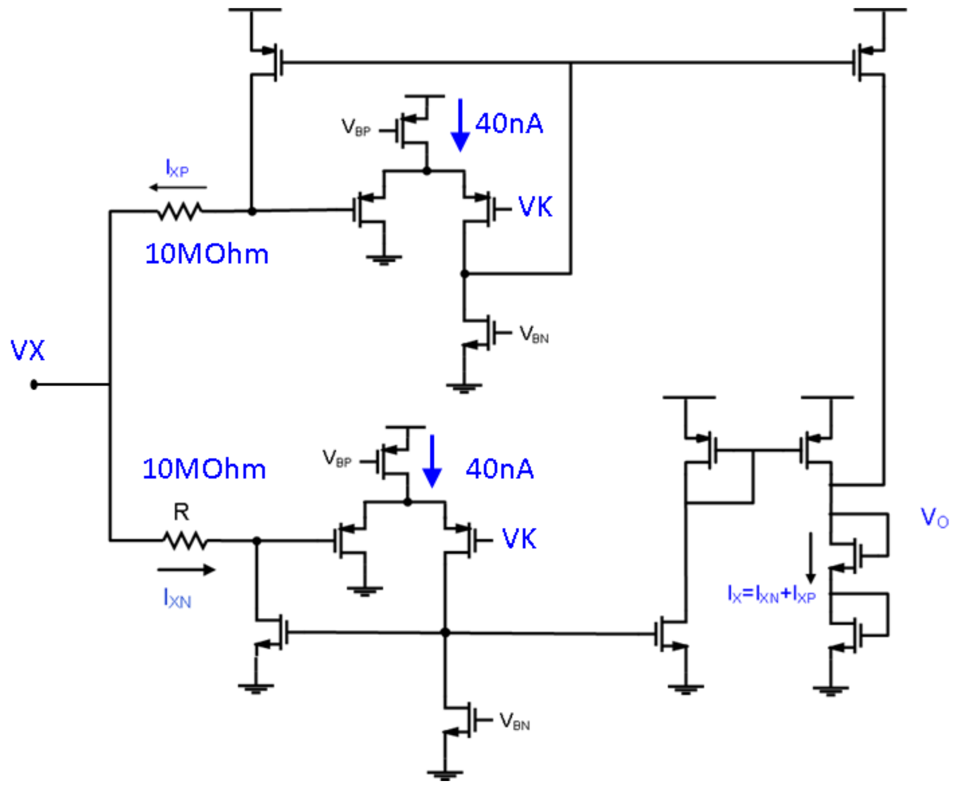


Figure 3.15: Schematic of the VI converter and the second log converter.

The currents in the NMOS branch I_{XN} and the PMOS branch I_{XP} are mirrored and summed into a stack of transistors designed in the subthreshold region. Therefore, the output voltage V_O will be linear to $log(I_{XN} + I_{XP})$. During the normal operation phase when V_K is conncected to VS_DAC, one can derive the expression of V_O from Equation 3.26 and Equation 3.25 to be

$$V_O = D + E \cdot ln(ln\frac{1}{R_S} - V_S) \tag{3.27}$$

where D, E are constants. This is the feature extraction algorithm as desired (Equation 3.15). Note the the constant shift D does not need to be matched between channels, because it will appear as a constant delay shift of the output spikes. Moreover, $E = n \cdot Vt$ is the same for all channels.

Pseudo-Resistor The high resistance at the input of the V-I converter requires large area if it's implemented as a passive resistor. A total of 18 such resistors will result in prohibitively large area. Using a transistor in the linear region, the source-drain voltage cannot be more than 100 mV. Therefore, in this design, we used pseudo-resistors [78], [79], [80], which can generate resistance in the order of MΩ to GΩ.

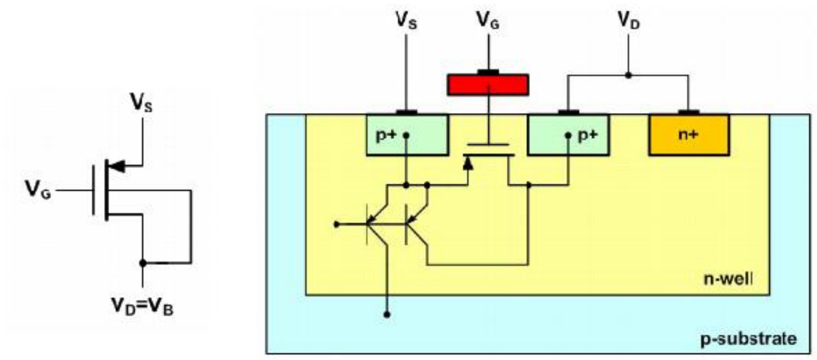
Figure 3.16 shows the implementation of the pseudo-resistor. For a PMOS transistor, when the bulk terminal is connected to the drain terminal, its threshold is modulated (decreased) as V_{SD} increases. Therefore, the I_{DS}-V_{DS} relationship at a constant V_{GS} would be linearized [78], compared to the normal configuration where the bulk is connected to the supply voltage. The resistance is determined by V_{SG}.

The pseudo-resistor can be made bi-directional and tunable in the configuration shown in Figure 3.16b. When one is in bulk-drain connected configuration, the other is diode-connected. Therefore, the resistance is dominated by the bulk-drain connected one. The resistance can be tuned by changing the source-gate voltage through I_B. In this design, I_B=24 nA.

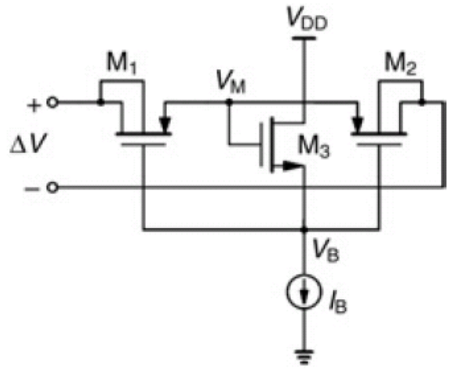
The drawback of this pseudo-resistor is that it can only operate at V_{SD} <0.4 V, otherwise the source-well PN junctions becomes forward-baised. In this design, the supply is 0.6 V and the voltage cross the pseudo-resistor is always less than 0.4 V. The other problem is the nonlinearity. The resistance variation is about ± 10% of the nominal resistance value.

Measurement Results The pseudo-resistor is characterized by measuring the input current I_X of the V-I converter when sweeping V_X directly from an external source. Figure 3.17 shows the simulation and measurement results when V_K is set at 0.25 V. The measured resistance is slightly lower than 10 MΩ due to the process variation. In this design, each pseudo-resistor consumes 72 nA from the 0.6-V supply.

The V-I converter is characterized by measuring I_X versus V_X at different V_K's. As shown in Figure 3.18, the I_X-V_X relationship is maintained at different V_K's less than 0.35 V. When V_K=0.4 V, the V-I converter operates out of its input common-mode range and therefore



(a) Bulk-drain connected PMOS transistor. Adapted from [79].



(b) Bi-directional and tunable pseudo-resistor. Adapted from [80].

Figure 3.16: Implementation of the pseudo-resistor.

cannot maintain the I_X-V_X relationship correctly. We also measured the output voltage of the log converter V_O versus V_X, as shown in Figure 3.19. It can also be seen that same input-output relationship is maintained until V_K=0.4V.

Finally, the functionality of the logarithmic converter is verified by plotting its output voltage V_O versus the measured input current to the V-I converter I_X (Figure 3.20). In this measurement, V_K is set at 0.2 V. It can be seen that the output voltage is linear to $log(I_X)$. However, the measured output voltage is about 50 mV higher than the simulation. It's because the process was shifted toward the SS corner. Therefore, to ensure linear V_X-$log(I_X)$ relationship for I_X up to 3 nA. The supply of the V-I and log converter is raised to 0.65 V.

Integrate-and-Fire Circuit

Figure 3.21 shows the schematic of the integrate-and-fire circuit. It is a modified version of the spiking neuron circuit in [81]. It consists of a comparator, an integrating capacitor C at the

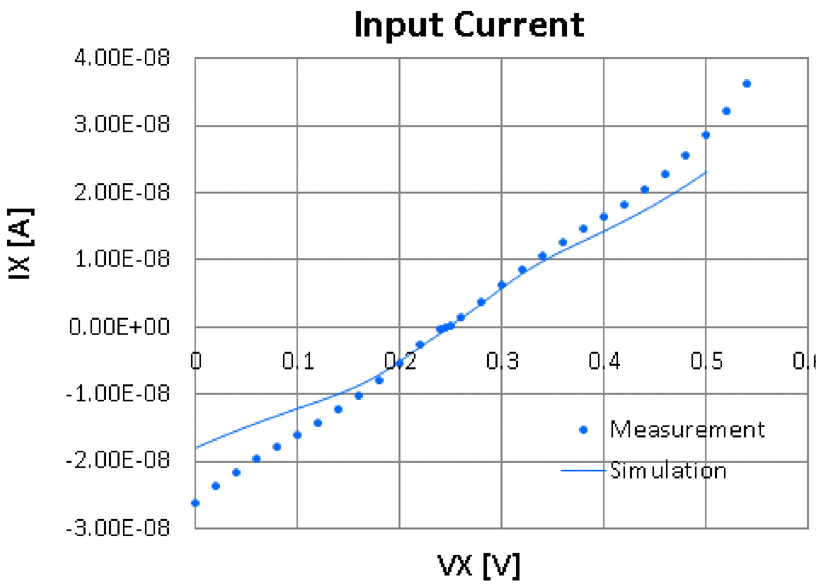


Figure 3.17: Input current $I_X = I_{XN} + I_{XP}$ of the V-I converter (V_K=0.25 V).

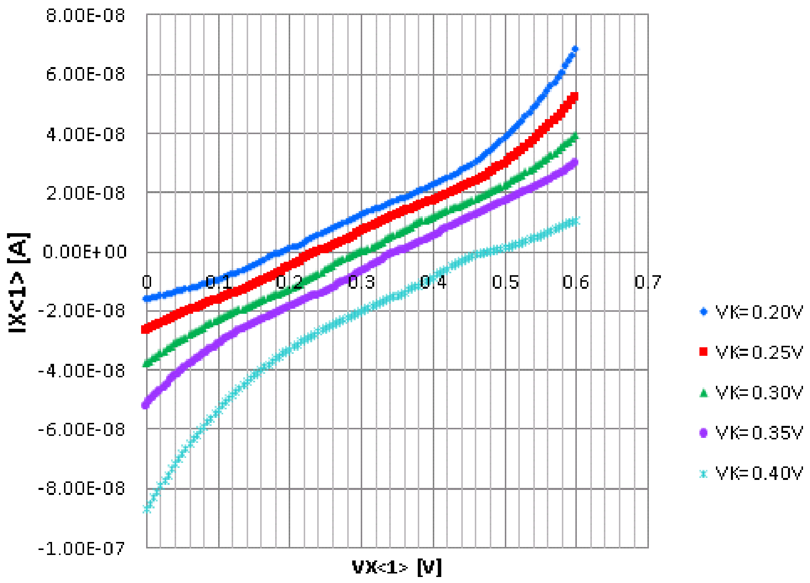


Figure 3.18: I_X versus V_X at different V_K's.

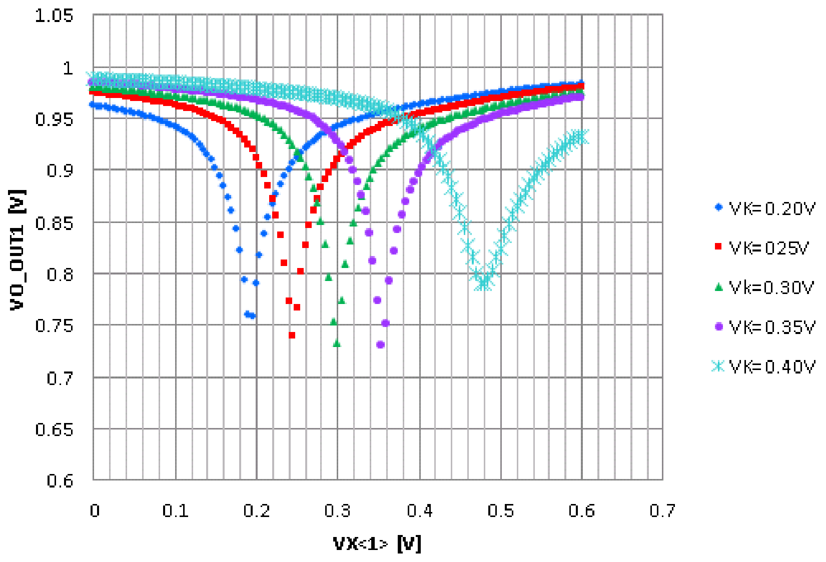


Figure 3.19: V_O versus V_X at different V_K's.

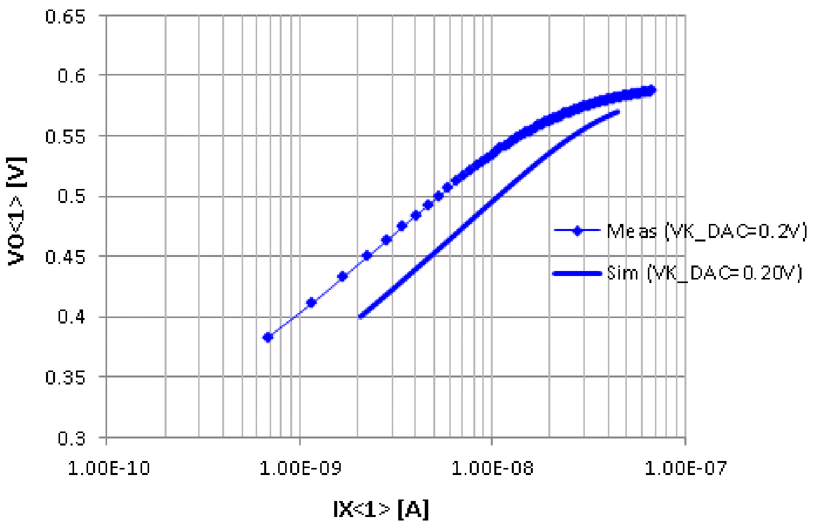


Figure 3.20: V_O versus $\log I_X$.

negative input terminal of the comparator, a feedback capacitor C_F, an SR latch, a charging and a discharging path. The output voltage from the previous stage V_O is connected to the positive input terminal of the comparator. When ENB signal transitions to 0, the circuit converts V_O into the delay of the output spike t as follows.

$$t = \frac{C \cdot V_O}{I_{INT}} \tag{3.28}$$

To understand the circuit operation, consider the circuit condition when ENB is initially high. The output of the OR gate is high. The charging path of C is closed, and the discharging path of C is open. Therefore, both V_C and V_{SPIKE} are at ground potential.

As soon as ENB signals low, the discharging path is closed, and the charging path is open. The integration capacitor is charged from 0 V by the integration current I_{INT}, and V_C increases linearly with time. When V_C reaches V_O, V_{SPIKE} switches to V_{DD}. The feedback capacitor C_F ensures the secure switching of the circuit. The new value of V_C is above the switching threshold V_O of the comparator, and depends on the values of C_F and C [81].

Once the pulse begins, the SR latch is set. Therefore, the charging path is closed, and the discharge path is open. $I_{DISCHARGE}$ sets the discharge rate of C, and thus the width of the spike. The circuit remains in this state until V_C decreases to the switching threshold of the comparator. At this point, V_{SPIKE} switches to ground potential, and the pulse is complete.

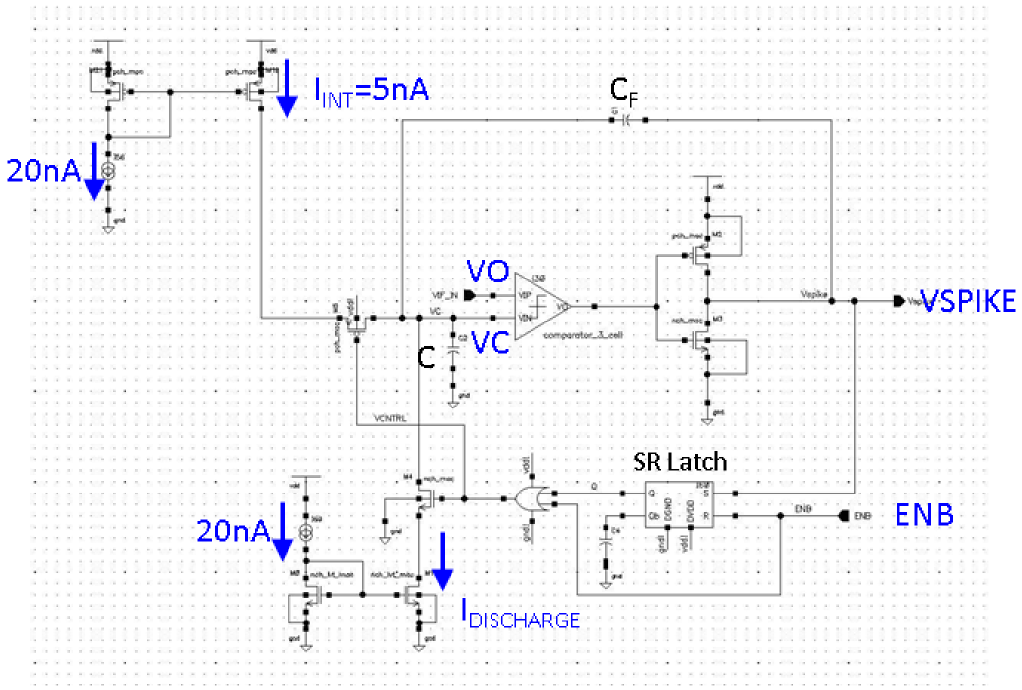


Figure 3.21: Schematic of the integrate-and-fire circuit.

After a spike is fired, the circuit no longer generates spikes because the output of the SR latch remains high. Only until ENB is brought high and back to low again will the circuit be

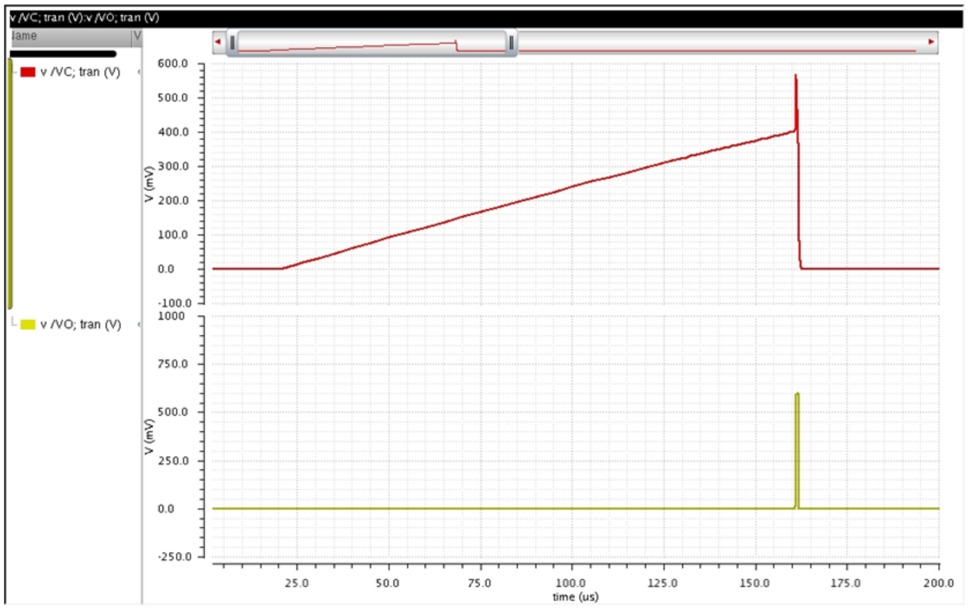


Figure 3.22: Simulation of V_C and V_{SPIKE}.

able to generate another spike. Figure 3.22 shows the transient simulation of V_C and V_{SPIKE} after ENB transitions to 0. In this simulation, the switching threshold of the comparator is set at 0.4 V. The delay of the spike is 175 μs and the width of the spike is about 1 μs.

Comparator Design This integrate-and-fire circuit must work for V_O ranging from 0.35 V up to 0.6 V. Therefore, a comparator with wide input range is required. Most comparators are designed at a particular quiescent input voltage and consumes a lot of static power. In this design, we used a self-based differential amplifier that can cover the entire input range from negative to positive supply [82].

The operation of this comparator can be understood through its derivation [82]. Figure 3.23(a) illustrate two folded-cascode differential amplifiers, each the complement of each other. While neither of them can cover the entire input range, the combination of them can do so. In the first step of the derivation, the loads of the two amplifiers are deleted, and their outputs are connected together to produce fully complementary but externally biased differential amplifier (Figure 3.23(b)). However, it cannot be biased stably, because any difference in currents between the top and bottom current sources would result in extreme shift in amplifier bias voltages.

Without the need to use complicated common-mode feedback circuitry, a simple modification can result in a complete stabilization of the bias voltages. By connecting all of the bias inputs to a single internal node, V_{BIAS}, the bias point of the amplifier can be stabilized through negative feedback, as illustrated in Figure 3.23(c). Any variations in processing parameters or operating conditions that shift the bias voltage away from their nominal values result in a shift in V_{BIAS} that corrects the bias voltages through negative feedback.

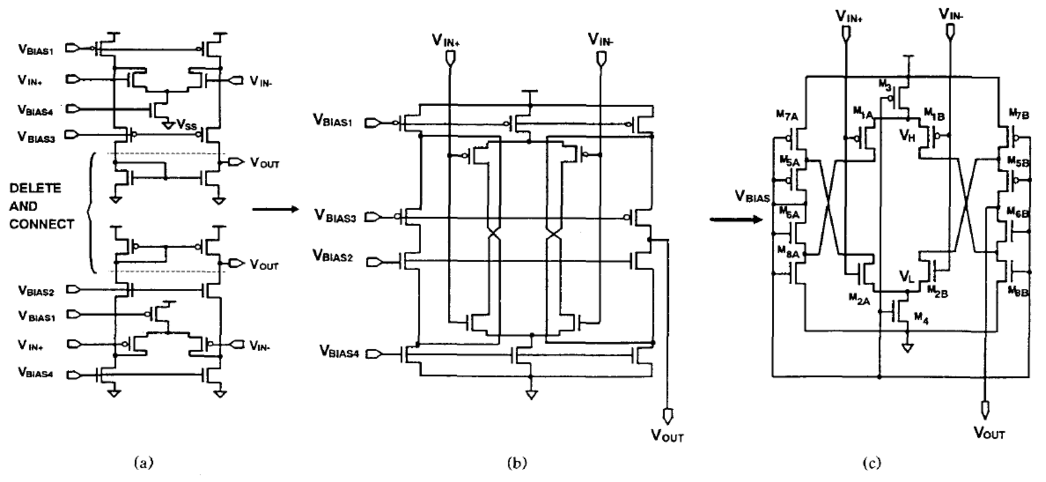


Figure 3.23: A self-biased comparator with rail-to-rail input range. Adapted from [82].

In this self-biased amplifier, the top and bottom current sources operate in the linear region. Consequently, their drain voltages can be set very close to the supply voltages. Since these two voltages determine the output swing of the amplifier, the output swing can be very close to the difference between the two supply rails. Another consequence of the linear operation of these two devices is that the amplifier can provide output switching currents that are significantly greater than its quiescent current. These two reasons are why it's suitable to be used as a comparator.

In this design, the nominal bias current is 64 nA from a supply of 0.6 V. It has a gain of 48.6 dB and a 3-dB bandwidth of 9.7 kHZ. The switching delay is 1.9 μs. Figure 3.24 shows the Monte Carlo simulation of the switching delay. The 6-sigma delay variation is about 3.7 μs. This variation would be significantly reduced after passing the subsequent inverter in the integrate-and-fire circuit. In this system, the switching of the comparator does not need to be fast, as only the matching of delays between channels matters. Therefore, this comparator can be designed with low power consumption, while having rail-to-rail input range.

Measurement Result Figure 3.25 shows the measured output spike in response to the transition of enable signal (ENB), which is global to all channels. The delay of the spike for each channel is measured with respect to the falling edge of ENB. The measured pulse width of the spike is 1.09 μs.

3.3.3 Implementation of the Learning Algorithm

Implementation Overview

The learning algorithm is mainly implemented by the current normalizer and sampled-and-hold circuits. By multiplexing V_K of the V-I converter between VS_DAC (which stores V_S

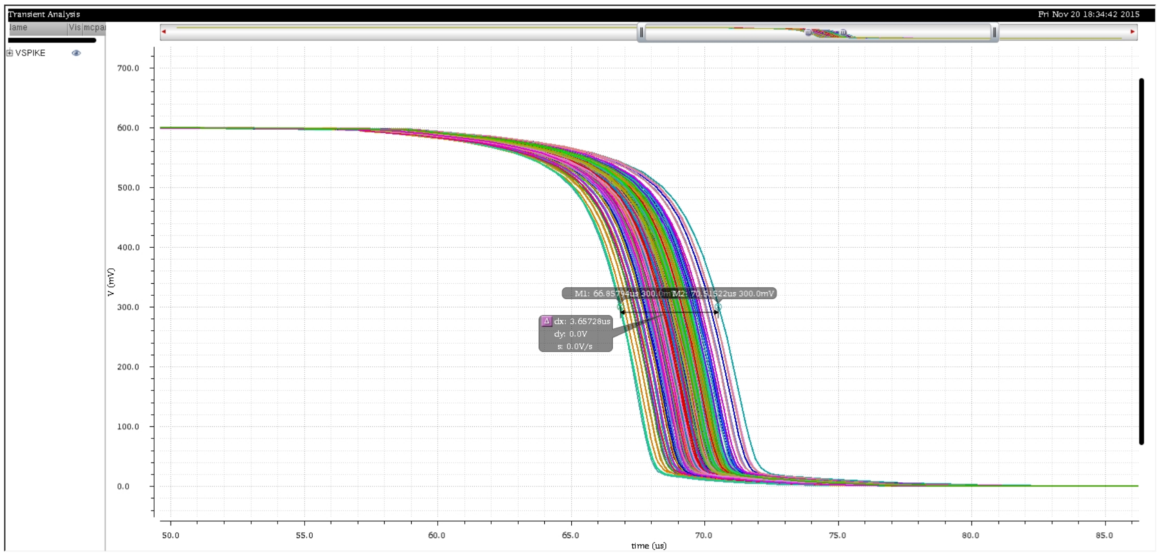
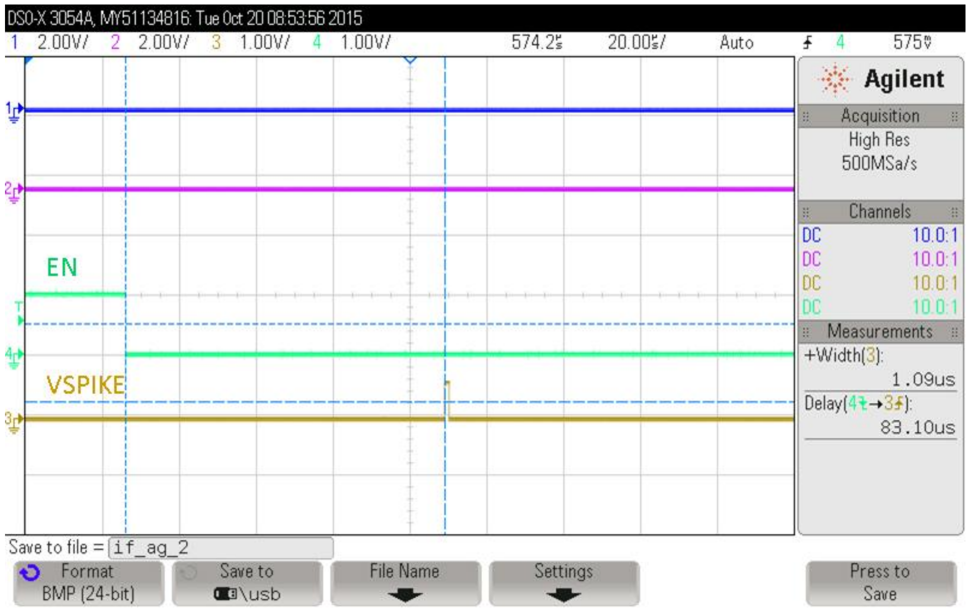


Figure 3.24: Monte Carlo simulation of the switching delay.


DSO-X 3054A, MY51134816: Tue Oct 20 08:53:56 2015
1 2.00V/ 2 2.00V/ 3 1.00V/ 4 1.00V/
574.2us 20.00us/ Auto 4 575mV
Agilent
EN
VSPIKE
Format
BMP (24-bit)
Save to
\usb
File Name
Settings
Press to
Save

Acquisition
High Res
500MSa/s

Channels
DC 10.0:1
DC 10.0:1
DC 10.0:1
DC 10.0:1

Measurements
+Width(3):
1.09us
Delay(4→3):
83.10us

Save to file = if_ag_2

Figure 3.25: Measured output spike in response to the transition of the enable signal (ENB).

from previous trial) and VX_DAC (which stores $V_X(c_1)$), the updated $\mathbf{V_S}$ vector can be obtained. Each trial of learning is divided into two phases. In Phase I, the analyte is presented at concentration level c_1. In Phase II, the analyte is presented at concentration level c_2.

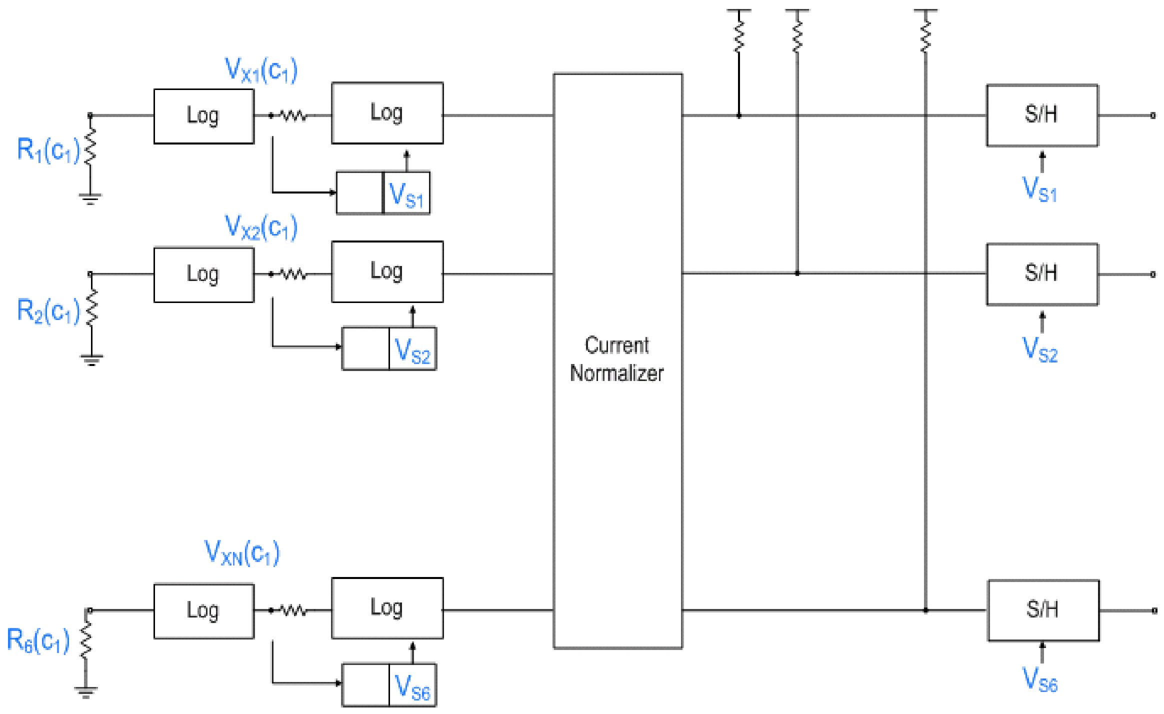


Figure 3.26: System states in Phase I of learning.

Phase I Figure 3.26 is the simplified system block diagram for better understanding the state changes in different learning phases. The two-column block under the second log converter in each channel represents values stored in off-chip memory. Their values are written through output ports VX_OUT⟨1 : 6⟩ and VS_OUT⟨1 : 6⟩ from the chip, and are read back to the chip through input ports VX_DAC⟨1 : 6⟩ and VS_DAC⟨1 : 6⟩. The right column stores V_S value updated from previous trial, and the left column stores $V_X(c_1)$ in Phase I of the current trial. The V_K node in the V-I converter is multiplexed between these two values to perform the computation of the update.

During Phase I of learning, the analyte is presented at a concentration level c_1, yielding a vector response $\mathbf{V_X}(c_1)$. This vector is stored in the left column in the memory block through VX_OUT⟨1 : 6⟩ ports and will be used in Phase II.

Phase IIa In Phase II, the analyte is presented at a concentration level c_2, yielding a vector response $\mathbf{V_X}(c_2)$. In Phase IIa, V_K nodes of the V-I converters are connected to $\mathbf{V_X}(c_1)$, as shown in Figure 3.27. Therefore, the input current vector to the V-I converter and subsequently to the current normalizer is

$$\mathbf{I_X} = \frac{\mathbf{V_X}(c_2) - \mathbf{V_X}(c_1)}{R} \tag{3.29}$$

The normalizer computes the Euclidean norm of the input vector and outputs:

$$\mathbf{V_{RW}} = \frac{\mathbf{V_X}^k(c_2) - \mathbf{V_X}^k(c_1)}{\|\mathbf{V_X}^k(c_2) - \mathbf{V_X}^k(c_1)\|} \cdot I_Y R_W \tag{3.30}$$

Also, the sample-and-hold circuits sample the $\mathbf{V_S}$ vector and output:

$$\mathbf{VS\_OUT} = \mathbf{V_S} \tag{3.31}$$

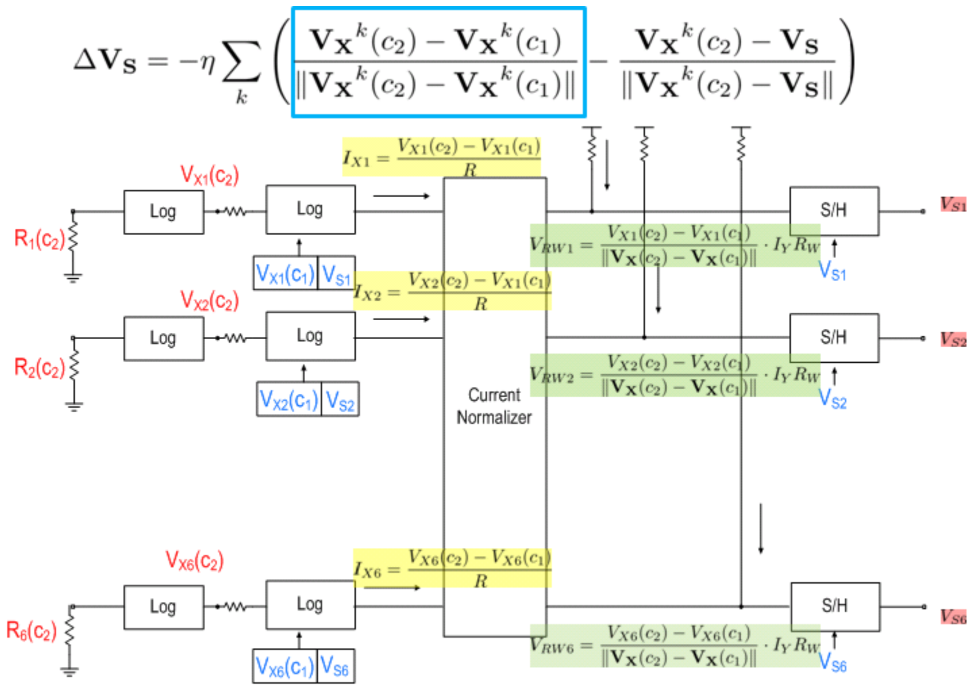

$\Delta \mathbf{V_S} = -\eta \sum_k \left(\boxed{\frac{\mathbf{V_X}^k(c_2) - \mathbf{V_X}^k(c_1)}{\|\mathbf{V_X}^k(c_2) - \mathbf{V_X}^k(c_1)\|}} - \frac{\mathbf{V_X}^k(c_2) - \mathbf{V_S}}{\|\mathbf{V_X}^k(c_2) - \mathbf{V_S}\|} \right)$
$I_{X1} = \frac{V_{X1}(c_2) - V_{X1}(c_1)}{R}$
$I_{X2} = \frac{V_{X2}(c_2) - V_{X1}(c_1)}{R}$
$I_{X6} = \frac{V_{X6}(c_2) - V_{X6}(c_1)}{R}$
$V_{X1}(C_2)$
$V_{X2}(C_2)$
$V_{X6}(C_2)$
$R_1(C_2)$
$R_2(C_2)$
$R_6(C_2)$
Log
$V_{X1}(C_1)$ V_{S1}
$V_{X2}(C_1)$ V_{S2}
$V_{X6}(C_1)$ V_{S6}
Current Normalizer
$V_{RW1} = \frac{V_{X1}(c_2) - V_{X1}(c_1)}{\|\mathbf{V_X}(c_2) - \mathbf{V_X}(c_1)\|} \cdot I_Y R_W$
$V_{RW2} = \frac{V_{X2}(c_2) - V_{X2}(c_1)}{\|\mathbf{V_X}(c_2) - \mathbf{V_X}(c_1)\|} \cdot I_Y R_W$
$V_{RW6} = \frac{V_{X6}(c_2) - V_{X6}(c_1)}{\|\mathbf{V_X}(c_2) - \mathbf{V_X}(c_1)\|} \cdot I_Y R_W$
S/H
V_{S1}
V_{S2}
V_{S6}

Figure 3.27: System states in Phase IIa of learning ($\mathbf{V_K} = \mathbf{V_X}(c_1)$).

Phase IIb In Phase IIb, V_K nodes of the V-I converters are switched to $\mathbf{V_S}$, as shown in Figure 3.28. Therefore, the input current vector to the V-I converter and subsequently to the current normalizer is

$$\mathbf{I_X} = \frac{\mathbf{V_X}(c_2) - \mathbf{V_S}}{R} \tag{3.32}$$

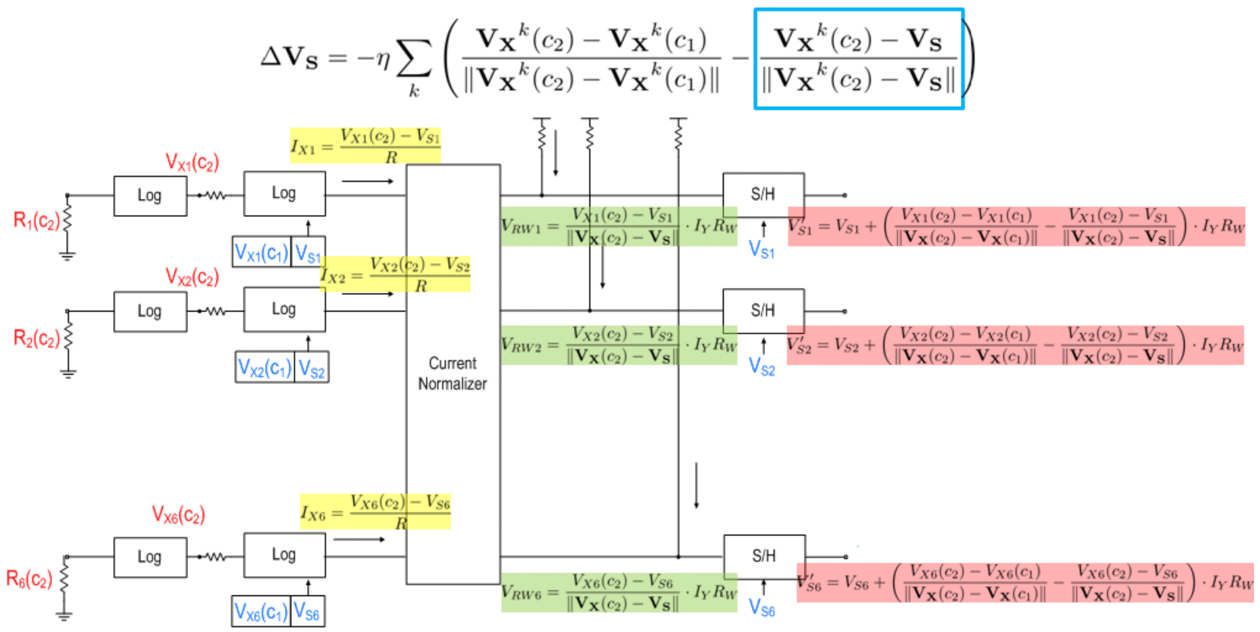

$\Delta \mathbf{V_S} = -\eta \sum_k \left(\frac{\mathbf{V_X}^k(c_2) - \mathbf{V_X}^k(c_1)}{\|\mathbf{V_X}^k(c_2) - \mathbf{V_X}^k(c_1)\|} - \boxed{\frac{\mathbf{V_X}^k(c_2) - \mathbf{V_S}}{\|\mathbf{V_X}^k(c_2) - \mathbf{V_S}\|}} \right)$
$V_{X1}(C_2)$
$R_1(C_2)$
Log
Log
$V_{X1}(C_1)$ V_{S1}
$I_{X1} = \frac{V_{X1}(c_2) - V_{S1}}{R}$
$V_{X2}(C_2)$
$R_2(C_2)$
$V_{X2}(C_1)$ V_{S2}
$I_{X2} = \frac{V_{X2}(c_2) - V_{S2}}{R}$
$V_{X6}(C_2)$
$R_6(C_2)$
$V_{X6}(C_1)$ V_{S6}
$I_{X6} = \frac{V_{X6}(c_2) - V_{S6}}{R}$
Current Normalizer
$V_{RW1} = \frac{V_{X1}(c_2) - V_{S1}}{\|\mathbf{V_X}(c_2) - \mathbf{V_S}\|} \cdot I_Y R_W$
$V_{RW2} = \frac{V_{X2}(c_2) - V_{S2}}{\|\mathbf{V_X}(c_2) - \mathbf{V_S}\|} \cdot I_Y R_W$
$V_{RW6} = \frac{V_{X6}(c_2) - V_{S6}}{\|\mathbf{V_X}(c_2) - \mathbf{V_S}\|} \cdot I_Y R_W$
S/H
V_{S1}
V_{S2}
V_{S6}
$V'_{S1} = V_{S1} + \left(\frac{V_{X1}(c_2) - V_{X1}(c_1)}{\|\mathbf{V_X}(c_2) - \mathbf{V_X}(c_1)\|} - \frac{V_{X1}(c_2) - V_{S1}}{\|\mathbf{V_X}(c_2) - \mathbf{V_S}\|} \right) \cdot I_Y R_W$
$V'_{S2} = V_{S2} + \left(\frac{V_{X2}(c_2) - V_{X2}(c_1)}{\|\mathbf{V_X}(c_2) - \mathbf{V_X}(c_1)\|} - \frac{V_{X2}(c_2) - V_{S2}}{\|\mathbf{V_X}(c_2) - \mathbf{V_S}\|} \right) \cdot I_Y R_W$
$V'_{S6} = V_{S6} + \left(\frac{V_{X6}(c_2) - V_{X6}(c_1)}{\|\mathbf{V_X}(c_2) - \mathbf{V_X}(c_1)\|} - \frac{V_{X6}(c_2) - V_{S6}}{\|\mathbf{V_X}(c_2) - \mathbf{V_S}\|} \right) \cdot I_Y R_W$

Figure 3.28: System states in Phase IIb of learning ($\mathbf{V_K} = \mathbf{V_S}$).

The output of the normalizer then becomes

$$\mathbf{V_{RW}} = \frac{\mathbf{V_X}^k(c_2) - \mathbf{V_S}}{\|\mathbf{V_X}^k(c_2) - \mathbf{V_S}\|} \cdot I_Y R_W \tag{3.33}$$

As will be described in the following subsection, the sample-and-hold circuit only senses the difference of its input $\mathbf{V_{RW}}$, so its output value would be updated to

$$\mathbf{VS\_OUT'} = \mathbf{V_S} + \left(\frac{\mathbf{V_{X1}}(c_2) - \mathbf{V_{X1}}(c_1)}{\|\mathbf{V_X}(c_2) - \mathbf{V_X}(c_1)\|} - \frac{\mathbf{V_X}(c_2) - \mathbf{V_S}}{\|\mathbf{V_X}(c_2) - \mathbf{V_S}\|} \right) \cdot I_Y R_W \tag{3.34}$$

Therefore, the V_S has been updated according to the update rule.

Analog Euclidian Vector Normalizer Design

Normalization is pervasive, from the frontmost automatic gain control (AGC), to the computation of orthonormal vectors in principle component analysis (PCA). In [83], a 16-channel analog normalizer was first demonstrated to perform concurrent scaling of the input vector such that the peak component value is adjusted to a reference value. A subtractive normalizer was also reported in [84] to implement a reverse water-filling algorithm. However, a Euclidian vector normalizer, which is essential in algorithms involving the computation of angles between vectors, hasnt been reported yet. In this section, we present a compact 6-channel analog Euclidian vector normalizer using feedback, without needing to synthesize the square root of the squared sum and generating replicas to each channel for division. A calibration method is also provided to overcome the mismatches between transistors.

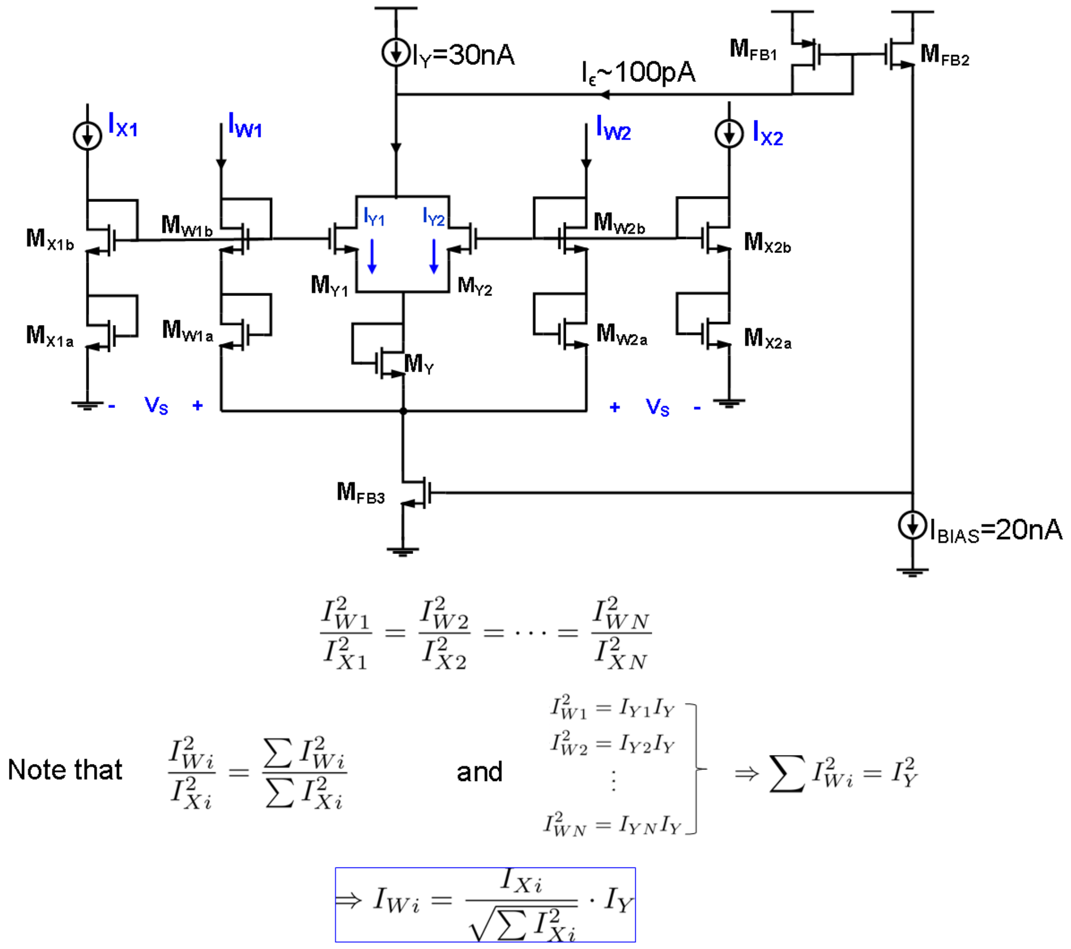

I_Y=30nA
I_ϵ~100pA
M_{FB1}
M_{FB2}
I_{X1}
I_{W1}
I_{W2}
I_{X2}
M_{X1b}
M_{W1b}
I_{Y1}
I_{Y2}
M_{W2b}
M_{X2b}
M_{Y1}
M_{Y2}
M_{X1a}
M_{W1a}
M_Y
M_{W2a}
M_{X2a}
- V_S +
+ V_S -
M_{FB3}
I_{BIAS}=20nA
$\frac{I_{W1}^2}{I_{X1}^2} = \frac{I_{W2}^2}{I_{X2}^2} = \cdots = \frac{I_{WN}^2}{I_{XN}^2}$
Note that $\frac{I_{Wi}^2}{I_{Xi}^2} = \frac{\sum I_{Wi}^2}{\sum I_{Xi}^2}$ and $I_{W1}^2 = I_{Y1}I_Y$, $I_{W2}^2 = I_{Y2}I_Y$, $\vdots$, $I_{WN}^2 = I_{YN}I_Y$ $\Rightarrow \sum I_{Wi}^2 = I_Y^2$
$\boxed{\Rightarrow I_{Wi} = \frac{I_{Xi}}{\sqrt{\sum I_{Xi}^2}} \cdot I_Y}$

Figure 3.29: Basic circuit concept of the current normalizer.

Figure 3.29 shows the basic circuit concept. The schematic shows a 2-channel normalizer to better illustrate the operation. I_{X1} and I_{X2} are input currents, while I_{W1} and I_{W2} are output currents. I_Y is a current source biased at 30nA. Transistors M_{FB1}-M_{FB3} form the feedback loop to adjust V_S. Adding one more channel i simply adds one more branch of I_{Xi} and I_{Wi}, and one more transistor M_{Yi} between transistor M_Y and the current source I_Y. Therefore, in the following derivation, we will generalize the number of channels to N.

If all the transistors are matched and operate in the subthreshold regime, then Equation 3.35 holds.

$$\frac{I_{W1}^2}{I_{X1}^2} = \frac{I_{W2}^2}{I_{X2}^2} = \cdots = \frac{I_{WN}^2}{I_{XN}^2} \tag{3.35}$$

Note that from Equation 3.36 and 3.37:

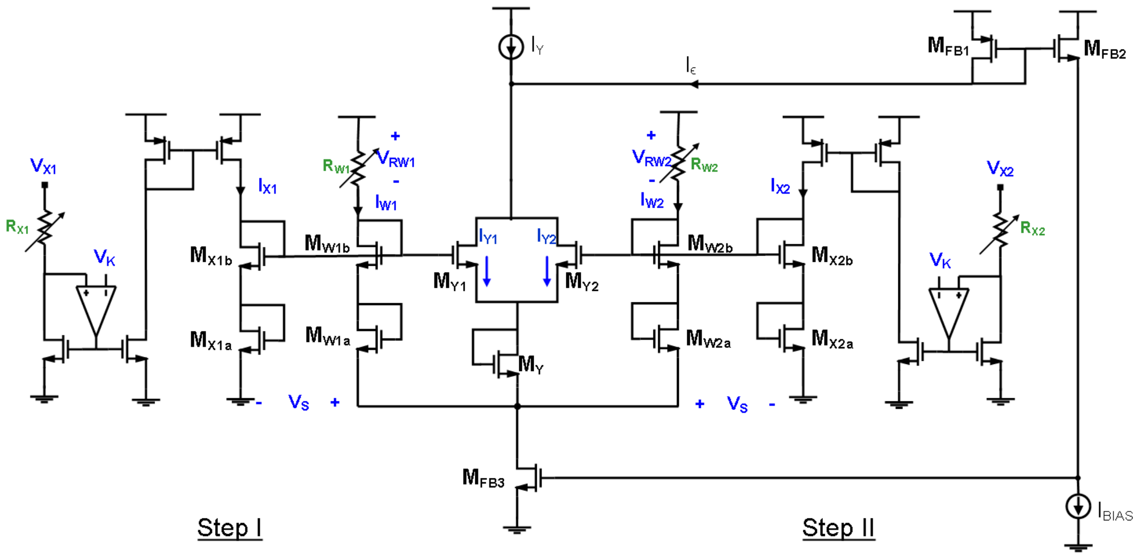
$$\frac{I_{Wi}^2}{I_{Xi}^2} = \frac{\sum I_{Wi}^2}{\sum I_{Xi}^2} \tag{3.36}$$

$$\sum I_{Wi}^2 = I_Y^2 \tag{3.37}$$

it can be shown that

$$I_{Wi} = \frac{I_{Xi}}{\sqrt{\sum I_{Xi}^2}} \cdot I_Y \tag{3.38}$$

Therefore, the output current vector $(I_{W1}, I_{W2}, , I_{WN})$ is a Euclidian norm of the input current vector $(I_{X1}, I_{X2}, , I_{XN})$.



Step I

$$\begin{cases} I_{W1}^2 = \alpha_1 I_{Y1} I_Y \\ I_{W1}^2 = \alpha_2 I_{Y2} I_Y \\ \vdots \\ I_{WN}^2 = \alpha_N I_{YN} I_Y \end{cases} \text{i.e.} \begin{cases} V_{RW1}^2 = R_{W1}^2 I_{W1}^2 = R_{W1}^2 \alpha_1 I_{Y1} I_Y \\ V_{RW2}^2 = R_{W2}^2 I_{W2}^2 = R_{W2}^2 \alpha_2 I_{Y2} I_Y \\ \vdots \\ V_{RWN}^2 = R_{WN}^2 I_{WN}^2 = R_{WN}^2 \alpha_N I_{YN} I_Y \end{cases}$$

Tune R_{wi} such that $V_{RWi} = V_{REF}$ when $I_{Yi} = I_Y$

Then $R_{W1}^2 \alpha_1 = R_{W2}^2 \alpha_2 = \cdots = R_{WN}^2 \alpha_N \equiv R_W^2$

$$\boxed{\Rightarrow \sum V_{Wi}^2 = R_W^2 I_Y^2}$$

Step II

$$\begin{cases} I_{W1}^2 = \beta_1 I_{X1}^2 \\ I_{W2}^2 = \beta_2 I_{X2}^2 \\ \vdots \\ I_{WN}^2 = \beta_N I_{XN}^2 \end{cases} \text{i.e.} \begin{cases} V_{RW1}^2 = R_{W1}^2 \beta_1 \left(\frac{V_{X1} - V_K}{R_{X1}} \right)^2 \\ V_{RW2}^2 = R_{W2}^2 \beta_2 \left(\frac{V_{X2} - V_K}{R_{X2}} \right)^2 \\ V_{RWN}^2 = R_{WN}^2 \beta_N \left(\frac{V_{XN} - V_K}{R_{XN}} \right)^2 \end{cases}$$

Tune R_{xi}'s such that $V_{RW1} = V_{RW2} = \cdots = V_{RWN}$

when $V_{X1} - V_K = V_{X2} - V_K = \cdots = V_{XN} - V_K$

Then $R_{W1}^2 \beta_1 / R_{X1}^2 = R_{W2}^2 \beta_2 / R_{X2}^2 = \cdots = R_{WN}^2 \beta_N / R_{XN}^2$

$$\boxed{\Rightarrow \frac{V_{RW1}^2}{(V_{X1} - V_K)^2} = \frac{V_{RW2}^2}{(V_{X2} - V_K)^2} = \cdots = \frac{V_{RWN}^2}{(V_{XN} - V_K)^2}}$$

Figure 3.30: The calibration procedure of the current normalizer.

However, any subthreshold circuits are subjected to threshold variations, and Equation 3.35 to Equation 3.38 no longer hold true. Therefore, a calibration procedure in conjuction with the V-I converter is proposed, as shown in Figure 3.30. Only the NMOS branch of the V-I converter is shown for clarity. As mentioned previously, the input currents I_X's are generated by V-I converters, converting input voltages V_{Xi} to currents $I_{Xi} = (V_{Xi} - V_K)/R_{Xi}$, where V_K is the input bias of the op-amp. Resistors R_X's and R_W's are both nominally 10-MΩ tunable pseudo-resistors as described in Section 3.3.2.

Step I calibrates the mismatches between I_Y's and I_W's. By directing all I_Y to each branch sequentially (by setting the input voltage of the channel to 0.6V while those of the rest to V_K) and tuning the R_W's such that each V_{RW} equal to an arbitrary reference voltage, then the squared sum of V_{RW}'s will be equal to a fixed value $R_W^2 I_Y^2$. That is,

$$\sum V_{Wi}^2 = R_W^2 I_Y^2 \tag{3.39}$$

Step II calibrates the mismatches between I_X's and I_W's, including the threshold mismatch and the gain error from the current mirror. Let input voltages of all channels V_{Xi}'s are equal. By tuning R_X's such that all V_{RW}'s are equal, then the condition described in Equation 3.40 can be obtained:

$$\frac{V_{RW1}^2}{(V_{X1}-V_K)^2} = \frac{V_{RW2}^2}{(V_{X2}-V_K)^2} = \cdots = \frac{V_{RWN}^2}{(V_{XN}-V_K)^2} \tag{3.40}$$

From Equation 3.39 and Equation 3.40, the output voltage vector is a Euclidian norm of the input voltage vector:

$$V_{RWi} = \frac{V_{Xi}-V_K}{\sqrt{\sum(V_{Xi}-V_K)^2}} \cdot R_W I_Y \tag{3.41}$$

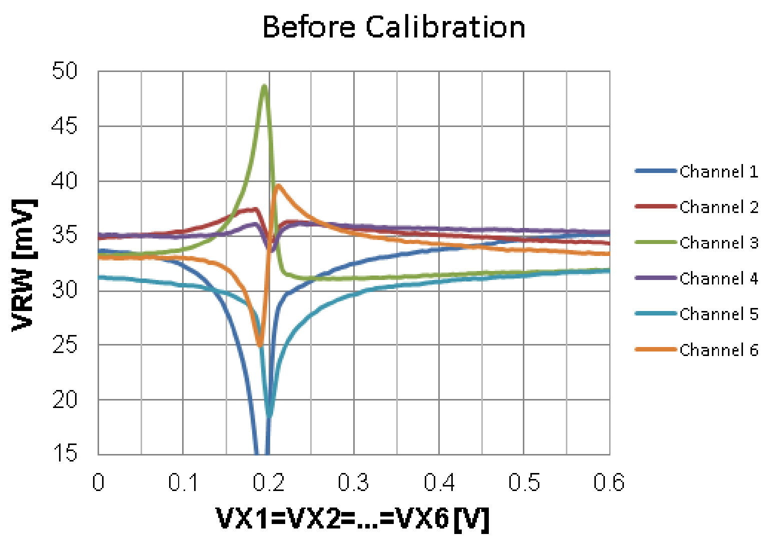
Figure 3.31 shows the output voltages of each channel when all the input voltages V_X's are swept from 0 to 0.6V with $V_K = 0.2V$. Before calibration, the maximum difference between each channel outputs after 0.25V is 9.04 mV. However, after calibration, the difference is reduced to 1.74 mV. In the regime where V_X is close to 0.2V and I_X's are small (<1nA), the large output differences are due to the op-amp offsets in the V-I converters.

Figure 3.32 shows the output voltages of each channel when the normalizer is used as a single channel (a), 2-channel (b), and 4-channel (c) normalizers. The output of the single-channel normalizer approaches 75.27 mV. The theoretical output voltages of the 2-channel, and 4-channel normalizers are $75.27/\sqrt{2}$=53.22 mV, and 75.27/2=37.64 mV. The errors for the on channels from V_X=0.25 to 0.6 V are less than 2.05 mV and 2.81 mV, respectively. The output error of the 6-channel normalizer in Figure 3.30 with respect to the $75.27/\sqrt{6}$=30.73 mV is also less than 3mV (9.76%). The errors are mostly due to the nonlinearlity in the pseudo-resistors.

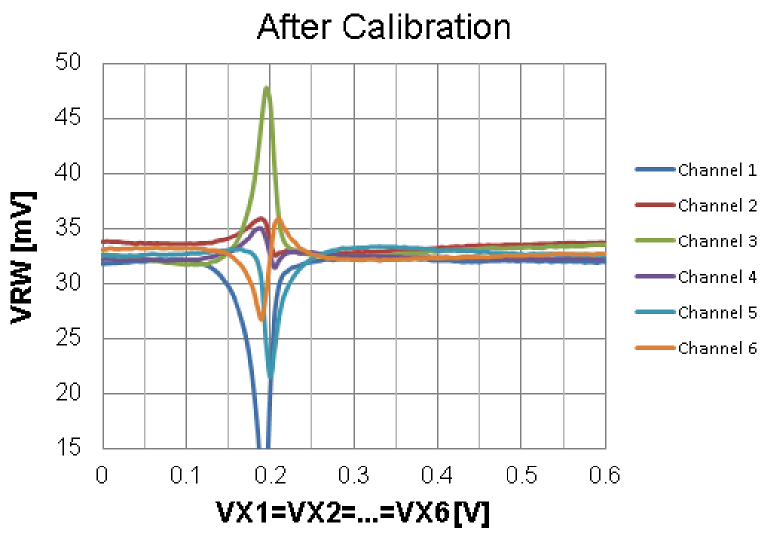
When V_{X3} is applied to a square wave toggling between 0.3 V and 0.35 V, while the inputs for channels 1, 2, 4, 5, 6 are 0.3V, 0.3V, 0.35V, 0.3V, and 0.3V, the rise time and fall time of channel 3 output for a 500-fF load are 13 μs and 6.2 μs, respectively. The rise time is mainly limited by the bias current I_Y. The normalizer consumes a maximum current of 143.5 nA from the 0.6-V supply.

Sample-and-Hold Circuit

Figure 3.33 shows the schematic of the sample-and-hold circuit. The timing between the clock signal Φ and the input signal V_W is shown in Figure 3.34.

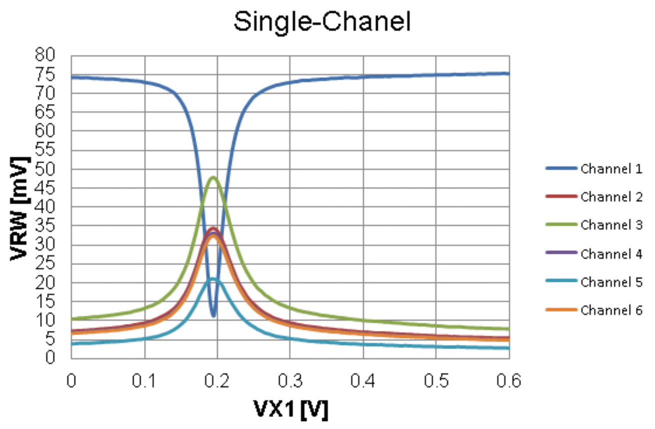


(a)

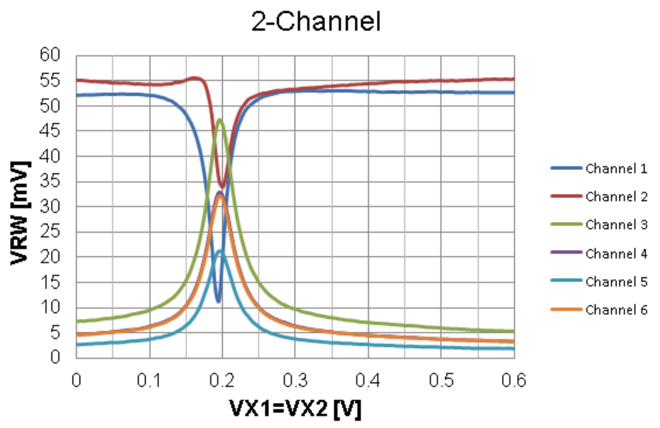


(b)

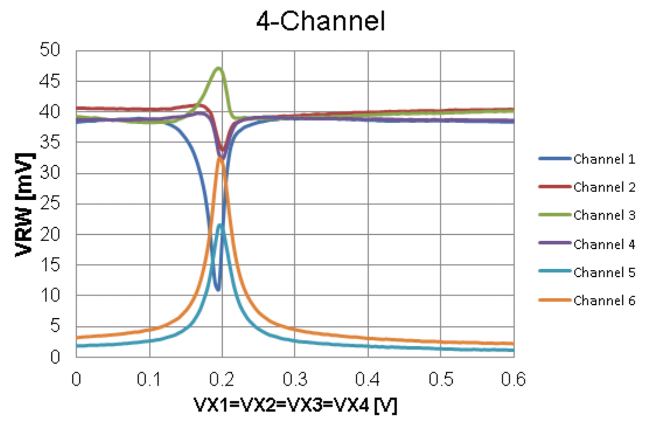
Figure 3.31: Outputs voltages (a) before and (b) after calibration when all channels are swept from 0 to 0.6V with V_K=0.2V.



(a)



(b)



(c)

Figure 3.32: Output voltages when the normalizer has only (a) 1 channel (b) 2 channels (c) 4 channels on. The input of the off-channels are connected to V_K=0.2V.

When Φ is high and Φ_B is low, the amplifier is configured in unity-gain feedback. Therefore, the output of the sample-and-hold circuit is $V_{CM} + V_{OS}$, where V_{CM} is input bias and V_{OS} is the offset voltage of the op-amp. Capacitor C_2 is connected to V_{DAC} from the off-chip DAC. Capacitor C_1 is connected to the output of the normalizer V_W, which does not affect the output in this phase.

When Φ becomes low and Φ_B becomes high, the amplifier is connected in a feedback through C_2. Since the voltage across a capacitor cannot change instantaneously, the output of the sample-and-hold circuit $V_{SH,OUT}$ becomes V_{DAC}. If V_W transitions from V_{WA} to V_{WB} in this phase, a total charge of $C_1(V_{WA} - V_{WB})$ would be transferred to the output of the sample-and-hold, resulting in a voltage difference of

$$\Delta V_{SH,OUT} = \frac{C_1(V_{WA} - V_{WB})}{C_2} \tag{3.42}$$

If $C_1 = C_2$, then the output becomes

$$\begin{aligned} V_{SH.OUT} &= V_{DAC} + \Delta V_{SH,OUT} && (3.43)\\ &= V_{DAC} + V_{WA} - V_{WB} && (3.44) \end{aligned}$$

Therefore, it implements the update function as desired in the learning phase.

This design is inherently immune to the amplifier offset, because only the difference in the input ΔV_W is sensed. Figure 3.35 shows the Monte Carlo simulation of the internal node between the two capacitors and the output of the sample-and-hold circuit. It can be seen that even though the amplifier has a large offset variation ($\approx$ 80 mV), the sampled voltage and updated voltage are the same for all Monte Carlo trials.

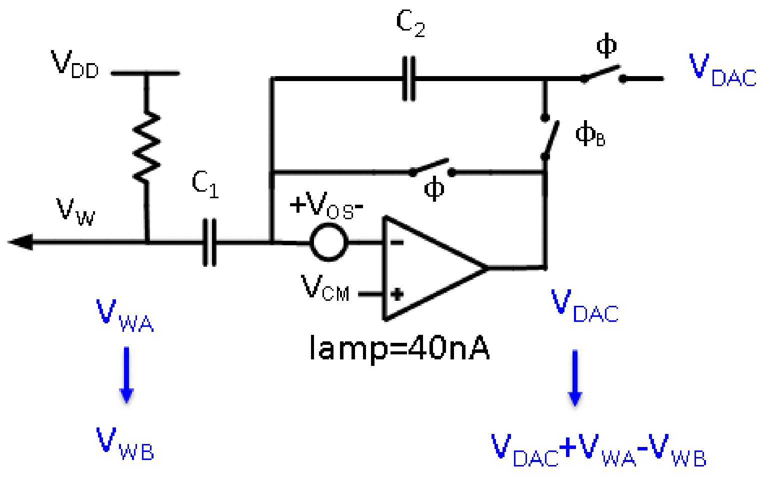


Figure 3.33: Schematic of the sample-and-hold circuit.

Figure 3.36 shows the measured sample-and-hold output in response to the clock and a test point voltage V_{TP}. The transition of V_{TP} will cause the transition of V_W in the opposite

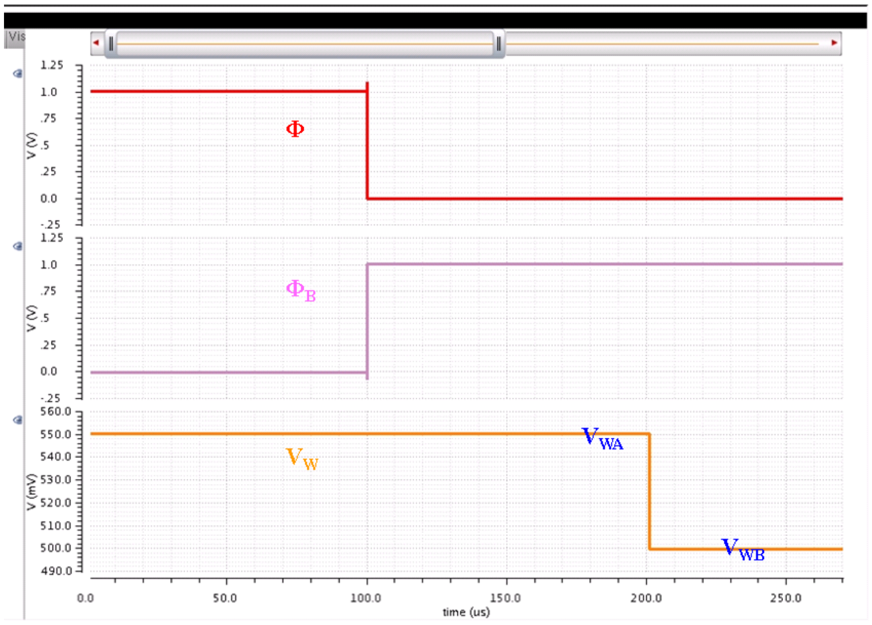


Figure 3.34: Timing between the clock signal Φ and the input signal V_W.

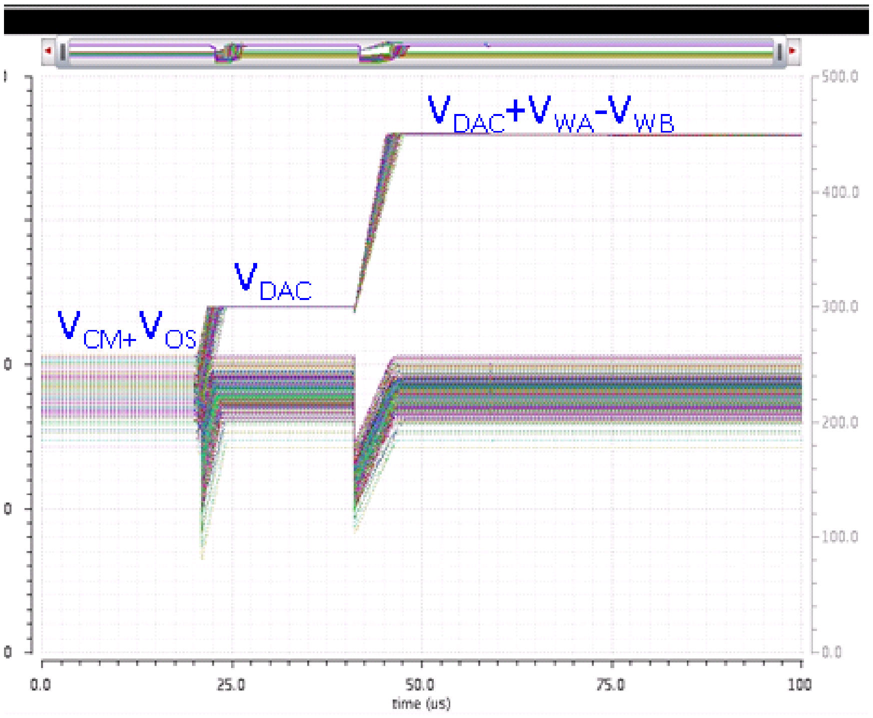


Figure 3.35: Monte Carlo simulation for the sample-and-hold circuit.

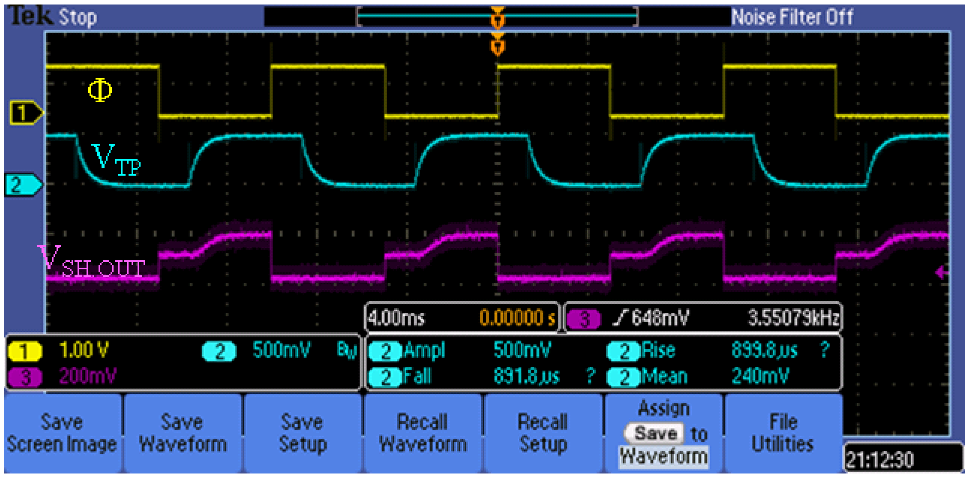


Figure 3.36: Measured output of the sampled-and-hold circuit.

direction. Therefore, V_W actually transitions to a more negative voltage after clock goes low. We can see that the sample-and-hold circuit first outputs $V_{CM} + V_{OS}$. Then after it samples V_{DAC}, it updates to $V_{DAC} + \Delta V_W$ when V_{TP} transitions. The large rise time of V_{TP} is due to its large series resistance. In actual operation, the rise/fall times are those of the normalizer.

In this design, the op-amp consumes 40 nA from the 0.6 V supply. Both C_1 and C_2 are 500-fF capacitors. The speed of the sample-and-hold circuit is mainly dominated by its slew rate, because the op-amp is driven to large-signal swings in response to most voltage differences. The slew rate is about 30 mV/μs.

3.3.4 System Functionality

Test Setup

Figure 3.37 shows the system-level test setup. The FPGA or PC controls Keithely 2612 Source-Meter Units (SMUs) to generate input currents to the chip-on-the-board (COB), programs the scan bits, as well as generates control signals. The two crucial control signals during learning are Φ and Φ_K: Φ controls the timing of the sample-and-hold circuit, while Φ_K controls the multiplexer to select either VX_DAC or VS_DAC to the V-I converter, as shown in Figure 3.10.

During the learning phase, ADCX$\langle 1:6\rangle$ and ADCS$\langle 1:6\rangle$ sample chip outputs from ports VX_OUT$\langle 1:6\rangle$ and VS_OUT$\langle 1:6\rangle$, respectively. These values are stored in the FPGA/PC and are then fed back through DACX$\langle 1:6\rangle$ and DACS$\langle 1:6\rangle$ to ports VX_DAC$\langle 1:6\rangle$ and VS_DAC$\langle 1:6\rangle$, respectively.

Figure 3.38 shows the timing diagram of some important signals for the k-th learning trial. During Phase I, currents corresponding to sensor resistance $\mathbf{R}^k(c_1)$ are applied to IIN$\langle 1:6\rangle$. Then signals from VX_OUTs (i.e. $\mathbf{V_X}^k(c_1)$) are acquired by ADCXs, stored in the FPGA/PC, and fed back to DACXs for the computation in Phase II.

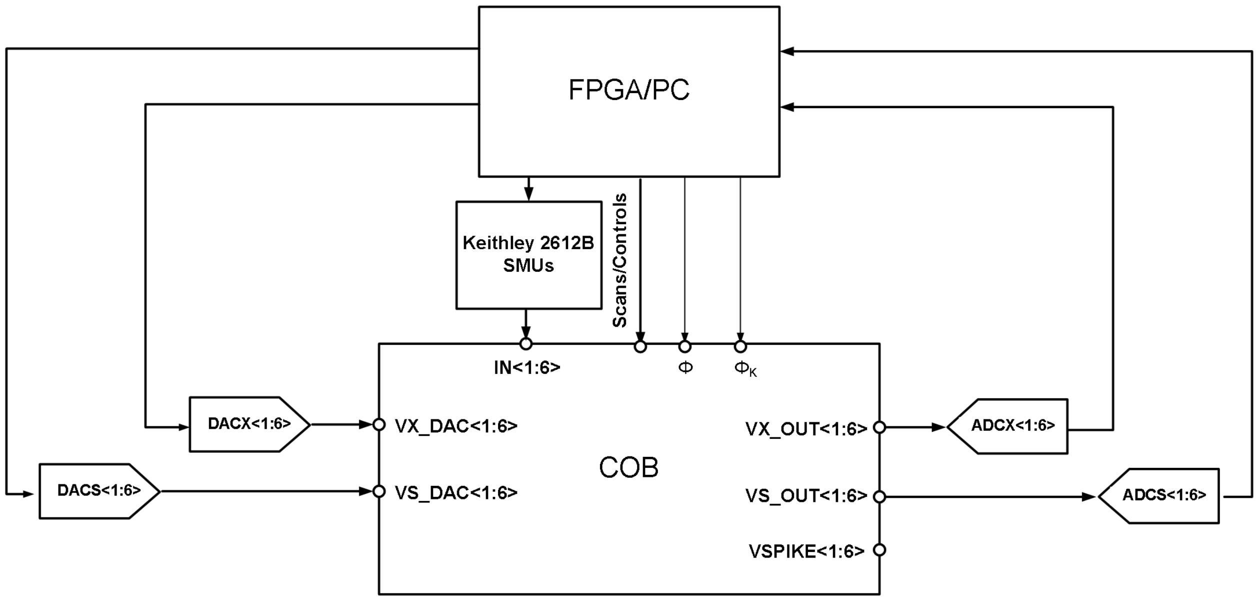


Figure 3.37: Block diagram of the system-level test setup.

During Phase II, currents corresponding to $\mathbf{R}^k(c_2)$ are applied. In Phase IIa, Φ_K is low, and V_K's are connected to VX_DAC's (i.e. $\mathbf{V_X}^k(c_1)$). In Phase IIb, Φ_K is high, and V_Ks are connected to VK_DACs (i.e. $\mathbf{V_S}^k$). Φ transitions before Φ_K such that sample-and-hold circuits can update $\mathbf{V_S}$ and output to VS_OUTs. These values are acquired by ADCSs, stored in the FPGA/PC, and fed back to DACSs for the computation in next trial. It can be seen that VS_DAC is updated at the end of the trial.

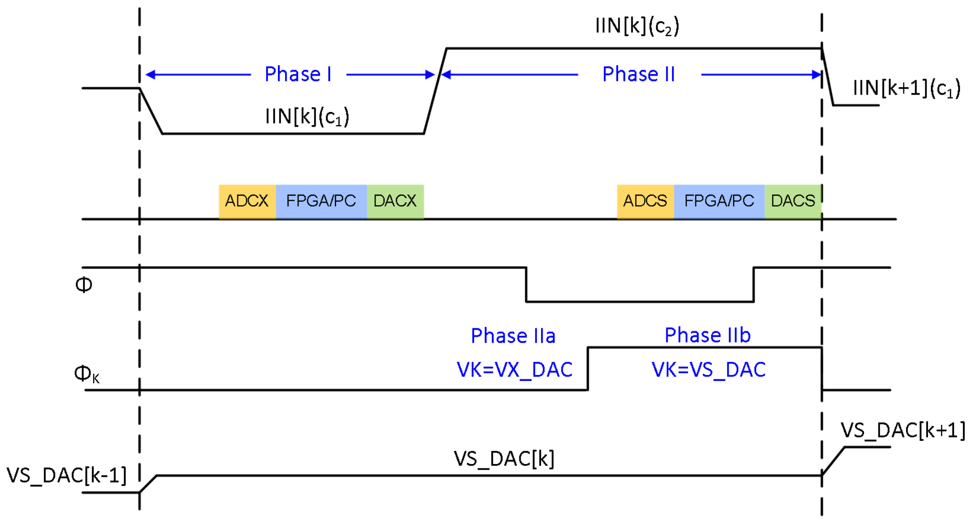


Figure 3.38: Timing diagram for one learning trial.

Figure 3.39 shows the testing setup. The chip is bonded to the daughter board. Two mother boards generate all the bias/supplies, as well as buffer signals. The outputs of the chip are connected to a multi-channel ADC board, stored in PC, and fed back to a multi-channel DAC board.

The input currents are provided by three Keithley 2612B 2-Channel SMUs. Figure 3.40 shows the input current excitations for the entire functionality test, including learning and feature extraction. During learning, in addition to the six analyte responses, two auxiliary responses are applied to help convergence of the algorithm. For each analyte, two current levels corresponding to sensor responses for the gas analyte at two concentration levels c_1 and c_2 are applied. After the learning phase, the same set of current excitations for the six analytes are applied again to verify the concentration-invariant spike encoding.

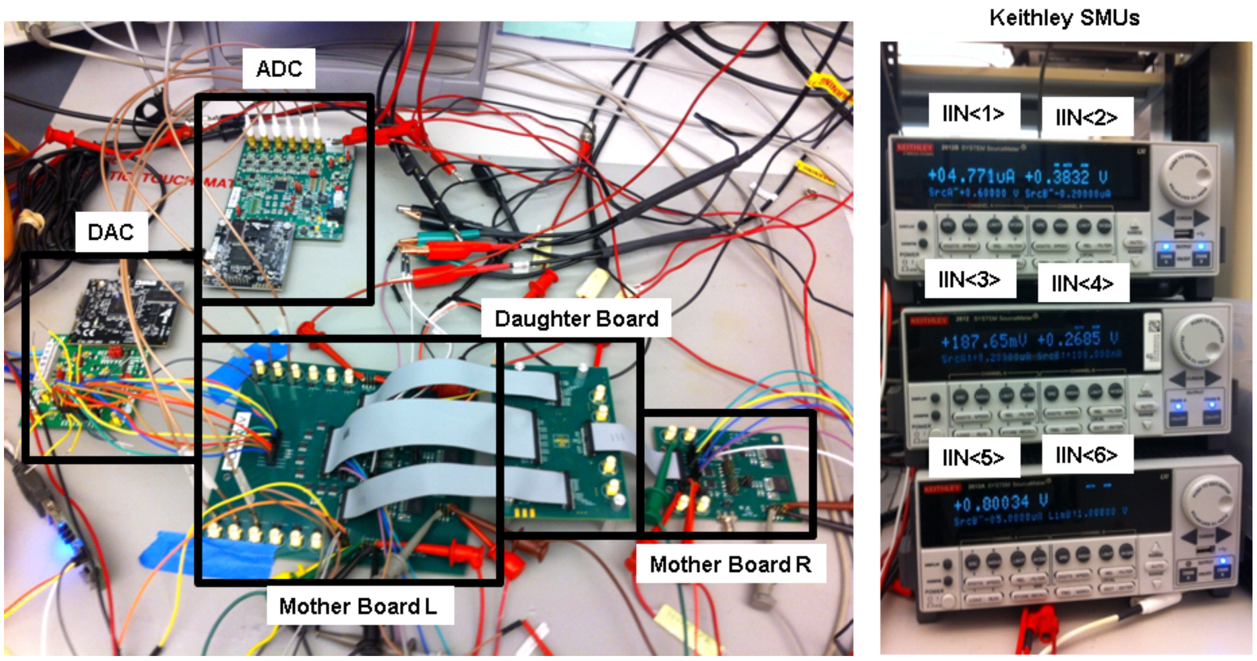


Figure 3.39: The testing setup.

State Evolution

Figure 3.41 shows the simulated state evolution of $\mathbf{V_S}$ from an initial state. The state converges around the desired final state (0.19, 0.19, 0.19, 0.19, 0.19, 0.19) V. Figure 3.42 shows the simulated final states (in black) from 64 initial states (in megenta). These 64 initial states occupy the vertexes of the hypercube $\{0.148, 0.234\}^6$ V. All of them converge around the desired state marked in circle.

The above simulations are for the case where devices are perfectly matched. However, in reality, device variations will lead to computations deviated from the ideal results. Therefore, we furthered invested the effect of device mismatches on the convergence behavior.

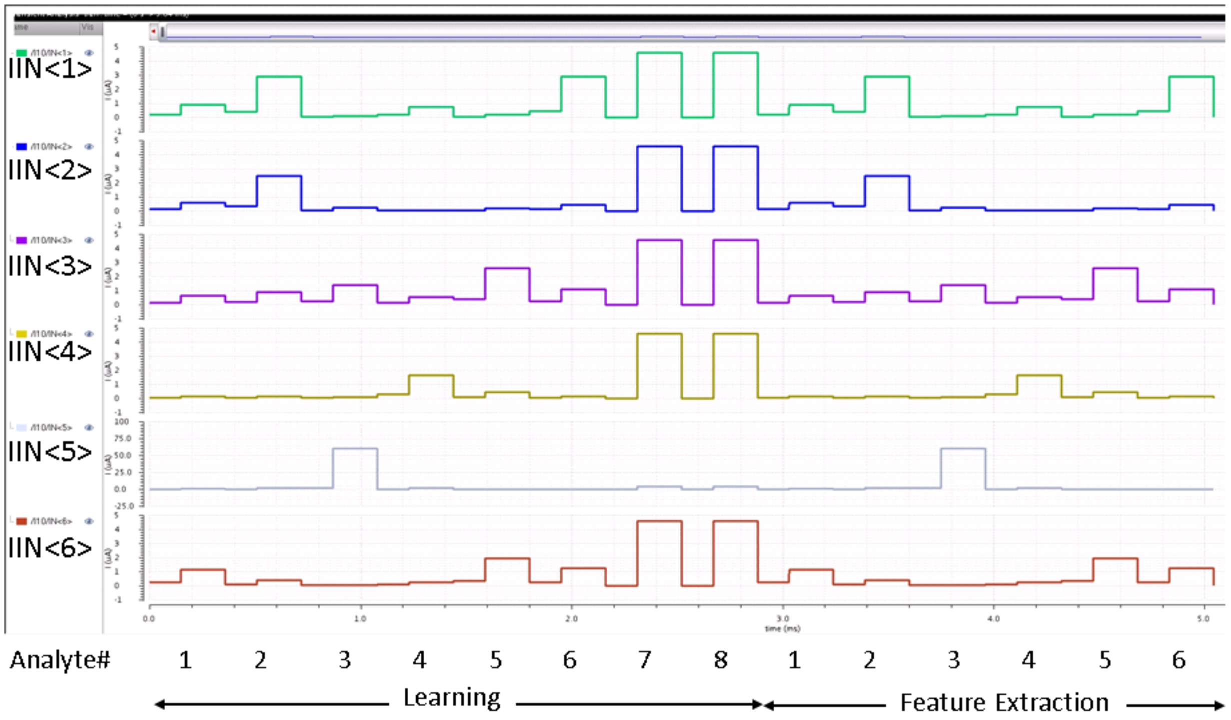


Figure 3.40: The input current excitations.

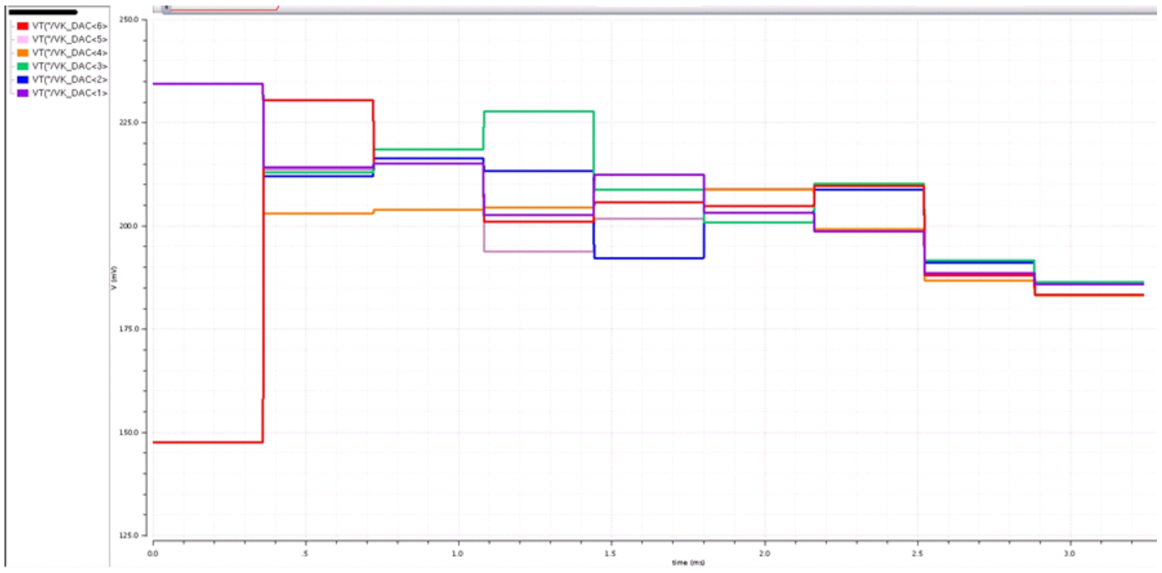
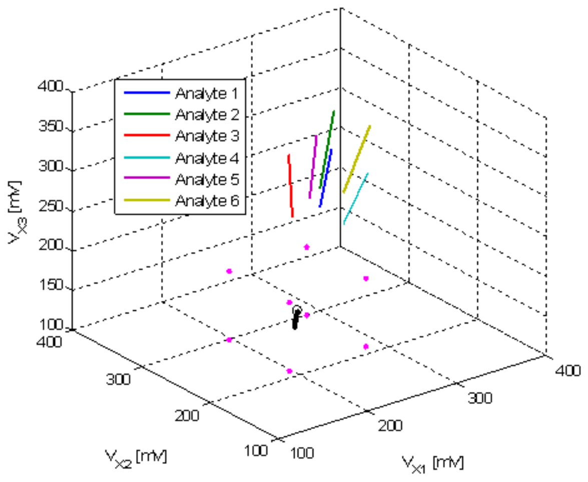
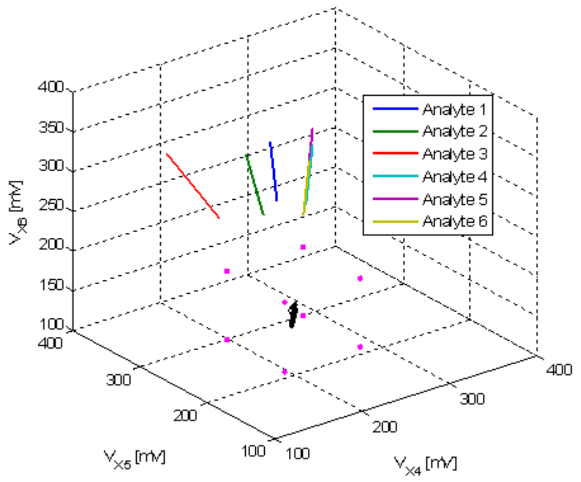


Figure 3.41: Simulated state evolution of $\mathbf{V_S}$ from an initial state.



(a) 3D view of the space V_{X1}-V_{X2}-V_{X3}



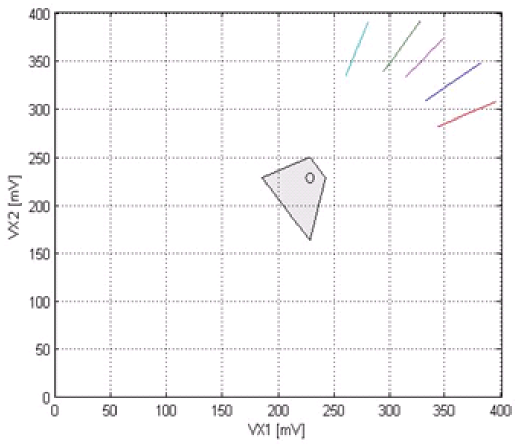
(b) 3D view of the space V_{X4}-V_{X5}-V_{X6}

Figure 3.42: Simulated final states (in black) from 64 initial states (in megenta) of $\mathbf{V_S}$.

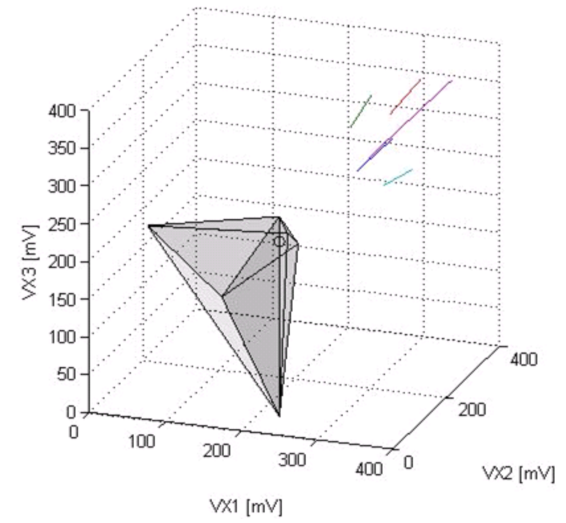
Firstly, we tried to understand how much the final state of $\mathbf{V_S}$ should be close to the ideal final state. We ued a program to find out a range of $\mathbf{V_S}$ such that at least four concentration levels are discriminable for each analyte in the form of output spike patterns. We started from the case with only the first two sensors, and gradually increased the number of sensors. It was found that to resolve the same number of concentration levels for each analyte, the tolerable range of final state $\mathbf{V_S}$ increases as the number of sensors increases, as shown in Figure 3.43. As N increases to 6, the allowable range for V_S in each dimension increases to ± 25 mV. This echoes Section 2.3: as the degrees of freedom increase, the minimum required precision per degree of freedom decreases. Therefore, the analog circuits can be designed in the low power and moderate-precision region.

After identifying the tolerable range of $\mathbf{V_S}$, we can investigate the effect of device variations on the convergence behavior. To avoid the daunting Monte Carlo simulation of the entire system for all initial states, we used the method described in Appendix B to simulate the final states of $\mathbf{V_S}$ from 729 initial states $\{0.15, 0.19, 0.23\}^6$, as shown Figure 3.44. All the simulated initial states arrive within the tolerable range of $\mathbf{V_S}$.

Figure 3.45 shows the measured state evolution of $\mathbf{V_S}$ from one initial state. The ideal final state is marked as bars at the right with different colors for different channels. Channel 1 has the largest deviation (36 mV) from the ideal state. Besides the residual errors from the chip after calibration, this error is mostly attributed to the measurement errors. For example, the off-chip ADCs and DACs have different reference voltages and gains, causing a discrepancy between the DAC outputs and the inputs to ADCs. Accumulating this error for eight iterations results in the deviation from the ideal state. The measurement accuracy can be improved by setting up an automatic pre-measurement calibration that characterizes

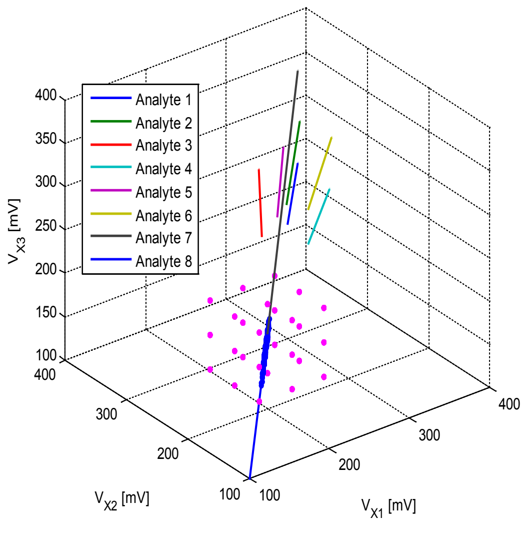


(a) $N = 2$

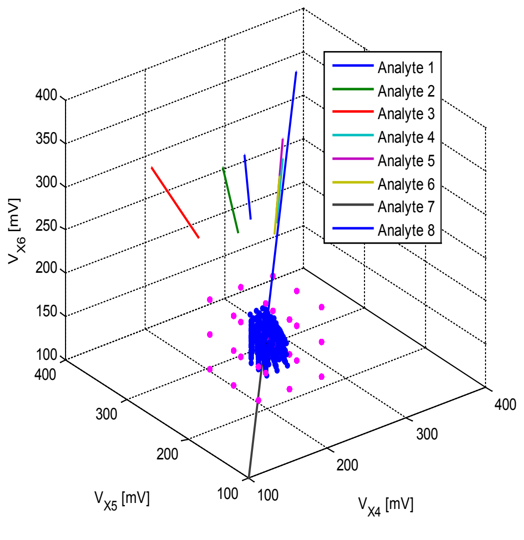


(b) $N = 3$

Figure 3.43: The tolerable range of final states $\mathbf{V_S}$ with different number of sensors N.



(a) 3D view of the space V_{X1}-V_{X2}-V_{X3}



(b) 3D view of the space V_{X4}-V_{X5}-V_{X6}

Figure 3.44: Simulated worst-case final states (in blue) from 729 initial states (in megenta) of $\mathbf{V_S}$ considering device variations.

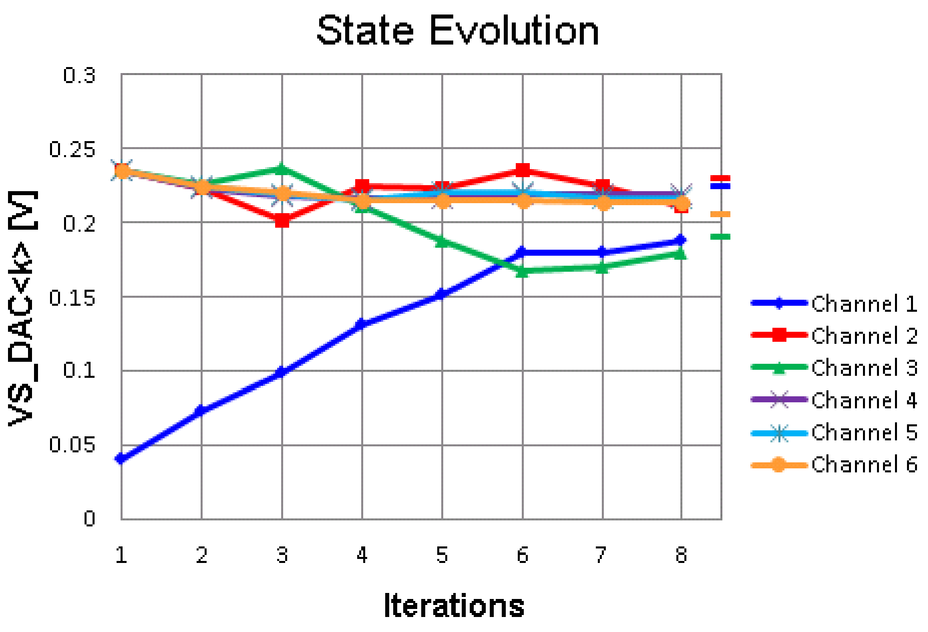


Figure 3.45: Measured state evolution from an initial state of $\mathbf{V_S}$.

the ADCs and DACs transfer functions to cancel their offsets and gain errors.

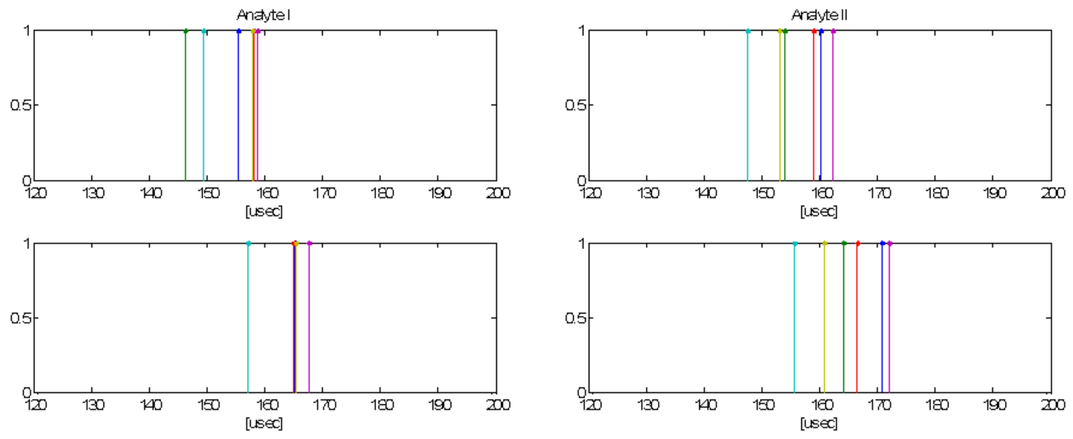
Spike Patterns

Figure 3.46 shows the measured output spike patterns. For each analyte, input currents corresponding to the sensor responses at two concentration levels are applied. All of the spike delays are measured with respect to the transition of the enable signal (ENB). Except for Analyte I, a unique spike pattern can be observed for each analyte. The difference between the spike patterns at different concentrations can be considered as the input noise for the spike pattern recognizer.

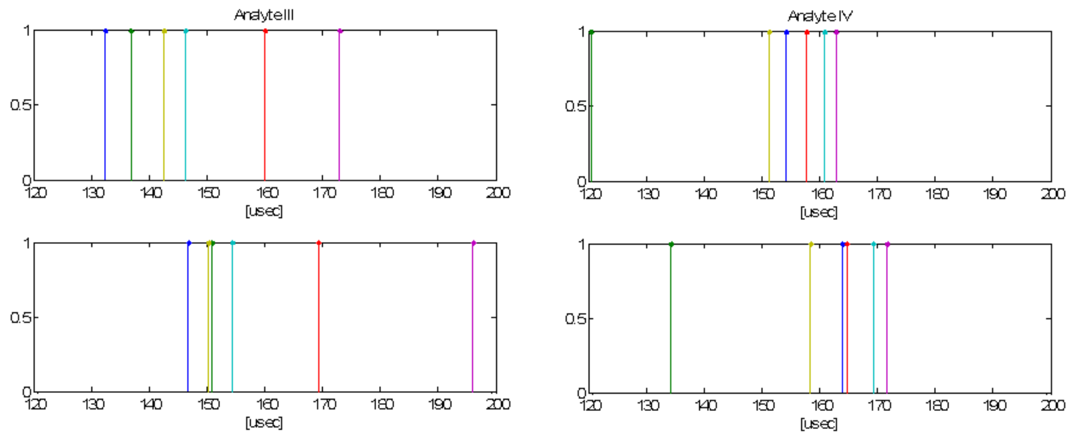
3.3.5 Summary

To summarize, the analog front end implements a novel trainable feature extraction algorithm for metal-oxide gas sensor arrays. The algorithm extracts one composite feature of all analytes and transforms the sensor responses into concentration-invariant spike patterns. This composite feature is extracted by performing the gradient decent algorithm during training. This 6-channel analog frond end consumes 519nW/channel in the training mode, and 463nW/channel in the normal mode.

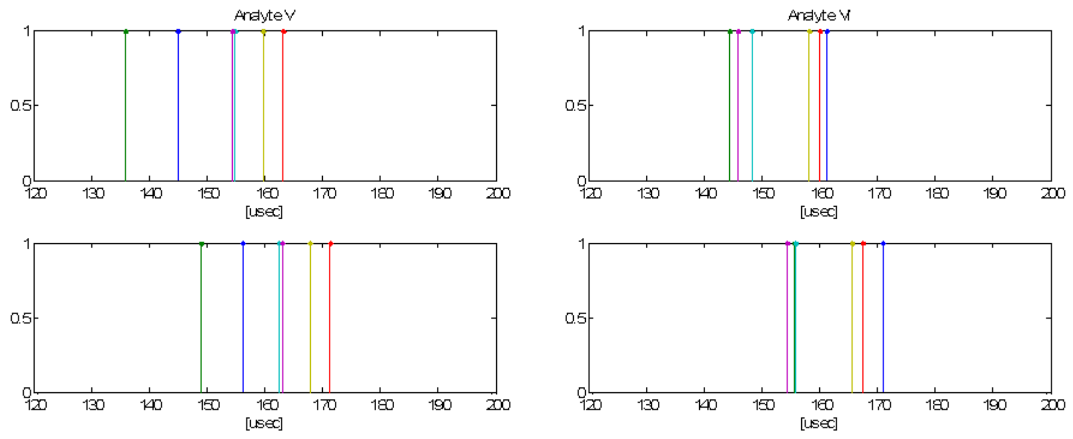
The spike-timing encoding and the calibration allow the design of transistor sizes to be small and therefore lower current consumption. However, the system will eventually be limited by noise. Table 3.1 shows the simulated total noise at the output of the second



(a)



(b)



(c)

Figure 3.46: Measured output spike patterns. Each analyte is presented at a low and a high concentration levels.

log converter and the gain from the input. The channel precision is calculated respect to the largest input current (4 μA). The system precision is greater than 4.66 bits/channel. Figure 3.47 shows the die photograph. The system performance is summarized in Table 3.2.

Table 3.1: Input referred channel precision at different input currents.

IIN[A]	Gain[V/A]	Total Output Noise [μV]	Input Referred Noise Current [nA]	Channel Precision [bits]
40 n	1.678 M	749	0.44	13.15
400 n	124 k	607	4.89	9.68
4μ	3.33 k	529	158	4.66

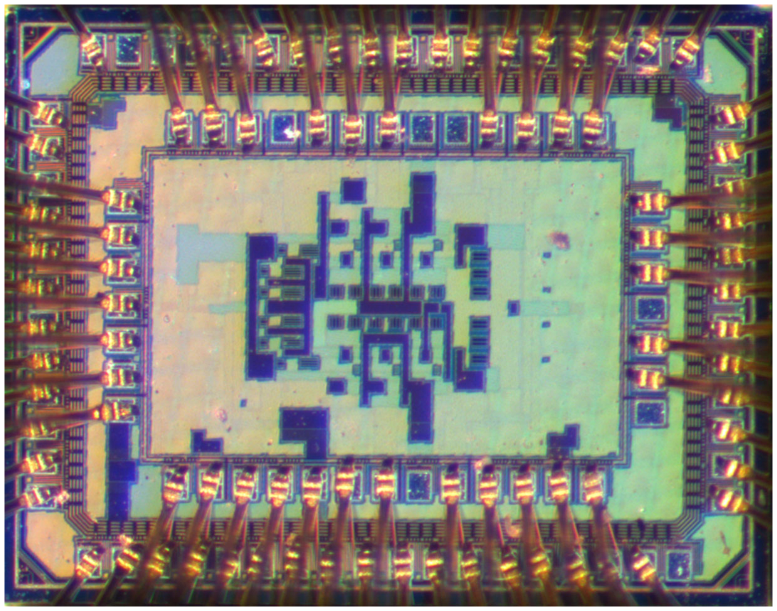


Figure 3.47: Die photograph.

Table 3.2: Performance summary of the analog gas sensing front end.

Supply	0.6/0.65 V
Power Consumption	519 nW/channel (learning) 463 nW/channel (normal)
System Precision	>4.66 bits/channel (input referred)
Operating Speed	360 μs/trial cycle

As a final remark, in this design, we have demonstrated an integrated sensor signal processing frontend with a learning mechanism to adapt itself to the input statistics. This is essential for processors at the sensor nodes that need to accommodate changing inputs and environments. Moreover, we used subthreshold analog circuits to directly implement functions such as logarithmic transforms and normalization. The low-power and low-precision regime of analog circuits is exploited by using distributed signal processing and calibration.

Chapter 4

Spike Pattern Classifier in Advanced Nanotechnologies

After the analyte features are extracted and represented in spike patterns. The next step of processing is to store these patterns together with their associative identities, and later to retrieve identities when patterns are presented. As we have seen in Chapter 2, after the olfactory sensory frontend converts the odor information into the activation patterns of glumeruli in the olfactory bulb, the patterns experience a dimensionality expansion when entering the cortex. It has long been hypothesized that higher levels of the brain computes with high-dimensional vectors [85], [86], [30], [31]. In this chapter, the hyper-dimensional computing paradigm is introduced. Next, a spike pattern classifier based on hyper-dimensional computing is proposed. It consists of a transformation of the input spikes into high-dimensional sparse vectors and an associative memory model. The transformation uses a random sampling scheme that can be efficiently performed with circuits exhibiting large parametric variations. An associative memory model is used to perform fast and efficient storage/retrieval of sparse vectors. Such computing paradigm could especially be a perfect match to the emerging 3D nanotechnologies. Finally, the implementation of this spike pattern classifier in a 3D CNFET-RRAM technology is described.

4.1 Hyper-Dimensional Computing

No two brains are identical, yet they can produce the same behavior, i.e. they can be functionally equivalent. This means that brains with different hardwares accomplish the same computing. Also, the functions of the brain is highly tolerant of component failures. Electrical recording from neurons shows that even seemingly simple mental events involve simultaneous activity of widely dispersed neurons, suggesting that large circuits are fundamental to the brain's computing. In [30], [31], it was proposed that the brains have leveraged some subtle properties in the hyperspace to achieve this.

Figure 4.1 shows the normalized distance histogram of randomly and independently generated binary vectors in N-dimensional spaces. As N increases from 10 to 10,000, the normalized distance approaches 0.5. This means with overwhelming probability, vectors randomly drawn from a hyperspace have the same distance to each other.

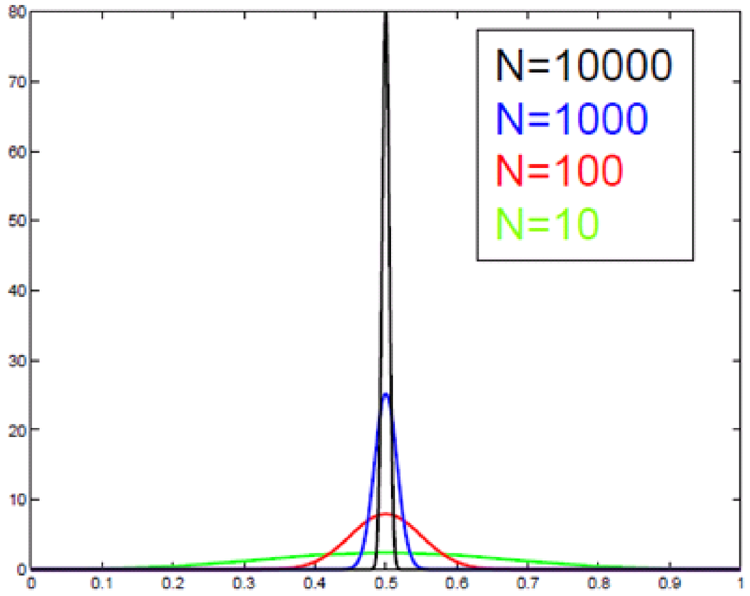


Figure 4.1: Normalized distance histogram of binary vectors in N-dimensional spaces.

The Law of Large Numbers governs this behavior. Let $\mathbf{x} = (x_1, x_2, \cdots, x_N)$, where $x_1, x_2, \cdots, x_N$ are i.i.d. Bernoulli random variables with probability of 0.5 to be 0 or 1. The normalized distance d between any randomly and independently generated vectors $\mathbf{x}_1$ and $\mathbf{x}_2$ is

$$d = \frac{|\mathbf{x}_1 - \mathbf{x}_2|}{N} \tag{4.1}$$

$$= \frac{\sum_{i=1}^{N} |x_{i1} - x_{i2}|}{N} \tag{4.2}$$

Since the mean μ of the random variable $y = |x_1 - x_2|$ is 0.5, the Strong Law of Large Number guarantees that

$$p\left(\lim_{N\to\infty} \frac{\sum_{i=1}^{N} |x_{i1} - x_{i2}|}{N} = \mu\right) = 1 \tag{4.3}$$

That is, as N approaches infinity, the normalized distance d converges to $\mu = 0.5$ almost surely.

This property has an important implication to the functional equivalence of two different implementations. It was suggested if a system builds its model of the world from random patterns [31], i.e., by starting with vectors drawn randomly from the hyperspace, then it's possible to achieve functional equivalence from different random origins. It's because the equivalence of systems no longer lies in the actual patterns, but lies in the relation of the patterns to one another in the system.

Consider a random sampling scheme as shown in Figure 4.2. This was originally proposed as a way to transform patterns into sparse vectors in the hyperspace [30] and is applied here to our spike pattern recognition problem. Suppose there are N gas sensors, then the output spike pattern from the front end consists of N spikes. The large circle represents the entire N-dimensional space spanned by the N spike delays $t_1, t_2, \cdots, t_N$. Each point represents a vector $\mathbf{t}$, or a particular spike pattern. In this space, there are M samplers, represented by dotted circles. They are randomly generated from a uniform distribution over the entire space. Each sampler has a reception range. Whenever a spike pattern falls in its reception range, it gets activated and outputs one ($y_m = 1$). Otherwise, it outputs zero ($y_m = 0$). The probability p that a sampler is activated is

$$p = \frac{V_{sampler}}{V_{space}} \tag{4.4}$$

where $V_{sampler}$ is the volume of the sampler, and V_{space} is the volume of the entire space.

If the spike patterns are also uniformly distributed in this space, then the process of transforming a spike pattern $\mathbf{t}$ into the activation pattern of the samplers $\mathbf{y}$ is the same as the process of randomly and uniformly drawing a binary vector $\mathbf{y}$ in the M-dimensional space, where each element of $\mathbf{y}$, y_m, is is a Bernoulli random variable with probability of p to be 1. It can be shown that, the mean of the inner product between any two activation patterns $\mathbf{y}_1$ and $\mathbf{y}_2$ is

$$E[\mathbf{y}_1^T \mathbf{y}_2] = p^2 M \tag{4.5}$$

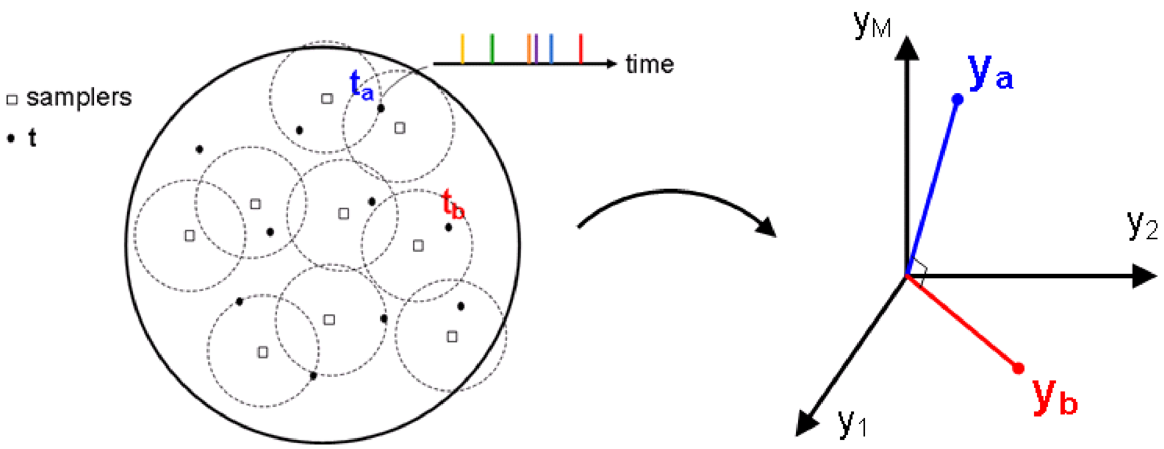
Again, the Strong Law of Large Numbers guarantees that

$$p\left(\lim_{M\to\infty} \mathbf{y}_1^T \mathbf{y}_2 = p^2 M\right) = 1 \tag{4.6}$$

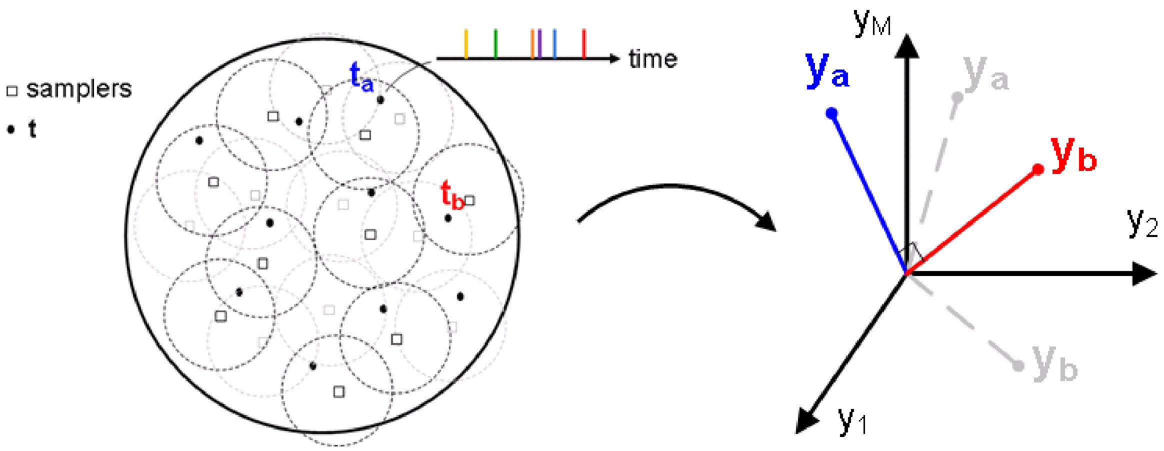
Therefore, the inner products between any two activation patterns is approximately $p^2 M$, when M is large. If we can judiciously choose the activation probability p and the number of samplers M such that $p^2 \ll M$, then any two activation patterns would be orthogonal to each other, as shown in the right of Figure 4.2.

However, the most important property of the random sampling is how it can achieve functional equivalence. As shown in Figure 4.2b, if another set of randomly generated samplers are used to transform the spike patterns, any output vectors will still be orthogonal to each other. Although they may be different vectors, their relation to each other (orthogonality) is still preserved. This is similar to the generation of random sensing matrices in compressed sensing [87]. This also has a very important implication to the implementation of a system in that the functions can be reliably performed with components having large variations[1]. Since the system level reliability is not achieved by component-level reliability, components can be designed in the low-power and low-precision region.

Now that the output patterns are mutually orthogonal, a pattern classifier whose operation involves projection will work.



(a)



(b)

Figure 4.2: Random sampling.

[1]The variations that we deal with here are process variations. For temporal variations, periodic retraining is required.

4.2 Spike Pattern Classifier

The proposed spike pattern classifier consists of a sparse vector generator and an associative memory. First, their mathematical models are described. Next, the behavior simulation is given.

4.2.1 Sparse Vector Generator

Figure 4.3 illustrates how to transform spike patterns into sparse, high-dimensional and orthogonal vectors using the random sampling scheme. This transformation is inspired from the proposed spatio-temporal integration network for olfactory processing [88], [89], [90], [91]. Again, suppose there are N sensors, then the output from the feature extraction frontend contains N spikes with a delay vector $\mathbf{t} = (t_1, t_2, \cdots, t_N)$. The transformation is performed by M sampling groups. Each group is composed of $n < N$ delay elements. Each delay element is (permanently) assigned a delay d_{mj} from random and connected to a randomly selected channel. There is a coincident detector at the output of each group. The coincidence detector outputs 1 ($y_m = 1$) if the output spikes align, as shown in Figure 4.4.

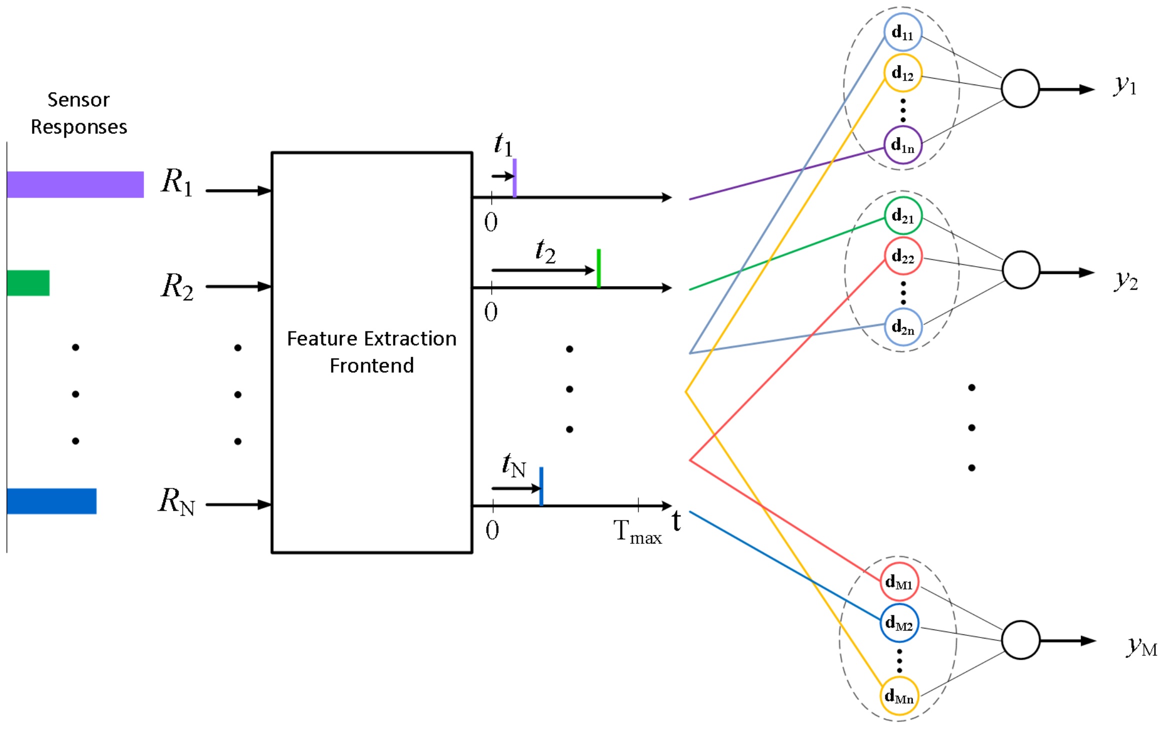


Figure 4.3: Sparse vector generation using random 'sub'-sampling of the input spikes.

Compared to the address matrix in the Sparse Distributed Memory (SDM) where each sampler connects to the full dimension of the input vectors [30], here each sampler only

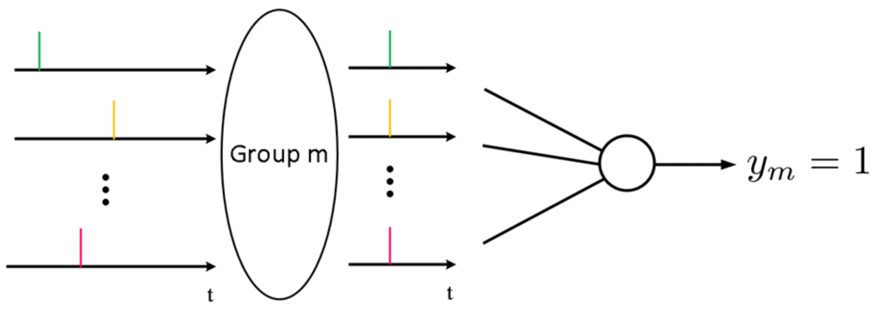


Figure 4.4: The coincidence detector outputs 1 when output spikes align.

connects to a subset of the input dimension and different samplers connect to different subsets. This scheme is derived from one variation of the SDM model [92].

How do these samplers look like in the N-dimensional space of spike delays? In Appendix C, the reception range of the spike sampling groups is derived. It is shown that each sampling group corresponds to a hyperplane. The orientation of the hyperplane is determined by the subset of selected channels and the delays in the group. Figure 4.5 shows the geometric interpretation of the spike sub-sampling scheme. In this case, the number of sensors N is 3, the number of elements per sampling group n is 2, and only three sampling groups are shown for clarity. Each blue point represents a spike pattern. Whenever a spike pattern falls on a hyperplane, the group corresponding to that hyperplane gets activated. It seems that the probability of falling onto a hyperplane is zero. However, in reality, each coincidence detector has a detection window such that spikes arrive within the window are considered 'aligned'. Therefore, the reception range of each sampling group is actually the union of a set of hyperplanes. The probability of falling into this range is then nonzero.

In summary, since the samplers are uniformly and randomly generated and M is large, the inner product between any two output activation vectors $\mathbf{y}_1^T\mathbf{y}_2$ converges to $E[\mathbf{y}_1^T\mathbf{y}_2] = p^2M$. Since the probability p that a spike pattern falls withing the reception range of a sampler is very small, then $\mathbf{y}_1^T\mathbf{y}_2 \approx 0$. Therefore, the output vectors would be mutually orthogonal.

4.2.2 Correlation Matrix Memory

Correlation matrix memory (CMM) [93] is one associative memory model. Similar to other types of associative memories, it can stores and retrieve patterns effectively when the patterns are sparse and orthogonal. As in [30], the unnormalized version of CMM is used here. The combination of the random sampling and an unnormalized CMM resembles the cortex of cerebellum [30].

Figure 4.6 shows the memory structure. During the storage phase, there are two inputs to the memory. One is the activation pattern it needs to recognize, $\mathbf{y}$, and the other is the word associated with the pattern $\mathbf{w}$. The word vector $\mathbf{w}$ consists of -1s and 1s and has a dimension of U. U can be arbitrary as long as $U \geq log_2K$, where K is the number of analytes. However, as will be explained shortly, larger U can make the retrieval more effectively. These words are

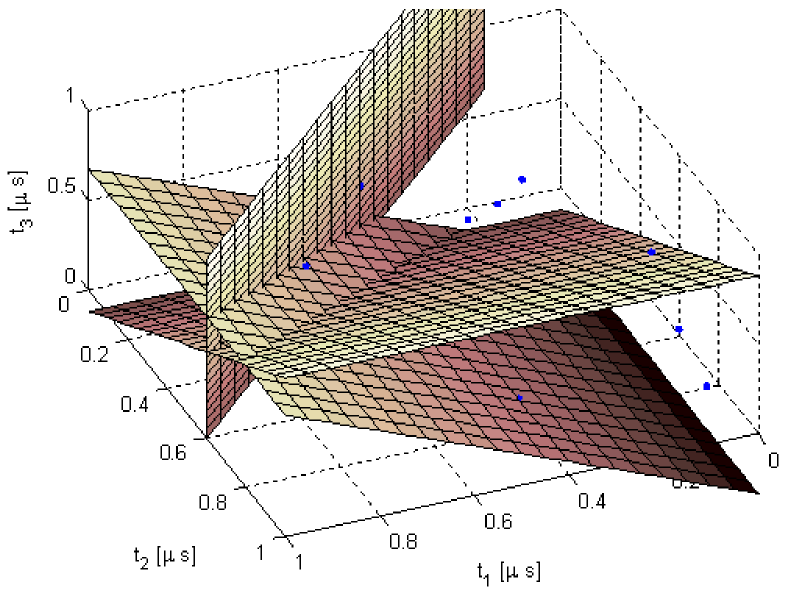


Figure 4.5: The geometric interpretation of the random spike sub-sampling. In this case, $N = 3$, $M = 3$, and $n = 2$.

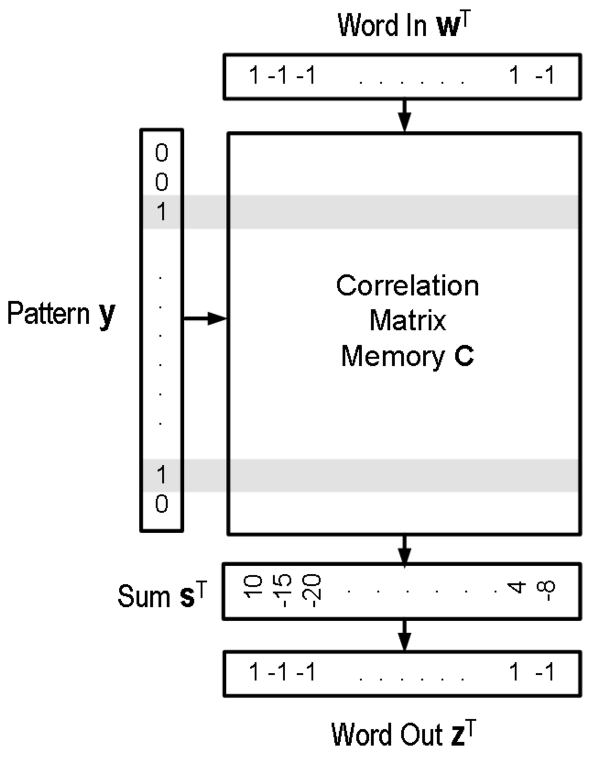


Figure 4.6: Structure of the unnormalized correlation matrix memory.

the identities of analytes and can simply be randomly generated vectors. The memory stores these patterns and words by accumulating their outer products on the memory elements (Equation 4.7). This is why it's named the correlation matrix memory.

$$\mathbf{C} = \sum_{k=1}^{K} \mathbf{y}_k \mathbf{w}_k^T \tag{4.7}$$
$$= \mathbf{Y}^T \mathbf{W} \tag{4.8}$$

Storing the outer products of $\mathbf{y}$'s and $\mathbf{w}$'s is done by adding $\mathbf{w}$ to those rows corresponding to 1s in $\mathbf{y}$. Each memory element is an accumulator. Therefore when the other pattern activates the same row later, the new word will be added on top of the previous written words. This is unlike the conventional memory where new data overwrites old data.

During the retrieval, only $\mathbf{y}$ is present. The successful recognition will retrieve the correct word associated with the pattern. The first step of retrieval is projection. The input pattern $\mathbf{y}_r$ is projected to the correlation matrix $\mathbf{C}$. In this way, we get a weighted sum $\mathbf{s}_r$ of words that have been written:

$$\mathbf{s}_r^T = \mathbf{y}_r^T \mathbf{C} \tag{4.9}$$
$$= \mathbf{y}_r^T \left(\sum_{k=1}^{K} \mathbf{y}_k \mathbf{w}_k^T \right) \tag{4.10}$$
$$= ||\mathbf{y}_r||^2 \mathbf{w}_r^T + \mathbf{y}_r^T \left(\sum_{k \neq r} \mathbf{y}_k \mathbf{w}_k^T \right) \tag{4.11}$$

The weights are the inner product of the input pattern and the stored patterns. Therefore, if $\mathbf{y}$'s are mutually orthogonal, these weights will be small. Furthermore, if $\mathbf{w}$'s are also mutually orthogonal, they can cancel out each other. This project can be done by summing up the rows activated by the input pattern.

The second step of retrieval is thresholding. The output word is obtained by taking the threshold of the sum vector at zero.

$$\mathbf{z}_r^T = z(\mathbf{s}_r^T) \tag{4.12}$$

where z is a thresholding function at zero. Since $\mathbf{y}$'s are mutually orthogonal, the sum is biased toward the word to be retrieved. Therefore, as long as $\mathbf{y}$'s are orthogonal, the retrieval will be successful, regardless of the exact patterns of $\mathbf{y}$'s. This is how functional equivalence can be achieved in hyper-dimensional computing.

4.2.3 Behavior Simulation

Given the mathematical models, the behavior of the spike pattern classifier is simulated. Figure 4.7 shows the simulation setup. The inputs are K spike patterns, each containing

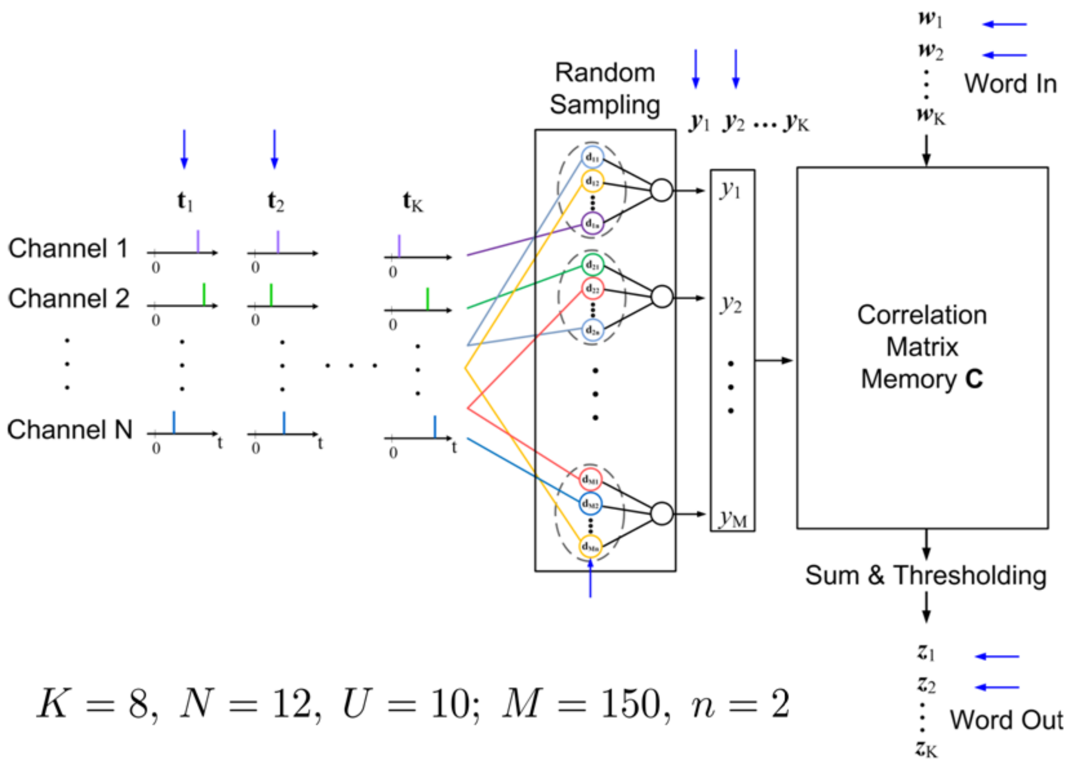


Figure 4.7: Simulation setup of the spike pattern classifier.

N spikes. These spike patterns are transformed into high-dimensional sparse vectors $\mathbf{y}$'s. During the training phase, each pattern $\mathbf{y}$ and its associative word $\mathbf{w}$ are written into the memory. During the retrieval phase, we observe how close the retrieved output word is to the written word when one of these K spike patterns is presented at the input.

In this simulation, 500 randomized trials were run. In each trial, the input spikes to the system are the same. But the delays of delay elements and the channels they are connected to are randomly assigned. The delay assignment is according to a distribution from a Monte Carlo simulation of a delay circuit; the channel assignment is according to a uniform distribution over N. For every trial, after storing all the K patterns and their associated words, all the K patterns are presented again and all the output words are compared to the written words.

In this setup, there are $K = 8$ odor analytes, $N = 12$ gas sensors, $M = 150$ sampling groups, and $n = 2$ delay elements per group.

As shown in Figure 4.8, strobing these randomized trials provides useful insights on how the sparse vector generator works. The bottom left plot shows the delays of the last delay elements over these 500 trials. It can be seen that it has huge variations. The right part of the figure shows the recorded patterns for the first and second analytes ($\mathbf{y}_1$ and $\mathbf{y}_2$) in trial 1, trial 2, and trial 500. In these plots, the vertical axis is the the index for the 150 sampling groups. A horizontal line at an index means the activation of that group. Although $\mathbf{y}_1$ and

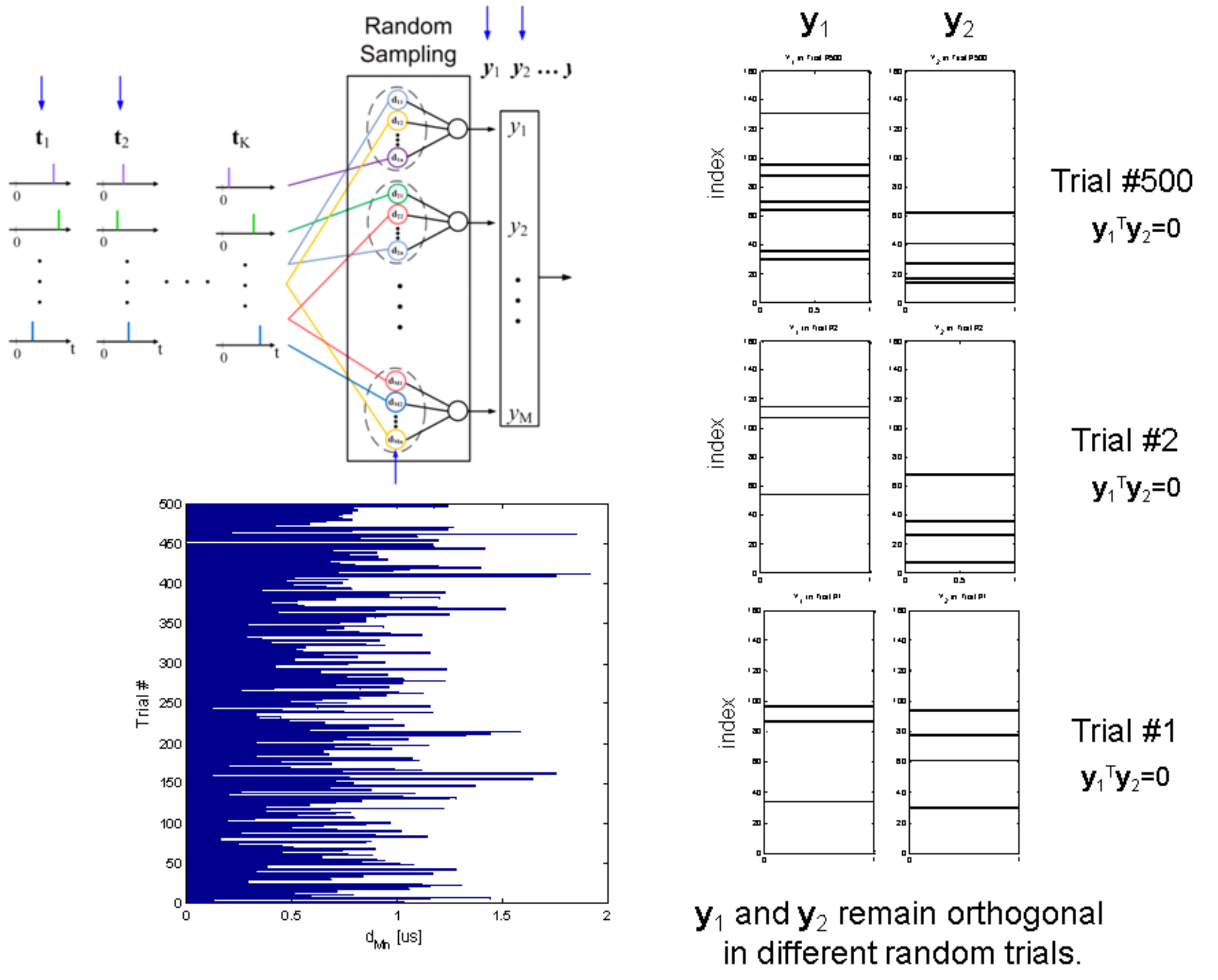
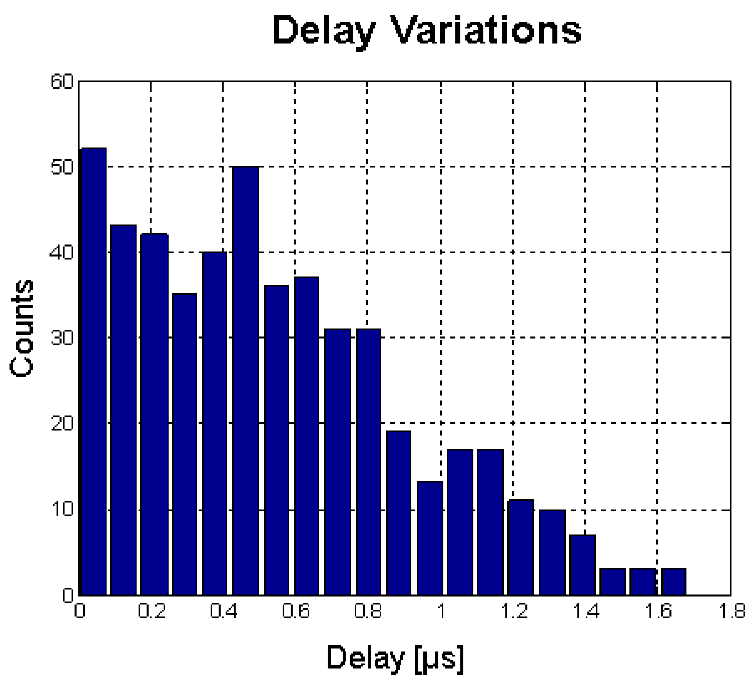


Figure 4.8: 'Strobing' the randomized trials.

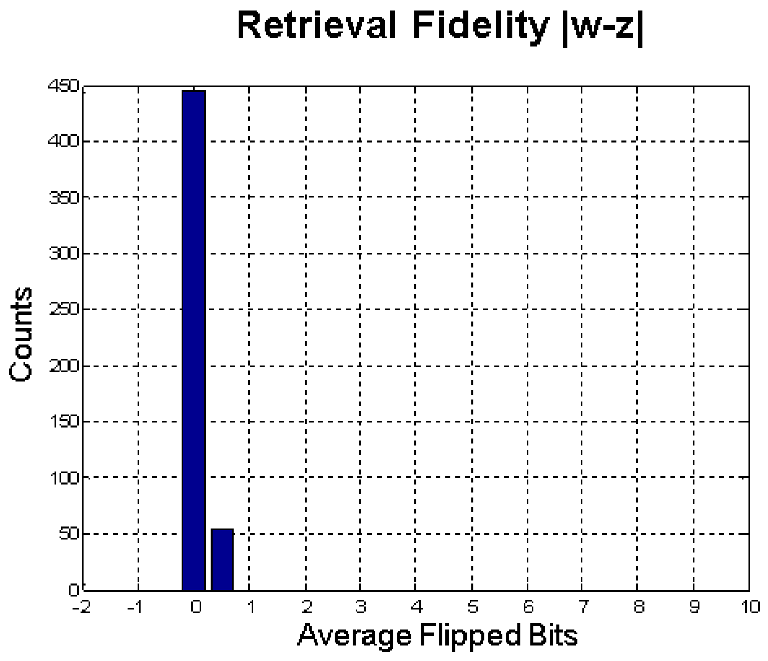
$\mathbf{y}_2$ look very differently across different trials, they remain orthogonal across trials.

Figure 4.9 shows the performance summary of the spike pattern classifier. Figure 4.9a is the delay distribution used in this simulation. They are obtained from the Monte Carlo simulation of a delay cell operating at a very low supply voltage (0.3 V). Because most device are minimum sized, the delay variation has a Fano facto close to 1 ($\mu = 533$ ns, $\sigma = 461$ ns).

Figure 4.9b shows the retrieval fidelity, which is the average flipped bits of the retrieved output words compared to the input words, averaged over the K words in each trial. We can see that, although the hardware devices have extreme statistical variations, this architectural can still perform recognition robustly and therefore have a peaking performance histogram.



(a) Delay variation of the delay cell



(b) Retrieval fidelity

Figure 4.9: Simulation results.

4.3 Implementation in Advanced Nanotechnologies

Hyper-dimensional computing could be a perfect match to the emerging 3D technologies. First, the computing architecture is either very difficult or impossible in traditional planar technologies, especially where logic and memory devices are separated. However, the emerging 3D nanotechnologies integrating large number of logic and memory devices makes its implementation possible. Second, as we have just seen, hyper-dimensional computing functions very robustly under large component variations. Therefore, the large variations and failure rates in the emerging devices are inherently taken care of in this computing paradigm. In this section, the monolithic 3D integration platform is first introduced. Next, the sparse vector generator in a 3D CNFET-RRAM technology is described. Finally the implementation of the correlation matrix memory using RRAMs is proposed.

4.3.1 The Monolithic 3D Integration Platform

This 3D integration platform is developed by our collaborators in Stanford University [24], [25], [26]. As mentioned in Chapter 1, CNFET can have an ordor of magnitude energy-delay-product (EDP) improvement compared to silicon CMOS. Such EDP benefits can also translate into cooling benefits for 3D ICs [94]. First, they have demonstrated monolithic 3D integration of inter-layer and fully-complementary CNFET logics, as shown in Figure 4.10. Instead of using Through-Silicon Vias (TSVs), Inter-Layer Vias (ILVs) are used to achieve high-density integration. A 3D metalic-CNT removal technique using local back-gates is proposed to overcome the diminishing gate control in higher layers of CNFETs [24].

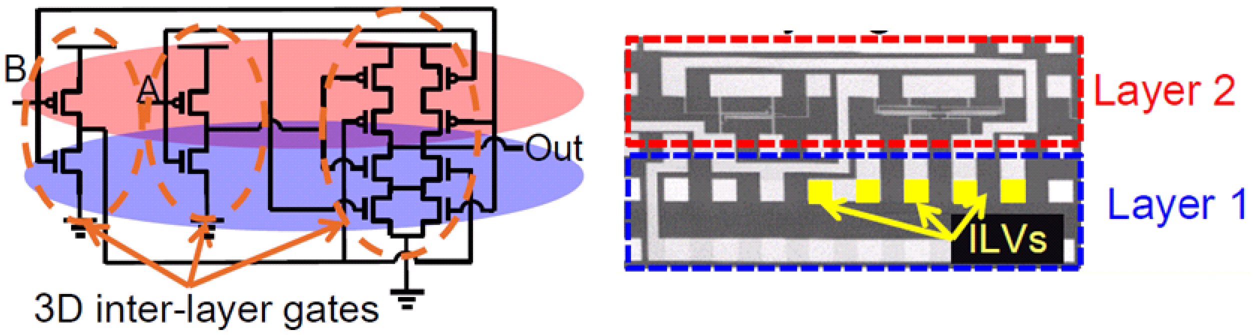


Figure 4.10: Monolithic 3D integration of inter-layer and fully-complementary CNFET logic gates. Adapted from [24].

Since silicon CMOS remain indispensable, 3D monolithic integration of CNFET and silicon CMOS FETs is also demonstrated. This is achieved by a low-temperature ($< 180°C$) CNFET fabrication processing which is compatible with silicon CMOS and can be integrated in back-end-of-the-line (BEOL), after silicon CMOS processing [25]. Figure 4.11 shows the correct operation of logic gates composed of both CNFETs and silicon CMOS FETs in different layers.

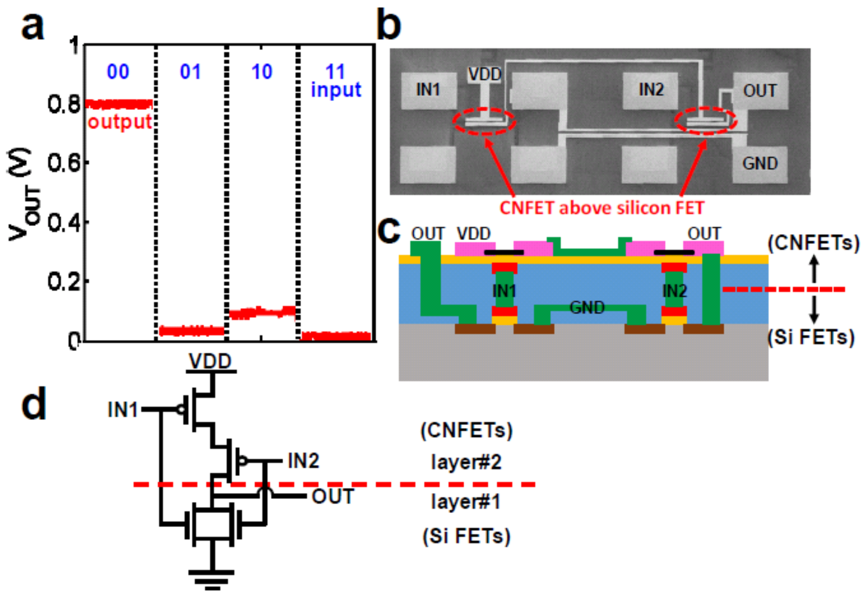


Figure 4.11: Monolithic 3D CNFET/silicon FET NOR2 gate. Adapted from [25].

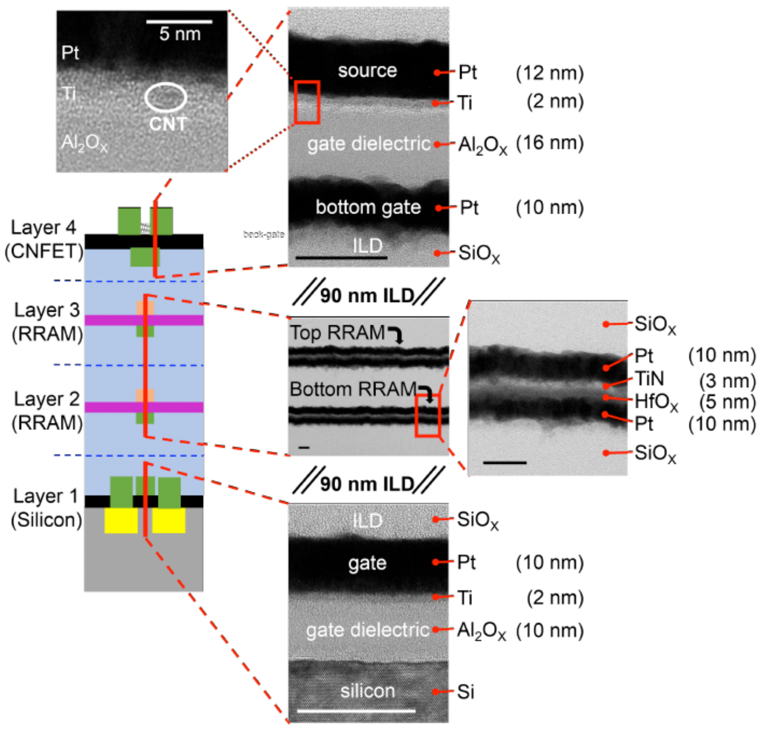


Figure 4.12: Transmission electron microscopy (TEM) images of four vertically stacked layers in the 3D integration platform. Adapted from [26].

Finally, to fully interweave logic and memory, they have successfully demonstrated integrating silicon CMOS FETs, RRAMs, and CNFETs monolithically in a 3D stack [26]. As shown in Figure 4.12, silicon FETs are on the bottom layer, followed by two layes of RRAM, followed by the top layer of CNFETs. This is likewise enabled by low-temperature RRAM processing [95].

In the following sections, the implementation of the spike pattern classifier in this technology is described. The sparse vector generator is implemented with CNFETs and RRAMs; the correlation matrix memory will be implemented with silicon CMOS FETs and RRAMs.

4.4 Sparse Vector Generator in CNFETs and RRAMs

Figure 4.13 shows the implementation of the delay cell and the coincidence detector. The delay cell consists of an RRAM and a CNFET inverter. A step timing detector consisting of an XOR gate and a D flip-flop is used as the coincidence detector.

The RRAM at the input determines the delay of the delay cell. Figure 4.14 shows the measured delays when the RRAM is in the high-resistance state (HRS) and the low-resistance state (LRS). The extracted resistance is 840 MΩ at the HRS and 1.4 kΩ at the LRS. To achieve a target delay variation of $\sigma/\mu > 1$, the required RRAM variation versus the average RRAM resistance is plotted in Figure 4.15a. The fabricated RRAMs have an average resistance of 11 MΩ and a variation of $\sigma/\mu = 3.5$, as shown in Figure 4.15b. Therefore, the delay cells can achieve a delay variation σ/μ greater than 1.

Figure 4.16 shows the operation of the coincidence detector. If the timings of the two input steps are close to each other, the output of the XOR gate is not able to trigger the D flip-flop, as shown in the left of the figure. However, if these two events are far away enough, the D flip-flop would be triggered. Therefore, the window that spikes considered aligned is the hold time of the D flip-flop. The D flip-flop is followed by an inverter to output 1 for signaling coincident events and 0 for non coincident events.

Figure 4.17 shows the cross-sectional view of the implementation. The first layer is for the RRAMs and the second layer is for CNFETs. They are densely connected through inter-layer vias.

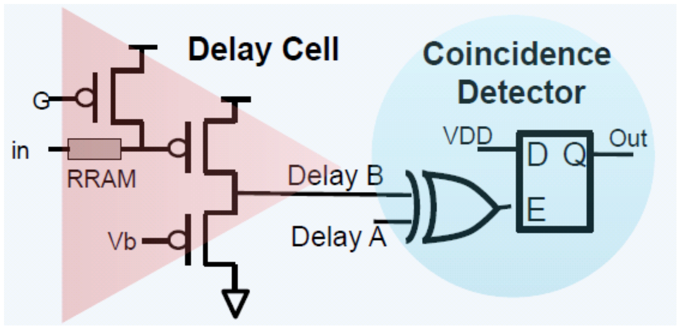


Figure 4.13: Implementation of the delay cell and the coincidence detector. Credit: Tony Wu.

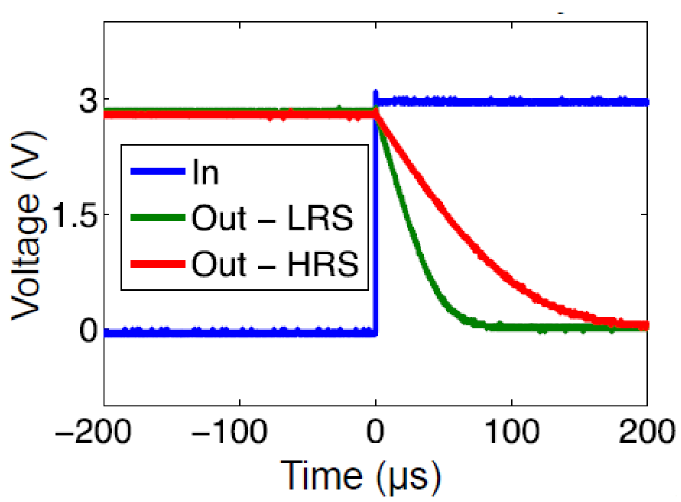
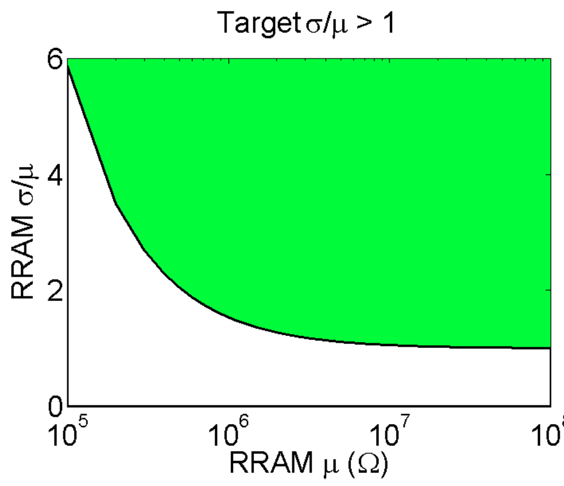
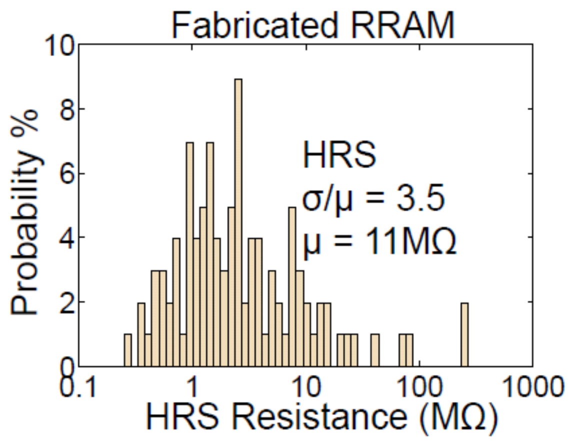


Figure 4.14: Measured output from the CNFET and RRAM delay cell. Credit: Tony Wu.



(a)



(b)

Figure 4.15: Required RRAM variation to achieve a delay variation of $\sigma/\mu > 1$. Credit: Tony Wu.

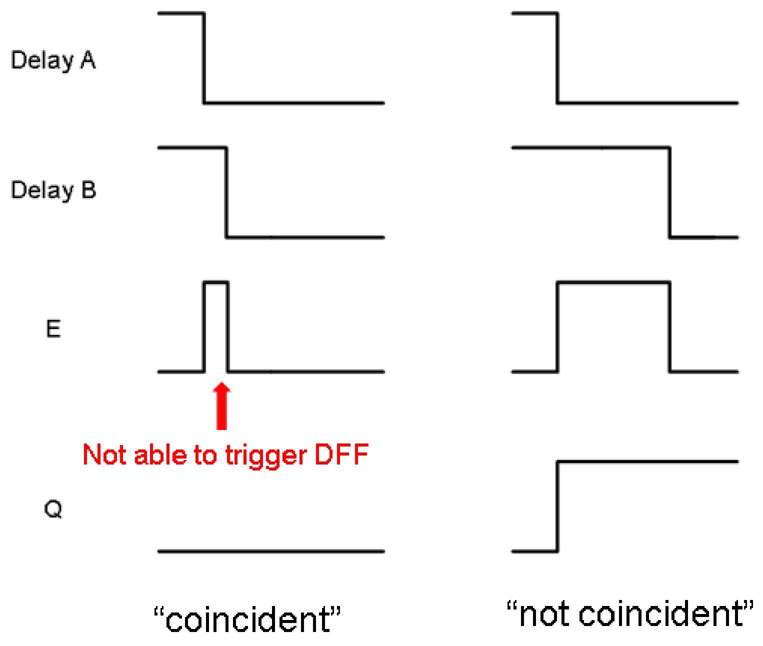


Figure 4.16: Operation of the coincidence detector.

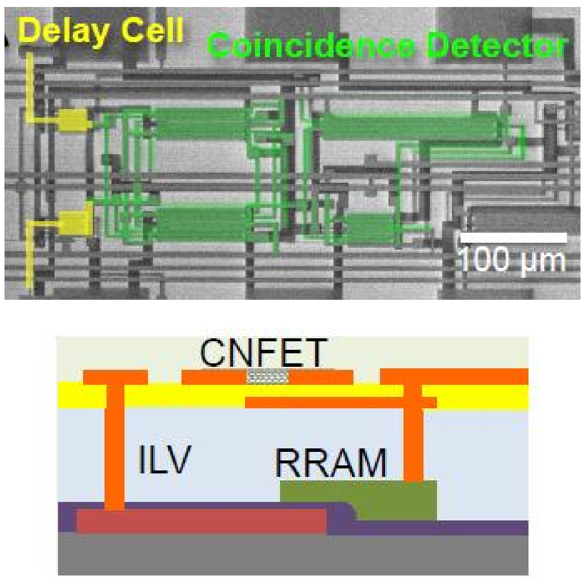


Figure 4.17: Cross-sectional view of the delay cell and the coincidence detector. Credit: Tony Wu.

4.5 Correlation Matrix Memory in RRAMs

4.5.1 The RRAM Model

The RRAM available to us is the Hafium-oxide RRAM [96], [97] developed by researchers in Stanford University. It is usually used in a 3D cross-point array architecture [98] for ultra-high non-volatile memory applications, as shown in Figure 4.18. We used the 1D filament model [99], [21] that can be applied to large scale simulations. This model is described as follows.

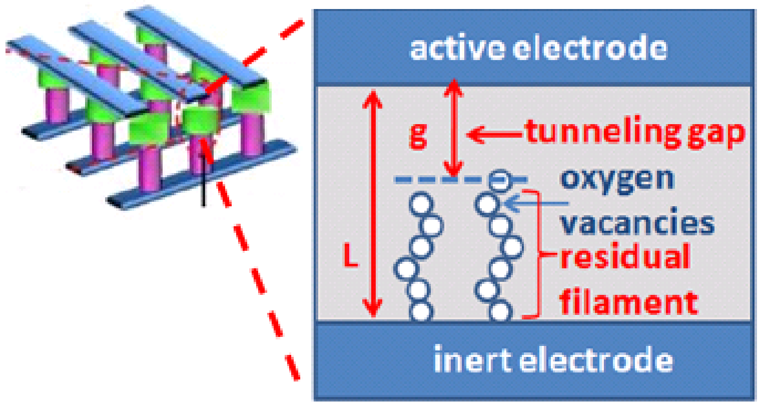


Figure 4.18: The 3D cross-point array architecture with RRAM devices at the junctions. Adapted from [21].

The size of the tunneling gap g, which is the distance between the top of the filament and the opposite electrode, is the primary state variable determining device resistance. Its change rate is a hyperbolic function of the input voltage V and the local temperature T:

$$dg/dt = -v_0 \cdot exp(-E_a/kT) \cdot sinh(\gamma \cdot a_0/L \cdot qV/kT) \tag{4.13}$$

where E_a is activation energy (~0.6 eV), γ is local field polarizability in high-k dielectrics, a_0 is atom spacing ($\sim$ 0.25 nm), L is oxide thickness ($\sim$ 12nm), q is electron charge, k is Boltzmann constant, and v_0 is fitting parameter (~10 nm/ns). The current conduction is exponentially dependent on the tunneling gap distance:

$$I = I_0 \cdot exp(-g/g_0) \cdot sinh(V/V_0) \tag{4.14}$$

where $I_0 = 1$ mA, $g_0 = 0.25$ nm, and $V_0 = 0.25$ V are fitting parameters from experiments. I-V curves obtained the experiments are reproduced by this model, as shown in Figure 4.19.

The RRAM variability is introduced by a Gaussian random number δg in the gap dynamics:

$$dg = dg_{(\text{ideal})} + \delta g \tag{4.15}$$

The simulated result with δg =0.0224 nm matches the experiments for the temporal variation with a measured variability of $\delta R/R$ ~9%. This is shown in Figure 4.20.

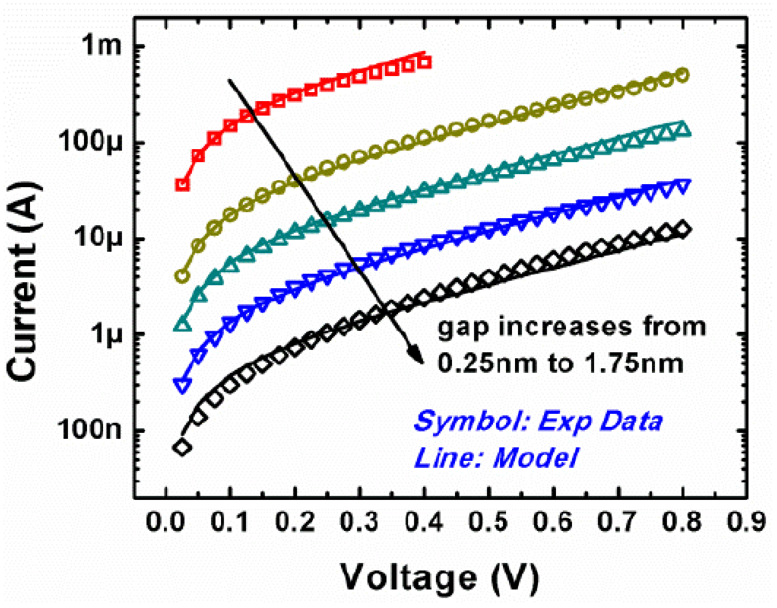


Figure 4.19: I-V fitting of multi-level resistance states by varying the gap tunneling distance. Adapted from [21].

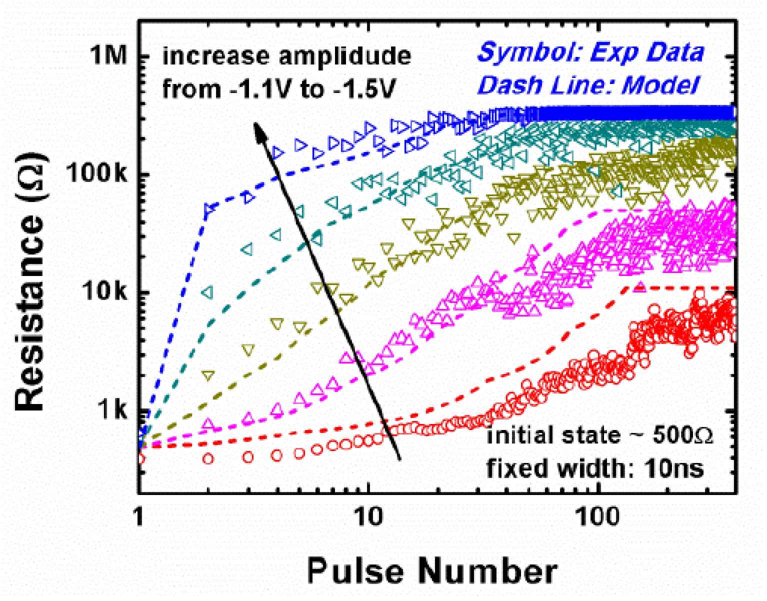


Figure 4.20: Measured and modeled gradual resistance change by varying the pulse amplitudes. Adapted from [21].

4.5.2 Pulsed-Programming of RRAM Resistance

Figure 4.21 shows the change rate of the tunneling gap dg/dt as a function of the applied voltage, using the model and parameter values from previous section. Because dg/dt is a hyperbolic sine function of the applied voltage, it is insignificant when the applied voltage is less than ± 1.5 V. However, when the applied voltage is greater than 2 V, it would experience a significant change.

Therefore, the RRAM resistance can be programmed using a pulsed-programming scheme [100], [101]. Figure 4.22 shows the numerical simulation of the applied voltage and the conductance change. In this programming scheme, pulses with alternating voltages between -1 and 1 V are applied to the source and sink terminals of the RRAM. When the applied pulses at these two terminals do not overlap, only ±1 V of voltage cross the resistor and the conductance change is insignificant. Furthermore, the conductance change caused by the 1-V pulse is offset by the -1-V pulse. However, when a -1 V pulse overlaps with a 1 V pulse (or the other way), a -2 V voltage across the RRAM causes significant changes in the tunneling gap and the conductance.

According to Equation 4.13, a ±2 V voltage will induce a change of the tunneling gap by Δg:

$$\Delta g = dg/dt_{|V=\pm 2V} \cdot \Delta t \tag{4.16}$$

where Δt is the pulse width. From Equation 4.14, the conductance will be scaled by a factor s:

$$s = exp(\Delta g/g_0) \tag{4.17}$$

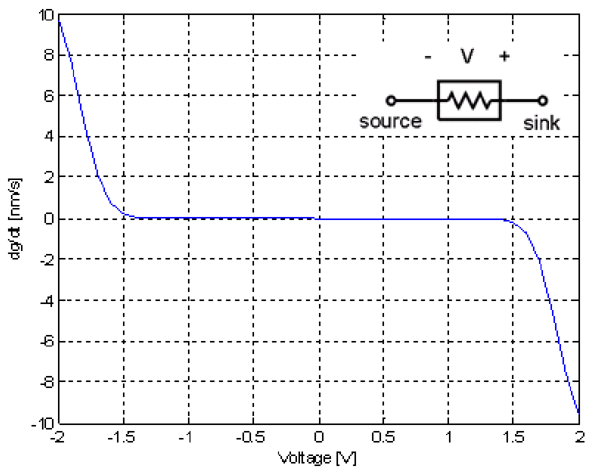


Figure 4.21: The change rate of the tunneling gap dg/dt as a function of the applied voltage V between the source and sink terminal, for a gap size g of 1nm.

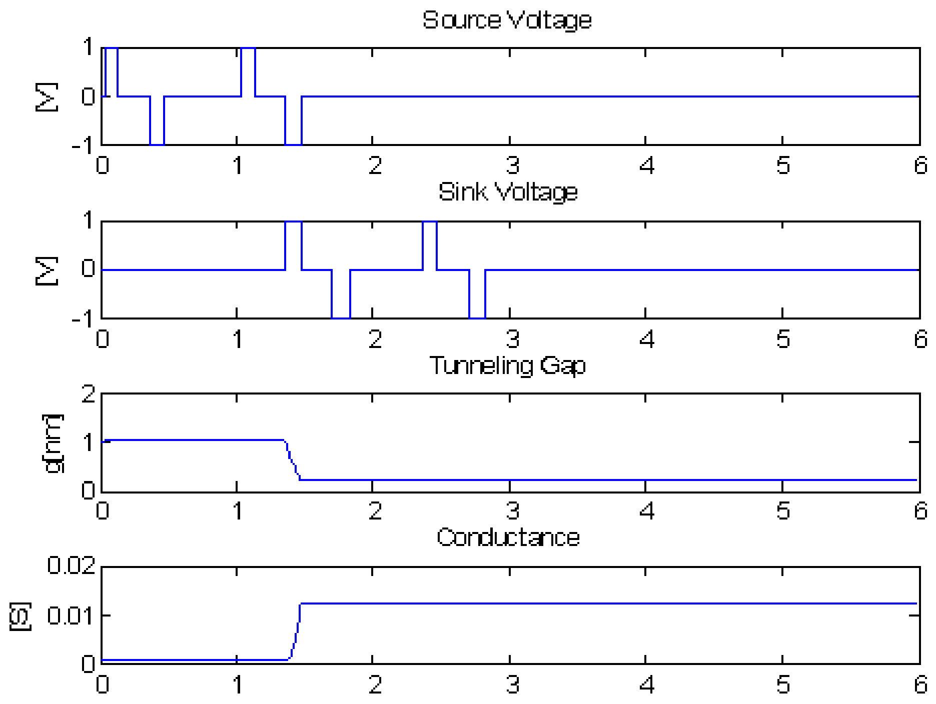


Figure 4.22: Pulsed programming of the RRAM resistance.

4.5.3 The RRAM Prototype

Figure 4.23 shows how to use the pulsed-programming scheme of RRAM resistance to implement the correlation matrix memory.

In this implementation, each memory cell is simply an RRAM. There are total $M \times U$ RRAMs plus a dummy conlumn of RRAMs, which will be explained shortly. Each RRAM in these $M \times U$ cells are connected to a word line y_m and a data line w_u. At the end of each column there are a current amplifier and a comparator.

During the training phase, the word lines for non selected rows remains at 0 V. For selected rows, an alternating $\pm$1-V pulse is applied to them. To add an one to the memory cell, the data line is applied a negative 1 V pulse when word line is at 1 V. This way, a 2-V voltage will be across that RRAM and cause a conductance increase by s times. Similarly, to add an minus 1, the data line is applied a +1V pulse when the word line is -1V. Therefore, a -2V pulse across the RRAM will cause its conductance to decrease by s times.

After trained with all the patterns, the conductance of each RRAM is scaled by s for c_{mn}

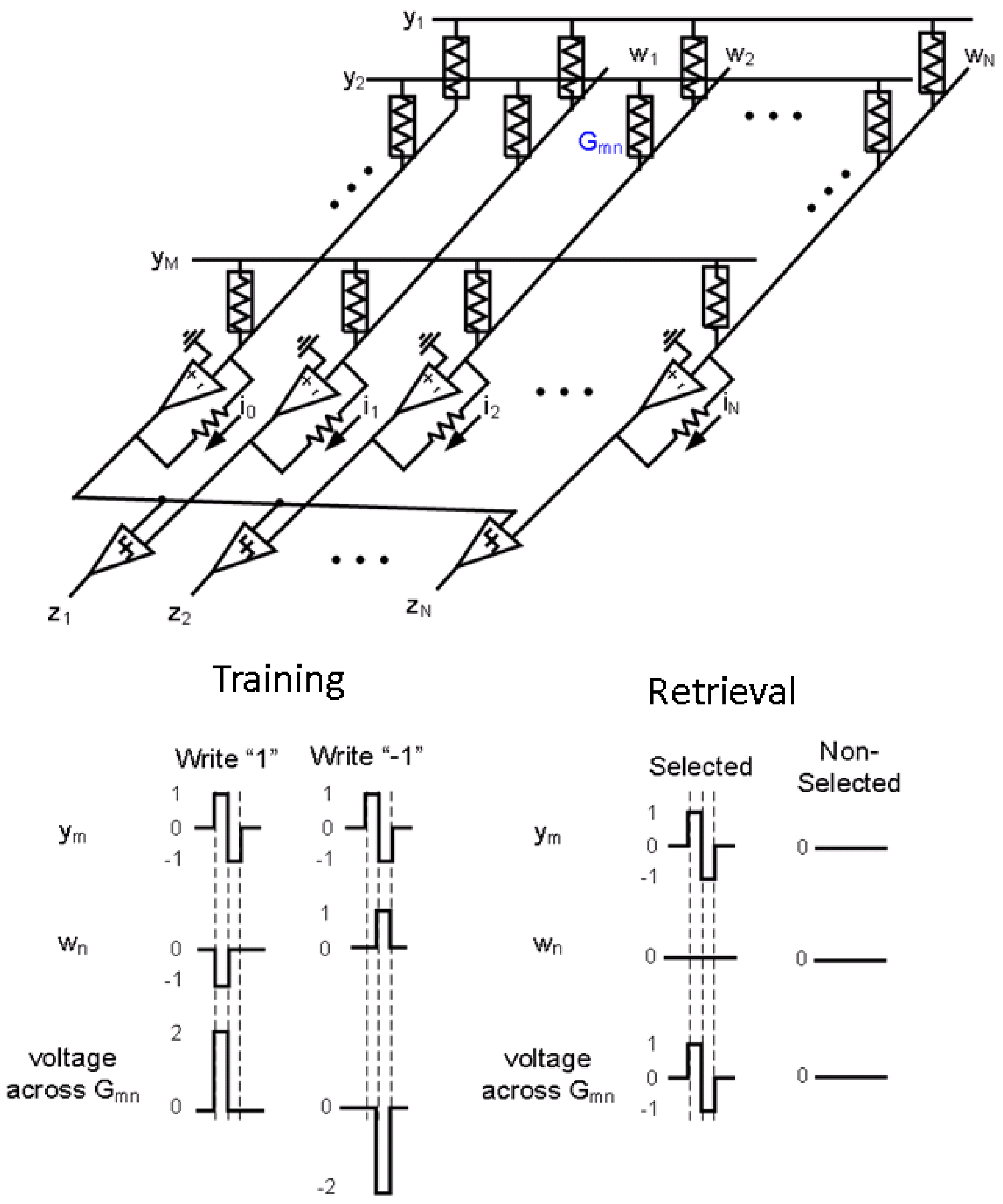


Figure 4.23: The RRAM prototype of the correlation matrix memory.

times, that is,

$$G_{mu} = G_{mu}(0) \cdot s^{\sum_{k=1}^{K} y_{m,k} w_{u,k}} \tag{4.18}$$

$$= G_{mu}(0) \cdot s^{c_{mu}} \tag{4.19}$$

$$\approx G_{mu}(0) \cdot (1 + c_{mu} x) \tag{4.20}$$

where c_{mu} is the final value of the memory cell in the correlation operation of CMM (Equation 4.7), and $s = 1 + x$.

In the retrieval phase, likewise the selection lines of the non-selected rows remain at 0 V, while the selected rows are applied an alternating pulse. The computation only happen when the pulse is at 1 V; the -1V pulse is only to offset the read disturbance on the RRAM. As mentioned in Section 4.2.2, the retrieval consists of a projection operation and a thresholding. The projection is done by summing up all the rows that are activated by $\mathbf{y}$, which can be easily implemented in this architecture. Looking at just one column, the rows that are activated will have currents flowing through RRAMs, and these current will be summed automatically at the input of the current amplifier and get amplified to a voltage. Therefore, the summed current at the end of each column i_u is

$$i_u = \sum_{m=1}^{M} y_m G_{mu} \tag{4.21}$$

$$\approx \sum_{m=1}^{M} y_m \cdot G_{mu}(0) \cdot (1 + c_{mu} x) \tag{4.22}$$

Compared to the desired sum at the output of each column s_u (which can be derived from Equation 4.9),

$$s_u = \sum_{m} y_m c_{mu} \tag{4.23}$$

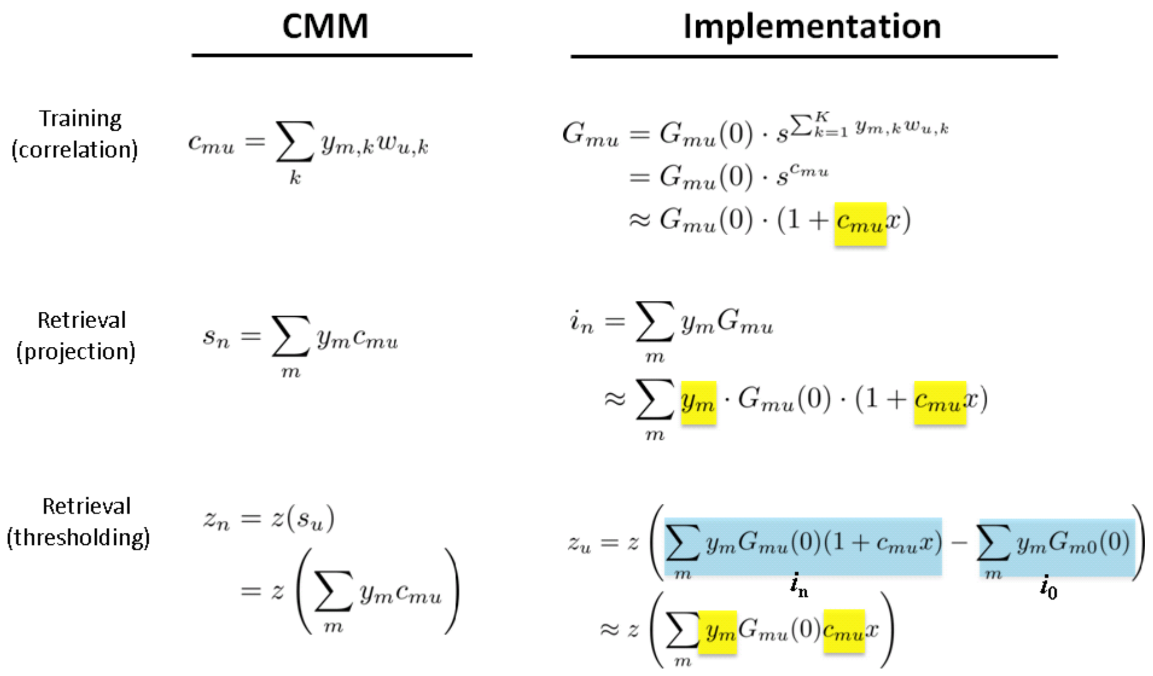
there is a nominal term that need to be canceled. This is the reason why the dummy column is added. The purpose of the dummy column is to generate a bias to be subtracted from each column. This is done by feeding the output voltage of the dummy column to the negative comparator inputs of all other columns. Since i_0 is

$$i_0 = \sum_{m} y_m G_{m0}(0) \tag{4.24}$$

the output of the comparator at each column is as desired for the thresholding operation:

$$z_u = i_u - i_0 \tag{4.25}$$
$$= z\left(\sum_m y_m G_{mu}(0)(1 + c_{mu}x) - \sum_m y_m G_{m0}(0)\right) \tag{4.26}$$
$$\approx z\left(\sum_m y_m G_{mu}(0) c_{mu} x\right) \tag{4.27}$$

Figure 4.24 summarizes the operations required by the CMM and those performed by the RRAM prototype.



	CMM	Implementation
Training (correlation)	$c_{mu} = \sum_k y_{m,k} w_{u,k}$	$G_{mu} = G_{mu}(0) \cdot s^{\sum_{k=1}^{K} y_{m,k} w_{u,k}}$ $= G_{mu}(0) \cdot s^{c_{mu}}$ $\approx G_{mu}(0) \cdot (1 + c_{mu}x)$
Retrieval (projection)	$s_n = \sum_m y_m c_{mu}$	$i_n = \sum_m y_m G_{mu}$ $\approx \sum_m y_m \cdot G_{mu}(0) \cdot (1 + c_{mu}x)$
Retrieval (thresholding)	$z_n = z(s_u)$ $= z\left(\sum_m y_m c_{mu}\right)$	$z_u = z\left(\underbrace{\sum_m y_m G_{mu}(0)(1 + c_{mu}x)}_{i_n} - \underbrace{\sum_m y_m G_{m0}(0)}_{i_0}\right)$ $\approx z\left(\sum_m y_m G_{mu}(0) c_{mu} x\right)$

Figure 4.24: Comparison of the operations required by the CMM and those performed by the RRAM prototype.

To summarize, in this implementation, there is no active element in the vast majority of the memory core. Therefore, there is no leakage problem. Also, since the rows are sparsely activated (activation probability less than 0.1%), the power consumption will be low. The future work includes behavior simulations taking into account the resistance variations, non-linearities and variations in the RRAM programming, and eventually building an integrated prototype of RRAMs and silicon CMOS FET or CNFET peripherals.

Chapter 5

Conclusions

In this chapter, we conclude with our thesis contributions and future works.

5.1 Thesis Contributions

The main contribution of this thesis is the demonstration of neuro-inspired computing paradigms for efficient implementation in present and future nano-technologies. The contributions are summarized as follows:

- An information-theoretical framework for joint architectural and circuit level optimization (Section 2.3).
- Dedicated feature extraction and learning algorithms for metal-oxide gas sensor arrays (Section 3.2). The concentration-invariant spike-timing encoding is inspired by the spatio-temporal firing patterns in the olfactory system.
- Low power CMOS implementation of the analog learning front end (Section 3.3), including a novel analog Euclidian vector normalizer (Section 3.3.3).
- A spike pattern classifier based on the hyper-dimensional computing principles (Section 4.2), performing reliable computations with devices having large parametric variations.
- Proposed a way to densely and efficiently implement the spike pattern classifier in a 3D CNFET-RRAM technology (Section 4.3).

5.2 Future Work

The works presented in this thesis are demonstrations of the first step toward alternative computing on future nano-technologies. Therefore, there are still many aspects that require further improvements.

For the analog gas sensing frontend:

- Current system can only deal with one odor analyte at a time. Future direction should target at detecting mixtures of odors [73].
- In the feature extraction algorithm, the analyte responses in the log space may not necessarily intersect at a sphere center. Therefore, we suggest using a high-dimensional manifold instead of a sphere, such that analyte response trajectories are orthogonal to the manifold surface and the projection will be effective.
- The circuits can adapt themselves to overcome variations/mismatches that cannot be dealt with by the learning algorithm.
- To be able to deal with sensor drifts.

For the spike pattern classifier:

- The random sampling assumes the inputs are uniformly distributed. For non-uniform input distributions, either the inputs should be whitened before entering the sparse vector generator. or the sparse vector generator should have a mechanism to adapt itself to the input statistics, such as Sparse Coding [32] mentioned in Chapter 2, Foldiak's network [102], etc.
- The correlation matrix memory is only suitable for recognizing one pattern at a time. For mixture of patterns, the patterns may need to be manipulated and decoded as in a hyper-dimensional computer [31].

In today's cloud center, the machine learning kernels require thousands of servers burning MWatts of power. Eventually invention in the kernel architecture for efficient hardware implementation will be dictated. For example, Deep Learning based on neural nets have been very successful in pattern recognition, adaptive control, etc. However, it is mostly implemented in traditional platforms such as GPUs. The bottleneck between the processing unit and the memory storage will require in-memory computing architectures in the future. This is when the 3D nano comes into play.

Given that nature and evolution have found in the human brain a (optimal?) solution balancing size, power budget, cell complexity/size, and wiring density, the lessons we have learned for future nano ICs include:

- 3D integration
- Locality
- Massive distributed parallel processing with in-memory computing
- Robust data representations

- Optimization of cell complexity/size versus interconnect density and speed

At the waning day of Moore's Law, we may need to take a fresh look on how optimal computational engines can be built. Based on our knowledge of physics, materials, nano-devices, noise and energy constraints, can we derive what such an optimal solution could look like for a physical computer? This might be the time when the paths between physical and biological information processing are converging. Therefore, a true understanding of information-theoretic and thermodynamic trade-off's will be needed. And the advance in this exciting field will require the collaboration of neuroscientists, architects, circuit and device engineers.

Appendix A

Power-Precision Models

In terms of signal representations, the precision in bits b is defined as

$$b = \frac{1}{2} log_2 SNR \tag{A.1}$$

where SNR is the signal-to-noise ratio. In analog circuits, the power is usually linear to the output SNR, whereas in digital circuits, the power is linear to b. Therefore, the analog power is exponential to b while digital power is linear to b:

$$P_A \propto SNR = 2^{2b} \tag{A.2}$$

$$P_D \propto log_2 SNR = 2b \tag{A.3}$$

This leads to an argument that analog circuits are more efficient in the low-precision regime, while digital circuits are more efficient in the high-precision regime [61].

To understand whether this arguments applies to computations, we investigated a specific case, i.e. logarithmic transform. In this appendix, the power-precision relationships for analog and digital logarithmic transforms are derived.

A.1 Analog Logarithmic Transform

Figure A.1 shows the schematic of the logarithmic amplifier. If transistor M_f is biased in subthreshold, the output voltage is linear to the log of the input current:

$$\Delta v_o = \frac{A}{A+1} n V_t ln \left(\frac{i_{IN}}{I_{BIAS}} \right) \tag{A.4}$$

where $i_{IN} = I_{BIAS} + \Delta i_{IN}$.

This logarithmic amplifier is designed in ST Microelectronics' 65 nm SVTLP process. With a supply voltage of 0.5 V, $I_{D1} = 2.5$ nA, $I_{D2} = 1.3$ nA, $C_L = 500$ fF, the amplifier achieves a gain of 43.2 dB and a closed-loop bandwidth of 90 kHz. The DC response is shown in Figure A.2.

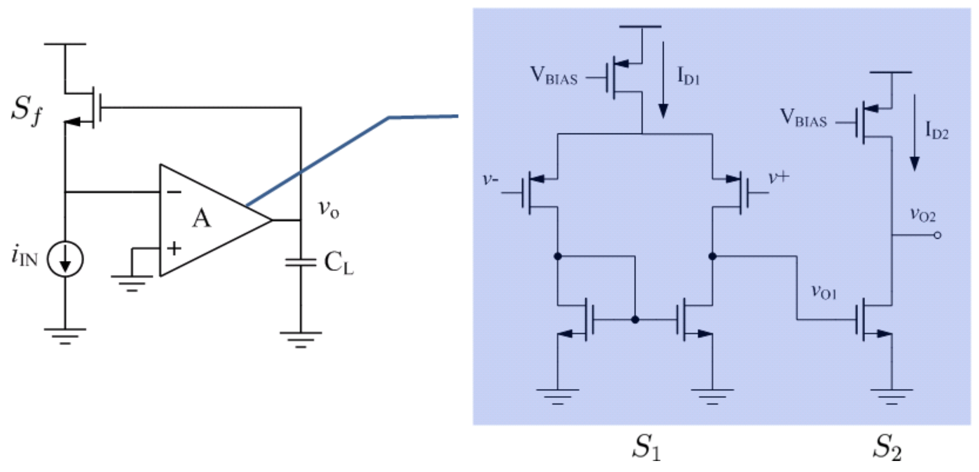


Figure A.1: Schematic of the logarithmic amplifier.

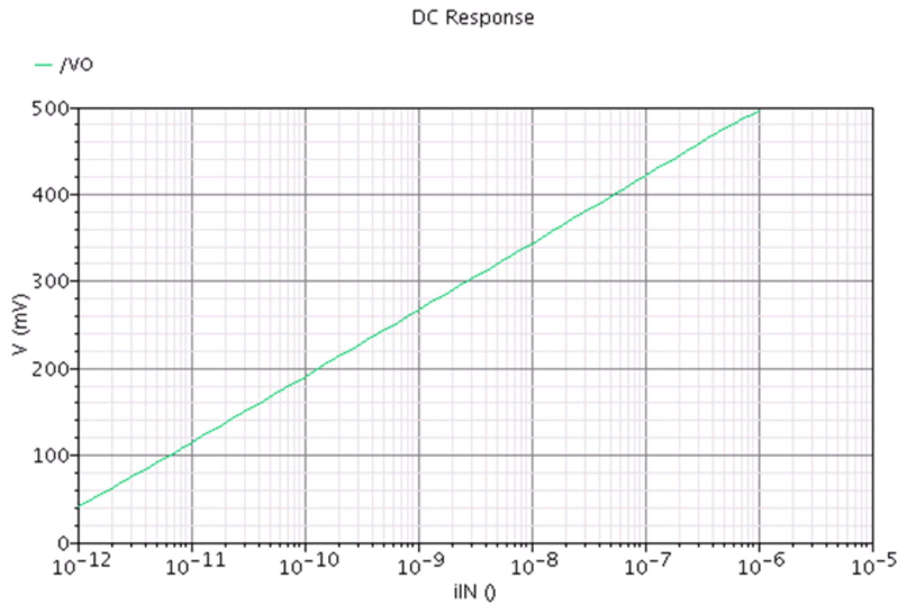


Figure A.2: DC response of the logarithmic amplifier: V_o versus logi_{in}.

The output noise of the amplifier is inversely proportional to the load capacitance:

$$\sqrt{\overline{v^2_{on,T}}} \propto \left(\frac{1}{I_{D1}} + \frac{1}{I_{BIAS}}\right)\frac{A_{v1}g_{m2}}{C_L}, \tag{A.5}$$

and the bandwidth of amplifier is the unity-gain bandwidth of the loop-gain:

$$BW \approx \frac{A_{v1}g_{m2}}{C_L} \tag{A.6}$$

Therefore, to decrease the output noise, the load capacitance is increased. To keep the same bandwidth, the transistors must be sized up. Hence the total bias current increases.

Table A.1 shows the simulated current consumption, output noise and output voltage variations at different sets of sizing factors S_1, S_2, and S_f. These sizing factors are the scaling factors of transistor sizes in the first stage, second stage of the op-amp, and that of the feedback transistor M_f. As transistor sizes increase, not only the output noise $V_{on,T}$ decreases, the standard deviation of the amplifier offset $\sigma_{V_{OS,a}}$ and that of the output voltage σ_{V_o} decrease as well. In this case, the output precision is actually dominated by device variations, not by the thermal noise.

Table A.1: Current consumption, output noise, and output voltage variation at different sizing factors.

S_1	S_1	S_f	C_L (pF)	I_{TOT} (nA)	BW (kHz)	$V_{on,T}$ (μV)	$\sigma_{V_{OS,a}}$ (mV)	σ_{V_O} (mV)
1	1	1	0.5	4.585	90	762	9.54	16.69
2	2	2	0.8	8.676	90	544	7.57	11.16
4	4	4	1.5	15.236	90	372	6.11	8.35
8	8	8	2.5	29.496	90	279	4.57	5.90
16	16	16	4.75	58.11	90	198	3.33	4.17
32	32	32	9	105.41	90	143	2.38	2.95
64	64	64	18	230.01	90	101	1.68	2.09

From Figure A.2, the output range corresponding to 60 dB input dynamic range (0.1-100 I_{BIAS}) is roughly 250 mV. Then the output precision in terms of bits can be defined as follows.

$$R = log_2\left(\frac{V_L}{2\left(\sqrt{\overline{v^2_{on,T}} + \sigma^2_{Vo}}\right)}\right) \tag{A.7}$$

A.2 Digital Logarithmic Transform

Figure A.3 shows a look-up table for digital logarithmic transform. It consists of a 10-bit decoder (comparable to the 60-dB input dynamic range in the analog logarithmic amplifier), and a NAND-ROM. There are B outputs corresponding to B bits of precision.

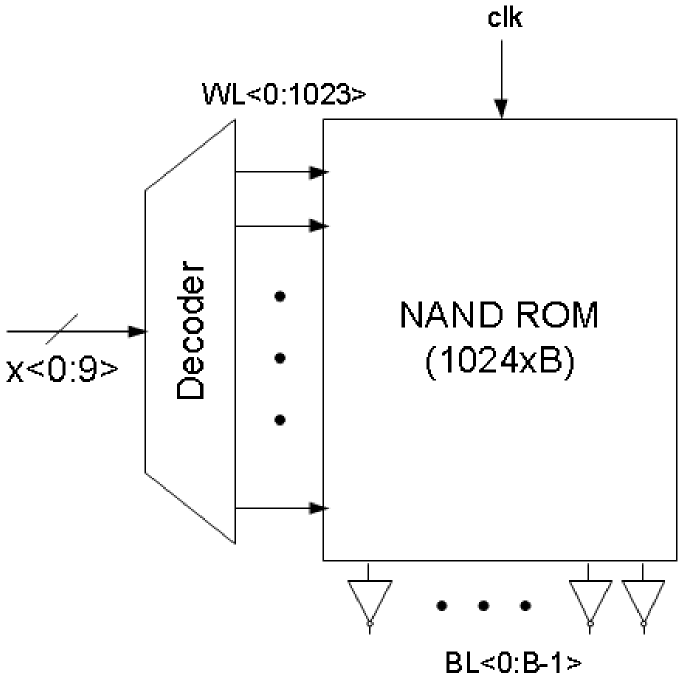


Figure A.3: Digital look-up table (LUT) for logarithmic transform.

This digital LUT is implemented in ST Microelectronics' LVTLP process and operated at a supply of 0.5 V. The sizes of the transistors are determined by DC output levels in the worst-case configurations. As shown in Figure A.4, the left branch shows the worst case for output low, and the right branch shows the worst case for output high.

A.2.1 Leakage Power Estimation

The leakage power is estimated by taking the average of the maximum and minimum leakage currents for the three scenarios: precharge, 1-evaluated, and 0-evaluated. The worst and best cases for these three scenarios are shown in Figure A.5.

Assuming equal precharge and evaluation period, the leakage power is then

$$P_{Leakage} = \frac{1}{2}\left[P_{PRE} + \alpha P_0 + (1-\alpha)P_1\right] \times B \tag{A.8}$$

where α is fraction of zeros in the LUT, which can be calculated using a program.

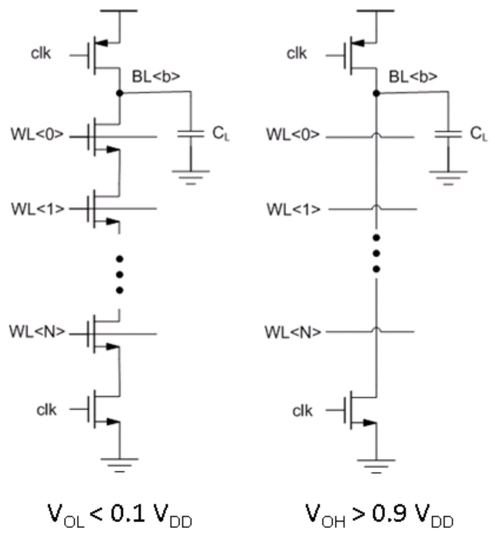


Figure A.4: The worst case configurations for output low (left) and output high (right).

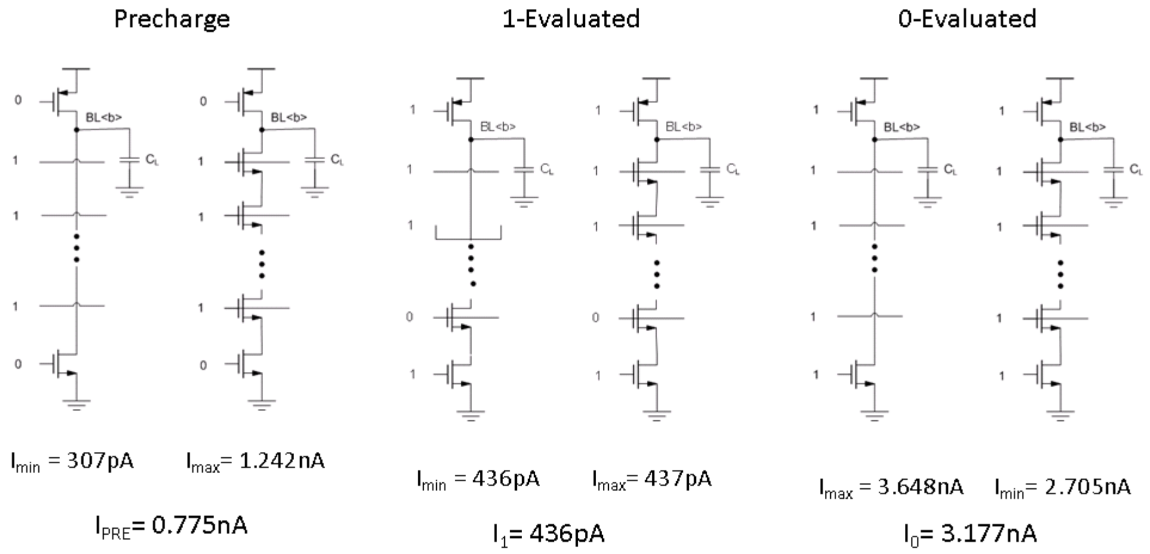


Figure A.5: Leakage estimation for the NAND-ROM.

A.2.2 Dynamic Power Estimation

Figure A.6 shows the schematic for load capacitance estimation. Assuming a wire capacitance of 0.2 fF/μm^2 and a wire resistance of 0.2 Ω/sq, the word line capacitance is $C_{WL} = 0.2f \times 20\mu m \times B = 4BfF$. Since the maximum wire capacitance is $C_{W,max} = 0.2 \times 1024\mu m^2 = 205$ fF, the average wire capacitance is $C_{W,ave} = 103fF$.

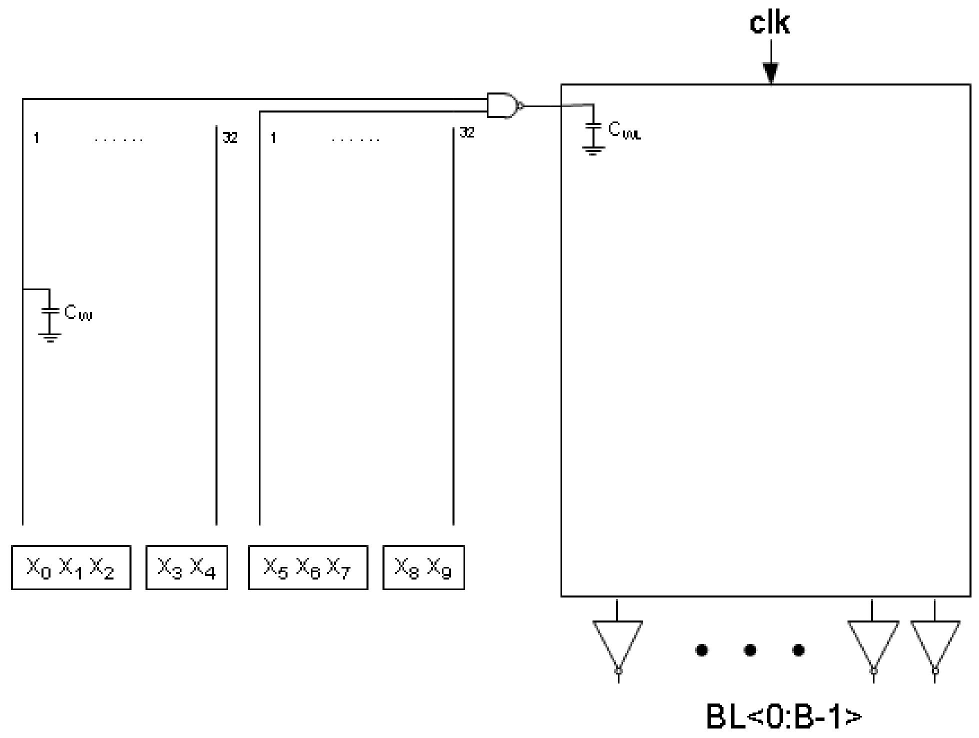


Figure A.6: Schematic for loading capacitance estimation.

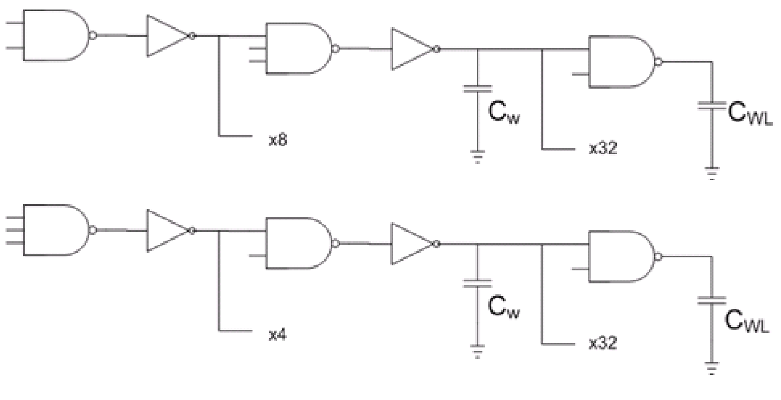


Figure A.7: Different branchings in two-input and three-input decoders.

Since the inputs are grouped into two or three, there are two kinds of branchings in the decoder chains, as shown in Figure A.7. The dynamic power of the decoder is

$$P_{DEC} = n_{predec} P_{predec,ave} + n_{nand} P_{nand,ave} + \frac{1}{2} C_{w,ave} V_{DD}^2 f \tag{A.9}$$

where $n_{predec} = 64$, $P_{predec,ave} = 271$ pA, $n_{nand} = 1024$, $P_{nand,ave} = 50$ pA, and $f = 100$ kHz.

The dynamic power also includes the clock power P_{CK}, the power to drive the bit line P_{BL}, and the power to drive the word line P_{WL}:

$$P_{CK} = C_{ckload} V_{DD}^2 f \times B \tag{A.10}$$

$$P_{BL} = \alpha C_{BL} V_{DD}^2 f \times B \tag{A.11}$$

$$P_{WL} = C_{WL} V_{DD}^2 f \tag{A.12}$$

A.2.3 Total Power Consumption

Therefore, the total power consumption of the digital LUT logarithmic transform is

$$P_{TOT} = P_{Leakage} + P_{DEC} + P_{CK} + P_{BL} + P_{WL} \tag{A.13}$$

which is a function of the output precision B.

A.3 Analog VS. Digital

Finally, the simulated power of the analog logarithmic transform and the estimated power of the digital logarithmic transform LUT as a function of the output precision B are summarized in Table A.2 and plotted in Figure A.8. It can be concluded that analog logarithmic transform is more efficient in the low-precision regime, while digital logarithmic transform is more efficient in the high-precision regime.

Table A.2: Simulated analog power and estimated digital power of the logarithmic transform as functions of output precision.

Digital Precision [bits]	Digital Power [nW]	Analog Precision [bits]	Analog Power [nW]
2	57.505	2.9	2.2925
4	59.97	3.48	4.338
6	62.487	3.9	7.618
8	65	4.4	14.748
10	67.51	4.9	29.055
12	70.29	5.4	57.705
14	72.86	5.9	115

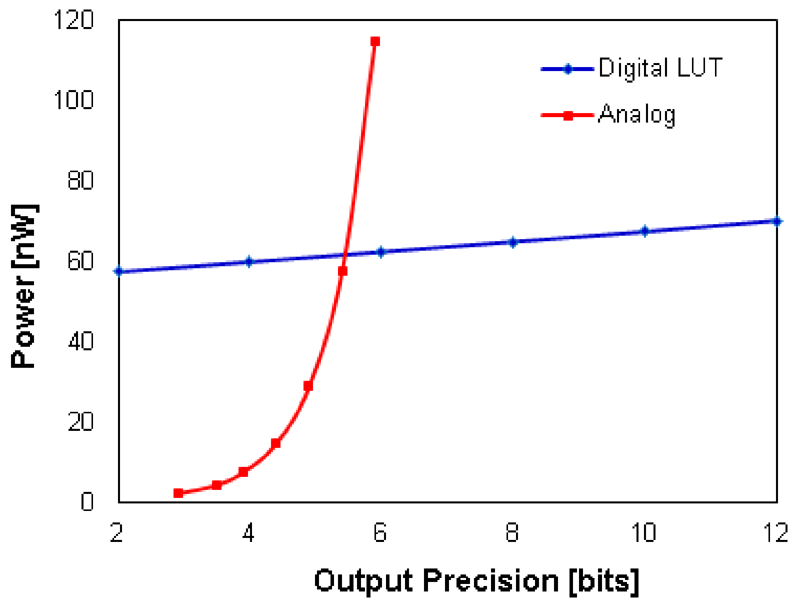


Figure A.8: The power-precision relationship of the analog and digital logarithmic transforms.

Appendix B

Error Analysis

This appendix provides error modeling of the analog learning frontend and investigates how it impacts the convergence behavior of the learning algorithm.

B.1 Error Modeling

Figure B.1 shows error sources for computation in the learning phase. Note that the output variations from the input logarithmic converters will be captured by the learning algorithm, and that the sample-and-hold circuit is immune to device mismatches/variations, the errors come from the V-I converter, and the normalizer. In this figure, only one channel (channel i) is shown for clarity. However, the error analysis is generalized to N channels.

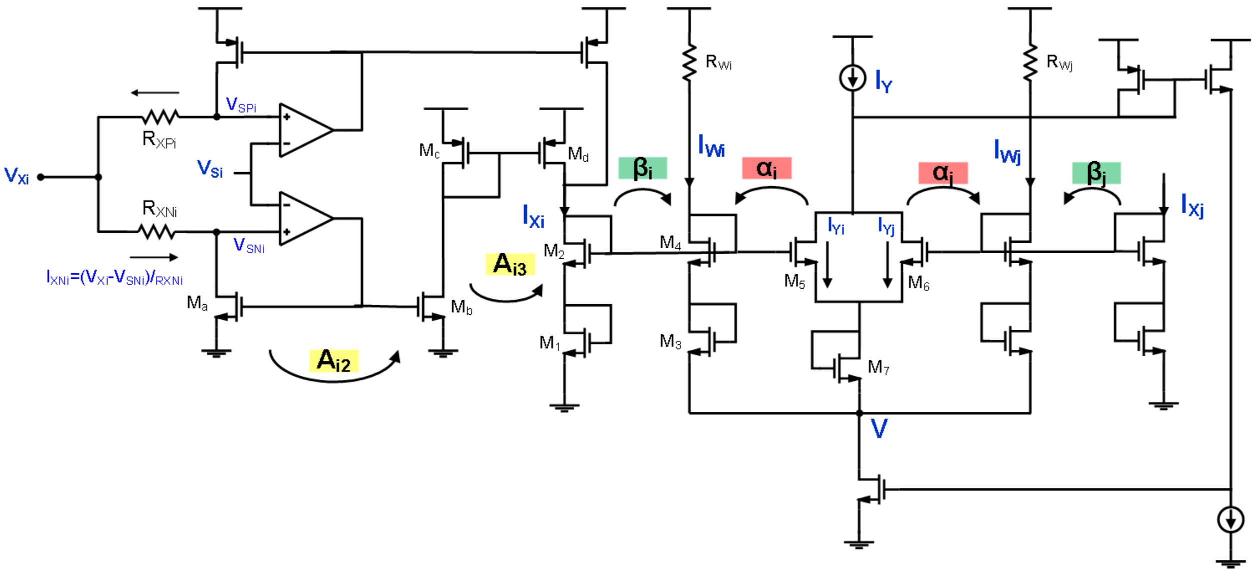


Figure B.1: Modeling the errors due to device mismatch for the computation in the learning phase.

B.1.1 Errors from the V-I Converters

Ideally, the V-I converter should produce an output current of

$$I_{Xi} = \frac{V_{Xi} - V_{Si}}{R_{Xi}}.$$

However, in reality, due to device mismatches, the actual output current is

$$I_{Xi} = \frac{V_{Xi} - V_{Si}}{R_{Xi}} \cdot \left(\frac{A_{i1} A_{i2} A_{i3}}{r_{Xi}} \right) \tag{B.1}$$

where r_{Xi}, A_{i1}, A_{i2}, and A_{i3} are random variables nominally at 1. r_{Xi} is the factor that resistance R_{Xi} is scaled by, A_{i1} is due to the amplifier offset ΔV_{Si}:

$$A_{i1} = 1 + \frac{\Delta V_{Si}}{V_{Xi} - V_{Si}},$$

A_{i2} is due to the mismatch between M_a and M_b in the first stage of current mirroring:

$$A_{i2} = \frac{I_{0a}}{I_{0b}} \frac{(W/L)_a}{(W/L)_b} \cdot exp\left(\frac{V_{THb} - V_{THa}}{nV_t} \right),$$

and A_{i3} is due to the mismatch between M_c and M_d in the second stage of current mirroring:

$$A_{i3} = \frac{I_{0c}}{I_{0d}} \frac{(W/L)_c}{(W/L)_d} \cdot exp\left(\frac{V_{THd} - V_{THc}}{nV_t} \right).$$

B.1.2 Errors from the Normalizer

Ideally, the output currents I_W's and input currents I_X's of the normalizer should be related as follows.

$$\begin{aligned} I_{W1}^2 &= I_{X1}^2 \cdot \frac{1}{K} \\ I_{W2}^2 &= I_{X2}^2 \cdot \frac{1}{K} \\ &\vdots \\ I_{WN}^2 &= I_{XN}^2 \cdot \frac{1}{K} \end{aligned}$$

where K is a function of the voltage V at the source of transistor M_7. Also the output current I_W's and the inner branch currents I_Y's should be related as follows.

$$\begin{aligned} I_{W1}^2 &= I_{Y1} I_Y \\ I_{W2}^2 &= I_{Y2} I_Y \\ &\vdots \\ I_{WN}^2 &= I_{YN} I_Y \end{aligned}$$

However, due to device mismatches, output currents are related to input currents through β's.

$$\begin{aligned} I_{W1}^2 &= I_{X1}^2 \cdot \beta_1 \cdot \frac{1}{K} \\ I_{W2}^2 &= I_{X2}^2 \cdot \beta_2 \cdot \frac{1}{K} \\ &\vdots \\ I_{WN}^2 &= I_{XN}^2 \cdot \beta_N \cdot \frac{1}{K} \end{aligned}$$

for example,

$$\beta_1 = \frac{I_{03} I_{04} (W/L)_3 (W/L)_4}{I_{01} I_{02} (W/L)_1 (W/L)_2} \cdot exp\left(\frac{V_{TH1} + V_{TH2} - V_{TH3} - V_{TH4}}{nV_t} \right).$$

And the output currents are related to inner branch currents through α's.

$$\begin{aligned} I_{W1}^2 &= \alpha_1 I_{Y1} I_Y \\ I_{W2}^2 &= \alpha_2 I_{Y2} I_Y \\ &\vdots \\ I_{WN}^2 &= \alpha_N I_{YN} I_Y \end{aligned}$$

for example,

$$\alpha_1 = \frac{I_{03} I_{04} (W/L)_3 (W/L)_4}{I_{05} I_{07} (W/L)_5 (W/L)_7} \cdot exp\left(\frac{V_{TH5} + V_{TH7} - V_{TH3} - V_{TH4}}{nV_t} \right).$$

Note that $I_{Y1} + I_{Y2} + \cdots + I_{YN} = I_Y$, then

$$\left(\frac{\beta_1}{\alpha_1} I_{X1}^2 + \frac{\beta_2}{\alpha_2} I_{X2}^2 + \cdots + \frac{\beta_N}{\alpha_N} I_{XN}^2 \right) \frac{1}{K \cdot I_Y} = I_Y.$$

Therefore,

$$\frac{1}{K} = \frac{I_Y^2}{\sum \frac{\beta_i}{\alpha_i} I_{Xi}^2}.$$

The output voltages V_{RW}'s then become

$$V_{RWi}^2 = R_{Wi}^2 \beta_i I_{Xi}^2 \cdot \frac{1}{K} \tag{B.2}$$

$$= R_{Wi}^2 \alpha_i \cdot \frac{\frac{\beta_i}{\alpha_i} I_{Xi}^2}{\sum \frac{\beta_i}{\alpha_i} I_{Xi}^2} I_Y^2 \tag{B.3}$$

B.2 The Output Error

Combining Equation B.1 and Equation B.3, the output voltages expressed in terms of the input voltages are

$$V_{RWi}^2 = R_{Wi}^2 \alpha_i \cdot \frac{\frac{\beta_i}{\alpha_i} \frac{(V_{Xi} - V_{Si})^2}{R_{Xi}^2} A_i^2}{\sum \frac{\beta_i}{\alpha_i} \frac{(V_{Xi} - V_{Si})^2}{R_{Xi}^2} A_i^2} I_Y^2. \tag{B.4}$$

where $A_i = A_{i1} A_{i2} A_{i3}$. In vector notation, the output voltage vector can be expressed as

$$\mathbf{V_{RW}} = \begin{bmatrix} r_{W1}\sqrt{\alpha_1} \\ r_{W2}\sqrt{\alpha_2} \\ \vdots \\ r_{WN}\sqrt{\alpha_N} \end{bmatrix} .* \left(\begin{bmatrix} \sqrt{\frac{\beta_1}{\alpha_1}} \frac{A_{i1}}{r_{X1}} \\ \sqrt{\frac{\beta_2}{\alpha_2}} \frac{A_{i2}}{r_{X2}} \\ \vdots \\ \sqrt{\frac{\beta_N}{\alpha_N}} \frac{A_{iN}}{r_{XN}} \end{bmatrix} .* \begin{bmatrix} V_{X1} - V_{S1} \\ V_{X2} - V_{S2} \\ \vdots \\ V_{XN} - V_{SN} \end{bmatrix} \right)_u \cdot R_W I_Y \tag{B.5}$$

$$\equiv \mathbf{s}_1 .* \left(\mathbf{s}_2 .* \begin{bmatrix} V_{X1} - V_{S1} \\ V_{X2} - V_{S2} \\ \vdots \\ V_{XN} - V_{SN} \end{bmatrix} \right)_u \cdot R_W I_Y \tag{B.6}$$

Therefore, the input vector $(V_{X1} - V_{S1}, V_{X2} - V_{S2}, \cdots, V_{XN} - V_{SN})$ first undergoes an element-wise multiplication (.*) with vector $\mathbf{s}_2$, then a vector normalization ($(.)_u$), and finally another element-wise multiplication with vector $\mathbf{s}_1$.

From block-level Monte Carlo simulations, the worst-case variations in the parameters are as follows.

$$\begin{aligned} r_{Wi} &= 1 \pm 30\% \\ \alpha_i &= 1 \pm 30\% \\ \beta_i &= 1 \pm 25\% \\ \frac{A_i}{r_{Xi}} &= 1 \pm 40\% \end{aligned}$$

Therefore, $\sqrt{\beta_i} A_i / \sqrt{\alpha_i} r_{Xi}$ could vary between 0.46 to 1.87, and $r_{Wi}\sqrt{\alpha_i}$ could vary between 0.59 to 1.48.

If $1 - \Delta_1 \leq s_1, s_2, \cdots, s_N \leq 1 + \Delta_2$, it can be shown that

$$\mathbf{s}.*\mathbf{x} = \begin{bmatrix} s_1x_1 \\ s_2x_2 \\ s_3x_3 \\ s_4x_4 \\ s_5x_5 \\ s_6x_6 \end{bmatrix} = \begin{bmatrix} (1+\Delta_2)x_1 \\ (1+\Delta_2)x_2 \\ (1+\Delta_2)x_3 \\ (1-\Delta_1)x_4 \\ (1-\Delta_1)x_5 \\ (1-\Delta_1)x_6 \end{bmatrix}$$

will result in the largest angle between $\mathbf{s}.*\mathbf{x}$ and $\mathbf{x}$.

Therefore, in the simulation of the convergence behavior of the learning algorithm, the worst-case vectors $\mathbf{s}_1$, $\mathbf{s}_2$ are used in the computation of each update.

$$\mathbf{s}_1 = \begin{bmatrix} 1.87 \\ 1.87 \\ 1.87 \\ 0.46 \\ 0.46 \\ 0.46 \end{bmatrix}, \mathbf{s}_2 = \begin{bmatrix} 1.48 \\ 1.48 \\ 1.48 \\ 0.59 \\ 0.59 \\ 0.59 \end{bmatrix}.$$

The final states for 720 initial states using this method are those shown in Figure 3.44.

Appendix C

Geometric Interpretation of Spike Sampling

This appendix provides the derivation of the reception range of the spike sampling groups and its geometric interpretation. They corresponds to hyperplanes with different orientations in the N-dimensional space of spike delays.

C.1 Parameters

This section summarizes the parameters that were defined previously and parameters that will be used in the following analysis.

K: Size of the data set; number of odor analytes.

N: Input dimension; number of sensors.

M: Output dimension; number of sampling groups.

n: Number of elements in each sampling group; $n \leq N$.

$\mathbf{T}$: $K \times N$ matrix; timing patterns.

$$\mathbf{T} = \begin{bmatrix} t_{11} & t_{12} & \cdots & t_{1N} \\ t_{21} & t_{22} & \cdots & t_{2N} \\ \vdots & \vdots & \cdots & \vdots \\ t_{K1} & t_{K2} & \cdots & t_{KN} \end{bmatrix} \tag{C.1}$$

$\mathbf{d}_m$: n-element vector; delays of the n elements in the m^{th} sampling group.

$$\mathbf{d}_m = \begin{bmatrix} d_{m1} \\ d_{m2} \\ \vdots \\ d_{mn} \end{bmatrix} \tag{C.2}$$

N_m: Set of indices of the channels that are connected to the m^{th} sampling group. That is, $N_m \subset \{1, 2, \cdots, N\}$ and $|N_m| = n$. These n elements are uniformly and randomly selected from $\{1, 2, \cdots, N\}$.

$\mathbf{T}_{N_m}$: The $K \times n$ submatrix of $\mathbf{T}$ by extracting the columns of $\mathbf{T}$ according to the indices in N_m; these are the timing patterns *seen* by the m^{th} group. This is illustrated by Figure C.1.

$\mathbf{t}_{N_m,k}$: The k^{th} row of the $\mathbf{T}_{N_M}$ matrix; this is the timing pattern of the k^{th} analyte *seen* by the m^{th} group.

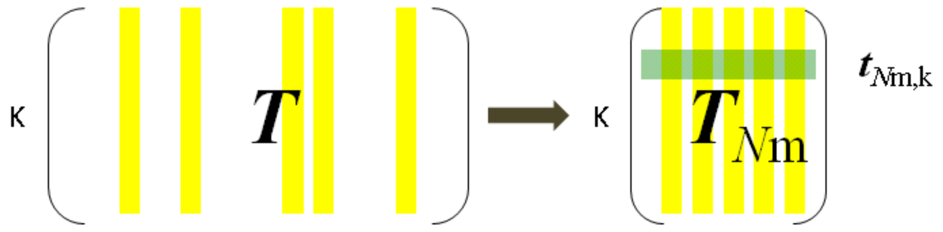


Figure C.1: Extracting columns of $\mathbf{T}$ according to the indices in N_m to form $\mathbf{T}_{N_M}$.

C.2 Reception Ranges

Given the delay $\mathbf{d}_m$ of group m, what is the set of timing patterns that will activate this group? More specifically, we want to find the set of timing patterns $\mathbf{t}$ satisfying

$$\mathbf{t}_{Nm}^T + \mathbf{d}_m = s\mathbf{1},\ 0 < s < \infty. \tag{C.3}$$

In vector notations, this set of $\mathbf{t}$ can be expressed as

$$\mathbf{t} = s_1\mathbf{v}_1 + s_2\mathbf{v}_2 + \cdots + s_{N-n+1}\mathbf{v}_{N-n+1} + \mathbf{p} \tag{C.4}$$

where

$$\mathbf{v}_1 = \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \quad \mathbf{v}_2 = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \quad \mathbf{v}_3 = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 0 \\ 1 \\ \vdots \\ 0 \end{bmatrix}, \quad \cdots, \quad \mathbf{v}_{N-n+1} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix}, \quad \mathbf{p} = \begin{bmatrix} -d_{j1} \\ -d_{j2} \\ \vdots \\ -d_{jn} \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}. \tag{C.5}$$

$0 < s_j < \infty$, $j = 1, 2, \cdots, N-n+1$, and $\mathbf{v}_1, \mathbf{v}_2, \cdots, \mathbf{v}_{N-n+1}$ are linearly independent.

According to the definition in geometry [103], the set of $\mathbf{t}$ is a $(N-n+1)$-dimensional affine subspace of $\mathcal{R}^N$. If $n = 2$, this is a $(N-1)$-dimensional affine subspace of $\mathcal{R}^N$, i.e. a hyperplane in $\mathcal{R}^N$.

Figure C.2 shows the geometric interpretation. In this case, $N = 3$, and $n = 2$. Only three reception ranges are shown for clarity. It can be seen that the reception range for each group is a hyperplane in $\mathcal{R}^3$, i.e. a plane. The orientation of the planes depends on the subset of input channels N_m that a group is connected to and also the delays in the group.

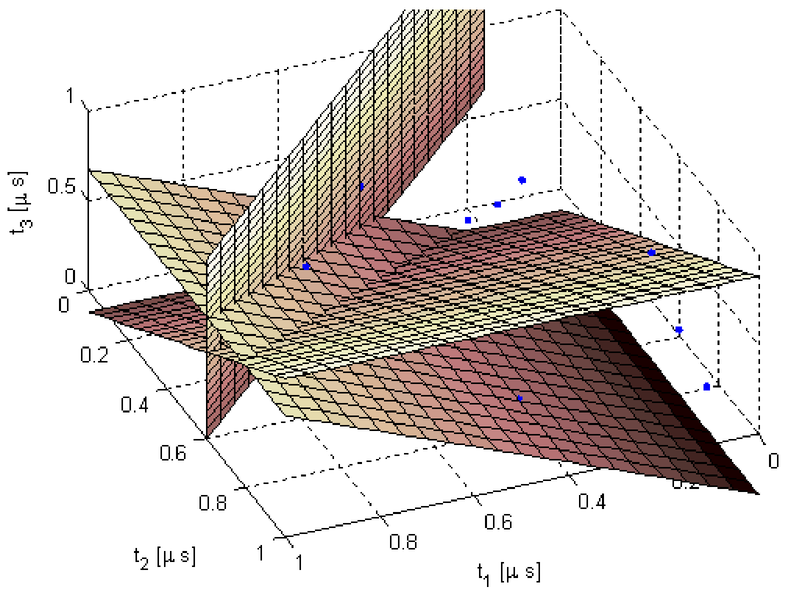


Figure C.2: The geometric interpretation of the random sampling groups. In this case, $N = 3$ and $n = 2$.

Bibliography

[1] K. Rupp, "40 Years of Microprocessor Trend Data." www.karlrupp.net/2015/06/40-years-of-microprocessor-trend-data/, 2015.

[2] Y. Ye, S. Gummalla, C.-C. Wang, C. Chakrabarti, and Y. Cao, "Random variability modeling and its impact on scaled cmos circuits," *Journal of computational electronics*, vol. 9, no. 3-4, pp. 108–113, 2010.

[3] B. Wu, A. Kumar, and S. Ramaswami, *3D IC Stacking Technology.* McGraw-Hill Professional, 2011.

[4] V. V. Zhirnov, R. K. Cavin, J. A. Hutchby, and G. I. Bourianoff, "Limits to binary logic switch scaling-a gedanken model," *Proceedings of the IEEE*, vol. 91, no. 11, pp. 1934–1939, 2003.

[5] J. Hutchby, "The Nanoelectronics Roadmap," in *Emerging Nanoelectronic Devices* (A. Chen, J. Hutchby, V. Zhirnov, and G. Bourianoff, eds.), John Wiley & Sons Ltd, 2015.

[6] "Semiconductor Industry Association (2011) Emerging Research Devices Chapter of the ITRS." http://www.itrs.net/Links/2011ITRS/2011Chapters/2011ERD.pdf, 2011.

[7] S. Das, "ITRS Assessment and Benchmarking of Emerging Logic Devices," in *Emerging Nanoelectronic Devices* (A. Chen, J. Hutchby, V. Zhirnov, and G. Bourianoff, eds.), John Wiley & Sons Ltd, 2015.

[8] J. Deng, K. Ryu, C. Zhou, *et al.*, "Carbon nanotube transistor circuits: Circuit-level performance benchmarking and design options for living with imperfections," in *2007 IEEE International Solid-State Circuits Conference. Digest of Technical Papers*, pp. 70–588, 2007.

[9] N. Patil, J. Deng, S. Mitra, and H. P. Wong, "Circuit-level performance benchmarking and scalability analysis of carbon nanotube transistor circuits," *Nanotechnology, IEEE Transactions on*, vol. 8, no. 1, pp. 37–45, 2009.

[10] N. Patil, A. Lin, J. Zhang, H.-S. P. Wong, and S. Mitra, "Digital vlsi logic technology using carbon nanotube fets: Frequently asked questions," in *Proceedings of the 46th Annual Design Automation Conference*, pp. 304–309, ACM, 2009.

[11] V. V Zhirnov and M. J Marinella, "Memory technologies: Status and perspectives," in *Emerging Nanoelectronic Devices* (A. Chen, J. Hutchby, V. Zhirnov, and G. Bourianoff, eds.), John Wiley & Sons Ltd, 2015.

[12] V. V. Zhirnov, R. K. Cavin III, S. Menzel, E. Linn, S. Schmelzer, D. Braühaus, C. Schindler, and R. Waser, "Memory devices: Energy–space–time tradeoffs," *Proceedings of the IEEE*, vol. 98, no. 12, pp. 2185–2200, 2010.

[13] "(2013) The International Technology Roadmap for Semiconductos." `www.itrs.net`, 2013.

[14] Micron, "16 nm triple-level cell (TLC) NAND." `http://www.micron.com/products/nand-flash/tlc-nand`, 2015.

[15] Micron, "3D NAND Flash Memory." `http://www.micron.com/about/innovations/3d-nand`, 2015.

[16] Samsung, "V-NAND Technology." `http://www.samsung.com/semiconductor/products/flash-storage/v-nand/`, 2015.

[17] M. J. Marinella and V. V. Zhirnov, "Emerging memory devices: Assessment and benchmarking," *Emerging Nanoelectronic Devices*, pp. 246–276, 2014.

[18] A. Makosiej, O. Thomas, A. Amara, and A. Vladimirescu, "CMOS SRAM scaling limits under optimum stability constraints," in *Circuits and Systems (ISCAS), 2013 IEEE International Symposium on*, pp. 1460–1463, IEEE, 2013.

[19] A. Chen, J. Hutchby, V. V. Zhirnov, and G. Bourianoff, "Outlook for nanoelectronic devices," in *Emerging Nanoelectronic Devices* (A. Chen, J. Hutchby, V. Zhirnov, and G. Bourianoff, eds.), John Wiley & Sons Ltd, 2015.

[20] T.-Y. Liu, T. H. Yan, R. Scheuerlein, Y. Chen, J. K. Lee, G. Balakrishnan, G. Yee, H. Zhang, A. Yap, J. Ouyang, *et al.*, "A 130.7 mm^2 2-Layer 32-Gb ReRAM Memory Device in 24-nm Technology," *Solid-State Circuits, IEEE Journal of*, vol. 49, no. 1, pp. 140–153, 2014.

[21] S. Yu, B. Gao, Z. Fang, H. Yu, J. Kang, and H.-S. P. Wong, "A neuromorphic visual system using RRAM synaptic devices with sub-pJ energy and tolerance to variability: Experimental characterization and large-scale modeling," in *Electron Devices Meeting (IEDM), 2012 IEEE International*, pp. 10–4, IEEE, 2012.

[22] H.-S. P. Wong, H.-Y. Lee, S. Yu, Y.-S. Chen, Y. Wu, P.-S. Chen, B. Lee, F. T. Chen, and M.-J. Tsai, "Metal-oxide RRAM," *Proceedings of the IEEE*, vol. 100, no. 6, pp. 1951–1970, 2012.

[23] D. B. Strukov, G. S. Snider, D. R. Stewart, and R. S. Williams, "The missing memristor found," *Nature*, vol. 453, no. 7191, pp. 80–83, 2008.

[24] H. Wei, M. Shulaker, H.-S. P. Wong, and S. Mitra, "Monolithic three-dimensional integration of carbon nanotube FET complementary logic circuits," in *Electron Devices Meeting (IEDM), 2013 IEEE International*, pp. 19–7, IEEE, 2013.

[25] M. M. Shulaker, K. Saraswat, H.-S. P. Wong, and S. Mitra, "Monolithic three-dimensional integration of carbon nanotube FETs with silicon CMOS," in *VLSI Technology (VLSI-Technology): Digest of Technical Papers, 2014 Symposium on*, pp. 1–2, IEEE, 2014.

[26] M. M. Shulaker, T. F. Wu, A. Pal, L. Zhao, Y. Nishi, K. Saraswat, H.-S. P. Wong, and S. Mitra, "Monolithic 3d integration of logic and memory: Carbon nanotube FETs, resistive RAM, and silicon FETs," in *Electron Devices Meeting (IEDM), 2014 IEEE International*, pp. 27–4, IEEE, 2014.

[27] D. M. Gordon, "The ecology of collective behavior," *PLoS Biol*, vol. 12, no. 3, p. e1001805, 2014.

[28] C. Mead, "Neuromorphic electronic systems," *Proceedings of the IEEE*, vol. 78, no. 10, pp. 1629–1636, 1990.

[29] D. H. Ackley, G. E. Hinton, and T. J. Sejnowski, "A learning algorithm for boltzmann machines," *Cognitive Science*, vol. 9, no. 1, pp. 147–169, 1985.

[30] P. Kanerva, *Sparse Distributed Memory.* MIT Press, 1988.

[31] P. Kanerva, "Hyperdimensional computing: An introduction to computing in distributed representation with high-dimensional random vectors," *Cognitive Computation*, vol. 1, no. 2, pp. 139–159, 2009.

[32] B. A. Olshausen and D. J. Field, "Emergence of simple-cell receptive field properties by learning a sparse code for natural images," *Nature*, vol. 381, no. 6583, pp. 607–609, 1996.

[33] D. Hernandes, "The Man Behind the Google Brain: Andrew Ng and the Quest for the New AI." `http://www.wired.com/2013/05/neuro-artificial-intelligence/`, 2013.

[34] R. Sarpeshkar, *Ultra-Low Power Bioelectronics.* Cambridge University Press, 2010.

[35] D. H. Brainard, D. R. Williams, and H. Hofer, "Trichromatic reconstruction from the interleaved cone mosaic: Bayesian model and the color appearance of small spots," *Journal of Vision*, vol. 8, no. 5, p. 15, 2008.

[36] A. A. Faisal, L. P. Selen, and D. M. Wolpert, "Noise in the nervous system," *Nature Reviews Neuroscience*, vol. 9, no. 4, pp. 292–303, 2008.

[37] L. B. Buck, "Unraveling the sense of smell (nobel lecture)," *Angewandte Chemie International Edition*, vol. 44, no. 38, pp. 6128–6140, 2005.

[38] J. Lazzaro and C. A. Mead, "A silicon model of auditory localization," *Neural Computation*, vol. 1, no. 1, pp. 47–57, 1989.

[39] IHS, "Emerging Sensors in Handsets and Tablets Report." `https://technology.ihs.com/452713/emerging-sensors-in-handsets-tablets-report-2014`, 2014.

[40] D. J. Field, "What is the goal of sensory coding?," *Neural Computation*, vol. 6, no. 4, pp. 559–601, 1994.

[41] B. A. Olshausen and D. J. Field, "Sparse coding with an overcomplete basis set: A strategy employed by v1?," *Vision Research*, vol. 37, no. 23, pp. 3311–3325, 1997.

[42] T. K. Alkasab, J. White, and J. S. Kauer, "A computational system for simulating and analyzing arrays of biological and artificial chemical sensors," *Chemical Senses*, vol. 27, no. 3, pp. 261–275, 2002.

[43] L. B. Harberly, "The Olfactory Cortex," in *The Synaptic Organization of the Brain* (G. M. Shepherd, ed.), Oxford University Press, 1998.

[44] C. Poo and J. S. Isaacson, "Odor representations in olfactory cortex:sparse coding, global inhibition, and oscillations," *Neuron*, vol. 62, no. 6, pp. 850–861, 2009.

[45] H. B. Barlow, "The ferrier lecture, 1980: Critical limiting factors in the design of the eye and visual cortex," *Proceedings of the Royal Society of London B: Biological Sciences*, vol. 212, no. 1186, pp. 1–34, 1981.

[46] B. Olshausen, "VS265 Lecture Notes." University of California at Berkeley.

[47] J. Hertz, A. Krogh, and R. G. Palmer, *Introduction to the Theory of Neural Computation*, vol. 1. Basic Books, 1991.

[48] Y. Dan, J. J. Atick, and R. C. Reid, "Efficient coding of natural scenes in the lateral geniculate nucleus: experimental test of a computational theory," *The Journal of Neuroscience*, vol. 16, no. 10, pp. 3351–3362, 1996.

[49] H. . Barlow, "Single units and sensation: A neuron doctrine for perceptual psychology?," *Perception*, vol. 1, pp. 371–394, 1972.

[50] A. B. Lee, K. S. Pedersen, and D. Mumford, "The nonlinear statistics of high-contrast patches in natural images," *International Journal of Computer Vision*, vol. 54, no. 1-3, pp. 83–103, 2003.

[51] L. Wiskott and T. J. Sejnowski, "Slow feature analysis: Unsupervised learning of invariances," *Neural Computation*, vol. 14, no. 4, pp. 715–770, 2002.

[52] B. A. Olshausen and D. J. Field, "Sparse coding of sensory inputs," *Current Opinion in Neurobiology*, vol. 14, no. 4, pp. 481–487, 2004.

[53] J. Perez-Orive, O. Mazor, G. C. Turner, S. Cassenaer, R. I. Wilson, and G. Laurent, "Oscillations and sparsening of odor representations in the mushroom body," *Science*, vol. 297, no. 5580, pp. 359–365, 2002.

[54] M. R. DeWeese, M. Wehr, and A. M. Zador, "Binary spiking in auditory cortex," *The Journal of Neuroscience*, vol. 23, no. 21, pp. 7940–7949, 2003.

[55] L. Thompson and P. Best, "Place cells and silent cells in the hippocampus of freely-behaving rats," *The Journal of Neuroscience*, vol. 9, no. 7, pp. 2382–2390, 1989.

[56] J. J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities," *Proceedings of the National Academy of Sciences*, vol. 79, no. 8, pp. 2554–2558, 1982.

[57] J. Hawkins and D. George, "Hierarchical temporal memory: Concepts, theory and terminology," tech. rep., Technical Report, Numenta, 2006.

[58] B. J. Hosticka, "Performance comparison of analog and digital circuits," *Proceedings of the IEEE*, vol. 73, no. 1, pp. 25–29, 1985.

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

[60] A. G. Andreou and P. M. Furth, "An information theoretic framework for comparing the bit-energy of signal representations at the circuit level," *Low-Voltage/Low-Power Integrated Circuits and Systems*, vol. 8, 1998.

[61] R. Sarpeshkar, "Analog versus digital: extrapolating from electronics to neurobiology," *Neural Computation*, vol. 10, no. 7, pp. 1601–1638, 1998.

[62] N. R. Shanbhag, "A mathematical basis for power-reduction in digital VLSI systems," *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 44, no. 11, pp. 935–951, 1997.

[63] T. M. Cover and J. A. Thomas, *Elements of Information Theory.* John Wiley & Sons, 2012.

[64] A. F. Murray, D. Del Corso, and L. Tarassenko, "Pulse-stream VLSI neural networks mixing analog and digital techniques," *IEEE Transactions on Neural Networks*, vol. 2, no. 2, pp. 193–204, 1991.

[65] E. J. Severin, B. J. Doleman, and N. S. Lewis, "An investigation of the concentration dependence and response to analyte mixtures of carbon black/insulating organic polymer composite vapor detectors," *Analytical Chemistry*, vol. 72, no. 4, pp. 658–668, 2000.

[66] W. Mickelson, A. Sussman, and A. Zettl, "Low-power, fast, selective nanoparticle-based hydrogen sulfide gas sensor," *Applied Physics Letters*, vol. 100, no. 17, p. 173110, 2012.

[67] FIGARO, "TGS8100: Air Contaminants Detector." `http://www.figarosensor.com/feature/tgs8100.html`.

[68] Cambridge CMOS Sensors, "CCS80x Series: MOX Sensors for TVOC, CO, Ethanol Detection." `http://ccmoss.com/sensors`.

[69] BOSCH, "BME680: Integrated Environmental Sensor." `https://www.bosch-sensortec.com/en/homepage/products_3/environmental_sensors_1/bme680/bme680_2`, 2015.

[70] T. C. Pearce, S. S. Schiffman, H. T. Nagle, and J. W. Gardner, *Handbook of Machine Olfaction: Electronic Nose Technology.* John Wiley & Sons, 2006.

[71] N. Yamazoe and K. Shimanoe, "Theory of power laws for semiconductor gas sensors," *Sensors and Actuators B: Chemical*, vol. 128, no. 2, pp. 566–573, 2008.

[72] FIGARO, "TGS822: Organic Solvent Vapors Detector." `http://www.figarosensor.com/products/822pdf.pdf`.

[73] K. Shen, S. Tootoonian, and G. Laurent, "Encoding of mixtures in a simple olfactory system," *Neuron*, vol. 80, no. 5, pp. 1246–1262, 2013.

[74] M. Stopfer, V. Jayaraman, and G. Laurent, "Intensity versus identity coding in an olfactory system," *Neuron*, vol. 39, no. 6, pp. 991–1004, 2003.

[75] J. J. Hopfield, "Pattern recognition computation using action potential timing for stimulus representation," *Nature*, vol. 376, no. 6535, pp. 33–36, 1995.

[76] J. M. Rabaey, A. P. Chandrakasan, and B. Nikolic, *Digital Integrated Circuits*, vol. 2. Prentice hall Englewood Cliffs, 2002.

[77] B. Razavi, *Design of Analog CMOS Integrated Circuits.* Tata McGraw-Hill Education, 2002.

[78] F. Cannillo, C. Toumazou, and T. Lande, “Bulk-drain connected load for subthreshold MOS current-mode logic,” *Electronics Letters*, vol. 43, no. 12, pp. 662–664, 2007.

[79] A. Tajalli, E. Vittoz, Y. Leblebici, and E. Brauer, “Ultra-low power subthreshold current-mode logic utilising PMOS load device,” *Electronics Letters*, vol. 43, no. 17, pp. 911–913, 2007.

[80] A. Tajalli, Y. Leblebici, and E. J. Brauer, “Implementing ultra-high-value floating tunable CMOS resistors,” *Electronics letters*, vol. 44, no. 5, pp. 349–351, 2008.

[81] J. Lazzaro and J. Wawrzynek, *Low-Power Silicon Neurons, Axons and Synapses.* Springer, 1994.

[82] M. Bazes, “Two novel fully complementary self-biased CMOS differential amplifiers,” *Solid-State Circuits, IEEE Journal of*, vol. 26, no. 2, pp. 165–168, 1991.

[83] B. Gilbert, “A monolithic 16-channels analog array normalizer,” *Solid-State Circuits, IEEE Journal of*, vol. 19, no. 6, pp. 956–963, 1984.

[84] S. Chakrabartty and G. Cauwenberghs, “Sub-microwatt analog VLSI trainable pattern classifier,” *Solid-State Circuits, IEEE Journal of*, vol. 42, no. 5, pp. 1169–1179, 2007.

[85] D. E. Rumelhart, J. L. McClelland, P. R. Group, *et al.*, *Parallel Distributed Processing*, vol. 1. IEEE, 1988.

[86] T. Plate *et al.*, “Holographic reduced representations,” *Neural Networks, IEEE Transactions on*, vol. 6, no. 3, pp. 623–641, 1995.

[87] E. J. Candès, “Compressive sampling,” in *Proceedings of the International Congress of Mathematicians*, vol. 3, pp. 1433–1452, Madrid, Spain, 2006.

[88] J. Hopfield, “Computing with action potentials,” in *Advances in Neural Information Processing Systems 10: Proceedings of the 1997 Conference*, vol. 10, p. 166, MIT Press, 1998.

[89] J. J. Hopfield and C. D. Brody, “What is a moment?cortical sensory integration over a brief interval,” *Proceedings of the National Academy of Sciences*, vol. 97, no. 25, pp. 13919–13924, 2000.

[90] J. J. Hopfield and C. D. Brody, “What is a moment? transient synchrony as a collective mechanism for spatiotemporal integration,” *Proceedings of the National Academy of Sciences*, vol. 98, no. 3, pp. 1282–1287, 2001.

[91] C. D. Brody and J. Hopfield, “Simple networks for spike-timing-based computation, with application to olfactory processing,” *Neuron*, vol. 37, no. 5, pp. 843–852, 2003.

[92] L. A. Jaeckel, *An alternative design for a Sparse Distributed Memory.* Research Institute for Advanced Computer Science, NASA Ames Research Center, 1989.

[93] T. Kohonen, "Correlation matrix memories," *Computers, IEEE Transactions on*, vol. 100, no. 4, pp. 353–359, 1972.

[94] H. Wei, T. F. Wu, D. Sekar, B. Cronquist, R. F. Pease, and S. Mitra, "Cooling three-dimensional integrated circuits using power delivery networks," in *Electron Devices Meeting (IEDM), 2012 IEEE International*, pp. 14–2, IEEE, 2012.

[95] Y. Y. Liauw, Z. Zhang, W. Kim, A. E. Gamal, and S. S. Wong, "Nonvolatile 3d-fpga with monolithically stacked rram-based configuration memory," in *Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2012 IEEE International*, pp. 406–408, IEEE, 2012.

[96] Z. Zhang, Y. Wu, H.-S. P. Wong, and S. S. Wong, "Nanometer-Scale RRAM," *Electron Device Letters, IEEE*, vol. 34, no. 8, pp. 1005–1007, 2013.

[97] S. Yu, H.-Y. Chen, B. Gao, J. Kang, and H.-S. P. Wong, "HfOx-based vertical resistive switching random access memory suitable for bit-cost-effective three-dimensional cross-point architecture," *ACS nano*, vol. 7, no. 3, pp. 2320–2325, 2013.

[98] X. Guan, S. Yu, and H.-S. P. Wong, "On the switching parameter variation of metal-oxide RRAMPart I: Physical modeling and simulation methodology," *Electron Devices, IEEE Transactions on*, vol. 59, no. 4, pp. 1172–1182, 2012.

[99] Z. Jiang, H. Li, J. H. Engel, S. Yu, X. Guan, and P. Wong, "Stanford RRAM Model." `https://nano.stanford.edu/stanford-rram-model`, 2015.

[100] G. S. Snider, "Self-organized computation with unreliable, memristive nanodevices," *Nanotechnology*, vol. 18, no. 36, p. 365202, 2007.

[101] G. S. Snider, "Spike-timing-dependent learning in memristive nanodevices," in *Nanoscale Architectures, 2008. NANOARCH 2008. IEEE International Symposium on*, pp. 85–92, IEEE, 2008.

[102] P. Földiak, "Forming sparse representations by local anti-hebbian learning," *Biological Cybernetics*, vol. 64, no. 2, pp. 165–170, 1990.

[103] D. Sloughter, *The Calculus of Functions of Several Variables.* Furman University, 2001.