University of Oslo
Department of Informatics

Multi-Level Static Memory for On-chip Learning

Cand. Scient. Thesis

Håvard Kolle Riis

August 2003

Acknowledgments

This thesis concludes my work for the Candidatus Scientiarum (Cand. Scient.) degree at the Institute of Informatics, University of Oslo. My work was initiated in February 2002 and was completed August 2003.

I would like to thank my supervisor Philipp Häfliger for accepting me as his student, for guidance and help during the process and for always being present to answer my questions. It has been inspiring.
Siri, Svein, Maren and the rest of my family for unconditional support and help. Also Thorleif, for a soft fur no matter what happens.
Annika, for her patience and understanding when nothing else than “synaptic storage” was on my mind.
Finally, fellow students and friends, especially Jørgen Walberg Bakke, Vegard Sandvold and Omid Mirmotahari for tips and help (and coffee breaks) which saved me a lot of work. Also Anders Tollefsen, Bjørn Erik Hansen, Sølve Huse and Emil Magnussen for non-curriculum activity.

Preface

This thesis investigates different memory types for use as a synaptic storage in a neuromorphic application for on-chip learning. After initial investigation for a relevant implementation, no optimal solutions were found, and we decided to test a new Multi-Level Static Memory (MLSM) which is presented in this thesis. We will only give a brief introduction to the different alternative memory types and concentrate on the implementation with the MLSM.

This thesis is divided into four chapters.

- **Chapter 1**: Introduction to basic theory and principles.
- **Chapter 2**: The main objective and the environment for the implementation are introduced.
- **Chapter 3**: The main part of this thesis introduces the circuit components. The functionality and implementation of each component are presented as well as test results.
- **Chapter 4**: Final words, where the implementation is discussed. Furthermore, future work is proposed

Contents

Chapter 1

Introduction

In this chapter, we explain some of the basic theory and principles used in this thesis. We give an introduction to the neurvous system and neuromorphic electronics.

1.1 The neuron

The human body is made up of trillions of cells. Cells of the nervous system, called nerve cells or neurons, are specialized to carry information through an electrochemical process. The human brain has about 10^{11} neurons and they come in many different shapes and sizes. Some of the smallest neurons have cell bodies that are only 4 μm wide, while some of the biggest neurons have cell bodies that are 100 μm wide. A sketch of a neuron can be seen in Figure 1.1.

The neuron connects with other neurons through synapses, where dendrites bring information to the cell body and an axon transmits information away from the cell body. Information is transmitted with Action Potentials (APs), or "spikes", which is sent from the neuron when the integrated input from all synapses exceeds a certain voltage threshold. When a neuron receives an AP, a Post Synaptic Current (PSC) is sent to the cell body. The size of this PSC is determined by a variable synaptic weight. Thus, an input AP to one synapse may itself trigger an output AP, while an input AP to a different synapse may completely be ignored. These weights play an important part in the learning process in the brain. It is believed that it is the weights that store the information in the brain, and that they are locally adapted according to complex interaction between neurons.

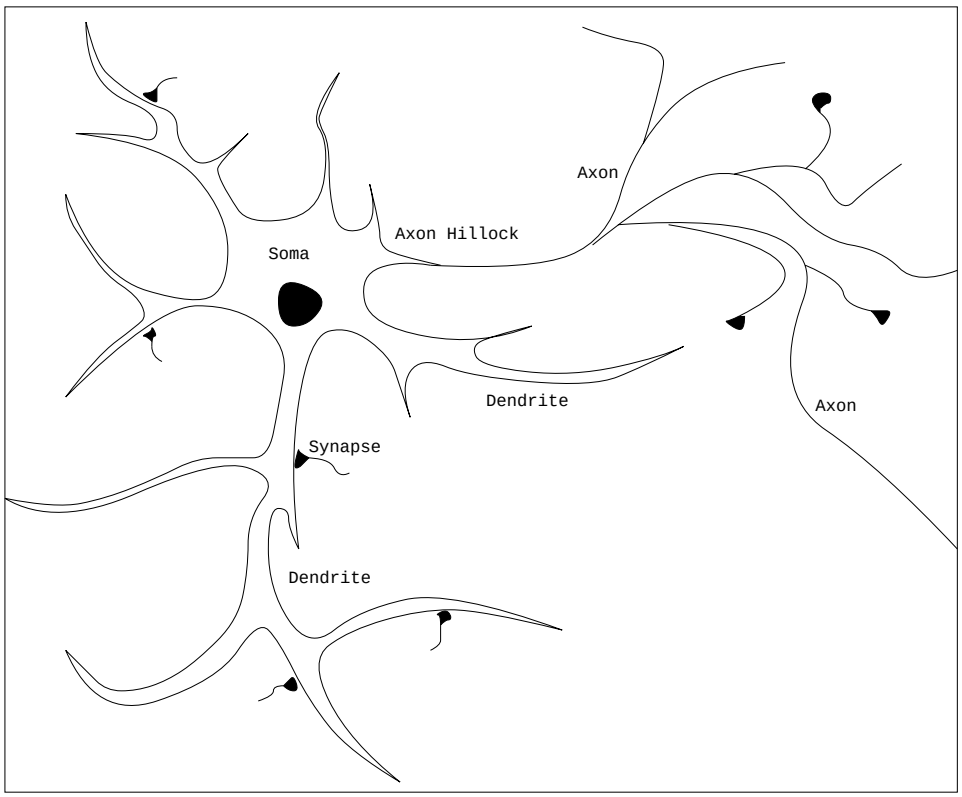

Figure 1.1: *A sketch of a nerve cell. A singe nerve cell can connect to as many as 10.000 other neurons.*

1.2 Neuromorphic systems

There has been extensive research on the human brain, but there are yet many unsolved and unexplored parts of the central nervous system. To understand the complex communication and interaction between the neurons in the brain is in itself a difficult task, and to copy this complexity to electronic circuits is even more difficult. Since neurons use currents and voltages to communicate, electronic circuits can be used efficiently to emulate real neurons and complex neural networks. There have been proposed several models such as “The silicon neuron” [1], the “Integrate-and-Fire neuron” [2] and “Perceptrons” (McCulloch Pitts neurons) [3], which emulates the behavior of the nerve cell. These circuit models, amongst others, can be combined to construct a network of neurons. Such a network is an example of a neuromorphic system, a term that was first defined by Mead [4]. Neuromorphic systems are artificial systems based on computational principles used by biological

nervous systems. Neuromorphic engineering attempts to implement devices in order to solve tasks that biological systems perform so easily, like visual and auditory perceptive processing, navigation and locomotion, classification, recognition, forecasting and prediction to mention a few. An example is the success within the field of emulation of peripheral sensory transduction and processing performed by biological retinas [5, 6] and cochleae [7, 8].

1.3 Neural coding

In the nervous system, APs are the main form of information transmission. APs have a fixed amplitude (-70mV to +30mV) and a fixed duration of approximately 1ms. The classical view is that information is transmitted using a firing rate code, but resent experiments have shown that this may not be the case in certain parts of the nervous system. Thorpe *et. al.* [9], conducted a psychophysical experiment to show that the human visual system can process complex natural images in roughly 150ms. This would make very few cells able to fire more than one spike before the next stage has to respond. Clearly, one or two spikes are not enough to differentiate different frequencies. This indicates that there must also be some finer temporal information in an AP sequence.

Markram *et. al.* [10] performed an experiment which further validates that there are some temporal information in an AP. They observed that when a depolarizing current was injected into a presynaptic neuron to produce a presynaptic AP, no changes in the average excitatory postsynaptic potential (EPSP) amplitude was seen in the postsynaptic cell. However, when such an injection was followed by a similar injection in the postsynaptic neuron to produce a postsynaptic AP, an increase in the average EPSP amplitude was observed in the postsynaptic cell. Figure 1.2 illustrates this behavior. This indicates that the timing of the presynaptic spike was crucial for the metabolic growth of the synaptic connection.

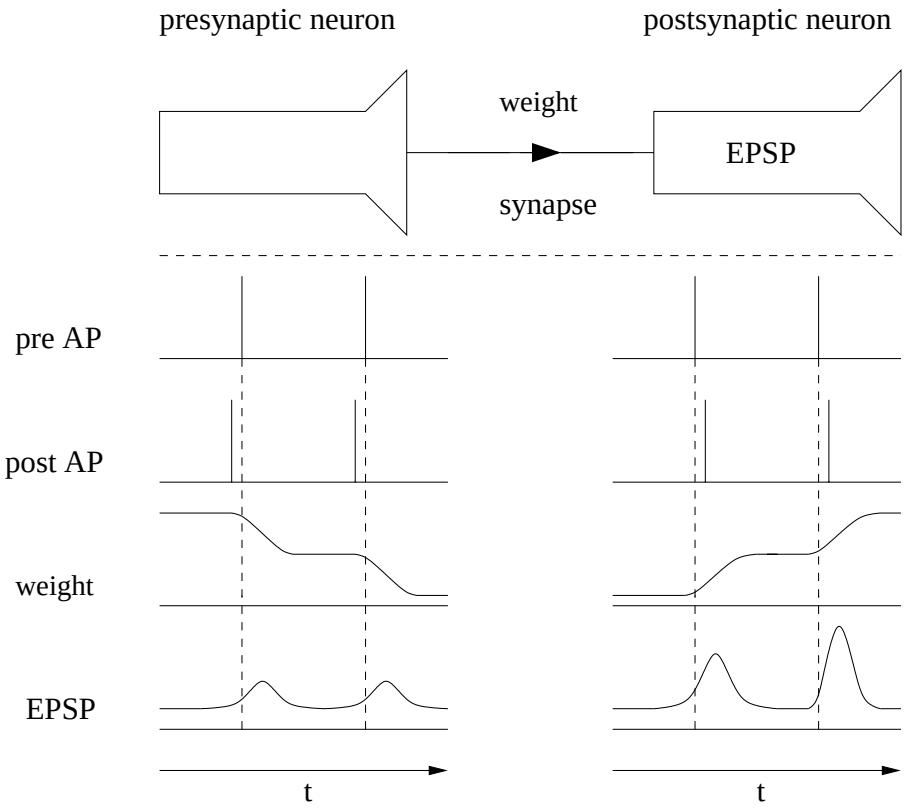

Figure 1.2: *An illustration of the test setup used in [10]. Below, two graphs show the behavior observed. To the left, the postsynaptic spike comes before the presynaptic spike and the EPSP amplitude remains the same for both stimulations. The weight is the synaptic efficacy which controls the rate of PSC which reaches the postsynaptic neuron when the presynaptic neurons sends an AP. We see that the weight decreases hence the effect of a presynaptic AP to the postsynaptic neuron decreases. To the right, the presynaptic AP comes first and an increase in the average EPSP amplitude is observed. At the same time the synaptic efficacy is strengthened.*

Chapter 2

Objective

We have now explained the basic principles of neuromorphic electronics. In this chapter, we focus on various aspects which in some way affect our implementation. We will introduce the environment and the primary goals of this thesis.

2.1 Synaptic storage

In the human brain there are no external storage. Memory is distributed throughout the brain as weights stored in the connections between neurons. In the field of neuromorphic electronics, distributed analog or multi-valued memory is preferred to reflect this behavior, and it also minimizes speed and space compared to an external digital memory. Today, there exists several types of multi-level or analog memory which can be categorized in three groups: Dynamic, static and non-volatile memory.

2.1.1 Dynamic memory

Dynamic memory is a short term volatile memory, which is optimal for storing data that will not be preserved for a long time. Usually a capacitor is used to store the value and this makes it space conservative and easy to implement. Because of leakage, a refresh mechanism is needed if the value is to be maintained over longer periods. This increases the complexity and noise from digital components. Transmission lines will also interfere with the analog signals and lower the signal-to-noise ratio.

In digital dynamic memory, there are only two values, high (V_{dd}) and low (V_{ss}). To preserve the stored value, memory has to be refreshed before the value exceeds a certain threshold value, i.e. $V_{dd}/2$. For example, if the leakage is 1mV per second and V_{dd} = 5V, then it is necessary to refresh

every 2500 seconds (approximately 42 minutes) to preserve the stored value.
In dynamic multi-level memory, the memory space is divided into slots. In the same manner as digital memory, the capacitances value has to be restored to its original value after a time t. This may be done using an external digital mechanism, such as a digital register with digital-to-analog and analog-to-digital converters, and many such implementations have been proposed for use in neural networks as synaptic storage [11–17]. How accurate you want this memory element to be, sets the spacing of these slots and consequently the refresh frequency.

2.1.2 Multi-level static memory

Multi-level or analog static memory is memory where the values stored are preserved through a local feedback path, e.g. latches and flip-flops. Like dynamic memory, the stored value is not preserved when the power is turned off, but since it preserves its own state, no external refresh algorithm is needed. In digital static memory, the stored value switches between two values, V_{dd} and V_{ss}. This is a fairly easy task, and many different digital static memory elements exists. However, multi-level static memory is a more complex task, and only a few implementations have been proposed [18, 19]. The implementation by Cilingiroglu *et. al.* [19] seemed to be the most promising for our purpose, where a multiple-valued static CMOS cell has been proposed for synaptic storage. This is the first and only implementation of this kind we could find, which is used for synaptic storage. However, the proposed element has some disadvantages which makes it unsuitable for our implementation. We will discuss this further in Chapter 4.

2.1.3 Non-volatile analog memory

In non-volatile analog memory, the memory does not loose its stored value even when the power is turned off. An example is analog EEPROM (Electrical Erasable Programmable ROM). Here, the charge is kept on electrically isolated conductances, i.e. floating-gates (FG), which can be programmed on-chip [20]. Non-volatile memory is used to store parameters and constants, or values which do not change rapidly over time. Since learning in neuromorphic circuits often depend on slow adaption, non-volatile memory has been used to a wide extent as a storage element in such models [21–28]. However, there are severe device property mismatches and specialized initialization and programming techniques are required to alter the value stored.

2.2 Learning

We perform the process of learning every day, but what is learning? One definition is that learning is the search of a parameter space in order to optimize performance. A more common definition is that learning is a change in behavior as a result of experience.

Since we are able to learn, we can easily perform tasks that originally require a high degree of concentration. Even more important, we can adapt to new situations. So the process of learning is important since we cannot be designed for every possible situation we encounter.

Modern integrated circuits (IC) can be very complex and powerful. But most commercial ICs are designed for certain applications and tasks. Therefore, they are not flexible and when conveyed with an unknown task, it will not be able to determine what to do. This is of course a crude generalization and there exists IC which can be programmed after fabrication. But the programming usually needs to be supervised. Therefore, we wish to use learning to create adaptable and flexible circuits which can optimize themselves "on the fly", specially where input is not precisely defined when designing the system. A future goal may be to create a circuit that could adapt and function in all possible applications. But there is still a long way to go before the "one-circuits-fits-all" is a reality, so ICs manufactured today, which use the process of learning, needs an application specific implementation and predefined parameters.

There exists many different approaches to learning, and each are customized according to input, the environment and the desired accuracy on the output. Learning rules can be categorized as reinforced, supervised and unsupervised.

2.2.1 Reinforcement and Supervised learning

The core in reinforcement and supervised learning is to optimize a performance measure of a system according to feedback external to the system. For supervised training, the optimal output of the neural network $D(I,t)$, is known for a subset of inputs I, which is the training set of the learning algorithm. The performance measure $P(I,O,t)$, where O is the function to be optimized, is an error function where the goal is to minimize this measure P [29]. For reinforcement learning, the performance measure is more vague, e.g. the system is rewarded when certain outputs generates a wanted effect. The system will then tend to act the same way with the same set of inputs the next time.

2.2.2 Unsupervised learning

Unsupervised learning does not adapt according to external feedback. Instead, it adapt its weight W to the statistical properties of the input. The goal is to optimize the output, but unlike reinforcement and supervised learning, where the performance measure P should be minimized, unsupervised learning usually tries to optimize the representation of a huge amount of input data to a reduced set of output data. In the brain it is believed that unsupervised learning is used in this way, to compress and extract relevant data out of the huge amount of data that we perceive.

Many of the unsupervised learning rules that are currently being used today, is based on "a neurophysiological postulate" presented by D.O. Hebb [30]:

> *When an axon of cell A is near enough to excite a cell B and repeatedly or persistently takes part in firing it, some growth process or metabolic change takes place in one or both cells such that A's efficiency, as one of the cells firing B, is increased.*

2.3 The learning rule

The learning rule used in this thesis is based on a learning rule proposed by Häfliger in [29]. It lets synapses compete for synaptic strength while awarding causal relationship between inputs and outputs. It has the form:

$$\frac{d}{dt}w_i = w_i(\alpha\tilde{c}_i - \beta w_i O) \tag{2.1}$$

where w_i is the weight, α and β are learning parameters, $\tilde{c}_i$ is a correlation signal which reflects the activity level of the synapse, and O is the output of the synapse. The $w_i\alpha\tilde{c}_i$ term is computed with the *learn up circuit* presented in Section 3.4.1.2 and the $w_i^2\beta O$ term by the *learn down circuit* presented in Section 3.4.1.1.

2.4 The learning neuron

This thesis is related to the Convolution Address Event Representation (AER) Vision Architecture for Real-Time (CAVIAR) project [31], which is a collaboration between the Institute of Informatics, University of Oslo and three foreign participants, two resident in Sevilla, Spain and one in Zürich, Switzerland. The research project is funded by the IST Program of the European Union and its primary objectives are:

- To develop a general AER infrastructure for constructing bio-inspired hierarchically multi-chip systems for *sensing* + *processing* + *actuation*.
- To implement a particular perceptive-action demonstrator vision system exploiting this infrastructure.

The AER infrastructure is used in this thesis and a thorough presentation is given in Appendix A.

The vision system intended to be designed during the CAVIAR project is mounted on a stationary robot and will follow a specified object with its optical lens. To make it easier for the robot to follow this object, e.g. a ball, we want the robot to gradually learn to classify its trajectories. For instance, if the ball hits a wall, it will quickly change direction. This requires the robot to suddenly react and move its lens in the almost opposite direction with the same speed. If it can predict on beforehand that the ball bounces back when hitting a wall, the robot's movement will be smoother and will not require the same amount of computational power to perform the required action. If we think of humans, this is exactly how we work. If a child is playing with a ball for the first time, or watching other people playing with a ball, they will not be able to follow the ball with such ease as adults. An analogy may be to watch a ball in random movement. It is a tiresome view and requires concentration and stamina. We know that when a tennis player hits a ball, it bounces quickly back, and therefore we start to see the other way before the ball bounces off the player's racket. If we were not able to predict and to learn this behavior, not many people would bother watching.

The visual field of the robot is perceived by it as a two dimensional pixel array. Every time the ball is in a square, a spike is sent to the learning circuit. An illustration is seen in Figure 2.1. This would give a pair of x and y coordinates for a discrete period of time. Very little information can be drawn from this since there are no temporal connection between events. Therefore, we need to sample events over a period of time, and simultaneously send that spike train to the learning circuit. This is done using a delay line circuit. Such a circuit is designed, implemented and tested. The circuit is implemented on the same test chip.

The previously mentioned learning circuits, or neuron, will therefore receive a certain amount of input spike trains. If the same spatio-temporal spike pattern is presented to it several times, it should learn that pattern, and react to it every time by sending an AP. How fast it learns the pattern depends on the learning rule, the learning parameters and, of course, the implementation. On the other hand, when a neuron has

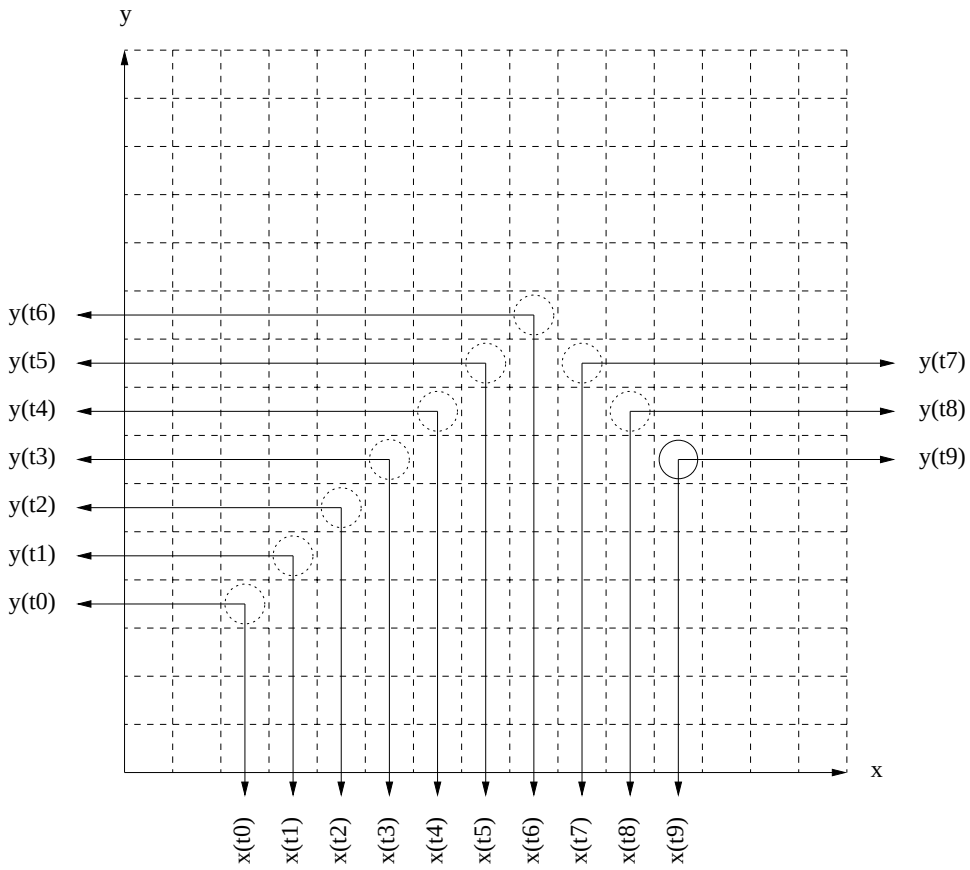

Figure 2.1: *Trajectory of ball in a two dimensional grid.*

learned a pattern, and a new and different pattern is presented, should it then learn the new pattern immediately or just ignore it? We want neither of the alternatives, but a combination of them. This is viewed as the stability-plasticity dilemma. The neuron should hold its internal states while stimulated by irrelevant pulses or noise, but still change its internal states quickly if conveyed by new and relevant input [32].

The network of neurons will use competitive Hebbian learning. Competitive learning networks learns by internally adjusting its weights based on the input and local feedback signals. This means that it requires no external feedback or guidance to adjust its weights. The network behaves as a classifier, where each neuron responds to an input vector which closely matches the weight vector of that neuron. This can be better explained by an example. Two neurons receive two inputs and store two weights where both have a value of either one or zero. If we have an input vector of [1,1] and weights for the two neurons [0,0] and [1,0]

respectively, the second neuron will win and adjust its weights to [1,1]. After learning, when we have the same input vector to the two neurons, neuron two will spike and neuron one remain inactive. Thus, the network works as an adaptive winner-take-all network, since one neuron will learn the input pattern stronger than others, or even hinder others from learning at all.

We will focus on the functionality and implementation of the learning neuron with the MLSM as synaptic weight, and describe and give test results of both its components and the neuron itself. We will also present test results from a small neural network consisting of only two neurons. The interaction between these two neurons can be expanded to an array of neurons. We have implemented such an array on the chip. This array consists of 32 neurons connected in parallel, and is used to test the AER infrastructure and to observe how the single learning neuron behaves in such a structure.

Chapter 3

Neuromorphic circuit components

We have given an introduction to the field of neuromorphic electronics and presented the objective of this thesis. In this chapter we will present the different circuit components. We start with the “fusing” transconductance amplifier, and continue hierarchically with each element until we have constructed the final learning neuron. All measurements are conducted on the single neuron if not stated otherwise.

3.1 “Fusing” transconductance amplifier

The central building block in the proposed MLSM is the “fusing” transconductance amplifier (“fusing” transamp). We will only give a brief introduction to the “fusing” transamp here. The details are presented in the enclosed article [33], where the “fusing” transamp and the MLSM are presented.

The “fusing” transamp consists of a normal transconductance amplifier and a so called “bump circuit”. This “bump circuit” only delivers a current if the two input voltages are close, i.e. within a range of about 100mV, and it provides the bias current for the transamp. Therefore, the “fusing” transamp only works as a transconductance amplifier for small differences of input voltages, while turning off if the voltages are too widely spaced. A schematic of the “fusing” amplifier can be seen in Figure 3.1 with its symbol in the upper right corner.

The “fusing” transamp is used to attract the weight to the stable weight levels of the MLSM. Several “fusing” transamps are connected in parallel, one for each stable level. We wish to minimize the attractor current to maximize the time for the weight to settle on a steady weight level.

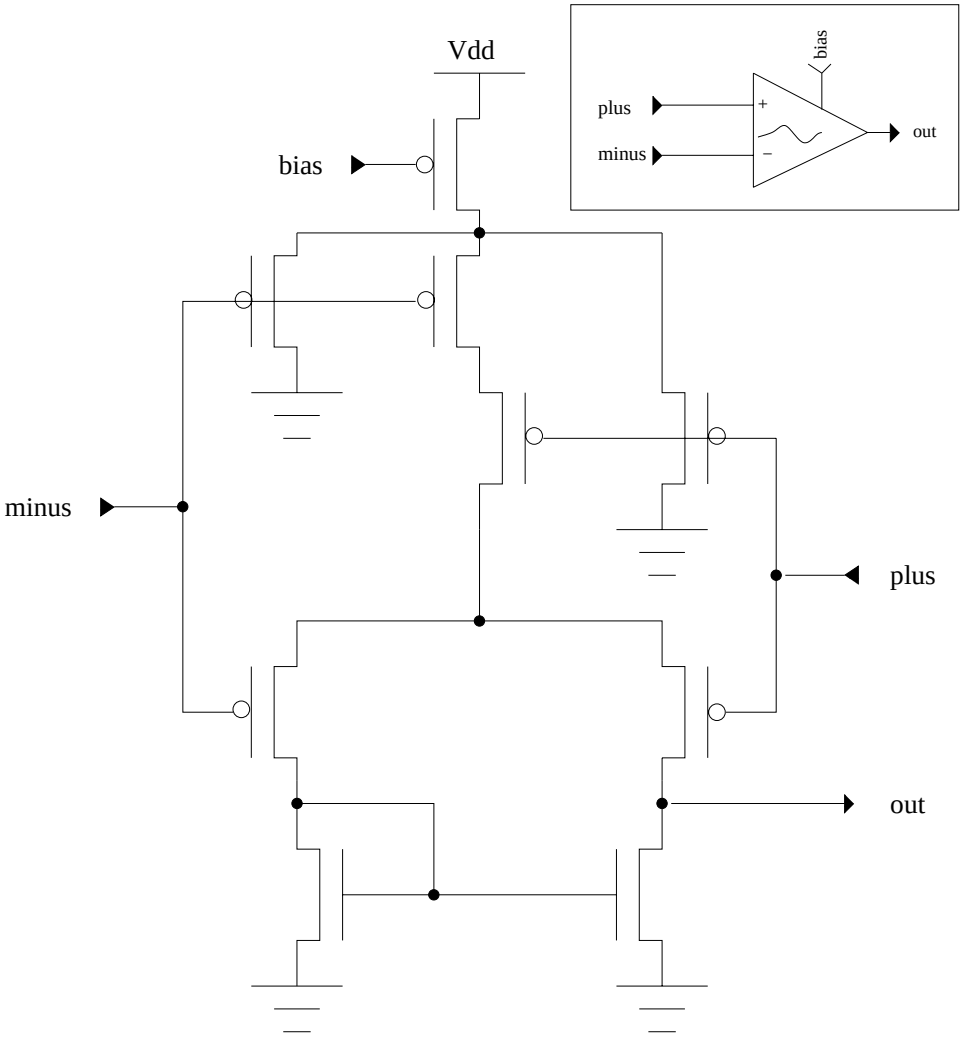

Figure 3.1: *A chematic of the "fusing" transconductance amplifier. The "bump circuit" consists of the five pMOS transistors above the transconductance amplifier. The "fusing" transamp's symbol is pictured in the upper right corner.*

Furthermore, we wish to minimize the range of the “fusing” transamp, such that the spacing between stable weight levels can be minimized.

Test results

In Figure 3.2, a plot of the current out of the “fusing” transamp is shown for five different biases measured on-chip.

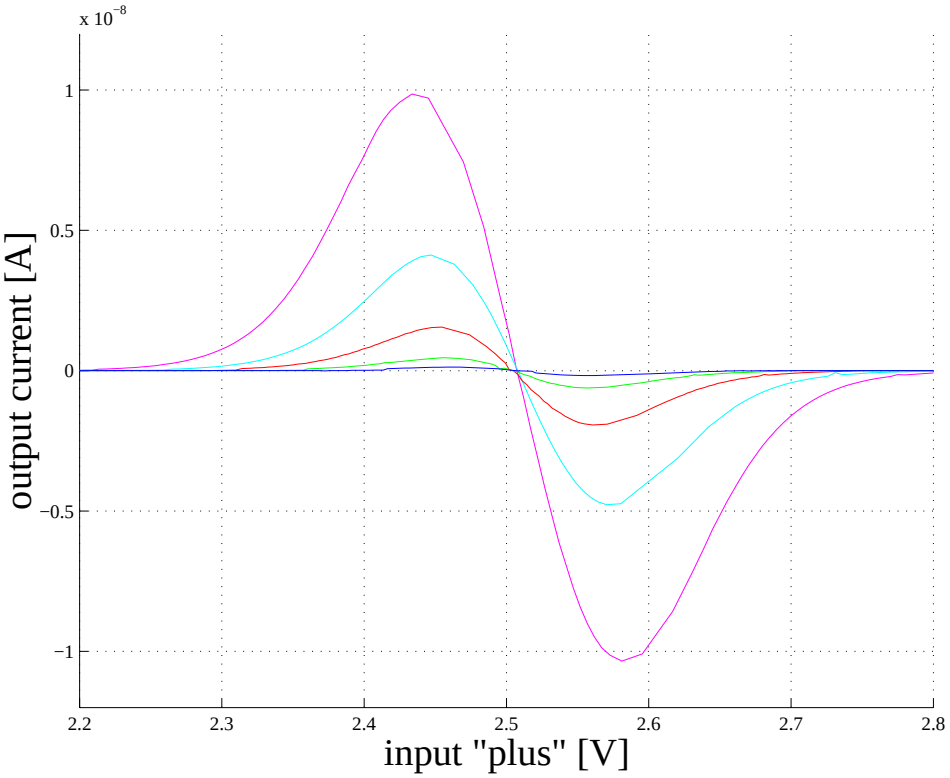

Figure 3.2: *Plot of output current from the output node of the “fusing” transamp for five different biases.* ***minus*** *is fixed at 2.5V and* ***plus*** *is sweeped from 0V to 5V. The maximum current increases for increased bias voltage, both in and out of the transamp. The distance between the two extrema along the input voltage axis also shows a slight increase for higher bias voltage. The biases where, from highest to lowest curve, 4.00V, 4.05V, 4.10V, 4.15V and 4.20V. Distance between maximum and minimum current at different biases are 147mV, 123mV, 106mV, 101mV and 95mV.*

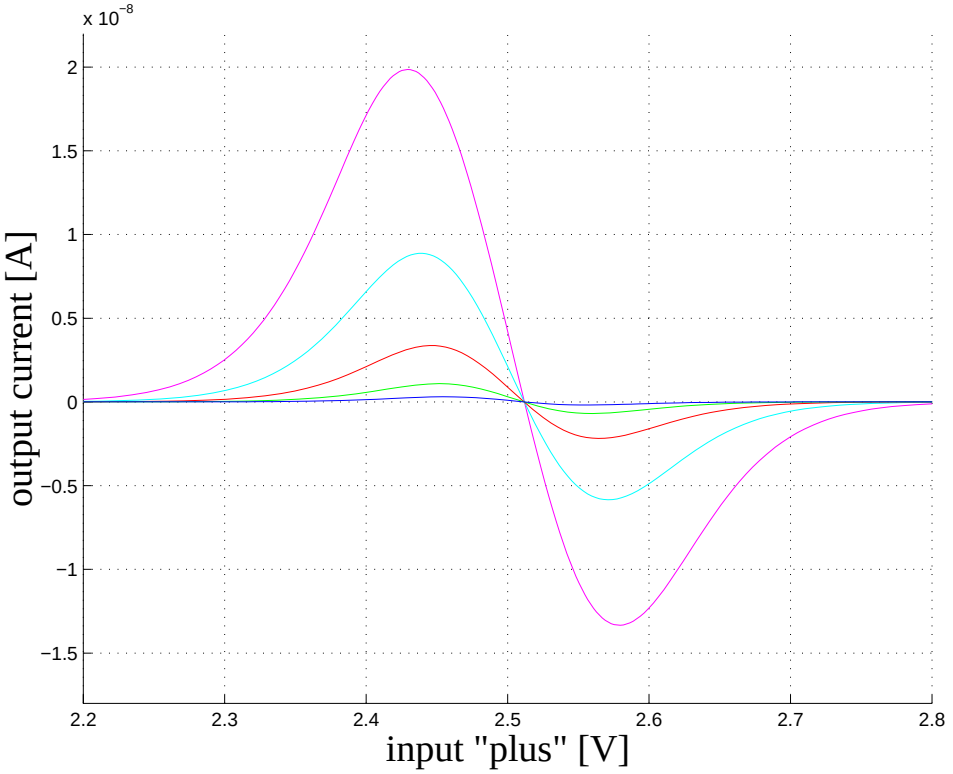

Figure 3.3: *Simulated response of the "fusing" transamp. Settings as in Figure 3.2.*

Compared to the simulation results from [33], depicted in Figure 3.3, we see that the current is somewhat smaller. This implies that the process parameters used in simulation were not exact. This change can be adjusted with the bias voltage. Furthermore, the current seems to be more symmetric than during simulation. The range of the amplifier is approximately the same for both plots.

It is clear that an increased bias voltage will both minimize attractor current and range of the "fusing" transamp. From simulations, we found that the optimal bias was 4.3V. Above this voltage, the "fusing" transamp could not deliver enough current to hold its stable weight level.

3.2 Multi-level static memory

As with the "fusing" transamp, we will only give a brief introduction to the MLSM and again refer to the paper enclosed. A schematic of the MLSM is plotted in Figure 3.4.

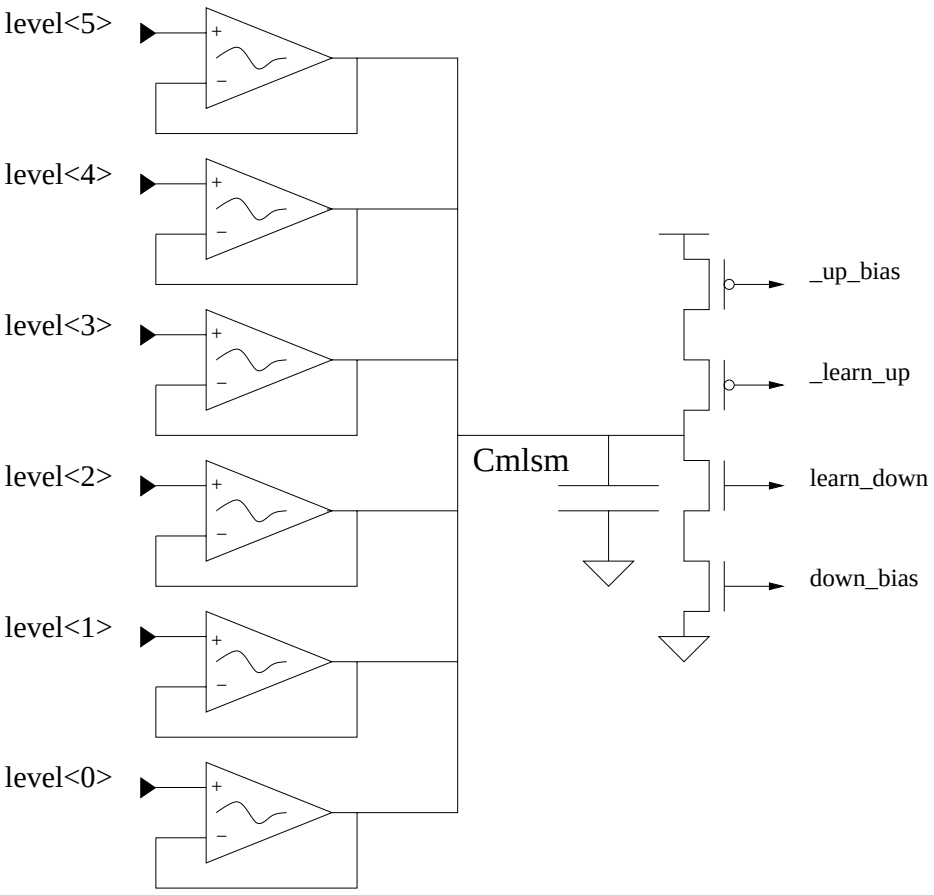

Figure 3.4: *A schematic of the MLSM.* ***level<1:5>*** *is set by a local voltage source. The memory capacitance* C_{mlsm} *stores the weight. The transistors on the right controls the increment and decrement of the weight.* ***_up_bias*** *and* ***down_bias*** *are external biases, while* ***_learn_up*** *and* ***learn_down*** *are controlled by the* ***learn up circuit*** *(Section 3.4.1.2) and the* ***learn down circuit*** *(Section 3.4.1.1), respectively.*

The MLSM consists of a capacitor which stores the actual weight and an arbitrarily amount of “fusing” transamps. In this thesis, we use six “fusing” transamps, which are set up as voltage followers. Therefore, they will compete among eachother to attract the weight on the capacitor. The input voltages to the voltage followers are set by voltage sources. We have chosen to produce these voltage levels locally, to prevent extensive routing. It consists of eight so called “Toby elements”, which are diode connected pMOS transistors, with the bulk tied to the source. This gives nine evenly spaced voltage levels from upper level to ground, where we use the upper six levels as voltage sources to the “fusing” transamps. The weight value is adjusted by current sources set by internal control signals. A schematic of the voltage supply can be seen in Figure 3.5.

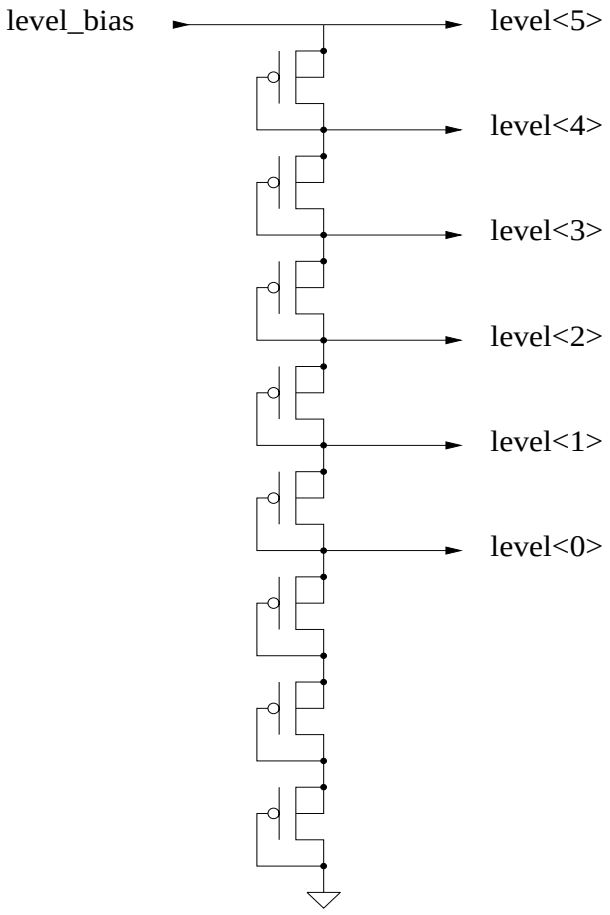

Figure 3.5: *A schematic of the memory voltage supply.*

Test results

As we used six “fusing” transamps, we simulated with six discrete voltage levels. After early simulations, we decides to work with stable weight levels between 1.10V and 413mV. It turned out that the results of testing the MLSM implemented on-chip would be different. When measuring the stable weight levels on the memory capacitance, we only achieved six discrete voltage level for a *level_bias* of 1.4V and five levels for 1.3V and 1.2V (see Table 3.1).

Simulation(V)	Measurements(V)		
1.10	1.40	1.30	1.20
963m	1.11	1.00	930m
825m	800m	710m	620m
688	515m	430m	340m
550m	250m	170m	165m
413m	167m	*	*

Table 3.1: *Stable weight levels used during simulations and measurements.*

This is clearly a setback, since we designed to work with six discrete levels. And since we are supposed to operate in the sub threshold area, an upper stable weight level of 1.4V is not desirable. We came to the conclusion that, regardless of the loss, a *level_bias* of 1.2V was the best solution under the given conditions, since the neuron later turned out to behave as we wanted.

The reason for the difference between simulation and testing is not completely clear. The voltage levels are created with diode connected Toby elements, which would give evenly spaced voltage levels. We used eight elements, using only the upper six for voltage sources to the “fusing” transamp. The lower diodes are used to decrease the voltage span between the individual weights and increase the lowest stable weight level. Furthermore, the lowest stable weight level measured on chip for an arbitrarily *level_bias* is approximately the same; 160-170mV. So it is more likely that it is the “fusing” transamps that cause the shift in stable weight levels. Since the range of the “fusing” transamp between the two extrema is about 100mV for a transamp bias of 4.3V, and the spacing between weights during simulation was approximately 150mV, there will exist some overlap of attractor currents. This can be seen in Figure 3.6. Also, offset currents in the transamp exist. This will changes the location where the output current is zero from all “fusing” transamps. In Figure 3.7, the sum of all currents out of the “fusing” transamp is plotted.

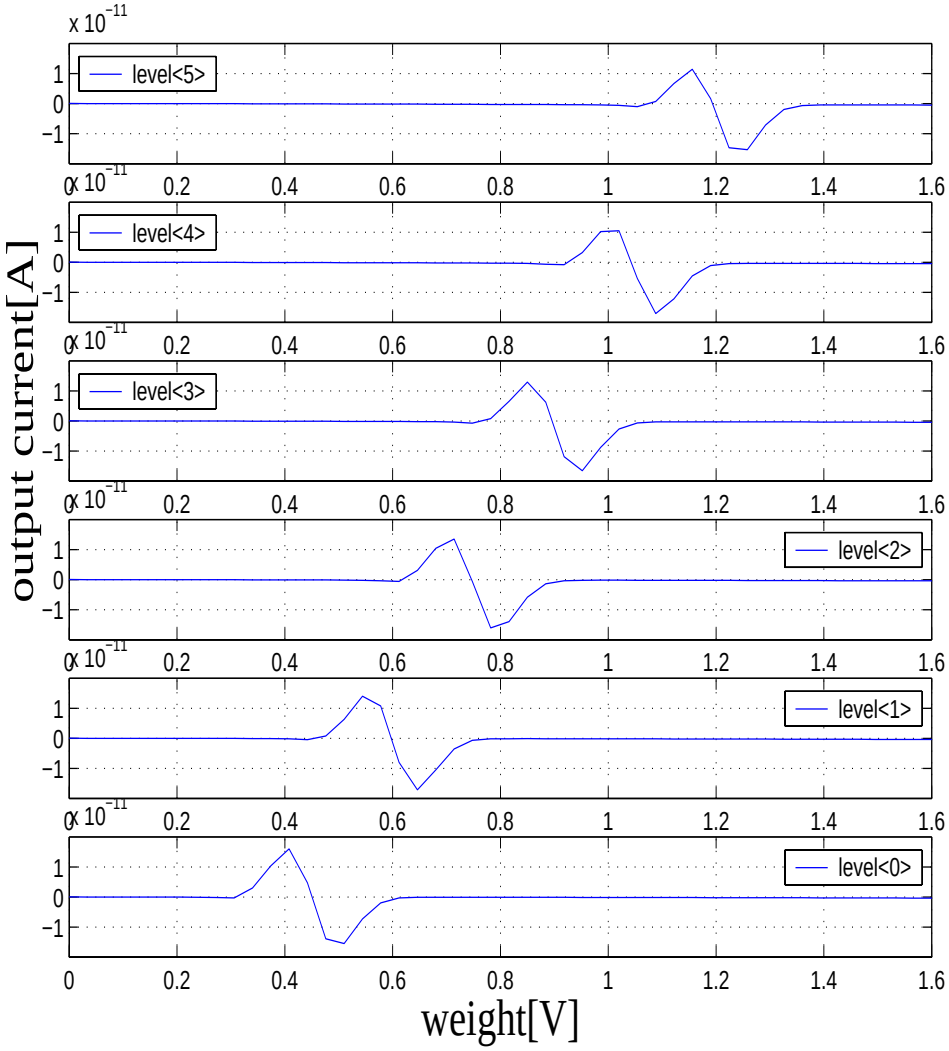

Figure 3.6: *Output current from all six memory "fusing" transamps for* ***level_bias****=1.2V. Transamp bias is 4.3V. The overlap is substantial and neighboring transamps will affect eachothers behavior.*

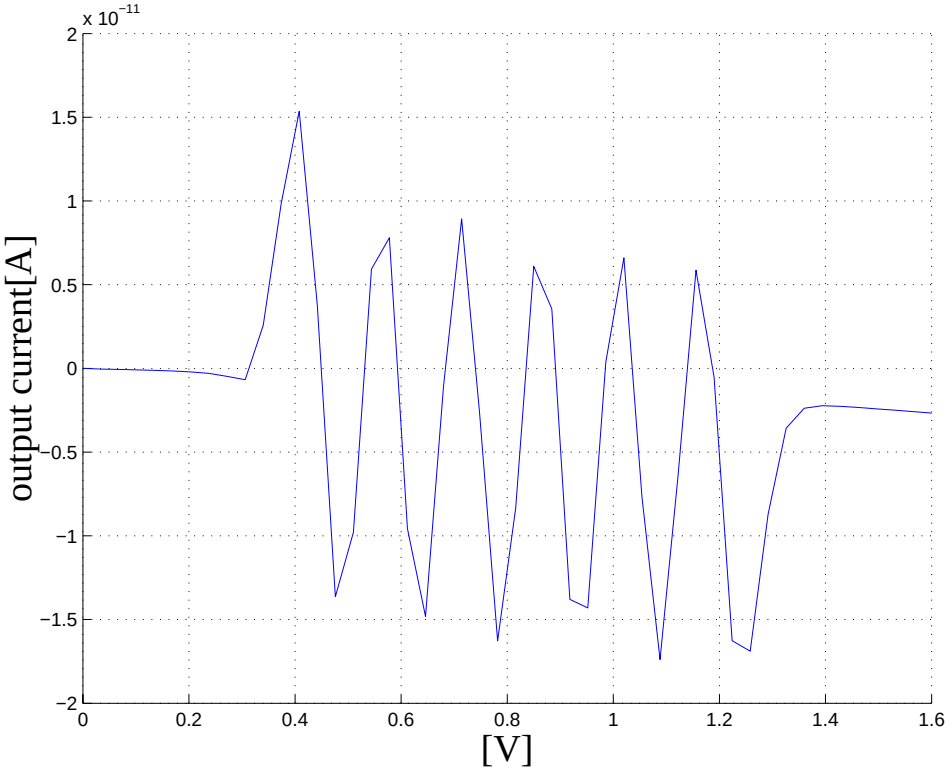

Figure 3.7: *Sum of all currents from the six memory "fusing" transamps. Zero crossing with negative gradient are stable weight levels. Other zero crossing points are non-attractive fixed points, since neighboring transamps compete and one transamp will attract the voltage to its stable state. Compared to the trace of single currents from each transamp in Figure 3.6, we see that the stable levels are slightly shifted down wards. There is very little difference in stable levels though, compared to the change experienced in measurements on-chip. However, on-chip, the offset voltage is probably bigger and the transamp has a slightly different characteristic, as described in the previous section. This may be the reason that the change in stable weight levels are greater on-chip than during simulations.*

In Figure 3.8, a plot of the basins of attraction of the memory cell in a circuit simulation from [33] can be seen. We performed a similar test on-chip, which is plotted in Figure 3.9.

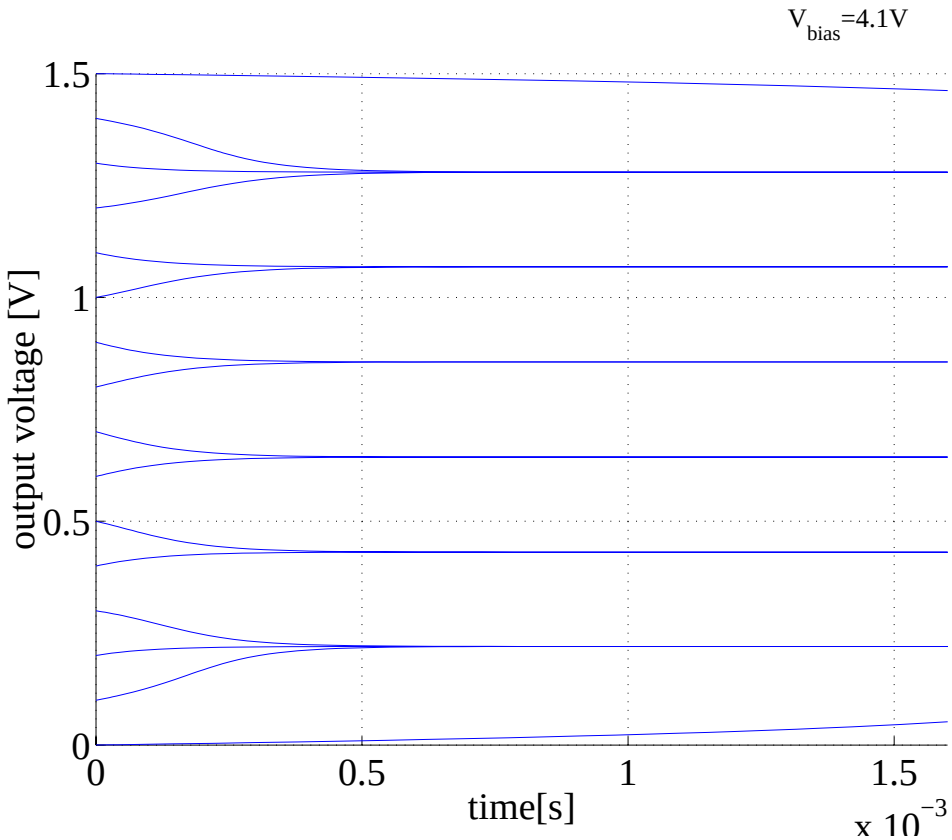

Figure 3.8: *The simulated time to attract a voltage to a stable level is found to be approximately 0.5ms.*

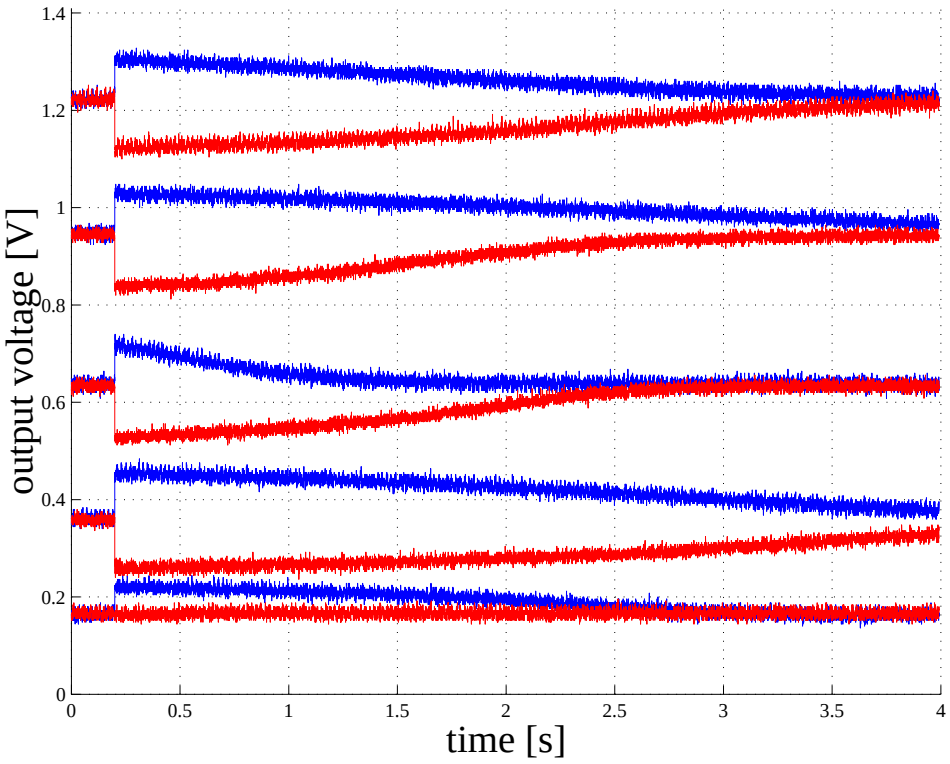

Figure 3.9: *Plot of voltage attractors for* ***level_bias****=1.2V. The transamp bias is 4.3V. The attractors work nicely and the time for the stable weight levels to settle after injection or removal of current can be as long as four seconds. We observe that a small increase in the voltage for the lowest weight will cause a jump to the next level. This can be a positive feature, since we experienced some reduced functionality with the learning, as described in Section 3.4.1.1.*

3.3 The soma

The soma is responsible for integrating the current from the individual synapses. The soma implemented is based on the integrate-and-fire model proposed by Mead [2]. In this thesis, the learning synapses (section 3.4.1) and the excitatory synapse (Section 3.4.3) draws a current from the soma while the inhibitory synapse (Section 3.4.2) injects a current. How much current that is removed or injected, is controlled by the weights of the learning synapses and external biases, *w+* and *w-*, for the excitatory and inhibitory synapses, respectively. Thus, the resting potential for the soma is at V_{dd}, while it triggers and AP for approximately $V_{dd}/2$, which is the switching point of the inverter. If we look at the schematic of the soma in Figure 3.10, the current through the transistor controlled by the *_ap_leak* adds charge to C_{soma}, and so pulls the voltage V_{soma} up.

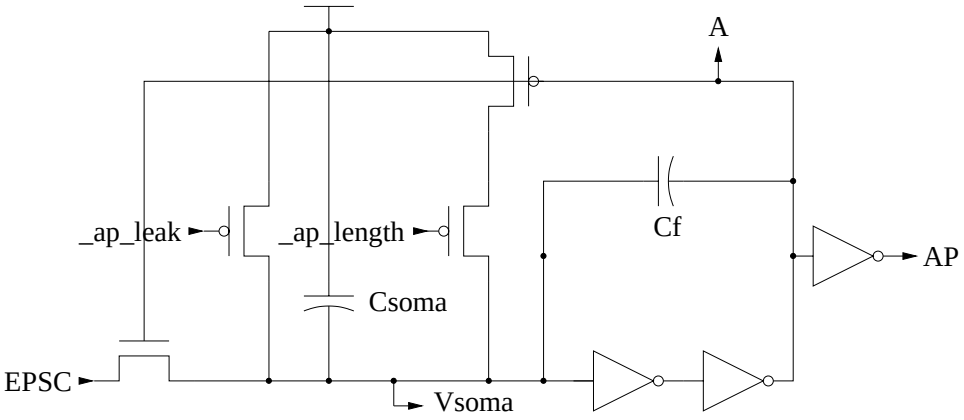

Figure 3.10: *Schematic of the soma.*

When sufficient current has been removed through the leftmost nMOS transistor such that $V_{soma} < V_{dd}/2$, the AP goes high, while node *A* goes low. This causes a voltage drop over the capacitative feedback to the soma. How big this voltage drop is, depends on the size of the capacitors C_f and C_{soma} . According to the capacitative division, the voltage change on V_{soma} is

$$\delta V_{soma} = \delta V_A \frac{V_f}{V_f + C_{soma}} = \delta V_A \frac{V_{feedback}}{C_{total}} \tag{3.1}$$

We implemented the two feedback capacitors such that they would cause the charge on C_{soma} to reach its rails, both V_{dd} and V_{ss}, e.g. $C_f = C_{soma}$. The length of the AP is controlled by the *_ap_length* bias. We have used an AP length of about 2ms in this thesis. When the charge on C_{soma} rises back to the switching point due to the current through the *_ap_length*

transistor, the AP goes low, and node A goes high. In the same manner as before, the capacitative feedback causes the charge on C_{soma} to reach V_{dd}.

Test results

Figure 3.11 shows a plot of the current removed from the soma by the learning synapses at different weights.

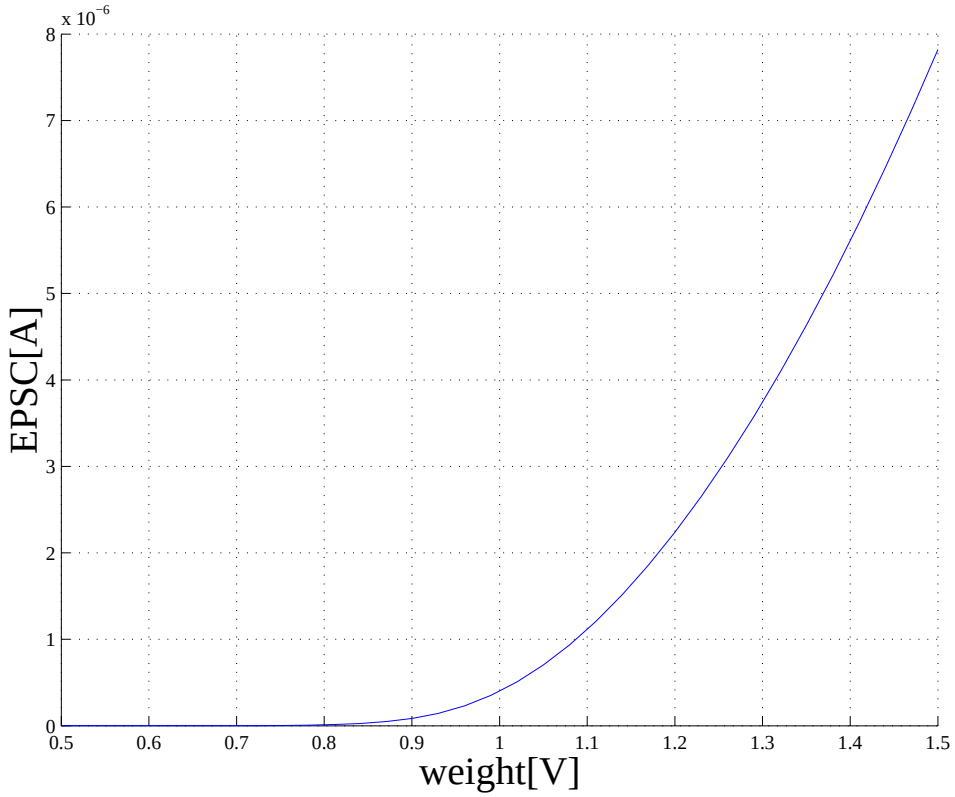

Figure 3.11: *Current removed from the soma for different weights in the learning synapse. The current has a fine exponential characteristic, but it remains approximately the same for the lower weights.*

Previous work with the same soma, showed that due to parasitic capacitances, the capacitative feedback was not sufficient, where $\delta V_{\text{soma}} = \delta V_{\text{A}} \times 0.5$. Thus the actual change is

$$\delta V_{\text{soma}} = \delta V_{\text{A}} \frac{C_{\text{feedback}}}{C_{\text{f}} + C_{\text{soma}} + C_{\text{parasitic}}} \tag{3.2}$$

Therefore we increased the size of C_f such that $\delta V_{soma} = \delta\ V_A \times 0.77$. This should give a change on V_{soma} of 3.85V since the change on V_A is 5V. Still the charge on C_{soma} was not pulled to the rails. Figure 3.12 and 3.13 plots traces of the voltage on the somatic capacitance C_{soma} measured on-chip.

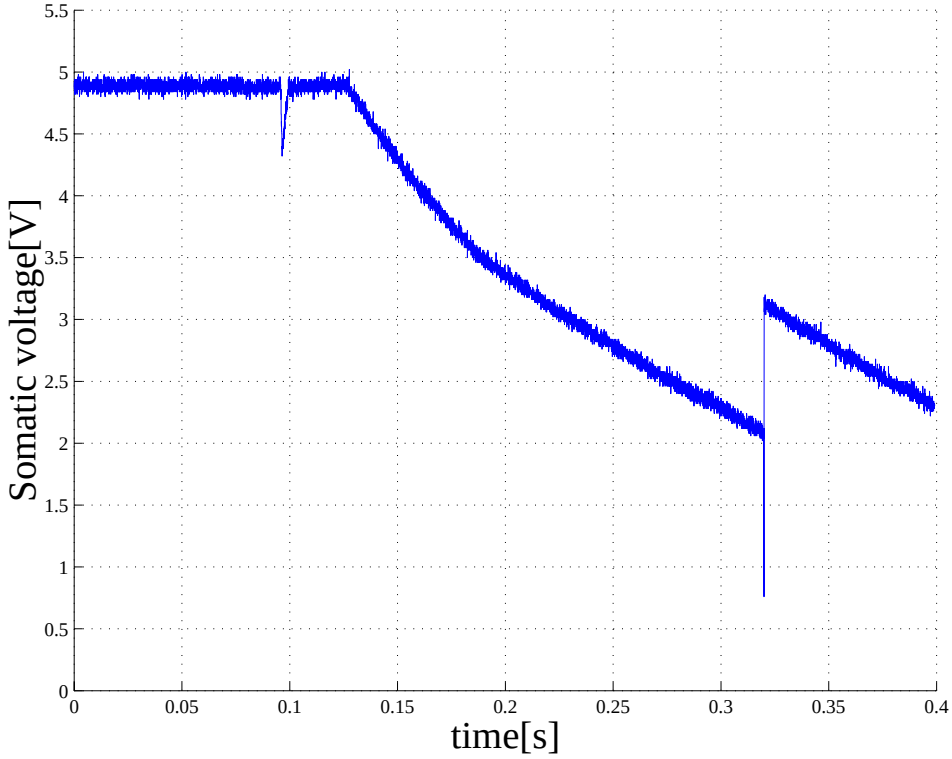

Figure 3.12: *Trace of the somatic voltage. The voltage is pulled down to around 2.1V where the inverter switches and the soma sends an AP.*

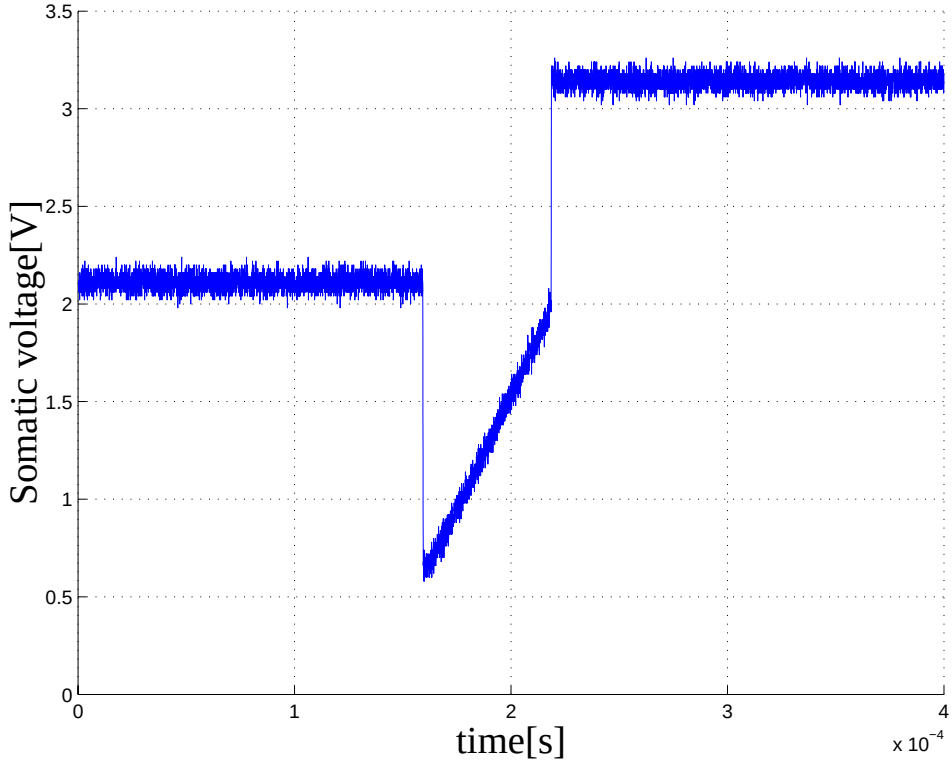

Figure 3.13: *Trace of the the somatic voltage with a higher time resolution. The voltage is pulled down to about 0.7V and up to about 3.2V. This is approximately a change of 1.5V for both cases. The ratio of the feedback capacitance to the total capacitance is 0.30, as compared to the theoretical calculation of 0.77.*

3.4 Synapses

3.4.1 The learning synapse

The learning synapse seen in Figure 3.14, is the main part of the test object in this thesis. It is derived from the synapse with FG storage [29]. It consists of a *learn up circuit*, a *learn down circuit* and the proposed MLSM. The EPSC output seen to the right, goes to the soma, where the AP is initiated. Each element has several biases, which are listed in Table E.1 in Appendix E. The extra buffer between the *_learn_up* output and the *_up* input to the MLSM, makes the *_learn_up* pulse sharper. The voltage follower before the *W* input on the *learn down circuit* will be discussed in the following section.

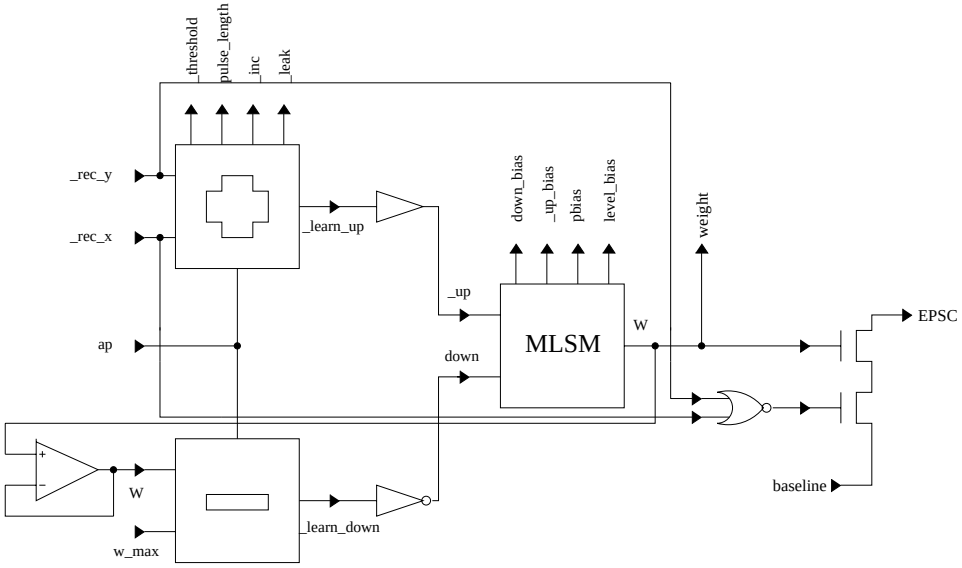

Figure 3.14: *A schematic of the learning synapse. The plus box is the* ***learn up circuit*** *and the minus box the* ***learn down circuit****. The weight W of the MLSM is used to control the current to the soma and to set the length of the down pulse.*

3.4.1.1 The learn down circuit

A schematic of the *learn down circuit* is shown in Figure 3.15. The circuit determines the length of the *_learn_down* pulse to the MLSM. The length of the pulse, T_{down}, is to be proportional to the weight in the specific learning synapse. When its idle, while no APs are sent from the neuron, the upper pMOS transistor is closed, and the capacitor C_{down} is set to V_{dd}.

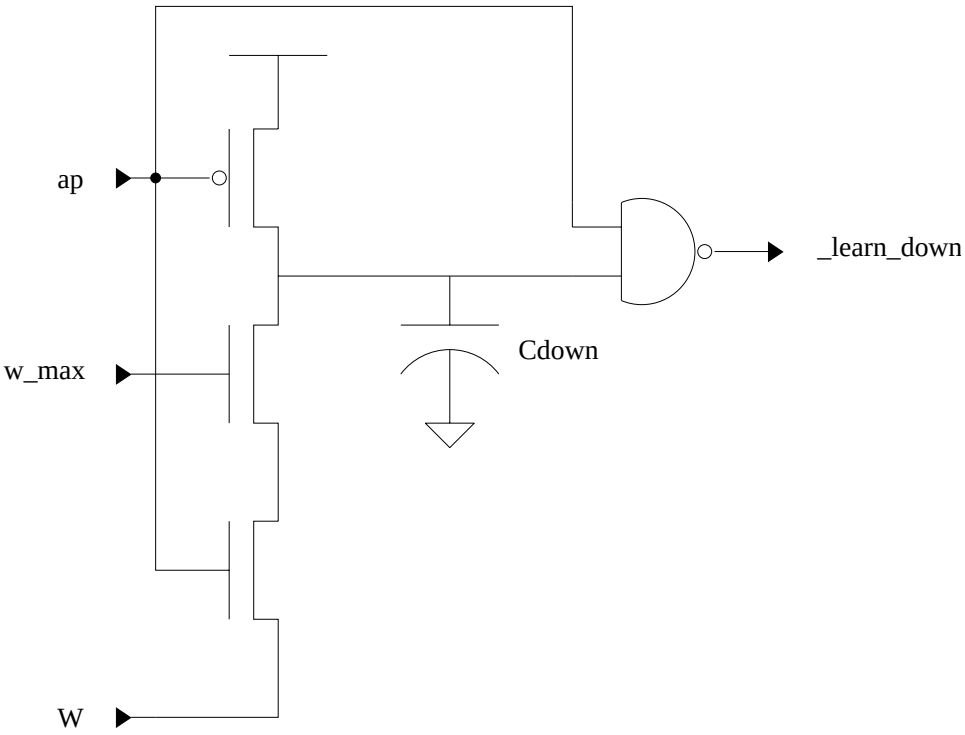

Figure 3.15: *A schematic of the learn down circuit.* ***w_max*** *is an external bias*

The output of the NAND gate, *_learn_down*, is then high, which means inactive. When an AP is initiated, the upper pMOS transistor is open, and the charge on C_{down} starts to fall since there goes a current I_{down} through the lower nMOS branch. If $I_s = 2n\beta {U_t}^2$, then

$$I_{down} = I_s e^{\frac{1}{U_t}(V_{\text{w\_max}} - V_w)} \tag{3.3}$$

T_{down} is therefore the time for the voltage on C_{down} to reach the switching point of the NAND gate (approximately $V_{\text{dd}}/2$).

$$T_{down} = \frac{(V_{\text{dd}}/2)C_{\text{down}}}{I_{down}} = \frac{(V_{\text{dd}}/2)C_{\text{down}}}{I_s e^{\frac{1}{U_t}V_{\text{w\_max}}}} e^{\frac{1}{U_t}V_w} \tag{3.4}$$

From equation 3.4, we see that there is an exponential relationship between the weight and the length of the down pulse.

The decrement of the weight on the memory cell can be calculated:

$$\delta w^- = \frac{1}{C_{\text{mlsm}}} \int_0^{T_{down}} I_{rem}\, \partial t = \frac{T_{down} I_{\text{rem}}}{C_{\text{mlsm}}} \tag{3.5}$$

where I_{rem} comes from a current source located in the memory element controlled by the *learn_down* bias.

Test results

Contrary to the theoretical calculations above, we saw from simulations (Figure 3.16) that the pulse length is only exponential for the upper half of the voltage range and does not distinguish between the lower weights. This is not a major problem, but ideally, it should be increasingly difficult to increments the weight the higher the weight is.

During early simulations with the design without the follower in Figure 3.14, we experienced that current from the learn down capacitor C_{down}, gave a positive jump on the MLSM capacitance C_{mlsm}. This caused low weights to rise a bit although they where supposed to decrease, even though the size of C_{down} is about 1/100 of the MLSM capacitance. We therefore placed the voltage follower in front of the weight input W to the *learn down circuit* to prevent a current flow to the MLSM capacitance.

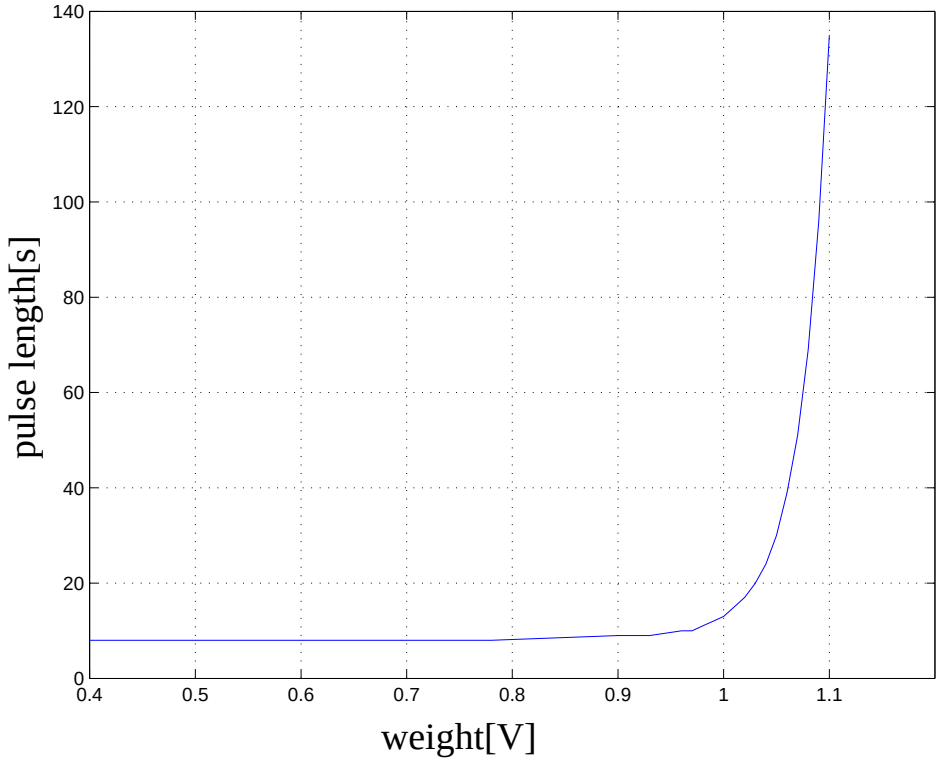

Figure 3.16: *Length of the down pulse for the learn down circuit.*

The down signal for two synapses at different stable weight levels measured on-chip, can be seen in Figure 3.17 and 3.18. The length of these pulses vary for the different synapses, but this is mostly due to the differences in the actual stable weights levels for the two synapses, than the learn down circuitry (see Table 3.2 in Section 3.5). As seen from the plot, the length of the down pulses are as during simulation: Exponential decay for high weights, while there is very little or no difference in pulse length for lower weights.

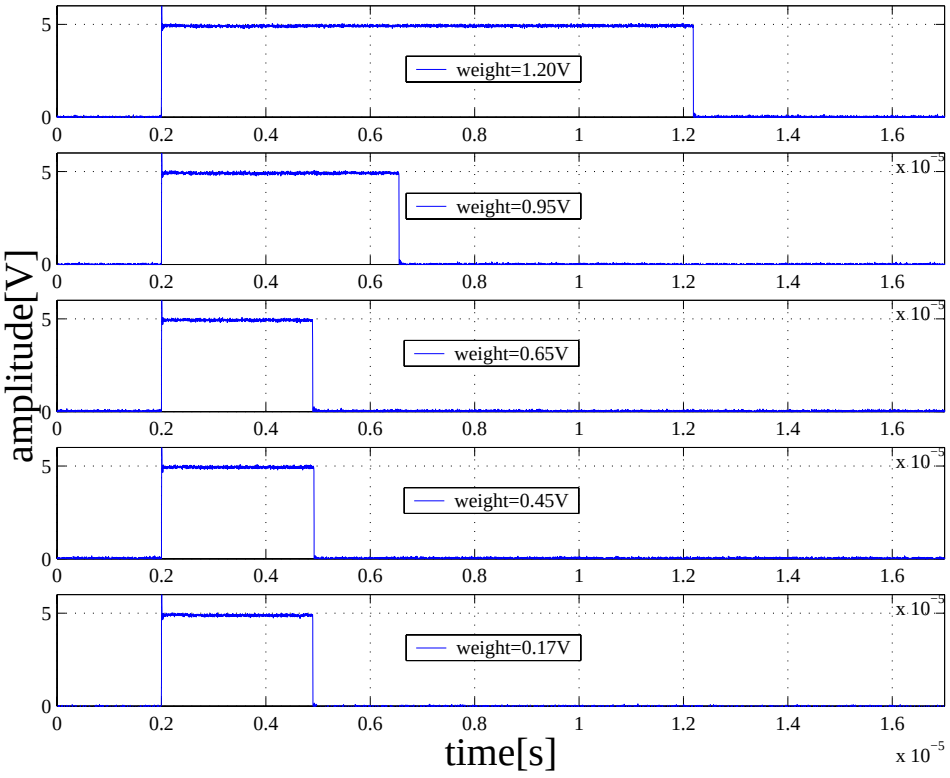

Figure 3.17: *Length of the down pulse for synapse one at different stable weight levels in single neuron.*

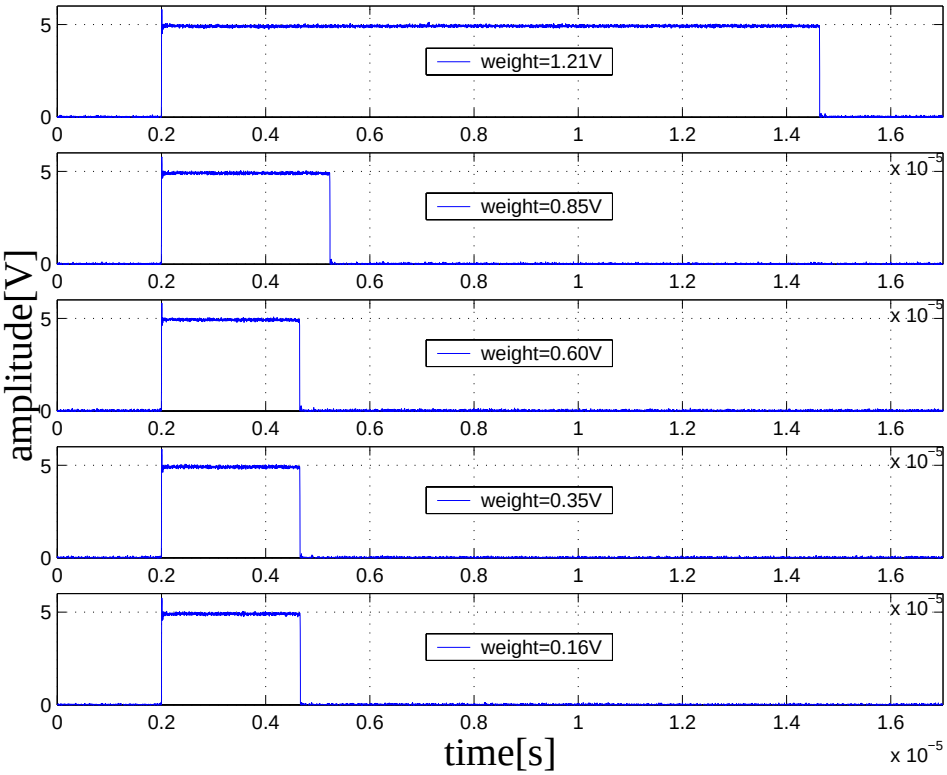

Figure 3.18: *Length of the down pulse for synapse four at different stable weight levels in single neuron.*

3.4.1.2 The learn up circuit

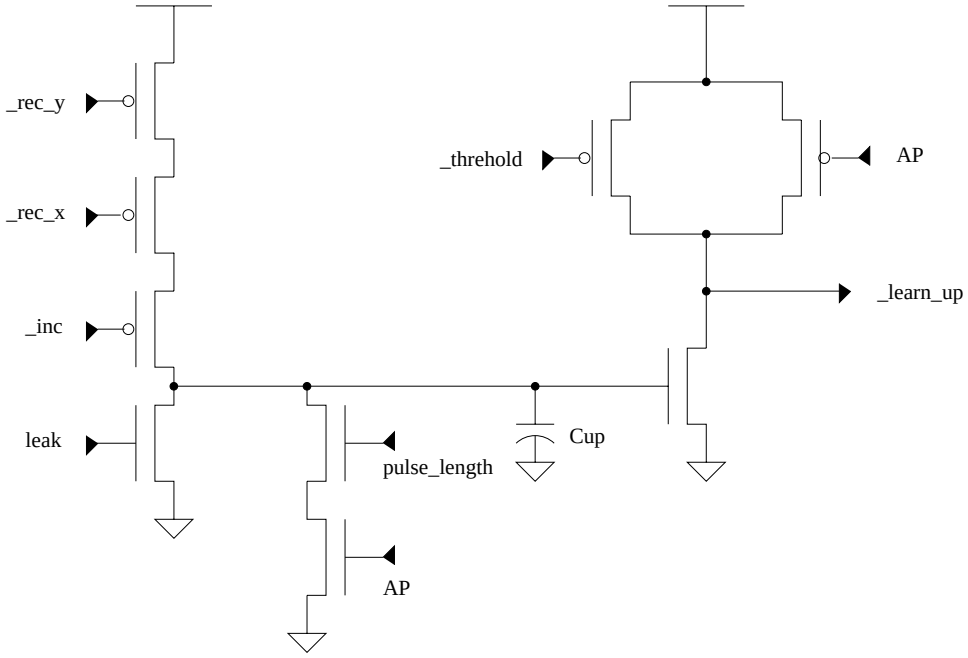

Figure 3.19: *A schematic of the learn up circuit.*

The learn up circuit is, as with the learn down circuit, responsible for adjusting the weight of the MLSM. But instead of using the weight as a variable, the input activity to the learning synapse determines the increment in the weight. From Figure 3.19, one can see that for every input (*_rec_x* AND *_rec_y*), an internal "correlation signal", or the charge on the capacitor C_{up}, is incremented. The value of the increment is controlled by the *_inc* bias. When there are no input spikes present, the *leak* bias controls the leakage from the capacitor. How large this leakage is, depends on which learning algorithm that is used. In this thesis, a spike based learning rule is used, not a rate based learning rule. Therefore, the leakage is relatively high, and the charge on C_{up} is removed within 1-2ms. Since the value of the correlation signal determines the length of the *_learn_up* signal, this clearly establishes a temporal dependence between input spikes and APs. If the spike present is the one that triggers an AP, then the synapse is awarded and can increment its weight. If some spike prior or about 3ms later triggers an AP, the temporal dependence is no longer and the weight is not incremented. If, on the other hand, a rate based algorithm was to be used, the time to remove the charge from C_{up} must increase. This will result in a correlation signal dependent on several input spikes, hence frequency.

If we analyze the circuit, we see that the current through the rightmost nMOS transistor I_{corr}, must be larger than the upper rightmost pMOS branch. When the AP is low, this is not true. But when the AP is high, the current I_{corr} must be larger than the current through the pMOS transistor controlled by *_threshold*, I_{thresh}, to ensure that the output *_learn_up* goes active low. We try to keep this current as close to zero as possible. The duration of the up pulse is then approximately the same as the time it takes to remove all the charge on C_{up}, T_{up}, through the reset branch controlled by *pulse_length* and AP.

$$T_{up} = \frac{V_{corr}}{I_{reset}} C_{up} \tag{3.6}$$

We find the increment of the weight in the memory cell:

$$\delta w^{+} = \frac{1}{C_{\text{mlsm}}} \int_{0}^{T_{up}} I_{inj}\, \partial t = \frac{I_{\text{inj}} T_{up}}{C_{\text{mlsm}}} \tag{3.7}$$

where I_{inj} is applied by a current source located in the memory element controlled by the *_up_bias*.

Test results

Due to mismatch, there are some differences in amplitude and length of the correlation signal for the different synapses as seen in Figure 3.20. In tests so far, we made the pulse T_{up} to behave approximately binary by having I_{reset} smaller than I_{leak}. If there was resent presynaptic activity, T_{up} would be equal to the duration of AP, without recent presynaptic activity T_{up} would be zero.

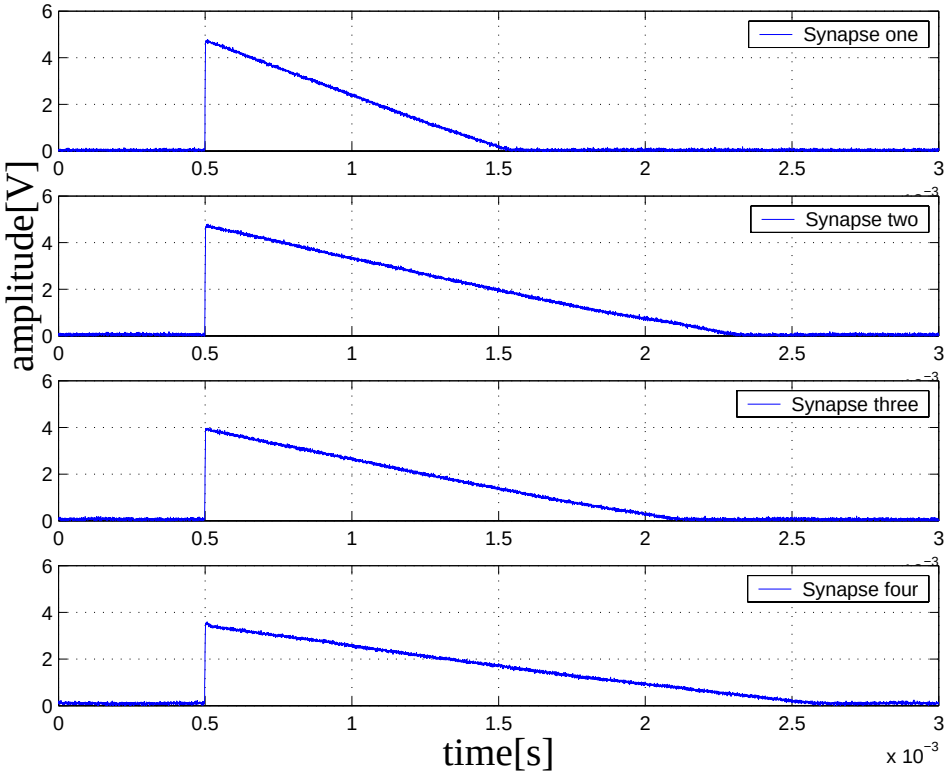

Figure 3.20: *Correlation signal for synapses one to four measured on chip when stimulated with an AP.*

3.4.2 The inhibitory synapse

The inhibitory synapse injects a current to the soma when stimulated. Thus it tries to prevent the soma from firing an AP. In a neural network, this property can be used to prevent neighboring neurons from learning the same pattern. This is done by connecting its neuron's AP to the inputs of its neighboring neurons inhibitory synapses. When then stimulated with a certain input pattern, the neuron that spikes first, will hinder others from spiking, thus not be able to learn, partially or completely, the input pattern. This behavior is shown in Section 3.5.

We see in Figure 3.21 that the external bias *w-* controls the amount of current injected to the soma, while *shunt_duration* determines the duration of the injection.

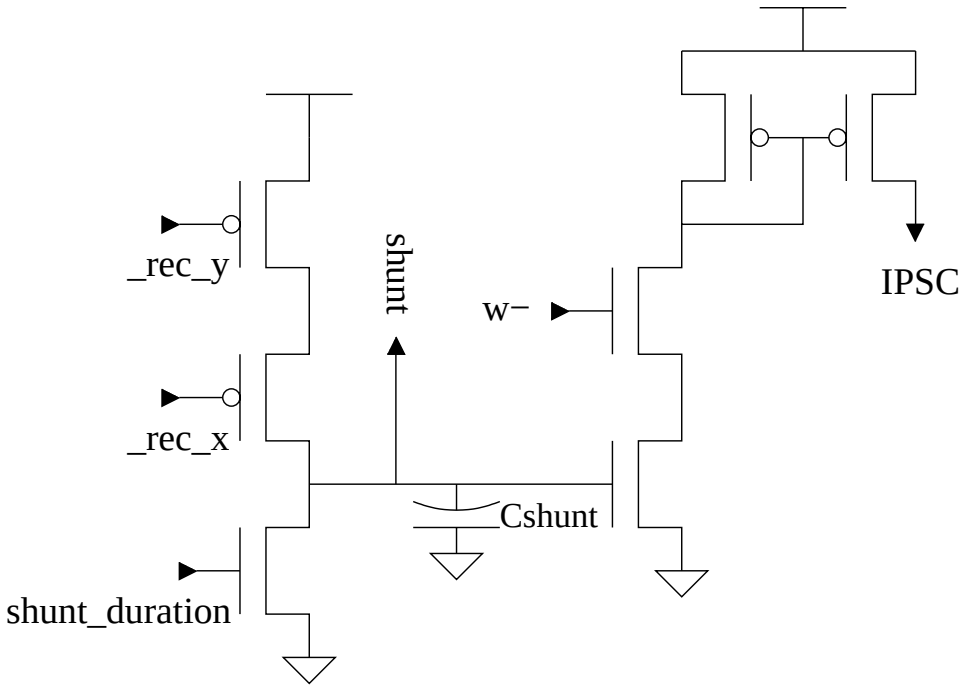

Figure 3.21: *A schematic of the inhibitory synapse.*

3.4.3 The excitatory synapse

The excitatory synapse, seen in Figure 3.22, removes a current from the soma. How much current that is removed is controlled by the *w+* bias. In the same manner as the inhibitory synapse, this feature can be of use in a neural network. The AP is connected with its neighboring neurons excitatory inputs, in this way helping each other to learn similar patterns.

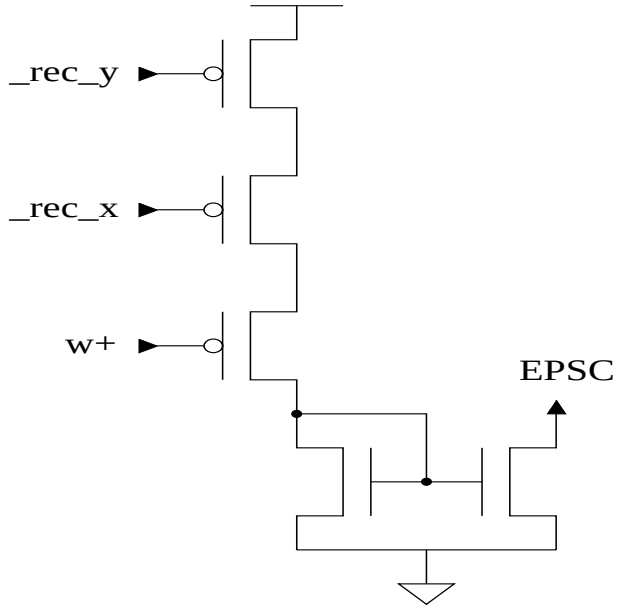

Figure 3.22: *A schematic of the excitatory synapse.*

3.5 The neuron

We have in the previous sections described all the necessary components for which is needed to construct the neuron. In this thesis, we have decided to use six synapses: Four learning synapses, one inhibitory and one excitatory synapse. Additionally, the soma is needed to complete the structure of the neuron. In Figure 3.23, a schematic of the neuron is depicted.

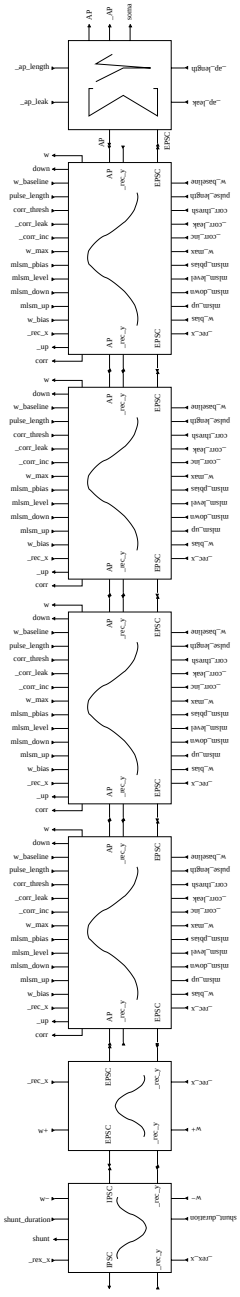

Figure 3.23: *A schematic of the neuron implemented. From bottom, the inhibitory synapse, the excitatory synapse, four learning synapses and finally the soma. Each elements has its own biases connected at the top and bottom (left and right in figure) so that they can easily be connected forming a multiple array. Common signals are situated at the side such that the signals can be distributed to all elements and to make it possible to construct a matrix of neurons.*

Test results

All measurements are conducted with a *level_bias* of 1.2V and a "fusing" transamp bias of 4.3V. Other parameters are adjusted to fit the different experiments based on input frequency.

In Table 3.2, we list the attractive stable weight levels measured on-chip in the four learning synapses in the neuron.

Synapse 1	Synapse 2	Synapse 3	Synapse 4
1.20V	1.18V	1.19V	1.21V
975mV	913mV	930mV	895mV
675mV	645mV	641mV	625mV
408mV	355mV	418mV	365mV
182mV	150mV	150mV	160mV

Table 3.2: *Attractive weight levels for neuron.*

The task set to the neuron is to learn spike patterns and then react with sending an AP when such a pattern is presented to it. After learning the input pattern, when a different spike pattern is given, it should not trigger an AP.

First, we tested the neuron on-chip with two different spike patterns. To simplify, the spike patterns consisted of only one spike. The first pattern stimulates learning synapse one and the second pattern stimulates learning synapse two. We would expect the neuron to learn one of the patterns and depress the other. For the test to work properly, we set both synaptic weights to a maximum. The neuron would then, at first, trigger an AP for each stimulation, thus activating the learning. This means that the neuron already have learned both patterns and we actually monitor if the neuron can depress one pattern while maintaining the other. Different traces can be seen in Figure 3.24, 3.25 and 3.26.

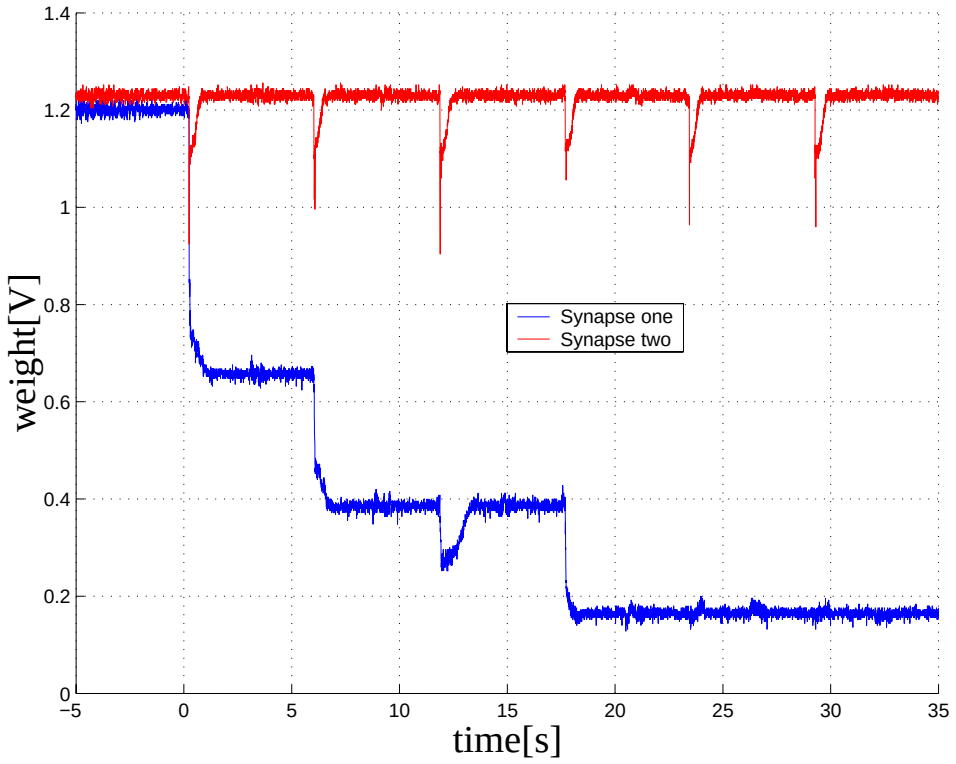

Figure 3.24: *Trace of weights for learning synapse one and two. The individual spikes for a spike pattern have a spacing of approximately 5s, where spike pattern synapse one is delayed 2.5s. We observe that the neuron depresses spike pattern synapse one and maintains spike pattern synapse two.*

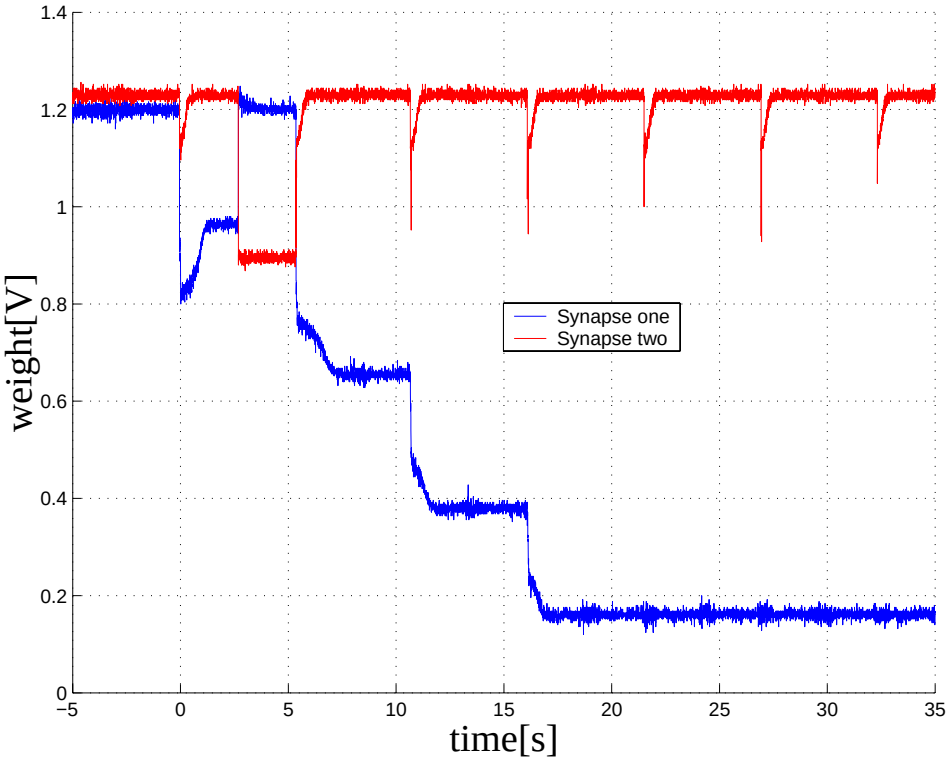

Figure 3.25: *Trace of weights for learning synapse one and two. Spike patterns as in Figure 3.24. Again the neuron depresses spike pattern synapse one and maintains spike pattern synapse two. But unlike the previous trace, we see that spike pattern synapse one is not depressed immediately and manages to increase its weight to the upper level when stimulated the first time. At the same time it decrements the weight of synapse two. When synapse two is stimulated again, it decrements the weight of synapse one such that the neuron does not trigger an AP when spike pattern synapse one comes next. Therefore depressing spike pattern synapse one.*

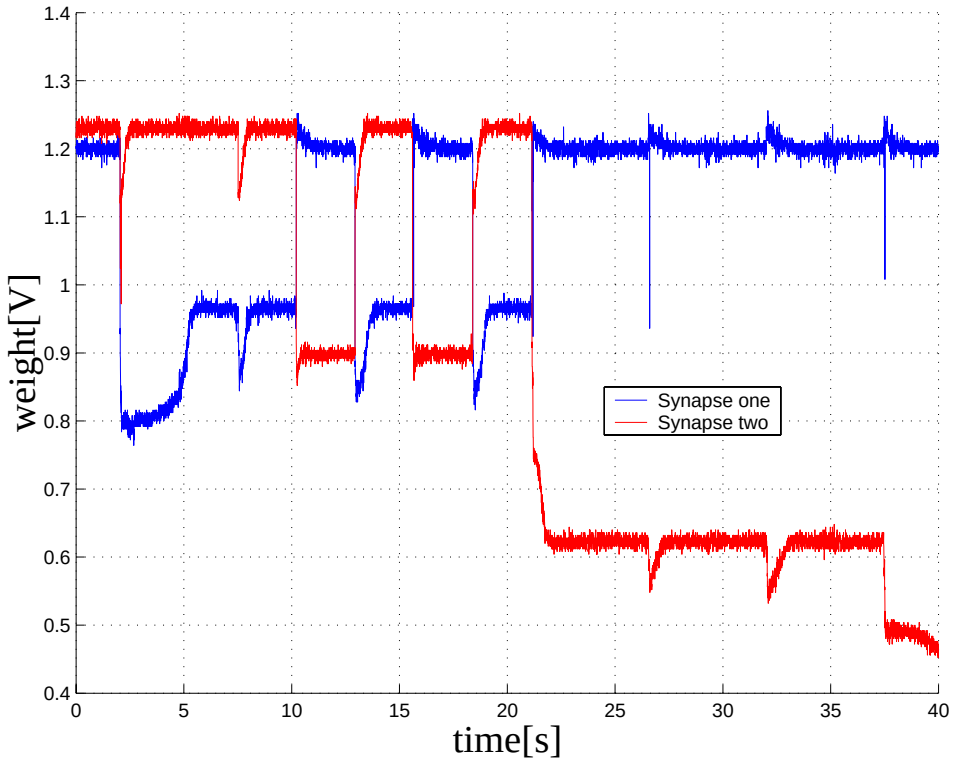

Figure 3.26: *Trace of weights for learning synapse one and two. Spike patterns as in Figure 3.24. At first, seemingly spike patter synapse one is depressed. But at the end of a "fierce battle", spike pattern synapse two is depressed. This scenario was more an exception than the rule. We performed 20 test runs with the same initial conditions and spike patterns, and only two times the synapse that was first stimulated, lost.*

Next, we wanted to test the inhibition signal. As described before, cross inhibition is used to prevent different neurons learning the same input pattern. In this way the network of neurons behaves as a winner-take-all network, where each neuron specializes on one type of input pattern or even one specific input pattern only. The inhibition signal is used to decorrelate other neurons such that there will be little or no overlap between input patterns learned. This means that for a specific spike pattern, only one neuron will trigger an AP.

The degree of cross inhibition is set by the bias voltage *w-*. In the first measurements we wanted to find this degree of cross inhibition for different values of *w-*. Therefore we set up two circuit boards, each with a duplicate of the chip, and gave them the same input pattern. The AP of the neuron was connected to the inhibition input signal of the adjacent chip's neuron. Again, for simplicity, the input spike pattern consisted of only one spike. Traces of the weights can be seen in Figure 3.27, 3.28, 3.29 and 3.30.

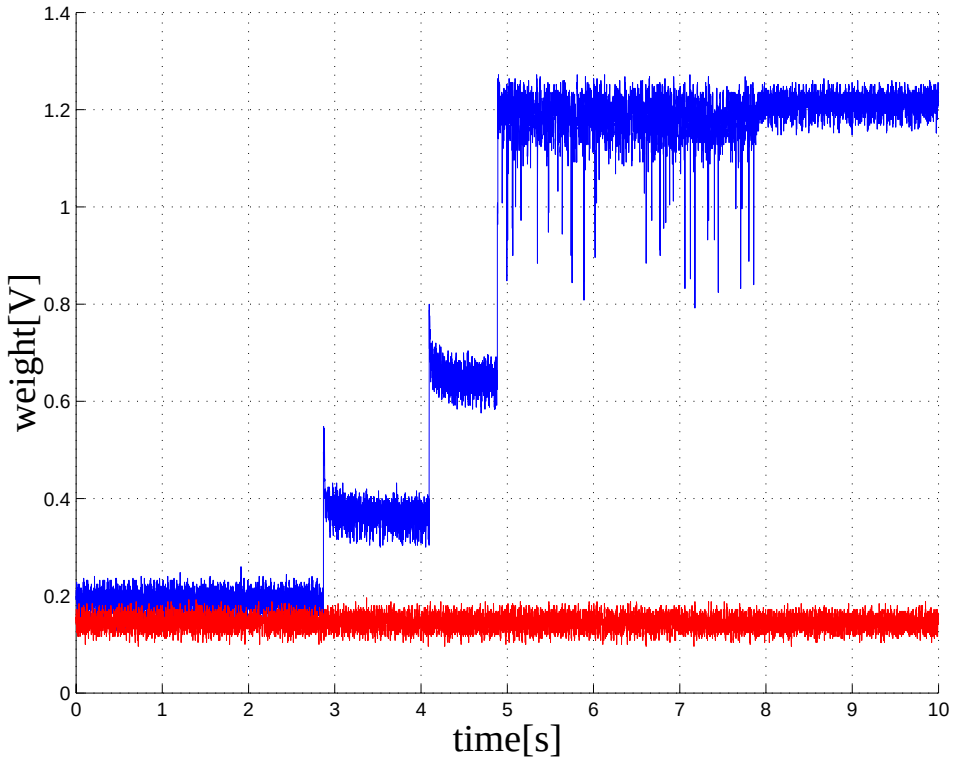

Figure 3.27: *Trace of weights of learning synapse one for two different neurons with active inhibition for* $w_- = 850mV$. *Poisson distributed input frequency of 10Hz to learning synapse one in both neurons. The degree of inhibition is fairly strong at this value. The first neuron to spike quickly learns the input pattern while the other neuron is unable to learn the input pattern or even react at all. As the synaptic weight increases for one neuron, it becomes increasingly difficult for the other neuron to learn the same pattern.*

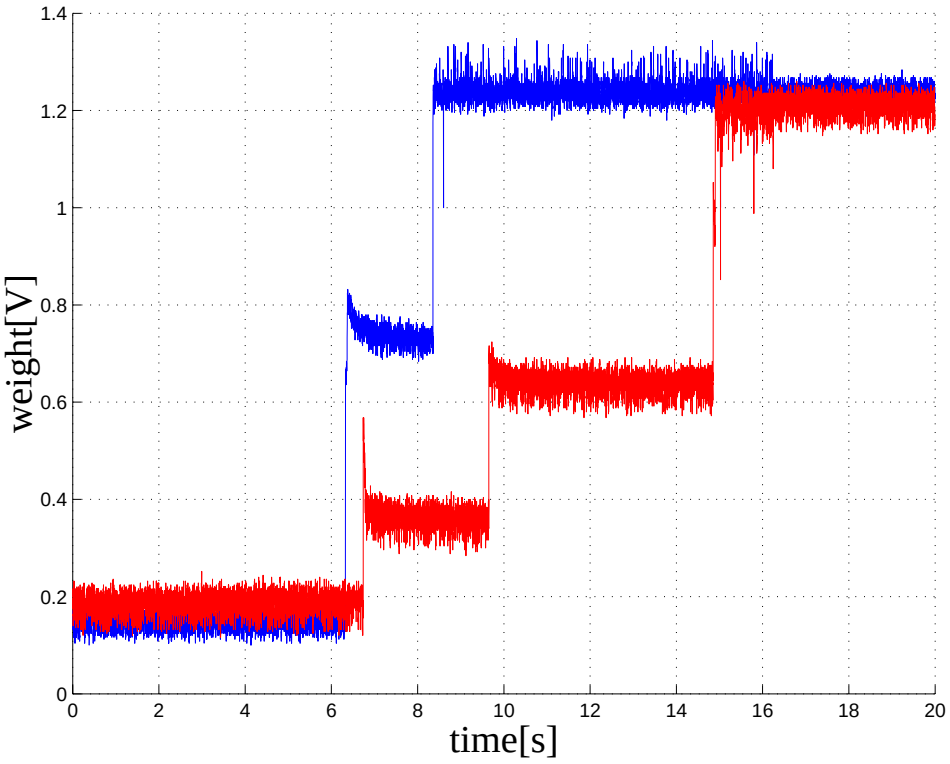

Figure 3.28: *Trace of weights of learning synapse two for two different neurons with active inhibition for* w- *= 450mV. Poisson distributed input frequency of 10Hz to learning synapse two in both neurons. The degree of inhibition is low and both neurons learns the input pattern. We observe that one neuron learns with a small delay, which increases as the weights increase. So there exists some degree of inhibition, but not enough to decorrelate the two neurons.*

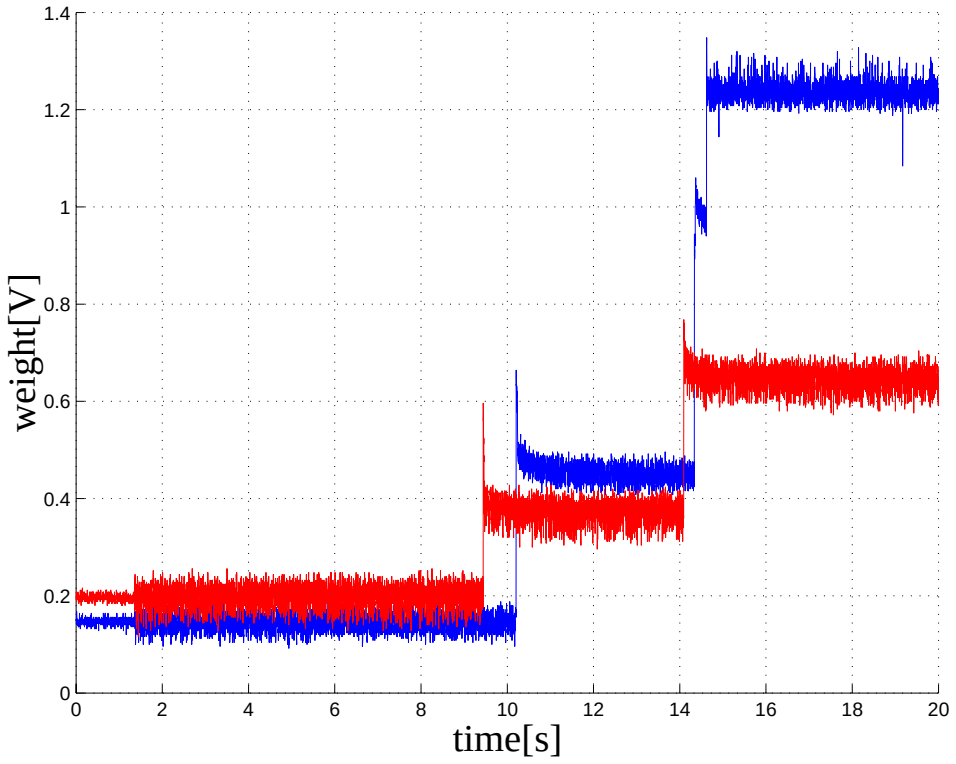

Figure 3.29: *Trace of weights of learning synapse three for two different neurons with active inhibition for* $w_- = 550mV$. *Poisson distributed input frequency of 10Hz to learning synapse three in both neurons. As expected, the degree of inhibition is at an intermediate level. One neuron learns the input pattern while the other partially learns the pattern. The neuron that partially learns the input pattern, will remain unchanged until a new and different pattern is presented. This is because it is prevented from spiking, thus unable to activate the learning.*

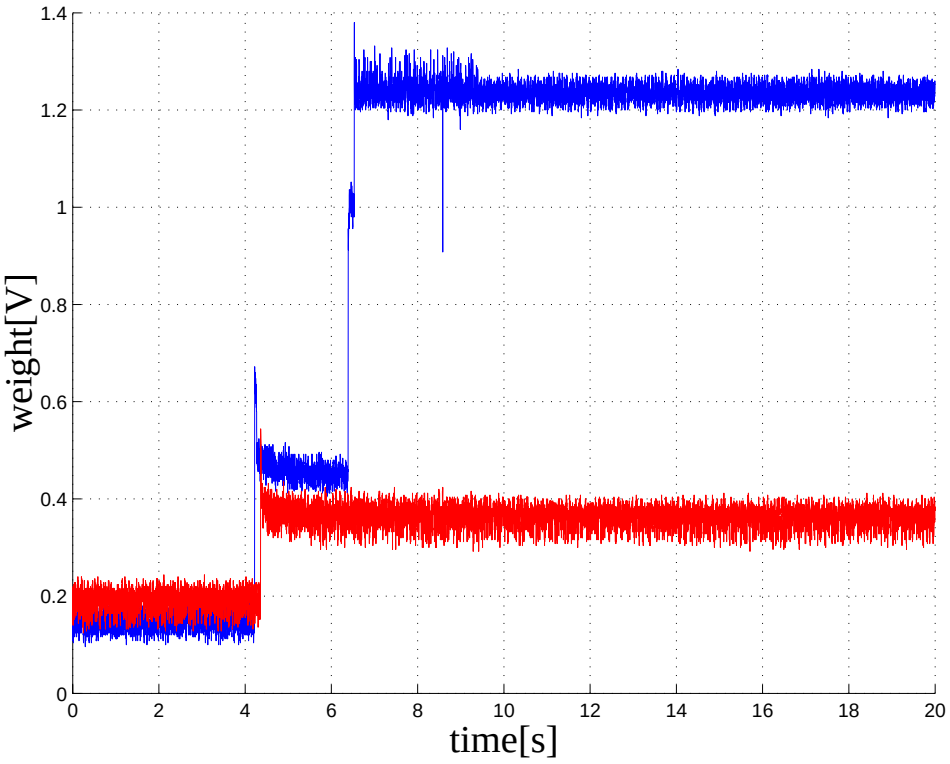

Figure 3.30: *Trace of weights of learning synapse four for two different neurons with active inhibition for* $w_- = 700mV$. *Poisson distributed input frequency of 10Hz to learning synapse four in both neurons. Again we obtain partial inhibition.*

Finally, we wanted to test the cross inhibition with two different input patterns. We used the same test setup as in the previous measurements. All weights in both neurons where set to a minimum. First, we stimulated learning synapse one in both neurons for a second with Poisson distributed input frequency of 100Hz. Then learning synapse two in the same neurons for the same period of time with the same input frequency. We repeated the stimulation over several runs, so that each learning synapse was stimulated for 10s. In theory, this would make the different neurons learn different patterns. In Figure 3.31, we see a plot of the output frequency of the two different neurons when synapse one and two are stimulated. Inhibition parameter $w\text{-} = 550\text{mV}$.

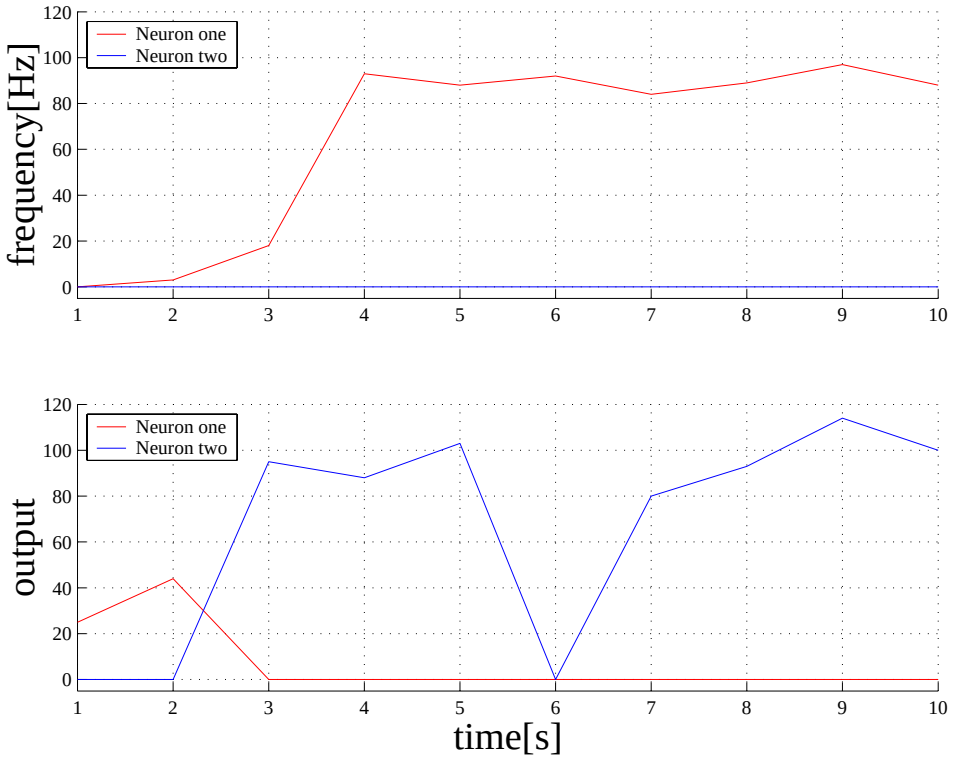

Figure 3.31: *Plot of output frequency for two neurons with active inhibition. Learning synapse one and two stimulated. In the upper graph, we show the activity while learning synapse one is stimulated. Neuron one slowly increases its output frequency while neuron two remains inactive over the whole period. In the lower graph, where we show the activity during stimulation of learning synapse two, the opposite is the case though neuron one shows some activity at start. It is clear that neuron one learns input pattern one while neuron two learns input pattern two. We observed the same behavior over several runs. A sudden drop in the output frequency of neuron two is observed in this particular recording. We do not know for sure why this happens, but the data file recording the output spikes contained several spikes from an unknown source. It may be other test objects on the same chip that spiked simultaneously and therefore interfered with our signals.*

We performed the same test where input pattern one stimulated learning synapses three and input pattern two stimulated learning synapse four for the same two neurons with same Poisson distributed input frequency. The graph can be seen in Figure 3.32. Inhibition parameter $w\text{-} = 550\text{mV}$.

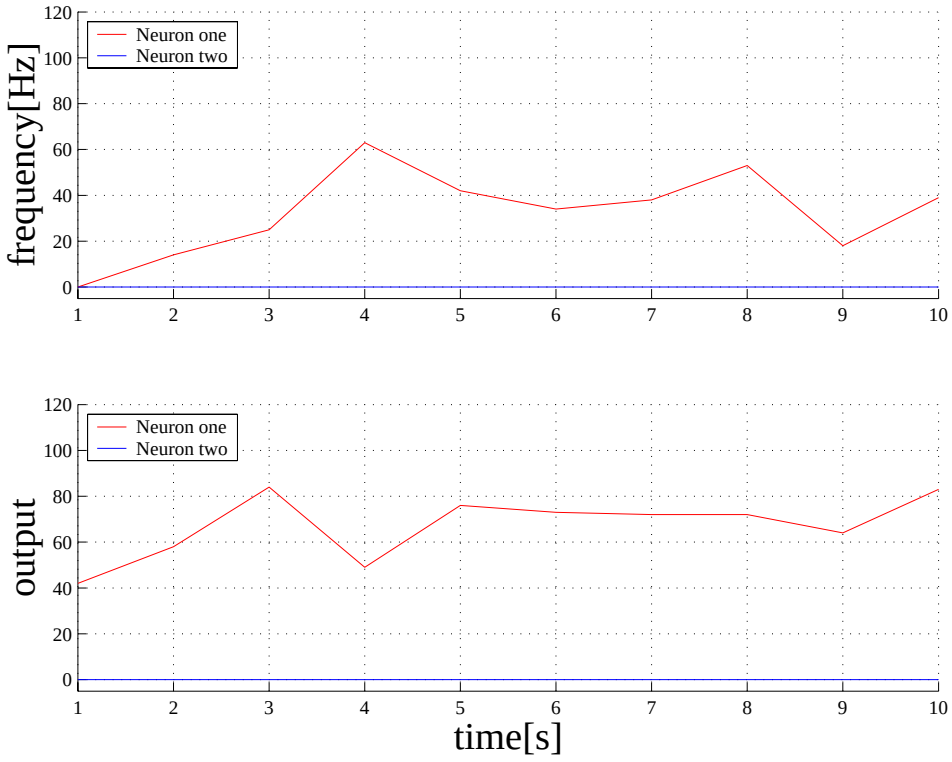

Figure 3.32: *Plot of output frequency for two neurons with active inhibition. Learning synapse three and four stimulated. The upper and lower graph shows the activity when learning synapse three and four is stimulated, respectively. As before, neuron one seemingly learns input pattern one in the upper graph, though the output frequency is not as high as before. Surprisingly, neuron one also learns input pattern two as seen in the lower graph. Neuron two is inactive during both stimulations. The output frequency of neuron one is again lowered and it seems as neuron one is not able to choose which pattern to learn. At the same time it prevents neuron two from learning either pattern. Again we observed the same behavior over several runs.*

From the two previous measurements, we conclude that the neuron shows promising results, but exhibits some problems with the inhibition. We tried to adjust the inhibition parameter *w-*, but it had no effect on the test results on the last experiment. We also increased the stimulation time from 1s to 2s for each synapse without results.

When testing the neurons in the neural array implemented on-chip, we observed clear and consistent variation in different learning synapses in the same neuron. One learning synapse was stimulated with a Poisson distributed input frequency of 100Hz for 10s. The output frequency was approximately 10Hz. We repeated the same experiment with a different learning synapse in the same neuron. There was zero activity. We had to increase the input frequency to 500Hz to observe the same output frequency of 10Hz. This shows that there was severe mismatch in the synapse strength for different synapses. This is probably the explanation for the lack of activity in neuron two in the latest experiment.

Chapter 4

Final words

We have designed, implemented and tested a MLSM for on-chip learning in a neuromorphic application. In the following sections we access a few of the key elements.

4.1 Memory size

The size of the implemented MLSM may be a problem in future implementations. We used 128 memory elements which covered an area of 1700×1200 μm, where each memory capacitance is about 250×30 μm in size. This nearly covered the entire available chip area of 2.2×2.2 mm. If an increased amount of memory elements are necessary, the size of the memory capacitor has to be reduced. However, this will affect the behavior of the MLSM and decrease its temporal analog behavior. It will also be more vulnerable to noise, since the memory capacitance capacitively shunts noise.

4.2 Functionality

The MLSM as a stand-alone application is flexible and robust. The voltage levels, attractor currents and number of discrete levels can all be chosen separately to optimize for a certain application. We operated in the sub threshold area of the nMOS transistor with minimal spacing of attractive voltage levels and minimal attractor current. We could easily have altered the implementation and parameters of the MLSM to make it function in another application.

We have used the MLSM as a synaptic storage in a learning neuron. We have shown that when the neuron is conveyed with two input patterns, the neuron learns one patterns while depressing the other input pattern.

Also, the neuron is able to interact with other neurons. We observed that two different neurons were able to use cross inhibition to learn separate and different input patterns . This is an important feature, which means that neurons in a neural network can decorrelate each other and therefore be able to store all possible input patterns. For our purpose, the MSLM showed promising results as a synaptic storage and will be a good alternative for use in the vision system designed during the CAVIAR project.

However, we experienced some problems throughout the process. The poor differentiated down pulse was of one concern. This had no serious effect on the learning behavior, but a more exponentially shaped curve would increase functionality of the MLSM. Furthermore, the system must be designed for one type of input. In other words, knowing the frequency of input spikes is essential for setting the proper bias voltages. If, for example, there is a relatively low input frequency, say of 1Hz, and the *down_bias*, which controls the current source in the MLSM, is set to work under 100Hz, the weights will never reach the lowest stable weight levels since they immediately will be attracted to its present stable weight level after a few ms. We have previously shown that the time to attract a weight to a stable weight level could be as long as 4s. However this only occurs when the change in weight voltage is at a maximum without directly jumping to the next stable weight level. When high frequencies are involved, the increase and decrease in the weight is much less, hence the time to attract is decreased.

We also experienced some reduced functionality in the MLSM itself since the stable weight levels in the memory were not as expected. Throughout the testing we were only able to use five stable weight levels, and the spacing between levels were increased. We do not know for sure if it was the voltage supply itself or the “fusing” transamp that shifted the levels, since we only could monitor the voltage on the weight capacitor. We have tried to given an explanation to the misbehavior, where we state that it is likely that the problem is caused by the overlap of attractor and offset currents in the “fusing” transamp.

4.3 Noise

Neuromorphic systems are, as previously defined, artificial systems based on computational principles used by biological nervous systems. And neurons are by nature noisy. Carver Mead states that [7],

Nerve pulses are by their very nature a horrible medium into

> *which to translate sound. They are noisy and erratic, and can work only over a limited range of firing rates*

Moreover, according to Adams [34], neurons have three flaws that make them noisy. APs are fixed in width an amplitude, independent of the excitation. The timing of APs are poor and suffers from large amount of time-based jitter. Last, the firing rate of a neuron is at max 500Hz, thus severely limits the frequency band of the nerve cell. Despite this, the nervous system is, as an example, capable of computing complex visual and auditory signals with high accuracy. So it is clear that the nervous system has found a way to compensate for its noisy parts which it is constructed of. I will not address why the nervous system is so accurate despite its problems, since it extends beyond this thesis, and more important, because only theoretical assumptions exist.

Therefore, our implementation will exhibit noise since it is an neuromorphic systems. If we examine signals from the chip, e.g. from Figure 3.25, the statement is validated. When we implemented the circuit we did not consider utilizing noise reduction techniques. Digital and analog signals lie close and without shielding. Nevertheless, the noise does not affect the performance of the circuit, since there are no critical signal paths or computations performed that require a "correct" answer. This is coherent with the neural system, which is a fault tolerant and robust system. If the down pulse is different for the same synapse where the weight is held constant, it has no crucial effect on the overall behavior of the neuron. This applies for every part of the neuron.

We have described that it is possible that the "fusing" transamp cause a shift in stable weight levels. This affects the behavior of the MLSM, but is an example of the flexibility of the circuit. We were able to adapt to the new situation and operate with a fully functional circuit.

4.4 Future work

Cilingiroglu *et. al.* [19] proposes an alternative MLSM. For every stable weight level N, it requires $(N-1) \times 4$ transistors excluding the voltage supply. It is clear that the size of the memory will be drastically reduced. Moreover, the functionality of the memory is somewhat similar to ours, where it exhibits an analog behavior over short periods of time and holds stable voltage levels for longer periods. Furthermore, the memory voltage can either be set directly through a pass transistor or be adjusted by injecting or drawing a current. The voltage levels must also be produced external to the memory, as in our case. But there is one

drawback to the memory: The resolution is fairly poor. The maximum stable levels for a voltage supply of 5V is six. So to operate in the sub threshold area is not an option. But we will here propose an alternative *learn down circuit* which can accommodate for this. If we replace the present *learn down circuit* and the present MLSM with the alternative MLSM and *learn down circuit*, we no longer have to operate in the sub threshold region. If we look at our implementation, we use the weight to set the down pulse length and control the current to the soma. In Figure 4.4, the alternative *learn down circuit* can be seen where the length of the *learn_down* pulse is linearly dependent on the weight. The pulse could also be used to control the current to the soma instead of the weight as done now. However, this is not the case when the lowest weight is 0V, since it then would not draw any current from the soma and hence never be able to learn and increase its weight. As stated before, the dependence on the weight should be exponential. But as simulations and test results has shown, this is not exactly the case. The length of the down pulse is merely exponential for the two highest weights, and the same applies for the current to the soma. Therefore, it might be preferable to use the linear dependence over the exponential since it works over the entire voltage span.

The new proposed implementation changes the dependence on the weight. Previously, the current to the soma was proportional to e^{V_w}. Now, the current to the soma is proportional to the weight itself, V_w. Therefore the learning rule used in this thesis is not longer valid (equation 2.1). In fact, the original local learning rule presented in [29] is valid for the new implementation:

$$\frac{d}{dt}\vec{w} = \alpha\vec{\tilde{C}} - \beta\vec{\tilde{w}}\tilde{O} \tag{4.1}$$

The original learning rule lacks the extra multiplier w_i, which made the learning “accelerate” and more turbulent with higher weights.

One may wonder why we have not implemented these new elements instead. The main reason is that we late in the process discovered the alternative MLSM. As a result, we have not tested the alternative MLSM itself and as a synaptic storage in such a degree as the MLSM we present in this thesis. It may be that the alternative MLSM is not so suited as it seems.

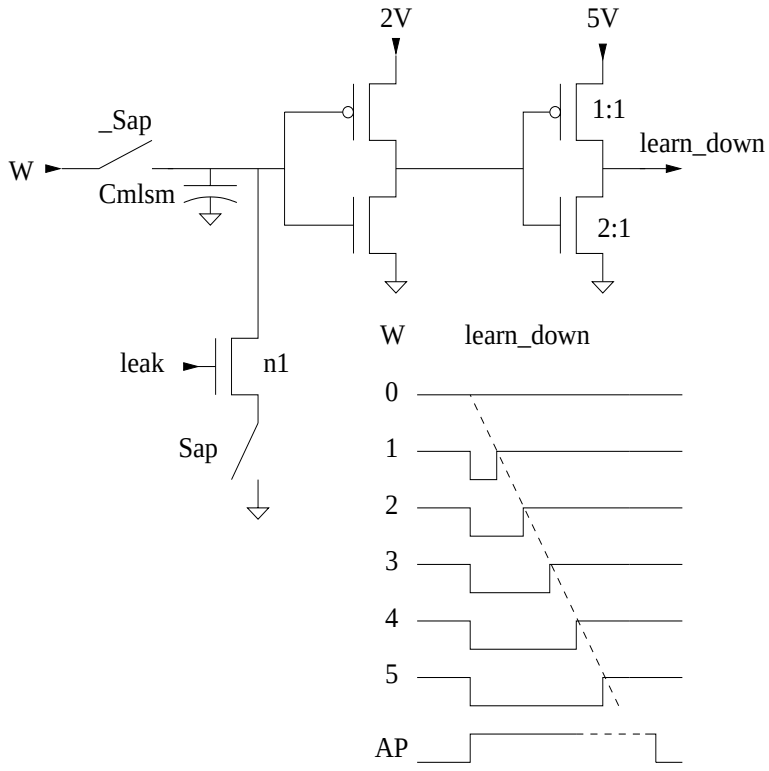

Figure 4.1: *The alternative learn down pulse shaper. V_{dd} = 5V. The first inverter switches at approximately 0.5V. The switch $\_S_{\text{ap}}$ is open (not conduction) and S_{ap} closed (conducting) when AP is high. Therefore, when an AP goes high, the charge on C_{mlsm} will start to flow through the leak branch controlled by* ***leak****. Since $V_{\text{leak}} > V_{th_{n1}}$ and $V_{ds_{n1}} \geq V_{\text{leak}} - V_{th_{n1}}$ for all weights ($V_{\text{leak}} \approx 1V$), the transistor is saturated. Therefore the current drawn is $I_{leak} = I_S(V_{leak} - V_{th_{n1}})^2(1 + \lambda_n V_{mlsm})$ where $I_S = \frac{W}{L}\mu_n C_{OX}\frac{1}{2}$. The term $(1 + \lambda_n V_{mlsm})$ modulates the early effect and is usually negligible. All other parameters are constant. The length of the down pulse is then $T_{down} = \frac{V_{mlsm}}{I_{leak}} C_{\text{mlsm}}$. Therefore we have a linear dependence on the weight since the current will be constant for all values and the spacing between weights are the same. The pulse from the inverter will have a peak of 2V, so the second inverter assures that the value of the pulse is from rail to rail (0V-5V). The nMOS transistor in the second inverter should at least have two times the W/L-ratio compared to the pMOS transistor. This makes the pull down strength stronger, hence the inverter is able to pull down at 2V. The length of the down pulse must be shorter than the length of the AP to prevent the AP from setting an upper limit of the* ***learn_down*** *pulse length.*

Bibliography

[1] M. Mahowald and R. Douglas. **A Silicon Neuron**. *Nature*, vol. 354:pp. 515–518, 1991.

[2] C. Mead. ***Analog VLSI and Neural Systems***. Addison Wesley, 1989. Chapter 4.

[3] P. Häfliger. **Perceptrons (McCulloch Pitts neurons)**. http://www.ifi.uio.no/infneuro/Gamle/H2002/hafliger/neurons.html#NeurPerceptron.

[4] C. Mead. **Neuromorphic Electronic Systems**. *Proceedings of the IEEE*, vol. 78(no. 10):pp. 1629–1636, October 1990.

[5] M. Schwarz, B.J. Hosticka, R. Hauschild, W. Mokwa, M. Scholles, and H.K. Trieu. **Hardware Architecture of a Neural Net Based Retina Implant for Patients Suffering from Retinitis Pigmentosa**. *IEEE Conference on Neural Networks*, vol. 2:pp. 653–658, June 1996.

[6] M.A. Mahowald and C. Mead. **The Silicon Retina**. *Scientific American*, pages 76–82, May 1991.

[7] R.F. Lyon and C. Mead. **An Analog Electronic Cochlea**. *IEEE Trans. on Acoustic Speech and Signal Processing*, vol. 36(no. 7), July 1998.

[8] R. Sarpeshkar, R.F. Lyon, and C.A. Mead. **Low-Power Wide-Dynamic-Range Analog VLSI Cochlea**. *Analog Integrated Circuits and Signal Processing*, vol. 16(no. 3):pp. 245–274, August 1998.

[9] S. Thorpe, D. Fize, and C. Marlot. **Speed of processing in the human visual system**. *Nature*, vol. 381:pp. 520–522, 1996.

[10] Henry Markram, Joachim Lübke, Michael Frotscher, and Bert Sakmann. **Regulation of Synaptic Efficacy by Coincidence of Postsynaptic APs and EPSPs**. *Science*, vol. 275:pp. 213–215, 1997.

[11] B. Hochet. **Multivalued MOS Memory for Variable-Synapse Neural Networks**. *Electronic Letters*, vol. 25:pp. 669–670, 1989.

[12] D.B. Schwartz, R.E. Howard, and W.E. Hubbard. **A Programmable Analog Neural Network Chip**. *IEEE J. Solid-State Circuits*, vol. 24(no. 2), 1989.

[13] R. Castello, D.D. Caviglia, M. Franciotta, and F. Montecchi. **Selfrefreshing Analog Memory Cell for Variable Synaptic Weights**. *Electronic Letters*, vol. 27(no. 20):pp. 1871–1873, September 1991.

[14] Y. Arima, M. Murasaki, T. Yamada, A. Maeda, and H. Shinohara. **A Refreshable analog VLSI Neural Network Chip with 400 Neurons and 40k Synapses**. *IEEE J. Solid-State Circuits*, vol. 27:pp. 1854–1861, 1992.

[15] B. Linares-Barranco, E. Sánchez-Sinencio, A. Rodríguez-Vázquez, and J.L. Huertas. **A CMOS Analog Adaptive BAM with On-Chip Learning and Weight Refreshing**. *IEEE Trans. Neural Networks*, vol. 4(no. 3), May 1993.

[16] T. Watanabe, K. Kimura, M. Aoki, T. Sakata, and K. Ito. **A Single 1.5-V Digital Chip for a 10^6 Synapse Neural Network**. *IEEE Trans. Neural Networks*, vol. 4(no. 3), 1993.

[17] T. Morie and Y. Amemiya. **An All-Analog Expandable Neural Network LSI with On-Chip Backpropagation Learning**. *IEEE J. Solid-State Circuits*, vol. 29(no. 9), September 1994.

[18] O. Ishizuka, Z. Tang, and H. Matsumoto. **On Design of Multiple-Valued Static Random-Access-Memory**. *Proceedings of the 20. Int. Symposium of Multiple-Valued Logic*, pages 11–17, May 1990.

[19] Ugur Cilingiroglu and Yaman Özelci. **Multiple-Valued Static CMOS Memory Cell**. *IEEE Trans. on Circuits and Systems*, vol. 48(no. 3), March 2001.

[20] T.S. Lande. ***Floating-gate Circuits and Systems***, chapter 1, pages 1–21. Kluwer Academic, 2001. Trade-Offs in Analog Circuit Design.

[21] Chris Diori, Paul Hasler, Bradley A. Minch, and Carver Mead. **A Floating-Gate MOS Learning Array with Locally Computed Weight Updates**. *IEEE Transactions on Electron Devices*, vol. 44(no. 12), Desember 1997.

[22] R.R. Harrison, J.A. Bragg, P. Hasler, B.A. Minch, and S.P. Deweerth. **A CMOS Programmable Analog Memory-Cell Array using Floating-Gate Circuits**. *IEEE trans. on circuits and systems-II: Analog and digital signal processing*, vol. 48(no. 1), January 2001.

[23] L.W. Buchan, A.F. Murray, and H.M Reekie. **Floating Gate Memories for Pulse-Stream Neural Networks**. *IEEE electronic letters*, vol. 33(no. 5):pp. 397–399, February 1997.

[24] P. Häfliger and C. Rasche. **Floating Gate Analog Memory for Parameter and Variable Vtorage in a Learning Silicon Neuron**. *ISCAS*, 1999.

[25] C. Sin, A. Kramer, V. Hu, R.R. Chu, and P.K. Ko. **EEPROM as an analog storage device, with particular application in neural networks**. *IEEE trans. on electron devices*, vol. 39(no. 6), June 1992.

[26] D.A. Durfee and F.S. Shoucair. **Comparison of Floating Gate Neural Network Memory Cells in Standard VLSI CMOS Technology**. *IEEE Trans. on Neural Networks*, vol. 3(no. 3), May 1992.

[27] T. Shibata, H. Kosaka, H. Ishii, and T. Ohmi. **A Neuron-MOS Neural Network Using Self-Learning-Compatible Synapse Circuits**. *IEEE Journal of Solid-State Circuits*, vol. 30(no. 8), August 1995.

[28] A.J. Montalvo, R.S. Gyurcsik, and J.J. Paulos. **Toward a General-Purpose Analog VLSI Neural Network with On-Chip Learning**. *IEEE Trans. on Neural Networks*, vol. 8(no. 2), March 1997.

[29] Ph. Häfliger. ***A Spike Based Learning Rule and its Implementation in Analogue Hardware***. PhD thesis, ETH Zürich, Switzerland, 2000.

[30] D.O. Hebb. ***The Organization of Behavior***. Wiley, New Tork, 1949. page 62.

[31] **Convolution AER Vision Architecture for Real-Time**.

[32] Stefano Fusi. **Long term memory: Encoding and storing strategies of the brain**. *Neurocomputing*, vol. 38-40:pp. 1223–1228, 2001.

[33] Ph. Häfliger and H. Kolle Riis. **A Multi-Level Static Memory Cell**. *IEEE International Symposium on Circuits and Systems (ISCAS)*, vol. 1:pp. 25–28, May 2003.

[34] R. W. Adams. **Spectral Noise-Shaping in Integrate-and-Fire Neural Networks**. *International Conference on Neural Networks*, vol. 2:pp. 953–958, June 1997.

[35] M. Mahowald. ***The Silicon Optic Nerve from An Analog VLSI System for Sterescopic Vision***, chapter 3. Kluwer Academic Publishers, 1994.

[36] Stephen R. Deiss, Rodney J. Douglas, and Adrian M. Whatley. ***A Pulse-Coded Communication Infrastructure for Neuromorphic Systems***, pages 157–178. The MIT Press, 1999.

[37] K.A. Boahen. **Point-to-Point Connectivity Between Neuromorphic Chips Using Address Events**. *IEEE Trans. Circuits and Systems*, vol. 47(no. 5), 2000.

List of Figures

List of Tables

Appendix A

Address Event Representation

The neurons in the human brain are a part of a complex system, where each neuron connects to approximately 10^4 other neurons [29]. In present VLSI technology this is almost impossible to achieve, and even more difficult if dealing with multi-chip configurations. So the need to handle internal and external communication in another way is necessary to make neuromorphic electronics reflect the actual behavior of the brain. One of these approaches is *Address Event Representation* (AER) developed by M. Mahowald *et. al.* [35], which uses an asynchronous digital multiplexing technique. In same manner as real neuronal AP, there is only one type of event, a digital pulse where the interval between events are analog. When a neuron spikes, it sends a request to the sender encoder which broadcast its unique digital address on the common communication bus. On the receiver side, i.e. the input to a new chip with a neural network, this address is decoded an distributed to the neurons which it is logically connected to. Thus the protocol emulates directly connected neurons with a single bus, a encoder and a decoder. This is possible since the width of the events are small (down to 100ns) compared to the with of the AP itself (approximately 1ms). Therefore the chances of collisions are small. If they however do occur, that is, if there are several neurons spiking close in time, events can be multiplexed on the bus sequentially such that they loose very little of their temporal information. The AER has been used successfully in several neuromorphic systems [36, 37]. An illustration of the basic concepts of AER is shown in Figure A.1.

A.1 AER circuit components

The sender handles the output from the chip to an external bus. Figure A.2 shows a schematic of the sender.

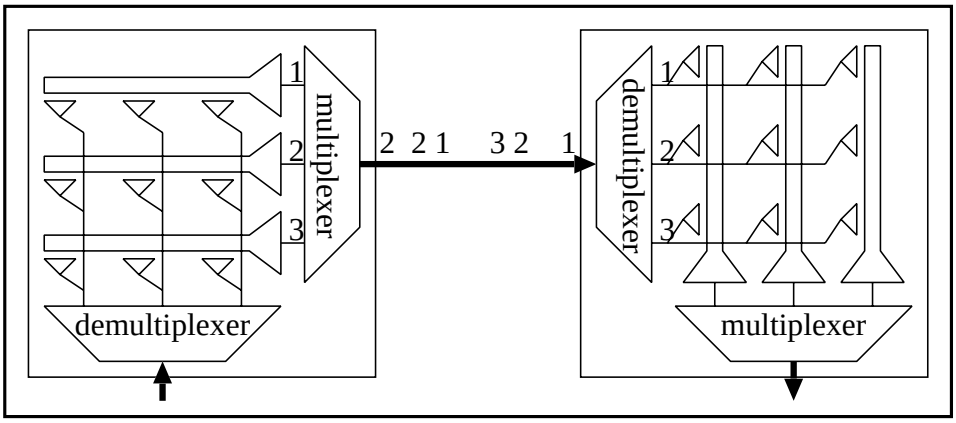

Figure A.1: *Address event representation.*

The receiver handles the input to the chip from an external common bus. A schematic of the AER receiver can be seen in Figure A.3.

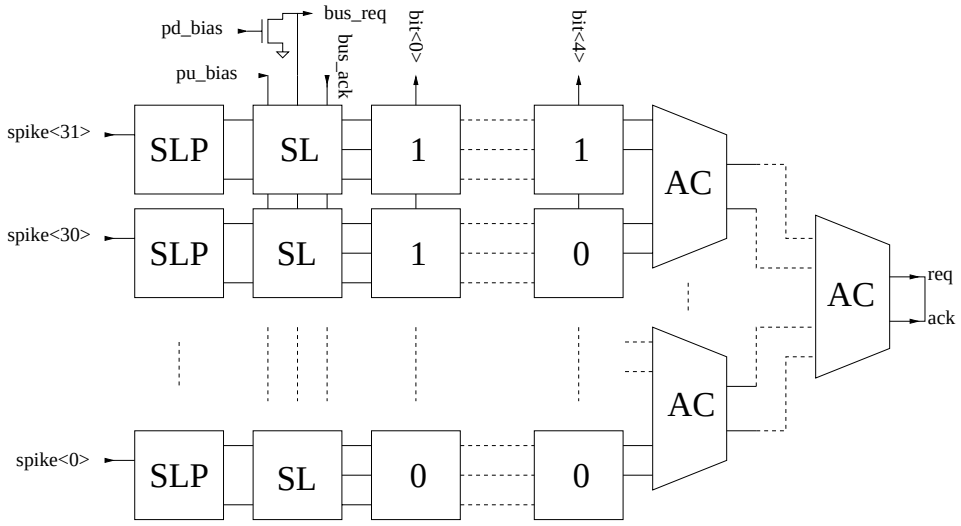

Figure A.2: *AER sender. Takes as input **spike<0-31>** from the neural network and sends a four bit address out on the common AER bus after a request-acknowledge is performed. The **bus_req** and **bus_ack** are signals for the off-chip handshake. The blocks on the left, the arbiter cells (AC), controls the access to the AER bus if there are several spikes present. The two rightmost outputs **req** and **ack** are directly connected. Therefore an acknowledge is returned immediately if no other spikes are awaiting an acknowledge. If there are several spikes present the one that sends a request first is granted access, while the other spikes awaits their turn.*

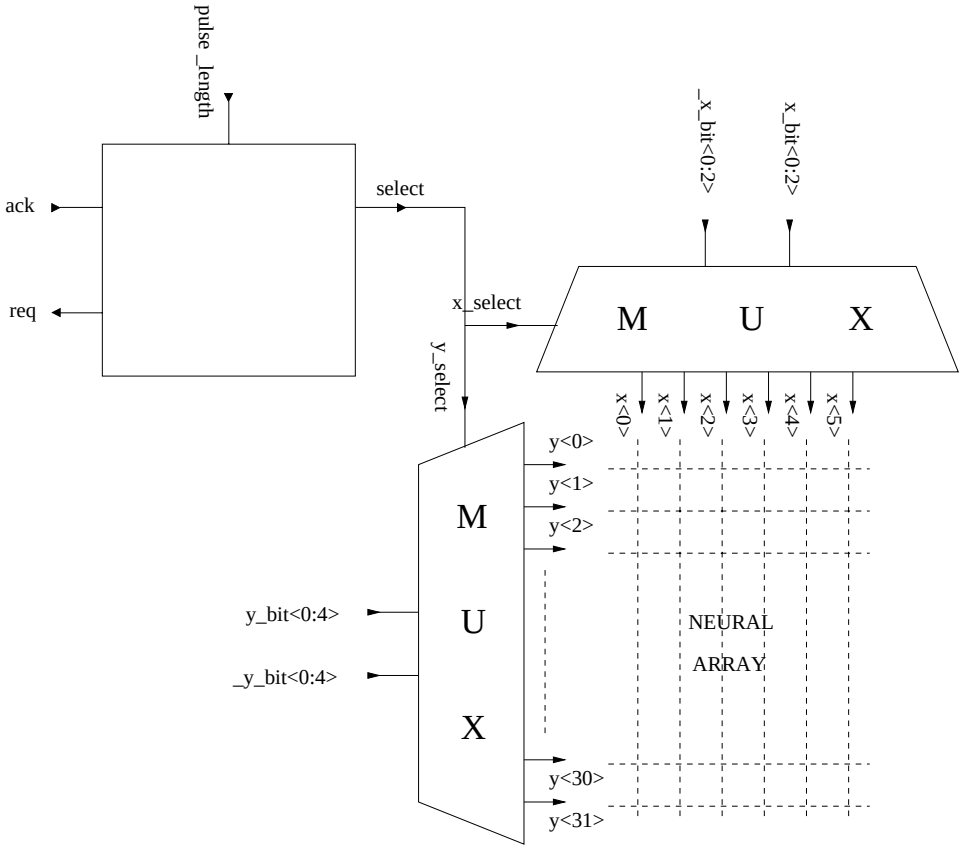

Figure A.3: *AER receiver. The block to the left is the request-acknowledged circuit, which translates a four phase handshake into a data valid pulse, or the select signal seen to the right. The input* ***pulse_length*** *determines the length of the select signal, hence the length of the synaptic inputs* ***x*** *and* ***y****. The two ν-shaped blocks are demultiplexers where the five and three bit address is decoded to a 32×6 matrix which holds the coordinates of the synapse or synapses to be stimulated.*

Appendix B

Layout

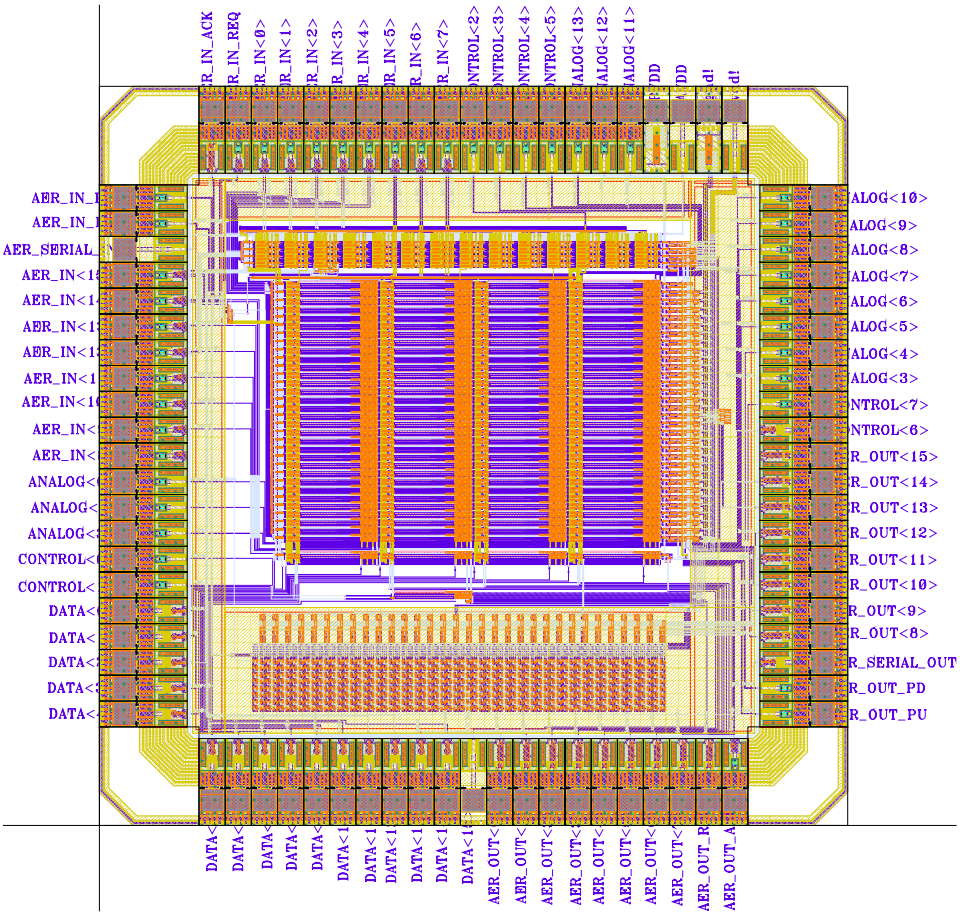

Figure B.1: *Chip layout.*

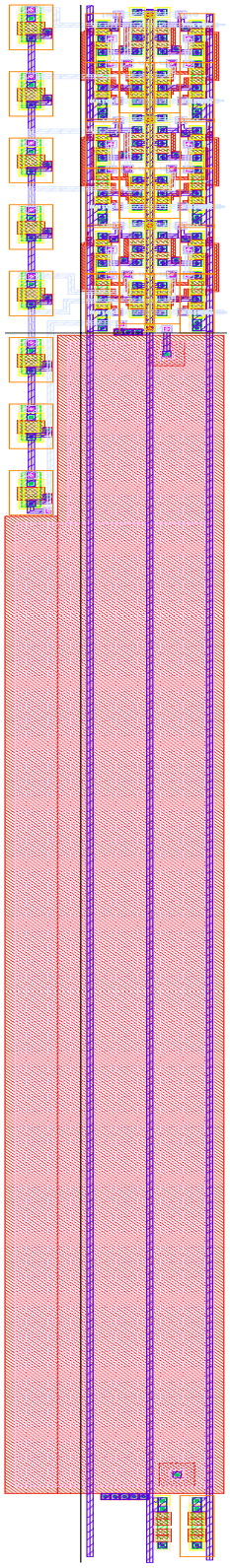

Figure B.2: *MLSM layout.*

Appendix C

Test setup

The chip's schematic and layout where designed with *Cadence* version 4.4.5.100.10 and simulated with *SpectreS* under *Analog Environment* in *Cadence*.

The chip was manufactured by Austria Micro Systems(AMS) in a 0.6μm CMOS process.

Instrument	**Description**
Hewlett Packard E3614A	DC power supply
Keithley 213	Quad voltage source
Keithley 6512	Programmable Electrometer
Tektronix TDS 3052	Digital Phosphor Oscilloscope
Agilent 33250A	Waveform generator

Table C.1: *Instruments used during measurements.*

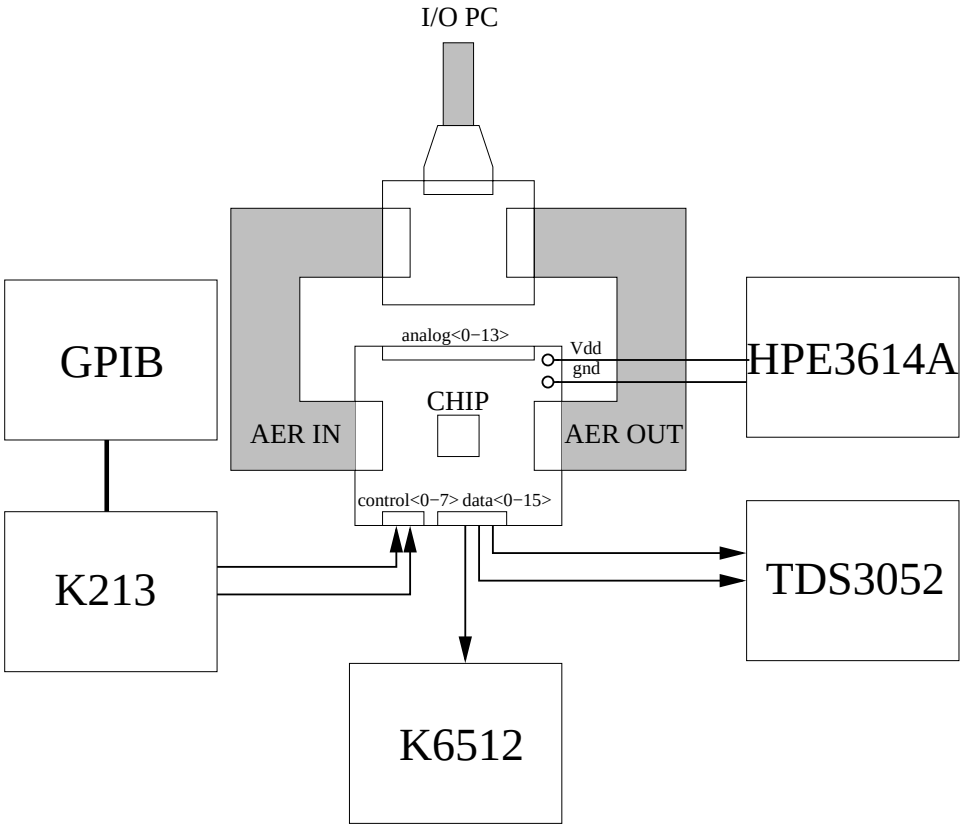

Figure C.1: *Test setup. The AER interface connects to the chip through a print-board where I/O from the PC is distributed. The* ***Keithley 213*** *Quad voltage source is connected to the GPIB bus and was used to set the* ***minus*** *and* ***plus*** *inputs when testing the "fusing" transamp while the* ***Keithley 6512*** *Programmable Electrometer measured the output current. The* ***Tektronix TDS 3052*** *Digital Phosphor Oscilloscope traced the output voltages of the single neuron.* ***Hewlett Packard E3614A*** *supplied the print-board and chip with DC power supply.*

Appendix D

Pin list

Pin name	Pin	Type
shunt_duration	ANALOG<0>	analog input(n bias)
w-	ANALOG<1>	analog input(n bias)
w+	ANALOG<2>	analog input(p bias)
follower_padbias	ANALOG<3>	analog input(n bias)
amp_in-	ANALOG<4>	analog input(p bias)
_corr_threshold	ANALOG<5>	analog input(p bias)
w_bias	ANALOG<6>	analog input(n bias)
corr_pulselength	ANALOG<7>	analog input(n bias)
corr_leak	ANALOG<8>	analog input(n bias)
_corr_inc	ANALOG<9>	analog input(p bias)
w_max	ANALOG<10>	analog input(n bias)
_ap_length	ANALOG<11>	analog input(p bias)
_leak	ANALOG<12>	analog input(p bias)
sam_level_bias	ANALOG<13>	analog input
mem_down	CONTROL<0>	analog input(n bias)
_mem_up	CONTROL<1>	analog input(p bias
w_baseline	CONTROL<2>	analog input(gnd)
_sam_up_bias	CONTROL<3>	analog input(p bias)
sam_down_bias	CONTROL<4>	analog input(n bias)
sam_pbias	CONTROL<5>	analog input(p bias)
amp_in+	CONTROL<7>	analog input(p bias)
y_bit<0:4>	AER_IN<0:4>	digital input

Table D.1: *Pin list (1/2).*

Pin name	Pin	Type
_y_bit<0:4>	_AER_IN<0:4>	digital input
x_bit<0:2>	AER_IN<5:7>	digital input
_x_bit<0:2>	_AER_IN<5:7>	digital input
_rec_y	_AER_IN<9>	digital input
_rec_x+	_AER_IN<10>	digital input
_rec_x-	_AER_IN<11>	digital input
_rec_x<0:3>	_AER_IN<12:15>	digital input
shunt	DATA<0>	analog output
ap	DATA<1>	digital output
corr<0:3>	DATA<3,5,7,9>	analog output
w<0:3>	DATA<4,6,8,10>	analog output
soma	DATA<13>	analog output
mem_out	DATA<14>	analog output
amp_out	DATA<15>	analog output(barepad)
_up<0:3>	AER_OUT<0,2,4,6>	digital output
down<0:3>	AER_OUT<1,3,5,7>	digital output
bit<0:4>	AER_OUT<8:12>	digital output
bus_req	AER_OUT_REQ	digital output
_bus_ack	AER_OUT_ACK	digital input
pd_bias	AER_OUT_PD	analog input(n bias)
pu_bias	AER_OUT_PU	analog input(p bias)
pulse_length	AER_IN_PD	analog input(n bias)

Table D.2: *Pin list (2/2).*

Appendix E

Bias voltages

Bias name	*V*
w+	4.10
_sam_up_bias	4.24
w-	x.xx
shunt_duration	1.50
sam_down_bias	920m
sam_level_bias	1.10
sam_pbias	4.29
w_max	2.10
_corr_inc	3.70
_corr_leak	600m
_corr_threshold	4.30
_corr_pulselength	570m
w_baseline	0.00
_leak	5.00
_ap_length	4.25
follower_padbias	1.19
w_bias	4.10

Table E.1: *Bias voltage list.*

Appendix F

Publications

Ph. Häfliger and H. Kolle Riis. **A Multi-level Static Memory Cell.** *IEEE International Symposium on Circuits and Systems (ISCAS)*, Vol. 1:pp. 25-28, Bangkok May 2003

A MULTI-LEVEL STATIC MEMORY CELL

P. Häfliger and H. Kolle Riis

IEEE International Symposium on Circuits and Systems (ISCAS)
vol. 1 , pp. 25 -28, Bangkok May 2003

A MULTI-LEVEL STATIC MEMORY CELL

P. Häfliger and H. Kolle Riis

Institute of Informatics
University of Oslo
PO Box 1080, Blindern
N-0316 Oslo
Norway
e-mail: hafliger@ifi.uio.no

ABSTRACT

This paper introduces a static multi-level memory cell that was conceived to store state variables in neuromorphic on-chip learning applications. It consists of a capacitance that holds a voltage and an array of 'fusing' amplifiers that are connected as followers. These followers drive their output towards the voltage level of the input like normal followers, but only if the difference between input and output is smaller than about 120mV. The inputs to this 'fusing' follower array determine the stable voltage levels of the memory cell. All follower-outputs are connect to the storage capacitance and thus the voltage is always driven to the closest stable level. The cell content can be changed by injecting current into the capacitance. This form of storage offers arguably a better compromise between desirable and undesirable properties for neuromorphic learning systems than alternative solutions (e.g. non-volatile analog storage on floating gates or digital static storage in combination with AD/DA conversion), as shall be discussed in the following.

1. INTRODUCTION

A central problem in bringing learning to analog hardware is the storage of the learning state. The state of learning in neural network models is represented by the 'weights', i.e. the connection strengths between the neurons in the network. In most models these weights are analog values that change slowly as the system learns through experiences. Therefore these values require to be preserved over long periods of time but still they need to be easily changeable.

Engineers coming from a digital background would implement such a memory by combining digital storage with digital to analog conversion [7]. Such storage is reasonably reliable and also easy to change. One disadvantage is the need for a clock signal which is a drawback in continuous time systems inspired by the nervous system. And it is not real analog storage.

The easiest form of real analog storage are capacitors. Their voltage level can easily be changed by injecting currents. They are easy to use, also in continuous time circuits. They are however not holding a stored voltage for long periods of time. Leakage currents decrease the voltage in the order of millivolts per second. Some sort of refresh can be applied like in digital dynamic memory, but this requires quite complex extra circuitry and also some digital control signals which add extra noise to an analog circuit.

Another source of inspiration is digital non-volatile memory like magnetic memory or flash ROM (Also called EEPROM or floating gate storage). Values that are thus stored are not so easy and fast to change. But they require no form of refresh and they can just as well be used in an analog way. Especially promising is floating gate storage since it can be implemented on the same VLSI chip as transistors and capacitors. Thus memory can be located just besides processing elements, which makes it very attractive for neuromorphic systems ('...systems that are based on the organizing principles of the nervous system...' [9]): There is no spatial separation between storage and processing in the brain, i.e. no centralized memory and no central processor. Therefore floating gates begun to be used for analog storage some time back [8, 3] and more recently also in dynamically changing systems, among them learning systems [2, 5, 4]. They are very compact and offer non-volatile and real analog storage. Their major drawback is bad property matching, especially of the tunneling structures that are used to change the charge of the floating gate. The efficacy of those tunneling structures diminishes with their use. Thus the mismatches among those tunneling structures increase beyond what is normal for other structures like transistors or capacitances. Also when used in standard CMOS processes the writing to floating gates by tunneling requires high voltages beyond the supply voltage rails.

This paper offers another possibility, which we consider to be an optimal compromise for analog neural network implementations. It operates like static digital memory that has not just two attractive voltage levels but several. The voltage level of the cell can be changed just like on a capacitor, by simple current injections. In fact, on a short time scale it really is nothing but a capacitance that holds an analog voltage. On a long time scale its resolution is limited by a discrete number of attractors.

This work was supported by the EU 5th Framework Programme IST project CAVIAR.

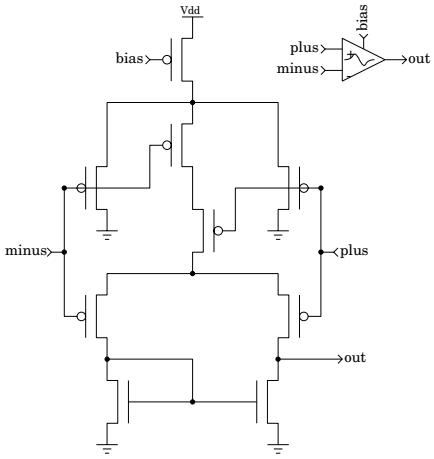

Figure 1: The 'fusing' transconductance amplifier. It only works like a transconductance amplifier for small differences (100mV-200mV) of the input voltages. The upper five transistors form a so called bump circuit. The middle branch is only conductive, if the input voltages are close to each other. It supplies the bias current to the below transconductance amplifier, such that it is turned off, if the inputs get too far apart. The characteristics of this circuit is depicted in figures 2'and 3, and expressed by formula (3)

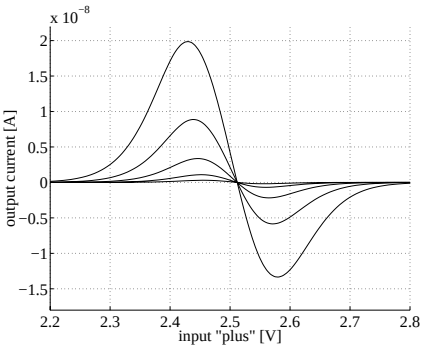

Figure 2: The characteristics of the 'fusing' amplifier for different bias voltages. The input 'minus' is 2.5V and the voltage of input 'plus' is drawn on the x-axis. The y-axis is the current into the output node. All transistors were square with W=L=1.4μm. The different traces are simulations with different bias voltages. It was set to be (from the flattest to the steepest curve) 4.20V, 4.15V, 4.10V, 4.05V, and 4.00V. The same as for transconductance amplifiers, the transconductance can be increased in the linear range by a bigger bias voltage, which also slightly increases the distance of the characteristic's maximum and minimum. Those distances were: 104.3mV, 108.3mV, 119.4mV, 131.1mV, 150.4mV.

2. METHODS

The central building block of the multi-level static memory cell is a novel 'fusing' transconductance amplifier (figure 1). The fusing transconductance amplifier is a combination of a normal transconductance amplifier and a so called bump circuit. This bump circuit was originally put forward in [1]. It issues an output current if two input voltages lie close to each other, i.e. within a range of about 120mV. The formula given in [1] is

$$I_{bump} = \frac{I_b}{1 + \frac{4}{S}\cosh^2\frac{\kappa\Delta V}{2}} \tag{1}$$

where κ is the slope factor, I_b is the bias current, and ΔV is in units of $\frac{kT}{q}$. S is defined as

$$S = \frac{(W/L)_{middle}}{(W/L)_{outer}} \tag{2}$$

where $(W/L)_{middle}$ is the width to length ratio of the transistors in the middle branch of the bump circuit, $(W/L)_{outer}$ that of the two outer branches. S is the main factor determining the width (defined as the width where I_{bump} is but a fraction $\frac{1}{e}$ of its maximum) of the 'bump' which scales approximately with $logS$ for $S >> 1$. The bias current on the other hand does not influence the width of the bump in the subthreshold regime.

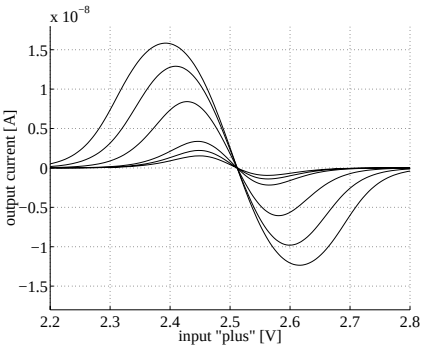

Figure 3: The characteristics of the 'fusing' amplifier for different parameters S. The input 'minus' is 2.5V and the voltage of input 'plus' is drawn on the x-axis. The y-axis is the current into the output node. The bias voltage was 4.1V. All transistor widths and lengths were W=1.4μm with the exception of the middle branch of the bump circuit. The different traces are simulations with different $(W/L)_{middle}$. From the flattest to the steepest curve it was: 0.25, 0.5, 1, 2, 4, 8. The distance of the maxima was: 112.5mV, 116.6mV, 119.4mV, 152.9mV, 191.0mV, 226.0mV.

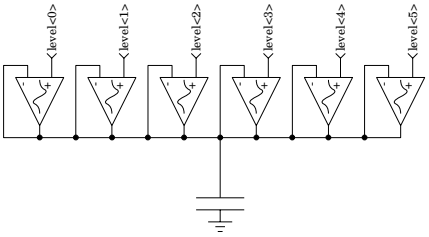

Figure 4: The schematics of the multi-level static memory. The target voltage levels can be produced globally or locally, e.g. by diode connected transistors in series.

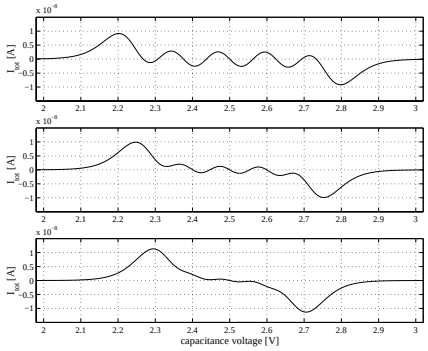

Figure 5: An illustration of the effect of spacing the attractive target voltages to tightly. The traces are computed from the theoretical formula (3). They represent the total current from the 'fusing' follower array to the capacitive storage cell in dependency of the voltage on that cell. From the top the spacing of the 5 intended attractors is $1.2d_{min}$, $1.0d_{min}$, and $0.8d_{min}$. The middle attractor is placed at 2.5V. The effective attractive states of this example memory cell can be seen as the zero crossings with negative gradient. Only in the top graph are all intended attractors effective. In the middle graph all attractors become weaker and the top most and bottom most are lost. In the lowest graph only one attractor remains.

If one now uses that output current I_{bump} as the bias current of a transconductance amplifier sharing its two inputs with the bump circuit, the formula for the output current will be

$$I_{out} = I_{bump} \tanh(\frac{\kappa \Delta V}{2}) \quad (3)$$

The result is a transconductance amplifier that only operates for small differences of input voltages, turning itself off for big differences. Circuit level simulations of the characteristics are shown in figures 2 and 3. In contrast to (3) they are slightly asymmetric. This is mainly due to the offset of the transconductance amplifier output which is slightly shifted to the right relative to the bump circuit output. Thus the slight asymmetry in the product of the two. This 'fusing' amplifier characteristics is very similar to the one of the 'resistive fuse', proposed in [6]. The difference is that the circuit in this paper is not a resistive element: it does not draw any current from the input, and the circuit is much simpler.

If now a capacitive storage cell is connected to an array of those 'fusing' amplifiers, that are connected as followers and that receive different target voltages as inputs (figure 4), then the voltage level of the capacitor will on a long timescale always settle on the closest attractive target voltage (see figure 6). If the attractors come close to Vdd, i.e. within the subthreshold range of pFET transistors, the 'fusing' amplifier can be changed to a nFET version (by exchanging all pFET transistors with nFET transistors and vice versa and exchanging Vdd and Gnd.)

The theoretical lower limit for the spacing of the attractors d_{min} is given by the distance of the maximum and the minimum of the characteristics of the 'fusing' amplifier. The total current flowing into the storage capacitance I_{tot} is the sum of all currents of the 'fusing' followers. If the attractors are spaced to closely, such that the maximum and the minimum current output of two neighbouring followers overlap, then they will cancel out each other. This effect is illustrated in figure 5. To achieve many attractors it is thus desirable to have 'fusing' followers with a 'narrow' output characteristics. How can this be achieved? The position of the two extrema of this function (equation (3)) that define d_{min} are the zero crossings of the derivative. According to (3), changing the bias current I_b will not affect them, since it is only a constant factor and will thus not influence the zero crossings of the derivative. This statement is contradicted by circuit level simulation which shows a slight dependency of this width on the bias current (see figure 2) and to achieve a small width a low bias current is advantageous. It cannot be chosen too small though, otherwise the attractors become too weak and leakage currents will be stronger. Another parameter that influences the characteristics width is S. Intuition may tell us that if the 'bump' becomes narrower then also the width of the 'fusing' amplifier characteristics becomes narrower. It was already stated that changing the design parameter S influences the width of the 'bump'. And circuit level simulation confirm that by having S small, also the width of the amplifier characteristics decreases (see figure 3). If S should be smaller than 1 though the circuit layout grows proportionally in size.

3. RESULTS

Simulations have been conducted using simulation parameters of the 0.6μm AMS process. S was chosen to be 1 and the bias voltage was 4.1V. Thus from simulations of the 'fusing' amplifier the minimal distance for the attractors would be 119.6mV. The empirical limit from the simulations was somewhat bigger just below 140mV. This would allow for 35 different attractors between 0V and 5V. In the simulation that is shown in figure 6 the distance of the 6 attractors was 215mV and they were equally spaced between 1.29V and 0.215V. The initial condition of the output in the simulation

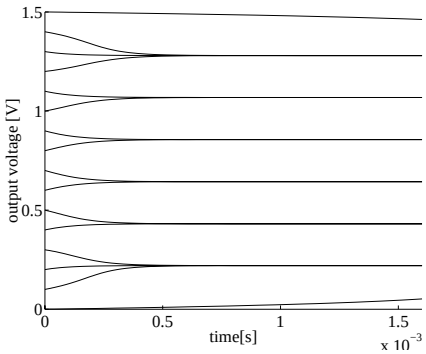

Figure 6: The graph illustrates the basins of attraction of the memory cell in a circuit simulation. The voltage traces are measured on the capacitance in figure 4, starting from different initial voltages.

were set to voltages between 0V and 1.4V with 100mV spacing. The bias voltage of the 'fusing' followers was set to 4.1V. The capacitance was 8.6 pF. All transistor used were square with W=L=1.4μm. The voltages settle to the nearest attractor within 500μs. The exception are the traces that starts at 0V and 1.5V. They are outside the basin of attraction of the lowest level and the topmost level. Thus they move only very slowly and only start to approach their nearest attractor more rapidly about 3ms later.

The speed at which the voltages settle can be adjusted by adjusting the ratio between the bias current and the capacitor size. Note though that by increasing the bias current the distance between the maximum and the minimum of the amplifier characteristics and thus the minimal spacing of the attractors is increased too.

4. CONCLUSION

A novel 'fusing' transconductance amplifier has been used to design a multi-level static memory cell. This memory cell has been conceived to be used for weight storage in a model of a learning synapse where floating gate non-volatile storage had been used before [5, 4]. The device is generally suited in asynchronous applications that require easily changeable analog storage on a short time scale, but that require only multi-level storage on a long time scale. In comparison with floating gate storage it offers better matching properties (especially no change of write efficacy over time) and easier handling (i.e. no high voltage is necessary). These improvements are traded for a limited resolution for long term storage and a larger circuit. Also digital storage in combination with AD/DA conversion would offer better voltage resolution. Advantages of the multi-level static memory cell of this paper for the intended applications are its continuous time mode of operation, without need for any control signals, and real analog storage on a short time scale.

5. REFERENCES

[1] T. Delbrück. Bump circuits. In *Proceedings of International Joint Conference on Neural Networks*, volume I, pages 475–479, Seattle Washington, July 1991.

[2] C. Diorio, P. Hasler, B. A. Minch, and C. Mead. A floating-gate MOS learning array with locally computed weight updates. *IEEE Transactions on Electron Devices*, 44(12):2281–2289, December 1997.

[3] C. Diorio, S. Mahajan, P. Hasler, B. Minch, and C. Mead. A high-resolution non-volatile analog memory cell. *Proc. IEEE Intl. Symp. on Circuits and Systems*, 3:2233–2236, 1995.

[4] P. Häfliger. *A spike based learning rule and its implementation in analog hardware*. PhD thesis, ETH Zürich, Switzerland, 2000. http://www.ifi.uio.no/~hafliger.

[5] P. Häfliger and C. Rasche. Floating gate analog memory for parameter and variable storage in a learning silicon neuron. In *IEEE International Symposium on Circuits and Systems*, volume II, pages 416–419, Orlando, 1999.

[6] J. Harris, C. Koch, and J. Luo. Resistive fuses: Analog hardware for detecting discontinuities in early vision. In C. Mead and M. Ismail, editors, *Analog VLSI Implementations of Neural Systems*. Kluwer, 1989.

[7] P. Heim and M. A. Jabri. Long-term CMOS static storage cell performing AD/DA conversion for analogue neural network implementations. *Electronic Letters*, 30(25), December 1994.

[8] M. Holler, S. Tam, H. Castro, and R. Benson. An electrically trainable artificial neural network (ETANN) with 10240 'floating gate' synapses. *International Joint Conference on Neural Networks*, (II):191–196, June 1989.

[9] C. A. Mead. Neuromorphic electronic systems. *Proc. IEEE*, 78:1629–1636, 1990.