Neuromorphic Computation Circuits for Ultra-Dense Mobile Platforms

by

Laura E. Fick

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Electrical Engineering)
in The University of Michigan
2017

Doctoral Committee:

Professor Dennis M. Sylvester, Chair
Professor David T. Blaauw
Professor Mark B. Moldwin
Associate Professor Zhengya Zhang

To my husband.

His love, support, encouragement, and invaluable advice made this possible.

TABLE OF CONTENTS

LIST OF FIGURES

Figure

LIST OF TABLES

ABSTRACT

Neuromorphic Computation Circuits for Ultra-Dense Mobile Platforms

by

Laura E. Fick

Chair: Dennis M. Sylvester

Ultra-dense mobile platforms have the potential to be a ubiquitous form of computing. From low-power voice recognition for wearables to high speed image recognition for self-driving cars, the mobile platform space is large, with a wide range of requirements. The different area, power, and speed requirements necessitate a computing platform that is highly scalable, ultra-dense, and with the potential for very low power consumption. Neural networks are currently the leading algorithms for recognition problems - taking in many inputs and correlating against learned weights, these algorithms are able to make informed decisions off of a large amount of generalized data.

Neural network algorithms are facilitating the advancement of intelligent systems through highly parallel, large-scale processing of sensor inputs. These algorithms are memory intensive, reading out neuron weights for every computation. In digital implementations, total energy consumption is dominated by the amount and frequency of memory reads. Each neuron typically has between hundreds and thousands of 8-bit synaptic weights, and a neuron layer has hundreds of neurons, resulting in thousands of weights per layer. Each weight is accessed once per neuron computation, or 1 out of N computations for N neurons. Because these weight accesses are one time use there is little temporal locality that can be exploited to reduce the total energy consumption. Implementing state-of-the-art neural networks in ultra-dense form factors will require large energy improvements over current architectures.

For example, Amazon's best-in-class speech recognition algorithm used in the Echo requires more than 1 Watt of power. Porting this algorithm to a smartwatch device would require at least a 100× energy reduction over current approaches.

Traditionally we could expect constant energy improvements through ideal constant field scaling. However, achieving 100× reduction would take 35 process steps ($S = \sqrt{2}Step$), as energy scales roughly at $1/S^3$. Recent years have seen an end to ideal constant field scaling, with supply voltage and threshold voltage scaling tapering off, providing diminishing returns. This dissertation works to address the limitations of memory accesses and energy inherent to neural networks through storing weights in on-chip non-volatile arrays and combining read-out and calculation current to amortize energy costs. The proposed circuits are ultra-dense and low-power, with the potential for very low power consumption through subthreshold calculation.

To further advance the development of neural networks in the ultra-dense mobile platform space, this work proposes four projects: a high-speed ultra-dense neural network accelerator, an ultra-low power subthreshold neural network accelerator, an ultra-dense scalable sensor interface, and a low area low power gigabit receiver equalizer for chip-to-chip communication.

CHAPTER 1

Introduction

This introduction briefly discusses current computing, information and algorithm trends, and the importance of research in the field of neural networks in power and area constrained systems.

1.1 Ultra-Dense Computing

The past 50 years have seen a continuous scaling of process technologies in accordance with Gordon Moore's observation that the number of transistors on a single die would double every two years with fixed cost [1]. Less discussed, but considered a partial corollary to Moore's Law, is Bell's Law of computer classes [2]. This states that a new computer class is formed every decade, consisting of lower cost, smaller components. This prediction has been demonstrated in scaling from the original main frame computers of the past [3, 4], to personal computers of the 1990s, down to today's smart phones. The computing power that currently fits into our pockets is $432,000\times$ more powerful than that of the 1960s mainframes which occupied 2,326 cubic feet of space and drew 170,000 watts of power, an increase in $FLOPS/W/mm^3$ of nearly $1.7E+19\times$, representing the amount of performance per watt for a given volume [5]. As shown in Figure 1.1, It is predicted that the next disruptive class of computers will come in the form of wireless sensor platforms with a variety of applications [6–9]. These devices will have a vast amount of computing power in a form factor on the order of centimeters, necessitating ultra-dense, high performance computing circuitry.

Applications for wireless sensor platforms range from security (facial recognition on video feeds), to personal use (voice and facial recognition in smart devices) [10]. These applications also have the ability to process multiple feeds of information from different input sensors [11]. A wireless platform that has both a camera and microphone can process both streams of data to determine who is interacting with the device and what they are saying - allowing for personalized assistance, interactions, analysis and response. These types of systems facilitate further development of smart devices, intelligent and responsive home features and other advances within the internet of things (IOT).

Development of ultra-dense, high performance sensor platforms requires very low area, low power circuits. These designs must implement algorithms to process very large data sets for manipulation and analysis, take in multiple sensor inputs, and allow high throughput communication for real-time interaction.

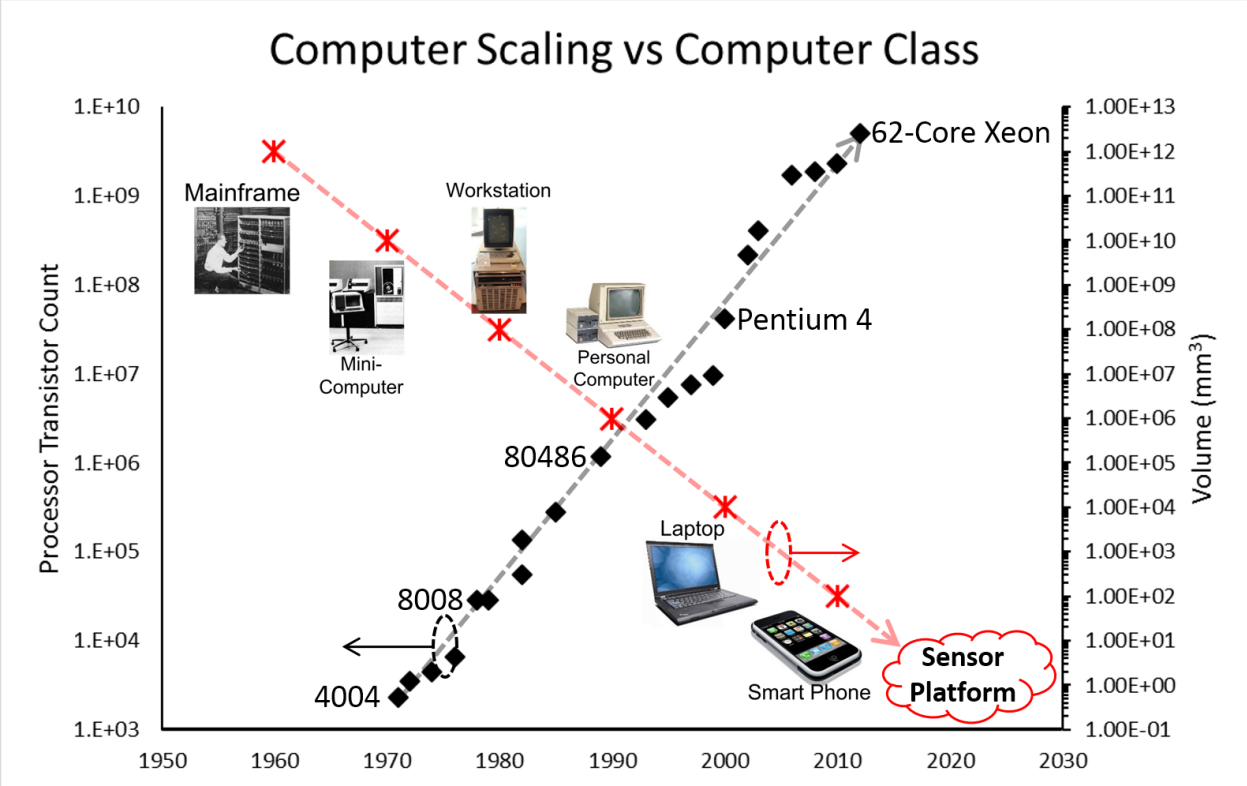


Figure 1.1: **Scaling of Form-Factor and Processing Power.** Moore's Law and Bell's Law predict the scaling of CMOS technology, which facilitates the exponential increase in transistor count, as well as the exponential decrease in computer volume. Together, these relationships result in ever shrinking computer classes with exponentially more computing power [12, 13].

1.2 Technology Process Scaling

Very-Large Scale Integration (VLSI) process scaling over the past 60 years has allowed for large improvements in computing power, performance and area. Moore's Law has become a self-fulfilling prophecy in setting guidelines for technology improvements, resulting in a constant exponential increase in transistor count. However, the number of transistors on a die is not the same as system performance. Figure 1.2 shows that while transistor count has consistently increased, single-thread performance and frequency are starting to saturate, or have already saturated. Additionally, power consumption has saturated because processors have already reached the maximum power limit of 125 W due to thermal design power (TDP) constraints [14]. Due to this limitation, perfect transistor scaling has not been achievable, as leakage increases caused by lower threshold voltage (V_{th}), increased V_{th} variation and drain induced barrier lowering result in greater power density [15]. This increased power density must be offset in other ways, typically by lowering frequency or decreasing die size [16].

Global and local interconnections affect system performance as well. Table 1.1 shows how the resistor-capacitor (RC) delay of both global and local interconnects scale with each process node. Local interconnect benefits only slightly from process scaling, as the improvement is limited to $\sqrt{S}$. Since transistor performance scales as S, this results in relatively slower local connections. Because global interconnects span the length of a die, and die size is assumed constant in scaling estimations, these interconnects actually see a $\sqrt{S}$ increase in RC delay. This results in even slower global interconnects relative to device performance. In addition to relative and total interconnect slowing, process scaling results in a reduction in the relative availability of global interconnections, as this scales with $1/S$.

Process scaling results in increased computational power and number of transistors, allowing modern computers to solve increasingly hard and complex problems. However, interconnect scaling limits the types of algorithms that can be effectively implemented using traditional digital computing strategies. These algorithms are limited by the delays caused by increasingly resistive wires, and the limited area for interconnections both locally and globally.

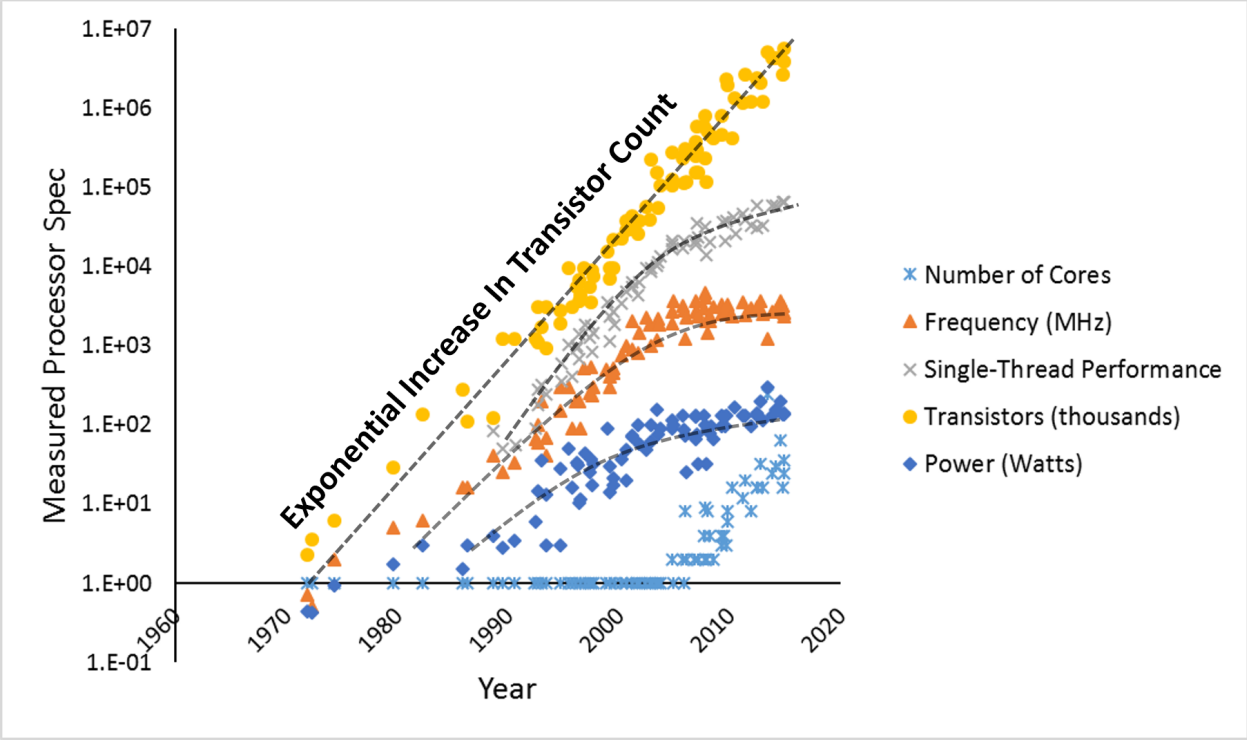


Figure 1.2: **VLSI Process Scaling Trends.** Moore's law has dictated the scaling of process nodes over the past 50 years, resulting in an exponential increase in transistor counter per node. Because of imperfect scaling, this does not result in continuously increasing single-thread performance or frequency [17].

1.3 Big Data Computation

Data collection is generated from a variety of sources, and provides information on a multitude of different topics. Sensors on mobile platforms collect information on climate [19], surveillance [20] and health metrics [21]. Pictures, posts and videos to social media sites inundate servers with personal information and provide the ability to train massive algorithms for highly accurate facial recognition [22]. Purchase transaction records allow large companies to analyze buying patterns of customers to send targeted advertisements and to make strategic decisions [23]. In the digital world, data are generated in droves every day and stored in server farms for analysis by all major companies, trading firms, and governments. It is currently estimated that we create 2.5 quintillion bytes of data per day, and that 90% of data in the world today has been created in the last 2 years alone [24].

Effectively using the information generated from all of this data requires a massive amount

Table 1.1: **Scaling Effects on Interconnect.** Comparison between MOS scaling with local and global interconnect. Global interconnect scaling becomes slower with lower technology nodes, and available interconnect is reduced [18].

Parameter	Sensitivity	Scale Factor
Scaling Parameters		
Width: W		$1/S$
Spacing: s		$1/S$
Thickness: t		$1/S$
Interlayer Oxide Height: h		$1/S$
Die Size		D_c
Local Interconnect Characteristics		
Length: l		$1/S$
Unrepeated Wire RC Delay	$L^2 t_{wu}$	1
Repeated Wire Delay	lt_{wr}	$sqrt(1/S)$
Energy Per Bit	lE_w	$1/S^3$
Global Interconnect Characteristics		
Length: l		D_c
Unrepeated Wire RC Delay	$l^2 t_{wu}$	$S^2 D_c^2$
Repeated Wire Delay	lt_{wr}	$D_c sqrt(S)$
Energy Per Bit	lE_w	D_c/S^2

of storage, infrastructure and analytical software. It is currently estimated that the total number of data centers in the world will peak at 8.6 million by 2017, occupying around 1.94 billion square feet of space [25], and will consume 140 terrawatt-hours of electricity [24]. This amount of data and information is infeasible to manipulate with traditional computation strategies. High performance servers would take days to process even a small percentage of the data on a given topic using standard analysis procedures. Mobile platforms are limited to centimeter scale volumes for full systems and are not guaranteed to have internet access, limiting the types of algorithms that can be implemented. Specialized algorithms and architectures are required to harness the power of this big data revolution.

Standard computing algorithms cannot possibly store all of the required data from these big data sets, nor can they implement the required local and global interconnections that are necessary for processing this amount of information. Additionally, the time required to sort through the data, and the inherent inaccuracy and noise of the data are not suited to traditional approaches [26]. Algorithms built to analyze big data problem sets include genetic algorithms, support vector machines (SVMs), nearest neighbor estimations, and deep neural networks [27]. These types of algorithms spend a large amount of offline training time pre-processing data and understanding trends and characteristics of a given data set. Essentially,

these algorithms break down the data set into a minimal amount of information for storage, similar to primary component analysis. Then, during operation, the algorithms use that smaller amount of information in order to generate real-time estimates of input relationships with the stored data. As an example, an algorithm trained on millions of photos of human faces would take in a photo or video frame. It would then send that image through a network of highly interconnected stored data values that represent all of the trained images to determine how similar the input is to a given known value (*i.e.* Is this a human face?).

Benchmarking these types of algorithms is important for understanding and comparing which types are more suited to different data sets, and to different sizes of data sets. Smaller data sets with less complex data may have many algorithms that produce very good classification results. When increased by a couple orders of magnitude, these data sets can show large differences in the efficacy of different implementations. The ImageNet Large Scale Visual Recognition Challenge (ILSVRC) has been run annually for 5 years and is the standard benchmark for large-scale object recognition. This competition utilizes a data set of 1,461,406 (r,g,b) images and 1,000 object classes. It requires competitors to take in a series of images and determine which object appears in that image, out of the given 1,000 possibilities. Guesses are given as a soft-max output of the algorithm's top-5 guesses, giving the competitors a 0.5% chance of probabilistically guessing correctly [28].

As shown in Figure 1.3, the reported error percentages of the winning entries over the last 5 competitions has decreased from 28.2% down to 6.7% for a total reduction of 4.2×. Additionally, the algorithms used by entrants have converged onto an optimal strategy. During the 2010 competition there was a large amount of variation in reported results, ranging from a high of 78.7% error down to the low of 28.2%. Most algorithms submitted, including the winning entry, used some type of assisted SVM. The 2012 competition showed a clear divergence in both strategy and outcome, with all but one team using SVMs the winning team was able to achieve an error of only 16.4%, nearly 10% better than the runner up of 26.2%. The winning team in this competition diverged from the typical SVM topology and used a deep, multi-layer, convolutional neural network (CNN). Years 2013 and 2014 saw further improvements and clustering of results, with nearly all teams implementing some form of multi-layer CNN from this point forward. These results show that while SVMs can

achieve moderate results on very large data sets, neural networks are currently the optimal algorithm for such tasks.

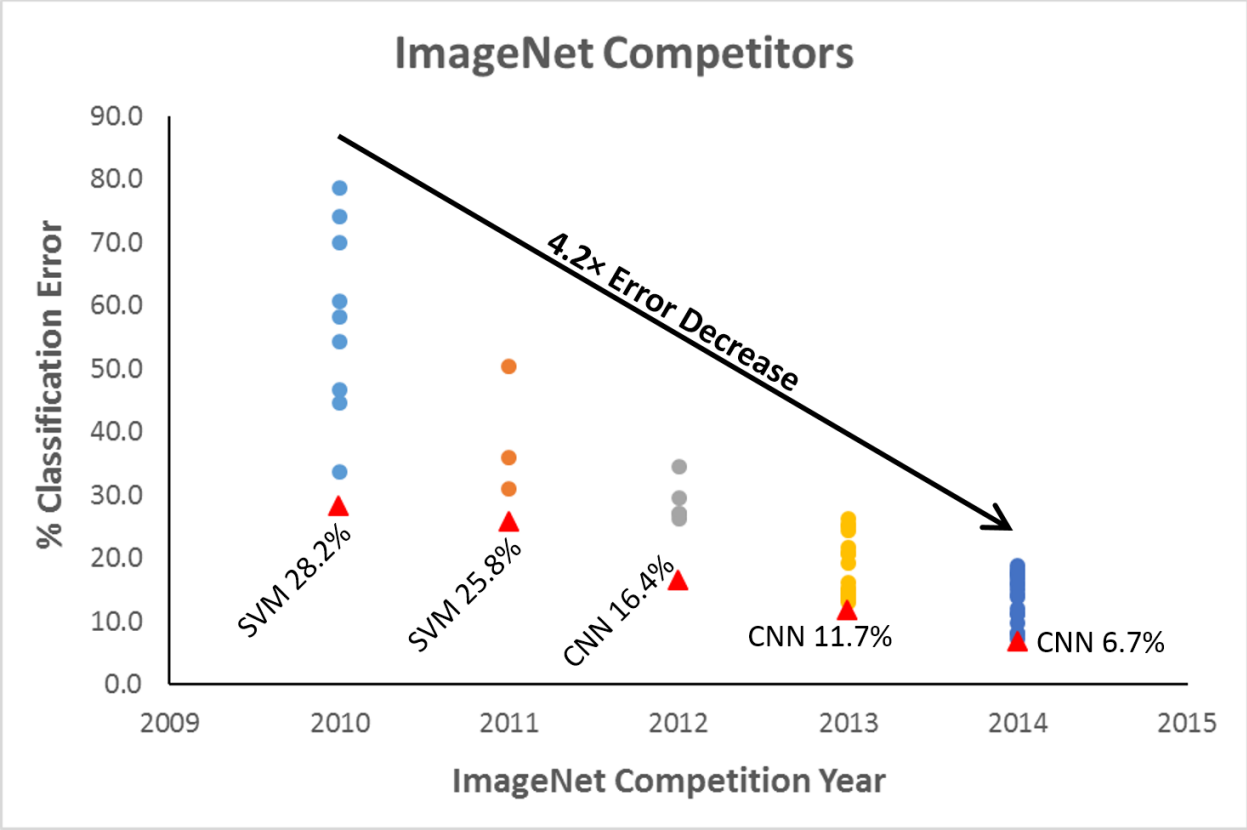


Figure 1.3: **ImageNet Competition Results.** The benchmark for image recognition algorithms is ILSVRC, and the results of the past 5 years of this annual competition are shown. Trending downward, with a 4.2× decrease in the error percentage of the winning entry, the competition results show that neural networks are able to significantly out-perform SVMs for large scale recognition problems [28].

1.4 Neural Network Algorithms

Neural networks have long been proposed as a way to mimic the power of the mammalian brain. In 1949 McCulloch and Pitts proposed the perceptron [29], a simple computational model to represent neuron functionality. A biological neuron is an electrically excitable cell that takes inputs in through axon terminals connected to any number of other neurons via synapses. These synaptic connections have an associated strength (*i.e.* How strongly they affect the output of the subsequent neuron) and can be either *excitatory* or *inhibitory*,

meaning that they either increase or decrease activity in the target neuron. The synaptic connections take outputs from previous neurons, sum them based on their strength and polarity along the *axon*, and the target neuron produces a response depending on the strength of the signal it received. McCulloch and Pitts translated this biological explanation into a basic multiply-accumulate (MAC) function. Their proposal takes in N inputs, multiplies each by a corresponding stored weight ω_n, sums the products and then applies a neuron response function to the summed value. This model is relatively simple and can classify a limited number of data sets, most notably it was unable to learn an exclusive-or (XOR) function [30]. This issue was overcome by further mimicking the brain by developing networks of multi-layer perceptrons. Current work in this field has produced a variety of neural network types, the most popular being deep neural networks (many layers) that employ convolution.

Convolutional neural networks work on small patches (*receptive fields*) of a given input and are able to re-use the stored weights on all similarly sized patches in the full image (*visual field*) to produce an output [31]. The patches are stepped across the full input image in order to apply the set of stored filters to all receptive fields. These networks allow for analysis of large input images or data sets with significantly reduced memory requirements. Convolutional networks employ a significant amount of overlap in their receptive fields, and often include max-pooling and fully connected layers as well. A max-pooling layer takes the maximum value in a small portion of the output of the previous neural network layer, and is used to reduce the dimensionality of the overall problem. A fully connected layer is also used to reduce dimensionality and is often used as the final layers of a deep neural network (DNN). In these layers, as the name implies, all inputs are weighted and connected to all outputs, unlike the convolutional layers. Effectively, a convolutional network works to find increasingly higher-level features in a data set, then takes those features and determines which ones are present in the possible outputs of the network. These features are then sent to fully-connected layers which work to reduce the data set down to a single decision. State-of-the-art DNNs often have 2-3 convolutional layers, 2 max-pooling layers and 2-3 fully connected layers.

Synaptic weights used to identify features in given data sets are calculated through a series of training exercises [32–34]. In these flows, synaptic weights are incrementally adjusted

based on error calculations. Large data sets are split into *training* and *testing* categories. Inputs from the training category are fed into the network and weights are adjusted via *backpropagation* (*i.e.* Backward propagation of errors). This is known as a supervised learning method [35], because the training data set must be "labeled" so that the network knows when its predicted answer is right or wrong. With this information, the network can then propagate errors backward through the network using the training input image as the target in order to generate the difference between the ideal and actual inputs and outputs of all network layers. Using this error calculation, weights can be incrementally adjusted by calculating a gradient through multiplying the output error and the input activation, then subtracting some percentage of the value from the current weight to achieve $\omega_{(t+1)}$, this is known as the *gradient descent* training method [36].

Gradient descent is not guaranteed to find the optimal set of weight values because it can only find local minima in cost functions, not global minima. To avoid getting stuck in a local minima, training algorithms can employ dithering (adding noise) to the weights [37], or dropout (random omission of neurons from the network) [38]. These methods hope to push the algorithm out of local minima to find a more optimal solution, as well as to prevent over-fitting of weights to trained data [39]. In practice, these training algorithms take days to run and are only feasible in high performance servers. As such, hardware implementations of neural networks should expect to receive a pre-trained set of weights and not require training updates on-chip, or in online operation.

Typical weight set sizes for large data sets (*i.e.* ImageNet) are on the order of 60 million unique values, requiring close to a week of training time, and a significant amount of on-chip weight storage for a hardware implementation. Using 8-bit weights and on-chip SRAM requires 60MB of storage, roughly $528mm^2$ of area in a 22nm technology. The next section explores ultra-dense, analog weight storage solutions for complementary metal-oxide semiconductor (CMOS) neural network implementations.

1.5 Hardware Neural Networks

1.6 Non-Volatile Memories

In order to store millions of on-chip synaptic weights, a highly dense non-volatile memory (NVM) with low read energy is needed. Density can be affected by the size of a single storage cell which is measured through form factor (F^2), and by the ability of a cell to store multiple bits of information in a single cell (multi-level cell (MLC) capability). A traditional SRAM memory implementation has a form factor of roughly $100F^2$ and is only capable of storing binary information. Comparing to a flash cell that has a form factor of $4F^2$ and can store 8-bit weight values results in a 200× reduction in area. One advantage of static random access memory (SRAM) is its scalability with process nodes. Flash technologies are notoriously difficult to scale, and often designs are required to be implemented in 90nm nodes and up. SRAM is available at all process nodes, allowing for implementations in the smallest processes (10nm and lower). Future work into non-volatile memory scaling will produce memories that are available in much newer nodes, and companies are currently developing advanced node non-volatile processes such as 45nm and 28nm. Silicon-Oxide-Nitride-Oxide-Silicon (SONOS), which is similar to flash but uses charge trapping on an insulated Oxide-Nitride-Oxide (ONO) layer, is currently in development for use in 28nm and shows true scaling properties in both size and voltage requirements for programming and erasing values.

Table 1.2 shows a detailed study of different memory solutions to determine the feasibility of each for neural network weight storage. In this comparison, SRAM and electrically programmable fuses (E-Fuse) require the largest F^2 area, both upwards of $100F^2$. SRAM and dynamic random access memory (DRAM) are both volatile memories, SRAM would require a re-write on boot-up, and DRAM requires near constant re-writing of weights to achieve MLC. Another concern with some memory solutions is whether the read operation is destructive, meaning that reading the cell disturbs the stored value to the point that it needs to be re-written afterward. Both ferroelectric random access memory (FeRAM) and DRAM are considered to have destructive read operations.

In terms of energy, the read operation is the only one that matters for neural network implementations. While all memories need to be written at some point, non- volatility allows for very long storage (low write/erase duty cycles), and once training has been optimized, weight updates are unnecessary. During operation, all stored weights in a given neuron will be read every cycle, with no weight values skipped, as this is the primary computation required for neural networks. Read energy is shown to be lowest in magnetoresistive random access memory (MRAM) and SONOS memories, at 0.02pJ and 0.11pJ respectively.

Table 1.2: **Non-Volatile Memory Comparison.** Estimates of feasibility and usefulness of different non-volatile and volatile memory options of neural network weight storage [40–44].

Memory Type	**Cell Size** (F^2)	**MLC**	**Read Speed (ns)**	**Read Energy (pJ)**	**Destructive Read**	**Non-Volatile**
SRAM	90-150	No	8	1	No	No
NAND Flash	4	Yes	60	300	No	Yes
NOR Flash	8-10	Yes	60	300	No	Yes
E-Fuse	300	Yes	100	80	No	Yes
PCRAM	5-8	Yes	50	100	No	Yes
FeRAM	18	Yes	80	1450	Yes	Yes
MRAM	10-20	No	30	0.02	No	Yes
ReRAM	6-10	Yes	8	2	No	Yes
DRAM	6-12	Yes	8	2	Yes	No
SONOS	37	Yes	28	0.11	No	Yes

CHAPTER 2

Massively Parallel Neuromorphic Computation

2.1 Motivation

Neural networks are currently the most efficient and accurate algorithms for big-data problems such as image and voice recognition [45, 46]. Traditional computing strategies struggle with the complexity and ambiguity of these types of data sets because imprecise input data and information are not well suited for Turing-machine style computation. The redundant precision of these Turing-machine computers costs energy and area, as many registers, adders and data paths are needed to represent even a small set of information. Some styles of computing can tolerate, and possibly benefit from the randomness of these large, imprecise data sets [47].

Leading software-based neural network implementations can achieve 15.3% error on data sets consisting of 1.2 million high-res color photographs with 1000 possible object classes [48]. This is comparable to the accuracy with which humans can recognize the same images. In order for mobile platforms such as cell phones, robots, or unmanned crafts to properly recognize and analyze surroundings we need algorithms that are as good or better than their human counterparts. Additionally, we need these algorithms to be mapped to hardware efficiently and with little loss in accuracy. Figure 2.1 shows three prevalent applications for hardware neural networks: voice recognition, self-driving cars, and augmented reality for smart phones. The primary source of power from deep neural networks is from weight accesses and dot product calculations. As shown in the equations in Figure 2.1, both the

number of weight accesses and the number of multiply accumulate (MAC) functions are dependent on the number of neurons in the system N. The proposed system aims to reduce power and energy in hardware neural networks by implementing on-chip embedded synaptic weights, with highly parallel in-memory analog calculation to amortize synaptic weight reads and calculations. Completely eliminating synaptic weight accesses reduces memory reads by $1/(N+1)\times$. This is a significant amount when considering state-of-the-art deep neural networks (DNNs) often use between 64 and 512 neurons (N).

Biologists have long known that the mammallian brain is capable of highly dense and efficient processing. The average human brain contains about 100 billion neurons [49], and is capable of recalling years worth of information, data, images, voices, and other sensory inputs. Since 1957, scientists have been working toward modeling and harnessing the power of the human brain through bio-inspired neural network algorithms [50]. While these original works focused on modeling single neuron structures, current state-of-the-art neural networks use multi-layer, convolutional topologies. Convolutional refers to the fact that the weights in this network are re-used, where small sub- regions of the array are called *receptive fields*. These sub-regions are tiled together to cover the entire *visual field*, allowing the network to effectively have many times more neurons than are physically present. Through employing many receptive fields (filters), as shown in Figure 2.2, these DNNs can represent a complex problem (What object is this?), with a series of smaller problems (Is there a horizontal line in this area? What shapes are made out of these lines?)

Efficiently mapping these state-of-the-art software neural networks to hardware has not yet been achieved on a large scale. Previous approaches follow two different strategies: all-digital computation, or analog charge-summing. Both of these strategies suffer from the same fundamental problems of area/power efficient weight storage and energy required to access these weights.

Digital neuron approaches store weights in shift registers, latches, or on-chip SRAM - writing all values into digital memory from off-chip non-volatile storage during boot-up, or accessing memory from off-chip when needed. Analog implementations typically store weights [51] via resistors [52], capacitors [53], or electrically erasable programmable read-only memory (EEPROM) [54]. Analog weight storage also requires off-chip non-volatile

memory storage, with the exception of floating-gate EEPROM, which allows for on-chip analog weight storage and integrated access of these weights using computational elements. In using on-chip embedded NVM weight storage, systems are able to fully power-down and boot up without requiring a full re-write of synaptic weights, or incurring leakage from still-running volatile storage. This allows for the implementation of low power, low duty-cycle applications such as always-on voice wake-up systems.

Previous work on neural networks using EEPROM implements the floating gate cells as distinct units, and implements differential current steering, with two EEPROM cells representing one weight. While the current steering strategy in [54], an electrically trainable analog neural network (ETANN), is similar to the one proposed in this work, it is fundamentally different in its read methodology. Both works assume that the floating gate cells can contribute to create what is essentially an electrically tunable resistor with two inputs (the input voltage α_i and the neural network weight ω_i). ETANN treats these resistors as a large network of floating values that can be used to generate currents. Our proposed network carefully controls operating conditions (node voltages) to ensure precise resistor values and calculations. ETANN presents a non-linear resistor network whose output value changes depending on the values of inputs fed to the network, which does not accurately model the calculations of a neural network, and creates non-linear distortion in the output.

Both analog and digital approaches result in an area penalty by double storing the memory (on- and off-chip), an energy penalty from accessing memory often (neural network computations are largely weight accesses), and a time or efficiency cost when waiting for an off-chip memory access.

Our approach expands upon the use of EEPROM, using an embedded non-volatile process to store weights semi-permanently on-chip for lower power access, but ultra-dense storage. Additionally, instead of using an all-digital approach or analog charge summing, we use a current summation strategy that allows us to perform all computation within the non-volatile memory array itself. This results in an ultra-dense, power efficient algorithm that allows for a high number of on-chip weights, thus a large, efficient hardware-based neural network. Additionally, it completely eliminates all synaptic weight memory accesses, allowing for a highly parallel use of in-memory computation. Unlike the EEPROM implementation in [54],

our approach provides a precise dot-product calculation based on the exact neural network inputs and weights.

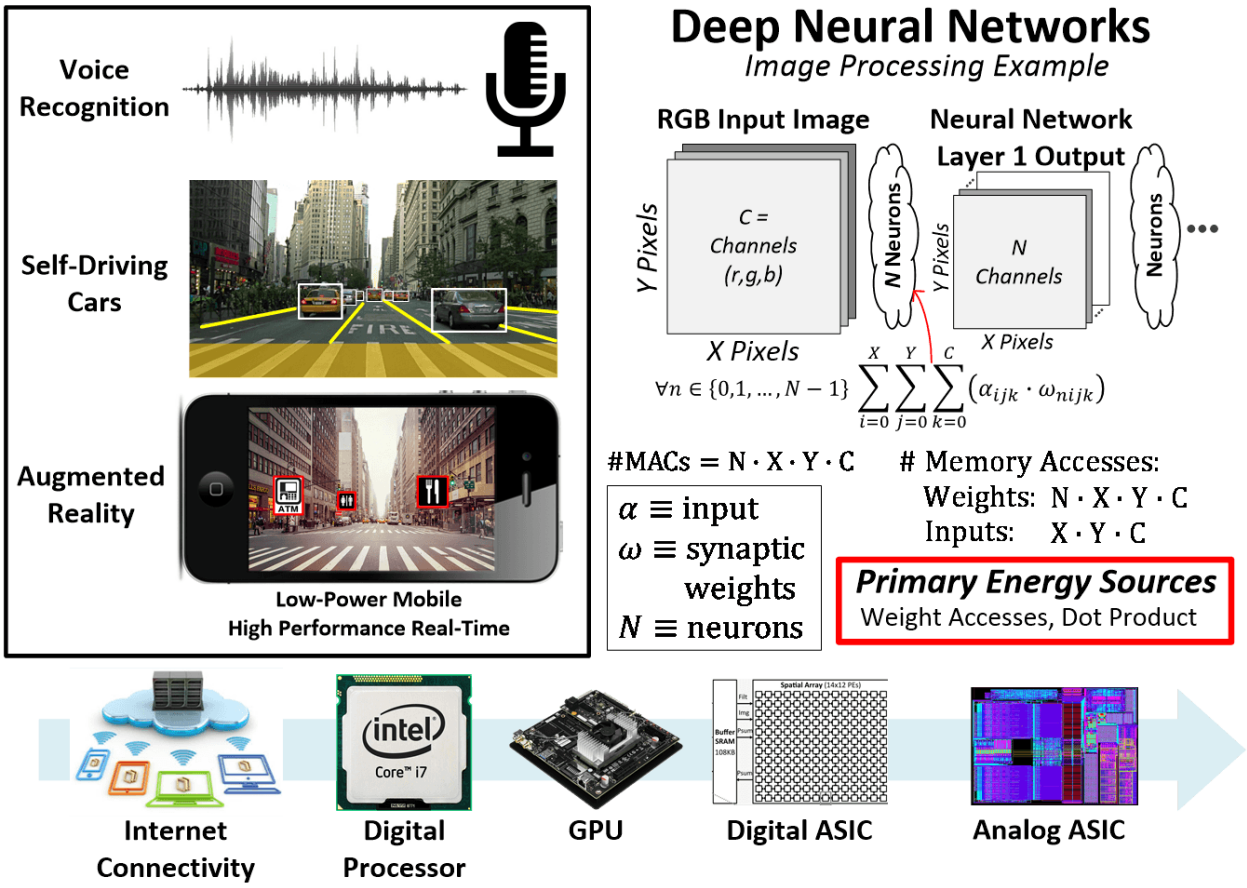


Figure 2.1: **Motivation and Applications for Deep Learning and Hardware Neural Networks.** With applications in voice and video recognition tasks, deep learning algorithms are becoming ubiquitous in current computation algorithms. The primary source of power and energy in these systems comes from weight accesses and dot-product calculations, as shown by the factor of N channels that is applied to both of these equations. The proposed system aims to reduce overall energy and power consumption through embedded non-volatile synaptic weight arrays wtih analog in-memory computation. This results in a highly parallel system that amortizes both synaptic weight reads and calculation, reducing memory accesses by $1/(N+1)\times$.

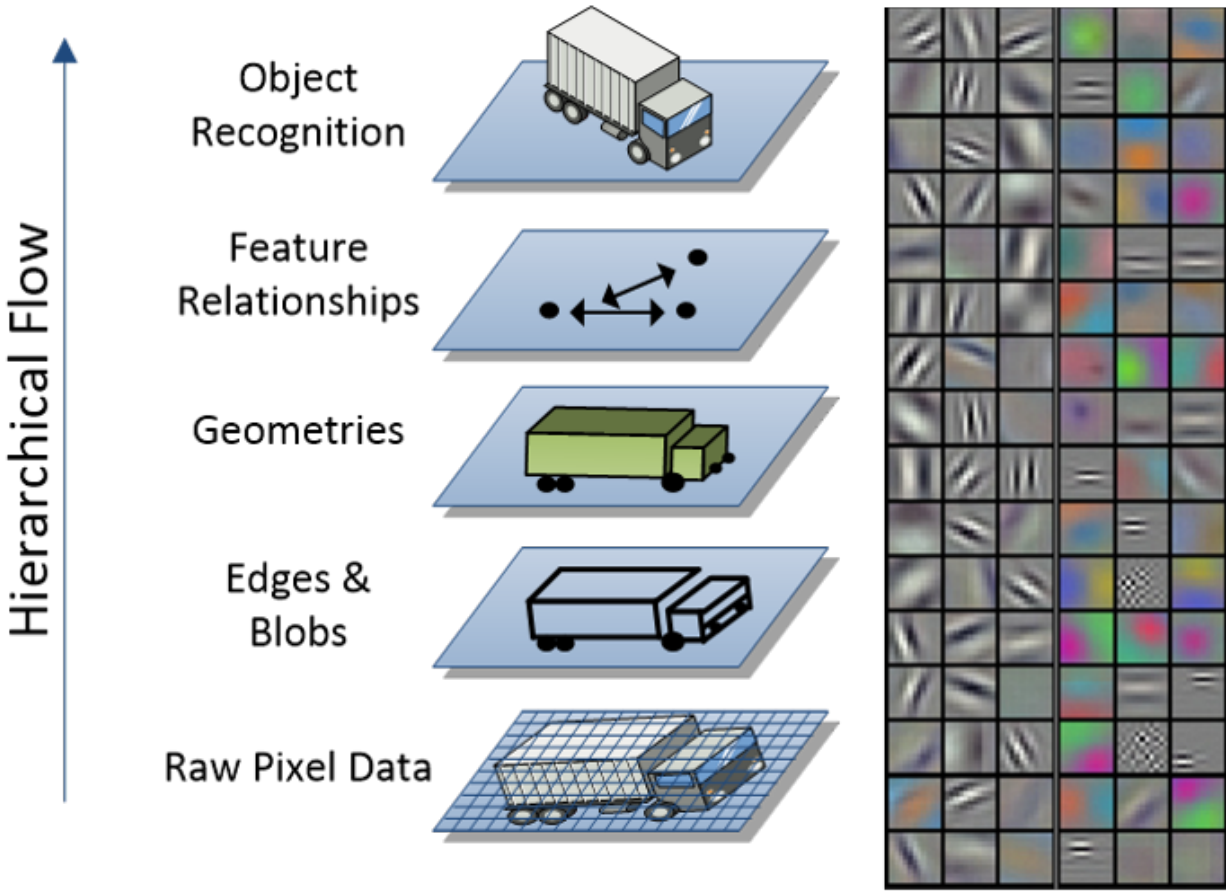


Figure 2.2: **Hierarchical Image Recognition.** Neural networks break down information from raw pixel data through the use of neuron filters in order to determine high level information from pictures (*i.e.* What edges exist in this image? What shapes are formed from these edges? And finally, what object (if any) is in this image?)

2.2 Neural Network Algorithm

Neural networks consist of a series of interconnected calculation units that take inputs from previous "neuron layers" and feed their outputs to subsequent layers. Each computational element implements a weighted sum algorithm, otherwise known as a MAC function, or a dot-product. Shown in Figure 2.3, these networks are made up of multiple layers, with an input layer, N hidden layers, and an output layer which groups the possible network decisions (*i.e.* Is the object in the image a cheetah, leopard or lion? This question would have a neural network output layer size of 3). Each circle takes in i inputs from the previous layer and has i corresponding weight values stored internally, detailing how strong of a connection the corresponding input has on that neurons output value. Each neuron computation element represents its output according to the following equation:

$$y = f(\Sigma_{i=o}^{i=N}(\alpha_i \times \omega_i) + b) \tag{2.1}$$

Different functions are applied to neurons depending on the architecture of the given neural network, to signify how strongly a neuron is firing, or whether the neuron fires at all. As shown in Figure 2.4, standard functions to apply are rectified linear, binary threshold and logistic. Additionally, some networks use an alternative to rectified linear, which is a standard linear function, allowing for negative values as well. Rectified linear, or linear functions, hold many times more information than a simple binary threshold function, as outputs can be analog values, showing the strength with which the given neuron fired. In hardware, the rectified linear function could be implemented with true analog circuitry, or with a quantized response. Based on simulated results, we found that the rectified linear output need only be quantized to 6-bit in order to preserve the accuracy of the network. The proposed network uses a 6-bit rectified linear, or linear, output response.

To demonstrate the feasibility of implementing a state-of-the-art neural network on silicon, we chose to focus on the winning network from ILSVRC-2010 [48], which achieved a 'top-5' error rate of 15.3%. This competition tested the networks ability to correctly identify an image in its top-5 guesses. It featured 1.2 million 256x256 pixel images organized into 1000 different classes. The winning network featured 650,000 neurons organized into a 6-

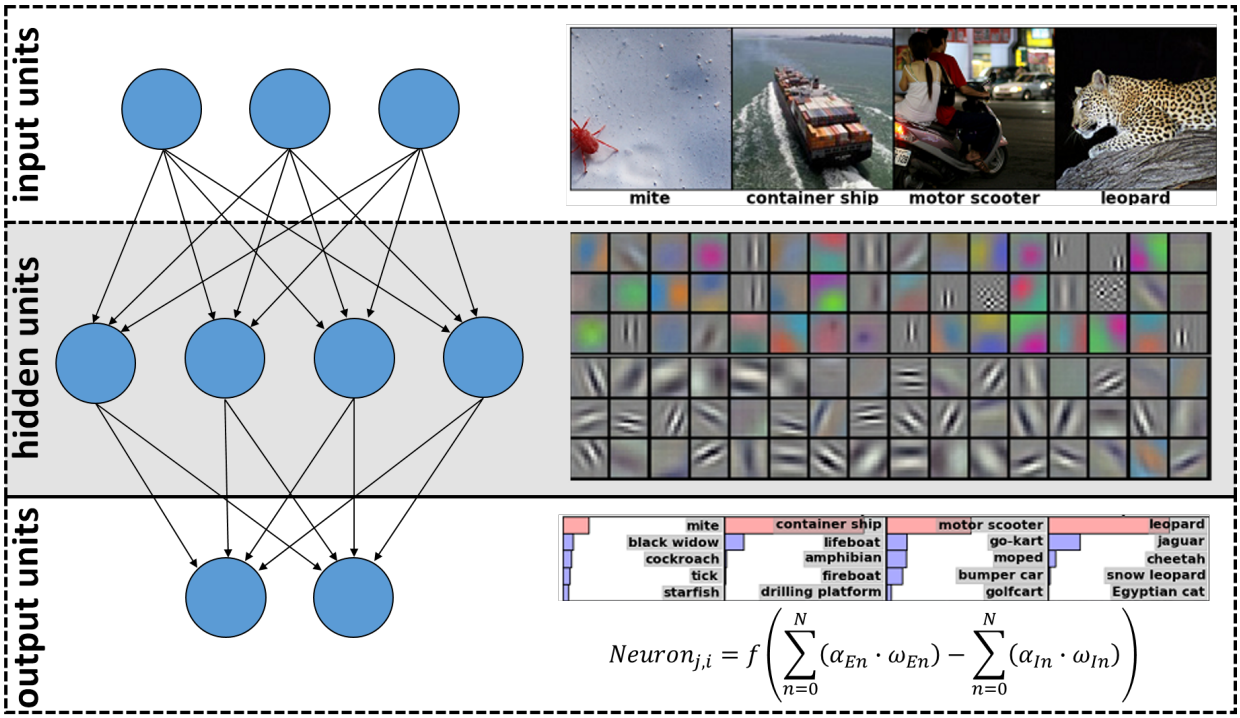


Figure 2.3: **Neural Network Flow.** Neural networks consist of a series of interconnected units that calculate and pass on information in order to determine high-level relationships between input data. These networks can consist of any number of inputs, hidden units, layers, output units and interconnections in order to achieve this analysis.

layer convolutional neural network trained with back-propagation, with a 1000-way soft-max output. The soft-max function is commonly used as the final layer of a classification network, and is a logistic function that maps a vector of arbitrary neural-layer output values to a vector of real values on the range (0,1) adding up to one, effectively giving the probability of each output.

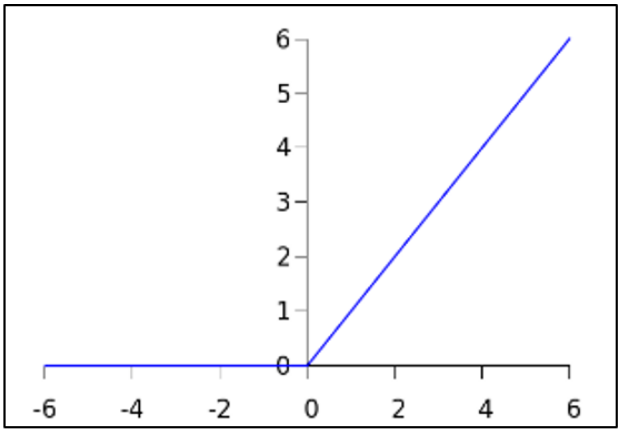
Convolutional networks are amenable to hardware implementation since they re-use the same weights across many neurons, allowing for many times more effective neurons than physically exist on-chip. In this example, the 650,000 convolutional neural network really only has between 10 and 20 thousand unique (physical) neurons.

For the purposes of this project, we scaled the network down to a manageable size for proof-of-concept hardware measurements. The chip that was taped out uses a single-layer convolutional neural network with 225 inputs and 64 neurons (sets of synaptic weights), with input and output SRAM banks. All output data are then fed off-chip onto a neural network

simulator to implement the rest of the network, and to show the effect of using less-accurate hardware (noisy, analog) for one stage.

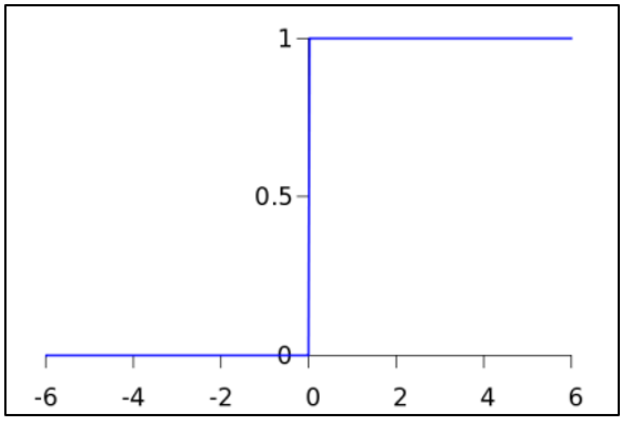
Rectified Linear:

$$y = \begin{cases} z \; if \; z > 0 \\ 0 \; otherwise \end{cases}$$



Binary Threshold:

$$y = \begin{cases} 1 \; if \; z > 0 \\ 0 \; otherwise \end{cases}$$



Logistic:

$$y = \frac{1}{(1 + e^{-z})}$$

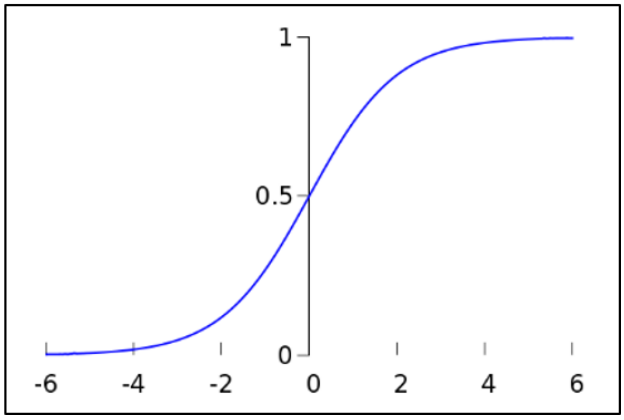


Figure 2.4: **Types of Neurons.** The calculation units that are the building blocks of neural networks are called neurons, and typically are implemented using one of the three shown functions. (Top) Rectifed linear, which uses only the positive portion of the linear curve. (Middle) Binary threshold, which outputs a '1' or '0' depending on whether the output is above a pre-defined threshold (doesn't need to be 0). (Bottom) Logistic, which is a sigmoid function dependent upon the input, with values ranging from -1 to +1. Additionally, a linear neuron can be implemented similar to the rectified linear, without zeroing out the negative portion of the graph.

2.3 Cypress SONOS Technology

This project was designed and fabricated using Cypress' 130nm SONOS technology. SONOS uses an insulating layer (silicon nitride) with traps to store charge. Unlike traditional floating gate technologies SONOS does not require a second polysilicon gate in order to store the programmed charge. Additionally, SONOS is more scalable than current flash processes, with researchers currently exploring implementations in 55nm, 40nm and 28nm [55]. With true voltage scaling, SONOS has the potential to deliver non-volatile, flash-like memory, for much lower program/erase energy and complexity [55].

Another key difference between SONOS and floating gate transistors is the threshold voltage range. Typical Flash technologies have programmable threshold voltage ranges between 3V and 6V, whereas SONOS threshold voltages are shifted negatively. The programmable range for a SONOS device is between -1V and +1V. Because of this, SONOS devices must be two-transistor (2T) structures, pairing a high voltage CMOS access transistor with the flash storage transistor in order to cut-off current flow without needing to provide -1V to the gate of the non-read SONOS cells. The cross-sectional layout and 2T circuit schematic are shown in Figure 2.5.

Similar to other floating gate topologies, SONOS requires above and below rail voltages for program and erase functionality. Unlike traditional Flash, SONOS uses Fowler-Nordheim (FN) Tunneling for both program and erase functions. Typical Flash uses Hot Carrier Injection (HCI) for programming because it is considerably faster than FN tunneling - on the order of microseconds compared to 10s of milliseconds. Because the primary function of the memory in a neural network is reading (not writing), this aspect of SONOS is amenable to our project. The weight values of the neural network will be re-written on the scale of days, so it is not necessary to have a fast write process.

Based on simulation results obtained from our software-based neural network, the required accuracy of the synaptic weights needs to be around 6 to 8-bit accuracy. Typically SONOS is used as a 1-bit storage device (positive or negative threshold voltages). Storing up to 8-bits required analysis of the device reliability and leakage of the SONOS charge trapping structure. Based on data provided by Cypress [55], charge leaks away from the SONOS

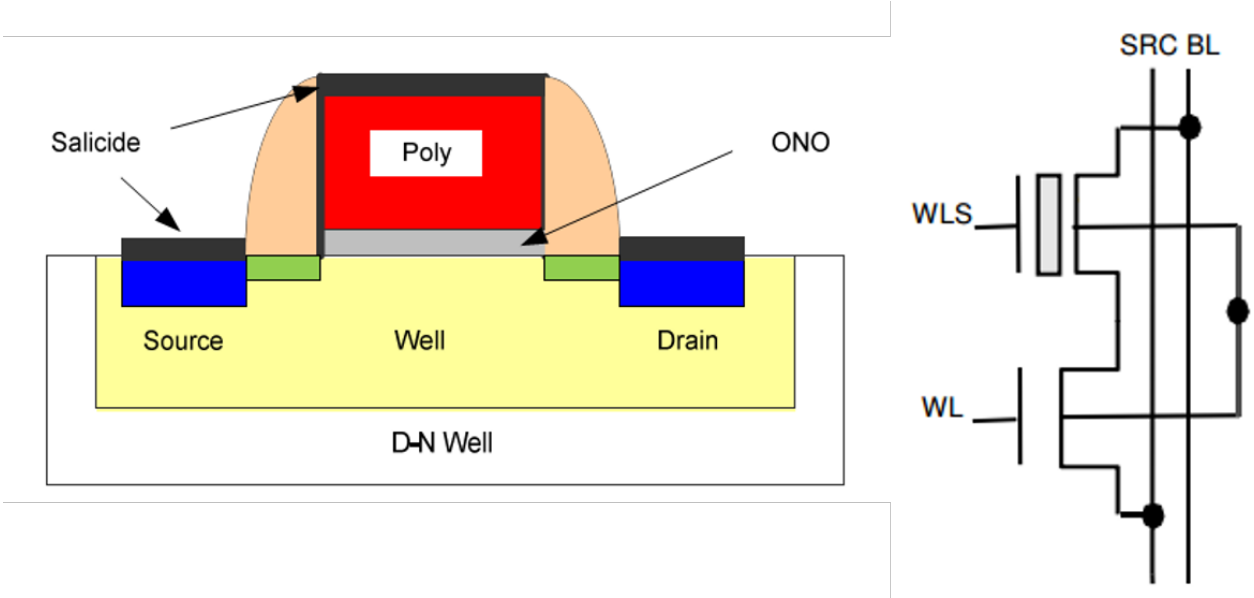


Figure 2.5: **SONOS 2T Structure.** (Left) The cross-sectional diagram for Cypress' SONOS non-volatile memory transistor. This device uses an ONO insulating layer to trap charges in order to change the threshold voltage of the device between -1V and +1V. (Right) Negative threshold voltages necessitate the use of a 2T structure to cutoff current flow of unread cells.

devices at a logarithmic rate, dependent upon the voltage itself, as shown in Figure 2.6. More positive voltages leak away orders of magnitude faster than very negative voltages. To reduce the required re-write time of our synaptic weights, we chose to use threshold voltages between -1V and 0V.

To precisely program the trained neural network weights, we implemented an on-chip, closed feedback loop programming routine. In this algorithm, we program one SONOS row at a time (all devices with connected gates). Feeding the devices a series of program $\rightarrow$ read $\rightarrow$ program operation commands allows us to analyze whether or not the threshold voltage to each cell has been precisely set to the given weight value. Using the same readout structure as used in calculation mode allows us to cancel out inherent offset and mismatch associated with the calculation flow. Constant mismatch and offset values in the amplifiers and analog-to-digital converter (ADC) will be accounted for in the programming process.

The feedback programming routine also needs to take into account the amount of threshold voltage shift that occurs due to the SONOS inhibit properties. Because SONOS cells use FN- tunneling for programming and erasing, voltage across V_{GB} will cause a shift in the

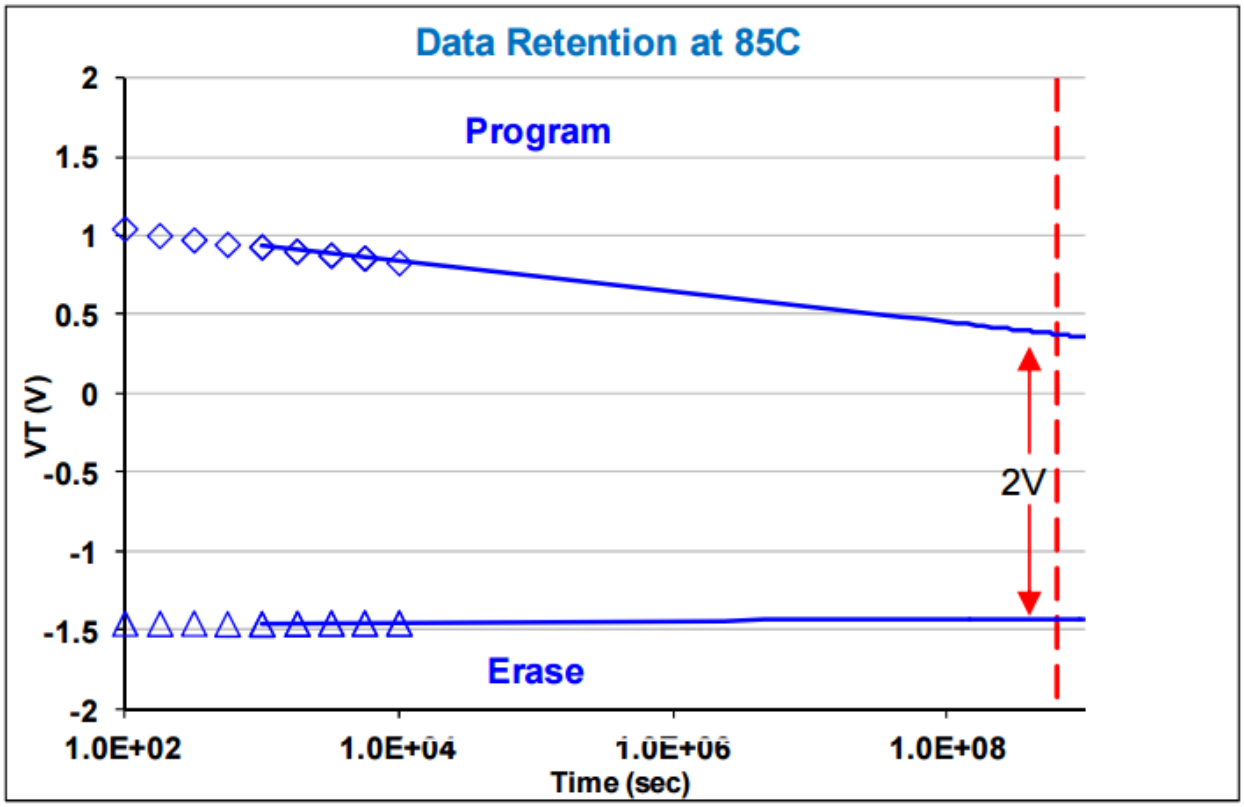


Figure 2.6: **SONOS Vth Leakage.** Trapped charges in the insulating layer leak away at a rate proportional to the one shown above. Very negative voltages leak away super-exponentially slower than very positive voltages. To achieve high precision in the programmed weights in our neural network, we chose to use voltages between -1V and 0V, to reduce the amount of time required between re-writes.

threshold voltage. When programming, 7.2V is applied to the gate of *all* SONOS in the same row, and -3.8V is applied to the body of all cells in the same array (well). If one cell in the given row needs a very low threshold voltage (-1V) and another needs a very high threshold voltage (0V), then the low voltage cell will be *inhibited* during most of the programming routine for that row. In this process, the bit-line (BL) and source-line (SL) (drain and source) voltages are raised to 1.0V to limit the amount of threshold voltage shift in cells that are not meant to be programmed. However, a small amount of shift over time still occurs (weak-write) and needs to be accounted for in the programming routine. Because our weights are calculated off-chip and offline, and the inhibit threshold voltage shift over time is known, we are able to apply a transformation to the neural network weight values before starting the programming routine that accounts for the expected shift due to inhibition -

based on the largest weight for each row.

2.4 In-Array Computation Circuitry

The key calculation required in a neural network is the dot product (MAC). After the cost of accessing thousands of stored 8-bit synaptic weights, this is the largest cost in implementing DNNs in silicon. Similar to weight accesses, as shown in 2.1, the number of MACs is dependent upon the number of neurons in the system, which can be large. All-digital approaches use traditional summation blocks with registers, multipliers, and adders. The weights are read cycle-by-cycle in a dot-product loop, traversing through the array and summing each weight $\times$ input product to the running tally [56]. This approach suffers from increased energy consumption in both MAC calculations and weight accesses, taking $X-$cycles to traverse through an $X-$input array of weights, and reading out each weight individually, requiring on-chip buffer memories to store weights before use. Analog implementations of these networks typically use charge summing, where they read a weight element from off-chip memory, and sum its value onto a capacitor [53]. Both of these strategies require many cycles, and are dominated by the energy required to read the weight value out of the non-volatile memory array off-chip.

One of the main advantages of this project is the combination of weight storage and computational element. By performing the dot-product calculation in-array where the weights are stored, we amortize the read out cost of each weight by reading out fully parallel, and using the readout current to perform current summation. This strategy completely eliminates synaptic weight memory accesses, reducing the number of total memory accesses by $1/(N+1)$ as shown in 2.1. Additionally, we avoid the costly penalty of fully parallel interconnect by confining all calculations to in-memory analog computation, performing an entire neuron-level MAC, for all inputs and synaptic weights, on a single shared wire. With thousands of 8-bit weights required for readout, traditional neuromorphic topologies use all digital implementations and read out a single 8-bit weight at a time. This is because it would be near impossible to fit thousands of 8-bit wires in the pitch of a standard memory array. Since we combine both the multiplication and summation functions within the array

(the full dot-product), there is no need for massive interconnect to the array - each neuron produces its output on a single wire which is fed to an 8-bit ADC for digitization. Thus, the proposed system implements the full dot-product calculation, scalable to thousands of multi-bit weights, on a single wire, in the push-rule density of a non-volatile memory array.

2.4.1 Dot-Product Calculation

The desired dot-product calculation for a neural network structure is given in equation (2.1).

To achieve the full dot-product calculation in our 2T SONOS structure, we force the transistors into a linear range of operation - triode mode. In order to achieve this type of operation we control each of the transistor node voltages: gate-to-source (V_{GS}), drain-to-source (V_{DS}) and threshold (V_{th}). This forces all devices into a defined region of operation, allowing us to perform current-based calculations with low-distortion. The triode region is defined by the following operating conditions: $V_{GS} > V_{th}$ and $V_{DS} < (V_{GS} - V_{th})$. These conditions produce a drain current relationship defined by the following equation, where W and L are the transistor length and width:

$$I_D = \mu_n C_{ox} \times (W/L) \times ((V_{GS} - V_{th}) \times V_{DS} - V_{DS}^2/2) \tag{2.2}$$

If we force V_{DS} to be small (compared to 1), then the term $V_{DS}^2/2$ can be ignored, giving us the simplified equation:

$$I_D = \mu_n C_{ox} \times (W/L) \times ((V_{GS} - V_{th}) \times V_{DS}) \tag{2.3}$$

This equation shows a linear relationship between $(V_{GS} - V_{th})$ and V_{DS}, where $(V_{GS} - V_{th})$ is the synaptic weight ω_i and V_{DS} is the input α_i. In practice, V_{DS} is set by amplifiers, and chosen based off simulation results to produce a linear multiplication response. Some non-linearity is expected and can be compensated for in the neural network algorithm by characterizing the neuron response.

Through SONOS cell programmability, we are able to control ω_i, but directly modulating V_{DS} would require significant drive capability of the input devices. This is because the drain

and source nodes of a transistor are low resistance, and modulating V_{DS} would require X amplifiers to source the SONOS cell current for each input pair. To avoid driving current consuming devices (drains and sources), we used the 2T structure of the SONOS device to achieve a similar result. By modulating the gate voltage (V_{GS}) of the SONOS cell, we can induce a semi-linear response in the SONOS cell V_{DS} which is proportional to the supplied input voltage α_i. Effectively we are using two transistors in the triode region as variable resistors, where one is varied using the threshold voltage (SONOS cell) and one is varied using the gate voltage (access cell). Thus, we end up with the following current equation for the SONOS cell:

$$I_D \propto \alpha_i \times \omega_i \tag{2.4}$$

Now, each 2T SONOS and access transistor pair is producing a current which is proportional to the multiplication of its corresponding input and weight. The summation portion of the dot product is achieved through the shared bit-line structure. Turning on all 2T pairs on the same bit-line produces a current flow based on Kirchoff's Current Law (KCL), that is the summation of all multiplied currents - the full dot product. This result is obtained in one cycle, and is not limited by the number of weights or inputs required for the network.

An important aspect of this calculation is that all devices must be held in a specific operating region for the duration of the calculation cycle, independent of inputs and threshold voltages, for the result to be accurate. Operational amplifiers are used to hold all nodes at a precise DC voltage, so that we can guarantee triode operation, at the same DC operating point, for all devices, and for all neurons, regardless of the neuron output. This guarantees a low-distortion output that is consistent for all input values.

Figure 2.7 shows a conceptual in-array implementation of the proposed analog voltage multiplication. This example shows a 1T flash structure, where a neuron consists of all devices connected via the same gate (highlighted in red). The neuron is enabled by providing a voltage to the gate of the neuron, turning on each cell in that column. An amplifier on the right holds the drains of each device at a common voltage (V_{common}), and the source side of each row is held at a separate analog voltage ($V_{in} < i >$) which represents the input to

that neuron weight pair. By modulating the V_{DS} of each cell in a neuron, and programming the flash cell's threshold voltage V_{th} to represent the neural network weight, we are able to define a current flow in each row proportional to the multiplication of $V_{th} \times V_{DS}$.

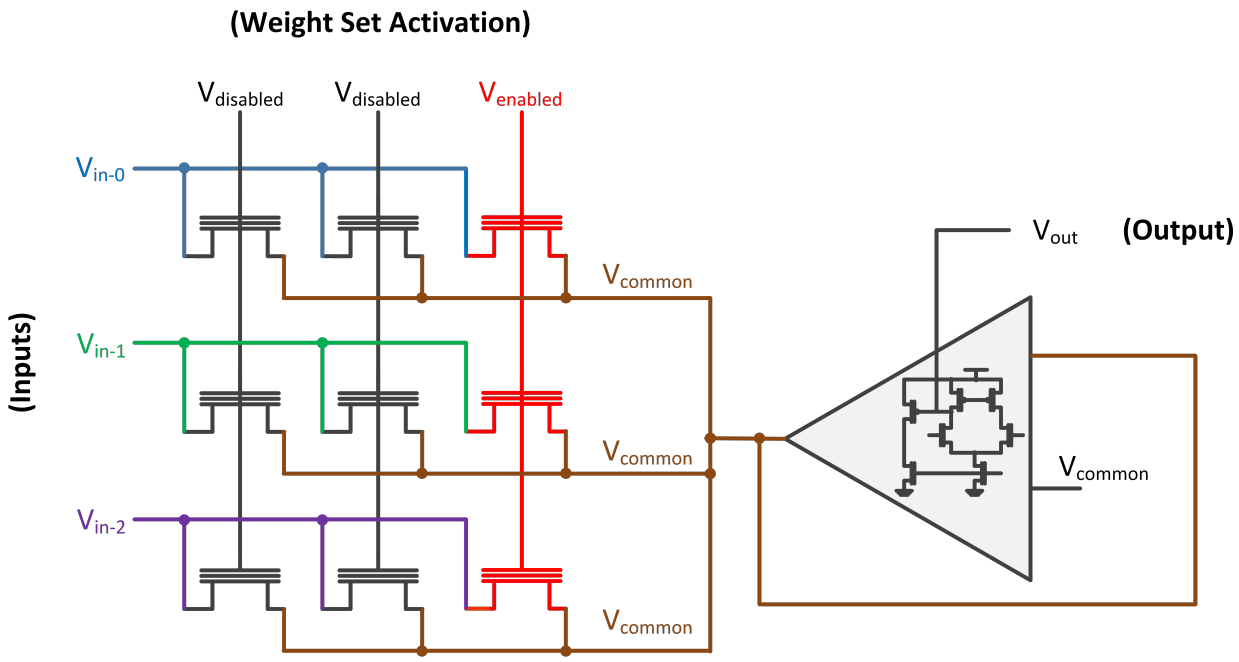


Figure 2.7: **In-Array Multiplication.** Multiplying two analog voltages together can be accomplished through manipulation of both the threshold voltage in of a device, as well as its V_{DS}, if the device is in the triode (linear) mode of operation. However, providing a voltage directly to the drain or source of a device requires a current supplying circuit, which is infeasible for a network consisting of thousands of inputs.

Figure 2.8 is a conceptual diagram that shows the change between direct $V_{DS} \times V_{th}$ multiplication and indirect multiplication via the 2T structure. In this example we use an access gate to each row of the 1T flash array, and send the input voltages to the gate of each access transistor instead of the source side of the flash cells. This example requires two amplifiers to hold the common nodes to precise values, as opposed to N amplifiers required to drive each source side of the row to a precise value equal to the input of the neural network. Since SONOS cells come with individually addressable access cells we can move the access gate into the cell structure and provide the input voltage to all access transistors connected to the same gate voltage, as shown in Figure 2.9. A significant advantage that the SONOS 2T structure provides is the ability to implement fully parallel neuron readout. Because each

SONOS cell has its own access transistor, thus its own V_{DS}, with the appropriate readout circuitry we are able to calculate all weight $\times$ input MACs for every single neuron at the same time. This type of scale-up capability in computation could allow the proposed technology to achieve massive computational efficiency.

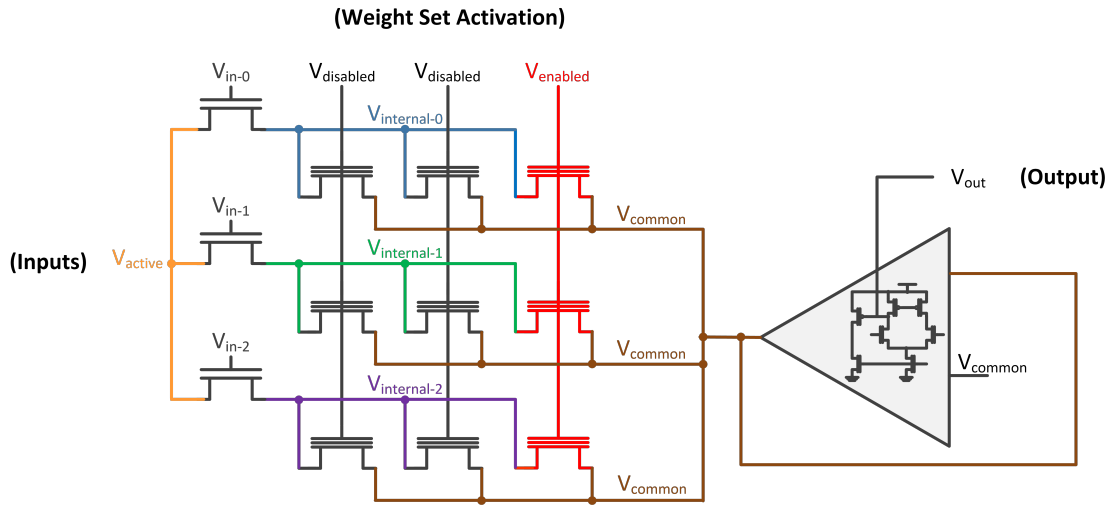


Figure 2.8: **In-Array Indirect Multiplication.** Indirect multiplication can be achieved via access gates in front of the cells. This shifts the voltage node from a current supplying node (resistive), to a gate (capacitive), meaning that the input drivers do not need to supply current. Additionally, this strategy takes advantage of the 2T structure that is already required with the use of SONOS technology.

Figure 2.9 also shows simulated graphs of an input weight 2T pair and the generated current. These two graphs were generated by sweeping the threshold voltage of a SONOS cell and the gate voltage of the corresponding access transistor. The current developed through this pair displays a semi-linear output with each variable. It is important to note that each variable is able to fully turn off the 2T pair, and is linear in a certain range of operation (which we will choose to achieve maximum accuracy from our neural network).

2.4.2 Excitatory and Inhibitory Arrays

Neural network algorithms use both positive and negative *weight* $\times$ *input* products to generate each neuron output. This gives the network a greater ability to define which features and filter types appear (or do not appear) in the given input image. For example, it is helpful

for the algorithm to distinguish whether or not an input image has a clear horizontal line (very positive), definitely does not have one (very negative), or is neutral about whether it exists (zero).

To accomplish this, weights are divided into two sets of arrays "excitatory" and "inhibitory". The excitatory set of weights *adds* to the neuron result, and the inhibitory weights *subtract*, giving us the following neuron equation:

$$y = \Sigma_{i=o}^{i=N}(\alpha_{Excite_i} \times \omega_{Excite_i}) - \Sigma_{i=o}^{i=N}(\alpha_{Inhibit_i} \times \omega_{Inhibit_i}) + b \tag{2.5}$$

To achieve this in our current summing topology we implement two identical SONOS arrays, with currents flowing in opposite directions. As shown in Figure 2.10, the arrays share a common bit-line, which is regulated by an operational amplifier in feedback. The inhibitory weights send current into the shared bit-line and the excitatory weights take current out. The regulating amplifier adjusts for the difference in current between the two arrays and produces an output voltage proportional to that current, which is dropped across a feedback resistor. If the current provided by both arrays were equal, then the regulating amplifier wouldn't need to do any work, and the voltage dropped across the resistor would be 0V. The calculation chain is shown in 2.11, and consists of the regulating amplifier with feedback resistor, a sample-and-hold amplifier (SHA) and a 7-bit neural transforming ADC. The digitized output from the ADC is sent to memory buffers for storage and to be used as inputs to the subsequent neural network layers. The ADC transforms the analog output of the cells into either a hard sigmoid or rectified linear transformation.

To readout the value all 64 neurons in the array we have pass gate transistors that are used to mux each of the amplifiers between the neurons. Reading out the entire array would take 64 cycles. A more parallel system could tradeoff power for speed by implementing 64 readout structures (calculation chains) to simultaneously readout every neuron in the array, as opposed to time multiplexing the neurons to lower the average power dissipation.

During offline neural network training, the weights are tuned so that the neurons produce a Gaussian distribution of outputs centered around 0. This lends itself well to our current summing topology because that creates the least amount of work for the amplifiers, allowing

the inhibitory and excitatory arrays to self-regulate. It also produces an energy optimal implementation of our circuitry. Previous work has focused on improving energy through reducing activity factors in neural networks by zeroing out inputs and weights to reduce the total number of MACs and memory accesses required [57, 58]. While this strategy results in improved energy efficiency for some applications, it is not a universal solution to hardware neural networks and will not produce the same results over different architectures and data sets. The proposed work achieves 3pJ/MAC regardless of architecture, size or data set, and is roughly equal to the 2.27pJ/MAC achieved using 30%-90% input and weight sparsity.

2.4.3 Full Calculation Flow

During operation, a 64x64x(r,g,b) image is fed into the input SRAM bank on chip. An FSM extracts the first sub-section from the image and feeds 225 values through the analog input voltage drivers to select which quantized values each neuron input is set to. After a set amount of time for the settling of these gate voltages, the system steps through each neuron to perform all 64 calculations, saving each result in a memory buffer after completion.

At each neuron step, the appropriate neuron-select muxes are set to connect all 3 amplifiers to drive all neuron nodes to precise voltages. This turns ON each 2T SONOS pair in the neuron by providing a non-zero V_{DS} across each device. All other neurons are OFF at this time because their drain and source voltages float toward each other and create an effective V_{DS} of 0V. Once the amplifier voltages are settled, the proportional multiplication current through each cell has developed, and the summed value of these currents is compensated at the middle (calculation) amplifier. This current is provided by the calculation amplifier to either source or sink excess current from the inhibitory and excitatory arrays, and the excess current passes through the feedback resistor on the output of the amplifier. The feedback resistor develops a proportional voltage across it, and both nodes of this resistor are sent to a pseudo-differential ADC to digitize the measured value. This value is sent to a memory buffer for later use, and the on-chip finite state machine (FSM) is incremented so that this process is repeated on the next neuron by shifting a '1' through a shift-chain to enable the next mux. This process is repeated for all 64 neurons, after which the windowing opera-

tion is shifted by one pixel in the image and a new input set is sent through drivers to the calculation array.

Neural network algorithms require a massive amount of computation in order to analyze a single image. The process described above, which implements only the 1^{st} stage of a neural network requires, 19.66 million MAC operations for a single 5x5x(r,g,b) image. Additionally, in previous digital neural network implementations [56], this process would take almost 40 million MACs. The proposed circuitry is able to perform fewer operations because SONOS cells are able to store 8-bit weights on chip, allowing for an 8-bit multiplication. Other implementations require a separate calculation per weight bit, increasing the number of operations by 8×. The proposed system takes 262,144 cycles to analyze the full image (64x64 for 64 neurons). Previous systems [59] require one cycle per MAC (weight × input pair), amplifying the number cycles to 19.66 million, even for 1-bit weight precision. Additionally, with a fully parallel version of this circuit - which requires adding 63 more calculation amplifiers - we would be able to calculate the full image in 4,096 cycles, one cycle per windowing operation.

This architecture can be expanded to implement a wide range of deep neural network types and input/weight configurations. The fabricated system has 228 inputs and 64 neurons in each excitatory and inhibitory array, using 3-bit quantized analog voltages. Binary inputs are also feasible, but do not present sizable energy savings as the input amplifiers are already power gated during the calculation sequence.

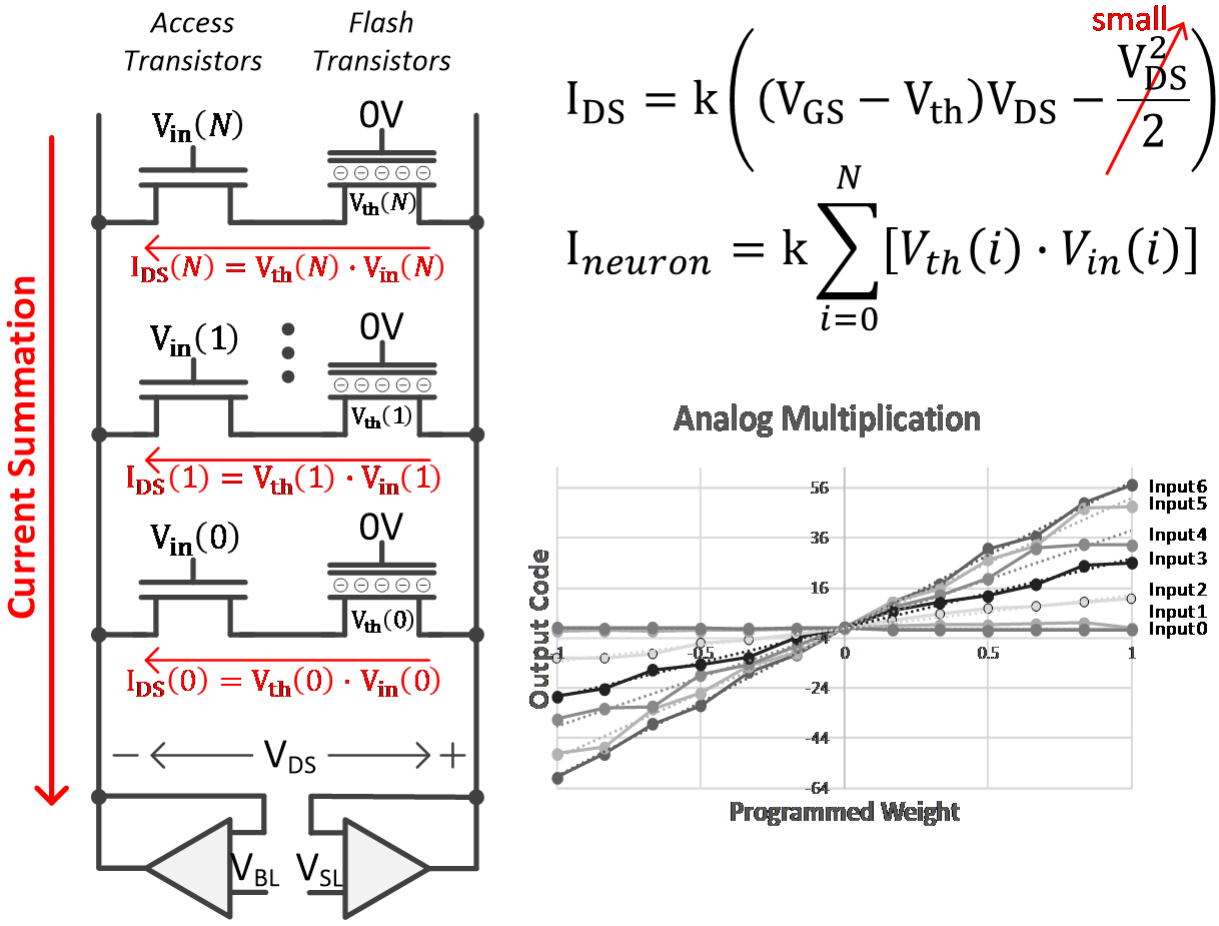


Figure 2.9: **In-Array Parallel Dot Product Calculation.** (Left) 2T SONOS implementation of indirect voltage multiplication in the current domain. The input voltages (input data) are supplied to the $Access < i >$ node, the weights are stored on the SONOS devices, and all other nodes are held constant. The SONOS word line is set to 0V - which is the highest voltage needed in our threshold range to keep the transistor ON. Many sets of these 2T devices are connected between the same $BL < i >$ and $SL < i >$ pairs. In the dot-product calculation, the 2T structures multiply each input and weight pair, while all 2T structures between the same BL and SL pairs implement fully parallel current summation. (Right) Measured 2D sweeps of both the input voltage and threshold voltage showing semi-linear behavior in both - meaning that linear multiplication between the two values is achieved.

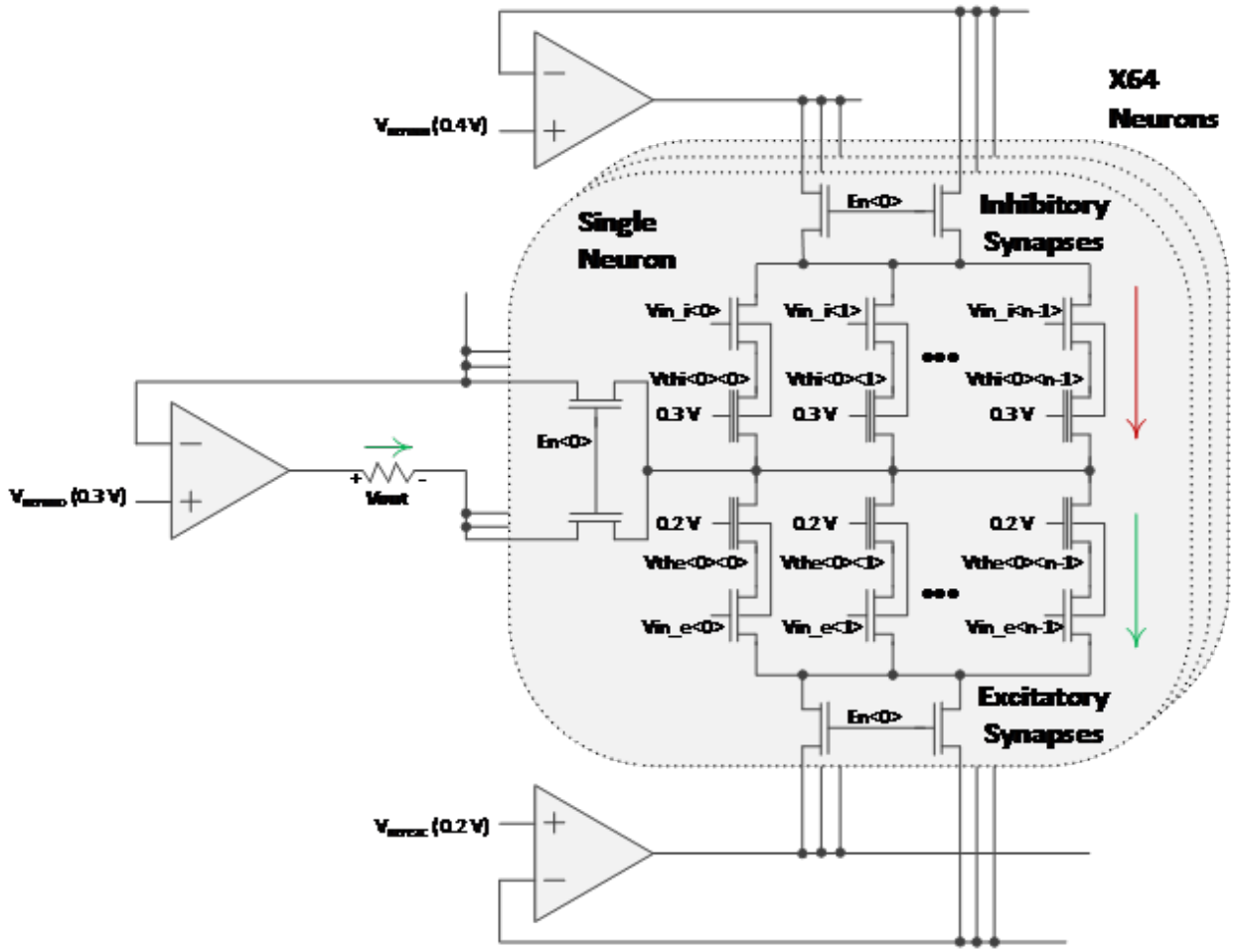


Figure 2.10: **Fully Parallel, In-Array, Subtractive Dot Product Calculation.** Both the excitatory (positive) and inhibitory (negative) arrays require amplifiers to hold the drain/source nodes at a precise voltage - to guarantee linear operation. Additionally, these amplifiers need to be muxed between all 64 neurons, and the middle amplifier (calculation amplifier) needs to be muxed. The voltage dropped across the resistor on the output of the middle amplifier represents the amount of current differential between the two arrays, and is proportional to the subtractive dot-product calculation of all inputs $Vin_{e/i} < i >$ and weights $Vth_{e/i} < i >$.

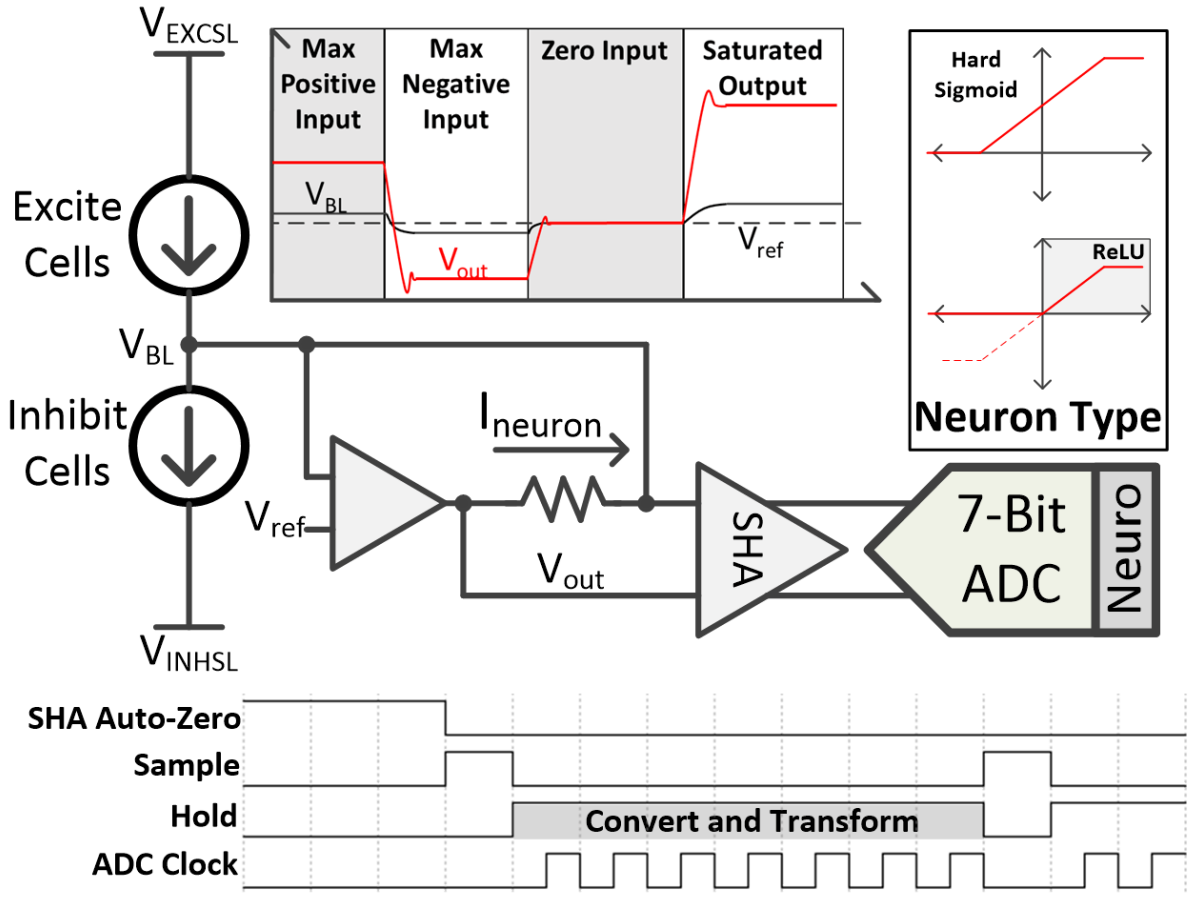


Figure 2.11: **Analog Calculation Chain.** Analog neural network cell currents are summed together on a common bit-line. This bit-line voltage is held at a precise value through the use of an operational amplifier in feedback with a resistor. The amplifier sources or sinks the required amount of current to hold the bit-line at the provided reference voltage V_{ref}. This current is passed through the feedback resistor and transformed from current to voltage. The voltage is sent to a SHA and then into a 7-bit ADC for digitization.

2.5 System Architecture

As shown in Figure 2.12, the proposed system requires synaptic arrays to store the neural network weights drivers and multiplixers to switch between the quantized input voltages sent to the access transistor gate lines, multiplexers to select between neurons, 3 amplifiers, an ADC, amplifiers driving the quantized input voltages, and high voltage (HV) charge pumps and switches for programming and erasing. All of these blocks require careful analysis and layout to determine which devices can share the same substrate, and which devices require a triple well layout to change the body, or to isolate it from noise generated by the digital supplies.

Figure 2.13 shows the full layout of the implemented system, including the input and output memories, custom logic for calculation and pads. The full size of the taped out die is $1.85mm \times 1.85mm$. Most of the die area is consumed by enough I/O memory in order to test this chip at speed. Because we are testing a proof-of-concept calculation structure, this chip does not include the I/O capabilities (including a serializer/deserializer (SerDes)) that are required to push all of the image information on-chip at speed. Figure 2.14 shows the full layout of the taped out die. This diagram shows each memory location as well as all of the blocks inside of the analog computation block. While the SONOS arrays take up a noticeably small amount of space in this layout, larger, more complete systems will have a much bigger percentage taken up by weight storage (SONOS arrays), giving a significant increase in "array efficiency".

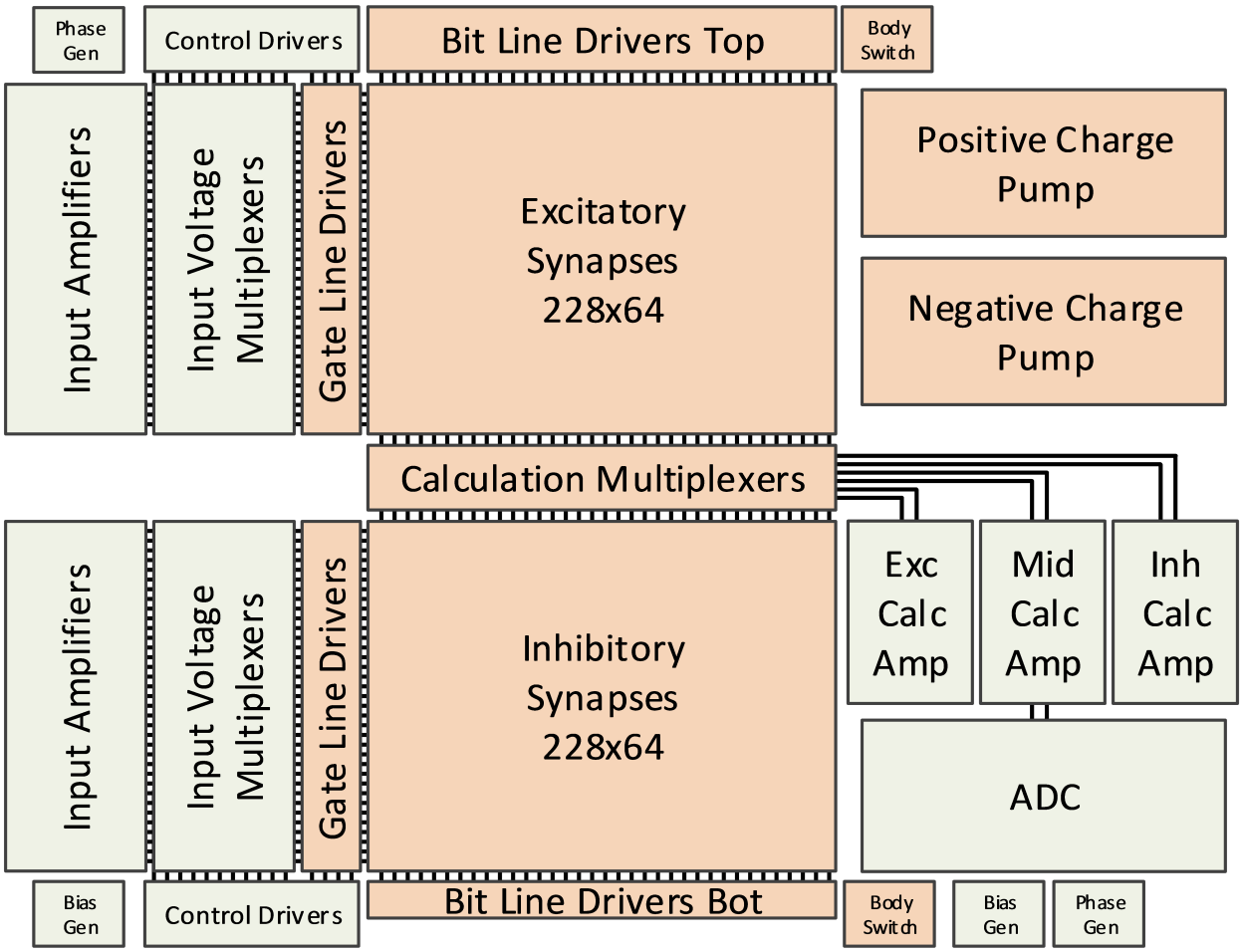


Figure 2.12: **Architectural Floorplan of CNN Calculation Flow.** Architectural floorplan showing all blocks required to perform the convolution neural network (CNN) calculation flow. This includes all input drivers and multiplexers for the input voltages, the excitatory and inhibitory arrays with precision amplifiers, the ADC used to read the voltage across the resistor, and the charge pumps needed to program the threshold voltages.

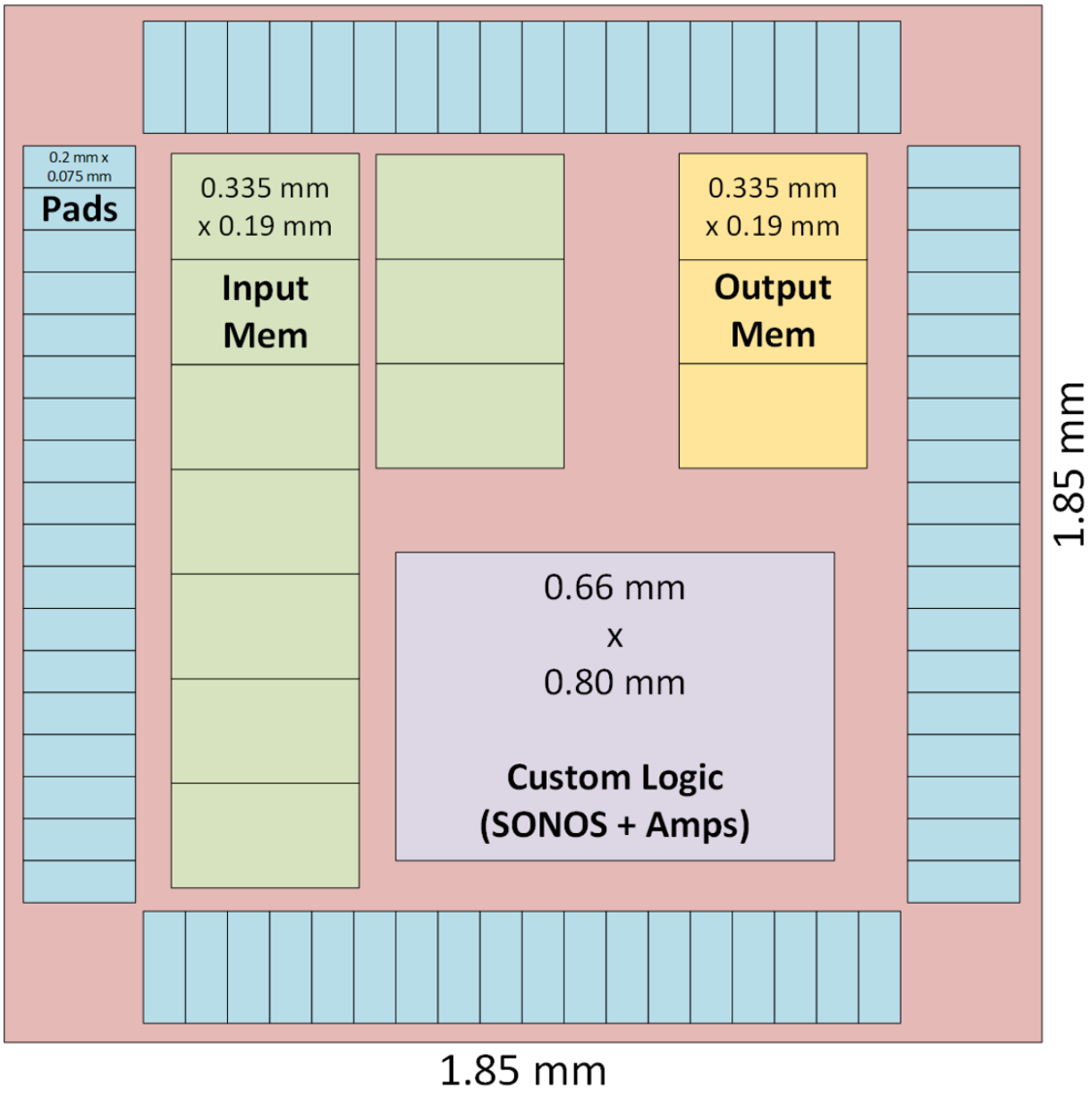


Figure 2.13: **Neural Network Test Chip Floorplan.** Total floorplan layout of the neural network test chip. Input and output memories are used for testing image recognition at full speed without requiring fast I/O interfaces to stream image data on and off chip.

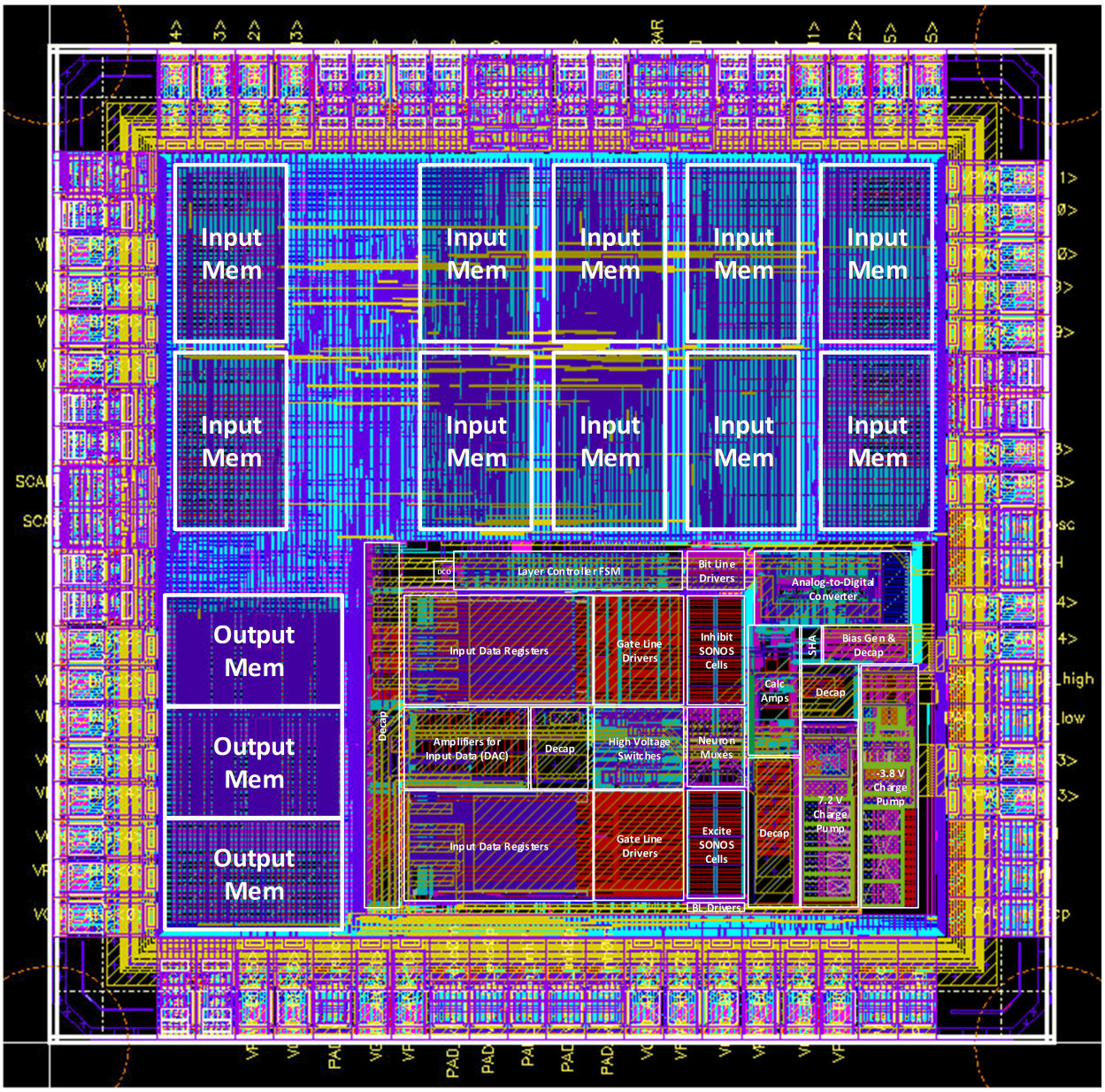


Figure 2.14: **Die Shot of Neural Network Test Chip.** Layout of the neuromorphic test chip with labeled components including input and output memories, pads, and all calculation blocks.

2.6 Measured Results

Measured results are shown in Figure 2.15. Threshold voltages were programmed in an offline routine to generate known current outputs. Input voltages are swept through all eight options from the 3-bit input drivers. The results show both analog summation and analog programming results. A filter is programmed into the excitatory weights of a given neuron (*i.e.* A diagonal line, or a square wave) and an opposing filter is programmed into the inhibitory weights. When summed together using a full dot-product calculation the output of the neuron should be equal to zero, shown in the graph. Measured accuracy of analog calculations are 5.02-bits, with correct summation responses shown in the top and bottom-right graphs.

Analog programming is shown in the bottom left graph in 2.15, where a SONOS cell is programmed in a program → read → program routine until a desired output code is reached. This measurement shows programming both a linear sweep of -64 to 64 (full ADC range), and all zeroes. Measured programming accuracy is 6.4-bits with calculation-noise averaged out. Because programming is performed offline and off-chip, multiple measurements can be taken for each point on this graph, averaging out the inaccuracies of the read routine and guaranteeing a more accurate program routine.

Energy efficiency is shown in Figure 2.16. The fabricated system uses 2.97 pJ/MAC, while achieving 3.19 GMAC/s. The pie chart shows a breakdown of the total power consumption of 9.26 mW, split as 5.7 mW of digital power (top level FSM, SRAMs, ADC) and 3.56 mW of analog power (amplifiers, SONOS cell current). Shown in the comparison table, previous work [57–62] ranges in energy efficiency between 2.27 pJ/MAC up to 257.31 pJ/MAC. Work presented in [57,58] achieve similar, or only 3× increased energy consumption, but is limited to convolution-only networks, and requires high data and weight sparsity to achieve these efficiency numbers. Most data sets cannot use 30-90% data sparsity proposed in [57]. Additionally, the architecture proposed in [58] requires 4× batch processing (parallel input feeds), achieving gains in efficiency through reusing off-chip weights between the different inputs. Not all data sets or applications can use batch processing. Single-user voice recognition, for example, has only one input feed, which would not benefit from the parallelized architecture.

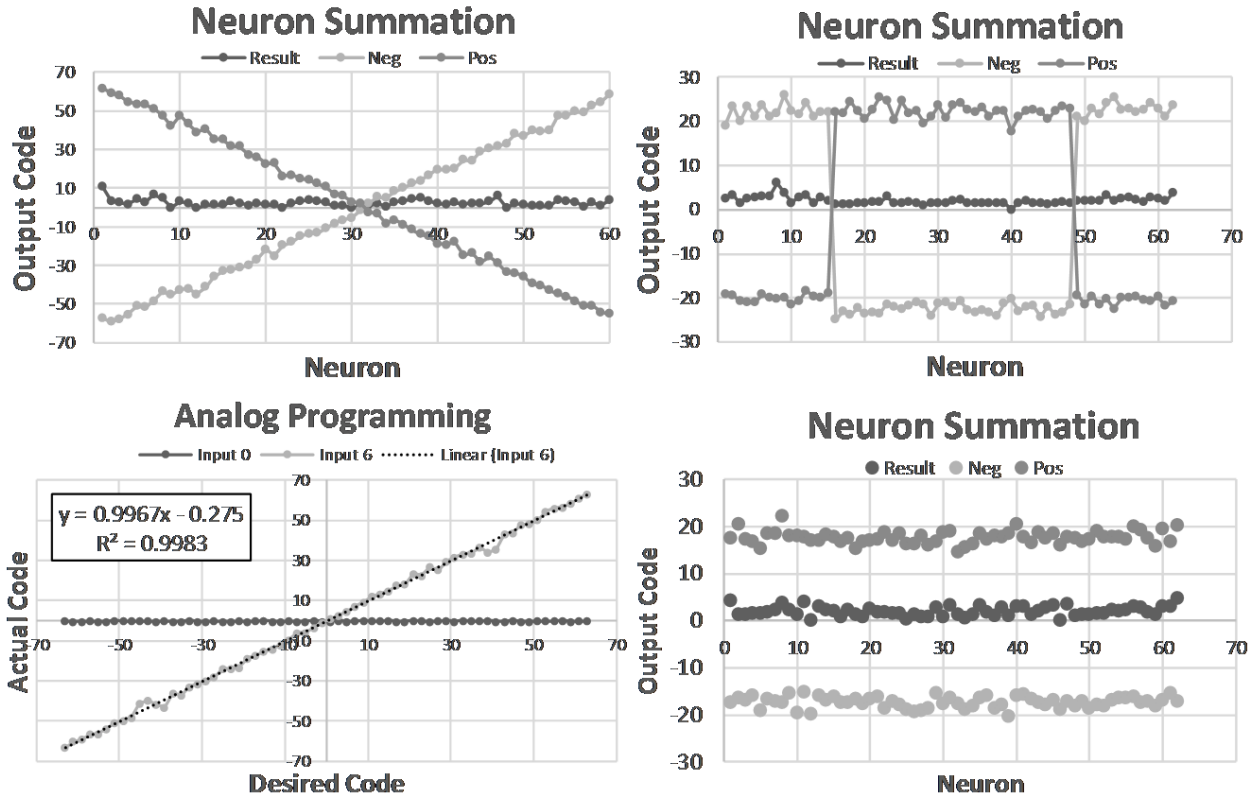
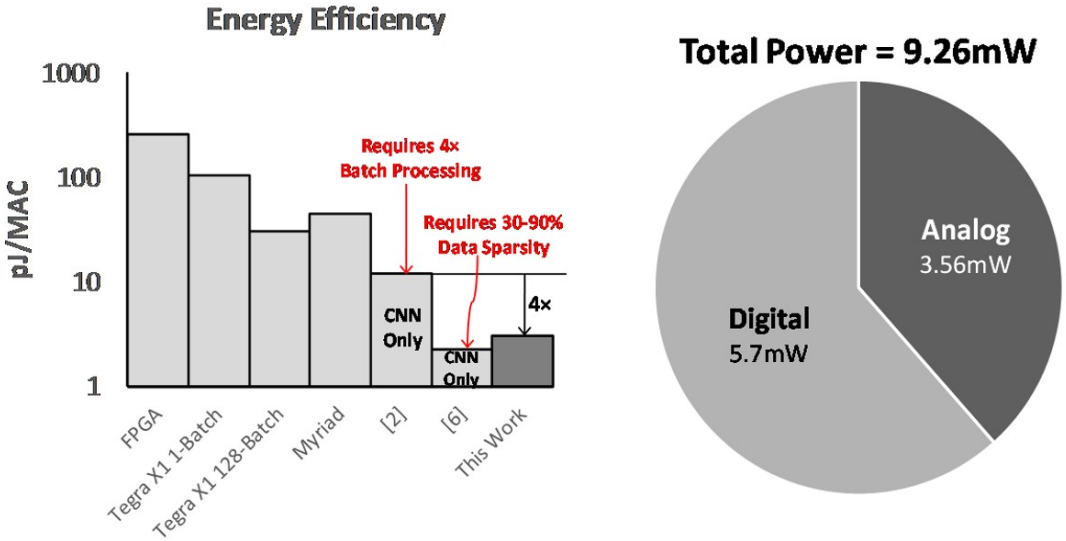


Figure 2.15: **Measured In-Memory Analog Calculation Results.** Weights are programmed to analog voltages, and inputs are swept to produce different filters (diagonal, square wave, static). These filters are used to demonstrate the efficacy of in-memory analog math. When summed, each filter is meant to produce a zero output code, showing noise and offset inherent to the calculation strategy. (Bottom-Left) Analog weights are programmed in an offline routine using averaging to factor out the noise of the calculation components.

Compared to commercial systems that also support all DNN topologies, as well as 1-way batch processing, we achieve between 15× and 86× energy reduction.

Figure 2.17 shows a die shot and layout of the fabricated system. Made in a 130nm SONOS process, this system uses 2mm×2mm of die area, and implements a 228×64 neural network layer. The analog accelerator is highlighted and layout is shown. This block includes input latches and amplifiers, SONOS cells with program/erase peripherals, amplifiers, an ADC and charge pumps. The system was tested at speed, with all inputs and neurons programmed. This system is a proof-of-concept and can be scaled larger for more state-of-the-art DNN architectures.



	Stratix V D5 FPGA	Tegra X1 (Jetson)		Movidius Myriad 2	Moons VLSI '16	Chen ISSCC '16	This Work
DNN Topologies Supported	All	All	All	All	Convolutional	Convolutional	**All**
Off-Chip Memory?	Yes	Yes	Yes	Yes	Yes	Yes	**No**
Technology (nm)	28	20	20	28	65	40	130
VDD (V)					0.82-1.17	0.85-0.92	1.8
Parallel Workloads Required	1	1	128	1	4	1	1
Sparisty of Weights/Activations for Measured Energy (%)	0	0	0	0	8-20	29-89	0
Power (W)	25	5.1	5.7	1	0.278	0.071	0.0093
Performance (GMAC/s)	97	49	187	22	23	13	3.19
Energy Efficiency (pJ/MAC)	257.31	104.98	30.47	45.05	12.02	2.27	2.97
Input Precision (bits)	32	16	16	16	16	4-9	3
Weight Precision (bits)	32	16	16	16	16	7-9	7

Figure 2.16: **Energy Consumption and Comparsion to Prior Work.** The proposed in-memory computation unit achieves between 15× and 86× energy reduction over similar previous work, and is similar to prior work that relies on batch processing and data/weight sparsity for convolution-only neural networks. Total power consumption is 9.26 mW, energy efficiency is 2.97 pJ/MAC with performance at 3.19 GMAC/s.

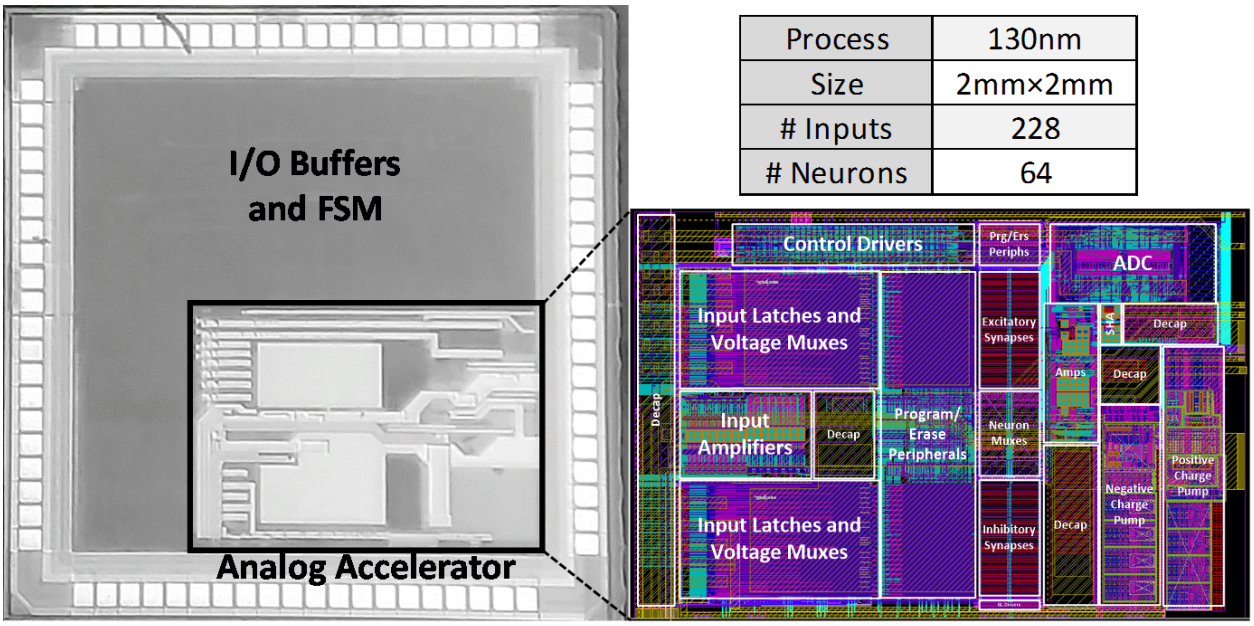


Figure 2.17: **Die-Shot, Layout and Performance Summary.** The system was fabricated in 130nm SONOS and consumes 2mm×2mm of area.

CHAPTER 3

Subthreshold Neuromorphic Computation

3.1 Motivation

The human brain is an ultra-dense, complex processing unit that scientists have been attempting to replicate in computing for decades [29]. The earliest representation of a single neuron is shown in Figure 3.1, known as the Perceptron, and was developed by Frank Rosenblatt in 1957 [50]. This example shows a single neuron with limited ability to represent large data sets, but current work focuses on tiling these models together in large, multi-layer networks that more closely represent the depth and complexity of the human brain's interconnectivity. The brain consists of approximately 100 billion neurons, with at least 100 trillion connections [63]. The immense scale of computation circuitry and interconnection restricts this style of computing from being implemented in traditional Turing-machine computers. As presented in Section 2.4, dense, non-volatile memories, with high precision weight storage and in-array analog computation can more closely approximate the massively interconnected networks seen in the human brain.

Previous approaches to brain-inspired hardware have focused on fast, highly accurate, digital implementations that exactly replicate software results [59]. While processor-style neuromorphic computers are useful for some applications, low-power mobile processors have stricter limitations on power density and consumption, and relaxed constraints on processing speed. Real-time recognition by neuromorphic hardware can benefit these mobile platforms by providing slower, but drastically lower power, neural network computation. Such systems

could be used in wake-up style applications, where very low duty-cycle inputs are fed to an always-ON ultra low power neural network. The output of this network would be used to turn on or off a high speed, high power neural network, shown in Figure 3.2, for in-depth processing and analysis.

The response time of a signaling neuron in the human brain is on the order of 0.01kHz to 0.1kHz, which is significantly slower than the multiple GHz clock speeds of state of the art processors today [64]. Using an ultra-low power, slow processor with a network of dense interconnected computation elements could most closely mimic the computing power of the human brain - allowing scientists to approach the performance and efficiency of one of nature's greatest computers. Section 2.4 presents a low power, dense neural network implementation using triode mode operation MOSFETs to perform highly parallel, in-memory, dot-products. Transforming this concept into the sub-threshold MOSFET domain allows for orders of magnitude reduction in power consumption, while keeping computation speed above the real-time response of the human brain.

This work presents a sub-threshold neuromorphic computer with in-array computation and logarithmic transformation functions to mimic the human brain and implement neural network algorithms on silicon.

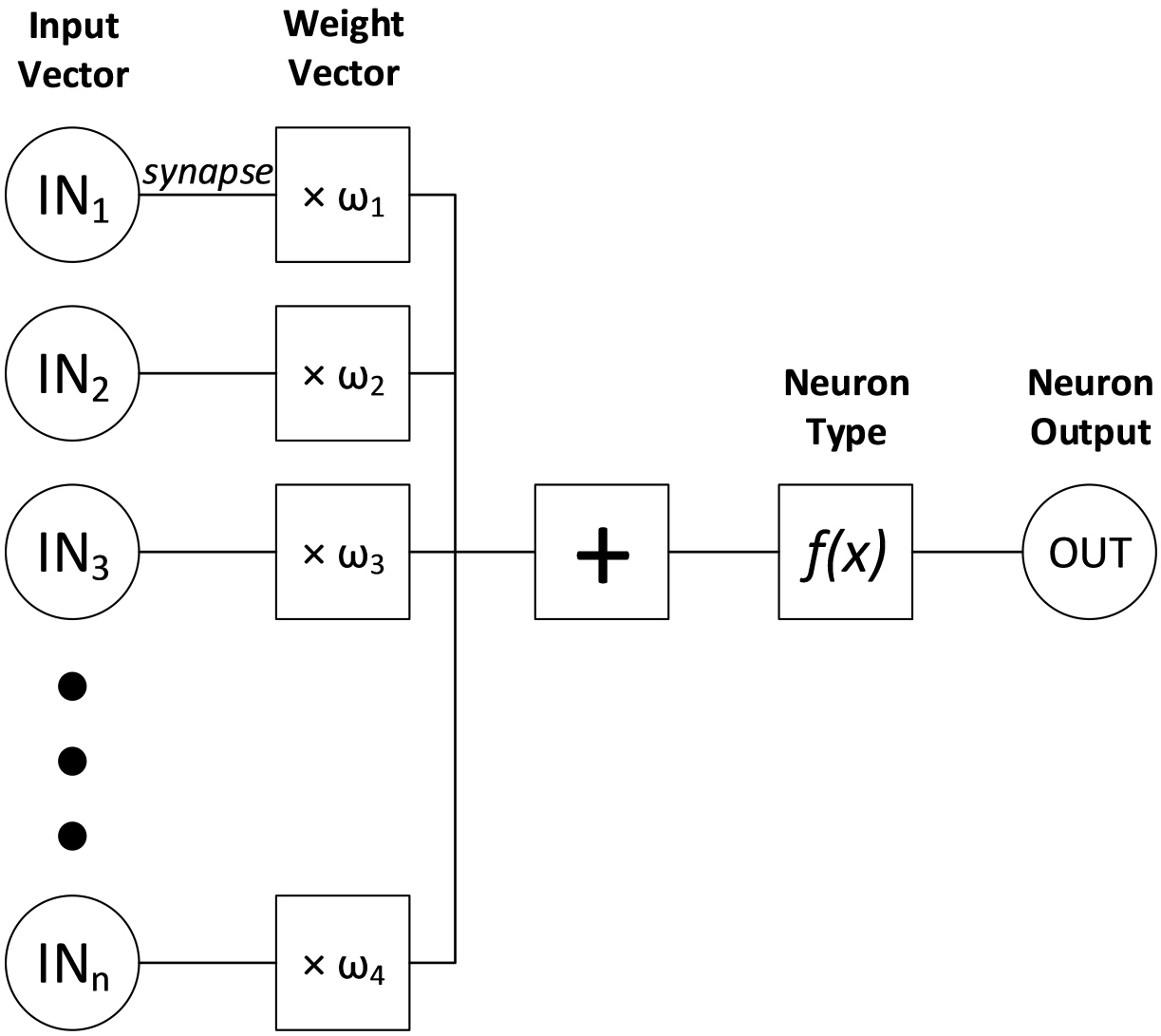


Figure 3.1: **Block Diagram of Perceptron Neuron Model.** The original neuron model, known as the perceptron, is represented as a weighted sum sent through a neuron function. This calculates a series of multiplications $IN_i \times \omega_i$, sums all of the products, and sends the sum through the function $f(x)$ to produce the neuron output.

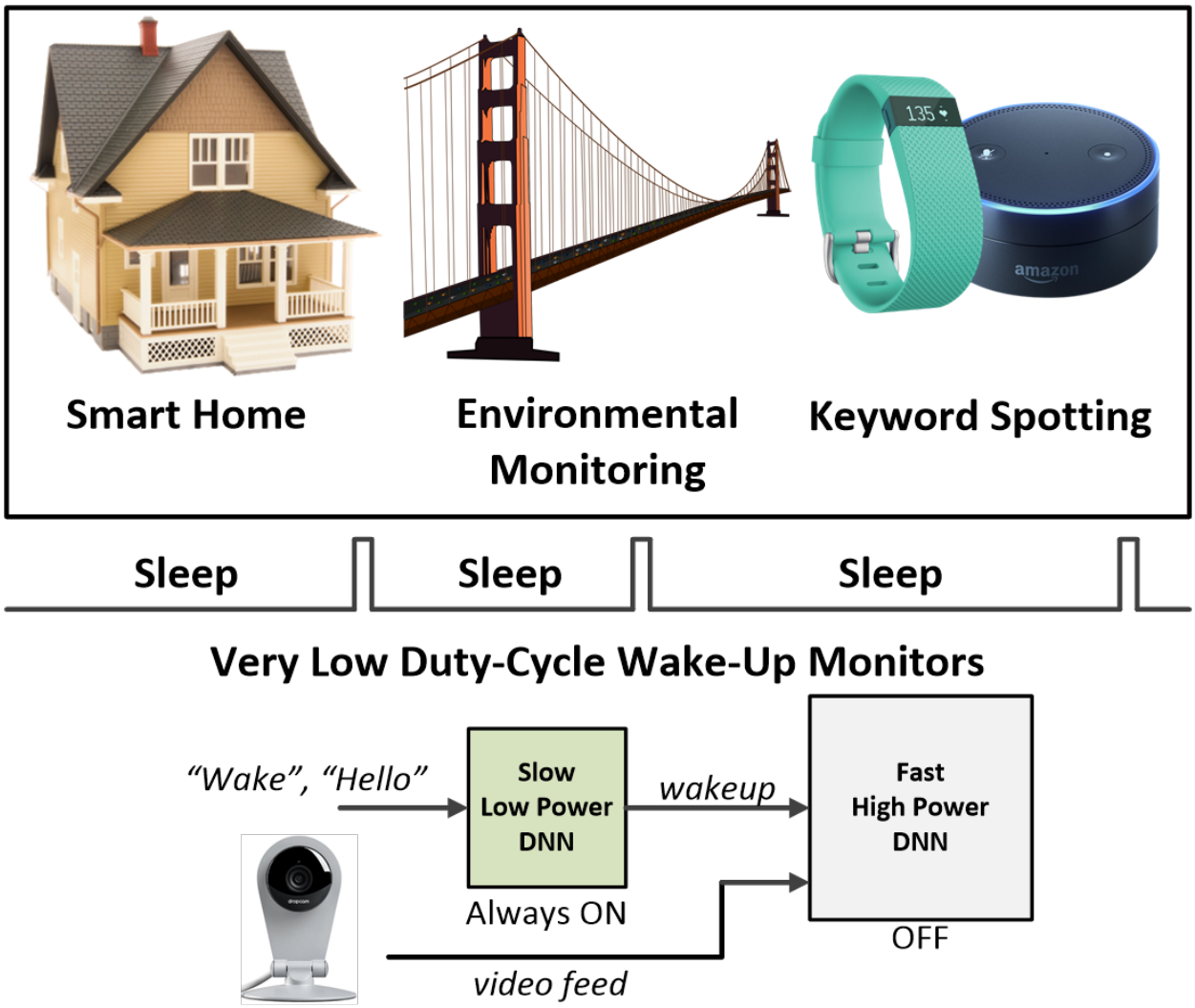


Figure 3.2: **Subthreshold Neural Network Wake-Up Scheme Motivation.** Subthreshold neural networks can be used to implement low-speed, low-duty cycle, ultra-low power recognition problems. Used in a wake-up topology, the always-ON subthreshold DNN could listen for key events in order to activate a higher speed, higher power neural network for more accurate recognition algorithms.

3.2 In-Array Computation Circuitry

By modulating the gate and threshold voltages of the SONOS transistors, we can achieve an in-array dot-product calculation for all weights simultaneously, eliminating all off-chip weight memory accesses. This real-time measurement in the sub-threshold region of MOSFET operation allows for very low power, dense, highly parallel neural network implementations on silicon.

Because sub-threshold MOSFETs have exponential current equations, the proposed system focuses on how to cancel the exponential properties and generate neuron output voltages which produce linear dot-product calculations in subsequent layers. This results in a linear multiplication of V_{GS} and V_{th}, and a natural summing of these multiplied current values in-array to form the full dot-product calculation.

3.2.1 Subthreshold Operation

Researchers have long used the characteristics of sub-threshold MOSFET operation to perform exponential analog computation [65, 66]. Known as cut-off, subthreshold, or weak-inversion, this region of MOSFET operation is where:

$$V_{GS} < V_{th} \tag{3.1}$$

Thus, there is no conduction between the drain and the source of the MOSFET because the channel is nonexistent. However, that does not mean that there is no current flow between drain and source. The effect of thermal energy on the Fermi-Dirac distribution accounts for very energetic electrons that are still able to flow through the "channel", resulting in an exponential current equation:

$$I_D = I_0 \times e^{(v_{GS}-V_{th})/n\phi_t}(1 - e^{-v_{DS}/\phi_t}) \tag{3.2}$$

With a small effective channel length, due to drain induced barrier lowering (DIBL), the sub-threshold current equation has a dependence on V_{DS}, as shown in the equation above. However, for long-channel devices, the current flow has little to no dependence on V_{DS}. If we

assume that V_{DS} is large compared to V_{GS} and V_{th}, then $e^{-v_{DS}/\phi_t} \approx 0$ and $(1 - e^{-v_{DS}/\phi_t}) \approx 1$, simplifying the sub-threshold drain current equation to:

$$I_D = I_0 \times e^{(v_{GS} - V_{th})/n\phi_t} \tag{3.3}$$

This equation can be re-arranged to show a multiplicative relationship between the exponentials for gate-to-source voltage and threshold voltage:

$$I_D = I_0 \times e^{v_{GS}/n\phi_t} \times e^{-V_{th}/n\phi_t} \tag{3.4}$$

After reducing and re-arranging the exponential sub-threshold function, we can see that this region of operation can yield a similar calculation as proposed in Section 2.4. These devices can therefore be used for neural network dot-product calculations, given that the exponentials can be eliminated to produce a linear relationship between current flow, V_{GS} and V_{th}. Additionally, because there is little V_{DS} dependence in the current equation, a sub-threshold current calculation will be less affected by the precision with which we are able to hold the drain and source voltages.

3.2.2 Subthreshold Dot-Product

This system was implemented in Cypress' SONOS technology, allowing us to explicitly program all threshold voltages V_{th} in our non-volatile arrays to precise values. In the system presented in Section 2.4, we chose to program these threshold voltages with a linear mapping. This is because we wanted a linear multiplication between the drain-to-source voltage and threshold voltage when the 2T SONOS structure was operating in triode mode (linear). When in sub-threshold mode we program our threshold voltages using an exponential mapping, which converts the exponential relationship between threshold voltage and current to a linear relationship, with the neural network weights defined as: $\omega = e^{-V_{th}/n\phi_t}$. This mapping changes the MOSFET current equation, eliminating the exponential relationship between current and V_{th}:

$$\omega = e^{-V_{th}/n\phi_t}$$
$$I_D = I_0 \times e^{V_{GS}/n\phi_t} \times \omega \tag{3.5}$$

The goal of manipulating the subthreshold drain current equation is to produce an equation with a linear multiplicative relationship between V_{GS} and V_{th}. In this way, we will generate a dot-product proportional to the multiplication of the neural network weights ($V_{th} < i >$) and the neural network inputs ($V_{gs} < i >$). With all 2T SONOS pairs between the same BL and SL as before, natural current summation is achieved on the shared bit-line with amplifiers holding the nodes to precise values.

Once the threshold voltages are programmed to achieve linearity, we need to transform the input voltage to cancel out the exponential relationship it naturally has in the sub-threshold regime. This can be achieved by logarithmically transforming the gate voltage V_{GS} with the input value α as follows:

$$v_{GS} = n\phi_t ln(\alpha) \tag{3.6}$$

When using this transformed V_{GS} voltage in the drain current equation, we see the following changes:

$$\begin{aligned} I_D = I_0 \times e^{v_{GS}/n\phi_t} \times \omega = I_O \times e^{(n\phi_t ln(\alpha))/(n\phi_t)} \times \omega \\ = I_0 \times e^{ln(\alpha)} \times \omega \\ = I_0 \times \alpha \times \omega \end{aligned} \tag{3.7}$$

Thus, by logarithmically transforming the gate voltage to represent input values we are able to develop a sub-threshold drain current that is a linear multiplication between input values and neural network weights.

Expected current draw from each 2T SONOS pair will be less than $1\mu A$, and typically be on the order of $0.1\mu A$, with gate voltage values less than the programmed threshold voltages. Because of the low current draw, this system will require 10s of microseconds to resolve the measured current values. This speed is still faster than the human brain by between $500\times$ and $5,000\times$. Slower processing past this is unnecessary as it will not produce improvements

in energy efficiency.

3.2.3 Exponential Programming

As explained in Section 2.3, it is possible to program the SONOS threshold voltages between -1.5V and +1V. Programming the threshold voltages between 0V and 1V allows us to achieve sub-threshold operation with gate voltages at 0V during calculations. Using the negative portion of the threshold voltage range would require negative voltages at the gate of the SONOS device during calculation, which is more complicated to generate and unnecessary if we are able to up-shift the threshold voltage range to achieve subthreshold voltage (sub-vt) operation.

The programming routine in Cypress' SONOS technology uses two charge pumps to generate high positive and negative voltages (7.2V and -3.8V) to create a $\pm 11V$ differential voltage between gate and body. Using FN Tunneling, the threshold voltages are slowly (on the order of 10s of milliseconds) increased or decreased depending on the sign of the V_{GB} voltage.

Since the programming and erase processes are relatively slow (milliseconds), the routine can be performed off-chip and offline. During a full program routine, the cell will be placed into the program state for 10s of microseconds, then read to determine the threshold voltage and analyzed offline to determine what value is required in the exponential transformation. A series of program $\rightarrow$ read $\rightarrow$ program steps are performed until the threshold voltage is equal to the exponential value required for linearity. These voltages can be programmed with high precision (8-bits) and the programming process is able to cancel out sources of variation in the chip by using the same readout process (circuits) to sense and program the threshold voltages that are used for normal calculations. Because these circuits are re-used, offsets and variations between devices will be negated as they will be factored into the closed-loop, precise programming routine.

As shown in Figure 3.3, the 2T SONOS structure has known program, erase and inhibit curves that allow an off-chip, offline algorithm to precisely set the threshold voltages to achieve different transformations. The current plot shown in Figure 3.3 emphasizes the ex-

ponential difference produced by different threshold voltages into the 2T current cell. This graph shows a 2-dimensional sweep of input current (*i.e.* the logarithmically transformed input from the previous layer) where each curve is a different threshold voltage. The threshold voltages shown were swept linearly and produce exponentially different current results. Since we are able to precisely program threshold voltages into the SONOS cells, we can select threshold voltages that are linearly spaced in the current domain.

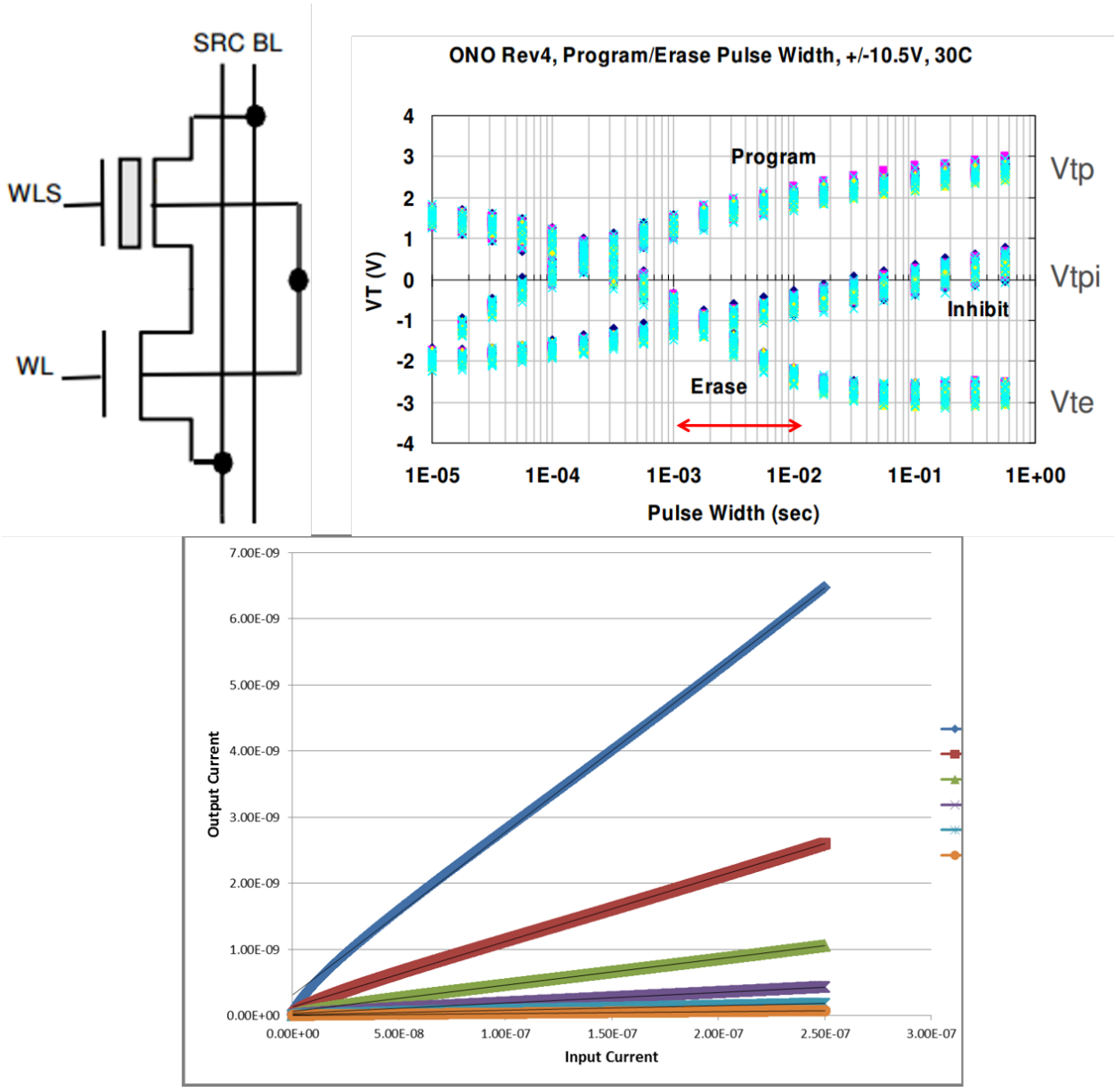


Figure 3.3: **2T SONOS Cell Program/Erase and Exponential Curves.** The 2T SONOS structure described in Section 2.3 has verified relationships between the time given for programming and erasing, and the amount of threshold voltage shift achieved. These curves are used during the offline programming feedback loop in order to achieve an exponential transformation between the neural network weight values and the threshold voltage programmed in the SONOS array. The simulated graph shows the current relationship between V_{th} and V_{DS}. This graph was made using an ideal logarithmic input voltage generator, and each line is a different threshold voltage. When properly programmed, we are able to transform this graph into separate, linearly spaced lines - an exponential transformation in the programmed threshold voltages.

3.2.4 Logarithmic Input Generation

In order to achieve linear multiplication, as described in Section 3.2.2, we need to logarithmically transform the input voltages to cancel out the exponential in the sub-threshold drain current equation. While several previous works have explored methods of developing logarithms on silicon [67, 68], we have chosen to again take advantage of the exponential properties of the sub-threshold region to accomplish this task.

Similar to the calculation strategy discussed in Section 2.4, the proposed circuitry uses amplifiers to hold the drain and source nodes of the 2T SONOS pairs to precise voltages. In the above-threshold version we used a resistor in feedback with the calculation (middle) amplifier to generate a voltage proportional to the current differential between the excitatory and inhibitory SONOS arrays. The resistor provides a linear transfer function with which to generate the proportional calculation voltage. In the sub-threshold version we replace the resistor with a diode connected MOSFET, as shown in Figure 3.4.

This system uses only excitatory neurons for a proof-of-conecpt calculation demonstration. In a full neural network implementation there would also be inhibitory neurons to provide current in the opposite direction. Because current will always be pulled *out* of the calculation amplifier, we know that the diode connected node is the *source*, as the output of the amplifier will always be larger in order to provide current, and is labeled the *drain*.

$$\begin{aligned} V_G &= V_S \\ V_{GS} &= 0V \\ V_{GS} &< V_{th} \end{aligned} \tag{3.8}$$

The diode connection of this MOSFET, and the current driving capability of the calculation amplifier, allows us to ensure that the device is operating in the subthreshold region, and will also have an exponential current relationship.

$$\begin{aligned} I_D &= I_0 \times e^{V_{GS}/n\phi_t} e^{-V_{th}/n\phi_t}(1 - e^{-V_{DS}/\phi_t}) \\ &= I_0 \times e^{0/n\phi_t} e^{-V_{th}/n\phi_t}(1 - e^{-V_{DS}/\phi_t}) \\ &= I_0 \times e^{-V_{th}/n\phi_t}(1 - e^{-V_{DS}/\phi_t}) \end{aligned} \tag{3.9}$$

Since V_{th} is constant in the case of a standard NMOS (as opposed to the equations derived previously for SONOS multiplication) we can reduce the current relationship to the following equation, where $K = I_0 \times e^{-V_{th}/n\phi_t}$:

$$= K \times (1 - e^{-V_{DS}/\phi_t}) \tag{3.10}$$

We previously proved that the current draw through the devices is linearly proportional to the multiplication of the inputs and weights - so the current is a known value and can be used to solve for the V_{DS} relationship. Solving for V_{DS} tells us what type of transformation is seen on the output of the calculation amplifier, and therefore what type of signal can be sent as an input to the next layer of the neural network. As derived previously, to achieve linear multiplication we want a logarithmic transformation of the voltages to the input of the next layer. Solving for V_{DS} gives us the following equations:

$$\begin{aligned} e^{V_{DS}/\phi_t} &= I_D - K \\ V_{DS}/\phi_t &= ln(I_D - K) \\ V_{DS} &= \phi_t \times ln(I_D - K) \end{aligned} \tag{3.11}$$

Based on these equations, we can see that V_{DS} is now logarithmically related to the drain current, which correctly allows us to have a logarithmic relationship between the input voltage of the next layer (V_{DS} in this example) and the current output of the previous layer. Now, when we insert V_{DS} into the current equation representing the drain current in the *next layer* neurons (V_{GS}), we get the following equations:

$$\begin{aligned} I_D &= I_0 \times e^{v_{GS}/n\phi_t} \times \omega \\ &= I_0 \times e^{(\phi_t \times ln(I_{i-1} - K))/(n\phi_t)} \times \omega_i \\ &= I_0 \times (I_{i-1} - K)^{1/n} \times \omega_i \\ &= I_0 \times \alpha_i \times \omega_i \end{aligned} \tag{3.12}$$

The logarithm cancels out with the exponent from the sub-threshold current equation, producing a current relationship that is proportional to the linear multiplication of the previous layer's output I_{i-1} and the current layer's weight ω_i. When summed together on

the shared array bit-line, the sub-threshold SONOS cells produce a total current that is proportional to the dot-product of all inputs and weights. Figure 3.3 shows a simulation of the logarithmic transformation output, as well as the resulting semi-linear current flow in the subsequent layer. The output voltage is measured on the output of the calculation amplifier (on the drain side of the diode connected MOSFET), and ranges from 0.5V to about 0.86V, producing a linear response in the output current of the neurons in the next layer of the network.

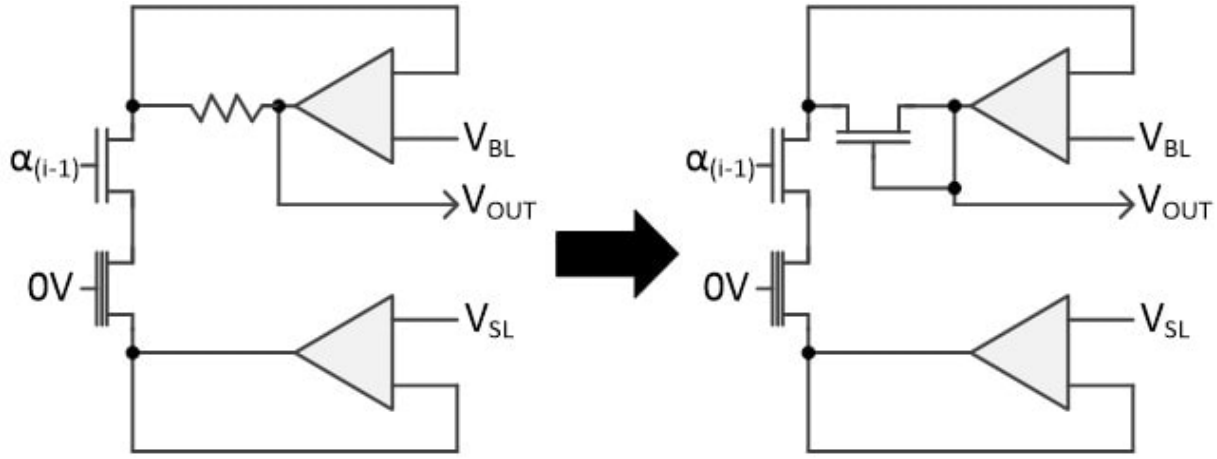


Figure 3.4: **Sub-threshold Neuron Calculation with Logarithmic Transformation.** Similar to the above threshold version proposed in Section 2.4, the sub-threshold version replaces the feedback resistor with a diode connected MOSFET. The diode in feedback produces a logarithmic neuron function that changes the linear multiplication of the input $\alpha_i \times \omega_i$, to the log domain for the input to the following stage.

3.3 Full System Implementation

The proposed system was taped out in Cypress 130nm SONOS technology. The test chip implements a single layer of a deep neural network and is used to determine the implications of a sub-threshold calculation accelerator on the speed, power, and accuracy of the full network. This network is tested at speed, and all programming algorithms will be fully implemented in software off-chip. Additionally, to completely test the accuracy and effect of this circuit, the subsequent layers of the neural network will be implemented off-chip. To test the linearity of the sub-threshold dot-product calculation, neuron output voltages are

calculated and stored off-chip, then fed directly back into the neuron layer inputs to see the resulting output voltage, shown in Figure 3.5. This effectively allows us to test multiple layer interactions on-chip.

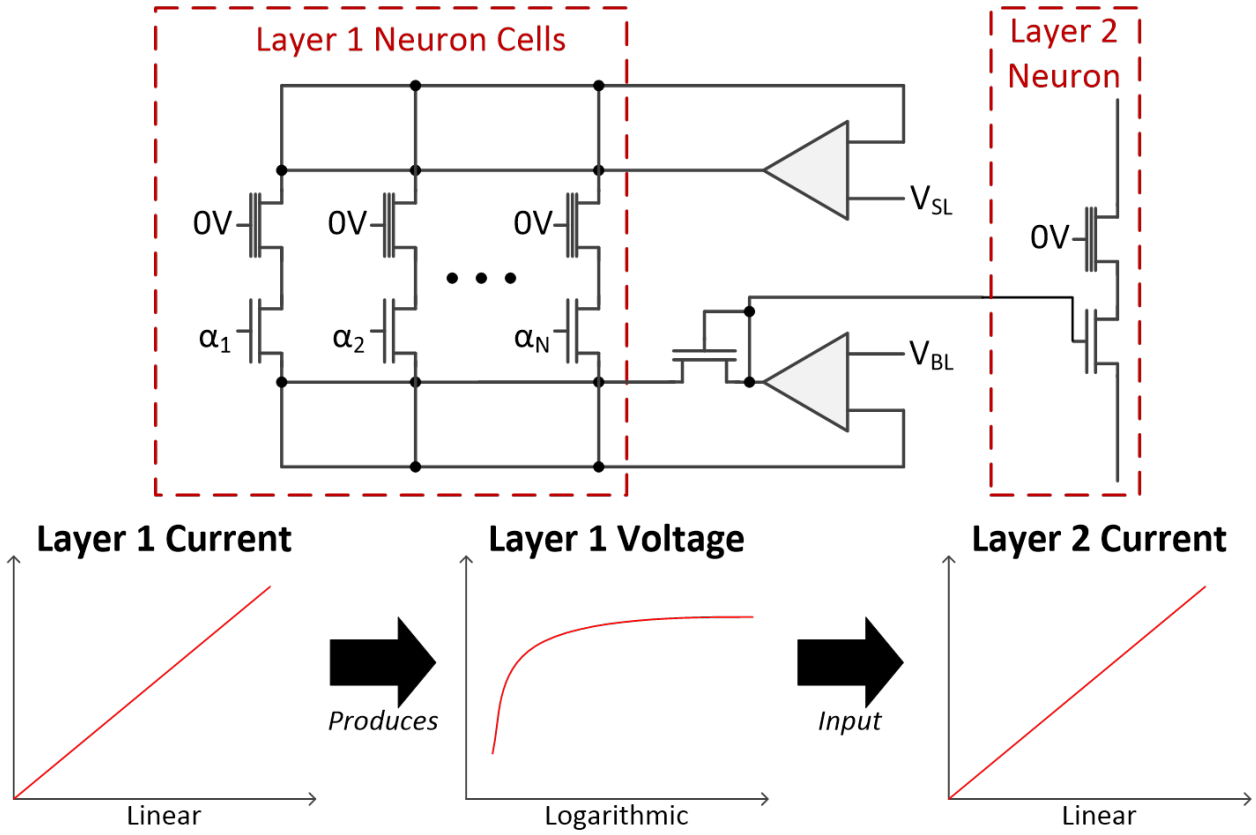


Figure 3.5: **Layer to Layer DNN Connections.** The fabricated system measures dot product linearity by applying a set of inputs and weights to the fabricated SONOS cell layers. This layer produces a total neuron current. When scaled linearly, this neuron current produces a logarithmic output voltage on the diode connected MOSFET. To test second-layer linearity, these logarithmic output voltages are applied back onto the input devices to produce a linear set of currents again.

3.3.1 System Architecture

The proposed system architecture is shown in Figure 3.6. The testchip includes a single synaptic array, with 228 inputs and 64 neurons. Similar to the array described in Section 2.5, this allows us to implement 6-bit input values with only 3-bits of available analog voltages in the input drivers. In addition to input drivers, the proposed system has multiplexers

to select between the 64 neurons, the diode connected MOSFET to readout neuron values, and positive/negative charge pumps to perform precise program operations for exponential threshold voltage values.

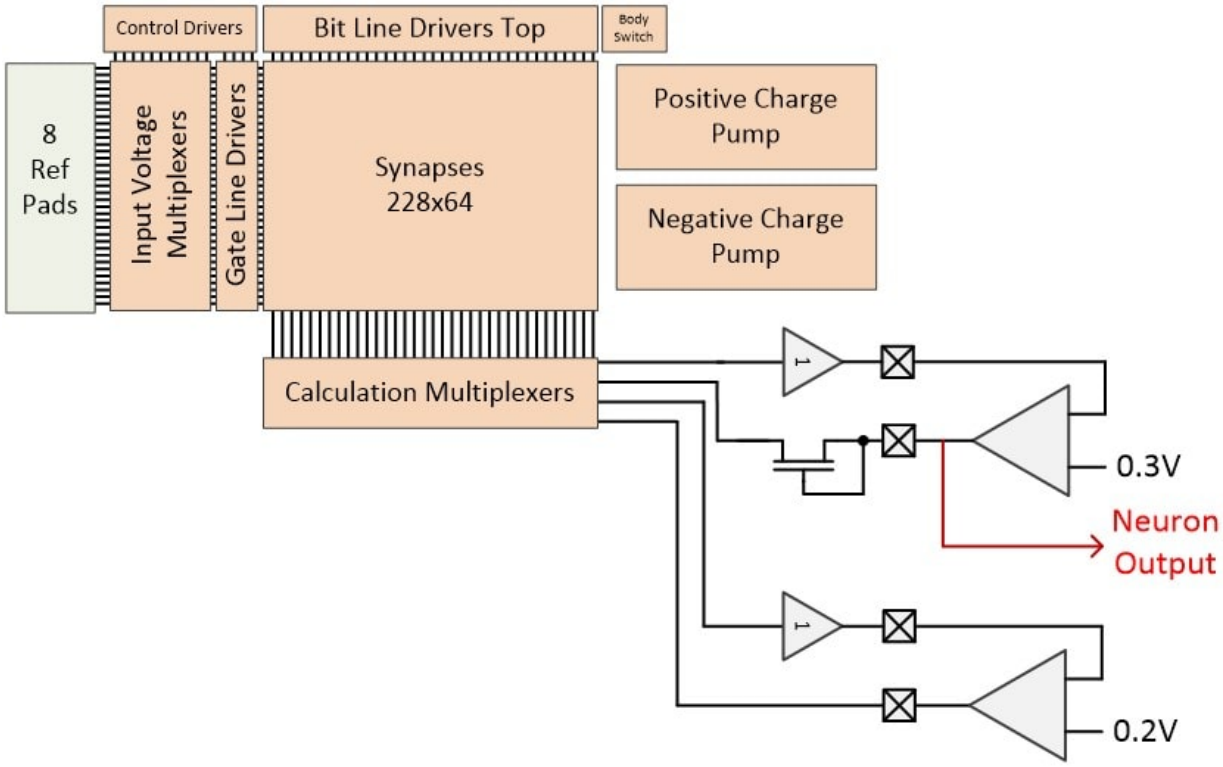


Figure 3.6: **Implemented System Architecture for Sub-threshold Neural Network Chip.** The proposed sub-threshold chip requires one SONOS array of synaptic weights with 228 inputs and 64 neurons. The taped out chip also includes all necessary multiplexers between input voltages and multiplexers between neurons. The diode connected MOSFET is used to readout the neuron voltage to be sent to the simulated next layer of the network.

Figure 3.5 shows the layer-to-layer connections that are demonstrated through the proposed system. With only excitatory neurons in the test chip, there are 228 2T current cells that take in the input voltage as a gate voltage to the SONOS cells (α_i) and have programmed threshold voltages ($V_{th} < i >$) to represent the neural network weights. With N synaptic connections (weight $\times$ input pairs) all sharing the same source and drain connections, these currents naturally sum and are sent through the diode connected MOSFET on the output of the calculation amplifier. The voltage on the drain-side of the diode connected FET is logarithmically proportional to the total current flowing through all N current sources. This voltage is then stored and sent to a *single* input cell in the subsequent layer. proposed

strategy.

3.3.2 Neural Network Requirements

The proposed neural network implementation has requirements for the precision of the threshold voltages, input voltages, and the calculated current and voltage values. These precision and resolution requirements dictate the accuracy with which the full neural network will be able to train and to recognize different data sets. In order to fully understand and properly design the proposed system, we worked with a state-of-the-art neural network simulator [69]. This simulator allows us to train and run large data sets on the network AlexNet, that was the winning algorithm from the 2012 ImageNet competition [48]. This simulator also allows for changes in the algorithm, neuron types, synaptic weights and responses. Our implementation used a modified set of neurons that could more accurately model the non-linearities and resolution restrictions of the proposed hardware implementation.

The ability to accurately model hardware neural networks is critical to the success of these implementations. Programming trained weights from an ideal software neural network will not produce the same results (accuracy) in the hardware version unless the hardware neural network is an identical copy of software. As such, it is necessary to modify the *software* to accurately model the implemented hardware, so that the neural network algorithm is able to train with non-ideal weights, neuron functions and readout strategies.

Figure 3.7 shows the effect on neural network accuracy caused by different resolution requirements in threshold voltage (V_{th}) and calculation (output voltage and current). The top graph shows the simulated DNN error (misidentification of images) on the ImageNet database over different threshold voltage resolutions. This plot was obtained through quantizing the weight values in the neural network in the Caffe framework, training and simulating the entire network on the ImageNet database. At or above 5-bit resolution in weights does not significantly change (*i.e.* no trend downwards) the DNN error. In fact, truly analog (or floating point) weights do not show a change in error, which are represented on this graph as a resolution of infinity (*i.e.* maximum quantization).

The bottom graph in Figure 3.7 plots the normalized DNN accuracy as compared to

the precision with which we are able to perform the dot-product calculation. Between 4-bit and 6-bit precision on the calculation show optimal accuracy values. Below 4-bit resolution becomes too noisy and accuracy values quickly decrease to 0.6 at 1-bit resolution. Above 6-bit resolution is more complicated, as typically we would expect higher resolution calculations to produce better (or at least not worse) accuracy values in the neural network. However, in simulation the accuracy of the neural network slightly decreases from a normalized value of 1 at 6-bit down to about 0.93 at 10-bit. This decrease in accuracy with increasing resolution is caused by over-fitting of the neural network. Over-fitting, or issues with training, have a significant effect on the accuracy of the DNN on a data set. The goal of training is to teach the network a set of images by adjusting weight values until the network can accurately recognize those images. Then, the networks is tested on a different (but similar) set of images to determine the accuracy. If the network was trained too much, or too hard, on the training data set, then it can become an extremely accurate predictor of those images, but be too specific and perform worse on the test set. The neural network weights should represent a generalization of the trained data set, not the exact data set itself, so that it can recognize different but similar images as well.

Figure 3.14 shows an estimation of the power and area benefits of using the proposed subthreshold, SONOS-based design. Power reductions over a graphics processing unit (GPU) implementation are estimated to be as low as 100× and with an optimized digital ASIC, the synthesized power consumption estimations around around 40× lower in the NVM design. In addition to significant power benefits, the proposed system sees a large reduction in area compared to using standard SRAM for weight storage. Storing 8-bit weight values in a single SONOS cell versus 8 SRAM cells gives a 96× reduction in area. Additionally, using a non-volatile memory eliminates the need for off-chip memory accesses and storage during operation, allowing for even more area and power benefits that are not accounted for in previously published results. All hardware neural network implementations that use more than 1-bit neural network weights require some form of non-volatile memory storage. The area, power and energy cost of memory accesses, and bandwidth limitations must be further analyzed in order to fully understand the benefits of the proposed system.

The proposed system full layout is shown in Figure 3.8, and consists of the full digital

FSM, the analog design block including the SONOS arrays, calculation amplifiers and multiplexers and charge pumps, as well as analog buffers designed to readout the neuron outputs to determine linearity in the sub-threshold regime.

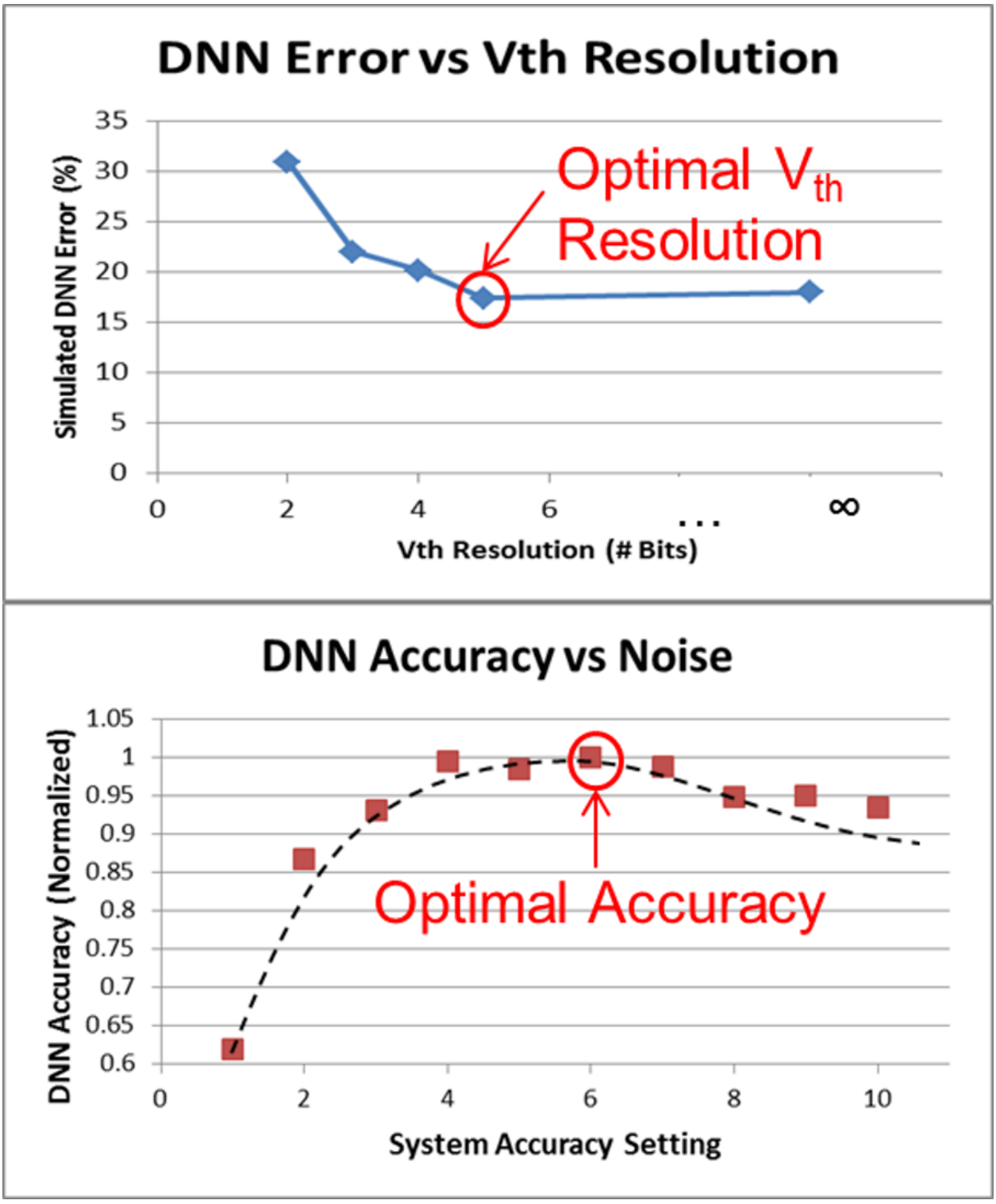


Figure 3.7: **Optimal Accuracy Curves for Threshold Voltage and Input Voltage.** (Top) Simulated neural network error with varying threshold voltage resolution. The optimal (lowest resolution value that still produces the "best" error) is at 5-bit. (Bottom) Simulated neural network accuracy with varying resolution on input values and on calculation accuracy. Optimal accuracy is at about 6-bit, but 4-bit accuracy also produces a similar accuracy value.

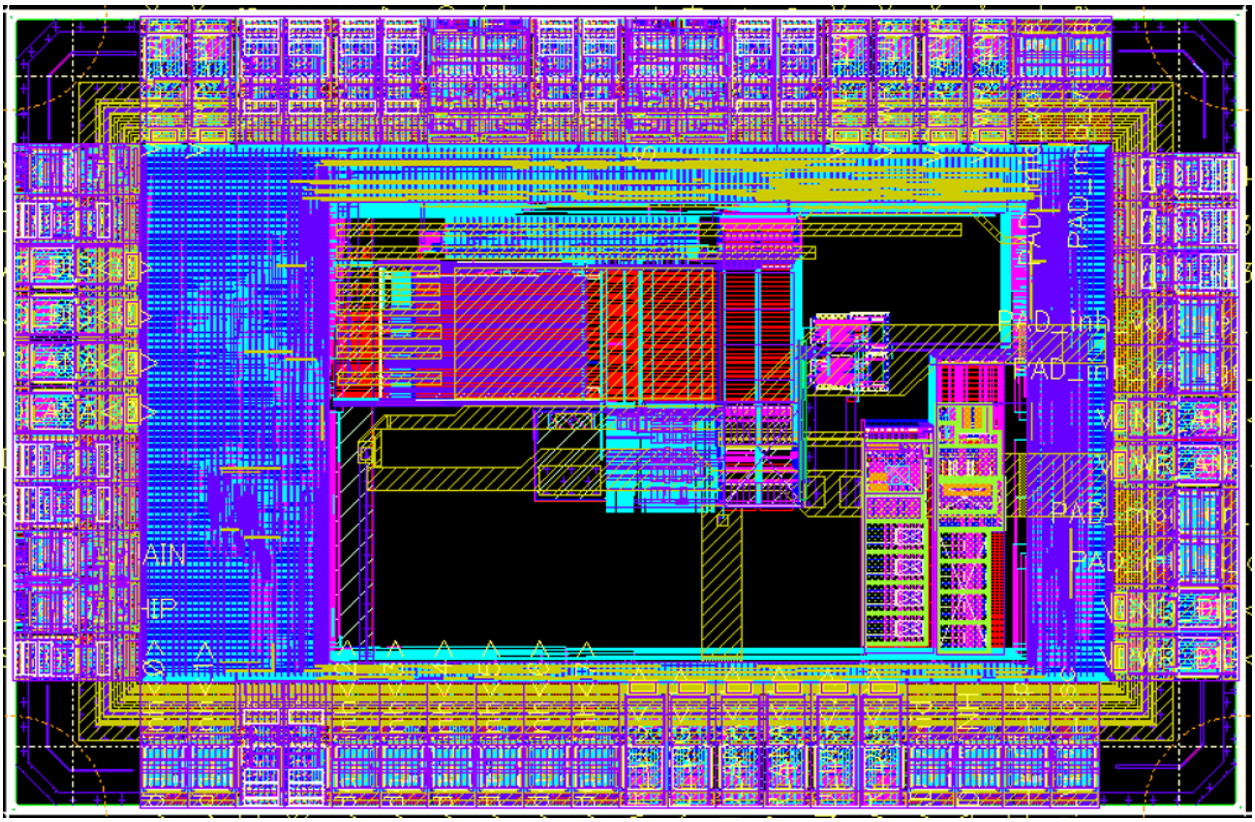


Figure 3.8: **Layout of Full System Design.** Layout of the full system design including the digital FSM, analog design block and amplifiers for driving analog signals off-chip.

3.4 Measured Results

The proposed system was fabricated in Cypress' 130nm SONOS technology. To analyze the subthreshold multiplication strategy we implemented one 228×64 array of synaptic weights with diode connected MOSFETS for readout. The amplifiers required to hold V_{DS} at a constant value and force a logarithmic output voltage on the diode were provided from off-chip. In a full system implementation these amplifiers would also be designed in the sub-threshold region for ultra-low power consumption, and would be placed on-chip. Because there is a reduced accuracy requirement for V_{DS} precision, these amplifiers could be small and pitch matched to the array. This would allow for fully parallel readout circuitry, compensating for the reduced speed due to sub-threshold currents.

Figure 3.9 is a measured sweep showing both programmability and logarithmic readout. Each point in the plot is generated during the programming routine. A short programming pulse (5 μs) is applied to the SONOS cells, shifting the threshold voltages. After each pulse, we read the cell current by applying 1.8V to the gate of the access transistors, reading the current and the diode connected output voltage. This routine is repeated until the cells are fully-OFF. The results show a logarithmic voltage output between 0 and 1μA SONOS cell current. Above 1μA the SONOS cells and the diode connected MOSFET are in the above-threshold region.

When plotted in the log-domain, shown in Figure 3.10, the output voltage shows a very linear response, proving that the generated output voltage is logarithmic. Voltages from the high voltage diode connected MOSFET range from 0.577V to 0.762V, noticeably larger than the voltages generated by the short and low threshold voltage (LVT) MOSFETs. Because of this discrepancy, and because the access transistors are high voltage devices, voltages generated by the short and lvt MOSFETs need to be shifted up by 0.1V and 0.2V respectively, in order to produce linear results in subsequent network layers.

The logarithmic voltages generated in Figures 3.9 and 3.10 are fed into subsequent neural network layers as input voltages. These voltages are applied to the gate of the access transistors in the 2T SONOS cell array as shown in Figure 3.5. When a linear current is generated by the previous layer, a logarithmic voltage is output from the readout circuitry.

This logarithmic voltage is fed into the subsequent layer and produces a linear current. This allows for a linear multiplication between inputs and weights, producing a linear neuron dot-product. Figure 3.11 shows the linear relationship between output current of Layer 1 versus the output current of Layer 2. Plotted in both dimensions (top and bottom graphs) these results show a linear multiplication of threshold and access gate voltages.

In order to test the efficacy of the full dot-product calculation, we applied a random set of input voltages and synaptic weight threshold voltages to the system. For each set of inputs and weights we calculated the expected dot-product and plotted against the measured value. The inputs and weights are multiplied together using flash-based subthreshold analog calculations, and summed along a shared bit line to produce a total neuron current. The results are shown in Figure 3.12, and the measured system achieves a dot-product error r-squared value of 0.9994.

In addition to achieving a linear sub-threshold dot-product calculation, the demonstrated circuit is capable of functioning linearly in above-threshold mode. As shown in Figure 3.13, when the input current of Layer 1 is increased above the logarithmic output voltage range (above-threshold), the output current of Layer 2 remains linear in relation to the current of Layer 1. The diode connect MOSFET effectively acts to invert the current-voltage relationship, and thus changes depending upon the region of operation. This allows the proposed system to operate in both sub- and above-threshold regions, using the same diode readout structure in both regions. In practice, this could be useful when different amounts of accuracy are needed, or faster operating times. The sub-threshold region consumes very little current but is also slow (microseconds), and selectively switching to above-threshold operation could trade-off speed and power.

Total neuron current is between 0 and 1μA of current, with an average output near 500nA. Total power consumption for a single neuron is 900nW, around 40$\times$ smaller than the cell current from the above threshold neurons, shown in Figure 3.14. Based on published sub-threshold operational amplifier power and bandwidth results presented in [70], we can estimate that the calculation amplifier consumes 75-nW and operates with a settling time of 100μs. Compared to previously published work, the estimated power consumption for the proposed system is 9500$\times$ smaller. When factoring in estimated bandwidth of the amplifier,

energy per multiply-accumulate function is estimated to be 0.43 pJ/MAC, roughly 5 $\times$ smaller than the work presented in [57].

The proposed system was fabricated in a 130nm SONOS process, and the die shot is shown in Figure 3.15.

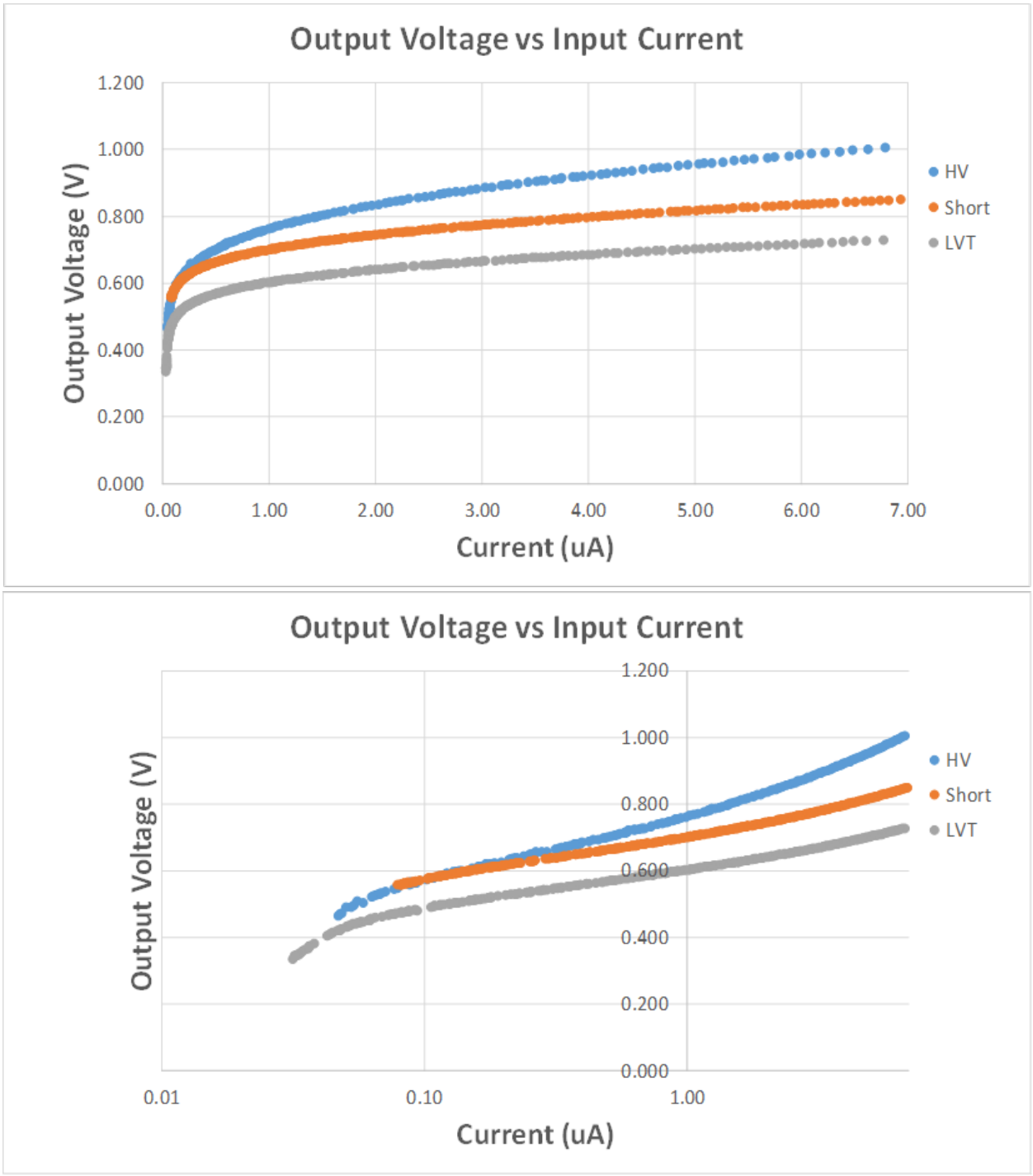
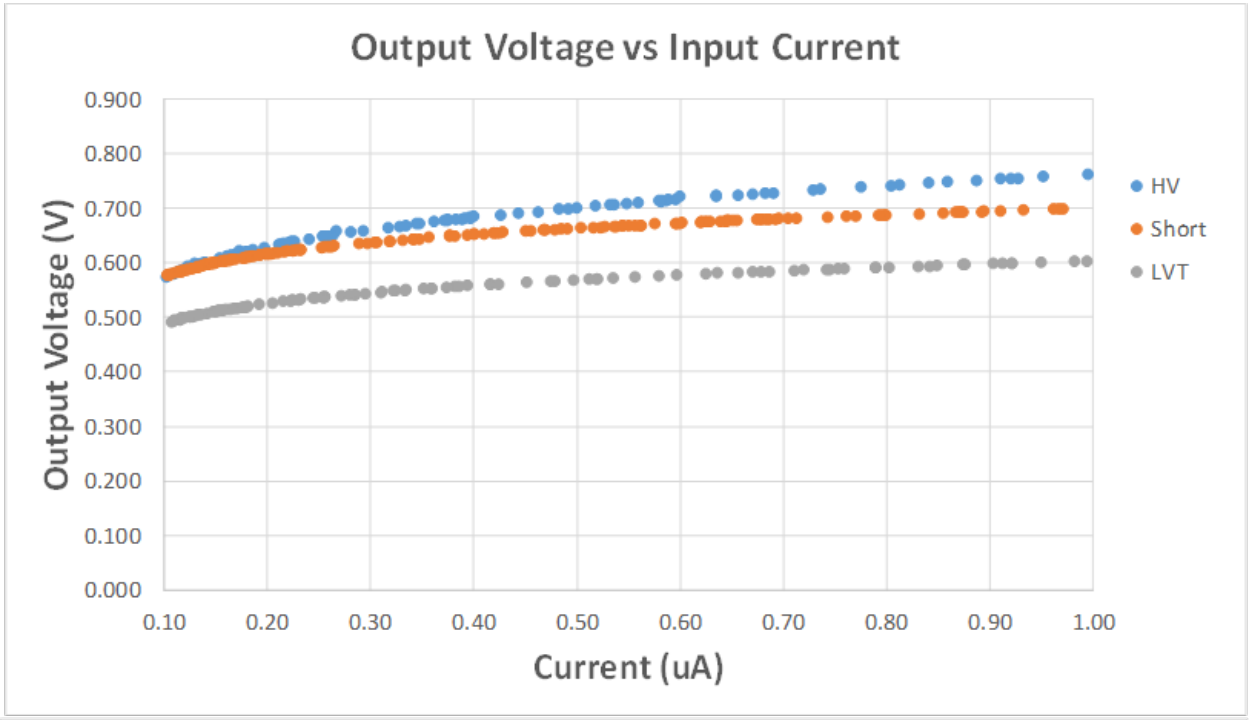


Figure 3.9: **Logarithmic Output Voltage Sweep.** SONOS cells in the synaptic weight array are programmed to precise current values (x-axis). At each program pulse the current is measured and the output voltage from the diode connected MOSFET is recorded. This process is continued down to fully off devices. This sweep shows a logarithmic response in output voltage between 0 and 1μA.



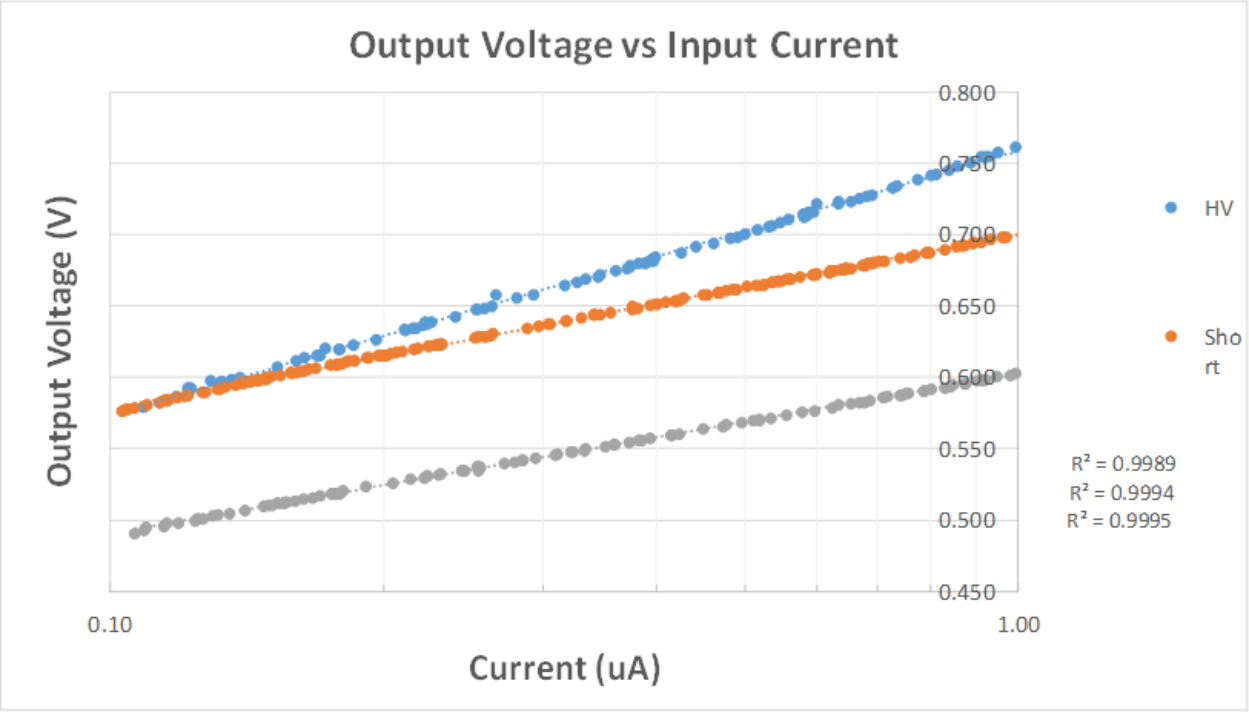


Figure 3.10: **Log Domain Output Voltage Sweep.** Figure 3.9 is plotted in the log-domain on the x-axis, showing linearity in the log-domain verifies that the output voltage between 0 and 1μA is logarithmic. This voltage range will be used as input voltages for subsequent neural network layers as voltages at the gates of the access transistors.

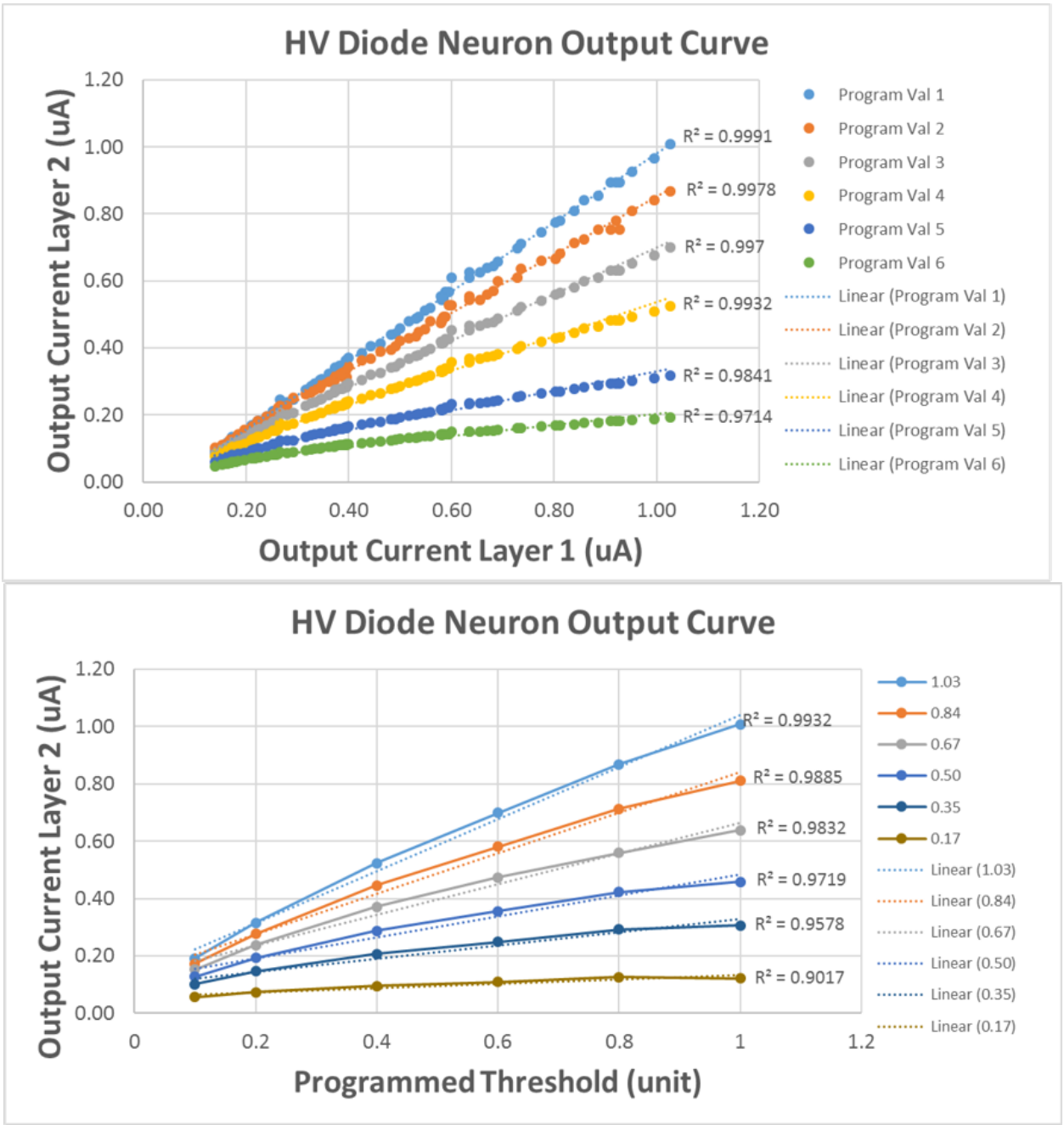


Figure 3.11: **Output Current Sweep Using HV Diode Voltages.** (Top) Output current of layer 2 is linear in relationship to the output voltage, or current, of layer 1. (Bottom) Output current of layer 2 is also linear with the programmed threshold voltages, therefore we achieve a linear dot-product.

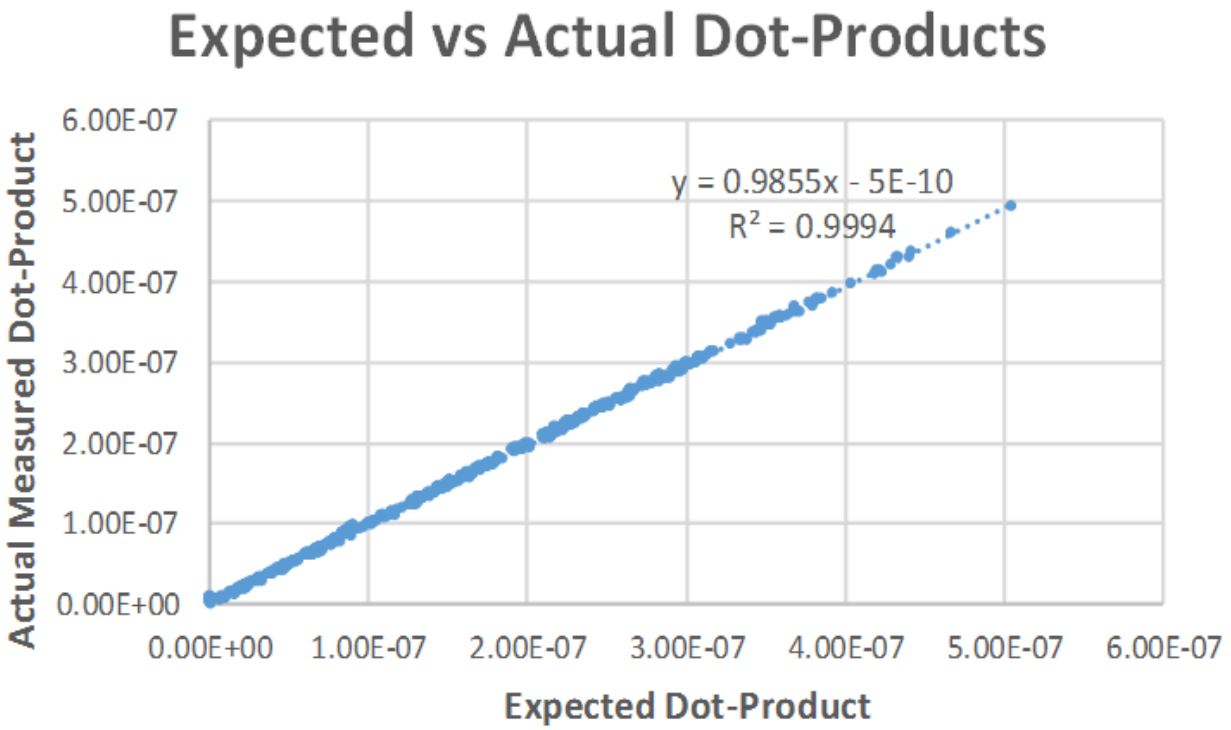


Figure 3.12: **Measured versus Expected Dot-Product Calculation.** Analog dot-product calculations are plotted versus expected values, using 626 different combinations of synaptic weights and analog input voltages. Four input lines are summed together with these different combinations to produce a full dot-product output within the subthreshold region of operation.

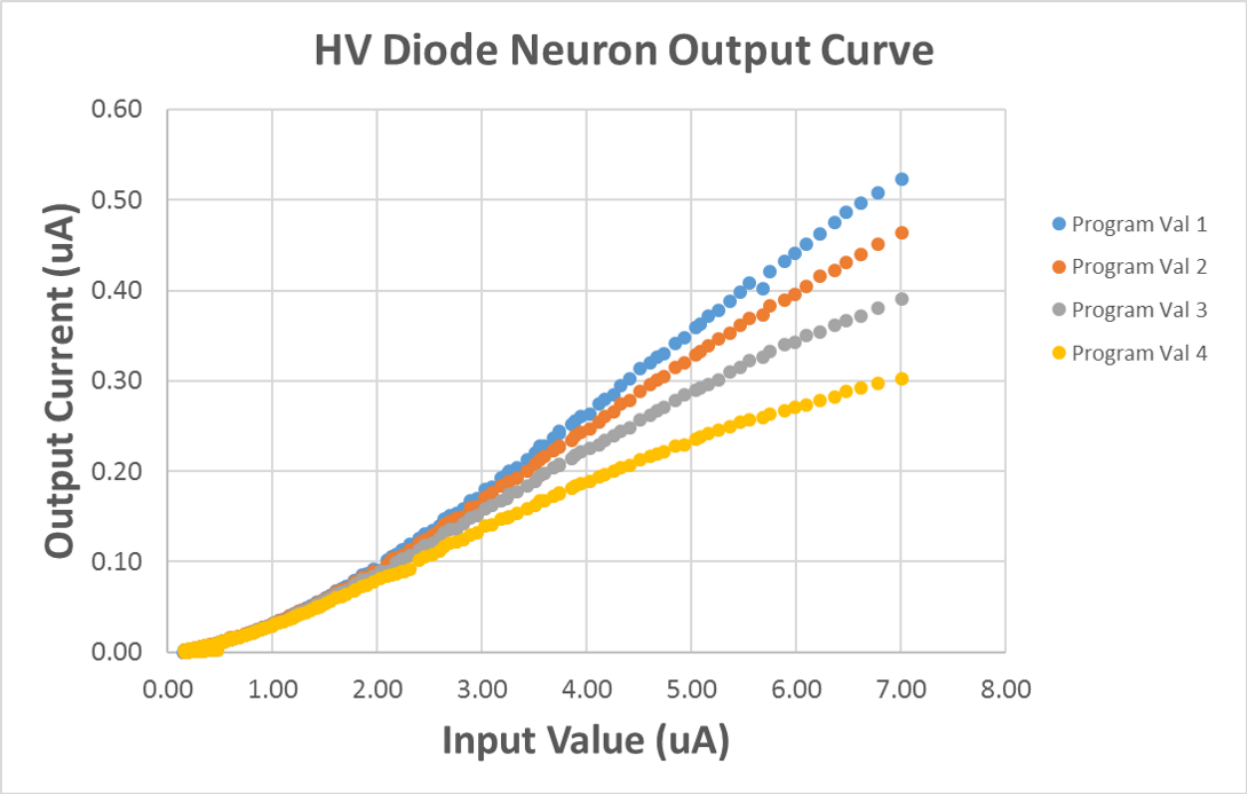
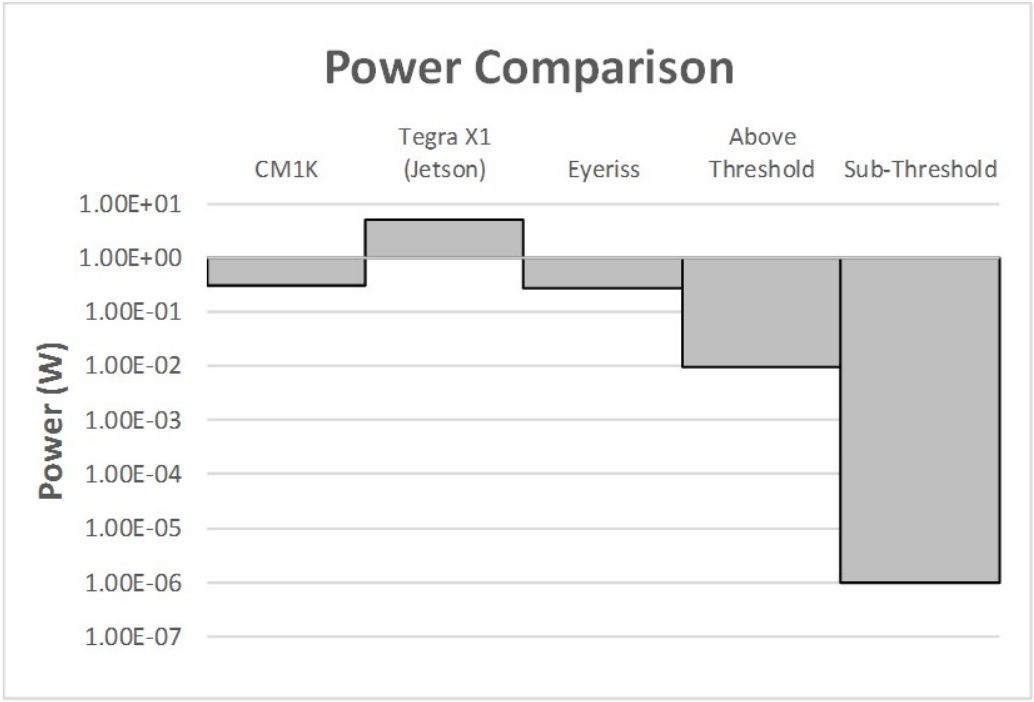


Figure 3.13: **Above Threshold HV Diode Sweep.** When neuron currents are above threshold, outside of the previously defined logarithmic output voltage region, the diode acts as a resistor instead of a logarithmic conversion device. In practice the diode connect MOSFET acts to invert the current relationship, and thus changes depending on the region of operation. This allows the demonstrated system to transition between sub-threshold and above-threshold operation depending on system requirements.



	Units	CM1K	Tegra X1 (Jetson)	Eyeriss	Above Threshold	Sub-Threshold
Neural Network Type	**Type**	RBF and KNN	DNN	DNN	DNN	DNN
Off-Chip Memory?		No	Yes	Yes	No	No
Non-Volatile?		No	No	No	Yes	Yes
Technology	**nm**	130	20	65	130	130
Parallel Workloads Required	**#**	1	1	4	1	1
Power	**W**	3.00E-01	5.10E+00	2.78E-01	9.30E-03	9.75E-07

Figure 3.14: **Comparison Table and Power Results.** Power measurements of the sub-threshold cells are an average of 900nW for a single neuron, 40× smaller than the cell current from the above threshold cells.

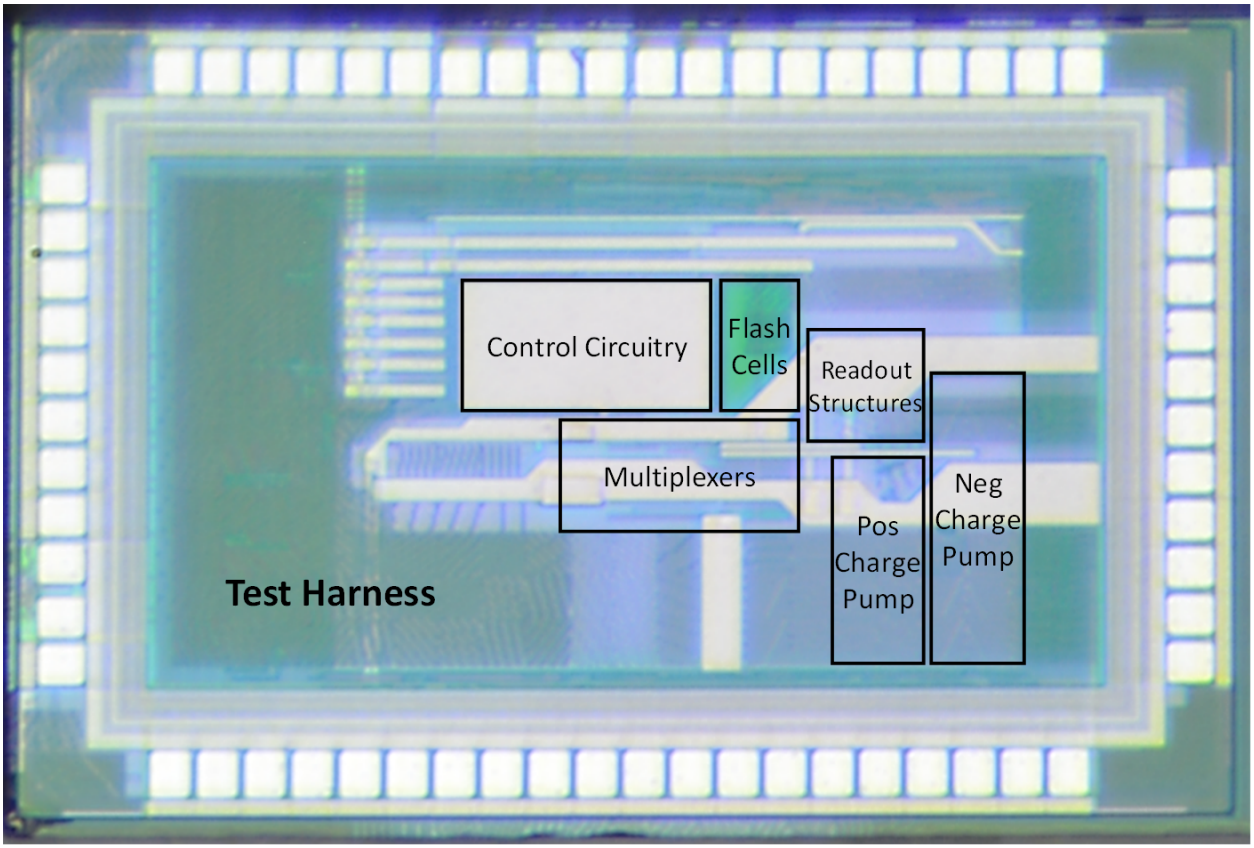


Figure 3.15: **Die Shot of Subthreshold Neuromorphic Chip.** Die shot of fabricated subthreshold neuromorphic chip in a 130nm SONOS process.

CHAPTER 4

A $346\mu m^2$ VCO-Based, Reference-Free, Self-Timed Sensor Interface for Cubic-Millimeter Sensor Nodes in 28nm CMOS

4.1 Motivation

Wireless sensor nodes form the foundation for many cutting-edge solutions in infrastructure monitoring, environmental monitoring, and medical applications, and are often deployed in hard-to-reach places that require a small form factor [71]. These applications create new challenges in circuit design, where for the smallest applications, sensor nodes can be cubic millimeter in volume with microwatt-level power consumption, requiring small, voltage scalable integrated circuit (IC) designs that emphasize area and energy efficiency. An example system is shown in Figure 4.1. This system is $1.0mm^3$ and includes a pressure sensor to monitor pressure in tumors for determining the effectiveness of chemotherapy treatments [72]. When sensor data are recorded and later post-processed, such as eye pressure monitoring of glaucoma patients or strain-gauge monitoring of bridges, large trade-offs in sensor interface design can be leveraged to achieve efficiency goals. Salient features to trade away include speed and linearity. Linearity in particular is an unusual feature to trade-off since it closely relates to effective number of bits (ENOB). However, in these applications a full look-up table can be available via off-line post-processing, and high-frequency signals are not being reconstructed from the data [73].

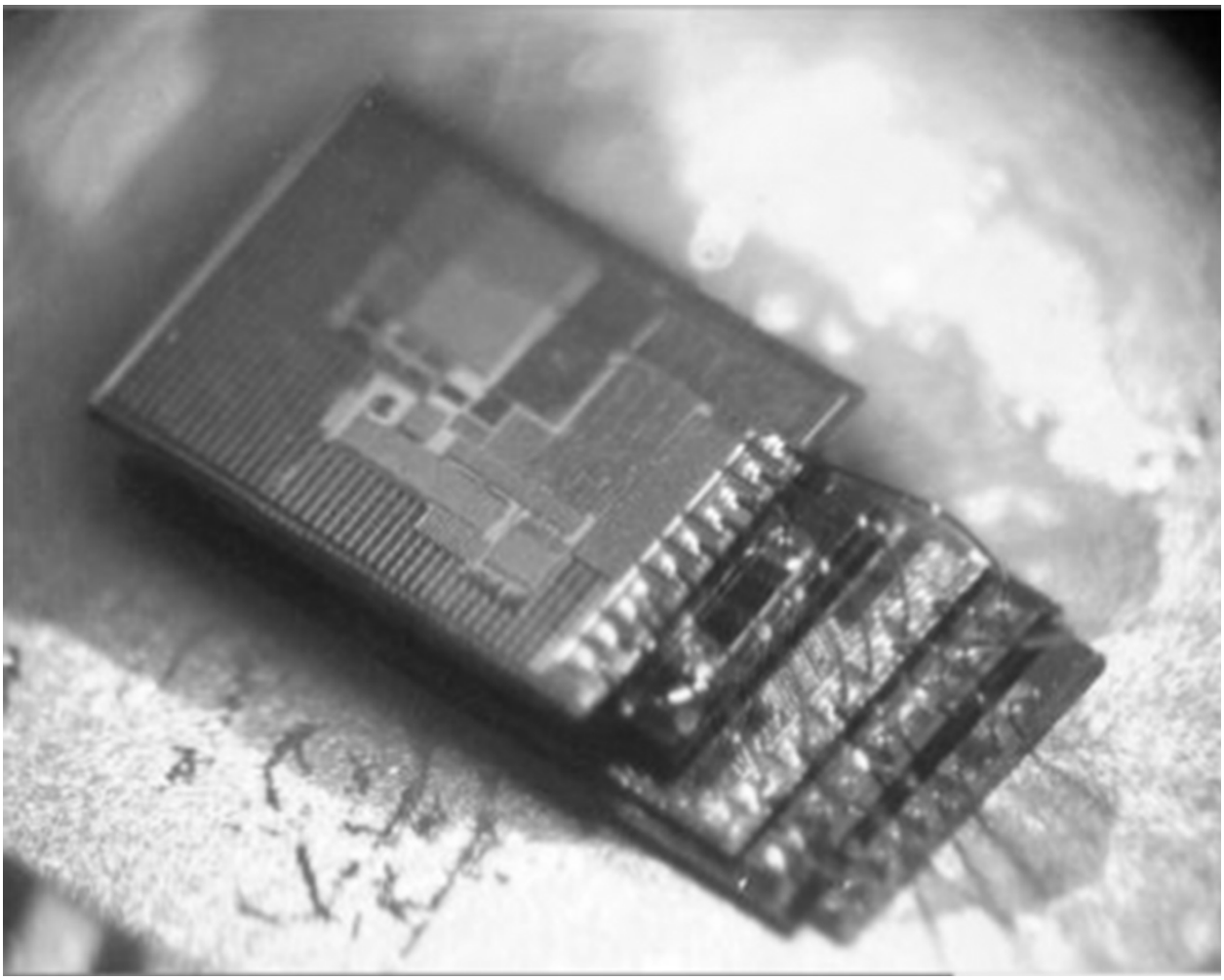


Figure 4.1: **Target Application.** Millimeter-scale wireless sensor nodes that cannot support high accuracy references

Healthcare monitoring provides one particularly compelling application area for wireless sensor nodes. Researchers have developed implantable devices that can monitor and record health data to relay critical information to doctors, or to operate in a closed loop fashion. A millimeter-sized intraocular pressure sensor [74] can be inserted into a glaucoma patient's eye to record continuous eye pressure measurements, giving doctors a more complete history of daily pressure fluctuations and allowing for more informed decisions for treatment options. These measurements can be post-processed off-chip [73], which enables sensor interfaces such as that described in this work, achieving energy efficiency in a compact footprint.

Energy efficiency can be improved by using an interface that scales energy requirements with performance requirements. Wireless sensor nodes often employ multiple sensors, each

with varying resolution requirements. The example system in Figure 4.2 has three sensors: a 9-bit strain gauge, 7-bit temperature sensor, and 5-bit battery sensor. One way to interface with all three sensors is to implement a 9-bit ADC, reading all sensors at the maximum resolution of 9-bits. This strategy wastes energy by reading the temperature and battery sensors with more accuracy than necessary. Assuming equal use of all sensors, up to 61% energy reduction could be achieved by reading each sensor with its appropriate resolution. Using this architecture, assuming that energy decreases by a factor of $(1/C)^R$, where R represents the number of resolution steps and C is a scaling factor, then a 7-bit interface would use $(1/C^2)Energy_{(9-bit)}$ and a 5-bit interface would use $(1/C^4)Energy_{(9-bit)}$, resulting in a total energy reduction of $(1/3)(2 - 1/C^2 - 1/C^4)$. Based on simulation results, C was calculated to be roughly equal to 2.6, resulting in a total energy reduction of 61%. In measured results, a 7-bit sensor used 19.25% of the 9-bit sensor energy, and the 5-bit sensor used 15.41% of the 7-bit sensor energy (3.01% of the 9-bit sensor). This resulted in measured energy reduction of 59%. This could be achieved by implementing three separate ADCs [75], each with different resolution capability, where each only communicates with its designated sensor. This increases the overall area and design cost by requiring a large number of readout circuits, and can increase energy consumption due to tail currents. A resolution scalable sensor interface can reduce energy consumption while occupying a small area, allowing the sensor node to scale energy with performance requirements by changing the resolution setting for each sensor individually.

Traditional ADC designs, such as successive-approximation register (SAR), achieve moderate resolution and energy efficiency through high accuracy capacitor arrays (DACs) and comparators, which require high accuracy references. However, high accuracy current, timing, and voltage references are often not feasible in wireless sensor nodes and these systems lack the space and power budget to include on-chip references. Many off-chip references require multiple components and are too large for such applications. For example, a typical off-chip current reference [76] is $24mm^2$ (packaged) without required resistors and capacitors, operates with a minimum 1V supply, and dissipates 400mW. Each of these specifications is individually infeasible for wireless sensor nodes, and the power requirement may even cause tissue damage when used in the eye pressure monitor. Due to the capability for off-chip

correction, highly-linear sensor readout circuits - and their associated references - are unnecessary. A previously published ADC design intended for use in wireless sensor nodes [77] attempts to address this design space by creating a resolution configurable, low-energy SAR which can operate at 8 or 12 bits. However, this implementation is large at $0.63mm^2$, consumes 25μW, and must operate at 1V V_{DD}.

Other common design approaches for sensor interfaces include sigma-delta, flash, and pipelined ADCs. Sigma-delta designs achieve high resolution (12-18 bits) through filtering and decimation. These designs are typically high power (36μW to 2.9mW) [78, 79] and have large area ($0.03mm^2$ to $2.32mm^2$) [80, 81]. A high accuracy timing reference for the modulator and digital filter is also required. Flash ADCs are extremely fast, but consume significant power (460μW to 0.54W) [82, 83] and are very large ($0.033mm^2$ to $0.88mm^2$) [84,85] due to the need for 2N-1 comparators in an N-bit converter. Finally, pipelined ADCs are popular for ADC architectures with sample rates up to 100 MS/s and resolutions from 8 to 16 bits. These designs are both high power (mW-scale) and large ($0.06mm^2$ to $0.63mm^2$) [86, 87], and also require high accuracy references.

In this chapter we describe a self-timed $346\mu m^2$ ($0.000346mm^2$) voltage controlled oscillator (VCO) based sensor interface in 28nm low power (LP) CMOS. The topology is mostly digital and technology portable, and the extremely compact nature frees up silicon area for other system components. Dynamic resolution scaling requires only two additional standard cell FFs per additional bit of resolution, whereas previous designs [88] require exponentially larger capacitors. These capacitors do not scale well with technology and increase the area penalty in advanced process technologies. The proposed design uses no voltage or current references, and is self-timed, which further enables compact design.

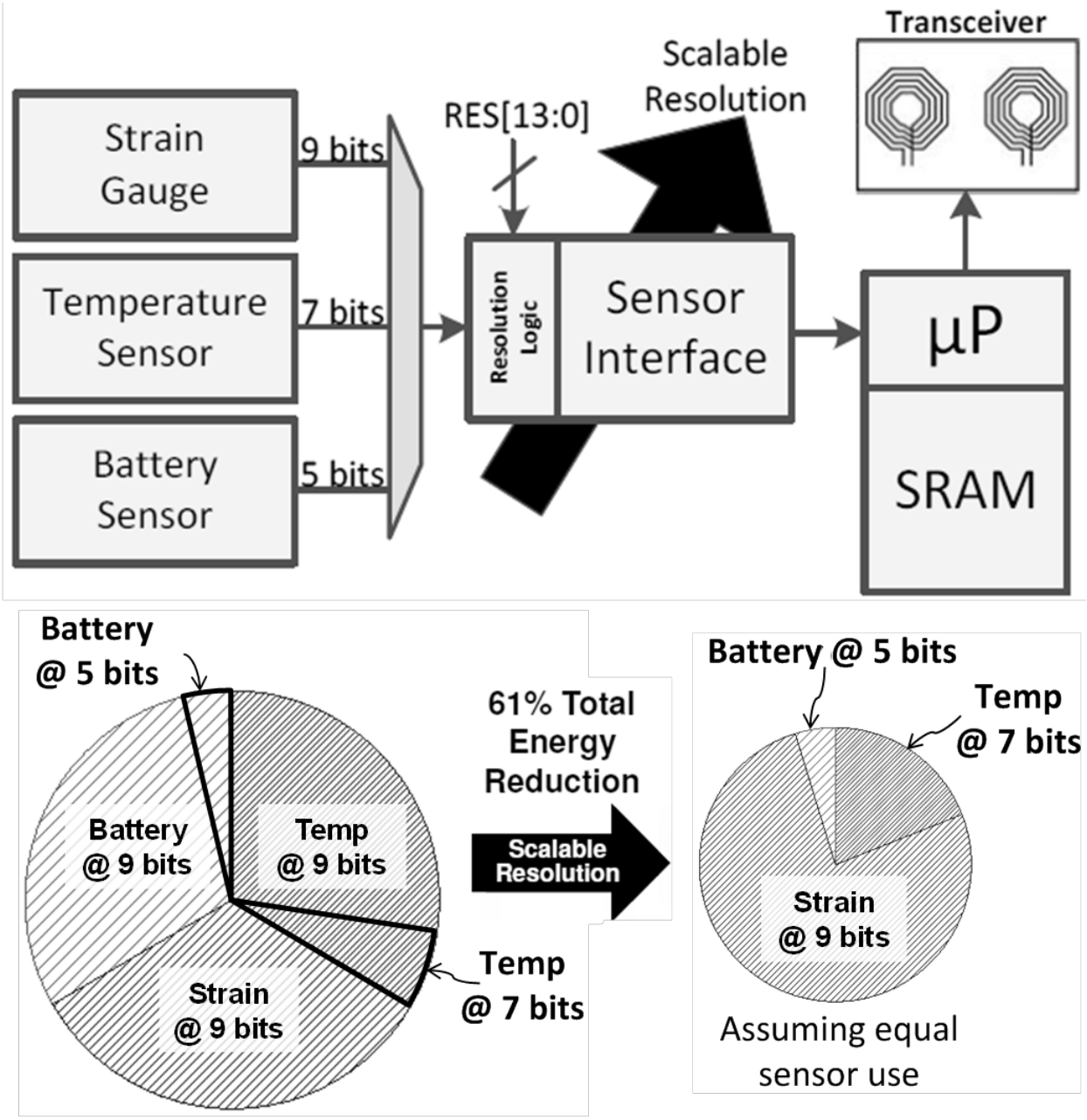


Figure 4.2: **Multiple Sensor Application.** Energy scalability can be applied to multi-sensor applications, adjusting resolution (and energy) based on application specific requirements.

4.2 Circuit Description

The proposed design is VCO based, where the analog input voltage is first converted to the frequency domain and then digitized through a time-to-digital converter. In order to facilitate the self-timed nature of the design, the analog input voltage is converted to two frequencies, and those frequencies are compared to each other to form the measurement. As shown in Figure 4.3, the circuit consists of two current starved VCOs, each followed by a pre-loadable ripple counter to count cycles, and a small finite state machine. The VCOs are designed to have opposing frequency responses - one with a positive response (PRVCO) where frequency increases with higher input voltage and one with a negative response (NRVCO) where frequency decreases with higher input voltage. The VCOs are designed such that the PRVCO and NRVCO have the same frequency at $V_{DD}/2$. These output frequencies are then fed into counters for digitization. Due to their opposing frequency responses, one of the two VCOs will be faster than the other for each given input voltage V_{IN} - the NRVCO will be faster when $V_{IN} < V_{DD}/2$ and the PRVCO will be faster when $V_{IN} > V_{DD}/2$. The measurement is self-timed and ends when the faster of the two counters causes an MSB transition from 0 to 1. The FSM detects this transition and stops the oscillation of both VCOs, which freezes the counter values. The value on the slower counter, as well as one-bit representing which counter finished first (PRVCO or NRVCO), are stored as the result of the measurement.

4.2.1 VCOs and Linearizing Amplifiers

The VCOs are implemented using 5-stage current-starved ring oscillators, with 4 current-starved inverters and one current-starved not-and (NAND) gate for control. The two oscillator frequencies become equal at the 'crossover point'. The current starving transistors and bias generators are sized such that this crossover point nominally occurs when $V_{IN} = V_{DD}/2$. This increases resolution by keeping the VCOs in a naturally linear range of operation, and increases linearity by equalizing the slopes of the two halves of the output code graph. The starving transistors, and the number of VCO stages can be optimized to achieve improved linearity, speed, or power consumption. Appropriately sizing the starving transistors and

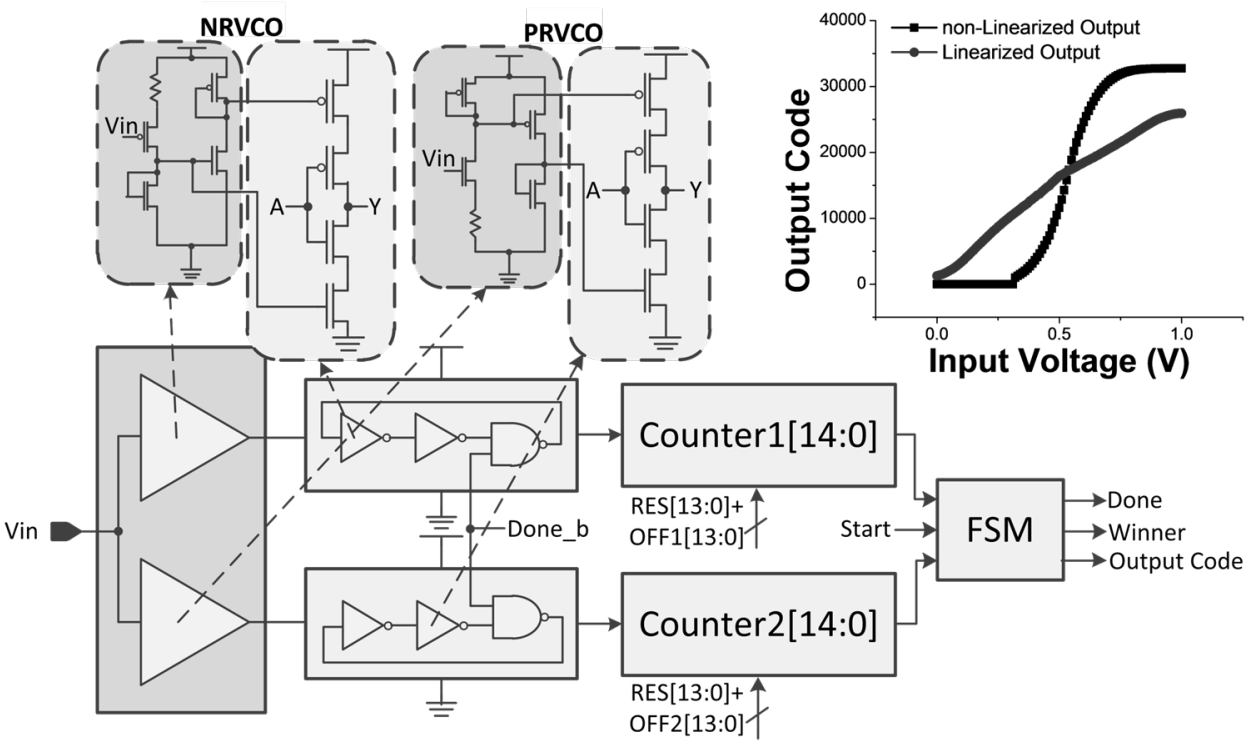


Figure 4.3: **Sensor interface block diagram and measured output codes.** The sensor output is taken in as the input voltage V and digitized into the above output code, which is linearized using simple degenerated amplifiers. The amplifiers increase the usable input voltage range from 0.4-0.6 V to 0.1-0.9 V. Output code measurements in the above graph are taken with an active counter bit setting of 14 bits.

in combination with the linearizing amplifiers can improve the linearity, and decreasing the number of VCO stages will increase the speed and energy of the system.

The opposing frequency responses are created in the bias generation through the use of degenerated common source amplifiers, as shown in Figure 4.4. These amplifiers also increase VCO linearities and greatly extend their input range (4). The PRVCO uses an NMOS-based common source amplifier and the NRVCO uses a PMOS-based common source amplifier. In both of these circuits (Figure 4.4) M1, M2, and the resistor comprise the common source amplifier, M3 mirrors the current to the output node and M4 acts as a load for M3. Increasing the sizes of both M2 and the degenerating resistor increase linearity at the cost of increased power consumption and area. To achieve a reduction in power, the linearizing resistor can be sized up, which also improves linearity, but greatly increases the area of the amplifiers. In simulation, when the linearizing resistor is sized up from 80kO to 160kO, the input

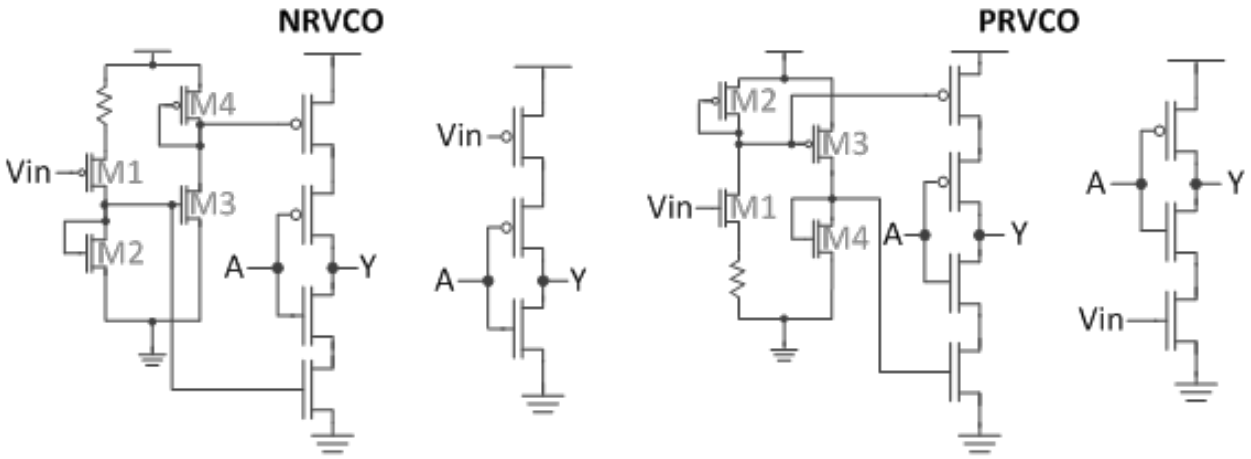


Figure 4.4: **Linearized vs. non-linearized voltage controlled oscillators.** Each stage of the linearized VCOs has PMOS and NMOS starving transistors with the linearizing amplifiers controlling the oscillator's frequency response. The non-linearized VCOs have either a PMOS header (NRVCO) or an NMOS footer (PRVCO).

range stays constant but the power reduces from 26.9μW (in one amplifier) to 15.2μW. This corresponds to a decrease in power by 0.56 for an area increase of 2.

These amplifiers do not require high accuracy current sources or bias voltages, and take only the sensor output voltage as an input. The goal of these amplifiers is to bias each oscillator in its highly linear range of operation for a wider range of input voltages, while simultaneously generating opposing frequency responses. Each amplifier provides two biases, one for VCO headers and one for VCO footers. Without linearizing amplifiers, the oscillators saturate and turn off in high and low voltage ranges, restricting the usable input voltage range to between 0.4V and 0.6V (for 1V V_{DD}), instead of the full 0.1 - 0.9V range of the linearized design. Full-range input voltage can also be used, but results in decreased linearity.

For high-accuracy measurements, linearity can be further corrected through post-processing of the data with a full table, which is common in wireless sensor node applications [89]. This correction is done off-chip after data collection from the sensor nodes is complete. In low-resolution applications, such as wake-up monitors, post-processing is not needed. Without the use of a look-up table, the output code response (slope) varies with input voltage. Measurements of the system with 14 active counter bits at 1V V_{DD} show a slope of 17540 codes/V in the range of 0.1V to 0.3V input voltage, 15368 codes/V from 0.3V to 0.5V, 10302 codes/V from 0.5V to 0.7V, and 11415 codes/V from 0.7V to 0.9V. This difference in slope affects the

overall linearity of the system and necessitates the use of a full, off-chip, look-up table for high accuracy measurements. The measurements in this design used a full look-up table via an off-chip computer. Smaller look-up tables can be implemented on-chip, eliminating the need for off-chip correction, but would result in an increase in linearity errors. The off-chip look-up table consists of a high resolution, full-range, input voltage to output code mapping. The smaller, on-chip, look-up table would contain a more sparse representation of the same mapping, and the output value can be calculated using linear interpolation of a subset of points in the table via software. The look-up table does not need to scale with resolution, as a larger table will always work well for smaller resolution values. If it is necessary to scale the number of entries in the look-up table, then for each fewer bit of resolution half of the entries could be eliminated (every other entry).

4.2.2 Non-linearized VCOs

Another design approach would directly use V_{IN} to starve the oscillators, which avoids the need for linearizing amplifiers to generate bias voltages. As shown in Figure 4.4, this could be accomplished by using just an NMOS footer for the PRVCO (i.e., no header) and just a PMOS header for the NRVCO (i.e., no footer). The circuit would have the same functionality with decreased resolution, linearity, and operating range, but also decreased power consumption and area. Crossover point sizing and calibration in the non-linearized VCOs is similar to the linearized version but does not require sizing between the linearizing amplifiers and the header/footer combinations. Instead only the NMOS footer and PMOS header need to be sized to achieve equal frequencies at the crossover point.

Some wireless sensor node applications could benefit from this type of functionality. Battery monitors typically operate in a narrow range of voltages, where users may prefer high accuracy in a small voltage range and very low accuracy outside of that region. This can save both power and area, as the linearizing amplifiers consume nearly half of the total power.

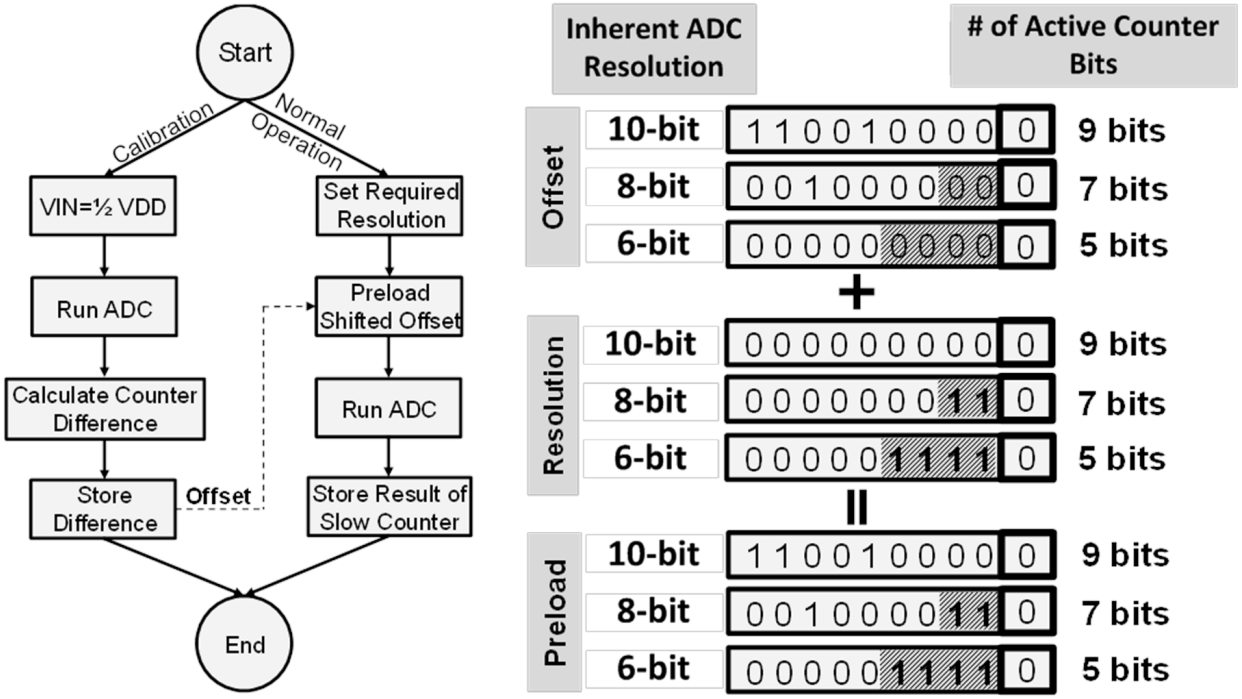


Figure 4.5: **Calibration flow and preload calculation example.** (Left) Single point calibration flow chart. (Right) Pre-load calculation combining the active counter bit setting and calibration offset code.

4.2.3 Crossover Point Calibration

The crossover point can be single-step calibrated to correct for VCO variation or sizing mismatch between the starving transistors. Figure 4.5 shows the calibration algorithm. First, V_{IN} is set to $V_{DD}/2$ and the converter is run until completion (one of the two counters has an MSB transition and both oscillators are stopped and the counters frozen). At this voltage, the oscillator frequencies should be equal and therefore the counter values should be equal. If there is any mismatch between the two devices, their frequencies will differ with a corresponding difference in counter values. The difference remaining between the two counters at the end of the measurement is loaded back onto the slower counter as its offset in each subsequent conversion. This effectively speeds up the slower counter enough to equalize the frequencies at $V_{DD}/2$, and the pre-calibration and post-calibration shift in output response is shown in Figure 4.6.

The calibration value needs to be measured only once, at the maximum resolution and

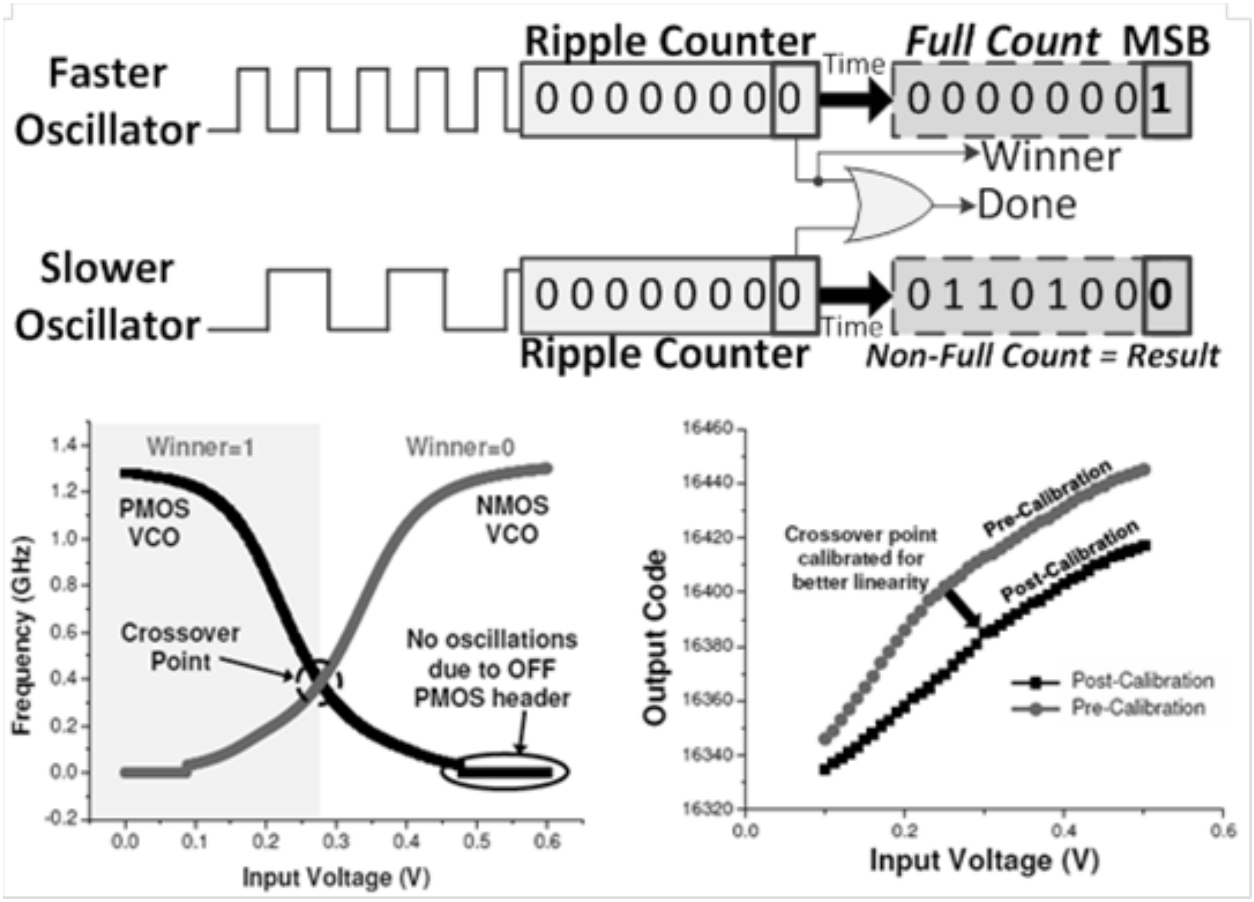


Figure 4.6: **Sample FSM operation and crossover point calibration.** (Top) FSM to monitor MSB transitioning. (Bottom Left) PSV and NSV simulated frequencies vs. input voltage. (Bottom Right) Measured output code vs. input voltage and crossover point calibration for a 6 bit active counter-bit setting.

mid-range power supply, and can then be shifted for lower resolutions since it does not significantly vary with V_{DD} or temperature, as shown in Section 4.3. The reference voltage of $V_{DD}/2$ can be provided directly from the power supply through a small reference circuit, or could be provided off-chip in a one-time testing scheme, since accuracy is not critical. Any errors in this reference value will translate into offsets in the crossover point, which may slightly decrease resolution. This calibration method allows for full range offset compensation.

4.2.4 Dynamic Resolution Scaling

Dynamically scalable resolution is achieved through pre-loading the upper bits of the ripple counters. Resolution is defined by the frequencies of the two VCOs, where the faster oscillator sets the conversion time, and the slower oscillator sets how fast the counter runs during the time period, resulting in a ratio:

$OutputCode \propto f_{slow}/f_{fast}$.

The resolution is limited by how finely fslow and ffast can be measured, but this can be controlled by counter length, which extends the time of the measurement. Thus, maximum achievable resolution is directly controlled by the maximum counter value. Since there are 2 counters in this design, if each counter has 10 active counter bits plus 1 bit of MSB, the maximum achievable resolution, which corresponds to the resolution of the sensor interface, is 11 bits (where one bit of information is obtained by recording which counter was faster). By reducing the number of active counter bits, the maximum conversion time, maximum resolution, and energy all reduce as well. The maximum counter value can be reduced by pre-loading a '1' into the higher order bits of the counters, thereby deactivating them as shown in Figure 4.5. Resolution is impacted as in the following equation:

$$OutputCode = (f_{slow}/f_{fast})maxCounterValue$$

Resolution scales linearly between 2.8 and 11.7 bits of effective resolution with 11 pre-loadable counter bits, with each additional-bit of resolution adding two flip-flops (one to each counter). Low resolution applications can achieve extremely small footprints by limiting the number of counterbits. The output code calculated by the FSM is based on the following logic:

IF (NRVCO is slower) THEN

OUTPUT CODE = NRVCO counter value

IF (PRVCO is slower) THEN

OUTPUT CODE = (2maxCounterValue) - (PRVCO counter value)

This output code is affected by the active counter bit setting. To decrease the overall system resolution, counter bits are deactivated by pre-loading a '1' into higher order bits. This reduces the amount of conversion time but also sets minimum and maximum bounds

on the output codes. When the system is at full resolution, the output codes can range from zero to 2 the maximum value stored on one counter. With resolution scalability, the output codes are offset by the following amount:

$$OutputCodeOffset = \Sigma_{(i=N)}^{(maxCounterBit-1)} 2^i$$

Where N is the active counter bit setting and maxCounterBit is the number of available counter bits in the design, excluding the MSB. This sets a minimum bound for the output code, restricting it to the number of pre-loaded counter bits. Thus, with resolution scaling, the output code always decreases from both sides of the graph toward the crossover point, keeping the midrange output code value constant across all active counter bit settings.

4.3 Measurement Results

This chip was fabricated in a 28nm LP CMOS process and included four different variations of the proposed interface for testing purposes. The four different variations were: 3-, 5-, and 7-stage ring oscillators using low threshold voltage (LVT) starving transistors, and a 5-stage ring oscillator with super-low threshold voltage (SLVT) starving transistors. Each sensor interface included 14-bit counters to measure maximum resolution and resolution scalability.

The length of the VCO determines the speed, power, energy, and noise characteristics with which the interface can operate. For this implementation, a shorter VCO was chosen to increase the speed (decrease the energy consumption), setting the frequency range to about 1.3GHz in simulation. Longer VCOs would give enhanced noise characteristics as a result of increased averaging over stages, but would greatly decrease the overall speed and increase energy.

4.3.1 Resolution, Energy, and Power with Voltage Scaling

Effective resolution, measured as $log_2(V_{inputRange}/(RMS_{Noise}))$, scales linearly from 2.8 to a noise-limited 11.7 bits over a power supply range of 0.6V-1.0V (Figure 4.7). Noise was measured by fixing the input voltage and measuring the standard deviation of output codes over 100 separate conversions. Due to the inherent non-linearity of the interface, the noise

was measured at 3 different input voltages: at the lowest V_{IN}, highest V_{IN}, and at $V_{DD}/2$, though only a weak dependence on input voltage was observed during noise measurements. At 0.5V V_{DD} (representing near-threshold operation in this LP process [90]), the noise-limited maximum resolution decreases to 9.2 bits. Each additional active counter-bit contributes nearly 1 additional bit of effective resolution. The linear relationship between counter-bit setting and measured resolution allows for simple resolution control logic, which is beneficial in targeted ultra-low power applications.

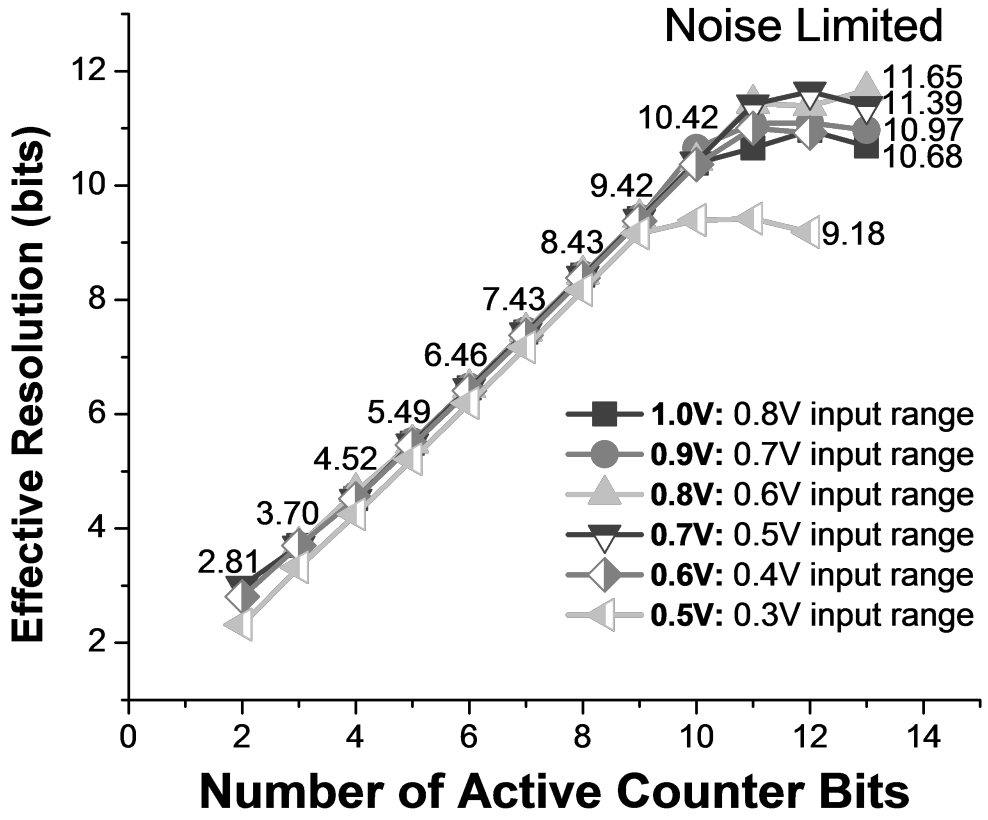


Figure 4.7: **Effective resolution versus the number of active counter bits.** Resolution scales linearly with the number of active counter bits and is not impacted by voltage scaling until $V_{DD} = 0.5$ V (near-threshold).

Average power scales from 105μW to 11.7μW as V_{DD} is reduced from 1.0V to 0.6V (Figure 4.8), tapering off at 5.2μW at 0.5V as the constant current draw of the linearizing circuit begins to dominate. Conversion time scales exponentially with the number of active

counter bits (Figure 4.9), from 36ns to 9.3μs (2.8 to 11.0 bits) at 0.6V, and also scales with input voltage V_{IN}. At very low resolution, the conversion time saturates as the FSM delay becomes limiting. These conversion times are acceptable for sensors where values change on the scale of milliseconds (e.g., temperature), and are comparable to low voltage (0.8V) ADCs presented in [82,87]. Energy per measurement also scales exponentially with resolution, from 0.6pJ to 217pJ at 0.6V. Thus, reducing the number of active counter bits is highly beneficial to system energy, decreasing it by up to 1000.

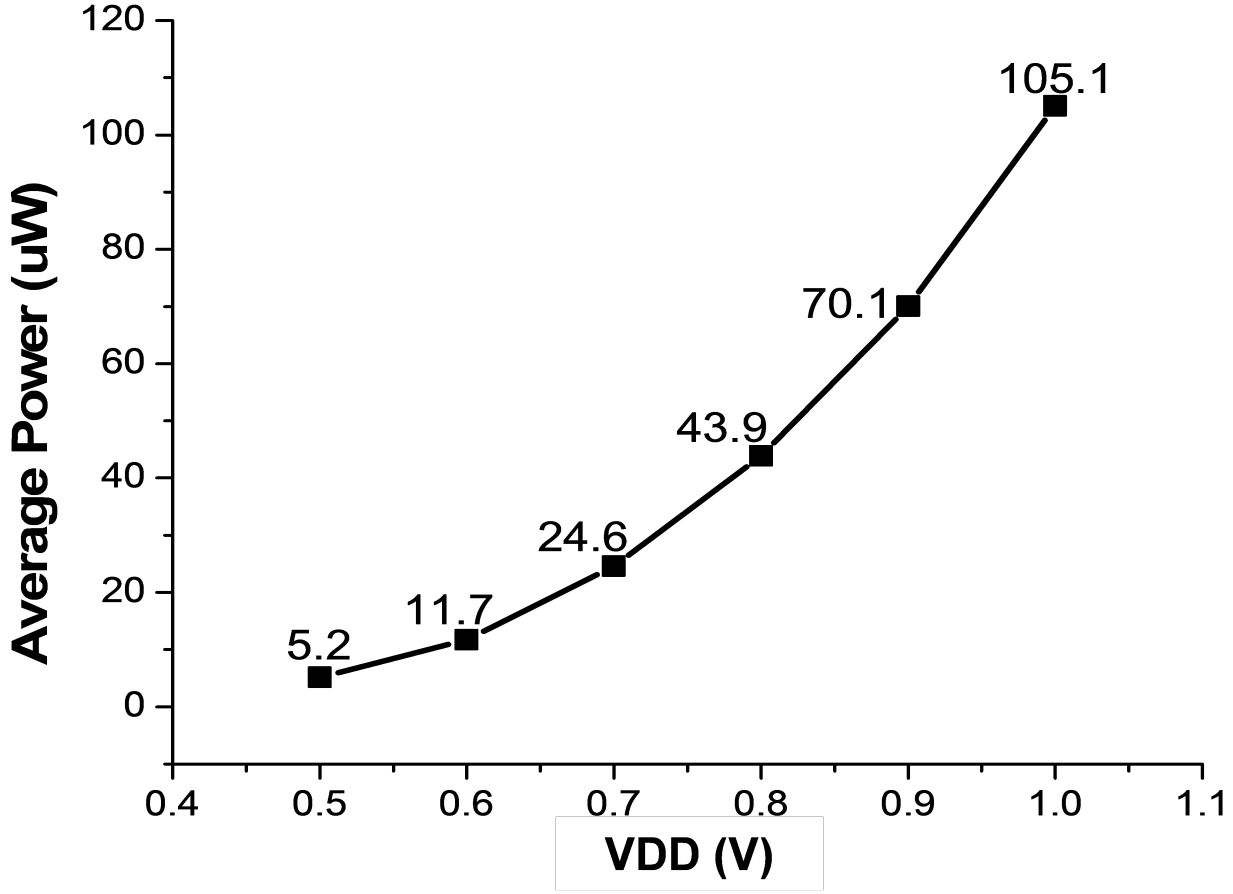


Figure 4.8: **Average power versus power supply.** Average power consumption over scaled V_{DD} from 0.5 V to 1.0 V.

Figure 4.10 shows an energy consumption breakdown for the 7 active counter-bit setting (7.4-bit effective resolution). Static energy from the linearization circuits starts increasing below 0.7V while minimum energy per measurement is 27pJ, which is achieved at 0.6V. Energy per conversion step at a fixed V_{DD} (Figure 4.11) remains relatively constant over

a range of 5-10 active counter bits, increasing at higher resolutions due to system noise limitations and at lower resolutions due to the saturation of both power and conversion time. The minimum achieved value for energy per conversion step at a setting of 9.4-bits of effective resolution is 41.2fJ/step at 0.6V with a maximum of 80.4fJ/step at 1V V_{DD}. Energy per conversion step is constant between 5 active counter bits and 10 active counter bits due to the exponential scaling of energy and linear scaling of effective resolution.

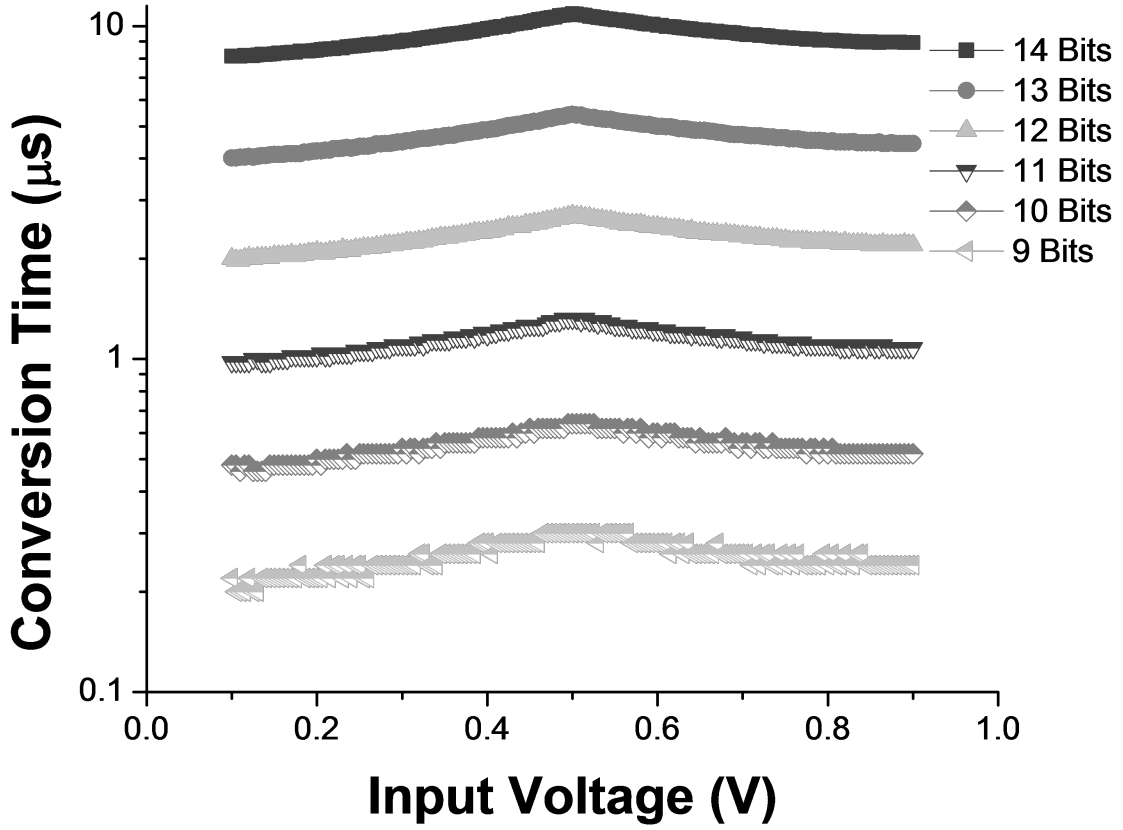


Figure 4.9: **Conversion time.** Measured conversion time for each input voltage and for active counter-bit settings from 9 to 14 bits.

4.3.2 Crossover Point Calibration

Figure 4.12 shows measured and calculated offset values versus voltage and active counter bit settings. The calculated values are shown as dotted black lines, where the offset is

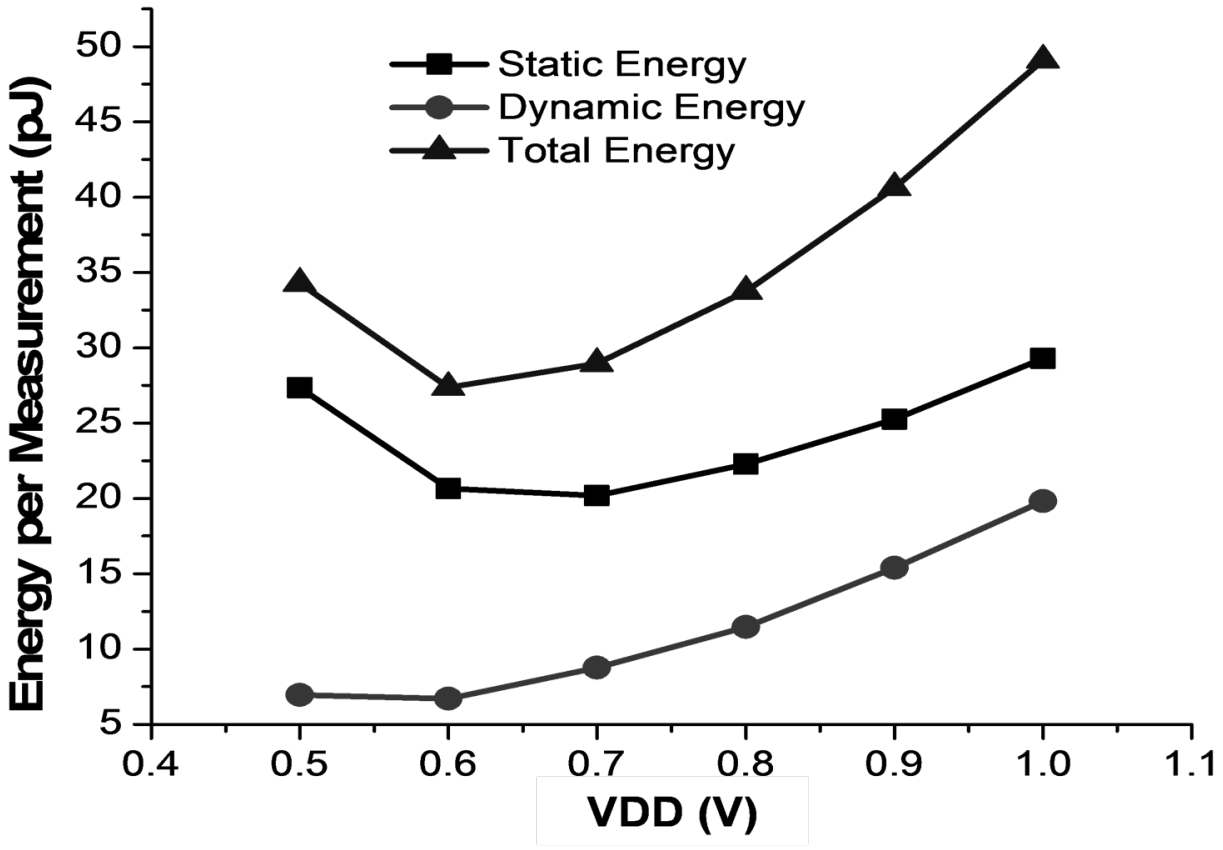


Figure 4.10: **Energy breakdown over power supply.** Static and dynamic energy breakdown over scaled V_{DD} for 9.4 bit effective resolution operation.

constant across power supply and scales by a factor of two for each additional active counter bit. At a measured active counter bit setting of 10 bits, the calibration offset average at 0.8V is 209 counts. The maximum observed error between measurement and calculation due to V_{DD} scaling is 16 counts, which corresponds to a shift of 8mV in input voltage. Calibration measurements show that this estimation is accurate for each active counter bit setting, and that V_{DD} offset error reduces with decreasing resolution, resulting in a maximum error of only 1 count (4mV input voltage) at the 7b active counter bit setting.

4.3.3 Noise and Calibration Error over Temperature

Sweeping temperature from $0°C$ to $80°C$ shows that the RMS noise of the system increases with temperature with a maximum increase of 0.18mV at 0.8V V_{DD}, which corresponds to a resolution degradation of 0.46 bits (Figure 4.13). This degradation affects the maximum

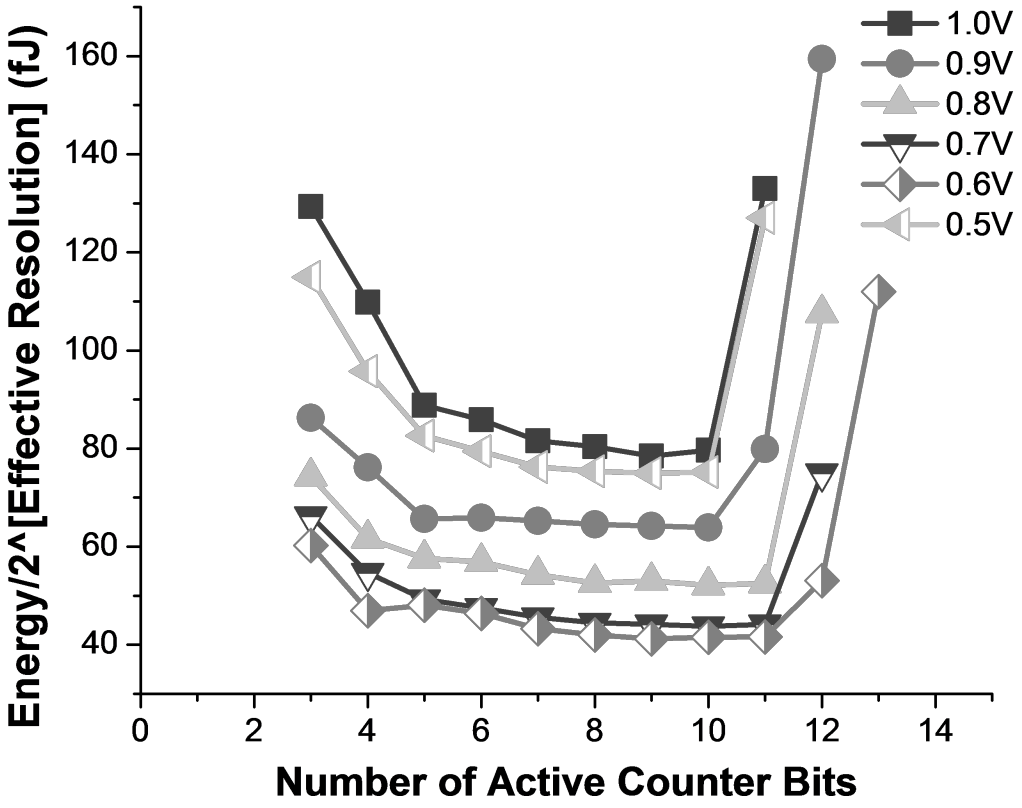


Figure 4.11: **Energy per conversion step (fJ/conv-step).** Energy per bit of resolution for each preset resolution value and each V_{DD} setting.

achievable resolution while all resolution values that are not noise limited remain unchanged. Calibration offset codes were also measured across temperature and power supply to determine the maximum error expected from a one-point calibration scheme (Figure 4.14). A maximum error of 34 counts (22mV input voltage shift) occurs at $0°C$ and 0.7V V_{DD}.

The results in Figure 4.15 show that for this implementation, calibration offset codes decrease monotonically with temperature. This is because the PRVCO is faster than the NRVCO for all points in the graph, due to either sizing or threshold voltage mismatch. To compensate, the slower counter (NRVCO) is pre-loaded with calibration offset codes - effectively speeding up the NRVCO by giving it a head start. When temperature increases, NMOS drain current decreases more rapidly than PMOS drain current. The linearizing

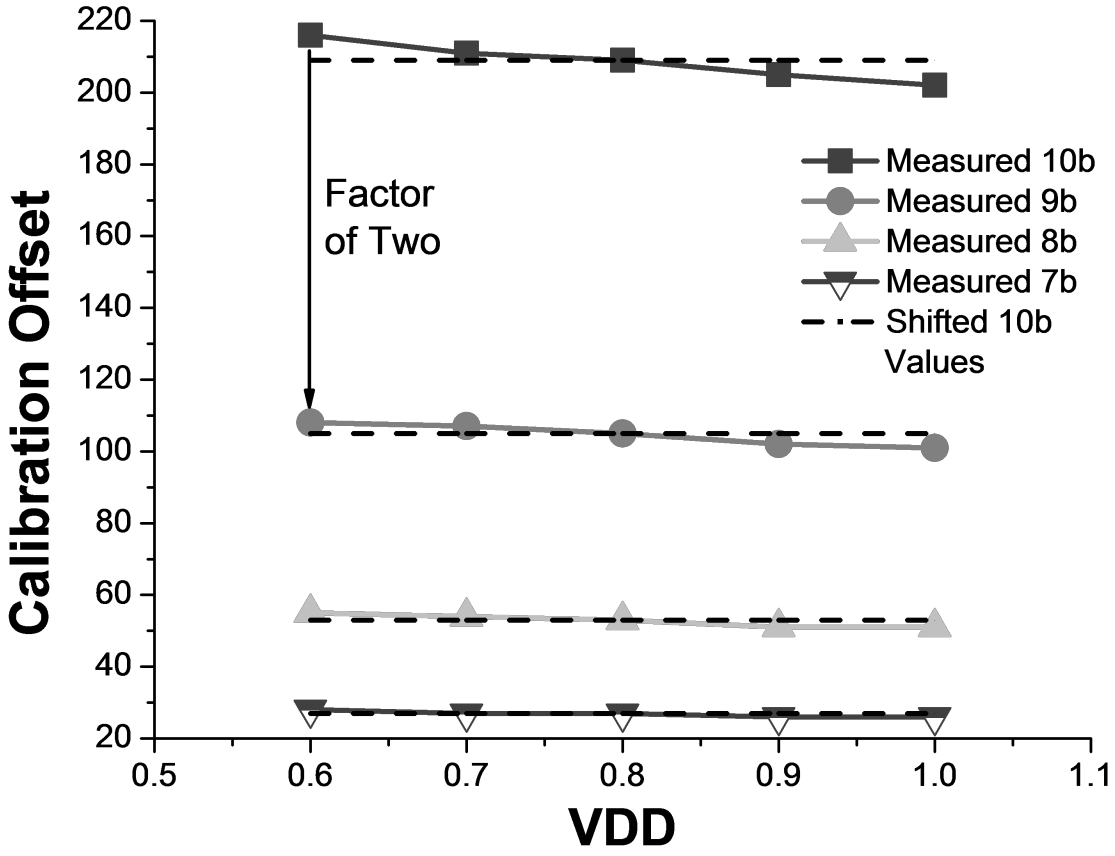


Figure 4.12: **Calibration offset code versus power supply.** Measured calibration offset code from stored and values scaled by the number of active counter bits.

amplifier in the NRVCO is controlled by a PMOS input transistor and the amplifier in the PRVCO is controlled by an NMOS. Due to this difference in temperature dependence, the PRVCO slows down more rapidly than the NRVCO, causing the oscillator frequencies to approach each other at higher temperatures, thus reducing the required calibration offset code monotonically.

Since the calibration offset codes decrease monotonically with temperature, the single point calibration measurement should be taken at a midrange temperature ($40°C$ for these measurements) to minimize error.

Temperature variation affects the output code range of the ADC. With increasing temperature the minimum output code value increases and the maximum output code value

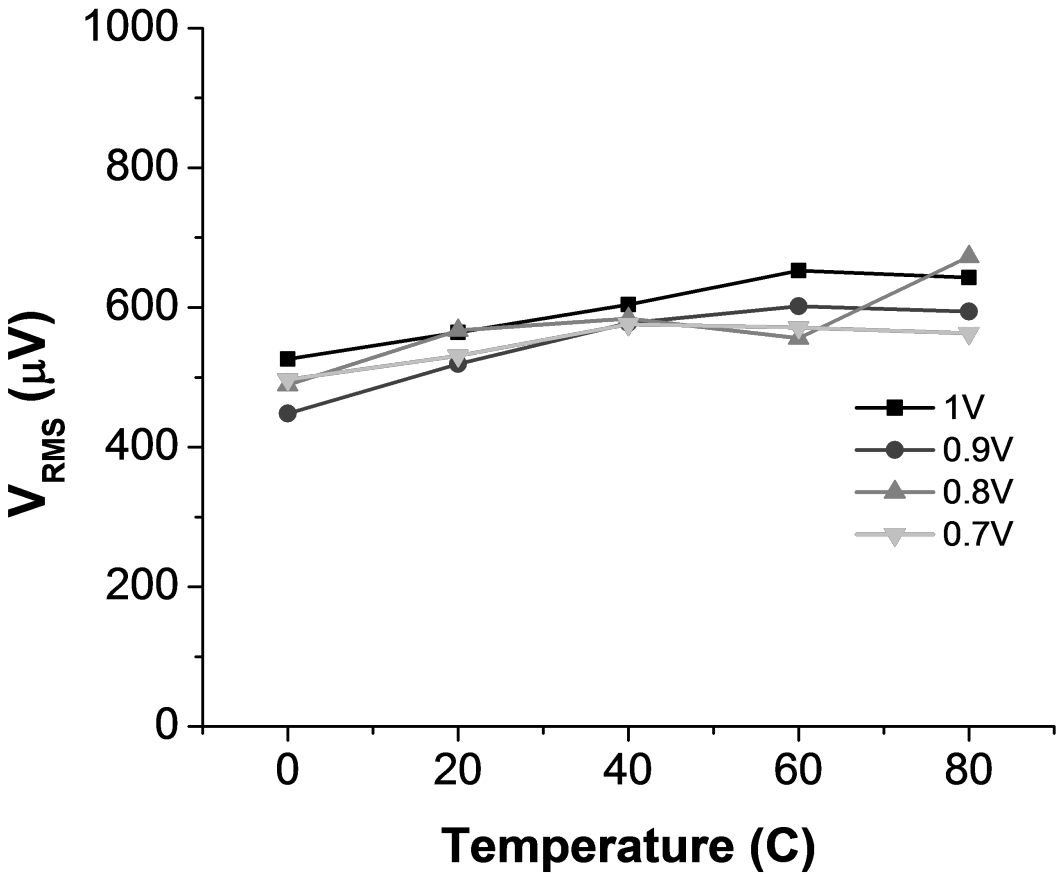


Figure 4.13: **RMS noise versus temperature.** RMS noise increases with temperature up to a maximum of 673μV at 0.8V V_{DD}, which corresponds to a resolution of 10.22 bits

decreases, decreasing the overall range. At 0°C, 1V V_{DD}, and an active counter bit setting of 7 bits, the output code range is 204 levels. When the temperature increases to 60°C, the output code range decreases to 177 levels. This change in output response affects the accuracy of the off-chip, full look-up table. Using one look-up table (nominal temperature measurements) over all temperatures would result in a maximum error of about 0.08V (input voltage), or 0.1 bits of resolution. However, multiple look-up tables could be used to reduce this error. Calibration measurements using the given oscillators provide an accurate estimation of temperature (Figure 4.14) if averaged over several measurements to account for power supply variation. These temperature estimations can be used to select an appropriate off-chip look-up table for use in subsequent measurements. With a look-up table for each

$20°C$ temperature step, the maximum error due to temperature variation can be reduced to 0.02V (input voltage), or 0.04 bits.

4.3.4 Output Response and Linearity

Figure 4.15 shows the measured output code versus input voltage of four different variations of the proposed sensor interface. The output response shows very little change with either threshold voltage or ring oscillator length. All of the variations showed similar responses with power supply scaling. Process and temperature analysis were not analyzed for the different interface variations.

At 1V, the average INL and DNL in the seven active counter bit setting (7.4-bit effective resolution) are +0.12/-0.52 LSB and +0.22/-0.17 LSB, respectively, after digital correction through a full look-up table, as shown in Figure 4.16. Figure 4.17 shows the pre- and post-transformation output codes. This table would likely be contained off-chip in most applications [71].

4.3.5 Non-Linearized VCOs

Included and measured on the test die was a non-linearized variant of the design. As described in Section 4.2.2, this version of the system connects V_{IN} directly to the headers and footers of the VCOs in order to decrease the sensor interface power. As seen in Figure 4.18, between 55.4% and 68.2% of overall power (at 1V and 0.6V V_{DD}, respectively) can be reduced, making the minimum power draw 3.7μW (0.6V V_{DD}). This reduces the input range from 0.1-0.9V to 0.4-0.6V (a 4× difference), for a 1.0V supply.

4.3.6 Area and Low Resolution Modifications

The proposed sensor interface core area occupies $346\mu m^2$, and the die micrograph is shown in Figure 4.19. This area does not include a register used to store the offset value for crossover calibration. An additional 8 registers to store the offset value would add roughly $50\mu m^2$ in additional area. Compared to more traditional implementations, this area is 100× smaller than a resolution scalable (up to 10.5 bits) SAR [88] (ideally scaled from 65nm) and

10× smaller than an 8-bit 28nm sub-ranged SAR [91]. For testing purposes, the proposed circuit was implemented with 15 counter bits (14 active bits and the MSB, used to determine which counter finished first). The maximum resolution achieved was 11.7 bits. In a typical implementation, there would be four fewer counter bits (eight fewer flip-flops), reducing overall area by $70\mu m^2$. In low resolution applications, area reduction can be even more aggressive, e.g., reducing the number of counter bits to six yields an overall area of $170\mu m^2$ (2× smaller). This implementation provides 5.5 bits of maximum effective resolution.

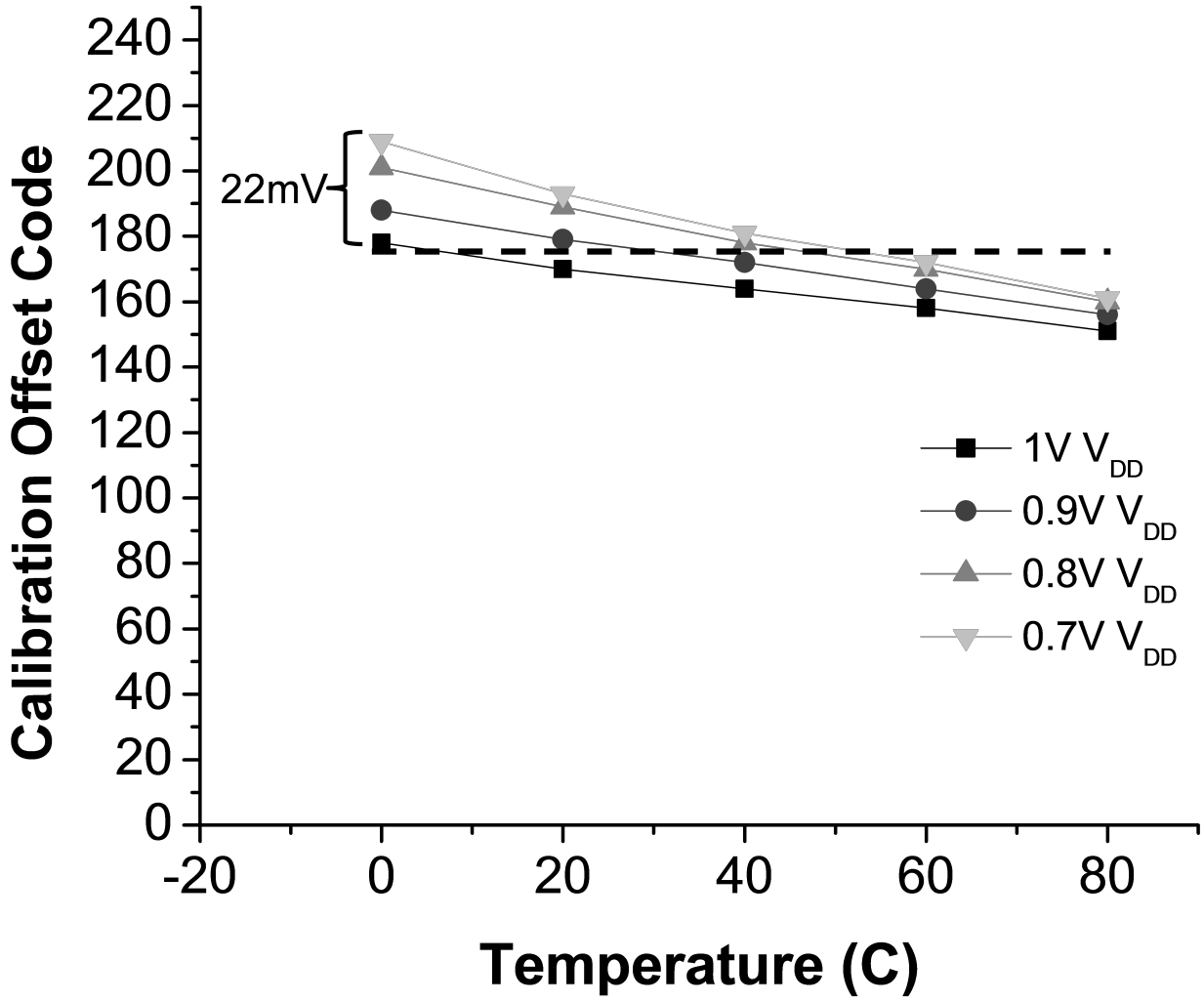


Figure 4.14: **Calibration offset code measurement versus temperature.** Measured calibration offset code decreases with increasing temperature. Maximum error of 34 counts at 0.7V V_{DD} (22 mV V_{IN} shift) between the stored single point calibration code (horizontal dashed line) and the measured codes.

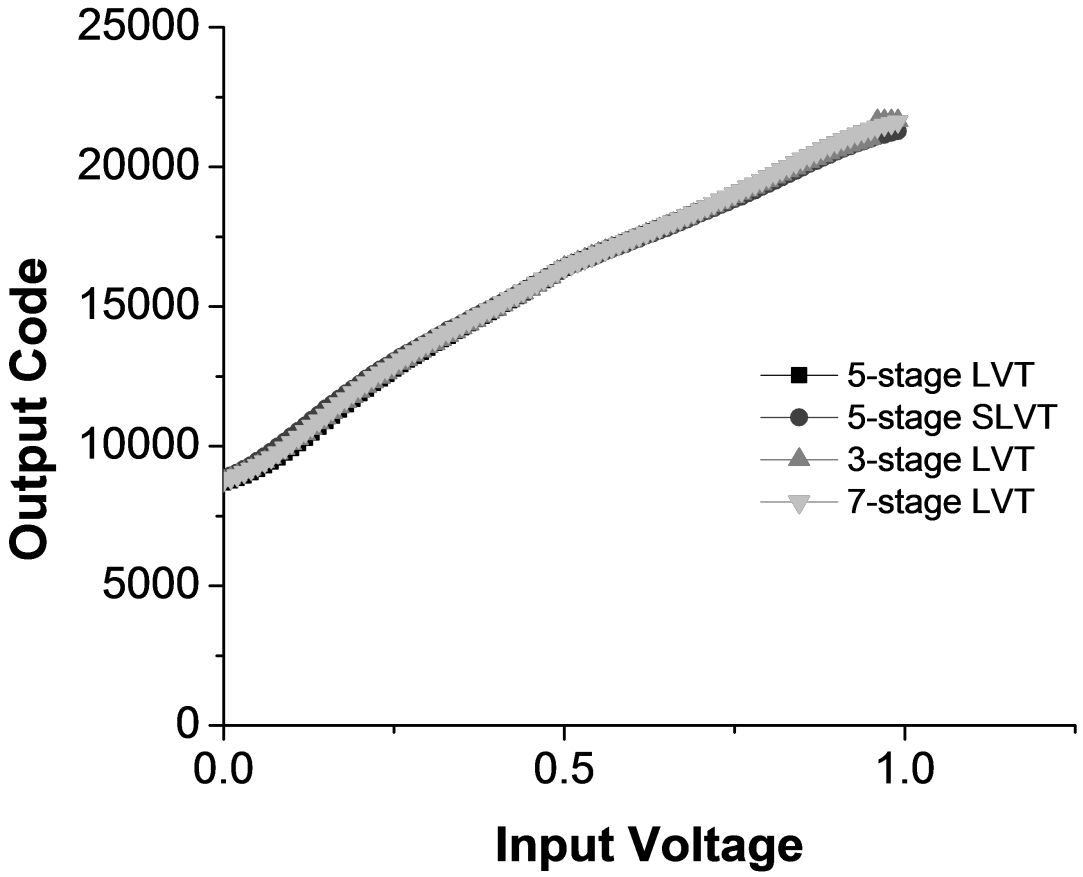


Figure 4.15: **Output code versus input voltage of four sensor interface variants.** Four different variations of the proposed interface were implemented, varying threshold voltage (LVT/SLVT) and ring oscillator length (3, 5, or 7 inverters), each measured at a 13 bit active counter-bit setting in this graph. The output response and resolution do not significantly vary with these modifications.

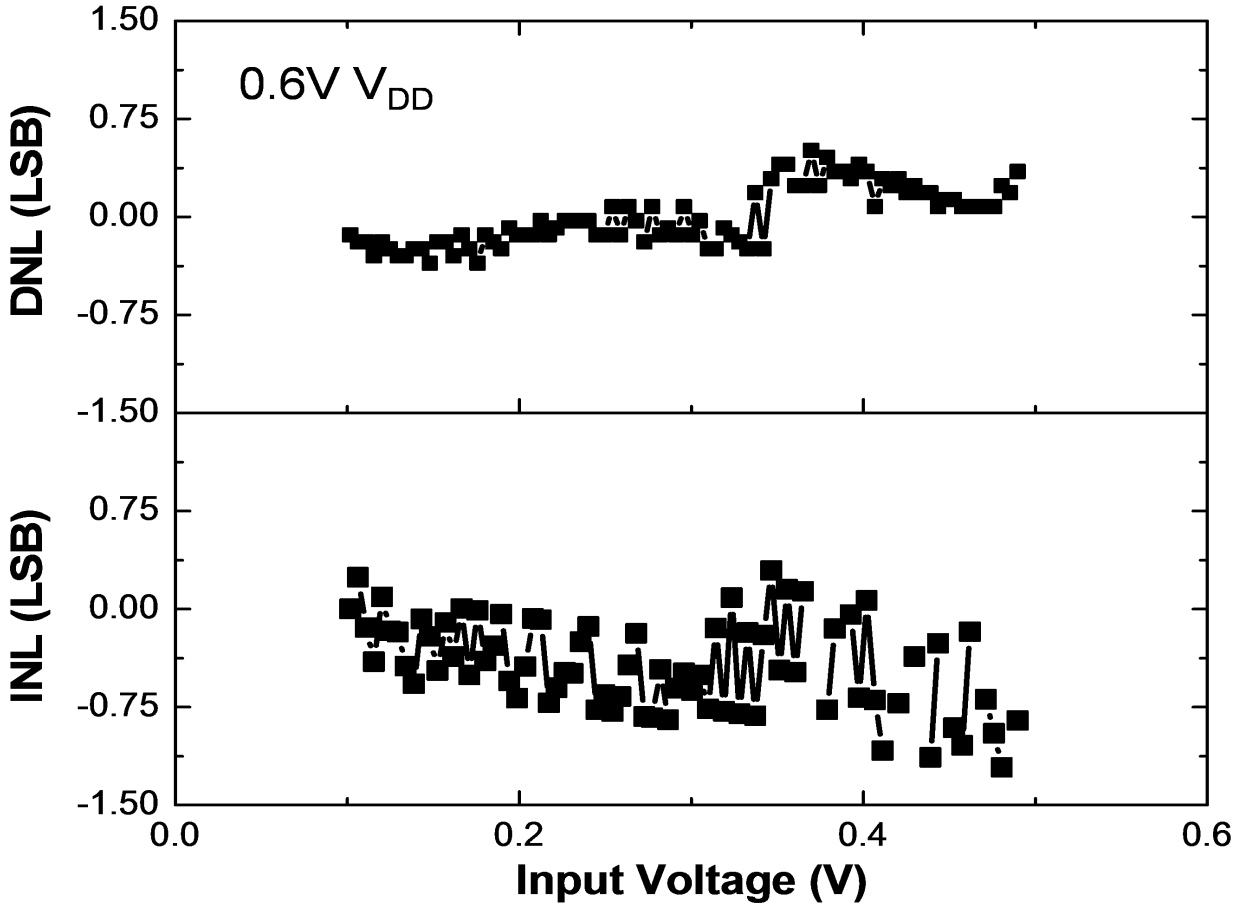


Figure 4.16: **Post-correction DNL, INL.** Post-correction DNL and INL for 7 bit active counter bit setting with a look-up table (0.6V V_{DD}).

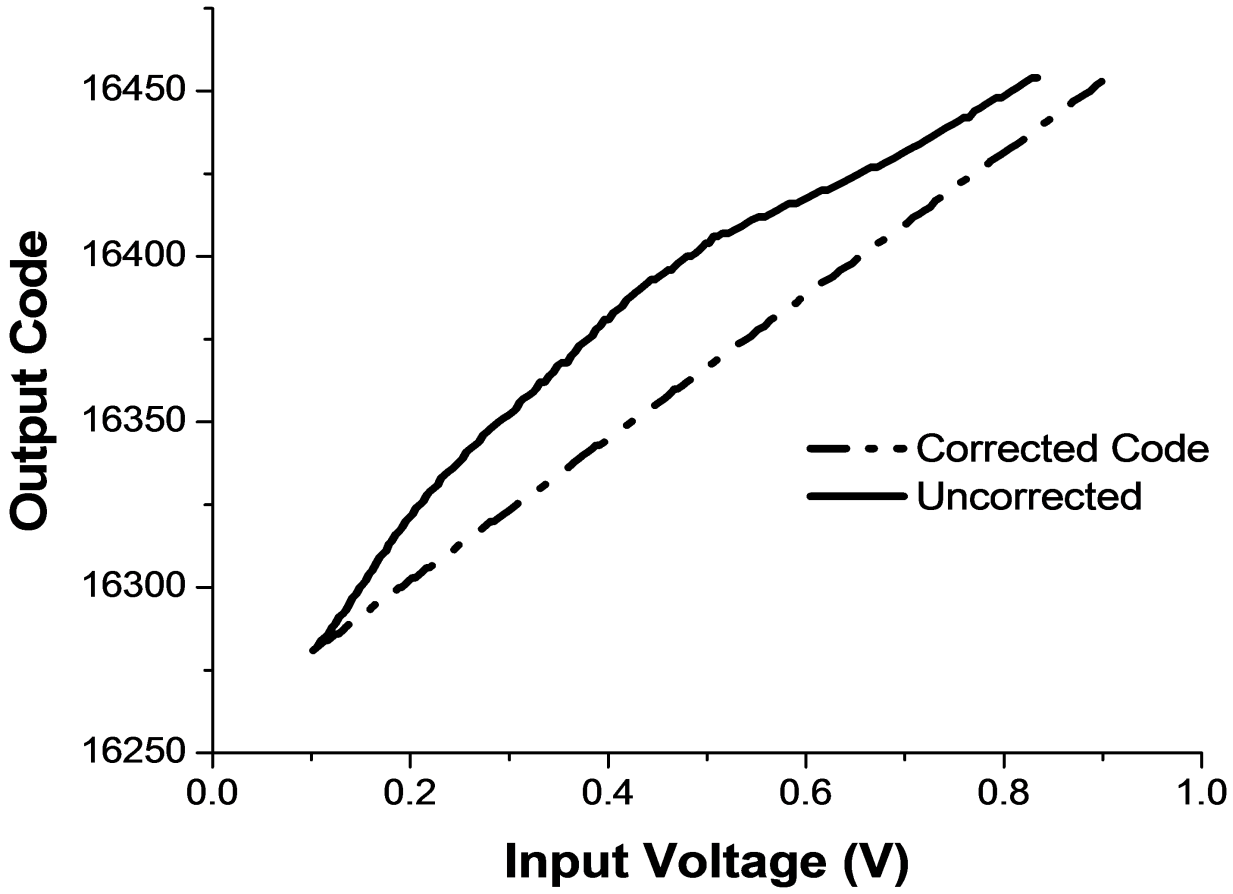


Figure 4.17: **Post-correction output code response.** Uncorrected and corrected (look-up table) output codes versus input voltage for an active counter bit setting of 7 bits. Post-correction is performed after data collection from the sensor nodes.

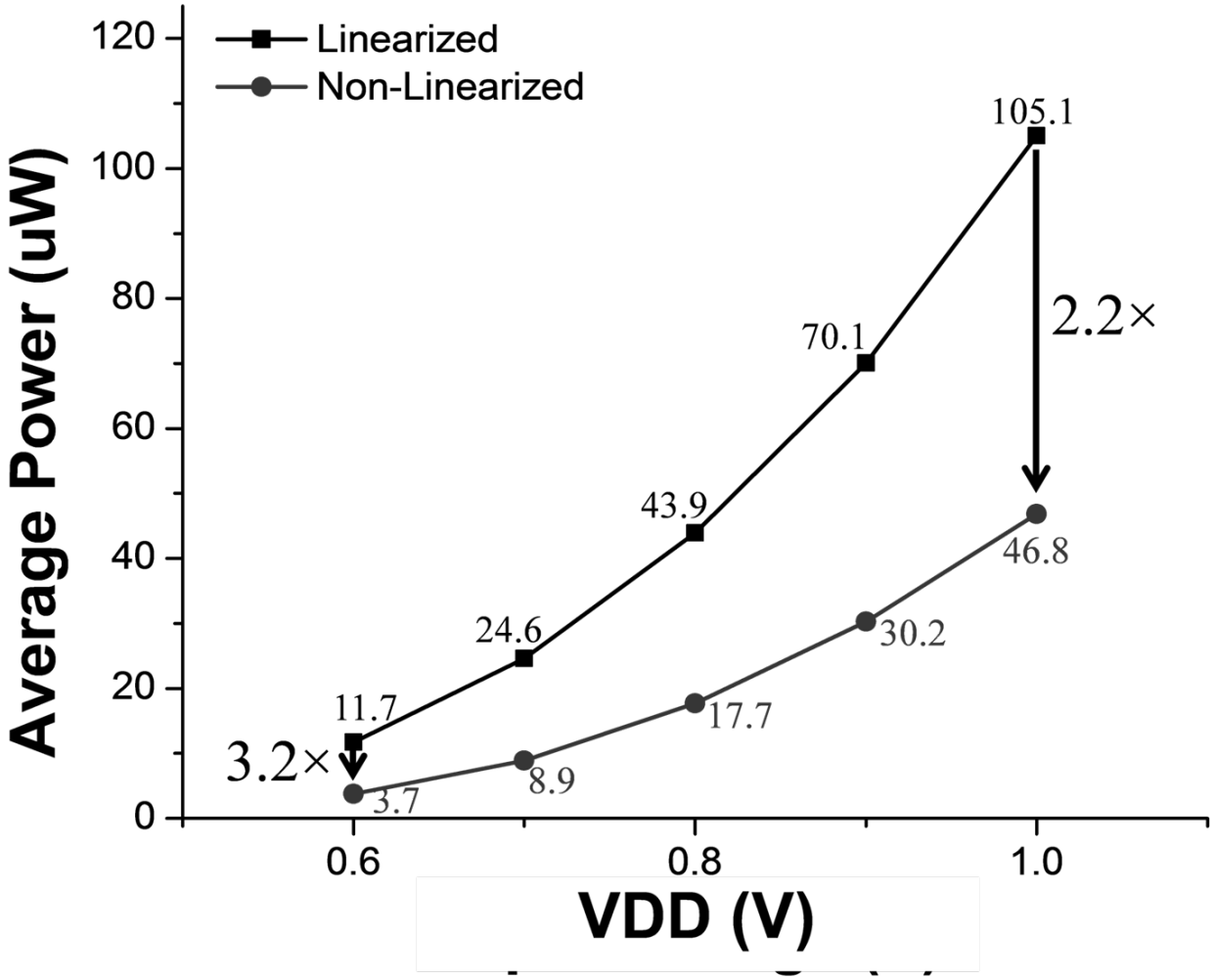


Figure 4.18: **Average power of the linearized and non-linearized sensor interfaces versus power supply voltage.** A small input voltage range (0.2 V) is acceptable for certain applications (e.g., battery monitors), reducing the average power consumption by 2.2× at full V_{DD} and 3.2× at 0.6V V_{DD} by eliminating the need for the linearizing amplifiers.

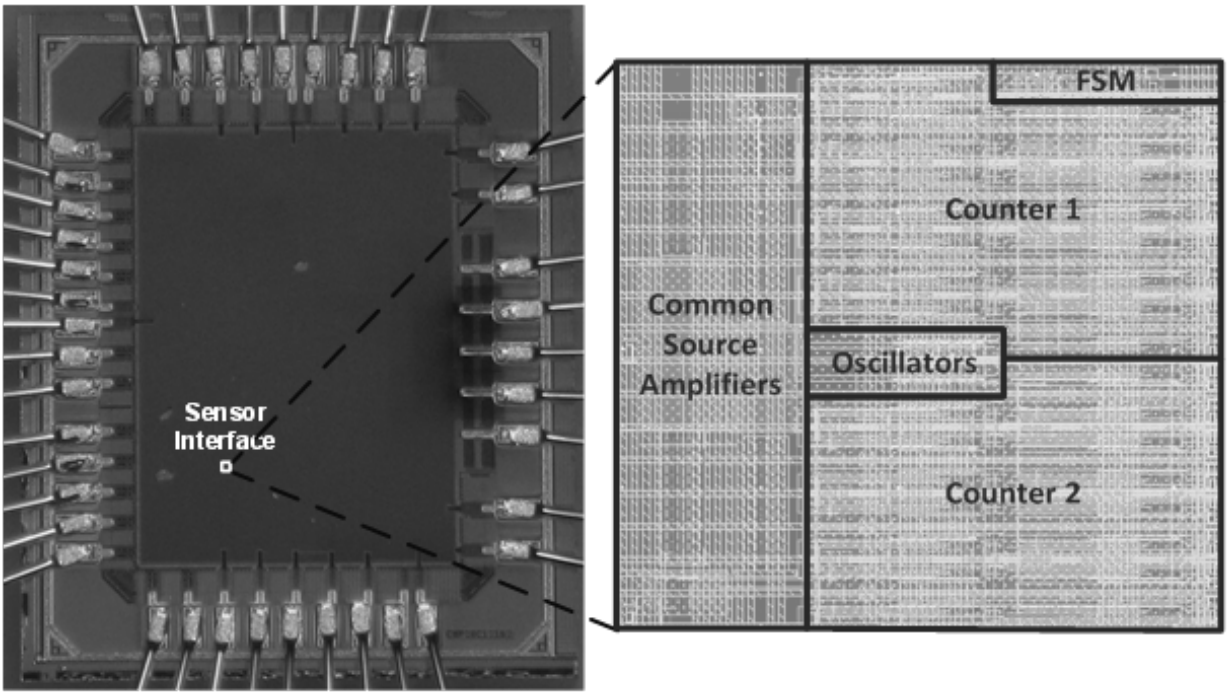


Figure 4.19: **Die microphotograph and chip layout.** Fabricated in a 28nm LP CMOS process and occupies $346\mu m^2$.

4.4 Comparison With Prior Work

Table 4.1 compares the proposed linearized design and three closely related, published ADCs with salient features listed. The first comparison point [88] is a resolution scalable SAR ADC implemented in 65nm CMOS. This paper leverages a reconfigurable DAC to target sensor networks and medical monitoring for energy-constrained systems. The application space, voltage scalability, and resolution re-configurability make this paper a pertinent comparison point to the proposed design. Compared to this work, resolution scalability for our design is increased by 4-bits (2.8 to 11.7-bits ER versus 4.77 to 8.84-bits ENOB), conversion time at low voltage is improved by 50× and area (assuming ideal scaling from 65nm to 28nm) is decreased by 100×. While power consumption is higher for our design (11.7μW to 0.2μW), that does take into account the high accuracy timing reference and high accuracy voltage references needed, which can be on the order of milliwatts [76].

The second comparison point [75] is a wireless strain sensing microsystem that incorporates two ADCs for multi-sensor readout. This system provides an important comparison between a resolution scalable implementation for multi-sensor wireless nodes, and an implementation that leverages multiple interface circuits to increase readout efficiency. Area is significantly increased in this design in order to facilitate two ADCs, and high accuracy references are needed for proper operation of the comparators and Sigma Delta ADCs. For wireless sensor nodes implemented in smaller processes, this implementation would be challenging to build.

The third comparison paper [92] is a wireless blood pressure sensing microsystem that leverages an 11-bit cyclic ADC. This paper provides a good comparison against another wireless sensor implementation, but one that uses a non-traditional ADC topology in order to achieve low power dissipation in a small area. This implementation achieves 12μW power dissipation at a power supply of 2V in 1.5μm CMOS. Because the cyclic ADC incorporates high-accuracy capacitors, it may not scale well to 28nm or other advanced process nodes. Additionally, the conversion scheme relies on a high resolution comparator and precise timing references in order to properly operate the large number of switches in the design.

The fourth comparison paper [93] is a reconfigurable VCO-based sigma-delta modulator.

This paper uses a mostly-digital design with digital background calibration and dithering, eliminating the need for high accuracy voltage, current and timing references. This paper provides an important comparison against another VCO-based ADC which also aims to eliminate high accuracy references, and capitalize on a digital implementation to achieve large area gains over traditional implementations. However, the large amount of calibration circuitry used in this implementation requires area usage of about 13,000μm^2 (38× larger than the proposed design - after ideal scaling), and between 8-17mW power consumption (683× - 1453× increase over the proposed design).

Additional comparison points can be readily found in Boris Murmann's ADC survey [94]. Other implementations that are less than $0.1mm^2$ in size can be found with an effective number of bits (ENOB) between 4 and 12-bits [95, 96], but power on the order of mW, and area between 4 and 290 larger than our proposed design. Other scalable resolution ADCs [77, 88, 97] present SAR topologies with limited resolution scalability from 2 settings (8-bits or 12-bits) up to a 5-bit range (5-to-10 bits), and comparable power consumption of 0.2μW to 17.4μW. In addition to requiring high accuracy references, these designs require large capacitor arrays for resolution scaling, up to 1800× more area than the proposed design, and a limited range of resolution scalability (2-to-4× less resolution range). As shown, our design is unique in its combination of exceptionally small area, wide range of resolution scalability, and simplicity in reference-free design.

Table 4.1: **Results and Comparison.** Comparing the proposed adc design with similar state-of-the art ADCs. The proposed work achieves an area consumption that is $1/100^{th}$ of prior approaches when all comparisons are ideally scaled to 28nm.

	This Work			**Yip [88]**	**Suster [75]**	**Cong [92]**	**Taylor [93]**
Technology	28nm			65nm	$1.5\mu m$	$1.5\mu m$	65nm
Architecture	VCO			SAR	1^{st}-order $\Delta\Sigma$ 2^{nd}-order $\Delta\Sigma$	Cyclic ADC	VCO-based $\Delta\Sigma$
Key Features	Scalable Resolution Low Area			Scalable Resolution	Multi-Sensor Multi-ADC	Low Power	Variable Rate
Needs Accurate Current Sources	**No**			Yes	Yes	No	No
Needs Low-Jitter Timing Reference	**No**			Yes	Yes	Yes	No
Area μm^2	346			212,000 39,339***	1,597,000 (total)**	532,400*	70,000 12,989***
VDD (V)	0.6	0.8	1.0	0.4-1.0	3.0	2.0	1.2 and 2.5
Effective Resolution	2.8 - 11.0	3.7 - 11.7	3.7 - 11.0	6.4-10.5*	11.2*-strain 12.6*-temp	12.3*	13.1-14.8*
Power (μW)	11.7	43.9	105.0	0.206 @0.7V	Not Reported	12	8,000- 17,000
Conversion Time (μs)	0.036 -9.3	0.022 -3.3	0.016 -1.0	0.5(@1.0V) 200(@0.4V)	(1^{st}-order) 5.88 (2^{nd}-order) 50	500 500	0.00086 - 0.002

*ENOB to ER conversion via IEEE standard 1057
**Not Reported - Estimated from die micrograph
***Ideal scaling applied for area comparisons

4.5 Conclusion

This work presented a reference-free, VCO-based sensor interface circuit in 28nm LP CMOS, designed to specifically address the constraints of wireless sensor nodes. This design is implemented in an area 1/100th that of prior approaches. Resolution scales between 2.8 and 11.7bits, and the power supply is scalable from 500mV to 1.0V. The design contained a single-point calibration scheme that functions well across temperature, voltage, and resolution. The ease of design and use, in addition to the wide range of operating conditions of this circuit, allow for implementation in a variety of sensor applications.

CHAPTER 5

25 Gbps Receiver Equalization Technique Using Active Inductors

5.1 Motivation

State of the art processors achieve clock rates on the order of 3-4GHz [64]. Combining many of these ICs on a single board, with chip-to-chip communication, allows for the construction of very large, complex system-on-chips (SoCs), with billions of transistors operating at multi-gigahertz frequency. Efficient communication between chips is necessary to facilitate further development of these systems. Off-chip bandwidth scales at a much lower rate than on-chip bandwidth [1, 98], and very long (100s of mm), lossy ($> 10dB$), serial links behave like transmission lines. This behavior degrades performance and introduces inter-symbol interference (ISI) in the signal. As shown in Figure 5.1, a signal sent across a 320mm channel at 12.5GHz produces a completely closed eye response (no clear transitions in the signal) at the input of a receiver, based on simulated results.

It has long been known that equalization is needed to restore the signals transmitted across these serial links [99], both for chip-to-chip communication, and for signals sent between blocks across a large chip. Prior work in this field has shown that equalization is possible through modulating the strength of either the transmitter output (*pre-emphasis*) or the receiver output (*post-compensation*) [100].

While pre-emphasis is easily implemented by a finite impulse response (FIR) filter with

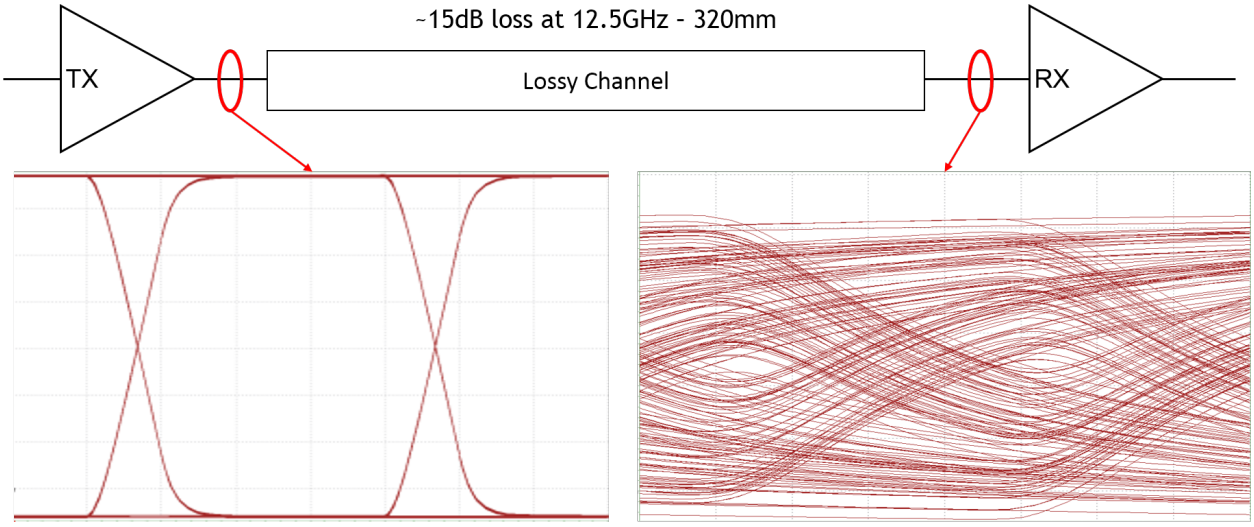


Figure 5.1: **Un-equalized lossy channel example.** With a perfect transmitter output, a 320mm channel with 15dB loss at 12.5GHz will produce a completely closed eye at the input of a receiver - due to ISI, frequency specific characteristics of the RLC channel.

digital control, it typically results in high power consumption and increased crosstalk because it relies on gain boosting to high-frequency components of the transmitted signal. Additionally, adaptive equalization tuning of the pre-emphasis block requires information sent back to the transmitter from the receiver side, as the receiver is the only element that sees whether or not the equalization strategy is effective. Due to signal swing level restrictions based on the power rails, pre-emphasis techniques also require more gain at the receiver side, as the gain boosting provided by the transmitter itself is not large enough to produce a full-rail signal on the receiver. Equalization is more commonly performed at the receiver side for these reasons [101].

Typical implementations of *post-compensation* also involve FIR filters with digital control for fine tuned equalization. These filters use a few signal samples (taps) in order to achieve better equalization and are referred to as decision feedback equalizers (DFEs). DFEs are heavily digital solutions to the equalization problem, and suffer from large costs in both area and power consumption. They require fast digital compensation techniques for 1^{st} tap equalization (the first sample), and registers and adders for subsequent taps (delayed samples). First tap equalization is costly to achieve because it involves a feedback loop within the unit impulse response of the system. While analog approaches to DFEs have proven to

achieve low power operation [102], they still cost around 11mW power consumption.

Another equalization strategy is to counteract the channel RLC response by adding complex poles to peak the frequency response at the desired sample rate. As shown in Figure 5.2, when combined with the channel transfer function, the equalizer produces a flat response up to the sample rate - canceling the distortion introduced by the channel. This work presents an ultra-low power receiver equalizer using active inductors to insert a complex pole into the frequency transfer function. This system consumes 470μW to equalize at 25Gbps.

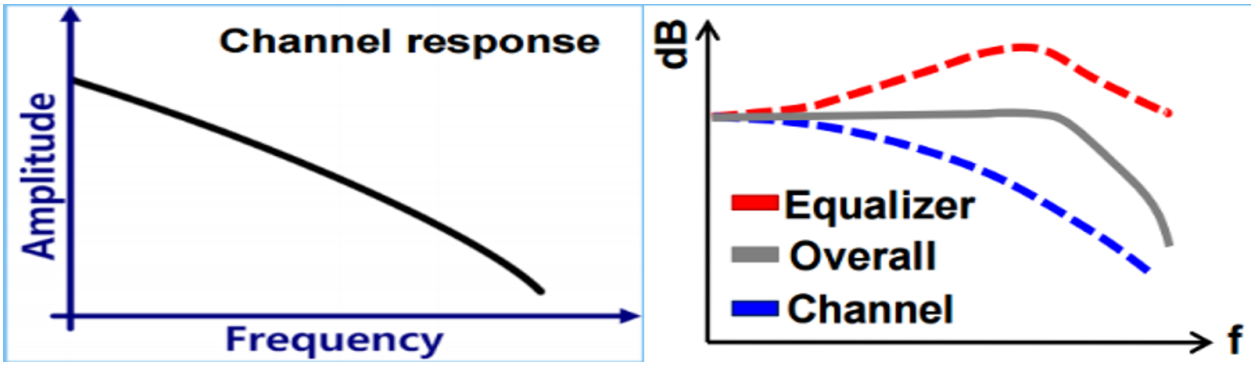


Figure 5.2: **Channel transfer response correction using complex poles.** (Left) Infinite pole channel transfer response, showing the distortion caused by its frequency dependent response. (Right) The combined transfer functions of the lossy channel and an equalizer with a complex poles produces a flat overall response up to the peaking frequency of the equalizer.

5.2 Receiver Design

The proposed receiver equalizer design uses a standard common gate amplifier receiver and substitutes an active inductor for the typical resistor load. The output of the common gate amplifier is fed into a series of 3 inverter stages to amplify the signal to full rail. The active inductor and common gate amplifier are tuned to achieve equalization at 25Gbps on channel inputs with up to 15dB loss at 12.5GHz. This design shows that equalization can be achieved with minimal circuitry above the standard receiver design.

5.2.1 Ground-Referenced Signaling

Traditional approaches to high-speed, off-chip signaling are often high power and consume large die area, and are differential which halves the effective signaling rate per trace. Differential signaling is typically preferred because of its beneficial properties of power and noise rejection; but differential costs two pins and two well-matched traces per signal in order to overcome the halved data rate drawback. Ground-referenced signaling (GRS) addresses the drawbacks of single ended signaling by using the ground plane as a reference voltage - creating a pseudo-differential signaling environment with one pin per signal. Because the ground plane is typically the lowest impedance node in a system, it provides a voltage reference that is easier to match and lowers the supply impedance for decreased switching noise.

GRS systems pass information across off-chip channels by sending a toggling data signal along with the ground plane voltage. The data signal is referenced to ground, swinging above and below ground by half the normal signal amplitude. The receiver takes in both the data and ground signals and determines whether the data signal is greater than or less than ground. While this system still uses two "signals" to transmit data, the cost is closer to single ended, as ground connections to the package ground are cheap, and can be amortized across other parts of the system - with other ground connections [103].

In addition to GRS, organic substrates can produce signaling environments with low attenuation, low crosstalk, and nearly constant impedance [104]. Unlike channels on a PCB which have high attenuation, cross-talk and discontinuities, organic substrate channels can facilitate systems with high speed, low area and low power. This type of environment, combined with GRS, facilitates the use of simpler, lower power equalization systems for very high speed off-chip signaling.

5.2.2 Baseline Receiver

The baseline receiver architecture used in GRS applications is shown in Figure 5.3 on the left. It is a standard common gate amplifier with a resistor load. The 50Ω resistor on the source side of the NMOS is used for proper channel termination, and the resistor load

on the drain side of the NMOS is used to control the amount of current (strength) of the receiver. More current allows the receiver to operate at higher frequencies. The output of the channel is fed directly into the input of this receiver, which is the NMOS source.

Figure 5.3 on the right shows the modification used in this proposed architecture to achieve equalization. Combined with a resistor, an inductive load forms an RL circuit, which is a single-pole filter. This filter has a complex pole located at:

$$s = -R/L \tag{5.1}$$

The current flowing through the receiver is dictated primarily by the common gate NMOS device, and the RL load controls the frequency response of the overall transfer function:

$$V_{out}/V_{in} = -g_m \times (R + Ls) \tag{5.2}$$

By adjusting the inductance value L, or the resistance R, we can place the pole at a precise location. This allows us to control the peak of the equalizer transfer function from Figure 5.2, to properly balance the transfer functions and effectively cancel the channel response.

5.2.3 Active Inductors

Passive inductors are notoriously difficult to place in silicon, due to large area requirements in achieving high levels of inductance and performance. On-chip *active* inductors achieve high inductance values independent from chip area, allow for electronic inductance tuning and high Q-factor, all with very low area cost. Figure 5.4 shows two different active inductor designs, the *Wu* and *Hara* inductors.

Fundamentally, an active inductor is a gyrator loaded with a capacitor. A gyrator consists of two back-to-back transconductors, and when loaded with a capacitor it is referred to as a gyrator-C network with the following transfer equation:

$$Y = \frac{I_{in}}{V_2} = \frac{1}{s \times (\frac{C}{G_{m1} \times G_{m2}})} \tag{5.3}$$

It can be seen that this transfer function has an inductive element:

$$L = \frac{C}{G_{m1} \times G_{m2}} \tag{5.4}$$

These equations represent a loss-less gyrator-C based active inductor. In implementation, the transfer functions will be more complex and will by lossy, as infinite input and output impedances, as well as constant transconductance, are unrealistic. Lossy gyrator-C networks produce active inductors that are networks of inductive, resistive and capacitive elements [105]. The simplest of these active inductor topologies are the single-transistor circuits shown in Figure 5.4. By simplifying and reducing the number of elements of the active inductor, we can achieve a very small, low-power implementation.

Both of these active inductors consist of a single MOSFET, capacitor and resistor. The only difference between the two is the use of an NMOS or PMOS device, and subsequently the order of the capacitor and resistor. The input impedance of the Hara inductor, derived from the small- signal circuit is as follows:

$$Z \approx (\frac{1}{RC_{gs}C_{gd}}) \frac{sRC_{gd} + 1}{s^2 + s\frac{g_m}{C_{gs}} + \frac{g_m}{RC_{gs}C_{gd}}} \tag{5.5}$$

From this equation, we can determine the self-resonant frequency ω_o and the frequency of the zero ω_z:

$$\omega_o = \sqrt{\frac{g_m}{RC_{gs}C_{gd}}} = \sqrt{\omega_t \omega_z}\omega_z = \frac{1}{RC_{gd}} \tag{5.6}$$

We can see that $C_{gd} = C_{gdNMOS} + C$, so the zero frequency can be tuned through changing the resistor, capacitor or transistor size.

Similarly, the Wu inductor has the following transfer equation:

$$Z \approx \frac{sRC_{gs} + 1}{sC_{gs} + g_m} \tag{5.7}$$

Following the same approach as with the Hara inductor, we find the zero frequency to be:

$$\omega_z = \frac{1}{RC_{gs}} \tag{5.8}$$

Both of these inductor topologies are implemented with the fewest number of elements, and allow for inductance tuning through all three devices. Because our baseline common gate amplifier is NMOS-based, we decided to use the NMOS-based Hara inductor, to take advantage of device matching and tracking across power supply and temperature. The main advantage of the Wu inductor over the Hara inductor is an increased voltage swing range, as the PMOS in the Wu inductor allows the output voltage to reach full rail. This is not necessary in the proposed equalizer, as the output of the common gate amplifier is biased around $V_{DD}/2$ and does not need a large voltage range because there are high gain stages after to boost the signal to full-rail.

5.2.4 Full System Design

The full system architecture is shown in Figure 5.5. It consists of a common gate (CG) amplifier, the active inductor load, and three inverter stages on the output of the CG amplifier. The active inductor is implemented using all MOSFET devices for tunability, area and matching. The resistor element is made up of multiple MOSFET switches to construct different series and parallel combinations of resistances. The capacitor is implemented with a metal oxide silicon capacitor (MOSCAP), capacitively connecting the CG amplifier output and the gate of the gyrator.

The bias voltage coming to the gate of the amplifier is generated using a small replica circuit that also employs digital tuning in order to properly set the DC bias of the CG amplifier output. The voltage is adjusted so that the output of the common gate amplifier is DC biased to the trip point of the 1^{st} inverter stage. Nominally this is $V_{DD}/2$, but needs to be tunable for process and mismatch correction. Setting the output to the trip point of the 1^{st} inverter sets the system in its largest gain state and is critical to the functionality of this technique. If the bias point is set too low or too high, the output of the receiver equalizer will be heavily biased toward one of the rails and will not transition properly.

We chose to use a 3-stage inverter chain to generate the full swing output in order to

reduce the amount of current per finger in each inverter, and to reduce the capacitive load on the output of the CG amplifier stage. This circuit could be implemented with 1 or 2 inverters, but those inverters would need very high gain and speed in order to work properly, requiring a large amount of per finger current. Additionally, the common gate amplifier equalization capability is sensitive to the amount of load capacitance on the CG output. By using 3 inverters instead of 1 or 2 we are able to reduce the size of the 1^{st} inverter, thus reducing the load on the amplifier.

5.2.5 Offset and Equalization Tuning

This equalizer strategy is dependent upon proper tuning for both *offset* and *equalization*. We refer to offset tuning when discussing the DC bias of the common gate amplifier output, and equalization tuning when talking about the tunable resistor in the active inductor load.

Offset tuning ensures that the DC bias point of the common gate amplifier is at the trip point of the 1^{st} inverter. Nominally this occurs at $V_{DD}/2$, but requires range for tuning out all process and mismatch variations. To accomplish this we use a replica circuit which is sized down proportionally to minimize added power consumption. This circuit does not require an active inductor because its small signal characteristics are unimportant, since it only operates in DC mode. Instead, the active inductor can be assumed to be in a DC state where the gate of the NMOS is at V_{DD} and there is no resistor or capacitor. The replica circuit is a common gate amplifier with a fully-ON NMOS load. To tune the output we use a 2-inverter feedback loop with tunable device sizes, as shown in Figure 5.6. The input of the inverter chain is the output of the common gate replica amplifier, and the output is the bias voltage which is connected to the gate of the NMOS in the replica and in the receiver equalizer. Tuning the sizes of the inverters allows us to change the value of the bias voltage. However, we need to observe the output of the actual equalizer - not the replica - because the two are not perfectly matched due to variation. Additionally, this test cannot be performed in a DC state, as the output of the receiver circuit comes from the last inverter in the amplifier chain. This value should always be either '1' or '0', not $V_{DD}/2$. Thus, we need to test the receiver in a transient state to determine if the CG amplifier is properly

biased for the 1^{st} inverter stage.

Figure 5.7 (top) shows the test that is used to perform offset correction. In this test we observe the duty-cycle of the receiver equalizer to determine whether or not it is properly biased. By using a 50% duty-cycle input, we guarantee that our receiver does not observe any inter-symbol interference (ISI) induced by the channel (only one frequency is passed), allowing us to tune out offset errors without needing to equalize (which also requires tuning). It is necessary to perform offset correction before equalization tuning because any offset errors will prevent the receiver from equalizing properly: we can fix offset without fixing equalization, but not the other way around. The goal of this process is to tune the bias voltage to the common gate amplifier until we see a 50% duty cycle output on the receiver.

After correcting the offset, the equalizer can be tuned for optimal eye opening. This is done by providing a digital code to the receiver to turn on or off the PMOS switches in the tunable resistor in the active inductor. From simulations we found that the receiver is optimally equalized with the minimum resistor value setting that allows it to generate a correct output. In the tuning process we input a pseudo random bit stream (PRBS) and observe the output of the receiver for correctness. Starting with the minimum resistor value digital setting, the value is increased until the output is correct (*i.e.* equal to the input stream). This makes equalization tuning trivial, as we no longer need the ability to measure an impulse response or observe the total eye opening, so it can be done without any intricate techniques or expensive equipment.

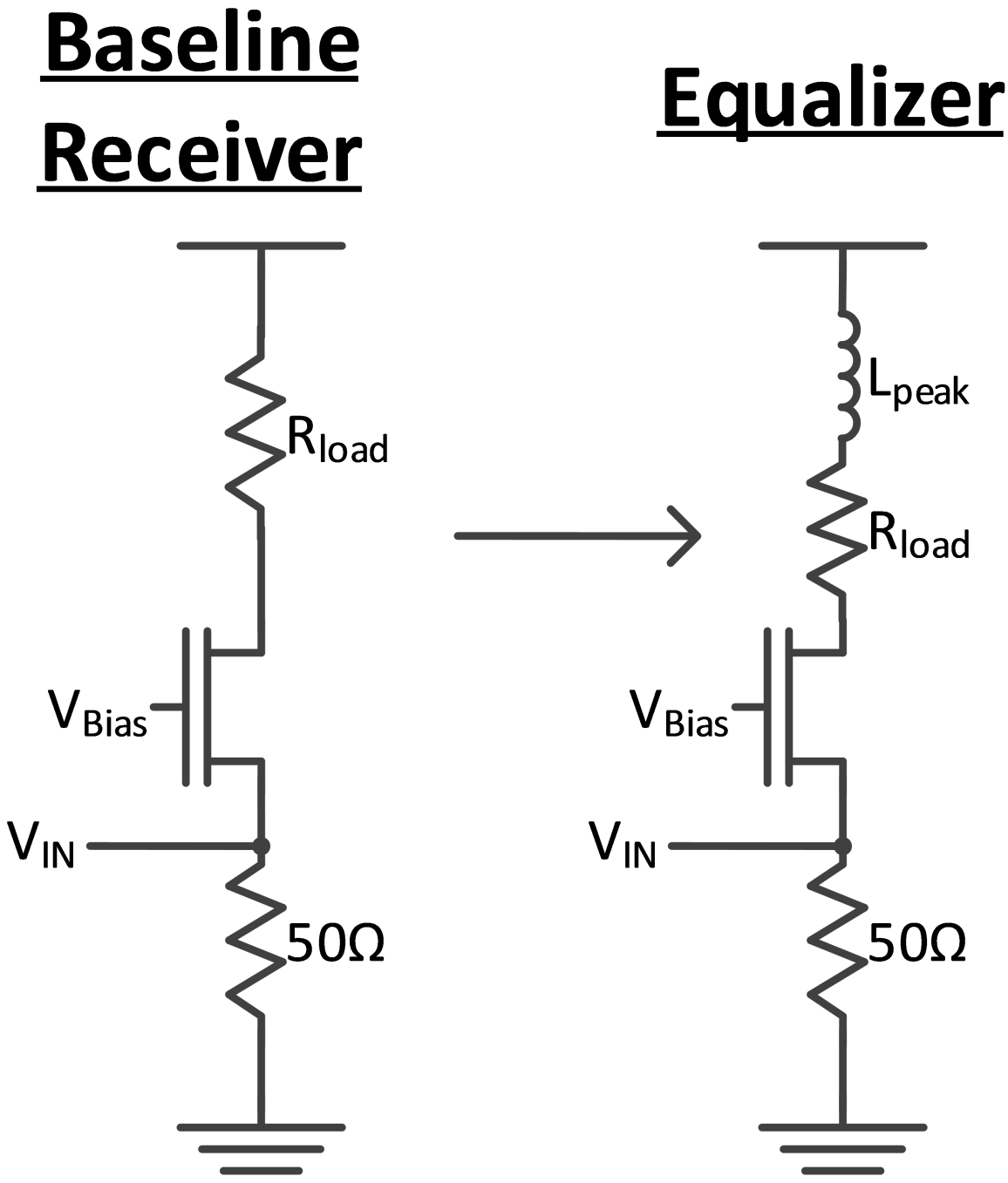


Figure 5.3: **Baseline receiver and equalizer circuits.** (Left) The baseline receiver is a resistor loaded common gate amplifier with 50Ω termination. (Right) A modified receiver with an inductor-resistor load provides a complex pole that places a peak in the equalizer transfer function, canceling out the ISI caused by the lossy channel.

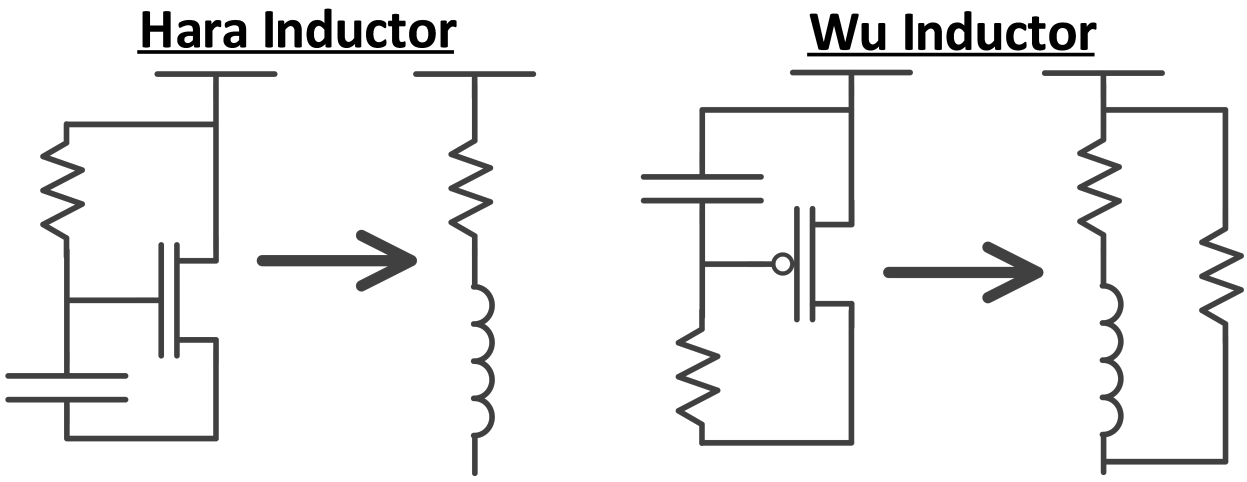


Figure 5.4: **Single transistor active inductor elements.** The Hara and Wu inductors are single transistor active inductors that can be used as the inductive load element in the receiver equalizer design.

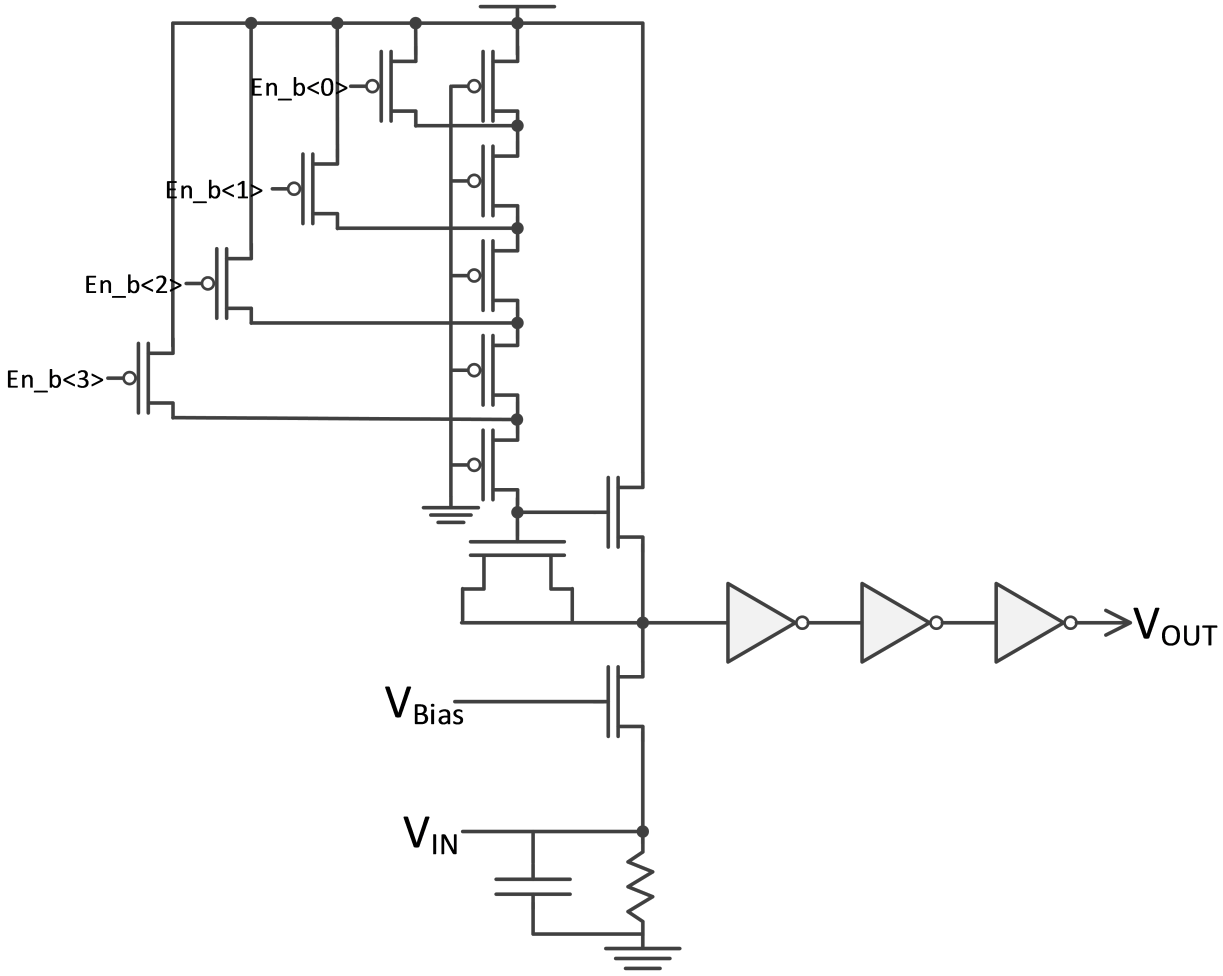


Figure 5.5: **Full receiver equalizer design.** The full receiver equalizer design consists of an active inductor with tunable MOS elements for the resistor and a MOSCAP, loading the baseline common gate amplifier. This inductive loaded amplifier is followed by a 3-stage inverter chain for full rail amplification.

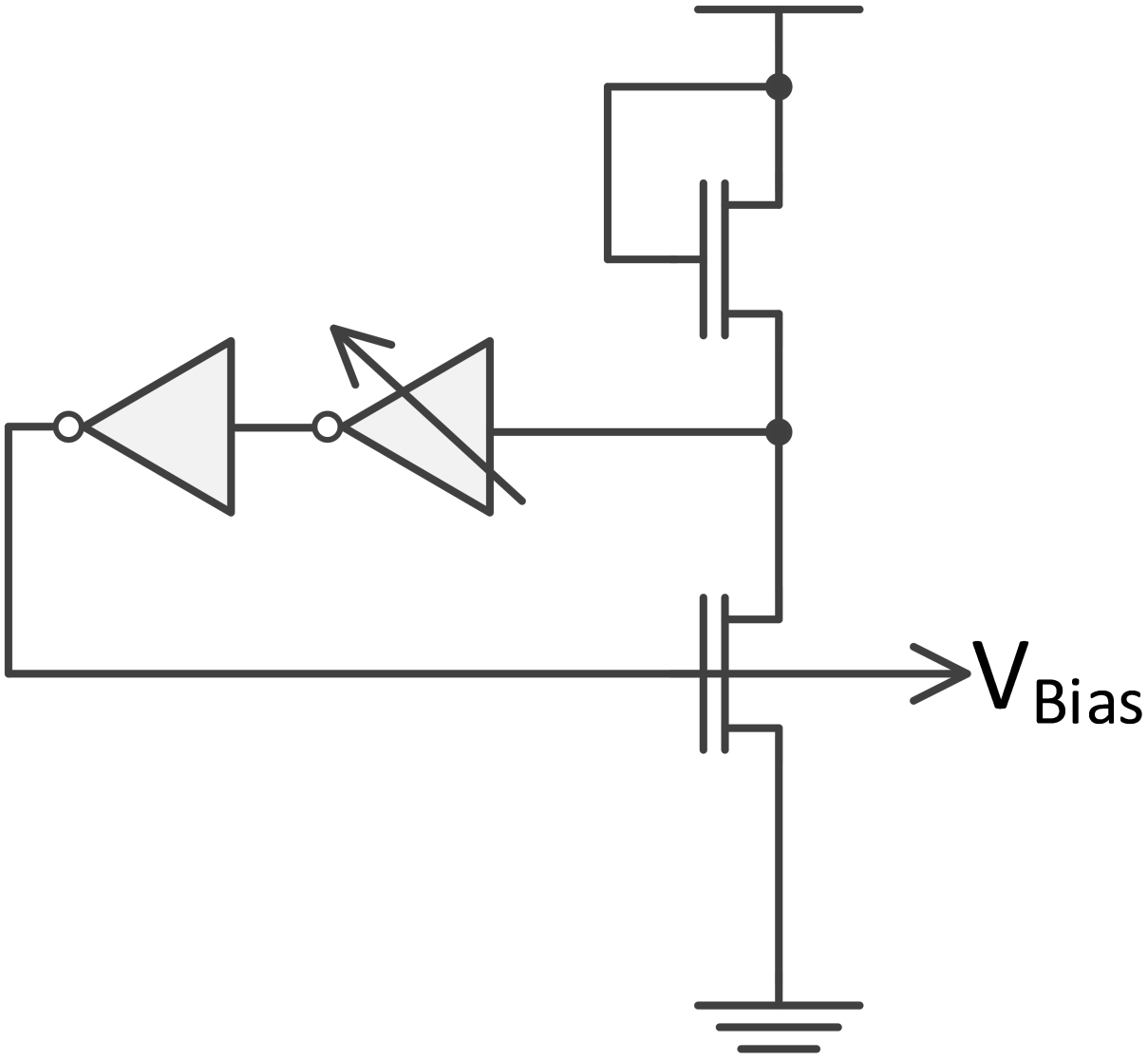


Figure 5.6: **Tunable Replica Circuit for Offset Cancellation.** Through digital tuning of the size of the inverter in the biasing feedback loop, we can adjust the bias voltage sent to the common gate amplifier in the receiver equalizer.

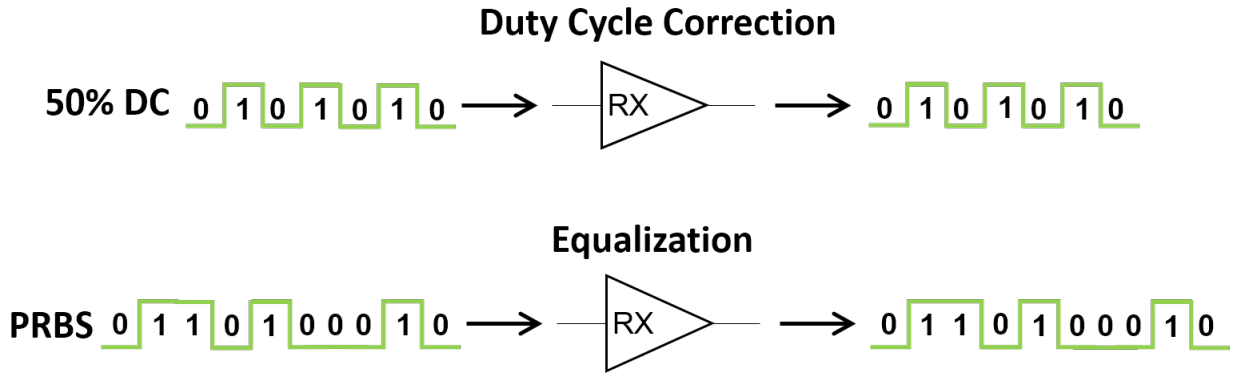


Figure 5.7: **Offset and Equalization Tuning.** (Top) Offset in the common gate amplifier output bias is tuned using a 50% duty cycle input to eliminate ISI, and the bias voltage is tuned until we observe a 50% duty cycle on the output of the receiver. (Bottom) After offset correction, the equalizer is tuned by inputting a PRBS and the resistor value is increased until a correct bit-stream is observed on the output of the receiver equalizer.

5.3 Simulated Results

This circuit was designed in a 16nm fin field effect transistor (FinFET) technology and simulated using Spice models and a resistance, inductance, conductance, capacitance (RLGC) model of the channel. To properly simulate, I developed a series of scripts to analyze and tune the receiver to its optimal state: correcting both offset and equalization settings for each simulation. These scripts were written in Ruby and used as wrappers to modify the digital codes sent to the resistor and replica circuits for sizing.

To analyze whether or not the receiver was properly equalized and how well the equalization technique worked, I wrote a script to measure the eye opening of the PRBS output. Shown in Figure 5.8, this script first tests the receiver by sending in a pseudo-random bit stream (PRBS) input from a linear feedback shift register (LFSR) and then observes the output of the receiver equalizer. By time folding the output based on the input response (1 unit-impulse (UI) = 40ps), we can generate a folded eye diagram. Time folding involves taking the modulo of each time step with the impulse response (or 2 UI as shown in Figure 5.8). It is important that the time step in the simulator is set correctly so that it prints out a small enough step to analyze properly. After time folding, the script analyzes the four different transitions: 1^{st} rise, 1^{st} fall, 2^{nd} rise, and 2^{nd} fall. By choosing the latest 1^{st} transitions and the earliest 2^{nd} transitions, we can identify the outline of the eye. At this point, taking the intersection of the chosen 1^{st} transitions and the 2^{nd} transitions will give us a value equal to the horizontal eye opening.

While a perfectly tuned and optimized equalizer would produce an eye diagram with transition point crossovers equal to exactly $V_{DD}/2$, realistically there will still be some residual offset error which results in a shift of these transition points up or down on the voltage scale. This method of measuring the horizontal eye opening measures the *largest* horizontal opening, not the eye opening at exactly $V_{DD}/2$, or at the proper inverter trip point. Additionally, this eye opening measure does not incorporate any information about vertical opening. A more complete and accurate algorithm would measure the rectangular *area* of the eye opening, which would take into account both horizontal and vertical information, as well as residual offset errors. This method was successful in allowing for a large amount of

analysis on the proposed circuit implementation over supply voltage, corners, and mismatch.

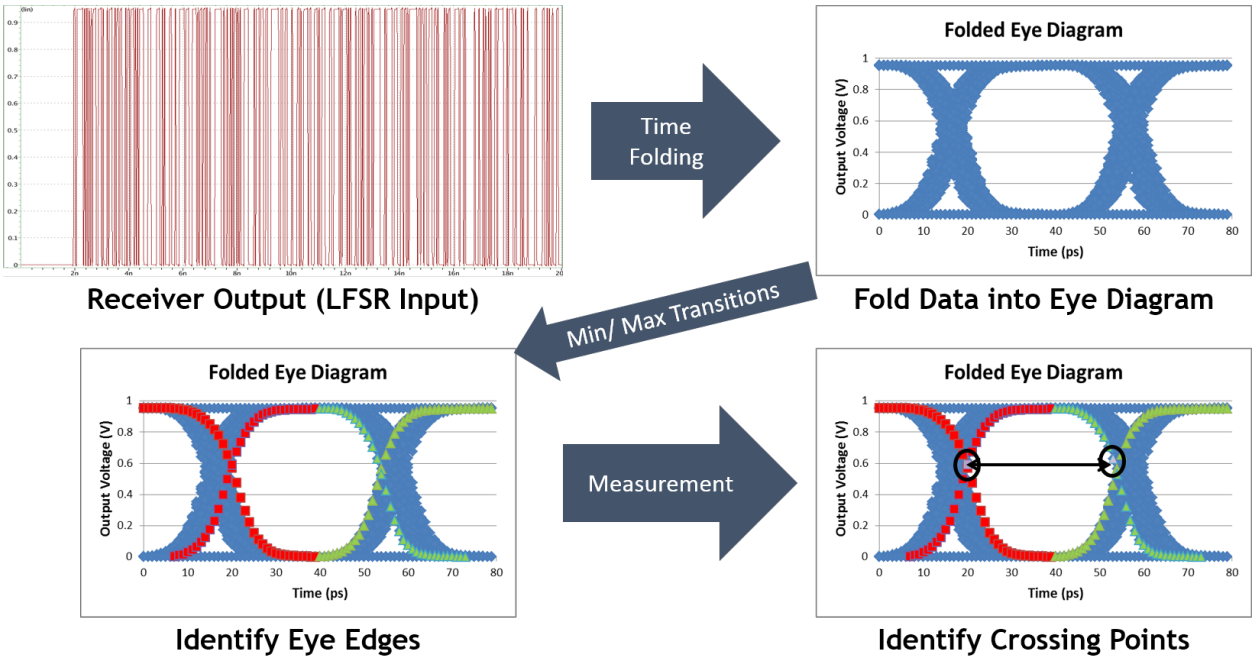


Figure 5.8: **Eye Opening Measurement Flow.** Describes the algorithm used to measure the horizontal eye opening (in picoseconds) to determine the effectiveness of the equalization technique.

5.3.1 Equalization Optimizing Routine

As described in Section 5.2.5, in its base state the receiver equalizer produces a completely closed-eye output response. Figure 5.9 shows the before-and-after outputs of each stage of the receiver during the equalization process. With a 320mm channel with 15dB loss, the ISI caused by the channel's poles produces a 100mV amplitude, fully closed-eye output - even with a perfect transmit signal. When fed through the common gate amplifier, the output is also a fully closed-eye, with an amplitude of 70mV. The voltage gain of a common gate amplifier is described in the following equation:

$$\frac{V_{out}}{V_{in}} = g_m R_L \tag{5.9}$$

The active inductor NMOS is sized equal to the common gate NMOS because it produced the best equalization results and having equal sizes makes biasing the output at $V_{DD}/2$ easier,

Table 5.1: **Eye Opening Across PVT.** Simulated measurements of equalizer eye opening with optimization scripts, using a PRBS input.

Horizontal Eye Opening (UI = 40ps)									
FF			SS		TT	SF		FS	
0.95,125	0.75,-20	0.95,-20	0.95,125	0.75,-20	0.85,85	0.95,125	0.75,-20	0.95,125	0.75,-20
33.7	33.8	34.5	33.4	32.6	33.5	33.3	33.6	34.3	33.8

as the devices naturally split the supply in half. Because of this, R_L is roughly equal to $1/g_m$, giving us a gain of about 1. After equalization, we see a fully open eye with clear transitions.

Following this receiver design will be samplers which require a full-swing digital signal from the output of the receiver. The un-equalized and equalized outputs of the full receiver are shown in Figure 5.9. The un-equalized version is clearly un-readable, with most outputs at either rail and many unfinished transitions. The equalized output shows a clear, open eye window with little leftover ISI.

5.3.2 Process and Mismatch Results

To test the tunable range of this system, we simulated the full receiver using the wrapper scripts described in Section 5.2.5 over 10 different process/voltage/temperature (PVT) corners with device mismatch (Monte Carlo) at each corner. As shown in Table 5.1, we simulated this receiver with power supply varying from 0.75V to 0.95V, where nominal is 0.85V (±11%), and temperatures ranging from -20°C to 125°C. Additionally, this system was tested at all 5 process corners (FF, SS, TT, SF, and FS).

The eye opening of the receiver output does not fall below 32.6ps (where 40ps is ideal) for any of the 10 PVT corners. The worst result is at the SSLL (SS, 0.75V, -20°C), and the best result is at the FFHL corner (FF, 0.95V, -20°C) and is 34.5ps. This receiver equalizer works better with higher current flow through the common gate amplifier. Any corner that produces more current such as FFHL will see a better result while low current corners such as SSLL see worse results. Because there are no PMOS devices in the critical CG amplifier or the active inductor, slow PMOS corners do not have a strong effect on the performance of this circuit.

Figure 5.10 shows the histogram resulting from the Monte Carlo simulations. Each of the

10 PVT corners was run through 100 Monte Carlo simulations - which involved performing the entire tuning, correction and optimization process (upwards of 30 simulations per Monte Carlo result). The results are tightly clustered between 31ps (worst) and 34.8ps, and display a log-normal response, with a 2ps tail. This shows that the tuning algorithm is able to correct a wide range of PVT and mismatch situations, equalizing all simulated cases.

Table 5.2 displays the Monte Carlo results, with statistical information including mean, sigma and mean minus three sigma. Similar to the PVT table, SSLL corner has the worst-case mean of 32.95ps, and FFHL has the best case mean value of 34.38ps. However, some corners have higher skew, with FSHH having a standard deviation of 0.5ps, versus the 0.09ps sigma measured in TTNN. The resulting worst-case $\mu - 3\Sigma$ is measured in the FFHH corner at 32.06ps, due to the high skew measured at that corner, 0.46ps. Best case $\mu - 3\Sigma$ is 33.64ps at the FFHL corner.

5.3.3 Channel Length

This receiver equalizer was designed for channel lengths *up to* 320mm, with losses near 15dB at 12.5GHz. While this is the maximum amount of ISI and loss that we expect, it is necessary for this system to work at shorter channel lengths that have less loss. As shown in Figure 5.11, reducing the channel length reduces the amount of ISI, which causes the system to over-equalize. This results in larger eye openings, but clustered output responses. When over-equalized, the eye diagram for the system will have transitions that are clustered tightly together and then separated from each other by a couple picoseconds. This is shown clearly in 5.11 where you can see four intersection points on each transition edge where they have clustered together. Clustering and over equalization are not problematic, but demonstrate the robustness of the proposed system for varying channels and channel lengths.

Simulation results show that as the channel length decreases from 320mm to 0.1mm, and the loss at 12.5GHz decreases from 15.4dB to 0.97dB, the horizontal eye opening increases from 34.2ps to 36.6ps. This demonstrates that the proposed equalization technique is not limited to a specific channel, but that it can be used on shorter channels as well.

Table 5.2: **Monte Carlo Results.** Simulated eye opening measurements across 10 process corners and with monte carlo mismatch applied. This table shows a statistical analysis of the horizontal eye opening achieved through the equalizer.

	μ (ps)	σ (ps)	μ-3σ
SSLL	32.95	0.23	32.27
TTNN	33.52	0.09	33.26
FFHH	33.45	0.46	32.06
FFNN	33.86	0.16	33.37
FFHL	34.38	0.24	33.64
SSHH	33.42	0.10	33.13
SFHH	33.26	0.30	32.35
SFLL	33.29	0.32	32.33
FSHH	33.85	0.50	32.35
FSLL	33.81	0.27	32.99

Table 5.3: **Comparison with Previous Works.** Comparison of data rate and power between the proposed work and other state-of-the-art equalizers.

	Wang [106]	**Dickson [102]**	**Toifl [107]**	**This Work**
Data Rate	21Gb/s	12Gb/s	12.5Gb/s	25Gb/s
Technology	65nm	45nm	32nm	16nm
Architecture	1-Tap DFE Analog Filters	5-Tap DFE	2-Tap Switch Cap DFE	Active Inductor RX
RX + EQ Power	42mW	11mW	32.5mW	0.47mW
FOM (Power/Gbps)	2	0.92	2.6	0.019

5.3.4 Power Analysis

Figure 5.12 shows a power breakdown of the full receiver equalizer. The total system consumes 470.8μW, and is dominated by the power consumption of the common gate amplifier, which is more than 90% of the total power. Based on simulation results, the lowest possible power with which the resistor loaded receiver could operate at 25Gbps was approximately 200μW. Adding the active inductor equalizer and increasing the current draw to properly equalize the channel cost about 270μW over the baseline (required) implementation.

Compared to previously published works, this system achieves significantly reduced power consumption - between 48× and 218× in mW per Gbps [102, 106, 107].

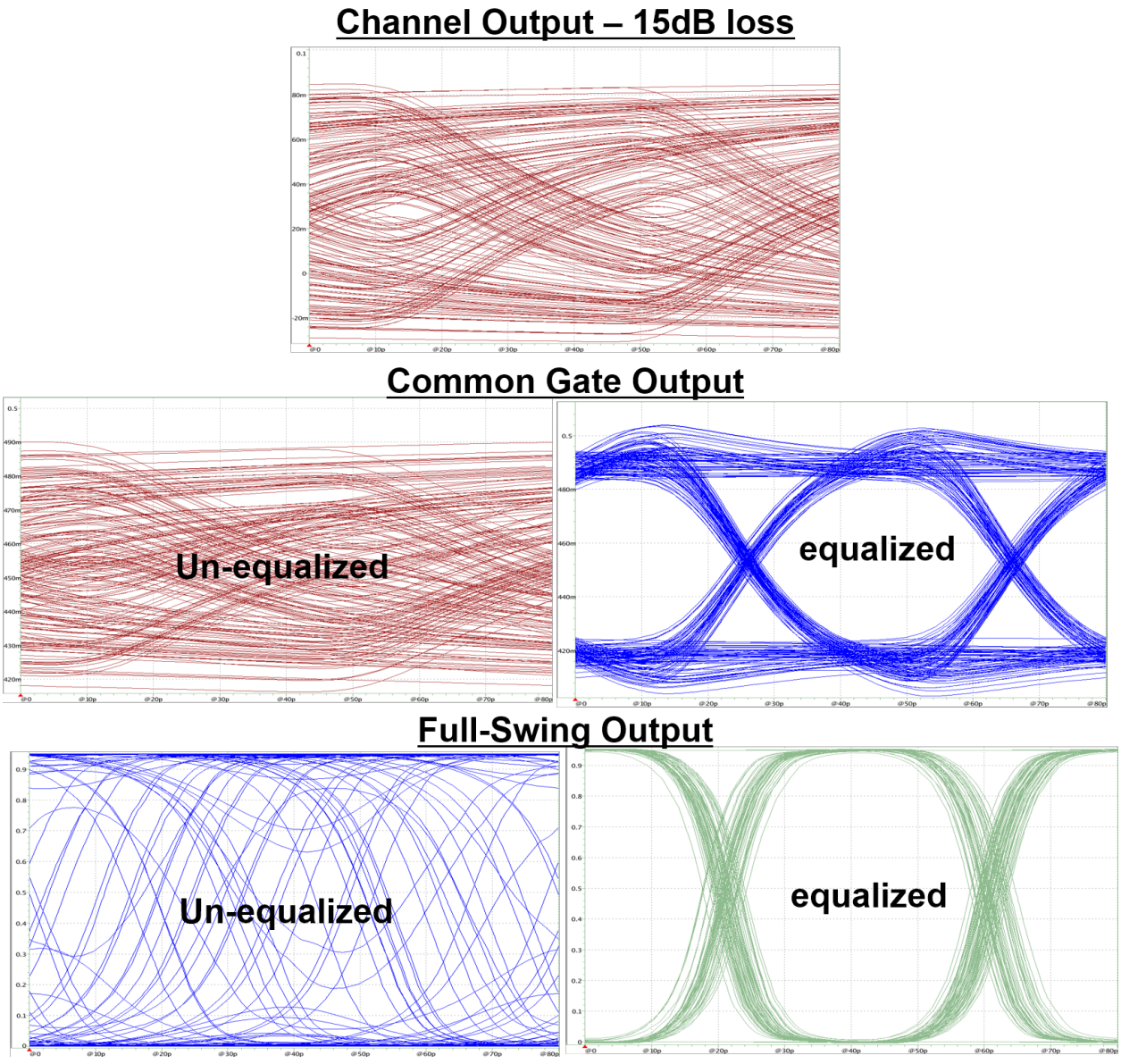


Figure 5.9: **Simulated Receiver Outputs: Un-equalized to Equalized.** Before equalization (after offset correction), the receiver produces a full-swing, fully- closed eye response. (Top) The input to the receiver, output of the channel is a fully closed eye with an amplitude of 100mV. (Middle) Un-equalized and equalized outputs of the 1^{st} stage of the receiver, the common gate amplifier. The equalized response shows a fully open eye. (Bottom) The output of the final stage of the receiver shows a fully-open, full-swing eye.

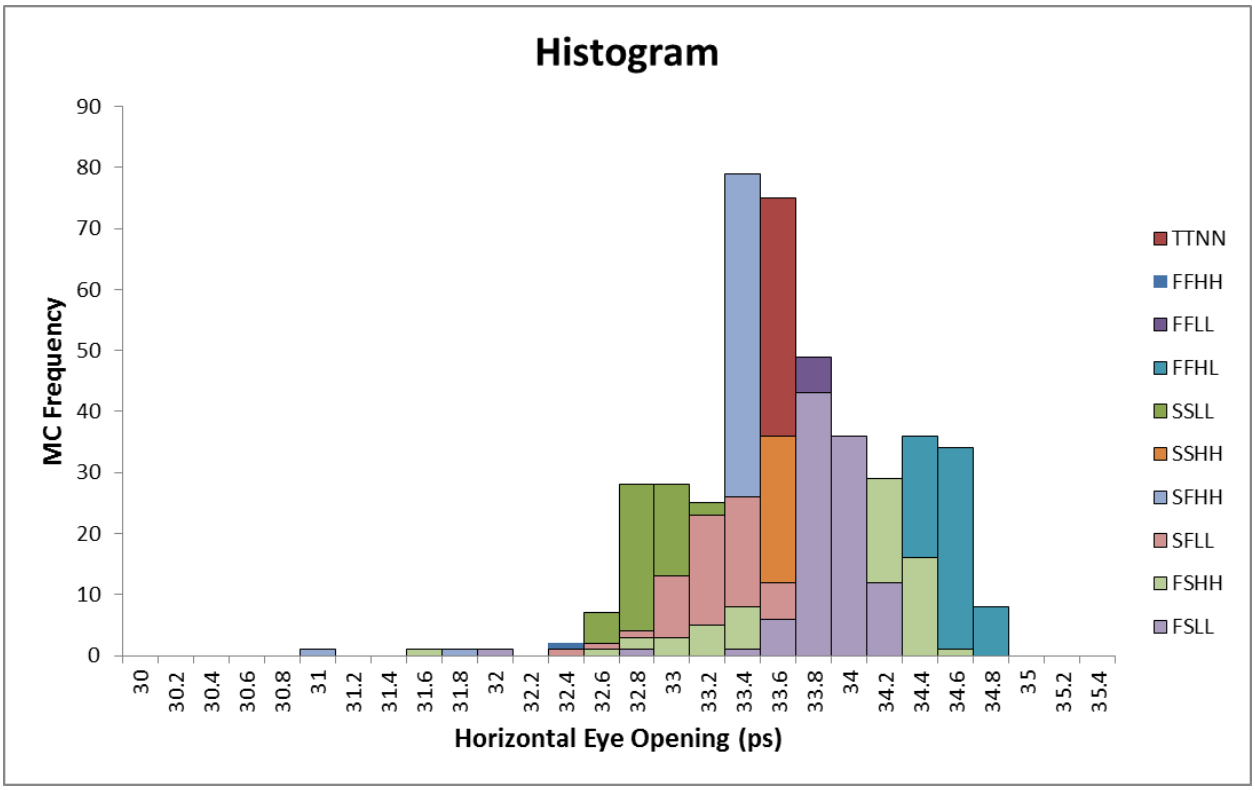
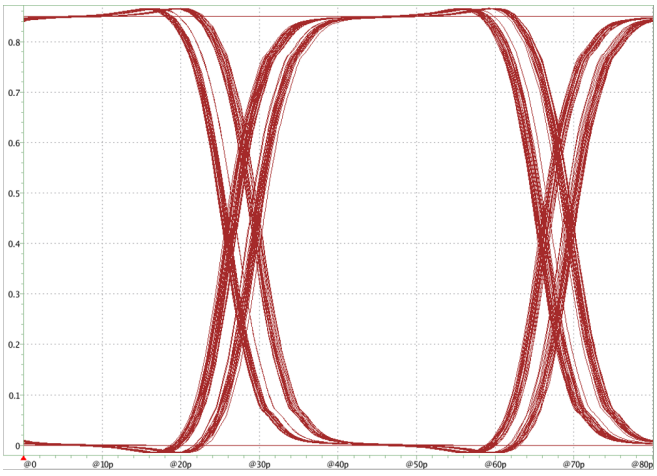


Figure 5.10: **Corner and Monte Carlo Eye Opening Measurements.** Monte Carlo results in a histogram plot, with the worst case result of 31ps. Most results are tightly clustered between 32.4ps and 34.8ps, and overall the output shows a log normal response with a 2ps tail down to 31ps horizontal eye opening.



Length (mm)	Loss @ 12.5GHz (dB)	Eye (ps)
320	15.4	34.2
220	10.9	34.9
120	6.42	35.9
20	1.89	36
0.1	0.973	36.6

Figure 5.11: **Channel Length Effect on Eye Opening.** This system was designed for a maximum channel length of 320mm. Smaller channel lengths produce larger eye openings and clustered eye diagrams due to over-equalization.

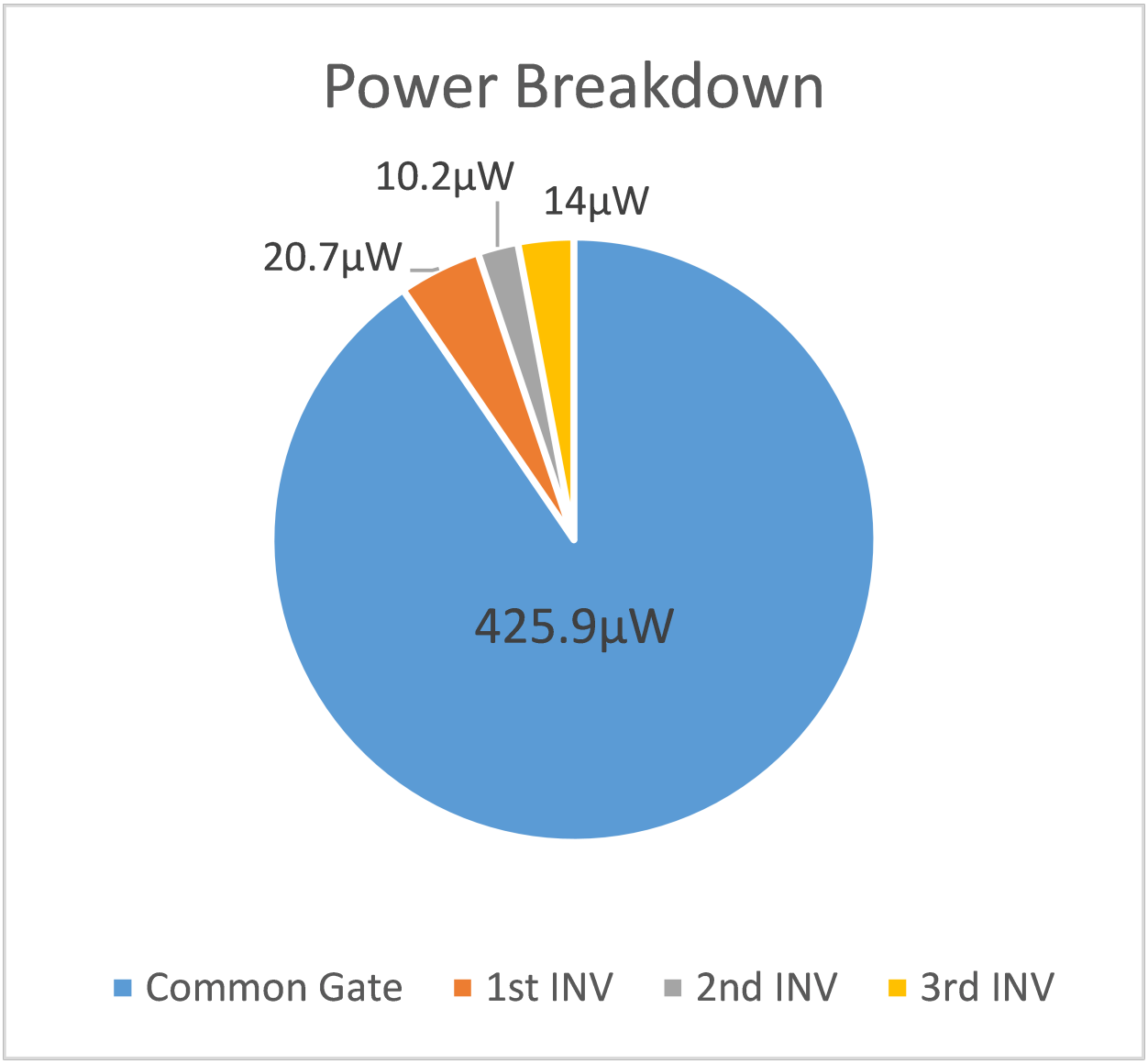


Figure 5.12: **Receiver Equalizer Power Breakdown.** The full receiver and equalizer consumes 470.8μW, with the common gate amplifier accounting for more than 90% of the total.

5.4 Conclusion

This work presented an ultra-low power, simple design that combines an active inductor with the standard common gate amplifier receiver topology for high-speed GRS systems. With 470μW power consumption, this equalizer design achieves between 48× and 218× lower power consumption per Gbps over previously published, low-power equalizer designs.

By combining the receiver and equalizer structures we have proposed a simplistic, dense, low-power equalizer that adds little power consumption above the required amount from the baseline receiver. A series of scripts and simulations have shown that the proposed design is capable of full equalization across 10 PVT corners and across mismatch variation; proving to be a robust design and equalization tuning algorithm.

CHAPTER 6

Conclusion

Current state-of-the-art neural network algorithms require data or internet connectivity to perform cloud-based computation.

6.1 Contributions

The key contributions of this dissertation are summarized as follows:

- Chapter 2 discussed a massively parallel neuromorphic computation circuit using analog in-memory computation. Storing all synaptic weights in non-volatile embedded memory eliminated all off-chip weight memory reads. Analog dot-product computation is performed through triode-mode current summation, where an analog current multiply is performed between the V_{th} of the NVM cells and the V_{GS} of the access transistors. The neuromorphic accelerator achieved a 15× reduction over similar digital hardware neural networks [108], a 4× reduction over a convolutaional-only batch-processed implementation [58] and is on-par with the a convolutional-only highly sparse network [57]. The demonstrated system is capable of supporting all DNN topologies, and its energy efficiency is independent of batch processing, data sparsity or architecture.

- Chapter 3 discussed a modification of the neuromorphic accelerator demonstrated in Chapter 2. Instead of performing the dot-product calculation using triode-mode NVM cells, the sub-threshold neuromorphic accelerator performs the same calculation

the sub-threshold operating region. The demonstrated system achieves a linear dot-product calculation through logarithmic gate voltage transformations and exponential programming routines. Because sub-threshold cells draw orders of magnitude less current, the demonstrated system is able to achieve a 40× reduction in cell power over the system proposed in Chapter 2.

- Chapter 4 presents a voltage and resolution scalable vco-based ADC. The demonstrated system achieves a form factor of $346\mu m^2$ in a 28nm CMOS process, 37.5× reduction over previous designs with ideal scaling applied. In addition to an ultra-dense form factor, the demonstrated system eliminates all high accuracy references (voltages and currents), and presents a highly digital system that allows for large scalability. The system achieves resolution scaling between 2.8 and 11.0 bits, and voltage scaling between 0.6V and 1.0V V_{DD}.

- Chapter 5 presents a novel active inductor receiver equalizer for gigabit communication systems. The proposed system is simulated in a 16nm technology and achieves a 48× reduction in figure of merit (Power/Gbps). An active inductor is used as the load to a standard common gate receiver, canceling out ISI through the addition of a complex pole. The equalizer is digitally tunable and can be calibrated for offset correction and equalization post-silicon.

6.2 Future Directions

The projects and ideas presented in this dissertation may be expanded upon, as discussed briefly in this section.

Neuromorphic Computing

The triode-mode neuromorphic accelerator achieved a highly parallel array of analog computational units within a non-volatile memory. Performance was limited by the multiplexing required to share amplifiers, in an effort to reduce power consumption. With a pitch matched design each neuron could have its own amplifier, allowing for a fully parallel readout

structure. This modification would consume more power, but two of the three amplifiers are shared between all neurons, amortizing their power consumption.

Another modification that would improve the quality of the neuromorphic results is a differential design. The demonstrated design implements excitatory and inhibitory synapses with opposing current flows to reduce the burden of the calculation amplifier required to sink/source the net current. This design suffers from an imbalance in voltages on the SONOS cells, where excitatory cells use a shifted gate and body voltage to compensate for source degeneration. This causes the two sides of the array to behave differently with both noise and offset. A fully differential design would use identical circuitry for both positive and negative weights, benefiting from increased power supply rejection ratio, noise rejection and PVT tracking.

Further area benefits could be acquired from moving toward a NAND flash design. NOR flash provides the benefit of parallel weights on shared bit-lines, allowing for up to fully parallel readout circuitry. NAND flash arrays use series connected cells for synaptic storage, and would be limited in readout structures to one row at a time in order to achieve linear current summation. However, they benefit from 2-3× area reduction, and are currently implemented in 3D topologies, further improving the area benefits.

Subthreshold Neuromorphic Computing

The sub-threshold adaptation of the neuromorphic accelerator achieved ultra-low power (900nA) operation while performing linear dot-product calculations via analog in-memory computational units. The demonstrated system uses off-chip amplifiers to verify the linearity of the sub-threshold calculation technique. Future work in this area could include designing and implementing on-chip subthreshold amplifiers. Additional work in the area of layer-to-layer voltage storage and routing is required.

Ultra-Dense VCO-Based ADC

An ultra-dense, reference free, voltage and resolution scalable ADC was presented in this paper, achieving an area consumption of $346\mu mm^2$ in a 28nm CMOS process. While the demonstrated system was able to scale in resolution from 2.8 to 11.7-bits, between 0.6V

and 1.0V V_{DD}, it was limited by linearity constraints from the voltage controlled oscillator. Future work could investigate the use of more linear inverter structures for implementation in the VCO, eliminating the need for linearizing amplifiers, preserving the opposing nature of the starving transistors, while increasing the linear operating region of the cells.

Active Inductor Receiver Equalizer

Future work on the active inductor receiver equalizer would involve silicon fabrication for testing at gigabit speed. The system proposed in this dissertation is based on simulation results and has not been tested post-silicon.

BIBLIOGRAPHY

[1] G.E. Moore. Cramming more components onto integrated circuits. *Proceedings of the IEEE*, 86(1):82 –85, Jan. 1998.

[2] G. Bell. Bell's law for the birth and death of computer classes. In *Communications of the ACM*, volume 51, pages 86 –94, 2008.

[3] A Computer Pioneer Rediscovered, 50 Years On. `http://www.nytimes.com/1994/04/20/news/20iht-zuse.html`.

[4] R. Rojas and U. Hashagen. *The First Computers:History and Architectures*. MIT Press, 2002.

[5] ENIAC Computer Specs. `http://www.eniaccompute.weebly.com/eniac-specs.html`.

[6] N. Mohamed and I. Jawhar. A fault tolerant wired/wireless sensor network architecture for monitoring pipeline infrastructures. In *International Conference on Sensor Technologies and Applications*, pages 179 –184, Aug. 2008.

[7] Y. Tachwali, H. Refai, and J. Fagan. Minimizing hvac energy consumption using a wireless sensor network. In *33rd Annual Conference of the IEEE Industrial Electronics Society*, pages 439 –444, Nov. 2007.

[8] N. Elvin, N. Lajnef, and A. Elvin. Feasibility of structural monitoring with vibration powered sensors. *Smart Materials and Structures*, 15, 2006.

[9] L. Schwiebert, S. Gupta, and J. WEinmann. Research challenges in wireless networks of biomedical sensors. In *International Conference on Mobile Computing and Networking*, pages 151 –165, 2001.

[10] J. Hill, M. Horton, R. Kling, and L. Krishnamurthy. The platforms enabling wireless sensor networks. In *Communications of the ACM*, volume 47, pages 41 –46, 2004.

[11] J. Yick, B. Mukherjee, and D. Ghosal. Wireless sensor network survey. In *Computer Networks*, volume 52, pages 2292 –2330, 2008.

[12] A history of microprocessor transistor count. `http://www.wagnercg.com/Portals/0/FunStuff/AHistoryofMicroprocessorTransistorCount.pdf`.

[13] The world's smallest computer. `http://www.computerhistory.org/atchm/the-worlds-smallest-computer/`.

[14] J. Lee and N.S. Kim. Optimizing throughput of power- and thermal- constrained multicore processors using dvfs and per-core power-gating. In *Design Automation Conference*, pages 47 –50, 2009.

[15] R. Rao, S. Vrudhula, and C. Chakrabarti. Throughput of multi-core processors under thermal constraints. In *International Symposium on Low Power Electronics and Design*, pages 201 –206, 2007.

[16] H. Esmaeilzadeh, E. Blem, R. St. Amant, K. Sankaralingam, and D. Burger. Dark silicon and the end of multicore scaling. In *IEEE Computer Society*, pages 122 –134, 2012.

[17] 40 years of microprocessor trend data. `https://www.karlrupp.net/2015/06/40-years-of-microprocessor-trend-data/`.

[18] N. Weste and D. Harris. *CMOS VLSI Design: A Circuits and Systems Perspective*. Pearson Prentice Hall, 4th edition, 2010.

[19] A. Mainwaring, J. PLastre, R. Szewczyk, D. Culler, and J. Anderson. Wireless sensor networks for habitat monitoring. In *Workshop on Wireless Sensor Networks and Applications*, pages 88 –97, 2002.

[20] T. He, S. Krishnamurthy, J.A. Stankovic, T. Abdelzaher, L. Luo, R. Stoleru, T. Yan, L. Gu, J. Hui, and B. Krogh. Energy-efficient surveillance system using wireless sensor networks. In *Conference on Mobile Systems, Applications, and Services*, pages 270 –283.

[21] C. Otto, A. Milenkovic, C. Sanders, and E. Jovanov. System architecture of a wireless body area sensor network for ubiquitous health monitoring. In *Journal of Mobile Multimedia*, pages 307 –326.

[22] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf. Deppface: Closing the gap to human-level performance in face verification. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1701 –1708, 2014.

[23] B. Brown, M. Chui, and J. Manyika. Are you ready for the era of 'big data'? In *McKinsey Quarterly*, pages 1 –12.

[24] Bringing Big Data to the Enterprise. `http://www-01.ibm.com/software/data/bigdata/what-is-big-data.html`.

[25] IDC Finds Growth, Consolidation, and Changing Ownership Patterns in Worldwide Datacenter Forecast. `http://www.idc.com/getdoc.jsp?containerId=prUS25237514`.

[26] C. Yadav, S. Wang, and M. Kuma. Algorithm and approaches to handle large data-a survey. *International Journal of Computer Science and Network*, 2(3):1701 –1708, 2013.

[27] V. Skorpil and J. Stastny. Comparison of learning algorithms. In *Biennial Symposium on Communications*, pages 231 –234, 2008.

[28] O. Russakovsky, J. Deng, H. Su, J. Krausse, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A.C. Berg, and L. Fei-Fei. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, pages 211 –252, 2015.

[29] W.S. Mcculloch. The brain computing machine. *Electrical Engineering*, 68(6):492 –497, June 1949.

[30] X.S. Zhang. *Neural Networks in Optimization*. Springer, 2000.

[31] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. In *ieee*, volume 86, pages 2278 –2324, 1998.

[32] Y. Bengio. Learning deep architectures for ai. In *Foundations and Trends in Machine Learning*, volume 2, pages 1 –127, 2009.

[33] L. Deng and D. Yu. Deep learning: Methods and applications. Technical report, 2014.

[34] J. Schmidhuber. Deep learning in neural networks: An overview. In *Neural Networks*, 2014.

[35] S.B. Kotsiantis. Supervised machine learning: A review of classification techniques. In *Emerging Artificial Intelligence Applications in Computer Engineering*, 2007.

[36] D.E. Rumelhart, G.E. Hinton, and R.J. Williams. Learning representations by back-propagating errors. *Cognitive Modeling*, pages 213 –222.

[37] A.J.R. Simpson. Dither is better than dropout for regularising deep neural networks. *arXiv preprint arXiv:1508.04826*, 2015.

[38] G.E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. In *Computing Research Repository (CoRR)*, 2012.

[39] I.V. Tetko, D.J. Livingstone, and A.I. Luik. Neural network studies: Comparison of overfitting and overtraining. *Journal of Chemical Information and Computer Sciences*, 35(5):826–833, 1995.

[40] S. Park, Y. Kim, B. Urgaonkar, J. Lee, and E. Seo. A comprehensive study of energy efficiency and performance of flash-based ssd. *Journal of Systems Architecture*, 2011.

[41] Comparison of Embedded Non-Volatile Memory Technologies. `http://www.kilopass.com/wp-content/uploads/2010/04/comparison\_of\_embedded\_nvm.pdf`.

[42] Efuse block: Description and simulation. `http://indico.cern.ch/event/91184/attachments/1094685/1561703/FEI4-efuse.pdf`.

[43] M. She. Semiconductor flash memory scaling. *University of California, Berkeley*, 2003.

[44] Advanced Digital Integrated Circuits: Embedded Memory - Flash. `http://bwrcs.eecs.berkeley.edu/Classes/icdesign/ee241\_s05/Lectures/Lecture26-Flash.pdf`.

[45] G.E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *CoRR*, 2012.

[46] G.E. Hinton, L. Deng, D. Yu, G. Dahl, A. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. Sainath, and B. Kingsbury. Deep neural networks for acoustic modeling in speech recognition. *Signal Processing Magazine*, 2012.

[47] S.S. Haykin. *Neural Networks and Learning Machines*. Pearson Prentice Hall, 3rd edition, 2009.

[48] IMAGENET Large Scale Visual Recognition Challenge 2015 (ILSVRC2015). `http://image-net.org/challenges/LSVRC/2015/`.

[49] M.B. Merki and D. Merki. *Glencoe Health A Guide to Wellness*. Glencoe, 2nd edition, 1989.

[50] F. Rosenblatt. The perceptron: a perceiving and recognizing automaton. *Report 85-460-1, Cornell Aeronautical Laboratory*, 1957.

[51] J. Misra and I. Saha. Artificial neural networks in hardware: A survey of two decades of progress. *Neurocomputing*, 74:239 –255, 2010.

[52] H.P. Graf, L.D. Jackel, R.E. Howard, B. Straughn, J.S. Denker, W. Hubbard, D.M. Tennant, and D. Schwartz. Vlsi implementation of a neural network memory with several hundreds of neurons. In *AIP Conference Proceedings on Neural Networks for Computing*, pages 182 –187, 1987.

[53] T. Morishita, Y. Tamura, T. Otsuki, and G. Kano. A bicmos analog neural network with dynamically updated weights. *IEICE Transactions on Electronics*, 75(3):297 –302, 1992.

[54] M. Holler, S. Tam, H. Castro, and R. Benson. An electrically trainable artificial neural network (etann) with 10240 "floating gate" synapses. In *IEEE Computer Society Neural Networks Technology Series*, pages 50 –55, 1990.

[55] Cypress SONOS Technology. `http://www.cypress.com/file/123341/download`.

[56] J. Seo, B. Brezzo, L. Yong, B.D. Parker, S.K. Esser, R.K. Montoye, B. Rajendran, J.A. Tierno, L. Chang, D.S. Modha, and D.J. Friedman. A 45nm cmos neuromorphic chip with a scalable architecture for learning in networks of spiking neurons. In *IEEE Custom Integrated Circuits Conference*, pages 1 –4, Sept. 2011.

[57] B. Moons and M. Verhelst. A 0.3-2.6 tops/w precision-scalable processor for real-time large-scale convnets. 2016.

[58] Y.H. Chen, T. Krishna, and J. Emer. Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks. In *Proc. International Solid-State Circuits Conference*, pages 262 –263, 2016.

[59] P.A. Merolla, J.V. Arthur, R. Alvarez-Icaza, A.S. Cassidy, J. Sawada, F. Akopyan, B.L. Jackson, N. Imam, C. Guo, Y. Nakamura, B. Brezzo, I. Vo, S.K. Esser, R. Appuswamy, B. Taba, A. Amir, M.D. Flickner, W.P. Risk, R. Manohar, and D.S. Modha. A million

spiking-neuron integrated circuit with a scalable communication network and interface. *Science*, 345(6197):668 –673, Aug. 2014.

[60] Cm1k hardware user's manual. `http://www.cognimem.com/_docs/Technical-Manuals/TM_CM1K_Hardware_Manual.pdf`.

[61] Jetson tx1 technical specifications. `http://www.nvidia.com/object/jetson-tx1-module.html`.

[62] Intel ark product specifications. `http://ark.intel.com/`.

[63] R.F. Thompson. *The Brain: A Neuroscience Primer*. Macmillan, 1st edition, 2000.

[64] ADC Performance Survey 1997-2015 (ISSCC and VLSI Symposium). `http://ark.intel.com/products/88195/Intel-Core-i7-6700K-Processor-8M-Cache-up-to-4\_20-GHz`.

[65] A.G. Andreou, K.A. Boahen, P.O. Pouliquen, A. Pavasovic, R.E. Jenkins, and K. Strohbehn. Current-mode subthreshold mos circuits for analog vlsi neural systems. *IEEE Transactions on Neural Networks*, 2(2):205 –213, March 1991.

[66] S.T. Kim, J. Choi, S. Beck, T. Song, K. Lim, and J. Laskar. Subthreshold current mode matrix determinant computation for analog signal processing. In *Proc. ISCAS*, pages 1260 –1263, May 2010.

[67] K.H. Abed. Cmos vlsi implementation of 16-bit logarithm and anti-logarithm converters. *IEEE Midwest Symposium on Circuits and Systems*, 2:776 –779, Aug. 2000.

[68] V.P. Hoang, X.T. Do, and C.K. Pham. An efficient asic implementation of logarithm approximation for hdr image processing. In *International Conference on Advanced Technologies for Communications*, pages 535 –539, 2013.

[69] Caffe: Deep Learning Framework by the BVLC. `http://caffe.berkeleyvision.org/`.

[70] L. Magnelli, F.A. Amoroso, F. Crupi, G. Cappuccino, and G. Iannaccone. Design of a 75-nw, 0.5-v subthreshold complementary metal-oxide-semiconductor operational amplifier. In *International Journal of Circuit Theory and Applications*, 2012.

[71] P. Corke, T. Wark, R. Jurdak, D. Moore, and P. Valencia. Environmental wireless sensor networks. *Proceedings of the IEEE*, 98(11):1903 –1917, Oct. 2010.

[72] Y. Lee, S. Bang, I. Lee, Y. Kim, G. Kim, P. Dutta, D. Sylvester, and D. Blaauw. A modular 1mm3 die-stacked sensing platform with low power i2c inter-die communication and multi-modal energy harvesting. *IEEE Journal of Solid-State Circuits*, 48(1):229 –243, January 2013.

[73] D.G. Vaughan, T. Asbury, and P. Riordan-Eva. *General Ophthalmology*. Appleton and Lange, 18th edition, 1999.

[74] G. Chen, M. Fojtik, D. Kim, D. Fick, J. Park, M. Seok, M. Chen, Z. Foo, D. Sylvester, and D. Blaauw. Millimeter-scale nearly perpetual sensor system with stacked battery and solar cells. In *Proc. International Solid-State Circuits Conference*, pages 288 –289, Feb. 2010.

[75] M. Suster, J. Guo, N. Chaimanonart, W. Ko, and D. Young. A wireless strain sensing microsystem with external rf power source and two-channel data telemetry capability. In *Proc. International Solid-State Circuits Conference*, pages 380 –381, Feb. 2007.

[76] LM134/LM234/LM334 3-Terminal Adjustable Current Sources. `http://www.ti.com/lit/ds/symlink/lm334.pdf`.

[77] N. Verma and A. Chandrakasan. An ultra low energy 12-bit rate-resolution scalable sar adc for wireless sensor nodes. *IEEE Journal of Solid-State Circuits*, 42(6):1196 –1205, June 2007.

[78] Y. Chae, I. Lee, and G. Han. A 0.7v 36w 85db-dr audio delta-sigma modulator using a class-c inverter. In *Proc. International Solid-State Circuits Conference*, pages 490 –491, Feb. 2008.

[79] N. Maghari and U-K. Moon. A third-order dt delta sigma modulator using noise-shaped bidirectional single-slope quantizer. In *Proc. International Solid-State Circuits Conference*, pages 474 –476, Feb. 2011.

[80] R. Veldhoven, R. Rutten, and L. Breems. An inverter-based hybrid delta sigma modulator. In *Proc. International Solid-State Circuits Conference*, pages 492 –493, Feb. 2008.

[81] S. Kwon and F. Maloberti. A 14mw multi-bit/spl delta sigma modulator with 82db snr and 86db dr for adsl2+. In *Proc. International Solid-State Circuits Conference*, pages 161 –170, Feb. 2006.

[82] J. Lin and B. Haroun. An embedded 0.8v/480w 6b/22mhz flash adc in 0.13m digital cmos process using nonlinear double-interpolation technique. In *Proc. International Solid-State Circuits Conference*, pages 244 –245, Feb. 2002.

[83] M. Choi and A. Abidi. A 6b 1.3gsamples/s a/d converter in 0.35m cmos. *IEEE Journal of Solid-State Circuits*, 36(12):1847 –1858, Dec. 2001.

[84] G. Van der Plas, S. Decoutere, and S. Donnay. A 0.16pj/conversion-step 2.5mw 1.25gs/s 4b adc in a 90nm digital cmos process. In *Proc. International Solid-State Circuits Conference*, pages 566 –567, Feb. 2006.

[85] S. Park, Y. Palaskas, and M. Flynn. A 4gs/s 4b flash adc in 0.18m cmos. In *Proc. International Solid-State Circuits Conference*, pages 2330 –2339, Feb. 2006.

[86] B. Verbruggen, M. Iriguchi, and J. Craninckx. A 1.7mw 11b 250ms/s 2 interleaved fully dynamic pipelined sar adc in 40nm digital cmos. In *Proc. International Solid-State Circuits Conference*, pages 466 –467, Feb. 2012.

[87] M. Yoshioka, M. Kudo, T. Mori, and S. Tsukamoto. A 0.8v 80ms/s 6.5mw pipelined adc with regulated overdrive voltage biasing. In *Proc. International Solid-State Circuits Conference*, pages 452 –453, Feb. 2007.

[88] M. Yip and A. Chandrakasan. A resolution-reconfigurable 5-to-10b 0.4-to-1v power scalable sar adc. In *Proc. International Solid-State Circuits Conference*, pages 190 –191, Feb. 2011.

[89] P. Clark, D. Johnson, M. Kniep, B. Huttash, A. Wood, M. Johnson, C. Mcgillivan, and K. Titus. An advanced, low cost, gps-based animal tracking system. In *Rangeland Ecology and Management*, pages 334 –340, 2006.

[90] N. Pinckney, K. Sewell, R.G. Dreslinski, D. Fick, T. Mudge, D. Sylvester, and D. Blaauw. Assessing the performance limits of parallelized near-threshold computing. In *Proc. Design Automation Conference*, pages 1143 –1148, June 2012.

[91] Y. Lien. A 4.5-mw 8-b 750-ms/s 2-b/step asynchronous subranged sar adc in 28-nm cmos technology. In *Symposium on VSLI Circuits*, pages 88 –89, June 2012.

[92] P. Cong, N. Chaimanonart, W. Ko, and D. Young. A wireless and batteryless 10-bit implantable blood pressure sensing microsystem with adaptive rf powering for real-time laboratory mice monitoring. *IEEE Journal of Solid-State Circuits*, 44(12):3631 –3644, Dec. 2009.

[93] G. Taylor and I. Galton. A mostly-digital variable-rate continuous-time delta-sigma modulator adc. *IEEE Journal of Solid-State Circuits*, 45(12):2634 –2646, Dec. 2010.

[94] ADC Performance Survey 1997-2015 (ISSCC and VLSI Symposium). `http://www.stanford.edu/~murmann/adcsurvey.html`.

[95] L. Kull, T. Toifl, M. Schmatz, P.A. Francese, C. Menolfi, M. Braendli, M. Kossel, T. Morf, T.M. Andersen, and Y. Leblebici. A 3.1mw 8b 1.2gs/s single-channel asynchronous sar adc with alternate comparators for enhanced speed in 32nm digital soi cmos. In *Proc. International Solid-State Circuits Conference*, pages 468 –469, Feb. 2013.

[96] H. Wei, C.-H. Chan, U.-F. Chio, S.-W. Sin, S.-P. U, R. Martins, and F. Maloberti. A $0.024mm^2$ 8b $400ms/s$ sar adc with 2b/cycle and resistive dac in 65nm cmos. In *Proc. International Solid-State Circuits Conference*, pages 188 –189, Feb. 2011.

[97] P. Harpe, Y. Zhang, G. Dolmans, K. Philips, and H.D. Groot. A 7-to-10b 0-to-40ms/s flexible sar adc with 6.5-to-16fj/conversion-step. In *Proc. International Solid-State Circuits Conference*, pages 472 –473, Feb. 2012.

[98] D. Schmidt. Circuit pack parameter estimation using rents rule. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 1(4):186 –192, Oct. 1982.

[99] P.K. Hanumolu, G.Y. Wei, and U.K. Moon. *Equalizers for High-Speed Serial Links*. World Scientific, 2005.

[100] L.N. Binh. *Digital Optical Communications*. CRC Press, 2008.

[101] J. Laskar, S. Chakraborty, M.M. Tentzeris, F. Bien, and A. Pham. *Advanced Integrated Communication Microsystems*. John Wiley & Sons, 2009.

[102] T.O. Dickson, J.F. Bulzacchelli, and D.J. Friedman. A 12-gb/s 11-mw half-rate sampled 5-tap decision feedback equalizer with current-integrating summers in 45-nm soi cmos technology. *IEEE Journal of Solid-State Circuits*, 44(4):1298 –1305, April 2009.

[103] J.W. Poulton, W.J. Dally, X. Chen, J.G. Eyles, T.H. Greer, S.G. Tell, and C.T. Gray. A 0.54pj/b 20gb/s ground-referenced single-ended short-haul serial link in 28nm cmos for advanced packaging applications. In *Proc. International Solid-State Circuits Conference*, pages 404 –405, Feb. 2013.

[104] J.W. Poulton, W.J. Dally, X. Chen, J.G. Eyles, T.H. Greer, S.G. Tell, J.M. Wilson, and C.T. Gray. A 0.54pj/b 20gb/s ground-referenced single-ended short-reach serial link in 28nm cmos for advanced packaging applications. *IEEE Journal of Solid-State Circuits*, 48(12):3206 –3218, Dec. 2013.

[105] F. Yuan. *CMOS Active Inductors and Transformers: Principle, Implementation and Applications*. Springer, 1st edition, 2008.

[106] H. Wang and J. Lee. A 21-gb/s 87-mw transceiver with ffe/dfe/analog equalizer in 65-nm cmos technology. *IEEE Journal of Solid-State Circuits*, 45(4):909 –920, April 2010.

[107] T. Toifl, C. Menolfi, M. Ruegg, R. Reutemann, D. Dreps, T. Beukema, A. Prati, D. Gardellini, M. Kossel, P. Buchmann, M. Brandli, P.A. Francese, and T. Morf. A 2.6mw/gbps 12.5gbps rx with 8-tap switched-capacitor dfe in 32nm cmos. *IEEE Journal of Solid-State Circuits*, 47(4):897 –910, April 2012.

[108] Myriad 2 Vision Processor. `http://uploads.movidius.com/1441734401-Myriad-2-product-brief.pdf`.