Analysis and reduction of mismatch in low power sub-threshold silicon neurons

Sun, Shuo

2012

Sun, S. (2012). Analysis and reduction of mismatch in low power sub-threshold silicon neurons. Master's thesis, Nanyang Technological University, Singapore.

<https://hdl.handle.net/10356/51101>

<https://doi.org/10.32657/10356/51101>

**ANALYSIS AND REDUCTION OF MISMATCH IN LOW
POWER SUB-THRESHOLD SILICON NEURONS**

SUN SHUO

School of Electrical & Electronic Engineering

A thesis submitted to the Nanyang Technological University

in partial fulfillment of the requirement for the degree of

Master of Engineering

2012

Acknowledgments

I'd like to thank my supervisor Assist Prof Arindam Basu for his guidance and support. His depth and breadth of knowledge is inspirational, and he is a gentleman whose reputation precedes him and whom it is a pleasure to be associated with.

Many thanks to members of VIRTUS lab for putting up with me during my two years master course. It has also been a pleasure working with Chen Yi, Li Fei, Dinup and Yao Enyi.

My research has been funded by EDB, to whom I am very grateful.

Table of Contents

Acknowledgments.....	i
Table of Contents	ii
Summary	v
List of Figures	vi
Chapter 1 Introduction.....	1
1.1 Motivation	1
1.2 Objectives.....	3
1.3 Contribution	4
1.4 Organization of the thesis.....	4
Chapter 2 Literature review	6
2.1 Biological neuron	6
2.2 Artificial neuron model	8
2.3 Spiking neuron model (I&F neuron)	10
2.4 Artificial neural network	11
2.5 Silicon neuron	13
2.6 A low power integrate and fire neuron circuit	16
2.7 The DPI neuron circuit with controllable threshold voltage	19
2.7.1 Differential pair integrator input stage.....	20
2.7.2 Spike frequency adaptation.....	21
2.7.3 Others.....	23
Chapter 3 Neuron mismatch analysis	24
3.1 MOSFET subthreshold characteristic	24

3.2	Mismatch properties of MOS transistor	28
3.2.1	Current mismatch.....	29
3.2.2	Voltage mismatch	31
3.3	Silicon neuron mismatch analysis	32
3.3.1	General Analysis	32
3.3.2	A specific neuron circuit mismatch analysis	34
3.4	Mismatch in silicon neuron application	38
3.4.1	A conventional image sensor	38
3.4.2	A spike-based vision sensor.....	39
3.4.3	Transistor mismatch effect on spike-based image sensor.....	40
Chapter 4	Proposed solutions to counter neuron mismatch	42
4.1	Floating gate transistor	42
4.2	Programming of floating gates	44
4.2.1	Electron tunneling.....	45
4.2.2	Electron injection	45
4.2.3	Floating gate selection circuits for programming	46
4.3	Lateral bipolar junction transistor	47
4.4	Proposed circuit.....	49
4.4.1	Analysis on low power silicon neuron circuit.....	50
4.4.2	Tuning out F-I curve mismatch.....	53
Chapter 5	Chip components	56
5.1	Address Event Representation	56

5.1.1	Description.....	57
5.1.2	Arbiter.....	58
5.2	Synapse.....	62
5.3	Bias generation.....	64
5.3.1	Constant Gm circuit.....	65
5.3.2	Current splitter.....	67
5.4	BJT current mirror.....	70
5.5	System architecture.....	72
5.6	Silicon neuron implementation of Extreme Learning Machines.....	75
Chapter 6	Conclusion and future work.....	79
6.1	Conclusion.....	79
6.2	Future work.....	80
Author's Publications.....		81
Reference.....		82

Summary

CMOS transistor mismatch is one of the factors limiting analog circuit performance. It is known that the variations of transistor size and threshold voltage (V_{TH}) are the two factors causing device mismatch. Threshold voltage mismatch effect is generally considered dominating over that of transistor size mismatch. This is especially true for transistors operating in subthreshold regime where terminal current is exponentially dependent on threshold voltage. Silicon neuron is an example of using transistors operating in subthreshold. Hardware implementation of neural network using silicon neurons is emerging in the past two decades and becomes popular in many applications. Unfortunately, neural network normally consists of a large number of identical silicon neurons, and threshold mismatch causes neuron circuits to behave differently across a neural chip. In this thesis, we analyze causes underlying silicon neuron mismatch, using sensitivity analysis to separate 'dominant' threshold voltages. Even though there are many different silicon neuron circuits in the literature, our general analysis on neuron mismatch using sensitivity analysis applies to all neuron designs. We propose to replace those 'dominant' transistors with some special devices, namely floating gate transistors and lateral BJT transistors. Using floating gate transistors to calibrate out neuron mismatch requires the use of Address Event Representation (AER) which is a very popular asynchronous communication protocol used in neural network communication. We illustrate our neuron mismatch correction concept using a popular integrate and fire silicon neuron. We have fabricated a chip using VIS 0.35um technology. The chip contains three different types of neuron array, synaptic circuits, and input/output AER interfacing circuits.

List of Figures

Figure 2.1: Neuron	6
Figure 2.2: Action Potential	7
Figure 2.3: Artificial Neuron Model	9
Figure 2.4: Schematic of Soma	11
Figure 2.5: (a) Feed forward neural network (b) Recursive neural network	13
Figure 2.6: Axon-Hillock circuit.....	14
Figure 2.7: Membrane and output node voltage waveforms	15
Figure 2.8: A low power I&F neuron	17
Figure 2.9: Neuron circuit transient response to a constant current injection	18
Figure 2.10: Typical I & F curve	19
Figure 2.11: DPI neuron with controllable threshold	20
Figure 2.12: Frequency adaptation	22
Figure 3.1: NMOS I_{ds} vs V_{gs} in VIS 0.35um	27
Figure 3.2: Subthreshold Saturation region	27
Figure 3.3: Monte Carlo analysis	35
Figure 3.4: Critical transistor threshold mismatch effect on F-I curve.....	37
Figure 3.5: Non-Critical transistor threshold mismatch effect on F-I curve.....	37
Figure 3.6: Frame-based image sensor working principle.....	38
Figure 3.7: Spiking image sensor structure.....	39
Figure 3.8: Grey code image.....	41
Figure 3.9: 4x4 neuron sensor firing frequencies	41
Figure 4.1: Layout, cross-section, and circuit diagram of the FG MOSFET.....	43
Figure 4.2: Electron tunneling	45
Figure 4.3: Electron injection	46
Figure 4.4: Selection circuit for FG programming	47
Figure 4.5: Lateral PNP BJT Layout	48
Figure 4.6: Lateral PNP BJT cross section view	48
Figure 4.7: Proposed neuron circuit.....	52
Figure 4.8: Mismatch correction.....	54
Figure 5.1: Dedicated point to point connection through AER	57
Figure 5.2: The asynchronous handshake	57
Figure 5.3: Arbiter tree	59

Figure 5.4: Arbiter element	60
Figure 5.5: Arbiter element processing illustration	61
Figure 5.6: Log domain synapse circuit.....	63
Figure 5.7: Switched capacitor	63
Figure 5.8: Bias generator architecture	64
Figure 5.9: Bootstrap Circuit	65
Figure 5.10: Current splitter.....	68
Figure 5.11: R-2R ladder	69
Figure 5.12: A section of current splitter circuit.....	69
Figure 5.13: Output current versus Bit position.....	69
Figure 5.14: BJT current mirror.....	70
Figure 5.15: A large number of load.....	71
Figure 5.16: Improved current mirror	71
Figure 5.17: System architecture	72
Figure 5.18: Chip layout	72
Figure 5.19: Schematic and symbol for AER transmitter.....	74
Figure 5.20: Schematic and symbol for AER receiver	74
Figure 5.21: A SLFN with random hidden nodes 1,2,...,L and Output weights β_1 , β_2 ,... , β_L determined by ELM.....	75
Figure 5.22: Architecture for implementing the ELM.....	76
Figure 5.23: The weighted summation for the ELM output layer	77

Chapter 1 Introduction

1.1 Motivation

The human brain exhibits many amazing properties, for example compactness, energy efficiency, and fault tolerance. It consists of a huge number of neurons ($\approx 10^{11}$) and interconnects ($\approx 10^{14}$) called synapses of diverse types, operating on analog principles of the physics of neural computation that are fundamentally different from digital principles in traditional computing. Inspired by these properties, scientists are trying to understand and mimic the brain through software simulations and through hardware emulations[1-4].

Brain simulations are difficult because computers operate sequentially, with many processing units executing instructions step by step, while the brain operates in parallel, with highly interconnected neurons processing information distributed throughout the network. Digital tools and simulators are convenient and practical for exploring the quantitative behaviour of neural networks. However they are not ideal for implementing real-time behaving systems, or detailed large-scale simulations of neural systems. Even the largest supercomputing systems to date are not capable of obtaining real-time performance when running simulations large enough to accommodate multiple cortical areas, yet detailed enough to include distinct cellular properties. Thus, the only possibility to get a significant gain in simulation speed within the current decade is parallelization of dedicated analog circuits.

Neuromorphic engineers are implementing neurons and synapses using hybrid silicon analog/digital circuits, and trying to build brain like artificial system which responds in real-time and operates in parallel. The term “neuromorphic” was coined by Carver

Mead in the late eighties to refer to artificial neural systems whose architecture and design principles are based on those of biological nervous systems[5]. Neuromorphic engineers are hoping to create artificial systems that can match the human brain in pattern recognition, hearing, vision, and learning tasks[6-12]. By following this design philosophy, they have been able to create practical devices that operate in real time and consume less power than standard digital design. For example, some ‘intelligent’ systems built on neuromorphic sensory devices emulating the retina[13, 14] and cochlea[15-17] can interact with its environment in real-time.

Silicon neuron circuits represent one of the main building blocks for implementing neuromorphic systems and they are hybrid analog/digital very large scale integration (VLSI) circuits that emulate the electrophysiological behavior of real neurons. A particular application example of the silicon neuron circuit which will be discussed in chapter 3 is image transmission and processing. A brief discussion is given here. In traditional image processing system, operation is performed frame by frame and a video is just a sequence of frames. However, in spike-based image processing system which is based on human vision system, information transmission and processing is done through a few layers of neurons [18-20]. Information (intensity, contrast, motion etc) is captured by the spiking image sensor and is translated into spikes which travel quickly through a sequence of following layers for complicated image processing. This format allows fast information identification and processing, without waiting to process complete image frames.

Although neuromorphic VLSI systems have been reported fast and powerful information sensing and processing capabilities, there are not many specific neuromorphic products in the market. One of the main reasons for this is the unavoidable excessive mismatch. The device mismatch severely affects system

functionality and is most pronounced in sub-threshold circuit design [21-23], for example silicon neural network [24-29]. The effect of statistical variations is exacerbated because of the exponential dependence of drain current on threshold voltage (V_{th}) differences. Typical analog design approaches consider only random mismatch assuming that systematic mismatch due to process gradients can be mitigated by techniques like common-centroid layout[30]. However, in arrays of neurons, systematic mismatch also dominates increasing the effect of variations even more.

A possible way to reduce circuit mismatch without increasing transistor sizes or operating currents is calibration. In this report, a calibration technique based on floating-gate MOS transistors in standard CMOS processes is presented, which can effectively reduces circuit mismatch while achieving circuit compactness.

1.2 Objectives

The objectives are listed as follows:

1. A neuron chip normally consists of a large number of silicon neurons, for example, our neuron chip containing 3x128 neurons. Thus to restrain the chip total power consumption to a practical value, neuron circuit must be of low power and small silicon area. We have to design a neuron circuit whose transistors operate in subthreshold, and there are actually some neuron circuits operating in above threshold region. We use miller MOS capacitor whenever possible to reduce circuit overall size.
2. The neuron circuit has to not only model the basic function of a spiking neuron, which is integrating input currents and generating output spikes with frequency proportional to the input current magnitude, but also model some other

behaviors of a biological neuron like leak current, refractory period and spike frequency adaptation.

3. Neurons encode the input analog information (current) into output spikes frequency, which is similar to a Sigma Delta ADC. Thus to reduce neuron mismatch or maintain neuron resolution, we need to use special transistors in our neuron circuit. For example, we can tune a neuron performance if the neuron circuit has some floating gate transistors.
4. Our neuron needs to be configurable. We need to have some control over neuron behaviors. For example, we can digitally control neuron leak current to be high or low, or control its refractory period to be short or long.
5. The chip we designed needs to be reconfigurable to accommodate different applications. Thus we have to explore the Address Event Representation which is a very popular neural network communication protocol.

1.3 Contribution

1. This thesis provides a general analysis on how transistor mismatch affect neuron circuits performance which is applicable for the analysis of any neuron circuit.
2. A systematic approach to reduce neuron mismatch by using floating gate transistor is presented.
3. A silicon neuron circuit is proposed and is used to build a programmable neuron network.

1.4 Organization of the thesis

Due to the diverse subject areas which this thesis covers, each chapter begins with a literature review relevant to the contents of that chapter. The detailed chapter by chapter structure is as follows.

In chapter 2, the literature on biological neuron, artificial neuron model and silicon neuron are reviewed. Silicon neuron especially integrate and fire neuron (I&F neuron) is the focus of this thesis. As a starting point, the simple but very influential "Axon-hillock" circuit is presented and is analyzed in detail to provide fundamental knowledge on neuron circuits. Then two other modern popular silicon neuron designs including a low power I-F neuron and a DPI neuron are introduced, analyzed, and will be used in the later chapters.

In chapter 3, the literature on subthreshold characteristic of MOS transistor and matching properties of MOS transistors are first reviewed. The performance of neuron circuit under MOS transistor threshold V_{th} mismatch is analyzed. This is done by Monte Carlo analysis to find variance in neuron firing rates. Contribution of individual transistor mismatch to this total variance is then studied by modifying V_{th} of transistors one at a time. This helps us identify critical transistors which can be calibrated. At the end of this chapter, one neuron application is presented to illustrate circuit mismatch problem.

In chapter 4, the literature on floating gate transistor as well as lateral BJT are reviewed. Proposed method of reducing mismatch by using floating gate transistor is analyzed. Neuron performance improvement by using this method is presented. The mismatch correction procedures of the sample neuron circuit are described.

In chapter 5, the complete testing system is presented, including system level architecture and chip layout. The literature on Address Event Representation (AER) is reviewed. Component circuits like synapse circuit, bias generation circuit, beta helper BJT current mirror are studied.

Finally the project is summarized, concluded and future work is proposed in chapter 6.

Chapter 2 Literature review

2.1 Biological neuron

There are approximately 10 billion neurons in the human cortex. Biological neuron uses electrical and chemical reactions to receive, process and transmit information between each other, and each neuron is connected to several thousands of other neurons to form neural networks.

Figure 2.1: Neuron

A typical neuron shown in Figure 2.1, is divided into three major parts: a soma, a single axon and dendrites. The soma is also known as the cell body. Dendrites are treelike extensions from the neuron body that help increase the surface area of the cell body and receive information from other neurons and transmit electrical stimulation to the soma. The signals from the dendrites are joined at soma. A single axon extends from the cell body at a junction termed axon hillock. The axon is the elongated fiber

that extends from the cell body to the terminal endings and transmits the neural signal. Some axons are covered with a fatty substance called myelin that acts as an insulator. At the end of axon, each terminal button is connected to another neuron across a small gap called a synapse where presynaptic neuron releases neurotransmitter chemicals and the postsynaptic neuron's receptors accept the chemicals.

Neuron membrane is able to selectively reject and absorb ions in the intracellular fluid by means of ion pumps with ion channels embedded in the membrane to generate intracellular versus extracellular concentration differences of ions such as sodium, potassium, chloride, and calcium. Thus there is a voltage gradient across the neuron membrane. In equilibrium the outside potential of a neuron cell is observed to be 70 mV higher than interior potential.

Figure 2.2: Action Potential

Synapses play an important role in the transmission of neural signal to other neurons. A neuron receives inputs from a large number of neurons via its synaptic connections. There are two types of synapses: inhibitory or excitatory synapses which either decrease (hyperpolarize) or increase (depolarize) activity in the target neuron. If

enough depolarization accumulates to bring the membrane potential at the axon hillock up to its threshold, the neuron will fire a signal (known as an action potential) which is a brief pulse originated at the soma and propagating rapidly along the axon. The action potential then activates the synapses at the end of axon. Once an action potential has been generated, the neuron is incapable of being re-excited for some time called refractory period. A typical action potential waveform is shown in Figure 2.2.

2.2 Artificial neuron model

As just mentioned above, the signal transmission between neurons through synapses involves releasing specific transmitter substances from the sending neuron, raising or lowering the receiving cell body electrical potential, receiver neuron firing if its body potential reaches a threshold. This signal transmission characteristic occurring in a biological neuron is what the artificial neuron model attempts to reproduce. The first artificial neuron was proposed by McCulloch and Pitts in 1943[31].

Artificial neuron is the basic computational element in an artificial neural network and is often called a node. It receives inputs from one or more other units, or even from an external source and sums the inputs to produce an output. Each input has an associated weight w , which can be modified so as to model synaptic connection with learning capability. Usually the sum of the weighted inputs passes through a non-linear function known as a transfer function or activation function which is usually a sigmoid function. A graphical presentation of the most widely used neuron model is given in Figure 2.3 below.

Figure 2.3: Artificial Neuron Model

The artificial neuron shown in Figure 2.3 has N inputs: $u_1, u_2, \dots, u_N$. Each input connects to the neuron through a connection having a weight denoted as $w_1, w_2, \dots, w_N$ respectively. Weights w_i and inputs u_i are real values. The input represents a temporal average of input pulses. A positive weight value indicates an excitatory connection while a negative value indicates an inhibitory connection. The threshold of this artificial neuron is denoted as θ . The activation corresponding to these N inputs is given by:

$$a = \left(\sum_{j=1}^N w_j u_j \right) + \theta$$

By assuming an imaginary input $u_0 = 1$ with a corresponding connection weight $w_0 = \theta$, the threshold value can be combined with other inputs in the activation formula. Thus the activation becomes:

$$a = \left(\sum_{j=0}^N w_j u_j \right)$$

This activation is feed into the activation function which generates a output:

$$x = f(a)$$

The activation function f usually is a sigmoid function, but it may also take the form of step functions, piecewise linear functions, or other non-linear functions. Though the artificial neuron structure looks simple, in principle an assembly of artificial neurons can perform any computations that an ordinary digital computer can [32-37].

2.3 Spiking neuron model (I&F neuron)

Recent years experimental evidence has suggested that information may be coded in synchronization and oscillations of populations of neurons, or in the precise timing of the neural pulses[38, 39]. A new category of neuron models has been introduced called spiking neural networks (SNN) which capture the spiking nature of the neurons and are more computably powerful and bio-realistic[40]. Two main classes of spiking neuron are conductance based neuron model[41] and integrate and fire neuron model[42]. Detailed conductance based neuron models, can reproduce electrophysiological measurements to a high degree of accuracy. Unfortunately, these models are usually difficult to analyze and are computationally expensive in numerical implementations and are inefficient for the implementation of a relatively big network because of their intrinsic complexity. For this reason, we only focus on one of the simple phenomenological spiking neuron models named integrate and fire (I&F) model which is very popular and has been used to discuss aspects of neural coding, network dynamics etc.

Unlike artificial neural networks which rely on a temporal average over the pulses, integrate and fire and similar spiking neuron models take into account the pulsed nature of the neuronal signal and consider spikes as potentially relevant for coding and information processing. Pulses are treated as formal events. The integrate and fire

neuron model describes the state of a neuron in terms of its membrane potential, which is determined by the synaptic inputs and the injected current that the neuron receives. When the membrane potential reaches a threshold, an action potential (spike) is generated. The schematic version of the leaky integrate and fire neuron is shown in Figure 2.4.

Figure 2.4: Schematic of Soma

Mathematically the leaky integrate and fire neuron are represented by the following

$$\tau_m \frac{\partial u}{\partial t} = -u(t) + RI(t)$$

with τ_m being the membrane time constant. Once the voltage over the capacitor goes above threshold value ϑ the neuron sends out a pulse. It has been shown that I&F neuron networks show a broad range of interesting and powerful computational properties, e.g. segmentation, onset detection, input prediction, pattern recognition etc[43].

2.4 Artificial neural network

An Artificial Neural Network (ANN) is a mathematical model that is inspired by the functional aspects of biological neural networks. It is composed of a large number of highly interconnected artificial neurons working in unison to solve specific problems

such as pattern recognition or data classification. And it processes information using a connectionist approach to computation. A very important property of neural networks is their adaptive nature where it changes structure based on external or internal information that flows through the network during the learning phase which involving adjustments to the synaptic connections that exist between neurons.

Figure 2.5(a) is probably the most common type of artificial neural network. It consists of three layers of artificial neurons: an 'input' layer, a 'hidden' layer and an 'output' layer. The input nodes take in information, e.g. image or sound. Information is passed through the hidden layers, until eventually the output nodes. Based on the pattern of connections between the units and the propagation of data, different classes of network architectures can be identified. Figure 2.5(a) is an example of feed forward neural network, where the data flow from input to output units is strictly feed forward and no feedback connections allowed. Figure 2.5(b) is an example of recurrent neural networks in which connections to the neurons on the previous layer or on the same layers are allowed. If a network is properly trained either by changing its connection weights according to some learning rule or by explicitly setting the weights using a priori knowledge, the neural network can be used to predict the outcome of new independent input data.

Artificial Neural Network emulates the learning process of human brain and can process problems involving non-linear and complex data even if the data are imprecise and noisy. It has been used for a variety of applications [44-46], for example, signal processing, speech or vision recognition and robotics. The most common ANN application is perhaps the classification or prediction scenario in which the network is trained to classify certain patterns into certain groups, and then is used to classify novel patterns.

Figure 2.5: (a) Feed forward neural network (b) Recursive neural network

2.5 Silicon neuron

Neuromorphic engineering aims to reproduce the spike-based computation of the brain by morphing its anatomy and physiology into custom silicon chips. By following this design philosophy, they have been able to create practical devices that operate in real time and are extremely useful for a wide range of applications, from high-speed modeling of large-scale neural systems to real-time behaving systems, to bidirectional brain–machine interfaces. The basic unit of these chips is the silicon neuron, designed using analog circuits for spike generation and digital ones for spike communication. Analog very large scale integrated silicon implementations of neurons was initially proposed by Carver Mead[47]. There are a wide range of neuromorphic silicon neurons, ranging in complexity from simple yet versatile leaky integrate and fire neurons to biophysically realistic conductance-based Hodgkin–Huxley models.

Shown in Figure 2.6 is the first simple VLSI version of integrate and fire neuron called Axon-hillock circuit which was proposed by Carver Mead in the late 1980s. Although it is simple in structure, it captures the basic operation principle of a biological neuron.

Figure 2.6: Axon-Hillock circuit

Shown in Figure 2.6, the capacitor C_m represents a neuron's membrane capacitance and is in charge of integrating postsynapse current input I_{in} to the neuron. Capacitor C_m and C_{fb} form a positive feedback structure which is the key to neuron spike generation. Amplifier A simply consists of two cascade inverters, and it behaves as a comparator which compares V_{mem} with the switching threshold (V_{TH}) of the first inverter inside it. Transistor M1 and M2 are used to reset V_{mem} . When V_{mem} rises above V_{TH} or the spiking threshold, a pulse is generated at the output node V_{out} . The pulse width and the rate at which membrane potential V_{mem} is decreasing depends on the resetting current controlled by V_{pw} . Assuming a constant current injecting into the neuron, the operation of the axon-hillock circuit is illustrated in Figure 2.7.

Figure 2.7: Membrane and output node voltage waveforms

Shown in Figure 2.7, solid line represents membrane node voltage V_{mem} and dash line denotes output node voltage V_{out} . The key concept in this neuron operation is that when V_{mem} just crosses V_{TH} , V_{out} changes almost instantaneously from 0 to V_{DD} or from V_{DD} to 0 and because of the positive feedback formed by C_{fb} and C_m , there is a ΔV_{mem} change on the membrane potential. Figure 2.7 can be further explicitly explained by the following equations with t_L denoting charging phase and t_H denoting discharging phase.

$$\Delta V_{mem} = \frac{C_{fb}}{C_m + C_{fb}} * V_{dd}$$

$$t_L = \frac{C_{fb} + C_m}{I_{in}} * \Delta V_{mem} = \frac{C_{fb}}{I_{in}} * V_{dd}$$

$$t_H = \frac{C_{fb} + C_m}{I_r - I_{in}} * \Delta V_{mem} = \frac{C_{fb}}{I_r - I_{in}} * V_{dd}$$

The Axon–Hillock circuit consists only two capacitors and six transistors, thus it is very compact. However it has a few drawbacks. First, as it can be seen from Figure 2.7,

the membrane potential changes slowly in the order of milliseconds. When V_{mem} is around V_{TH} , the first inverter conducts a short-circuit current and it dissipates significant amounts of power. Second, the Axon–Hillock circuit cannot reproduce many dynamic behaviours observed in real neurons such as refractory period mechanisms and spike-frequency adaptation properties. Lastly the spiking threshold voltage (the switching threshold of the inverter) depends only on CMOS process parameters.

2.6 A low power integrate and fire neuron circuit

Based on the axon-hillock circuit, several more elaborate neuron models with additional characteristics, such as spike-frequency adaptation properties and refractory period mechanisms, have been designed. These silicon neurons try to reproduce many real neuron behaviours, and at the same time achieve compactness and low-power. A compromise exists between neuron overall size and complexity. Because the invention of Address Event Representation allows large number of neurons operating in parallel, a neural chip achieves good power performance only if neuron circuit consumes minimal power locally. One of these low-power and more elaborate I&F neuron circuit is shown in Figure 2.8, which was originally proposed by Indiveri in 2006[48].

Shown in Figure 2.8, capacitor C_0 models the neuron membrane capacitor. Transistor M13 biased by V_{leak} models the membrane leakage current. In the absence of input current, the membrane potential will be discharged to 0V which is also the resting potential. Transistors M1 through M5 form a inverter with positive feedback which helps reducing the circuit's power consumption. Inverter formed by M10 and M11 is used to produce AER handshaking signal Req . AER handshaking signal Ack_bar is

feed to the gate of M8, and M9 biased by V_{rfr} controls the discharging rate at node V_{o2} which in turn sets arbitrary refractory periods.

Figure 2.8: A low power I&F neuron

The operation of this neuron circuit is discussed as follows:

1. The postsynapse current denoted I_{in} charges the membrane capacitor C_o , increases V_{mem} if it is an excitatory current, or discharges V_{mem} if it is an inhibitory input current. The injection currents range from tens of picoamperes to a few nanoamperes.
2. As V_{mem} voltage increases and approaches the first inverter's switching threshold voltage (V_{TH}), the feedback current flowing through M4 & M5 helps charging C_o , and V_{mem} increases more rapidly. Thus the time when M1 & M2 conducts a short circuit current is significantly reduced and this circuit achieves low power dissipation.

3. When V_{mem} is sufficiently high to switch the first inverter, Req voltage is driven to V_{dd} , which indicates the start of information transmission.
4. If Ack_bar which is the output of an AER receiver circuit is low, parasitic capacitor at the gate of reset transistor M12 is almost instantaneously charged to V_{dd} , and V_{mem} is quickly discharged to ground by M12. Output Req signal is then reset to zero. A complete pulse is thus generated.
5. When V_{mem} is discharged to ground, the first inverter output is driven back to V_{dd} , and M7 is turned on. The parasitic capacitor on node V_{o2} is then discharged by the current controlled by V_{ifr} . If this current is small, the voltage V_{o2} will decrease only slowly, and, as long as this voltage stays high enough to allow M12 to conduct a current higher than the input current, the membrane is discharged. Therefore V_{ifr} controls a time period called 'refractory period' during which the neuron cannot generate a spike. The refractory period mechanism is used to limit the maximum possible firing rate of a neuron.

Figure 2.9: Neuron circuit transient response to a constant current injection

The response properties of this low power I&F neuron is tested by injecting a constant current to the neuron and measuring its output. Shown in Figure 2.9 is the neuron circuit response to a constant current injecting the membrane capacitor C_o . The neuron circuit generates a pulse when V_{mem} rises above a switching threshold voltage which is around 500mV. Plotting output Req frequencies versus injection currents, we get a typical I & F curve shown in Figure 2.10, which is a very important diagram for analyzing neuron circuit characteristics. The I & F curve is frequently used when analyzing neuron circuit mismatch in Chapter 3.

Figure 2.10: Typical I & F curve

2.7 The DPI neuron circuit with controllable threshold voltage

The Axon-Hillock circuit produces a spike event when the membrane voltage rises above a voltage threshold that depends on the geometry of the transistors and on the VLSI process characteristics. In order to have better control over the spiking threshold, it is possible to use a comparator to define the spiking threshold voltage. Shown in Figure 2.11, the gm block which is simply a 9 transistors OTA behaves like a

[illegible]

2.7.1 Differential pair integrator input stage

20

Assuming the input stage transistors M1, M2 and M3 operate in subthreshold regime, the behaviour of the neuron can be derived analytically. The MOSET operation in subthreshold region is briefly discussed here and will be discussed in detail in chapter 3. In subthreshold, the drain current and gate voltage relationship of a saturated NMOS is given by

$$I_{DS} = I_o e^{\frac{kV_{GS}}{U_T}}$$

where I_o is the leakage current, U_T is the thermal voltage and k is the subthreshold slope factor. The current I_{SD2} which is charging C_{mem} can be derived to be:

$$\begin{aligned} I_{SD2} &= I_{in} \frac{e^{\frac{-kV_{mem}}{U_T}}}{e^{\frac{-kV_{mem}}{U_T}} + e^{\frac{-kV_{thr}}{U_T}}} \\ &= I_{in} \frac{1}{1 + e^{\frac{k(V_{mem} - V_{thr})}{U_T}}} \end{aligned}$$

Ignoring the adaptation current I_{DS10} and the feedback current I_{DS13} , apply Kirchhoff's current law on node v_{mem} :

$$\begin{aligned} C_{mem} \frac{d}{dt} V_{mem} &= I_{SD2} - I_{leak} \\ &= I_{in} \frac{1}{1 + e^{\frac{k(V_{mem} - V_{thr})}{U_T}}} - I_{leak} \end{aligned}$$

2.7.2 Spike frequency adaptation

The observation in a wide variety of neural systems that neurons respond to a current step with a rapid rise in spike frequency followed by a gradual decline to a new steady level is termed spike frequency adaptation. This mechanism is very important in neural

information processing. It can be used to reduce power consumption and bandwidth usage in VLSI systems comprising networks of silicon neurons. The way this neuron implementing this mechanism is to introduce a current increasing with spiking frequency and subtracted from the input current at membrane node.

Transistors responsible for spike frequency adaptation include four PMOS M5-M8, two NMOS M9 & M10 and a capacitor C_a . During a spike generation at output, when Req is high and Req_n is low, frequency adaptation circuit is activated and a fraction of the current controlled by v_w is sourced into the capacitor C_a . Thus, the voltage across the capacitor C_a increases with every spike. M9 biased by $vlkahp$ conducting a fixed leak current decreases the voltage across C_a when there is no spiking activity. As the voltage across C_a increases, it can be proven that the adaptation current conducted by M10 is exponentially increased and subtracted from the input. As a result, this neuron circuit has a fast spiking frequency initially and gradually decay towards a lower frequency. Figure 2.12 shows the neuron spike frequency adaptation in response to a constant input current.

Figure 2.12: Frequency adaptation

2.7.3 Others

Just like the low power I & F neuron shown in Figure 2.8, this neuron circuit makes use of a positive feedback around the op amp to create a positive up swing on the membrane potential when membrane voltage raises above the explicitly set spiking threshold v_{th} . There is another positive feedback around inverter 'INV0' to assist the inverter fast switch off, which in turn ensure a fast switch off of the positive feedback current I_{SD14} when voltage v_{mem} is reset to ground. The power dissipation of this circuit depends on the bias current of the gm block as well as the followed inverter. The neuron power dissipation has to be low, therefore the bias current of gm block has to be low like a few nanoamperes. It should be noted that the slew rate at the output node of the gm block is limited by this bias current.

This neuron also incorporates the address event representation like the low power I&F neuron. Request signal is generated at the output of 'INV2'. Transistors M15-18 and M4 control the refractory period.

Chapter 3 Neuron mismatch analysis

Having studied two different silicon I&F neuron circuits in chapter 2, in this chapter we will study the MOSFET transistor mismatch properties, neuron mismatch behaviours, and our proposed solutions to reduce neuron mismatch. To let the reader appreciate neuron mismatch analysis, we use a real case neuron application example to illustrate the problem of neuron mismatch. Since MOSFET transistors in neuron circuits are operating in subthreshold region, we will first study MOSFET transistors subthreshold characteristics.

3.1 MOSFET subthreshold characteristic

When V_{GS} is less than V_T , MOSFET is operating in the weak inversion region which is also said to be operating in the subthreshold region[49]. There are basically two reasons for using MOSFET transistors in subthreshold domain to design the neuron system. First of all, in subthreshold region the MOSFET source-drain current is exponentially related to the gate voltage which is similar to the relation between a synapse current and the voltage across the presynaptic membrane. Secondly, MOSFET operating in subthreshold domain draws much small current so power consumption is low. Delivering the necessary level of computational power is a pre-requisite to building a large scale neural network. Because the drain current is exponentially related to the gate source voltage and threshold voltage, any mismatch in these voltages can cause significant differences in the drain current.

We now analyze MOSFET subthreshold characteristic using NMOS as a example. Increasing the gate voltage increases the positive charge on the gate. This charge repels the holes in the substrate and leaves behind negatively charged ions, that balance out the gate charge. There is also a very thin layer of electrons beneath the

gate (the inversion layer). In subthreshold, we ignore the charge from the inversion layer because it is almost negligible compared with the depletion charge. At the source end of the channel the electron density is given by

$$N_s = N_o e^{-(\theta_o - q(\psi_s - V_s))/kT}$$

where N_o is the effective number of states per unit area in the channel, ψ_s is the surface potential, θ_o is the built-in energy barrier between the n^+ drain/source and the p^- substrate, and V_s and V_d are the source and drain potential respectively. At the drain end of the channel

$$N_d = N_o e^{-(\theta_o - q(\psi_s - V_d))/kT}$$

These equations are obtained by solving the boundary conditions of the energy states at the boundaries where the channel meets the source and drain depletion regions. Because the barrier at the drain is higher than that at the source, the electron concentration at the drain end of the channel is lower than at the source end. The concentration gradient leads to the diffusion of electrons from the source to the drain. The electrons that diffuse to the drain end of the channel are swept into the drain by the electric field in the depletion region around the drain. The electrons in the channel form an inversion layer. We can compute the current in the transistor using the electron diffusion current density equation

$$\begin{aligned} I &= J_{n,diff} W t \\ &= -q W t D_n \frac{dN}{dz} \end{aligned}$$

where $J_{n,diff}$ is the electron diffusion current density, W is the width of the channel, t is the depth of the channel, D_n is the diffusion coefficient of the electrons, and $\frac{dN}{dz}$ is the concentration gradient across the channel.

By assuming a small change around an operating point ψ_o , we can express ψ_s as

$$\psi_s = \psi_o + kV_g$$

where k is the capacitive coupling ratio from gate to channel and given by

$$k = \frac{C_{ox}}{C_{ox} + C_d}$$

where C_{ox} is the capacitance of the gate oxide per unit area, and C_d is the incremental capacitance of the depletion layer per unit area. The depletion capacitance is a non-linear function of the gate-to-bulk voltage, V_{gb} . As V_{gb} increases, the depletion width increases slowly; therefore C_d decreases, which in turn leads to an increase in k . We can write the current voltage characteristics for an NMOS in terms of the gate, source, and drain voltages which are referenced to the bulk voltage as

$$I = I_o e^{\frac{kV_g}{U_T}} (e^{\frac{-V_s}{U_T}} - e^{-V_d/U_T})$$

We can further express $I_o = \alpha \frac{W}{L} e^{\frac{kV_T}{U_T}}$, where W/L is the aspect ratio, α is a process dependent parameter and V_T is the threshold voltage. It is this exponential relationship between V_T and I which makes transistors in subthreshold prone to V_T mismatch.

Above equation can be separated into the forward current, I_f ; and the reverse current, I_r ; so that it can be rewritten as

$$I = I_f - I_r$$

These component currents are

$$I_f = I_o e^{\frac{kV_g - V_s}{U_T}}$$

$$I_r = I_o e^{\frac{kV_g - V_d}{U_T}}$$

The forward component I_f depends only on V_g and V_s , while the reverse component I_r depends only on V_g and V_d . If $V_{DS} > 4U_t$, the transistor is said to operate in the saturation regime:

$$I = I_o e^{\frac{kV_g - V_s}{U_T}}$$

Figure 3.1: NMOS I_{ds} vs V_{gs} in VIS 0.35um

Figure 3.2: Subthreshold Saturation region

3.2 Mismatch properties of MOS transistor

Matching is the key for analog integrated circuits to obtain their precision and performance. However, in reality, due to variations between lots, wafers, dies or even on a single die nominally identical and ratioed devices suffer from parameter variation, known as mismatch. Mismatch can be broke into random mismatch and systematic mismatch[30]. In neural network design which normally involves arrays of neuron circuit spreading across a entire chip, both random mismatch and systematic mismatch are important and proper design techniques have to make sure a neuron not only matches with its neighborhoods but also neurons far away from it.

It is convenient to think of the systematic and random as mismatch large signal and small signal variations. Random mismatches stem from stochastic nature of many physical processes which result in dimensions, dopings, oxide thicknesses, and other component parameters values variations. Although these statistical fluctuations cannot be entirely eliminated, their impact can be minimized through proper selection of device dimensions and component value. Systematic mismatches stem from process biases, nonuniform current flow, diffusion interactions, contact resistances, temperature gradients, mechanical stresses, and other causes. In general, if devices are placed close to each other, systematic mismatch is negligible in comparison to random mismatch and if devices are separated far apart systematic mismatch dominates. A major goal of designing matched components consists of rendering them insensitive to various sources of systematic errors.

Mismatch in device parameters can be modeled by using different techniques. Several researchers has extensively studied device mismatch between two geometrically identical MOS transistors[23, 50]. It has been observed experimentally that there are

two dominant sources underlying device mismatch: the threshold voltage difference (ΔV_T) and the transconductance parameter difference ($\Delta\beta$). A widely accepted and experimentally verified model for these random variations is a normal distribution with variance dependent on transistor length L , width W and mutual distance D between the devices:

$$\sigma^2(\Delta V_{T0}) = \frac{A_{VT0}^2}{WL} + S_{VT0}^2 D^2 \quad (3.2.1)$$

$$\left(\frac{\sigma(\Delta\beta)}{\beta}\right)^2 = \frac{A_\beta^2}{WL} + S_\beta^2 D^2 \quad (3.2.2)$$

where A_{VT0} , A_β , S_{VT0} and S_β are technology dependent parameters

An obvious observation from (3.2.1) and (3.2.2) is that by increasing the area the device mismatch can be countered. The price paid in increasing area is larger device parasitic capacitance and larger power dissipation required to achieve the same bandwidth. Therefore there is a trade off between device precision and area.

For transistors on a chip, there are generally two transistor mismatch scenarios. The first case is that transistors are biased by the same voltage and mismatch exists between their terminal currents. The other scenario is when transistors are biased by the same current mismatch exists between their terminal voltages. Mismatch in both scenarios will be analyzed.

3.2.1 Current mismatch

Before we analyze the first mismatch scenario, we first show a very important technique. Based on sensitivity analysis, if G is a function of two independent and normally distributed variables x and y with deviation σ_x and σ_y , $G = f(x, y)$, its deviation and standard deviation are shown as follows:

$$\sigma_G = \frac{\partial f}{\partial x} \sigma_x + \frac{\partial f}{\partial y} \sigma_y \quad (3.2.3)$$

$$\sigma^2(G) = \left(\frac{\partial f}{\partial x}\right)^2 \sigma^2(x) + \left(\frac{\partial f}{\partial y}\right)^2 \sigma^2(y) \quad (3.2.4)$$

For voltage biasing the terminal voltage and the current I_{DS} as the dependent variable, write the current as a function of terminal voltages and the transistor parameters:

$$I_{DS} = f(V_{GS}, V_{DS}, V_T, \beta) \quad (3.2.5)$$

For a pair of transistors with an identical V_{GS} , the difference in their currents is calculated using

$$\Delta I_{DS} = \frac{\partial I_{DS}}{\partial \beta} * \Delta \beta + \frac{\partial I_{DS}}{\partial V_T} * \Delta V_T \quad (3.2.6)$$

Based on the device equations in all regions of operation:

$$\frac{\partial I_{DS}}{\partial V_T} = -\frac{\partial I_{DS}}{\partial V_{GS}} = -g_m \quad (3.2.7)$$

$$\frac{\partial I_{DS}}{\partial \beta} = \frac{I_{DS}}{\beta} \quad (3.2.8)$$

so that the relative current difference in all operating regions of the transistor is expressed by:

$$\frac{\Delta I_{DS}}{I_{DS}} = \frac{\Delta \beta}{\beta} - \left(\frac{g_m}{I_{DS}}\right) * \Delta V_T \quad (3.2.9)$$

and the variance of the current difference is :

$$\left(\frac{\sigma(\Delta I_{DS})}{I_{DS}}\right)^2 = \left(\frac{\sigma(\Delta \beta)}{\beta}\right)^2 + \left(\frac{g_m}{I_{DS}}\right)^2 * \sigma^2(\Delta V_T) \quad (3.2.10)$$

Normally, transistors biased in above threshold saturation region are of concern.

Substitute g_m/I_{DS} with $2/(V_{GS}-V_T)$ and equation (3.2.10) becomes :

$$\left(\frac{\sigma(\Delta I_{DS})}{I_{DS}}\right)^2 = \left(\frac{\sigma(\Delta\beta)}{\beta}\right)^2 + \frac{4}{(V_{GS}-V_T)^2} * \sigma^2(\Delta V_T) \quad (3.2.11)$$

Substitute (3.2.1) and (3.2.2) in (3.2.11), we obtain for closely spaced devices:

$$\left(\frac{\sigma(\Delta I_{DS})}{I_{DS}}\right)^2 = \frac{1}{WL} \left(A_\beta^2 + \frac{4A_{VT0}^2}{(V_{GS}-V_T)^2}\right) \quad (3.2.12)$$

In practice, equation (3.2.12) can be approximated by:

$$\left(\frac{\sigma(\Delta I_{DS})}{I_{DS}}\right)^2 = \frac{1}{WL} \frac{4A_{VT0}^2}{(V_{GS}-V_T)^2} \quad (3.2.13)$$

For the transistor biased in subthreshold region, substitute g_m/I_{DS} with $1/(nU_T)$ and equation (3.2.10) becomes:

$$\left(\frac{\sigma(\Delta I_{DS})}{I_{DS}}\right)^2 = \left(\frac{\sigma(\Delta\beta)}{\beta}\right)^2 + \frac{\sigma^2(V_{T0})}{(nU_T)^2} \quad (3.2.14)$$

3.2.2 Voltage mismatch

For current biasing and the terminal voltage V_{GS} as the dependent variable, it can be shown

$$\sigma^2(\Delta V_{GS}) = \sigma^2(\Delta V_T) + \left(\frac{I_{DS}}{g_m}\right)^2 \left(\frac{\sigma(\Delta\beta)}{\beta}\right)^2 \quad (3.2.15)$$

For a transistor biased in strong inversion

$$\sigma^2(\Delta V_{GS}) = \sigma^2(\Delta V_T) + \frac{(V_{GS} - V_T)^2}{4} \left(\frac{\sigma(\Delta\beta)}{\beta} \right)^2 \quad (3.2.16)$$

For a transistor biased in subthreshold

$$\sigma^2(\Delta V_{GS}) = \sigma^2(\Delta V_T) + (nU_T)^2 \left(\frac{\sigma(\Delta\beta)}{\beta} \right)^2 \quad (3.2.17)$$

3.3 Silicon neuron mismatch analysis

Neuron circuit performance variation due to transistor mismatch is studied in this section. To be specific, the neuron performance we are going to study is neuron F-I curve which is the output spike frequency in response to certain constant input current. As mentioned in earlier chapters, many different versions of neurons exist. However, here we provide a general analysis on how transistor mismatch affect neuron circuits performance which is applicable for the analysis of any neuron circuit! Following the general analysis, we will apply it to a specific neuron circuit studied in chapter 2. And the analysis in this chapter serves as a guideline for optimal design of the proposed circuits.

3.3.1 General Analysis

A neuron circuit can be most generally described by a differential equation as follows:

$$\frac{d\bar{x}}{dt} = Q_{\bar{\alpha}}(\bar{x}, I_{in}, t) \quad (3.3.1)$$

where $\bar{x}$ is the vector of state variables including the membrane voltage (V_{mem}), I_{in} is the input current and $Q_{\bar{\alpha}}$ is the vector field that is dependent on a set of parameters $\bar{\alpha}$. Mismatch between parameters of different neurons will result in different vector fields $Q_{\bar{\alpha}}$. Analyzing the effect of variations rigorously will be very complicated since it

needs us to compare the evolution operators of the corresponding mismatched differential equations. Instead, we propose to use a particular signature of the evolution operator, the F-I curve to analyze neuron circuit mismatch. We make this choice because the F-I curve is a property shared by all neuron circuits and it would directly reflect overall neuron circuit's spiking behaviour. This is a commonly used method for encoding information in spiking neurons. Hence, circuits are already needed on the chip to extract this information leading to no extra overhead for calibration. Moreover, eliminating mismatch in the F-I curve will allow exact spike timings after stimulus presentation to also be matched reducing variability in time-to-first-spike type encoding strategies.

Consider the same current injected into an array of neurons and assuming threshold voltage mismatch is the dominant source underlying F-I curve mismatch, the variance in spiking period can be expressed as

$$\sigma_T^2 = f(\sigma_{V_{T1}}^2, \sigma_{V_{T2}}^2, \dots, \sigma_{V_{Tn}}^2) \quad (3.3.2)$$

where the neuron circuit has n transistors, σ_T^2 is the variance in the neuron spiking period and $\sigma_{V_{Ti}}^2$ is the threshold voltage variance of the i -th transistor. Equation (3.3.2) can be further simplified by assuming the variation of V_T is small, to get a first order approximation:

$$\begin{aligned} \Delta T &= \frac{\partial T}{\partial V_{T1}} \Delta V_{T1} + \frac{\partial T}{\partial V_{T2}} \Delta V_{T2} + \dots + \frac{\partial T}{\partial V_{Tn}} \Delta V_{Tn} \\ &= \sum_{i=1}^n \frac{\partial T}{\partial V_{Ti}} \Delta V_{Ti} \end{aligned} \quad (3.3.3)$$

where T and ΔV_{Ti} are neuron spiking period and threshold voltage difference respectively. Based on this equation we would say the frequency variation is most

sensitive to the term with largest $\frac{\partial T}{\partial T_i}$. Thus an effective idea of coping with F-I mismatch is to find out those transistors whose threshold voltage variations affect the F-I curve most, and use some technique to correct for this[51].

3.3.2 A specific neuron circuit mismatch analysis

Now we use the low power silicon neuron circuit shown in Figure 2.8 studied in chapter 2 to illustrate the above analysis. The circuit in Figure 2.11 or any other complicated neuron circuit can be used to illustrate the principle as well. We choose to use the low power silicon neuron circuit because its structure is simpler. The purpose of the following discussion is to provide a inside view of transistor mismatch effect on neuron performance as well as to serve as a guideline when design an easy-to-match neuron circuit.

First, we perform Monte Carlo analysis to show the problem of threshold voltage random mismatch effect on this neuron performance. Since transistors in neuron circuit are operating in subthreshold region and EKV model has a better prediction of transistor subthreshold behaviour, EKV models were used in all the following simulations. In this Monte Carlo simulation, we assume the standard deviation of transistor threshold voltage $\sigma(V_T)$ of 2mV, 5mV, 10mV and 20m and perform 100 runs for each value of input current to obtain the standard deviation and mean of the spike frequency. The four resulting F-I figures corresponding to $\sigma(V_T)$ of 2mV, 5mV, 10mV and 20mV are shown in Figure 3.3. For each figure, we can observe a spread of frequency in response to a certain input current which is comparable to the resolution reduction in a data convertor. It can also be clearly seen that as threshold voltage standard deviation increases the mismatch on F-I curve becomes more severe with the

coefficient of variation increasing to 0.28 in the worst case. Table 1 tabulates different threshold standard deviations and the corresponding coefficient of variation.

Figure 3.3: Monte Carlo analysis

σ_{VT}	$\left(\frac{\sigma}{\mu}\right)_{max}$	$\left(\frac{\sigma}{\mu}\right)_{ave}$
$2mV$	2.6519%	1.8268%
$5mV$	6.7193%	4.6127%
$10mV$	13.6503%	9.3654%
$20mV$	28.5524%	19.6072%

Table 1: Monte Carlo simulation

Next, as our general analysis implies that the F-I curve mismatch is actually a reflection of every transistor's V_T mismatch, and we are interested in figuring out those transistors that are dominant in affecting the F-I curve. In order to find those 'dominant' transistors in the neuron circuit, we perform another set of simulation. In each simulation, we deliberately change one transistor's V_T , and keep other transistors' V_T at nominal values. Sweep the input current and plot the corresponding F-I curve. For each transistor, we tried both increasing and decreasing V_T from nominal V_T by 100mV. After comparing $\frac{|f_{nominal}-f_{increase}|}{f_{nominal}}$ with $\frac{|f_{nominal}-f_{decrease}|}{f_{nominal}}$, the worst case among these two changes of V_T is plotted in Figure 3.4 and Figure 3.5. It can be seen that M1, M3, M4, M5, M9, M12 and M13 V_T change have the most impact on F-I curve as compared to the other transistors.

Intuitively we can explain why these transistors dominant affecting neuron F-I curve by analyzing the neuron membrane waveform which has the same frequency as the output spike frequency. This neuron membrane waveform can be divided into three distinct parts: the charging and discharging period and the refractory period. The discharging period of a spiking silicon neuron which incorporates AER topology is made much smaller compared to the other two periods. In contrast, other neurons without implementing AER normally have some transistors to control the resting rate at the membrane node in order to set different spike widths, and may result in a considerable amount of discharging time. We are interested in neuron implement with AER topology and hence ignore the discharging time on the membrane node. During charging time, the total current injecting onto the membrane capacitor is

$$I_{total} = I_{in} - I_{leak} + I_{fd}$$

where I_{fd} is the feedback current from M4. Transistor M13 determines the leak current. Transistors M1, M3, M4 & M5 are on the positive feedback path and would both alter the neuron switching threshold and affect the feedback current. The refractory period is directly affected by the voltage V_{rfr} on the gate of M9 and gate voltage of M12. Hence V_T of M9 and M12 affect the neuron F-I curve. Hence, the number of critical transistors is found to be seven. The detailed mathematical analysis is going to be shown in chapter 4.

Figure 3.4: Critical transistor threshold mismatch effect on F-I curve

Figure 3.5: Non-Critical transistor threshold mismatch effect on F-I curve

3.4 Mismatch in silicon neuron application

In this section, we use an example which simulates the working principle of an actual spike-based image sensor to illustrate transistor mismatch effect in silicon neuron applications. The working principle of a conventional image sensor and spike-based vision sensor and the advantages of a spike-based vision sensor over conventional image sensor will be briefly explained first.

3.4.1 A conventional image sensor

Conventional image sensors usually capture and process sequences of frames as shown in Figure 3.6. The sensor consists of many pixels, and the image information (intensity, contrast and etc) acquired by pixels is converted to voltages one pixel at a time as they are sequentially read by other circuitry for processing. In Figure 3.6, pixel values are scanned out from left to right and from top to bottom. The complete image is read after all pixel values are read. Thus some constant period is required to scan each frame. If enhancing the image, extracting features from the image or some other kind of image processing is required, complex processing is performed frame by frame.

Figure 3.6: Frame-based image sensor working principle

We can identify two disadvantages of the frame-based image sensor:

1. This pixel by pixel and frame by frame processing is slow, especially when image resolution increases or complex image processing is needed for each frame.
2. Because all pixels are allocated equal processing power regardless of its relevance, power is wasted on little relevant or nonrelevant pixels. And the complex image processing algorithms require tremendous amount of computing power.

Figure 3.7: Spiking image sensor structure

3.4.2 A spike-based vision sensor

A spike-based vision sensor[52] emulating the human vision system functions in a way conceptually different from that of a conventional image sensor. It does not operate on acquiring and processing sequences of still frames. The structure of the spike-based imager sensor shown in Figure 3.7 composes of a number of processing layers. Each layer consists of many silicon neurons and each neuron connects to a

"projection field" on the next layer. Each neuron sends spikes (also called events) to the connected neurons on the following layer when its activity level reaches a threshold. Spikes are asynchronous and neurons decide when to send an event. All these spikes are transmitted as they are being produced, and do not wait for an artificial "frame time" before sending them to the next processing layer. Connections between neurons are assigned different weights. In this way, image sensed by the first layer neurons is converted into spike trains which are then transmitted across the following layers while performing complex operations, transformations and recognition.

We can see spike-based image sensor has speed and power advantages over the conventional frame-base imager.

1. The spikes from one layer are sent simultaneously to the following layer through projection-fields, while in a traditional image sensor all pixels of a frame have to finish processing before being transmitted to the next processing stage. Consequently, the spike-base processing is structurally much faster.
2. Neurons only generate spikes when there is relevant information in contrast to frame-base approach where all pixels are allocated equal processing power regardless of its relevance. Therefore, spike based approach is much more power efficient.

3.4.3 Transistor mismatch effect on spike-based image sensor

Suppose we use a spike base image sensor to sense a grey coded image shown in Figure 3.8. We are only interested in the performance of the first layer neurons which sense the image and emit pulses at a frequency proportional to the density in this case the grey colour. The neurons are active according to the colour code in the observed

scene which means that if the neuron senses nothing (black) it will not generate spikes, if it senses white colour it will generate spikes with highest frequency. Suppose the sensor consists of 4x4 pixels and the corresponding neuron spikes are shown in Figure 3.9. It can be clearly seen that neurons spike only when there is relevant information in the scene. If neurons are perfectly matched, those firing neurons shall display the same spiking frequency. However, when we deliberately introduce mismatch between neurons, mismatch in firing frequency is observed and the frequencies can no longer represent the true colour.

Figure 3.8: Grey code image

Figure 3.9: 4x4 neuron sensor firing frequencies

Chapter 4 Proposed solutions to counter neuron mismatch

In general, we can compensate mismatch in a neural network from two different perspectives, through tuning either the weights between neurons or individual neuron frequency response to input current. However, fine tuning the weights will work well for rate encoded processing but not when exact timing of the neuron's spike is important. We will concentrate on tuning individual neuron frequency response to compensate mismatch. In this chapter, we will first discuss two special devices namely floating gate transistor and lateral bipolar transistor, which can be used to reduce mismatch in silicon neurons. Then we will present our proposed neuron circuit.

4.1 Floating gate transistor

Since Kahng and Sze reported the first floating gate structure as a mechanism for non-volatile information storage in 1967[53], floating gate transistors have been used widely to store digital information for long periods in a variety of non-volatile solid-state memory devices, including the EPROM (electrically programmable read only memory), EEPROM (electrically erasable, programmable, read-only memory), and flash EEPROM. The extensive researches on floating gate circuits in the past years have shown that floating gate devices can be used not only as digital memory element but also analog memory elements or adaptive circuit elements[54]. Neuromorphic engineers have adapted the floating-gate technology used in digital EEPROMs both to allow non-volatile analog storage, and to perform a local learning function.

A floating gate MOSFET (FGMOSFET) gate is isolated because it is surrounded on all sides by an insulator typically SiO_2 [49]. The basic organization of such a floating gate is shown in Figure 4.1. PMOS transistor is normally used to implement floating gate transistor. A poly-poly capacitor is used to implement input capacitor C_{in} , and

this is why double poly process is normally required to build floating gate transistors. Electrons on the floating gate are prevented from escaping by the surrounding insulator. Voltage V_{tun} inputs to a secondary control gate couple capacitively to the floating gate, thereby modulating the transistor's channel current. From the perspective of this control gate, the presence or absence of excess charge on the floating gate causes a voltage offset and so an apparent shift in the MOSFET's transfer function.

Figure 4.1: Layout, cross-section, and circuit diagram of the FG MOSFET

We quantitatively analyze a PMOS floating gate transistor operating in subthreshold region. Assume the PMOS transistor in Figure 4.1 operating in subthreshold saturation region and neglect body effect. Based on the subthreshold equation given in chapter 3, the drain current I_{ds} is expressed as:

$$I_{ds} = I'_o e^{\frac{V_s - k(V_{fg} + V_{th})}{U_T}} \quad (4.1.1)$$

Express voltage V_{fg} as a function of V_{in} , V_{tun} , and the charge Q stored on the floating node:

$$V_{fg} = V_{in} \frac{C_{in}}{C_T} + V_{tun} \frac{C_{tun}}{C_T} + \frac{Q}{C_T} \quad (4.1.2)$$

where C_T is the total capacitance looking from the floating node to ground and is given by:

$$C_T = C_{in} + C_{tun} + C_{parasitic} \quad (4.1.3)$$

Substituting (4.1.2) and (4.1.3) in (4.1.1) results in

$$I_d = I'_o e^{\frac{V_s - k(V_{fg} \frac{C_{in}}{C_T} + V_{tun} \frac{C_{tun}}{C_T} + V'_{th})}{U_T}} \quad (4.1.4)$$

where V'_{th} is the modified threshold voltage given by

$$V'_{th} = V_{th} + \frac{Q}{C_T} \quad (4.1.5)$$

From equation (4.1.5), we see that the V_{th} of a floating gate transistor can be changed by modifying the floating node charge Q , and because of the very good insulation properties of SiO₂, the charge on the floating gate leaks away very slowly, and the V_{th} will keep high for a long time. Hence transistor mismatch due to V_{th} can be reduced by programming Q .

4.2 Programming of floating gates

Channel hot-electron injection and Fowler-Nordheim tunnelling are the techniques used to inject and remove charge on the floating node[49]. Tunnelling is used for global erase and injection is used for precise programming. PMOS transistors are

normally used as floating gate transistors because their programming characteristics are well studied [20].

4.2.1 Electron tunneling

The electron tunnelling process is illustrated in Figure 4.2. The floating polysilicon gate and a n-well are the two conducting plates of the tunneling capacitor C_{tun} . Electron tunnelling direction is from the floating node to the n-well. A high voltage (e.g. 6.5V in a $0.35\mu m$ CMOS process) is applied to the n+ well contact, reduce the energy barrier between polysilicon and SiO₂, tunneling electrons from the floating gate to the oxide conduction band and then to the n+ well. The n+ is surrounded by a lightly doped n well to prevent reverse-bias *pn*-junction breakdown (from n+ to substrate) during tunneling. Depending on V_{tun} value and tunneling duration, we can control the amount of charge transferred.

Figure 4.2: Electron tunneling

4.2.2 Electron injection

Electron injection adds electrons to the floating gate, and is used to precisely control V_{TH} of the floating gate transistor. This electron injection process shown in Figure 4.3 requires a current flowing through the channel and large channel to drain electric field.

At the channel to drain depletion region, impact ionization occurs: some high energy channel holes collide with the semiconductor lattice and generate free electron hole pairs. A fraction of the free electrons scatter upward into the oxide's conduction band, and are swept to the floating gate by the electric field across the gate oxide.

In this work, during programming V_{SD} is set to 6.5V and the amount of charge injected to the floating gate will be a function of the drain source voltage V_{SD} , the source current I_{SD} , and the duration of the programming signal.

Figure 4.3: Electron injection

4.2.3 Floating gate selection circuits for programming

If a circuit has more than one floating gate transistor, floating gate selection circuits are needed. During programming phase, the selection circuit will select only the target transistor for programming. Our selection circuit is shown in Figure 4.4, including row decoder, column decoder and shift registers. Level shifters are needed because the decoder logic high output is at normal VDD of 3.3V and a higher logic high $avdd_{fg}$ is required in the floating gate circuit which has a VDD of 6.5V. The top transistor in

each floating gate transistor block is a switch and the bottom transistor is the floating gate transistor. The 'Drain line' provides the required current and voltage during programming and is disconnected during normal operation. The floating gate transistors shown in the figure have only their drains connectable. The floating gate transistors with source, drain and gate connectable are not shown.

Figure 4.4: Selection circuit for FG programming

4.3 Lateral bipolar junction transistor

The main cause of MOSFET transistor mismatch is due to threshold voltage mismatch. We can completely get rid of threshold mismatch by using Compatible Lateral Bipolar Transistors (CLBTs)[55] instead of MOSFETs. There is no threshold voltage in BJT exponential current voltage equation and CLBTs are compatible in a CMOS process, thus CLBT is a perfect substitution for subthreshold MOSET when matching is of the

most important requirement. Geometry mismatch is the only cause of mismatch in CLBT devices, and it can be better controlled compared to threshold voltage.

Figure 4.5 shows the layout of the lateral PNP (LPNP) bipolar transistors built in a 0.35 μm CMOS process with one and multiple emitters. The emitter, collector, and base of a CLBT device are just the source, drain, and n-well of a PMOS. The emitter is surrounded by the collector. The CLBT is surrounded by a guard ring. The CLBT performance greatly depends on the layout as will be discussed shortly. The cross section view of the LPNP is shown in Figure 4.5.

Figure 4.5: Lateral PNP BJT Layout

Figure 4.6: Lateral PNP BJT cross section view

The collector of CLBT only collects a fraction α of the emitter current, and the remaining fraction is collected by the vertical BJT that is always biased in the forward

active region. This α is determined by the sidewall to area ratio of the emitter diffusion. In order to maximize lateral collector current efficiency or maximize α , we need to maximize the PMOS source perimeter while minimizing its bottom plate area, and the base width which is the PMOS channel length should also be minimized. Thus, the CLBT structure shown on the left of Figure 4.5 results the best performance, and minimum PMOS channel length and p+ source area are used. To have different CLBT base area, we simply group the desired number of single emitter CLBT to form multi emitter device, shown on the right of Figure 4.5. The CLBT is surrounded by p+ substrate guard ring to prevent the latchup problem. To prevent the PMOS from turning on, the gate voltage must be higher or equal to the source voltage.

4.4 Proposed circuit

Continuing with the neuron circuit mismatch analysis in chapter 3, we now look at how to reduce neuron mismatch by using floating gate transistor or CLBT. An obvious solution seems to be replacing all PMOS transistors in the neuron with special devices. By replacing normal transistors with floating gate transistors, we can tune out mismatch due to any transistor. On the other end, neuron built from CLBT is inherent with good matching property. However these special devices are much larger in size than normal PMOS transistors and neuron built from these special devices will consume large silicon area. Our goal is to minimize the number of special devices used while achieving good matching results. The 'dominant' and 'non-dominant' transistors analysis in section 3.3.2 proves to be a good starting point. As shown in Figure 3.4, neuron F-I curve mismatch is mainly due to those 'dominant' transistors. We have given intuitive explanations on those 'dominant' transistors effect on F-I curve mismatch, and now we want theoretically analyze those transistors and see what we can do with those transistors to reduce neuron mismatch.

4.4.1 Analysis on low power silicon neuron circuit

Suppose the injection current I_{in} is constant, we are going to derive the spiking period of the neuron circuit in mathematical form. Since the spiking frequency is the same as membrane waveform frequency, and we will analyze membrane voltage waveform which consists of charging period, discharging period and a refractory period. We will ignore the discharge period since it is very short.

During charging phase, transistor M12 is off and the differential equation on membrane node is given by

$$C_o \frac{dV_{mem}}{dt} = I_{in} - I_{leak} + I_{fb} \quad (4.4.1)$$

where I_{leak} is the current conducted by M13 and I_{fb} is the feedback current flowing through M4 & M5. Transistors M1, M3, M4 and M13 operate in subthreshold saturation region and M5 operates in subthreshold triode region while M2 behaves like an on switch. Thus equation (4.4.1) can be approximated as follows:

$$\begin{aligned} C_o \frac{dV_{mem}}{dt} &= I_{in} - I_{13} e^{\frac{k(V_{leak} - V_{T13})}{U_T}} \\ &+ I_1 e^{\frac{k(V_{leak} - V_{T1})}{U_T}} * \frac{e^{\frac{k(|V_3| - |V_{T5}|)}{U_T}}}{1 + e^{\frac{k(|V_4| - |V_{T5}|)}{U_T}}} \end{aligned} \quad (4.4.2)$$

$$= I_{in} - \beta_1 I_{13} + \beta_2 I_1 e^{-\frac{kV_{mem}}{U_T}} \quad (4.4.3)$$

where k is the inverse of the sub-threshold slope of a MOSFET and U_T is the thermal voltage and I_x is the current that would flow in transistor M_x if it had both gate voltage and threshold voltage equal to 0.

The refractory period can be calculated based on

$$I_{in} = I_{leak} + I_{reset} \quad (4.4.4)$$

$$= I_{13}e^{-\frac{k(V_{leak}-V_{T13})}{U_T}} + I_{12}e^{-\frac{k(V_{o2}-V_{T12})}{U_T}} \quad (4.4.5)$$

where I_{reset} is the reset current flowing through M12. We can calculate V_{o2} based on

$$C_o \frac{dV_{o2}}{dt} = -I_{refractory} \quad (4.4.6)$$

$$= I_9e^{-\frac{k(V_{vrrfr}-V_{T9})}{U_T}} \quad (4.4.7)$$

which gives

$$V_{o2} = V_{dd} - \frac{t}{C_{o2}} I_9e^{-\frac{k(V_{vrrfr}-V_{T9})}{U_T}} \quad (4.4.8)$$

Substitute equation (4.4.8) into equation (4.4.4), we get the expression for refractory period:

$$\beta_3 t_{rfr} \approx \frac{C_{o2}(V_{dd} - V_{T12} - \frac{U_T}{k} \ln(\frac{I_{in} - I_{13}e^{-\frac{k(V_{leak}-V_{T13})}{U_T}}}{I_{12}}))}{I_9e^{-\frac{k(V_{rfr}-V_{T9})}{U_T}}} \quad (4.4.9)$$

By now we have successfully derived charging time in equation (4.4.2) and refractory period in equation (4.4.9). From equation (4.4.2), we see β_1 is a function of variable V_{T13} and β_2 depends on V_{T1} , V_{T3} , V_{T4} and V_{T5} . From equation 4.4.9, we see β_3 depends on V_{T9} , V_{T12} and V_{T13} . Thus we conclude threshold voltage of transistor M1, M3, M4, M5, M9, M13 and M12 will affect neuron spiking time and this conclusion meets with our simulation result shown in Figure 3.4. Next, we decide which transistors shall be replaced with special devices.

From equation (4.4.2), we see β_1 only depends on V_{T13} , thus M13 has to be a special device. From equation (4.4.9), we see β_3 depends on both V_{T9} and V_{T12} , however we

also find that we can increase or decrease β_3 by just changing one of these two variables. Thus only one special device is required, either M9 or M12; we chose M9 to in our design because of its exponential relation with β_3 . The case for β_2 is more complicated. Not all transistors in this equation have similar effects. For example, if M4 is chosen as the special device, β_2 cannot be made arbitrarily large. We decided to match M3, M4 and M5 using common centroid layout since matching the threshold voltages of M3 and M4 with M5 would eliminate the mismatch contribution of these three. M1 was chosen as the special device. Also, this choice allows all three special devices to be NMOS; as mentioned earlier this is useful while using special devices because special devices can only replace PMOS. Hence, we flip the polarity of the neuron circuit by replacing PMOS by NMOS and vice versa. The dominant transistors are now of p type and can be replaced with special devices. Thus, we find out only three transistors need to be replaced with special devices. In the case floating gate transistors are used as special devices, the resulting neuron circuit is shown in Figure 4.7.

Figure 4.7: Proposed neuron circuit

4.4.2 Tuning out F-I curve mismatch

In this section we demonstrate the procedure to correct the F-I curve mismatch step by step. Suppose there is a neuron array, and we first choose a neuron as a reference which gives a reference F-I curve. Then we tune all other neuron circuits so that their F-I curve match with the reference F-I curve. In this simulation, we still use the low power silicon neuron shown in Figure 2.8. We set all transistors in the reference neuron to nominal V_T , and transistors in a mismatch neuron are randomly assigned V_T generated from a Gaussian distribution whose mean at the nominal V_T and standard deviation of 50mV. Shown in Figure 4.8, reference neuron has its F-I curve denoted as 'nominal' and the mismatch neuron F-I curve denoted as 'random'.

It is important to note that the F-I curve mismatch between two neurons is the result of V_T mismatch between all corresponding transistors. Our approach is to separate the effects of each control 'knob', namely transistor M1, M9 and M13. The three transistors affect the neuron performance in different ways: M13 controls the offset of the F-I curve, FG M9 controls the refractory period and M1 is one of the transistors affecting the charging time. Thus, we divide the mismatch correction procedure into following three steps and the simulation procedure is shown in Figure 4.8.

1. Step one is to correct the slope for the mismatched F-I curve within currents range of interested. Leak current controlled by M13 is first turned off in both the reference neuron circuit and the mismatched circuit by applying zero voltage at gate of corresponding M13. Also refractory period in both reference neuron circuit and the mismatched circuit is minimized by applying a relatively high voltage at gate of corresponding M9. Thus, two new F-I curves result: one for the reference neuron circuit and the other for mismatched circuit. Then, V_T

of M1 in the mismatched circuit is modified to get a close approximation to new reference F-I curve.

2. The second step is to correct the offset of the mismatched F-I curve. Following the first step, turn on leak current in both reference neuron circuit and the mismatched circuit by applying desired voltage at the gate of corresponding M13. This creates an offset in both reference neuron circuit and the mismatched circuit F-I curve. Then tune V_T of M13 in the mismatched neuron to get correct offset.
3. Step three is to correct F-I curve mismatch caused by refractory period mismatch. Reduce gate voltage of corresponding M9 in reference neuron circuit and the mismatched circuit to a desired value. Two new F-I curves result and tune V_T of M9 in mismatched circuit to get a close approximation to the reference F-I curve.

Figure 4.8: Mismatch correction

By changing some critical transistors into FG transistors and through calibration, we can correct for neuron circuit mismatch. It can be seen that using this method, we can correct for very large mismatch in f-i curve through just three FG devices. We have tapeout a chip containing three arrays of neurons which are made from normal transistors, FG transistors and CLBT which will enable us to measure the matching properties of CLBT and our proposed FG based mismatch correction scheme.

Some other researchers have also attempted to tackle the problem of mismatch in silicon neural networks. In the work presented by Neftci et al., a method for configuring networks with neuronal activity represented by “mean firing rates” is presented. In contrast, the method we propose matches the exact differential equation—hence, we can match precise spike timing information with transient input currents. Grassia et al. present a method for tuning ion-channel parameters in silicon neurons by using an evolutionary optimization algorithm to match voltage clamp responses of the silicon channels with software simulations. The method we present has some significant differences compared to this. First, the method presented in Grassia et al. tries to match the behavior of a silicon neuron to a biological neuronal equation; on the other hand, we assume that there is one neuron in the neuron array whose parameters are tuned to get the desired properties and then use our algorithm to make the response of other neurons to be same as this “reference” neuron. The reference neuron parameters might be obtained using the methods described in the work by Grassia et al. Hence, we can use a much simpler algorithm to tune the parameters of the other neurons since the underlying equations are same for all the neurons. Finally, we use current clamp equations to tune the neuron parameters while voltage clamp methods are used in the work by Grassia et al.

Chapter 5 Chip components

In this chapter, we use our proposed silicon neuron circuit presented in the last chapter to build a programmable neuron network. We will first introduce the very important Address Event Representation. Then we will discuss some functional blocks used in our neuron chip. Lastly, we will show our chip designed using VIS 0.35um CMOS process.

5.1 Address Event Representation

Biological neurons connect to one another through extensive connections. However, directly creating the massive connectivity of silicon neurons through metal wires is impractical. Suppose there are 100 neurons on a chip and each neuron is connected to all others, then the number of wire required is 4950! These wires not only create routing problem but also take huge space. The problem becomes even worse when interconnections between multiple chips are needed.

Address event representation (AER)[56, 57] is a popular communication protocol used to solve the massive connectivity problem. In this theme, every neuron is assigned a digital address. The connectivity between neurons is emulated through time multiplexing the address on a high-speed data bus when a neuron generates a spike. The neuron action potential occurs relatively infrequently. As mentioned earlier, there exists a refractory period that limits the frequency at which neurons generate spikes. Thus neuron inter spike interval can be controlled to be much longer than the time needed to transmit the neuron's address. Suppose our neuron firing rate is 100 Hz, and we have a 100 MHz digital address bus, then during the inter-spike interval of a particular neuron one million other neurons' address can be transmitted. In addition, the output digital signals are robust against noise.

Figure 5.1: Dedicated point to point connection through AER

Figure 5.2: The asynchronous handshake

5.1.1 Description

Figure 5.1 illustrates the idea of using AER protocol to create flexible neuron to neuron connections. N_i and n_i represents neurons and s_i may represent either a inhibitory or excitatory synapse. Synapse circuit is basically used to convert pulse signal into a current and will be discussed in the following section. Every neuron in the system is assigned a unique digital address.

If one of the neurons' activity level reaches its threshold which is equivalent to say a 'event' is generated, that neuron sends a request (Req) to an AER transmitter which

includes an address encoder and an arbiter. The arbiter decides which neuron's address should be placed on the output address bus which either directly goes to a receiver chip or goes to a look-up table in an external RAM which stores a list of recipients corresponding to each sender's address. At the same time, the arbiter transmits a request 'chip_Req' to the receiver chip. Once the receiver receives the request from the sender and the connectivity information from the look-up table, the receiver activates a local synapse according the look-up table, which in turn activates a corresponding neuron on the receiver chip. At mean time, the receiver feeds back an acknowledge signal 'chip_Ack' to the sender chip. On receiving the 'chip_Ack' signal from the receiver, the 'chip_Req' signal will be removed and the address on the bus is now invalid. Figure 5.2 summaries the transmission order of 'Req', 'Ack' and 'Address'. When the data transfer cycle terminates, the state of the system is reinitialized so that the cycle is free to occur again when there is another event. In this way, the neuron on one chip is “connected” to neurons on another chip.

This AER approach has the advantage that the bandwidth of the fast digital bus is distributed only to those neurons requiring information transmission, thus, saving bandwidth and power.

5.1.2 Arbiter

As discussed, there is only one address bus shared by all neurons. What happens if multiple neurons generate request signals nearly at the same time and require the access to the share address bus and which neuron will be processed first? This problem is solved by the arbiter circuit which is used to arbitrate the access to the communication bus. Arbiter allows only one requesting neuron to access the address bus at one time and the other requesting neurons have to wait until the bus becomes available again.

Figure 5.3: Arbiter tree

The arbiter circuit on our chip is shown in Figure 5.3. Neurons represented by the circles send requests to the arbiter circuit which is built from many arbiter elements and forms a binary tree structure. Each arbiter element sends a request to the higher level if one or both of its lower level request inputs are active. Thus if one or more neuron fires, the request signal propagates to the top of the tree and a 'chip _Req' is generated. Each arbiter element receives a select signal from the higher level. If this higher level select signal is active, the arbiter element determines which one of the active request inputs to be process first and the corresponding select signal is active. Thus, the select signal propagates from the top of the tree to only one of the active neurons and the neuron address is put on the address bus. For a one dimensional neuron array of size N, the arbiter circuit will have N-1 arbiter elements.

Figure 5.4: Arbiter element

Figure 5.4 shows the gate level of an arbiter element used on our chip and it is designed by our collaborative group. This arbiter element receives two request inputs 'Req1_n' and 'Req2_n', and sends out one request output 'Req'. It receives one select input 'Sel', and sends two select output signals 'Sel1_n' and 'Sel2_n'. All signals in this circuit are active low. The arbiter element can be divided into four function units: (a) priority selection unit (b) arbitration unit (c) request propagation unit and (d) acknowledgement unit. The detail explanation on this arbiter element can be found in [58].

Figure 5.5 gives a good illustration on how this arbiter element works. The simulation is done on a single arbiter element with its two input request signals given by 'Req1_n' and 'Req2_n'. The output signals are 'Sel1_n' and 'Sel2_n'. As mentioned all signals are active low. The waveforms are divided into four sections.

1. Sections (a) and (b) tell the basic idea that who comes first is selected first. For example, in section (a), 'Req2_n' goes low first thus 'Sel12_n' goes low first which means 'Req2_n' is selected for transmission. The select signal goes high only its corresponding request signal goes high, thus 'Sel1_n' is low after 'Req2_n' is done.
2. Sections (c) and (d) give the idea of priority toggling. In section (c), both 'Req1_n' and 'Req2_n' go low at the same time, and the arbiter element first choose to process 'Req1_n' so 'Sel1_n' goes low. In section (d), again both 'Req1_n' and 'Req2_n' go low at the same time, this time 'Req2_n' is process first so 'Sel2_n' goes low.

Figure 5.5: Arbiter element processing illustration

5.2 Synapse

As shown in Figure 5.1, a crucial element in a neuromorphic chip is the 'S' block which represents either an excitatory or inhibitory synapse. A synapse is simply a connection between two neurons. A sender neuron generates spikes output which is then fed to a synapse and got translated into currents input to a receiver neuron. This presynaptic voltage pulses to postsynaptic currents translation has a gain known as the synaptic weight. Same spike arriving at synapses with different weights will generate different postsynaptic currents. Synaptic current is not instantaneous but rather has a temporal profile, typically rising quickly then decaying over a period of milliseconds. In typical VLSI neural network architectures, there may be many synapses connecting to one neuron, and the current generated by multiple synapses are summed by Kirchhoff's current law on the postsynaptic neuron membrane, and eventually the neuron generates output spikes when its membrane potential rises above the threshold voltage.

In VLSI neural networks, a general requirement for synaptic circuits is compactness: the smaller area the synapse circuit occupies, the larger number of synapses can be built on a chip. On the other hand, with the help of AER, one synapse has to act as several synapses in a time-multiplexed fashion. Thus another requirement for synaptic integrator circuits is linear response properties. Here, we just describe one of the circuits capable of doing this: the log domain synapse circuit shown in Figure 5.6. It can be mathematically proven that the circuit forms a first order log-domain filter for current pulses I_w caused by incoming spikes from a presynaptic neuron. Figure 5.6 right depicts the average current produced by this circuit against frequency of $1\mu\text{s}$ wide input pulses for three different values of V_w . It can be seen that changing V_w can effectively control the weight of the synapse.

Figure 5.6: Log domain synapse circuit

Figure 5.7: Switched capacitor

Several other synaptic circuits are described in details in [59]. The synapse circuit used on our chip is simply a switched capacitor circuit shown in Figure 5.7. Capacitors $C1$, $C2$ and $C3$ are 200fF, 400fF and 800fF. Voltage V_w is controlled externally. For a fixed V_w , The synapse weight can be changed by different combinations of control signals $D1$, $D2$ and $D3$. This circuit can respond to a very high input spike frequency.

5.3 Bias generation

Many biases on our chip need to be programmable. For example, in the neuron circuit, we need to tune v_{refr} to get different refractory periods and v_{lik} to get different offsets of the F-I curve. It is not wise to connect those programmable biases to pins because that will not only increase pin count but also increase board complexity and cost. And if high precision is required, off-chip analog components are to be avoided.

Figure 5.8: Bias generator architecture

The programmable biasing system used on our chip is illustrated in Figure 5.8. Biases for n type and p type transistors are denoted as $Vout1$ and $Vout2$ respectively. The core of this system is the current splitter block which can produce a wide range of output current based on the n bits digital inputs. The input current to each current splitter is a copy of the current I_{ref} generated in a bootstrap circuit. The diode connected transistor at the output of each current splitter converts output current into voltage which is then to a buffer. The latches store the control bits $D0, \dots, Dn$ and will update their value when signal cs is active. Shift registers are used to shift in control bit sequence. Thus, only three pads are needed: cs , $Data$, and clock. And these three pads will connect to a

FPGA. Matlab is used to programme the FPGA. Now, I am going to analyze bootstrap circuit and current splitter circuit.

5.3.1 Constant Gm circuit

The bootstrapped current reference is shown in Figure 5.9. Because of the top cascode current mirror formed by M5-M8, same current flows in the two branches. Transistors M1 and M3 have a gain ratio $(W1/L1)/(W3/L3) = K$, and the difference in their V_{GS} is across the load resistor R . Thus, the current in this circuit can be expressed in terms of K and R and is independent of supply voltage.

Figure 5.9: Bootstrap Circuit

In subthreshold, the current equation is expressed by

$$I = I_o e^{\frac{kV_{G1}}{U_T}} = KI_o e^{\frac{kV_{G1} - IR}{U_T}}$$

where V_{G1} is M1&M3 gate voltage. Thus the current can be found:

$$I = \log(K) \frac{U_T}{R}$$

And we get a formula for the voltage V_R across the load resistor R :

$$V_R = \log(K) U_T$$

Thus in subthreshold, V_R directly reflects the temperature.

We can also calculate this current for above threshold, ignoring body effect of M1, and the current equation is given by:

$$\sqrt{\frac{2I}{\beta_3}} + V_{T3} = \sqrt{\frac{2I}{K\beta_3}} + V_{T1} + I * R$$

$$I = \frac{2}{\beta_3 R^2} \left(1 - \frac{1}{\sqrt{K}}\right)^2, \quad \beta = \mu_n C_{ox} \frac{W_3}{L_3}$$

where μ_n is the electron mobility, and C_{ox} is the unit gate oxide capacitance.

A transistor with W/L the same as M3 biased with current I will have its transconductance given by

$$g_m = \frac{k \log K}{R}, \text{subthreshold}$$

$$g_m = \frac{2(1 - 1/\sqrt{K})}{R}, \text{above threshold}$$

It can be seen that the g_m of the transistor is independent of process and temperature for both subthreshold and above threshold region if k and R are independent of temperature and g_m depends only on R , K and k . Thus this bias circuit is known as the constant g_m circuit.

A start up circuit is included in Figure 5.9 because if there is zero current flows in the circuit initially, the current will remain zero forever. During normal operation the start up circuit will not affect the circuit operation.

The bootstrap circuit is also an example of a circuit that uses positive feedback. The addition of resistor kills the closed loop gain (a positive feedback system can be stable if its closed loop gain is less than one). However, if we decrease the size of the resistor, we increase the gain of the loop and push the feedback system closer to instability. An example of when this could occur is if the parasitic capacitance on the source of M1 to ground is large (effectively shorting M1's source to ground). If the resistor for example is bonded out off chip to set, it is likely that this bias circuit will oscillate. The circuit can be stabilized by making large compensation capacitors C_n and C_p .

5.3.2 Current splitter

The original MOS transistor current splitter was proposed by Bult and Geelen[25] who showed that independent of the transistors' bias conditions (weak or strong inversion, saturation or ohmic region), the splitting principle works the same way. The current splitter circuit used on our chip is shown in Figure 5.10. It has a input current I_{ref} which is a copy of the reference current generated by the constant gm circuit discussed above. It has a single end current output I_{out} . The switch steers a current either to ground or output, and can be easily implemented by two PMOS transistors as shown in Figure 5.11 right. Every PMOS in this circuit has the same aspect ratio W/L. This current splitter technique works in a similar fashion as the resistor ladder. Shown in Figure 5.11, each vertical branch has a resistance $2R$ and the resistance looking to the right at each node is R . Thus starting from the left, the first vertical branch has a current of $I_{ref}/2$, the second has current of $I_{ref}/2^2$, and the left most branch has current of $I_{ref}/2^n$. Intuitively, one can understand the current splitting technique by comparing

Figure 5.10 with the resistor ladder shown in Figure 5.11. In Figure 5.10, at each vertical branch two transistors are connected in series which is equivalent to a single transistor with aspect ratio $W/2L$. Ignore the switches and the diode connected transistor on the right most branch which is like an always on dummy switch. Looking to the right at each node, there are two transistors with $W/2L$ connected in parallel which is equivalent to a single transistor with W/L . Thus, starting from the left, current I_{ref} is divided by 2 in the first branch, by 4 in the second and by 2^n in the right most branch. The output current is given by

$$I_{out} = D1 * \frac{I_{ref}}{2^1} + D2 * \frac{I_{ref}}{2^2} + \dots + Dn * \frac{I_{ref}}{2^n}$$

where $D1, D2, \dots, Dn$ are digital control bits and are stored in corresponding latches shown in Figure 5.8. For N bits digital inputs, this current splitter can provides 2^N output options. The output current is then converted to a voltage by a diode connected transistor. Therefore, we can have a wide range of bias voltages. Layout of the current splitter can be made very compact, shown in Figure 5.12. The output current corresponding to the control bit ($D1, D2, \dots, D10$) position is plotted in Figure 5.13.

Figure 5.10: Current splitter

Figure 5.11: R-2R ladder

Figure 5.12: A section of current splitter circuit

Figure 5.13: Output current versus Bit position

5.4 BJT current mirror

As we mentioned, to reduce mismatch problem we also built neuron circuit using lateral PNP BJT. Bias current of BJT has to come from BJT current mirror. However, because of the finite base current, simple BJT current mirror has some problem. Consider the BJT current mirror shown in Figure 5.14, if the transistors are matched and $V_{CE1} = V_{CE2}$, then $I_{C1} = I_{C2}$ but

$$I_{IN} = I_{C1} + 2I_B = I_{C1} * (1 + \frac{2}{\beta}) \quad (5.4.1)$$

$$A_i = \frac{I_{C2}}{I_{IN}} = \frac{1}{1 + \frac{2}{\beta}} \quad (5.4.2)$$

Figure 5.14: BJT current mirror

There is an error term $2/\beta$ in the current mirror gain expression. Since β is typically small, this gain will be smaller than one. And the problem is even worse when the diode connected transistor is supposed to bias a large number of BJT transistors like Figure 5.15. And this is the case when we use BJT neurons to create neural network. Each BJT contributes a base current. If there are n BJT transistors, the error term in equation (5.4.2) will be n/β and the current mirror gain will be far less than one if n is

large. Two solutions to this problem are illustrated in Figure 5.16. We implemented Figure 5.16 right on our chip.

Figure 5.15: A large number of load

Figure 5.16: Improved current mirror

It can be shown that circuit on the left in Figure 5.16 has a current gain

$$A_i = \frac{I_{C2}}{I_{IN}} = \frac{1}{1 + \frac{2}{\beta(\beta + 1)}} \quad (5.4.3)$$

so that the systematic gain error has been reduced by a factor of $\beta+1$, which is the current gain of emitter follower $Q3$. In circuit on the right of Figure 5.16, $M1$ and $Q1$

form a feedback network which will stabilize the base voltage. Any change on base voltage will first affect Q1 collector voltage and in turn will be counter acted by the effect of the source follower M1.

5.5 System architecture

Figure 5.17: System architecture

Figure 5.18: Chip layout

Till now, we have introduced most elements on our chip. In this section I will give an illustration on our neural chip. Our system is implemented using VIS 0.35um CMOS process. It contains three subset systems with each having 128 neurons, 128 excitatory

synapses, 128 inhibitory synapses, an AER transmitter consisting of an arbiter and an address encoder, an AER receiver made up of a pulse generator and an address decoder, biasing circuits, and floating gate related circuits including floating gate selection circuits and a programmer circuit which is used to programme the floating gate transistors used in that system. The subsystem architecture is shown in Figure 5.17. Each neuron receives current input from either the excitatory or the inhibitory synapse. In addition to the synaptic input, the neurons can also be stimulated by direct current injection. Direct current injection is used in the first place to measure every neuron's spiking frequency. Since there are 128 neurons in each neuron array, it would require a 7 bits address bus. The difference between the three subset system is that the neuron circuit is implemented with normal MOSFET transistors, Lateral BJT transistors and float gate transistors.

The chip layout is shown in Figure 5.18. The neuron circuitry implemented by normal transistors occupies an area of 26 μ m by 4.48mm, while the neuron circuitry implemented by Floating gate transistors and Lateral BJT occupy area of 34 μ m by 4.48mm and 52 μ m by 4.48mm respectively. It can be seen that the width of these three neuron sets is the same and lateral BJT neurons occupy the largest area, normal transistor neurons has the least area consumption and floating gate neurons consumes moderate area. The inhibitory and excitatory synapses arrays measure 55 μ m by 4.48mm each.

Figure 5.19 depicts the details about our AER transmitter. Its major component is an arbiter tree discussed earlier. When a neuron generates a spike, the AER transmitter transmits a 'Reg' signal to the same chip or a different chip. At the same time, the address encoder circuit put the neuron address onto the address bus. The 'Ack' received from the target receiver chip and input to a 'Logic' block is used to reset the

neuron whose address is just communicated. Thus the neuron drops down its request and completes the four phase handshaking. Figure 5.20 depicts our AER receiver. It consists of an address decoder circuit and a pulse generator. When the AER receiver receives a input request, the pulse generator will generate a narrow digital pulse with dedicated pulse width and apply it to the synapse corresponding to the input address. Once this is done, the receiver will send the acknowledgement to the sender chip.

Figure 5.19: Schematic and symbol for AER transmitter

Figure 5.20: Schematic and symbol for AER receiver

5.6 Silicon neuron implementation of Extreme Learning Machines

Our chip has great potential to be used in many applications, for example, using our chip to implement extreme learning machines (ELM)[60]. ELM is a learning algorithm for single-hidden layer feedforward neural networks (SLFNs): in the case when the hidden nodes in a SLFN are random, ELM can be used to determine the output weights of the SLFN. This algorithm tends to provide good generalization performance at extremely fast learning speed. Figure 5.21 shows the architecture of such a network. The output weights β_i are only trained in this network. The final output is a weighted summation of the outputs o_j of the hidden layer.

Figure 5.21: A SLFN with random hidden nodes 1,2,...,L and Output weights $\beta_1, \beta_2, \dots, \beta_L$ determined by ELM

The mismatch in silicon neuron is typically considered to be a serious and limiting problem [22], but is actually a useful feature for ELM since it requires the nodes on the hidden layer to be random. Thus without mismatch calibration the neurons on our chip can be used to implement the nodes on the hidden layer. One problem with this approach might be that the random distribution of device parameters are typically

Gaussian while for ELM, better performance is achieved by choosing parameters from a uniform distribution. For an efficient ELM implementation, our FG neurons can be used instead and parameters of every neuron can be tuned, such that the distribution of currents in these transistors follows a uniform distribution.

The inputs to the hidden layer, denoted as x_i can be mapped to spike rates or frequency of incoming pulses from every input node. Depending on the application, they can be obtained from a sensor that detects light or sound.

Figure 5.22: Architecture for implementing the ELM

Figure 5.22 depicts one possible system level implementation of the input and hidden layers. Every individual block has been studied in the previous section. Input spikes

from a sensor are accepted by the neuron chip using AER receiver and is used to stimulate the desired neuron based on the address. Since only one synapse is time shared to mimic the performance of many synapses, the weights of the synapses also need to be set separately using a DAC for every stimulation. The input currents are then processed by the neurons and their output spikes can be interfaced asynchronously with the digital and memory blocks using an AER transmitter.

Figure 5.23: The weighted summation for the ELM output layer

For an ELM, the output weights $\beta_1, \beta_2, \dots, \beta_L$, that are calculated based on training need to be accurately set for good performance. We can exploit the high accuracy of digital implementations by setting these weights in digital domain. As mentioned earlier, the synapse weights can be stored in a look up table in RAM. Thus, the calculated output weights will be stored in a look up table.

Figure 5.23 shows the digital architecture for implementing the output layer of an ELM. The output layer should perform a linear weighted summation of the outputs of the hidden layer. We can use a multiplier-less, counter based approach to perform this function as shown in Figure 5.23. The sampling rate of the whole system is T_{clk} determined by the 'Sys clock' signal. The counter which stores the final output is reset

every clock cycle. Every time a neuron fires, its address, say 'j', is used to look up the corresponding weight β_j from a memory. Then β_j number of clock pulses are applied to the counter to increment its value. Hence, if neuron 'p' fires N_p times in a clock cycle, it contributes $N_p \times \beta_p$ to the counter's value in that cycle. Thus, the output of the counter in every cycle stores the weighted summation of the neuron spike rate in every cycle. Since the digital pulses being applied to the counter only need to charge and discharge capacitances internal to a chip, it can be driven at a high speed to implement a large dynamic range for β . The refractory period for every neuron limits its firing rate allowing the AER transmitter to easily multiplex all the output spikes on a single shared digital bus. Hence, this overall architecture provides an optimal partition of the ELM computation into analog and digital by exploiting the low-power analog processing as well as high accuracy digital computation.

Chapter 6 Conclusion and future work

6.1 Conclusion

Brain simulation and emulation is attracting more and more researchers' attention over the last few decades because of the amazing capabilities of the human brain which the digital processors cannot compete with. And neural networks have been applied across an extraordinary range of problem domains, in areas as diverse as finance, medicine, engineering and biology.

In this thesis, we have given a literature review from artificial neural network to hardware implementation of spiking silicon neuron. In particular, integrate and fire neuron is the focus of this thesis and we present two different I&F neuron circuits, namely the low power I&F neuron and DPI input neuron. Because neural chip normally consists of many silicon neurons and transistors in neuron circuits operate in subthreshold region, device mismatch poses a serious problem causing neuron performance to vary across a chip. We discussed the method of using sensitivity analysis to analyze each transistor threshold mismatch effect on neuron circuit's overall performance. We proposed to use special devices, like floating gate transistor and lateral BJT transistor, to replace some 'dominant' transistors in a neuron circuit. In this way, neuron performance can be controlled by calibration. We use the low power neuron circuit as an example to illustrate our idea of separating 'dominant' transistors, replacing them with floating gate transistors and demonstrating the procedure of mismatch correction using floating gate transistors.

We have implemented a system comprising three different types of neuron arrays, synaptic circuits, and input/output AER interfacing circuits. The three different sets of neurons are implemented using normal CMOS transistors, Lateral BJT transistors and

floating gate transistors. Neuron parameters, like leak current and refractory period, can be tuned through digital control using Matlab programmes. Our chip occupies 2.5mmx5.5mm area using VIS 0.35um technology. The on chip system is about to communicate with neurons on the same chip or on a different chip through transmitting and receiving spikes via AER transmitter and receiver. Our chip can be used in many applications and we have shown an example of using our chip to implement ELM.

6.2 Future work

We have tested our chip. And unfortunately, we discovered some design problems causing the chip to malfunction. As mentioned in the last chapter, there are three separate neuron arrays on our chip, and some digital controls are used to separate one system from the rest when the chip operates. There is a problem in the connections of these digital controls to the bias circuits causing bias circuits to malfunction. We are now revising the chip schematic and layout, and preparing for the second tapeout.

Author's Publications

S. Shuo and A. Basu, "Analysis and reduction of mismatch in silicon neurons," *IEEE BioCAS*, pp. 257-260, 2011

Arindam Basu, Shuo Sun, Ming H Zhou, Meng H Lim and Bin G Huang, "Silicon Spiking Neurons for Hardware Implementation of Extreme Learning Machines," *Neurocomputing* 2011, accepted.

Reference

- [1] *FACETS: Fast Analog Computing with Emergent Transient States*. Available: <http://facets.kip.uni-heidelberg.de/index.html>
- [2] *CAVIAR Project*. Available: <http://www2.imse-cnm.csic.es/caviar/goals.htm>
- [3] *SyNAPSE*. Available: http://www.ibm.com/smarterplanet/us/en/business_analytics/article/cognitive_computing.html
- [4] *Neurogrid*. Available: <http://www.stanford.edu/group/brainsinsilicon/>
- [5] C. Mead, "Neuromorphic Electronic Systems," *IEEE Proceedings*, vol. 78, pp. 1629-36, 1990.
- [6] K. Boahen, "Neuromorphic chips," *Scientific American*, pp. 56-63, 2005.
- [7] R. Douglas, M. Mahowald, and C. Mead, "Neuromorphic analogue VLSI," *Annual Review of Neuroscience*, vol. 18, pp. 255-281, 1995.
- [8] R. Sarpeshkar, "Brain power - borrowing from biology makes for low power computing [bionic ear]" *IEEE Spectrum*, vol. 43, pp. 24-29, 2006.
- [9] G. Indiveri, B. Linares-Barranco, T. J. Hamilton, A. v. Schaik, R. Etienne-Cummings, T. Delbruck, S.-C. Liu, P. Dudek, P. Hafliger, S. Renaud, J. Schemmel, G. Cauwenberghs, J. Arthur, K. Hynna, F. Folowosele, S. Saighi, T. Serrano-Gotarredona, J. Wijekoon, Y. Wang, and K. Boahen, "Neuromorphic silicon neuron circuits," *Frontiers in Neuroscience*, vol. 5, pp. 1-23, 2011.
- [10] K. A. Zaghloul and K. Boahen, "A silicon retina that reproduces signals in the optic nerve," *Journal of Neural Engineering*, vol. 3, pp. 257-267, 2006.
- [11] B. Wen and K. Boahen, "A Silicon Cochlea With Active Coupling," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 3, pp. 444-455, 2009.
- [12] R. J. Vogelstein, U. Mallik, E. Culurciello, G. Cauwenberghs, and R. Etienne-Cummings, "A Multichip Neuromorphic System for Spike-based Visual Information Processing," *Neural Computation*, vol. 19, pp. 2281-2300, 2007.
- [13] R. E. Cummings and J. V. d. Spiegel, "Neuromorphic vision sensors," *Sensors & Actuators: A. Physical*, vol. 56, pp. 19-29, 1996.
- [14] K. Zaghloul and K. Boahen, "A silicon retina," *Journal of Neural Engineering*, vol. 3, pp. 257-267, 2006.
- [15] L. Watts, D. Kerns, R. Lyon, and C. Mead, "Improved implementation of the silicon cochlea," *IEEE Journal of Solid-state circuits*, vol. 27, pp. 692-700, 1992.
- [16] R. Sarpeshkar, R. Lyon, and C. Mead, "An analog VLSI cochlea with new transconductance amplifiers and nonlinear gain control," *IEEE International Symposium on Circuits and Systems*, 1996.
- [17] A. v. Schaik, E. Fragniere, and E. Vittoz, "Improved silicon cochlea using compatible lateral bipolar transistors," *Advances in Neural Information Processing Systems*, pp. 671-677, 1996.
- [18] J. A. Pérez-Carrasco, B. Acha, C. Serrano, L. Camuñas-Mesa, T. Serrano-Gotarredona, and B. Linares-Barranco, "Fast Vision Through Frameless Event-Based Sensing and Convolutional Processing: Application to Texture Recognition," *IEEE TRANSACTIONS ON NEURAL NETWORKS*, vol. 21, pp. 609-620, April 2010.
- [19] R. Serrano-Gotarredona, T. Serrano-Gotarredona, A. Acosta-Jiménez, and B. Linares-Barranco, "A Neuromorphic Cortical-Layer Microchip for Spike-Based Event Processing Vision Systems," *IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS*, vol. 53, pp. 2548-2566, Dec 2006.

- [20] J. Costas-Santos, T. Serrano-Gotarredona, R. Serrano-Gotarredona, and B. Linares-Barranco, "A Spatial Contrast Retina With On-Chip Calibration for Neuromorphic Spike-Based AER Vision Systems," *IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS*, vol. 54, pp. 1444-1458, July 2007.
- [21] J. P. d. Gyvez and H. P. Tuinhout, "Threshold Voltage Mismatch and Intra-Die Leakage Current in Digital CMOS Circuits," *IEEE Journal of Solid-State Circuits*, vol. 39, pp. 157-168, 2004.
- [22] P. Andricciola and H. P. Tuinhout, "The temperature Dependence of Mismatch in Deep-Submicrometer Bulk MOSFETs," *IEEE Electron Device Letters*, vol. 30, pp. 690-692, 2009.
- [23] P. R. Kinget, "Device Mismatch and Tradeoffs in the Design of Analog Circuits," *IEEE Journal of Solid-State Circuits*, vol. 40, pp. 1212-1224, 2005.
- [24] R. Serrano-Gotarredona, T. Serrano-Gotarredona, A. Acosta-Jimenez, and B. Linares-Barranco, "A Neuromorphic Cortical-Layer Microchip for Spike-Based Event Processing Vision Systems," *IEEE Transactions on Circuits and Systems*, vol. 53, pp. 2548-2566, 2006.
- [25] R. Serrano-Gotarredona, T. Serrano-Gotarredona, A. Acosta-Jimenez, C. Serrano-Gotarredona, J. A. Perez-Carrasco, B. Linares-Barranco, A. Linares-Barranco, G. Jimenez-Moreno, and A. Civit-Ballcells, "On Real-Time AER 2-D Convolutions Hardware for Neuromorphic Spike-Based Cortical Processing," *IEEE Transactions on Neural Networks*, vol. 19, pp. 1196-1219, 2008.
- [26] B. Linares-Barranco, T. Serrano-Gotarredona, and R. Serrano-Gotarredona, "Compact low-power calibration mini-DACs for neural arrays with programmable weights," *IEEE Transactions on Neural Networks*, vol. 14, pp. 1207-1016, 2003.
- [27] J. A. M. Olsson and P. Häfliger, "Mismatch reduction with relative reset in integrate-and-fire photo-pixel array," *IEEE Biomedical Circuits and Systems Conference*, pp. 277-280, 2008.
- [28] F. Grassia, L. Buhry, T. Lévi, J. Tomas, A. Destexhe, and S. Saïghi, "Tunable neuromimetic integrated system for emulating cortical neuron models," *Frontiers in Neuroscience*, 2011.
- [29] E. Neftci, E. Chicca, G. Indiveri, and R. Douglas, "A Systematic Method for Configuring VLSI Networks of Spiking Neurons," *Neural Computation*, vol. 23, pp. 2457-2497, 2011.
- [30] A. Hastings, *The Art of Analog Layout*: Prentice Hall, 2006.
- [31] McCulloch and Pitts, "A logical calculus of ideas immanent in neural activity," *Bulletin of Mathematical Biophysics*, vol. 5, pp. 115-133, 1943.
- [32] W. Maass, "Networks of Spiking Neurons: The Third Generation of Neural Network Models," *Neural Networks*, vol. 10, pp. 1659-1671, 1997.
- [33] W. Maass and H. Markram, "On the computational power of recurrent circuits of spiking neurons," *Journal of Computer and System Sciences*, vol. 69, pp. 593-616, 2004.
- [34] K. Uchizawa, R. Douglas, and W. Maass, "On the computational power of threshold circuits with sparse activity," *Neural Computation*, vol. 18, pp. 2994-3008, 2006.
- [35] W. Maass, "On the Computational Power of Winner-Take-All," *Neural Computation*, vol. 12, pp. 2519-2535, 2000.
- [36] W. Maass, *Models of Neural Networks IV: Early Vision and Attention*. New York: Springer, 2001.

- [37] W. Maass, "Lower bounds for the computational power of networks of spiking neurons," *Neural Computation*, vol. 8, pp. 1-40, 1996.
- [38] W. Gerstner and W. Kistler, *Spiking Neuron Models*. : Cambridge University Press, 2002.
- [39] J. O'Keefe and M. L. Recce, "Phase relationship between hippocampal place units and the EEG theta-rhythm," *Hippocampus* pp. 3317-330, 1993.
- [40] W. Maass and C. M. Bishop, *Pulsed Neural Networks*: MIT Press, 2001.
- [41] H. AL and H. AF, "A quantitative description of membrane current and its application to conduction and excitation in nerve," *J Physiol*, pp. 500-44, 1952.
- [42] L. F. Abbott, "Lapicque's introduction of the integrate-and-fire model neuron," *Brain Research Bulletin*, vol. 50, pp. 303-304, 1999.
- [43] W. Maass and C. M. Bishop, *Pulsed Neural Networks* MIT Press, 1998.
- [44] N. Ganesan, K. Venkatesh, M. A. Rama, and A. M. Palani, "Application of Neural Networks in Diagnosing Cancer Disease Using Demographic Data," *International Journal of Computer Applications*, vol. 1, 2010.
- [45] L. Bottaci, P. J. Drew, J. E. Hartley, M. B. Hadfield, R. Farouk, Peter W R Lee, I. M. C. Macintyre, G. S. Duthie, and J. R. T. Monson, "Artificial neural networks applied to outcome prediction for colorectal cancer patients in separate institutions," *THE LANCET*, vol. 350, pp. 469-472, 1997.
- [46] R. M. Balabin and E. I. Lomakina, "Neuralnetwork approach to quantum-chemistry data: Accurate prediction of density functional theory energies," *THE JOURNAL OF CHEMICAL PHYSICS*, vol. 131, 2009.
- [47] C. Mead, *Analog VLSI and Neural Systems*: Addison-Wesley, 1989.
- [48] G. Indiveri, E. Chicca, and R. Douglas, "A VLSI Array of Low-Power Spiking Neurons and Bistable Synapses With Spike-Timing Dependent Plasticity," *IEEE Transactions on Neural Networks*, vol. 17, pp. 211-221, 2006.
- [49] S.-C. Liu, J. Kramer, G. Indiveri, T. Delbruck, and R. Douglas, *Analog VLSI: Circuits and Principles*: MIT Press, 2002.
- [50] M. J. M. Pelgrom, A. C. J. Duinmaijer, and A. P. G. Welbers, "Matching Properties of MOS Transistors," *IEEE Journal of Solid-State Circuits*, pp. 1433-1439, 1989.
- [51] S. Shuo and A. Basu, "Analysis and reduction of mismatch in silicon neurons," *IEEE BioCAS*, pp. 257-260, 2011.
- [52] R. Serrano-Gotarredona, M. Oster, P. Lichtsteiner, A. Linares-Barranco, R. Paz-Vicente, F. Gómez-Rodríguez, L. Camuñas-Mesa, R. Berner, M. Rivas-Pérez, T. Delbrück, S.-C. Liu, R. Douglas, P. Häfliger, G. Jiménez-Moreno, A. C. Ballcells, T. Serrano-Gotarredona, A. J. Acosta-Jiménez, and B. Linares-Barranco, "CAVIAR: A 45k Neuron, 5M Synapse, 12G Connects/s AER Hardware Sensory-Processing-Learning-Actuating ystem for High-Speed Visual Object Recognition and Tracking," *IEEE TRANSACTIONS ON NEURAL NETWORKS*, vol. 20, pp. 1417-1438, 2009.
- [53] D. Kahng and S. M. Sze, "A floating gate and its application to memory devices," *The Bell System Technical Journal*, vol. 46, pp. 1288-1295, 1967.
- [54] P. Hasler, "Floating gate devices they are not just for digital memories any more," *IEEE International Symposium on Circuits and Systems* vol. 2, pp. 388-391, 1999.
- [55] E. Vittoz, "MOS transistors operated in the lateral bipolar mode and their application in CMOS technology," *IEEE Journal of Solid-State Circuits*, vol. 18, pp. 273-279, 1983.

- [56] R. Serrano-Gotarredona, T. Serrano-Gotarredona, A. Acosta-Jiménez, C. Serrano-Gotarredona, J. A. Pérez-Carrasco, B. Linares-Barranco, A. Linares-Barranco, G. Jiménez-Moreno, and A. Civit-Ballcels, "On Real-Time AER 2-D Convolutions Hardware for Neuromorphic Spike-Based Cortical Processing," *IEEE TRANSACTIONS ON NEURAL NETWORKS*, vol. 19, pp. 1196-1219, 2008.
- [57] M. Mahowald, "VLSI Analogs of Neuronal Visual Processing: A Synthesis of Form and Function," California Institute of Technology Pasadena, 1992.
- [58] A. M. T. Linn, D. A. Tuan, C. Shoushun, and Y. K. Seng, "Adaptive priority toggle asynchronous tree arbiter for AER-based image sensor," *International Conference on VLSI and System-on-Chip (VLSI-SoC)*, pp. 66-71, 3-5 Oct. 2011 2011.
- [59] C. Bartolozzi and G. Indiveri, "Synaptic Dynamics in Analog VLSI," *Neural Computation*, vol. 19, pp. 2581-2603, 2007.
- [60] G. B. Huang, Q. Y. Zhu, and C. K. Siew, "Extreme learning machine: Theory and applications," *Neurocomputing* vol. 70, pp. 489-501, 2006.