A spike based learning rule and its implementation in analog hardware

Author(s):
Häfliger, Philipp D.

Publication Date:
2000

Permanent Link:
https://doi.org/10.3929/ethz-a-003886415

Rights / License:
In Copyright - Non-Commercial Use Permitted
A Spike Based Learning Rule and its Implementation in Analog Hardware

A dissertation submitted to the

SWISS FEDERAL INSTITUTE OF TECHNOLOGY ZÜRICH

for the degree of

Doctor of Natural Sciences

presented by

Philipp D. Häfliger
Dipl. Inf. Ing. ETH
born 18th March 1970
citizen of Richenthal, Reiden, Willisau Land und Willisau Stadt, LU

accepted on the recommendation of
Prof. Dr. Rodney J. Douglas, examiner
Prof. Dr. Wulfram Gerstner, co-examiner

2000
Acknowledgments

I want to thank
Misha Mahowald, who started all this and sadly could not see it finished;
Adrian Whatley, who is responsible for all the good English in this thesis.
I’m to be blamed for the bad;
all of the Institute of Neuroinformatics, especially the hardware group, for
patiently answering all my questions and helping me along on my way;
the Centre Suisse d’Électronique et de Microtechnique in Neuchâtel for
funding my position.
The project was funded by a Swiss National Science Foundation Schwerpunkt-
Programm grant.
And last but not least I thank Rodney Douglas for giving me the opportu-
nity to work on this project, and Wulfram Gerstner for his editorial comments.
Seite Leer /
Blank leaf
Abstract

In chapter one, this thesis introduces a local learning rule that describes a dynamics of the synaptic weights of artificial spiking neurons. The weight dynamics depend on the timing of pulses arriving at the synapses and departing from the soma. It is called the 'Modified Riccati Rule' (MRR), since it is derived from the Riccati Rule (RR). The rule that is called RR here is a non-Hebbian, rate-based learning rule. It is based on a Riccati equation and has been introduced in [41].

The MRR increases synaptic efficacies of synapses which receive input spikes that show causal relationship with the neuron’s action potentials (AP = output-spike). In that respect it resembles the learning behaviour observed in biological neurons in recently published experiments [57, 14]. The MRR in combination with an integrate and fire (i&f) neuronal model, and RR-neurons the output of which is the scalar product of the vector of input-frequencies and the weight-vector share the property of preserving the length of the vector of synaptic weights. That is to say, they normalize the weight-vector implicitly.

A MRR-neuron that receives Poisson distributed spike inputs that do not show temporal correlations with each other and with a Poisson distributed spike output expresses the same behaviour as the RR if the time constant of the neuron's membrane voltage and of the MRR's internal vector of synaptic 'correlation signals' is long compared with the average inter-spike interval of the APs. If that time-constant is relatively short as compared to the average output interval, the MRR behaves like Oja's Rule, a rate based Hebbian learning rule that also normalizes the weight-vector. If a MRR-neuron uses an i&f mechanism to produce APs the synapses, driven by the average input, compete in a hysteretic soft winner take all manner for weight. If the input spikes in addition show temporal correlations, synapses that tend to fire together gain an advantage in that competition. This latter property can be used to
train a MRR-neuron to be specific to temporal and/or spatio-temporal patterns. Thus in a network of MRR-neurons a concrete encoding of sensory spike input can be learnt that binds features to objects and that binds sequences of percepts. A simple example of such a learned encoding is simulated.

Detailed neural models like i&f neurons and MRR-synapses are computationally expensive and big network simulations are hard to compute on a serial computer in real-time. Therefore chapter two presents a aVLSI neuromorphic chip that approaches the behaviour of the MRR and that is able to simulate it in real-time.

To achieve the slow continuous dynamics of learning and to achieve non-volatile analog storage, floating gate (FG) analog memory cells [38, 48, 17, 32] are used to hold the synaptic weights and most biases. With those memory cells an analog FG memory was developed that can be accessed from off-chip by addressing and so we refer to it as FG analog random access memory (FGaRAM). The accuracy of the FG memory cells was observed for one week and determined to be better than the equivalent of 10-bit digital storage.

The chip model with long membrane and correlation signal time constants matches the predicted behaviour of the RR well and approximately normalizes the weight-vector. With a shorter time-constant only the relative synaptic strengths match the ones predicted by the MRR. The length of the weight-vector however is correlated with the length of the input-vector and so the weight-vector normalization is incorrect. The sensitivity of the MRR to temporal correlations of the inputs can also be observed in the chip model. Mismatches of the individual synapses' behaviours could mainly be blamed on documented mismatches of the tunnelling rates of the FGs. Multi-neuron experiments failed because of an unwanted behaviour of the particular i&f model that caused cross talk effects. However, suggestions of how to correct this in future designs are given. Adding a learning synapse to a more detailed neuromorphic aVLSI model of a neuron (a so called 'silicon neuron') it was possible to replicate observations of changes of synaptic efficacies from a recent physiological experiments described in [57, 14].

In chapter three, methods of integrating multiple MRR-neurons in a single chip and of integrating such a chip in an embedded system (the silicon cortex (SCX) project) currently under development at the Institute of Neuroinformatics in Zürich are presented. A chip design framework is developed that allows any VLSI model that uses temporal encoding (neuromorph) to be used instead of the MRR-neurons.

Arbitered address-event representation (AER) [55, 53, 49, 56, 61, 60, 9, 62]
is the communication protocol that is used to connect the neuromorphs’ spike inputs and APs with off-chip sources and targets. The arbitration circuits have been designed to successfully improve the speed of the arbitration and thus the maximal event frequency. Other approaches to increase arbitration speed however [9] were even more successful.

A memory access mechanism is also part of the framework. It allows access to any instrumented point on the chip. The selected ‘pixel’ is permitted to drive the voltage of a common analog output line that can be observed off-chip. Hence, read-out from memory cells or the observation of voltage variables can be achieved. If the selected pixel is a memory cell, global signals control the content of the currently selected cell.

The chip was interfaced to the SCX system without problems. Event communication and memory access work faultlessly.
Zusammenfassung


Ein MRR-Neuron mit Poissonverteilten Eingangsimpulsen, welche untereinander keinerlei zeitlichen Korrelationen aufweisen, und Poissonverteilten Aktionspotentialen verhält sich genau wie ein RR-Neuron, wenn die Zerfallszeitkonstante des Membranpotentials und der MRR-definierten synaptischen ‘Korrelationssignale’ im Vergleich zum durchschnittlichen AP-Intervall lange ist. Wenn allerdings diese Zeitkonstante relativ zum AP-Intervall kurz ist, dann entspricht das Verhalten demjenigen von Oja’s Lernregel. Oja’s Regel ist eine Hebbsche Lernregel, die wie die RR auf Durchschnittsfrequenz-Signalen


Im Kapitel drei werden die Methoden vorgestellt, um eine grössere Anzahl von aVLSI MRR-Neuronen auf einem einzigen Chip zu integrieren, der wiederum in ein integriertes System (‘silicon cortex’ SCX) passt, das ebenfalls am Institut für Neuroinformatik der ETH und der Universität Zürich entwickelt wird. Ein generelles Chip-Design-Gerüst wurde entwickelt, das es erlaubt, ein beliebiges VLSI-Modell, das Impuls-Kodierung verwendet (sogenannte ‘Neuromorphs’), anstelle von MRR-Neuronen zu verwenden.

‘Arbitered Address Event Representation’ (AER) [55, 53, 49, 56, 61, 60, 9, 62] wird als Kommunikationsprotokoll verwendet, das die Eingangsimpulse und die APs mit Zielen und Quellen ausserhalb des Chips verbindet. Das Tempo der Zuteilung von Kommunikationszeit (‘Arbitration’) und damit die Höchstfrequenz der Impuls-Kommunikation wurde gegenüber früheren Versionen erhöht. In der Zwischenzeit jedoch hat eine andere Forschungsgruppe mit einem anderen Verbesserungsansatz dieses Tempo noch übertroffen [9].


Seite Leer / Blank leaf
Contents

Acknowledgements i
Abstract iii
Zusammenfassung vii
Contents xi
List of Figures xv

1 Introduction 1
1.1 Spike based learning rules .............................................. 1
1.1.1 Learning ............................................................... 1
1.1.2 Artificial neural networks .......................................... 2
1.1.3 Supervised, reinforcement, and unsupervised learning ....... 2
1.1.4 Hebbian learning ..................................................... 3
1.1.5 Rate coding and rate based learning ......................... 4
1.1.6 Temporal coding and spike based learning ................. 5
1.2 Neuromorphic aVLSI ................................................... 6
1.2.1 Sensory neuromorphic devices .................................. 7
1.2.2 Processing neuromorphic devices ............................ 8
1.3 Analog storage in aVLSI .............................................. 9
1.3.1 Analog dynamic memory ........................................ 9
1.3.2 Analog non-volatile memory .................................... 10
1.3.3 Analog static memory ............................................ 11
1.4 Asynchronous spiking networks in VLSI ....................... 12
1.4.1 Infinite channel capacity ........................................ 13
1.4.2 Limited channel capacity .......................... 13

2 The modified Riccati rule (MRR) ......................... 17
   2.1 The Riccati rule (RR) .............................. 18
   2.2 The Riccati rule for spiking neurons (SRR) ........ 20
   2.3 Integrate and fire neurons .......................... 24
   2.4 Synaptic correlation signals ....................... 31
   2.5 The modified Riccati rule (MRR) .................. 32
   2.6 ‘Rate behaviour’ of the MRR ....................... 35
      2.6.1 Independent Poisson-distributed input and output spike-trains .......................... 36
      2.6.2 Independent Poisson-distributed inputs and an i&f neuron .................. 41
      2.6.3 Temporally correlated inputs and an i&f neuron .......................... 51
   2.7 Single neuron properties ............................ 51
   2.8 Network properties ................................ 54
      2.8.1 Network stability during learning ............... 54
      2.8.2 A coding scheme for spiking networks .......... 56
      2.8.3 Learning a temporal code ........................ 59
   2.9 Conclusion ........................................... 63

3 Analog VLSI implementations .............................. 65
   3.1 Analog parameter storage on floating gates .......... 65
      3.1.1 Reliability ...................................... 71
   3.2 MRR with non-volatile weight storage ................ 72
      3.2.1 Theoretical analysis ............................ 72
      3.2.2 Empirical analysis .............................. 83
      3.2.3 Practical problems of the implementation ....... 92
   3.3 An aVLSI model of biological LTP and LTD ........... 97
      3.3.1 The ‘silicon neuron’ ............................ 97
      3.3.2 Regulation of synaptic efficacy .................. 100
   3.4 Conclusion ........................................... 100

4 Integrating MNC into SCX ................................ 103
   4.1 AER in SCX ............................................ 104
   4.2 Receiving AE from SCX .............................. 107
   4.3 Sending AE to SCX ................................... 112
   4.4 Sender Performance .................................. 129
   4.5 Conclusion ........................................... 132
## 5 Discussion

5.1 Contribution in the field of learning theory ............................................. 136
5.2 Contribution in the field of encoding ......................................................... 138
5.3 Contribution to neurophysiology ............................................................... 138
5.4 Contribution to aVLSI engineering ........................................................... 139

<table>
<thead>
<tr>
<th>A</th>
<th>Complete schematics of SCX MNC</th>
<th>143</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>An old version of a plastic FG synapse</td>
<td>159</td>
</tr>
<tr>
<td>C</td>
<td>Very basic circuits</td>
<td>163</td>
</tr>
</tbody>
</table>

Bibliography .......................... 165
Index ................................... 172
Curriculum Vitae ..................... 177
Seite Leer / Blank leaf
List of Figures

1.1 Floating gate ................................................. 15
1.2 Basic AER ..................................................... 16

2.1 RR weight-vector normalization ................................. 19
2.2 SRR example traces ........................................... 21
2.3 SRR weight-vector normalization ................................. 22
2.4 i&f example traces ........................................... 25
2.5 MRR example traces ........................................... 33
2.6 MRR weight-vector normalization ................................. 34
2.7 Evolution of the weights in the rate model of the MRR ........ 37
2.8 Evolution of the difference between the norm and the predicted norm in the rate model of the MRR ......................... 38
2.9 Evolution of the weights in the rate model of the MRR with an i&f neuron ........................................... 42
2.10 Evolution of the difference between the norm and the predicted norm in the rate model of the MRR with an i&f neuron ........ 43
2.11 Example of discrete probability distribution of the membrane of an i&f neuron with Poisson inputs at low resolution ........ 49
2.12 Example of discrete probability distribution of the membrane of an i&f neuron with Poisson inputs at high resolution .......... 50
2.13 Simultaneously stimulated synapses ............................. 52
2.14 Learning temporal correlations .................................. 53
2.15 Feature/Object binding by simultaneous spike events .......... 57
2.16 Temporal binding by synfire chains .............................. 58
2.17 Connection scheme for an orientation and a direction sensitive neuron .................................................. 60
2.18 Statistics on direction and orientation selective neurons .......... 62
# LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
<td>Labeling examples</td>
<td>66</td>
</tr>
<tr>
<td>3.2</td>
<td>Memory cell core</td>
<td>67</td>
</tr>
<tr>
<td>3.3</td>
<td>High voltage switch</td>
<td>68</td>
</tr>
<tr>
<td>3.4</td>
<td>FGaRAM</td>
<td>69</td>
</tr>
<tr>
<td>3.5</td>
<td>Selectable analog FG memory cell</td>
<td>70</td>
</tr>
<tr>
<td>3.6</td>
<td>Monostable</td>
<td>73</td>
</tr>
<tr>
<td>3.7</td>
<td>Active high RS flipflop</td>
<td>74</td>
</tr>
<tr>
<td>3.8</td>
<td>MRR neuron</td>
<td>75</td>
</tr>
<tr>
<td>3.9</td>
<td>MRR synapse</td>
<td>76</td>
</tr>
<tr>
<td>3.10</td>
<td>Non-learning synapses</td>
<td>77</td>
</tr>
<tr>
<td>3.11</td>
<td>‘Learn up’ pulse emitter</td>
<td>78</td>
</tr>
<tr>
<td>3.12</td>
<td>‘Learn down’ pulse emitter</td>
<td>79</td>
</tr>
<tr>
<td>3.13</td>
<td>Soma schematics</td>
<td>80</td>
</tr>
<tr>
<td>3.14</td>
<td>Timing diagram of signals in MRR-chip synapse</td>
<td>81</td>
</tr>
<tr>
<td>3.15</td>
<td>The current version of an SCX multi neuron chip (MNC) : an Orbit 2.0μm bi-CMOS chip</td>
<td>84</td>
</tr>
<tr>
<td>3.16</td>
<td>Stimulation patterns, chip weights and theoretical weights</td>
<td>85</td>
</tr>
<tr>
<td>3.17</td>
<td>Chip weights vs. theoretical weights</td>
<td>86</td>
</tr>
<tr>
<td>3.18</td>
<td>Chip weight-vector norm</td>
<td>87</td>
</tr>
<tr>
<td>3.19</td>
<td>Stimulation patterns, chip weights and theoretical weights for MRR</td>
<td>89</td>
</tr>
<tr>
<td>3.20</td>
<td>Chip weights for MRR vs. theoretical weights</td>
<td>90</td>
</tr>
<tr>
<td>3.21</td>
<td>Chip weight-vector norm for MRR</td>
<td>91</td>
</tr>
<tr>
<td>3.22</td>
<td>Effect of coincident inputs on learning on chip</td>
<td>93</td>
</tr>
<tr>
<td>3.23</td>
<td>Decrease rates of memory cells when tunneling</td>
<td>94</td>
</tr>
<tr>
<td>3.24</td>
<td>Increase rates of memory cells when injecting hot electrons</td>
<td>95</td>
</tr>
<tr>
<td>3.25</td>
<td>One-compartment silicon neuron</td>
<td>98</td>
</tr>
<tr>
<td>3.26</td>
<td>Emulating LTP and LTD</td>
<td>99</td>
</tr>
<tr>
<td>4.1</td>
<td>SCX AER</td>
<td>104</td>
</tr>
<tr>
<td>4.2</td>
<td>SCX MNC framework pixel</td>
<td>105</td>
</tr>
<tr>
<td>4.3</td>
<td>SCX MNC framework example</td>
<td>106</td>
</tr>
<tr>
<td>4.4</td>
<td>AER receiver</td>
<td>108</td>
</tr>
<tr>
<td>4.5</td>
<td>Stackable one bit demultiplexers</td>
<td>109</td>
</tr>
<tr>
<td>4.6</td>
<td>Receiver logic</td>
<td>110</td>
</tr>
<tr>
<td>4.7</td>
<td>Latch</td>
<td>110</td>
</tr>
<tr>
<td>4.8</td>
<td>Learning MRR-synapse on SCX multi neuron chip</td>
<td>111</td>
</tr>
</tbody>
</table>
4.9 Memory cell extended to be used for weights in a SCX MRR-neuron .................................................. 112
4.10 A memory cell pixel to be used in the SCX MNC ................................................................. 113
4.11 Circuit to control a probe point on the SCX MNC ............................................................... 114
4.12 4-phase handshake and SCX handshake .................................................................................. 115
4.13 AER sender ............................................................................................................................ 116
4.14 Soma on SCX MNC ............................................................................................................... 117
4.15 Arbiter cell ............................................................................................................................. 118
4.16 Arbiter logic ........................................................................................................................... 119
4.17 Active low RS flipflop .............................................................................................................. 120
4.18 pull up and pull down transistors ............................................................................................ 121
4.19 Arbiter logic in the pixel ....................................................................................................... 122
4.20 Edge flipflop ........................................................................................................................... 123
4.21 latchpad .................................................................................................................................. 123
4.22 2D-Arbiter signal sequence: schematics .................................................................................. 124
4.23 2D-Arbiter signal sequence: schematics .................................................................................. 125
4.24 2D-Arbiter signal sequence: schematics .................................................................................. 126
4.25 2D-Arbiter signal sequence: schematics .................................................................................. 127
4.26 2D-Arbiter signal sequence: timing diagram .......................................................................... 128
4.27 AE raster plots ....................................................................................................................... 130

A.1 Complete SCX MNC .................................................................................................................. 144
A.2 Magnification of the left pads of SCX MNC ............................................................................. 145
A.3 Magnification of the right pads of SCX MNC ........................................................................... 146
A.4 Magnification of the top pads of SCX MNC ............................................................................. 147
A.5 Magnification of the bottom pads of SCX MNC ....................................................................... 148
A.6 Magnification of the top left corner of SCX MNC .................................................................... 149
A.7 Magnification of the bottom right corner of SCX MNC ............................................................ 150
A.8 Magnification of the middle left part of the core of SCX MNC ................................................. 151
A.9 Magnification of the bottom left corner of the core of SCX MNC .............................................. 152
A.10 UD latch used on the SCX MNC ............................................................................................ 153
A.11 Wide follower and wide comparator ....................................................................................... 154
A.12 FG memory pixel that is used on the SCX MNC .................................................................... 155
A.13 Soma pixel that is used on the SCX MNC ................................................................................. 156
A.14 Synapse pixel that is used on the SCX MNC ........................................................................... 157
A.15 RS-flipflop pixel that is used on the SCX MNC ....................................................................... 158
A.16 Tristate output pad that is used on the SCX MNC .................................................................. 158

B.1 Former learning synapse .......................................................................................................... 160
LIST OF FIGURES

B.2 Trigger circuit ............................................. 161

C.1 NAND, NOR and NOT ...................................... 164
Chapter 1

Introduction

1.1 Spike based learning rules

1.1.1 Learning

One definition of learning is that it is a process that increases the performance of a system, if the system is repeatedly exposed to similar experiences. When modellers speak of learning, they refer to rules for the gradual and 'sensible' adaptation of the parameters of a function, such that the function's performance in a certain task increases. Let's call the function $O(W, I, t)$; where $t$ is the time, $W$ (e.g. a matrix, a string, a set) contains the parameters that are slowly changed by learning, and $I$ is the function's input. Learning adapts the parameters $W$ (the memory) while experiencing some input $I$, and so changes the behaviour of the function $O$ in a useful way. Thus learning can be expressed as a rule for the changes $\frac{dW}{dt} = f(W, I, O, t)$ of $W$.

This can be compared to human learning as follows. It is believed that information is stored in the brain by adapting the synaptic efficacies, i.e. the strength and dynamics of the connections between brain cells (neurons). These connection properties correspond to $W$ in the mathematical model. In our brain there are approximately $10^{15}$ synapses (the connection sites between neurons) so the brain's storage capacity is enormous.

1 A conservative estimate would be that every synapse can have 256 (equivalent of 8...
then be the function $O$ as we interact with our environment perceiving sensory input $I$.

1.1.2 Artificial neural networks

In artificial neural networks (ANN) the function $\hat{O}(W, \vec{I}, t)$ is a network of $n$ model neurons that are described by state variables $S_i$ that express the individual neurons' outputs. These outputs are for example a real number expressing an activity level. The neurons have directed connections with each other with weights (=connection strength) $W$ (a $n \times n$ matrix). $\vec{I}$ (the input-vector) sets the states of a selection of neurons that are defined to be the input-neurons. The states of all other neurons (or their derivatives in time continuous models) are a function of the states or past states of the neurons that have a forward connection to them and the weights of these connections. The states $S_i$ of a set of selected individual output-neurons form the output vector $\hat{O}$ of the net.

Many learning rules for ANNs have been formulated, and they operate on just as many different types of model neurons. They can be categorized as follows.

1.1.3 Supervised, reinforcement, and unsupervised learning

Learning rules can be roughly classified as supervised, reinforcement or unsupervised learning algorithms. Supervised learning algorithms and reinforcement algorithms are optimization algorithms. They optimize a measure or estimate $P(\vec{I}, \hat{O}, t)$ of the performance of the neural net. For supervised learning algorithms the correct net-output $\hat{D}(\vec{I}, t)$ is known for a subset of all possible inputs $\vec{I}$ and the performance measure $P$ is an error function defined on the differences $\hat{D} - \hat{O}$. Supervised learning tries to minimize that error function.

The most successful supervised learning rule is the Error Backpropagation algorithm [66], that can now be found in every standard introduction to the bits) different states of efficacy. That makes the total storage capacity $10^{15}$ bytes. So we comfortably out rank the storage capacity of modern computers.
field of ANNs e.g. [37, 65]. The neuronal model that the error backpropagation algorithm is usually operated on is the perceptron in a multilayer network.

In reinforcement learning the performance measure $P$ is more vague than in supervised learning. It can for instance be a reward signal, indicating that the behaviour of the function $O$ lead to success. One reinforcement learning algorithm that has become famous in the last few years is Sutton's TD$\lambda$-algorithm [69]. It has been used by G. Tesauro to train a computer Backgammon player which has become as good as the best human players [70].

Unsupervised learning is not guided by some evaluation of a performance. It adapts the free parameters $W$ according to statistical properties of the input. It is for example used for data reduction (vector quantization, data compression). In that case it tries to achieve an optimal representation of input data. Also in our brain unsupervised learning is believed to extract relevant data out of the huge amount of sensory input that our sensory receptors receive.

One particular class of unsupervised learning rules that are often used are the so called Hebbian learning rules. Since the learning rule that is introduced in this thesis can be considered Hebbian, Hebbian learning will be described in more detail in the following section.

### 1.1.4 Hebbian learning

Hebbian learning rules are local unsupervised learning rules. Local learning rules are rules that treat every neuron individually. That means that the weight changes cannot depend on network performance. The rule must be capable of being fully expressed in terms of the input and output of the individual neurons. Local learning rules can be expressed neuron-wise as

$$\frac{d\vec{w}}{dt} = f(\vec{w}, \vec{I}, O, t)$$

where $O$ is now the neurons only output (formerly referred to as $S_i$), $\vec{w}$ is the neurons weight-vector and $\vec{I}$ its input-vector. In a network $\vec{I}$ is formed by other neurons outputs and/or 'external' sensory inputs. Using

\[\text{2The error backpropagation method optimizes } P \text{ by changing the weights } W \text{ according to the gradient descent method. For the gradient descent method the gradient } \frac{dP}{dW} \text{ of the function } P \text{ that is to be optimized with respect to every parameter } W \text{ is computed. In the case of the error backpropagation algorithm } P \text{ is an error function and therefore needs to be minimized. So } W \text{ changes with the opposite sign of the derivative: } \frac{dW}{dt} = -\mu \frac{dP}{dW}. (\mu \text{ is a parameter called 'learning rate'.})\]
local learning rules does not necessarily preclude a global behaviour of the neurons' weights, since the neurons do interact with each other via their normal inputs and outputs. Global patterns in the neurons' learning behaviour can emerge through self-organization.

Hebbian learning algorithms are so named after a quote in a publication by the psychologist D. O. Hebb. He formulated 'a neurophysiological postulate' [35] on which thereafter many learning rules have been based:

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.

Many 'Hebbian' learning rules rest on his postulate and contain a positive term $\Delta w$ in the change of weights $\frac{\Delta w}{\Delta t}$. Thus they reward activity correlations between a synapse's input and the neuron's output by increasing that synapse's weight. That correlation in activity of a cell A and B was considered equivalent to 'cell A ... repeatedly or persistently takes part in firing it (B)'.

1.1.5 Rate coding and rate based learning

To keep the simulations of neuronal models computable in a reasonably short time, the amount of detail in the model has always been kept to a functional minimum. So as was mentioned earlier a model neuron's activity is often expressed as a real number, whereas a biological neuron issues a train of action potentials. The model neuron's real number output stands for an average firing rate of the biological analogue. For a time it was assumed that the spike-patterns in biological neurons were more or less random and the information was coded in the frequencies only. Therefore such neuronal models were considered accurate enough. The learning rules that operate on networks of rate coding neurons are naturally formulated on rate signals. And indeed rate based neuronal models and rate based learning rules are successful. All the learning models mentioned so far are rate based models. And also in physiological experiments, by just looking at the rate response of neurons one can reconstruct many aspects of the stimulation that caused the response.
1.1.6 Temporal coding and spike based learning

More recently it has been questioned, whether the information contained in the rate of a neuron was really all the essential information that there is. There are many indicators that the exact temporal pattern of the action potentials (APs, nerve pulses) is important for the information processing in the brain, for example subjects respond to a stimulus with latencies so short, that only a few spikes per neuron can occur along the processing path [71]. Furthermore it has been shown that the exact spike pattern derived from a cell in visual cortex responding to the same visual random dot stimulus is not random at all, but can be reproduced with high accuracy [6]. So there is more information hidden in the spike pattern than can be seen in the average frequency. Thus researchers have begun to investigate computational power of spiking neuron models [52]. In this book it has been proven that temporal coding neurons are at least as powerful as a subset of rate coding neurons, the so called McCulloch-Pitts neurons that receive a binary vector as input and emit a 1 if the weighted sum of the inputs exceed a threshold and a 0 otherwise. It has been shown that any multilayer networks of these McCulloch-Pitts neurons can be emulated by neurons using temporal coding ([52], chapter 2).

Learning rules depend on the coding of the signals involved. If the temporal pattern of the spike-trains carries the relevant information, also the experience dependent changes of the parameters in this system should depend on that information and therefore on those temporal patterns. Some learning rules have been formulated on the relative timing of spikes [25, 29, 40, 39], and have for example been used to fine-tune models of coincidence detecting neurons as found in the auditory system of owls [43, 50].

In barn owls' auditory processing it is known how azimuthal information is coded by spike-patterns, but in the rest of the brain, it is less clear how temporal coding is used. Therefore it is just as unclear what kind of learning is appropriate. However there are some experiments shedding light on how changes in synaptic efficacy depend on temporal patterns of the input and output spike-trains of neurons [57, 14]. It has been found that causal relationship between a presynaptic and a postsynaptic spike (i.e. if the presynaptic spike precedes the postsynaptic) enhance synaptic efficacy. Those findings were consistent with the spike based learning rule proposed in this thesis (chapter 2 and [29]) and the one introduced by Gerstner et al. [25], that both had already been published at that time. Both the physiological findings and the theoretical spike based learning rules are actually more to the spirit of Hebb's
neurophysiological postulate than rate based Hebbian learning rules: They do not just reward correlation but causality.

If the learning behaviour depends on the neuronal code, one can also draw some conclusions about the neural code from the learning behaviour. In this thesis there is a spike based learning rule proposed, and the thesis also addresses the question of what kind of encoding could be used with that learning rule by giving an example of such an encoding.

1.2 Neuromorphic aVLSI

There is a substantial organisational difference between a digital computer and a brain. The brain computes in a distributed manner with a large number of relatively simple and slow processing units whereas a digital computer computes serially on one or on a few central fast processors. In a computer memory and processor are spatially separated whereas in the brain the data is stored in a distributed fashion near to the processing units that use it. The brain’s structure is optimized by evolution to perform the kind of computations that let us survive. We are for example able to segment natural scenes fast, recognize objects or faces, grab for an egg without breaking it etc. A computer is a most general computation device that could, if programmed accordingly, also perform all those tasks. If there was a computer with sufficient memory it would even be capable of simulating a whole brain and therefore would be able to compute anything that a brain can compute. But with the processing speed of today’s computers it would take many orders of magnitude more time than it takes the brain because computers compute serially. So many of the tasks that come naturally to us are very hard to solve at the same speed for a computer controlled system.

One way to construct a system that is better suited than a standard computer to quickly perform the kind of computations that the nervous system can compute, is to copy more closely the structure of the analogue. It has been proposed by Carver Mead to implement neural models in ‘neuromorphic’ analog VLSI chips [58]. Analog VLSI is well suited for this task because the basic elements of analog VLSI electronics have similar properties to those of some basic elements of brain cells. For example membrane capacitance can be modelled by the capacitance between polysilicon layers on a aVLSI chip, ion currents through channels can be modelled as currents through a transistor and photoreceptors can be modelled by photo diodes. Neuromorphic aVLSI
chips can model neural processes in considerable detail in real time or even faster, since the capacitances and currents in the model can be scaled to define faster time constants than those present in the biological system. Typically in a neuromorphic system a single analog VLSI chip contains multiple processing elements.

1.2.1 Sensory neuromorphic devices

Sensory neuromorphic devices are the most well developed at present. The reason for this is, that the neural structures doing low level computations in sensation are the best studied and understood subsystems of the nervous system, so successful models of them can be built. These sensory neuromorphic chips do not just record sensory input, but also perform some distributed processing on the input data. In that sense they can be called ‘intelligent sensors’.

The most often referenced sensor in this category is the silicon retina. As in a real retina (the photosensitive part of the eye), the silicon retina [23, 59, 53] does not record pictures but performs distributed processing on the optical information to enhance edges, extract movement and to adapt locally to light intensity. This is possible because the very same material can be used to implement optical sensors and processing elements.

The silicon retina is by now a well established technology and has been used in commercial products. The optical sensor that evolved from the research with silicon retinas that has had the most commercial success is part of a Logitech rollerball, where it visually tracks the motion of the ball by observing a random pattern on the ball’s surface [5]. So unlike in most computer mice there are no wheels touching the ball, wheels that wear out and are sensitive to dirt on the ball. The ball only needs to have some random structure printed on its surface. Recent research has also started to explore the possibility of implanting silicon photoreceptor-arrays into rabbit retinas, to develop prostheses [76].

Another example of a sensory neuromorphic device is the silicon cochlea [47, 49], a frequency filter array that mimics the biological cochlea (the part of the inner ear that translates sound into nerve pulses) and shows the same characteristics as the biological cochlea. Its suitability as a front-end for cochlear implants is now being investigated [44]. So it might become part of a commercial product too in the near future.
1.2.2 Processing neuromorphic devices

In the nervous system there is no sharp distinction between sensory and processing systems. Already very early on (e.g. in the retina) sensory input is not merely recorded but processed. In this thesis we choose to classify neuromorphic systems as processing rather than sensory if they are not dedicated to a particular category of sensor (e.g. optical, auditory or olfactory).

Neuromorphic processing devices are inspired by parts of the nervous system that are less well understood than the sensory parts. Research is still far from understanding how the 'higher' brain areas process information. Many neuromorphic models of those areas are consequentially the subject of basic research and support real-time modelling of parts of the brain.

An example of a neuromorphic processing device is the silicon neuron [54]. It is a compartemental model of a neuron, modelling in detail different processes of a biological nerve cell. It allows many aspects of a neuron to be modelled in real time. Therefore it can process information received by sensors observing a real environment and it can be used by brain researchers interested in real time models of networks of neurons.

Learning is also starting to be implemented in neuromorphic processing systems [4, 29, 19, 30, 16, 24]. It is of foremost interest to future neuromorphic applications, for example in semi-autonomous or fully autonomous agents, i.e. systems that must be capable of dealing with new environments and changing situations.

In section 1.1, learning was defined as a process that changes the parameters of a function. So a learning analog VLSI system needs to be able to store and adapt analog parameters. This adaptation operates on a long time constant as compared to the processing speed of VLSI circuits. So the storage mechanism needs to be able to store values over extended periods of time. This was the main consideration in choosing non-volatile analog storage on floating gates [17] for the learning neuromorphic aVLSI chip that is described in chapter 3. The next section gives a short review of analog storage and also explains the storage methods used for the work described in this thesis.
1.3 Analog storage in aVLSI

The variables and parameters of an analog VLSI system are analog voltages and currents. When the computation performed by the aVLSI system goes through states, the variables that represent those states must be stored. The choice of the method of storage depends mainly on the time period over which those values need to be stored accurately. In the following an overview of the different methods of analog storage and an explanation of where each is applicable are given.

Methods of off-chip storage are not discussed, since as mentioned earlier in the text it is considered an advantage of neuromorphic systems that the data is kept close to the processing units. Off-chip storage would mean that there is a spatial separation between memory and processing units. If analog values are stored off chip, they need to be conveyed somehow to the computational circuits and/or back. That can be done either in parallel in a non-multiplexed way, or serially (multiplexed). The former means that there needs to be a physical connection for every value. Since a VLSI chip has only a limited number of connecting pins available, this is only possible for small numbers. In the latter scenario the values are not continuously available for the processing units. For a multi-unit chip that only makes sense if the processing units have additional local storage capacity. So there would still be some form of on-chip memory required.

1.3.1 Analog dynamic memory

Dynamic memory is memory that can only hold its values for a limited amount of time. Dynamic memory in analog as in digital is typically achieved using simple capacitive storage. Voltage storage on capacitors is the most straightforward and simple form of analog memory. It does not require sophisticated circuitry. The capacitance can be charged and discharged through a single transistor. This makes capacitive storage very space efficient. High densities of storage cells are possible in this way. The major disadvantage is the leakiness of the transistor used for writing. Charge leaks from the capacitor through the channel of the ‘closed’ transistor such that the voltage changes typically in the order of millivolts per second. Therefore capacitive storage is best suited for applications where only short term memory is needed.

Examples for applications of capacitive analog short term storage are ana-
log sample and hold circuits (e.g. [74, 13]), circuits where a continuous analog signal is sampled at regular intervals and kept constant on a capacitor to allow some computation on the signal. Capacitive storage has also been used in continuous learning processes, where the values are updated at short intervals (e.g. [4, 29, 19, 30]). It is also used in this thesis in the aVLSI model of a learning neuron (chapter 3) to hold variables that are actually supposed to decay.

Capacitive storage is not as easy to use for analog long term storage. To retain voltage for a longer period of time, a refresh-mechanism is needed. In digital that refresh is relatively easy: a capacitive storage cell is accessed regularly and refreshed to one of the two digital voltage rails (Vdd or Gnd), dependent on the voltage being below or above the digital switching threshold. In a noiseless system if the leakage is 1mV per second, Vdd is 5V and the idealized switching threshold is 2.5V it is only necessary to access every storage cell every 2500 seconds. In analog however, the problem is not that simple. One solution is to digitize the analog range of the cell into slots and to refresh to the upper bound of the slot. The refresh frequency depends on the leakage and on the width of the slots and must therefore be higher than in the digital case. The slots must be small enough such that the circuits that connect to these analog memory cells are not affected by the jitter that is caused by the refresh. Another refreshing method makes use of digital memory and an analog to digital conversion (ADC). Here the analog values are stored additionally in digital and the periodical refresh is performed by restoring the capacitors voltage to the value stored in the digital memory. A very unsatisfying aspects of this solution though is the necessity to store the values in two places.

### 1.3.2 Analog non-volatile memory

Non-volatile memory is memory that does not loose its content when the power supply is turned off. Its content is normally slow to change or not changeable at all. It is best suited to hold parameters and biases that do not change after having once been set or that change only slowly.

Hardwired current or voltage sources belong to that category. The memory content in that case is already set when designing an aVLSI chip, by including circuits that provide constant currents or voltages. An example for such a current reference circuit can be found e.g. in [73].

Sometimes however the values of analog biases and parameters are not
already known in the chip design phase. For example when those parameters are used to calibrate a chip or to achieve different modes of operation for the same chip design. The kind of non volatile memory that is applicable in those cases is the analog EEPROM: Analog charges are kept on electrically isolated conductances, so called floating gates (FG). The FG has no electrical connection to any other conductor and so the charge on it is trapped. To a first approximation it won't leak away at all. In practice it remains stable for years, as shown by the experience with digital EEPROMs. As those isolated conductances are used as gates on MOSFET transistors (figure 1.1), the charge (i.e. the voltage) on the FG determines the current that flows through those transistors and so the charge can be read unobtrusively by measuring that current. The charge on the FG can be changed either by tunneling electrons through the insulator or by accelerating electrons through a transistor, such that they gain enough energy to jump the insulation barrier. The latter method is called hot electron injection.

EEPROMs are well established in digital designs already, where the charge on the floating gate is used to turn on or turn off the read out transistor. Recently it has been proposed to use EEPROMs for analog storage too [38, 48, 17, 32, 33]: the charge can be changed in essentially arbitrarily small steps and the read out transistor can provide any desired current between the digital limits.

In the work described in this thesis, such an analog EEPROM has been used to set the operation parameters of the neural model that was implemented on an aVLSI chip (chapter 3) and to hold the synaptic efficacies that are slowly changed by a learning mechanism.

1.3.3 Analog static memory

As static memory we consider circuits that can hold their value through a local feedback loop. Static memory retains its content for long time periods and can be changed fast. It is therefore well suited to hold variables that at times need to be updated quickly but at other times only need to be stored for extended periods of time. Static memory is quicker to change than non-volatile memory but it can only maintain its content while the power supply is switched on.

Digital static memory consists of flipflops, circuits whose outputs are kept at one of the two digital voltage rails through positive feedback to itself. Only
Asynchronous spiking networks in VLSI

Neurons are organized in networks. Implementing individual neurons in aVLSI is but a first step towards aVLSI models of neural systems. Connecting these aVLSI neurons to networks, especially across chips, poses some serious problems. The task is the following: we need to transmit spike-events from sender-neurons to receiver-neurons. A receiver needs to know the identity of the sender and the time of the event. In the brain, it is not yet clear whether that is all the information that is transmitted from one neuron to another. The shape of the action potential could carry additional information. In the present work however the AP's shape is neglected. In the following we will use the general term neuromorph to refer to any device that communicates by spike-events.
1.4.1 Infinite channel capacity

If one could solve the connection problem for neuromorphs with an arbitrarily big channel capacity, one would probably design the system the way nature has done it, i.e. with enough bandwidth to run all point to point connections in parallel. In the brain neurons are connected via dedicated 'cables', the axons. Every sending neuron has its own axon that connects to all its receivers. Every receiving neuron has at least one dedicated site of reception for each of the neurons that send APs to it. The sender can thus be identified by the reception site. The amount and density of cabling this requires is enormous. VLSI chips however are mounted on standard packages that only provide a certain number of pins where connections to other electronics can be made, typically numbers in the order of one hundred. Even if one only wanted to connect ten neuromorphs on one aVLSI chip with 100 pins with 10 other neuromorphs on another chip, which would require 10 outputs and 100 inputs per chip, simply the number of pins is not sufficient. Electronics comes closer to the brain's connection density in on-chip communication than in inter-chip communication, but it is still far from what nature is capable of. A major reason for this is that VLSI layouts are mainly two-dimensional, whereas the brain makes use of all three dimensions. Further development in the integration scale and three dimensional semiconductor-chip design will also reduce the problem, but are not subject of this thesis.

1.4.2 Limited channel capacity

Individual neuron-to-synapse connections between chips to transmit action potentials are not possible because of the restricted number of chip-pins. So the available resources to transmit signals between chips (the communication channel capacity) is thus limited. It has been proposed to 'time-share on demand' a number of pins that is smaller than the number of point to point connections. Since biological neurons transmit APs with rather low average frequencies and electronics can be much faster, this is a feasible approach. Address-event representation (AER) [55, 53, 49, 56, 61, 60] is the communication protocol is the method used in the present work to achieve this. In the AER protocol one asynchronous bus is shared among multiple emitters of spike events. The bus transmits the sender's identity, i.e. its address (figure 1.2). Hence the name 'address-event representation'. In the simple one to one scenario depicted in figure 1.2 a handshake protocol (see figure 4.12) can
be used to synchronize the sender and the receiver for the duration of one transmission.

One transmission on an address-event bus only needs a couple of tens or hundreds of nanoseconds, whereas in the brain an axon cannot go faster than roughly one spike per millisecond. One bus could handle the communication for 10,000 neurons. However if a AER bus is loaded close to that theoretical capacity limit there will be collisions on the bus. This requires a form of collision handling. One way to deal with collisions is to use error detecting codes as addresses and to dispose of illegal ones [61, 2]. So actually the success of a transmission would not be guaranteed. Another way, which has been chosen here, is to perform an arbitration [55, 53, 10, 9]: neuromorphs have to apply for the bus and can only access it one at a time. That however might interfere with the timing of events, since in the case of a collision, one event gets delayed (Abitered AER will be described in more detail in section 4.3). So whichever of these two methods is chosen, it has its drawbacks and relies on the system’s tolerance to small errors. In any case, to keep the number of errors small, the load on the AER bus should be planned in a manner that keeps the number of collisions low.

Recent work improved speed in arbitered AER [9] by optimizing the arbitration (see also in section 4.3). In this thesis a different approach is introduced that also improves the sender speed (chapter 4). In other work the idea of serial AER has been introduced [62]. In serial AER the address is transmitted serially, which on one hand slows the communication down, but on the other reduces the number of necessary cables and the size of the address-space is changeable more easily (see also in section 5.4).
Figure 1.1: A floating gate (FG) and the mechanisms used to charge and discharge it as proposed in [17]. The floating gate is a physically isolated conductance, typically in a polysilicon layer on a CMOS chip. Fowler Nordheim tunneling [20] is used to charge it. The actual tunneling voltage is mainly dependent on the thickness of the SiO₂-layer insulating the floating gate from the conductance (the substrate or another polysilicon layer) to which electrons are tunneling. In the two standard processes that we used through the MOSIS service (2μm Orbit and 1.2μm AMI), since we were tunneling from the substrate through the gate oxide or through the oxide separating the two poly-silicon layers, the required voltage was between 12V and 35V. The hot electron injection [17], which discharges the FG, required voltages between 4V and 8.5V across a NFET transistor embedded in a p-base region (p-region with high doping) [17] or a normal PFET transistor [34]. Electrons are injected onto the gate through the gate oxide of these transistors as they gain energy in the high electric field at the transistors drain. The analog state of the floating gate can be observed by the change in current flowing through the readout transistor to the right.
Figure 1.2: A simple example demonstrating the principles of address-event representation (AER). Two chips are connected by an asynchronous bus. They contain elements (3 neurons with 3 synapses each) that emit and receive event signals. Events are defined by the identity of the sender and the time of their occurrence. Events are encoded by a multiplexer into an address representing the source of the event. On the receiver side those addresses are decoded by a demultiplexer and routed to the appropriate target.
The modified Riccati rule (MRR)

When we devised the 'modified Riccati rule' (MRR) we were looking for a learning mechanism for spiking neurons that would let synapses compete among each other for synaptic strength and that would reward causal relationships between inputs and outputs.

These requirements were answered by a number of learning rules that were invented during this work. Finally though we favoured MRR because it uses only information that would be available to a biological synapse, and because the MRR synapses do not merely compete for synaptic strength, rather the sum of the squares of the synaptic weights of a neuron is implicitly kept constant. In other words, the weight-vector is normalized. This can be proven by relating the MRR to a rate based learning rule that we named Riccati rule [41], with which it shares this property. The Riccati rule is based on a Riccati equation and is introduced in [41].

Later physiological findings [57, 14] showed that biological synapses performing long term potentiation and depression (LTP and LTD, the terms used by physiologists to describe permanent changes in synaptic efficacies) behave very similarly to simulated synapses following the MRR.

The following description will follow the inverse path than the one taken by us originally: We will first discuss the rate based Riccati rule (RR), then
adapt it for spiking neurons (SRR) and finally develop the modified Riccati rule (MRR).

## 2.1 The Riccati rule (RR)

The learning rule that we named Riccati rule (2.1) is a rate based learning rule and defined in [41] as a form of a Riccati differential equation\(^1\)

\[
\frac{d}{dt} \vec{w} = \alpha \vec{I} - \beta \vec{w} \hat{O}
\]  

(2.1)

where \(\hat{O}\) is the output frequency of an artificial neuron, \(\vec{I}\) is the vector of input frequencies, \(\vec{w}\) the vector of synaptic weights and \(\alpha\) and \(\beta\) are parameters. Note that the Riccati rule is not a Hebbian learning rule, since the change of the weights \(\frac{d}{dt} \vec{w}\) does not depend on the product \(\vec{I} \hat{O}\) of the input and the output and therefore correlations between input and output do not influence the learning outcome.

**Theorem 1** If \(\vec{w}\) is the weight-vector of a neuron that changes its weight according to the Riccati rule (2.1), then the attractor \(\vec{w}^*\) of the weight-vector has the same direction as the input-vector \(\vec{I}\) (2.2).

\[
\vec{w}^* = \frac{\|\vec{I}\|}{\|\vec{w}^*\|} \vec{I}
\]  

(2.2)

If the neuron output is the scalar product of its weight- and input-vector, \(\vec{w}^*\) has length \(\sqrt{\frac{\alpha}{\beta}}\) (2.3)

\[
\|\vec{w}^*\| = \sqrt{\frac{\alpha}{\beta}}
\]  

(2.3)

---

\(^1\)In [8] the 18th century mathematician Jacopo Francesco Riccati is said to have been interested for instance in differential equations of the form \(\dot{x} = ax^2 + bx + c\), with time varying or constant parameters \(a\), \(b\) and \(c\). The RR would then be such an equation without the quadratic term.
CHAPTER 2. THE MODIFIED RICCATI RULE (MRR)

Figure 2.1: The convergence of the length of the weight-vector in the Riccati rule. The curve shows the length of the weight-vector evolving over time, of a simulated neuron with two synapses receiving constant input. The synapses are modified according to the Riccati rule. The input to synapse 1 is 0.12 and the input to synapse 2 is 0.2. The parameters $\alpha$ and $\beta$ are both 0.01. The initial weights are both 0.6. As the weights settle the weight-vector’s length approaches 1.
This has already been proven in [41]. By setting \( \frac{\partial}{\partial t} \bar{w} = 0 \) we can compute the fixed point of the dynamics:

\[
\bar{w}^* = \frac{\alpha}{\beta \hat{O}} \hat{I}
\]  

(2.4)

So since \( \frac{\alpha}{\beta \hat{O}} \) is a scalar, the fixed point’s direction will follow the input-vector’s direction(2.2). Even if \( \hat{O} \) is zero the direction of \( \bar{w}^* \) will still follow that of the input-vector, only its length will be infinite. However, if in addition the input is zero, then trivially there are no dynamics and every point is a fixed point. The fixed point (2.4) is attractive, since if the weight-vector is perturbed by \( \bar{e} \), it will be restored in the direction of the fixed point:

\[
\frac{d}{dt} (\bar{w}^* + \bar{e}) = -\beta \hat{O} \bar{e}
\]  

(2.5)

If we define the output \( \hat{O} \) as the scalar product of the input-vector and the weight-vector (which is a common input/output relationship in artificial neurons)

\[
\hat{O} = \bar{w}^T \hat{I}
\]  

(2.6)

then we can compute the length \( ||\bar{w}^*|| \) of the attractor by multiplying equation (2.4) from the left with \( \bar{w}^{*T} \), then substituting \( \bar{w}^{*T} \hat{I} \) with \( \hat{O} \), which eliminates \( \hat{O} \). Finally solving for \( \bar{w}^{*T} \bar{w}^* \) leads to (2.3).

### 2.2 The Riccati rule for spiking neurons (SRR)

In this section we will slightly modify the RR to enable it to deal with neuronal models that use temporal encoding. It will be called SRR1 (Riccati rule for spiking neurons, variant 1). A modification (SRR2, Riccati rule for spiking neurons, variant 2) will be introduced later in section 2.3. SRR2 is biologically less plausible but its weight-vector normalization is more precise.

To use the Riccati rule in a spiking neuron whose inputs and outputs are defined as sequences of spike-events \( t_{i,j} \) (time of the j’th input-spike to synapse i) and \( t_k \) (time of the k’th output-spike), it is actually not necessary to adapt the rule much. It is possible to simply replace the frequency signals \( \hat{O} \) and
Figure 2.2: An example of the values of simulation variables at one synapse, which changes its weight according to the first variant of the Riccati rule for spiking neurons (SRR1) (2.14). The signals are drawn together with their baselines with different offsets. With every incoming spike (rendered visible by the function $\chi_i$, defined in equation (2.17)) the weight is incremented by $\alpha = 0.2$. With every output spike (visualized by $\chi$) the weight gets decremented by $\beta w_i = 0.2w_i$. For the second variant (SRR2) (2.27) the term for that decrement is more complicated, namely $\beta w_i \lim_{T \to 0} \int_{t_k - T}^{t_k} \tilde{O}dt'$ (see also figure 2.4), but the integral in that term will only be slightly bigger than one (by at most the maximum weight of the neuron). The difference in the graph of $w_i$ for SRR1 and SRR2 would be small and the signals would look qualitatively the same for SRR2.
Figure 2.3: The convergence of the length of the weight-vector in the SRR1 and SRR2. The curves show the length over time of a weight-vector of a simulated neuron with two synapses receiving Poisson distributed spike input with constant average frequency evolving over time in two different trials. In the trial that results in the upper, somewhat more irregular trace, the weights are changed according to the first variant of the Riccati rule for spiking neurons (SRR1) (2.14). For the second trial SRR2 (2.27) was used. The input to synapse 1 is 0.12 (spike-probability per time-unit) and the input to synapse 2 is 0.2. The parameters $\alpha$ and $\beta$ are both 0.01. The initial weights are 0.6. As the weights settle in the second trial the weight-vector’s length approaches 1. In the first it stays above that value.
with ‘instantaneous’ signals $\dot{O}$ and $\dot{I}$ and leave the rule as it stands. First we shall define precisely what we mean by frequency signals. We define that frequencies ($\dot{O}(t, T)$ and $\dot{I}(t, T)$) in a time interval $[t - T, t]$ are given by the number of spike-events divided by the interval duration $T$. Finally we want the rule to operate on the instantaneous frequencies $O(t)$ and $I(t)$, which are defined as the limit for $T \to 0$ of $\dot{O}(t, T)$ and $\dot{I}(t, T)$:

$$O(t) = \lim_{T \to 0} \dot{O}(t, T)$$

and the same definitions for $I$

$$I(t) = \lim_{T \to 0} \dot{I}(t, T)$$

As we let $T$ approach 0, $I$ and $O$ become sums of Dirac delta functions, since they show all the defining properties:

1. They are zero where there is no spike-event.

$$\left( \exists k : t_k = t \right) \iff \lim_{T \to 0} \dot{O}(t, T) = O(t) = 0$$

because $\dot{O}(t, T)$ is zero if there is no spike-event in the interval $[t - T, t]$.

2. $O(t)$ is infinite where there is a spike event at time $t$

$$\forall k : \lim_{T \to 0} \dot{O}(t_k, T) = O(t_k) = \infty$$

because there will always be the spike at time $t_k$ in the interval $[t_k - T, t_k]$. Thus the left hand term is equivalent to $\lim_{T \to 0} \int_{t_k - T}^{t_k} O(t_k) = \infty$.

3. The integral over an interval that only includes one spike-event is one:

$$\forall k : \lim_{T \to 0} \int_{t_k - T}^{t_k} O(t_k, T) = \lim_{T \to 0} \int_{t_k - T}^{t_k} O(t_k) = 1$$

because if $T$ is small enough there will only be the spike at time $t_k$ in the interval $[t_k - T, t_k]$. Again we actually integrate over $\frac{T}{T}$. The same argumentation applies to $I$. 


We write
\[ O(t) = \sum_k \delta(t - t_k) \] (2.12)
and
\[ \tilde{I}(t) = \sum_{i,j} \delta(t - t_{i,j}) \tilde{e}_i \] (2.13)
where \( \tilde{e}_i \) is the unity vector with only the \( i \)'th element being 1 and all the others are 0.

To handle these instantaneous frequencies, the Riccati rule does need one small modification for spiking neurons. That is the replacement of \( w \) with \( \tilde{w} \) on the right hand side of the equation:

\[ \frac{d}{dt} \tilde{w} = \alpha \tilde{I}(t) - \beta \tilde{w}O(t) = \alpha \sum_{i,j} \delta(t - t_{i,j}) \tilde{e}_i - \beta \tilde{w}(t) \sum_k \delta(t - t_k) \] (2.14)

where \( \tilde{w} \) is the same as \( w \) except at singular points (at times \( t_k \) and \( t_{i,j} \)), where \( w \) changes instantaneously and is therefore not defined but \( \tilde{w} \) is defined as the limit towards that singularity from the left of \( \tilde{w} \).

\[ \tilde{w}(t) = \lim_{s \to t} \begin{cases} \tilde{w}(s) & s < t \\ w(s) & s \geq t \end{cases} \] (2.15)

An example of how the weights evolve in this scenario is depicted in figure 2.2. In this text we will refer to this rule (2.14) as SRR1 (Riccati rule for spiking neurons, variant 1).

### 2.3 Integrate and fire neurons

The weight-vector normalizing behaviour of the SRR1-dynamics (2.14) is dependent on the input-output relationship. If the earlier definition (\( O \) is the scalar product \( \tilde{w}^T \tilde{I} \) of the input-vector and the weight-vector (2.6)) were chosen, again the length of a fixed point of the weight dynamics would be constant. Such a neuron would issue an output spike with every input, the integral over
Figure 2.4: An example of variables involved in an i&f neuron with three synapses as defined in (2.16) and (2.19). The signals and their baselines are drawn with different offsets. With every incoming spike-event (rendered visible by the function $\chi_i$) the membrane $M$ is incremented by the synaptic weight. $M$ decays slowly with time-constant $\tau = 3$. An AP-event (visualized by $\chi$) is triggered when that increment would push $M$ over the threshold (which by our definition (2.19) is 1). Note that $M$ never actually crosses the threshold, since in our definition (2.16) it is reset at the same instant as when a synaptic input would increment it above the threshold. $\tilde{O}$ is the value that is depicted as error-bars on $\chi$ multiplied by a delta function.
which would be the synaptic weight. Unfortunately that definition of the output would not be consistent with its definition as a sum of delta functions (2.12) and the resulting behaviour would be far from biologically plausible. There is a common neuronal model for spiking neurons however that on average approximates (2.6): the integrate and fire neuron.

The state of an integrate and fire (i&f) neuron can be described by its membrane potential $M$:

$$\frac{d}{dt} M = -\frac{1}{\tau} \tilde{M} + \tilde{\omega}^T \tilde{T} - (\tilde{M} + \tilde{\omega}^T \tilde{\chi}) O \quad (2.16)$$

Here $\tilde{M}$ is analogous to $\tilde{\omega}$ in equation (2.15). It is the same as $M$ except for singular points, where it is equal to the limit from the left of $M$. $\tilde{\chi}$ is defined as:

$$\tilde{\chi}(t) = \lim_{T \to 0} \int_{t-T}^{t} \tilde{T} dt \quad (2.17)$$

$\tilde{\chi}(t)$ is the vector of characteristic functions $\chi_i$ on the sets of all $t_{i,j}$. $\chi_i(t)$ is one if there exists a $t_{i,j} = t$ and zero otherwise. $\chi$ is a function analogous to $\chi_i$ that is used in some of the graphs:

$$\chi = \lim_{T \to 0} \int_{t-T}^{t} O dt \quad (2.18)$$

The first term on the right hand side of (2.16) expresses an exponential decay with time-constant $\tau$. The second increments $M$ with every presynaptic spike by the weight $w_i$ of that synapse. The third is responsible for the reset at the very moment when a postsynaptic spike occurs: Since $O$ is a sum of delta functions, the latter term is only non-zero at the times $t_k$ when APs occur. Then in any case it will decrement $M$ by $\tilde{M}$, and additionally if there was an input at the same time and the second term $\tilde{\omega}^T \tilde{T}$ was non-zero, this increment will also be eliminated by the term $\tilde{\omega}^T \tilde{\chi} O$.

An example of how $M$ evolves is shown in figure 2.4.

To complete the description of an i&f neuron we need to define when an AP is fired. An i&f neurons fires as its membrane voltage plus the increment
from a synapse that just receives a spike is bigger than some threshold. The i&f neurons here always have firing threshold 1. In other words the threshold is the unit by which the weights and the membrane voltage are measured. We can describe the times $t_k$ of the i&f neuron firing as

$$t_k = \min_{t > t_{k-1}} \tilde{M} + \tilde{w}^T \tilde{X} \geq 1$$

(2.19)

So the time $t_k$ of an AP is the time $t_{i,j}$ of the next input that follows the last AP at time $t_{k-1}$ that would increment the membrane voltage $\tilde{M}$ above the threshold. Note that in this definition of an i&f neuron the membrane never actually does cross the threshold: if a synaptic input would push it over threshold, an AP is emitted and the membrane is reset instantly.

**Theorem 2** The average $\dot{O}$ over the interval between two APs $[t_k, t_j]$ of the output $O$ of an i&f neuron with parameter $\tau = \infty$ is a value slightly smaller than the average of the scalar product $\tilde{w}^T \tilde{I}$ of its weight-vector $\tilde{w}$ and of its input-vector $\tilde{I}$, if the weights $\tilde{w}$ do not change significantly in the interval between two subsequent APs.

$$\dot{O} \approx \tilde{w}^T \tilde{I}$$

(2.20)

more precisely

$$\dot{O} = \tilde{w}^T \tilde{I} \frac{1}{1 + \hat{\epsilon}}$$

(2.21)

where $\hat{\epsilon}$ is the average of the ‘overshoots’ of the membrane over threshold.

We must talk of averages again, since the input $\tilde{I}$ and the output $O$ are sums of delta functions. Only if we integrate over them do they assume values different from zero or infinity.

Let us prove the theorem for the average over the interval between two subsequent APs. From there the theorem extends to the interval between any APs, since the average over the interval between any two APs can be computed as a weighted sum of the averages over subsequent APs.
2.3. INTEGRATE AND FIRE NEURONS

So we average over the interval \((t_{k-1}, t_k]\) not including the starting point (Note that to be correct in the following equations, one would need to replace \(t_{k-1}\) by \(\tilde{t}_{k-1}\) and take the limit for \(\tilde{t}_{k-1} \to t_{k-1}, \tilde{t}_{k-1} > t_{k-1}\) of all expressions. This is not done so that the equations are easier to read). Then as there is only one AP in the interval \((t_{k-1}, t_k]\) \(\hat{O}\) will be

\[
\hat{O} = \frac{1}{t_k - t_{k-1}} \int_{t_{k-1}}^{t_k} O dt' = \frac{1}{t_k - t_{k-1}}
\]

(2.22)

And the average of the scalar product is

\[
\frac{\hat{\vec{a}}^T \vec{I'}}{t_k - t_{k-1}} = \frac{1}{t_k - t_{k-1}} \int_{t_{k-1}}^{t_k} \frac{\hat{\vec{a}}^T \vec{I'} dt'}{t_k - t_{k-1}} = \frac{1}{t_k - t_{k-1}} (\hat{M}(t_k) + \hat{\vec{w}}^T \hat{\vec{\chi}}(t_k))
\]

(2.23)

(2.23) follows since we assumed that \(\tau\) was infinite. So the membrane potential \(M\) between APs is just the integral over \(\hat{\vec{w}}^T \vec{I'}\) (see (2.16)), which is simply the number of inputs times their synaptic weight. At time \(t_k\) this is however not true. We need to correct \(\hat{M}\) by the last contribution of the synapse that triggers the AP, i.e. by \(\hat{\vec{w}}^T \hat{\vec{\chi}}(t_k)\). Considering (2.19): at times \(t_k\) (2.23) is bigger than (2.22) by the overshoot \(\frac{\epsilon}{t_k - t_{k-1}}\) over the threshold, which is smaller than the last synaptic contribution. Therefore, after averaging over at least one AP interval, the scalar product of the weight-vector and the input-vector approximates the average of the output, and this approximation will always be slightly too large. (q.e.d.)

**Theorem 3** If \(\hat{\vec{w}}\) is the average weight-vector (averaged during interval \((t_k, t_j]\) between two APs) of a spiking neuron (processing and emitting spikes) that changes its weight according to the Riccati rule for spiking neurons (SRR1) (2.14), and the changes during the interval of two subsequent APs are negligible, then the attractor \(\hat{\vec{w}}^*\) has the same direction as the average input-vector \(\hat{\vec{I}}\) (also averaged over the time-period \((t_k, t_j]\) (2.24).

\[
\hat{\vec{w}}^* = \frac{||\hat{\vec{I}}||}{||\hat{\vec{w}}^*||} \hat{\vec{I}}
\]

(2.24)
If the neuron is an i\&f neuron with parameter \( \tau = \infty \), then the length \( \| \hat{w}^* \| \) of \( \hat{w}^* \) is slightly bigger than \( \sqrt{\frac{\alpha}{\beta}} \) (2.25).

\[
\| \hat{w}^* \| \approx \sqrt{\frac{\alpha}{\beta}} \\
\| \hat{w}^* \| \geq \sqrt{\frac{\alpha}{\beta}} \tag{2.25}
\]

Again we will only prove the theorem for the average over the interval between two subsequent APs. The proof extends to the averaging-interval between any two APs because the average can be computed as the weighted sum of the averages of subsequent APs.

The assumption, that the weights do not change significantly and can be treated as constant during interval \( (t_{k-1}, t_k] \), is necessary because then we are allowed to claim \( \beta \hat{w} O = \beta \hat{w} \hat{O} \) and \( \hat{w}^T \hat{I} = \hat{w}^T \hat{I} \). So we simply take (2.20) and repeat the proof that led to (2.3) with that inequality to prove the statement about the length of \( \hat{w}^* \).

The direction of the fixed point being the same as the one of the input-vector can be proven by taking the average of the dynamics (2.14) in the time interval \( (t_{k-1}, t_k] \). Then in the same way as for (2.2) we can compute the fixed point of the average weight-vector by substituting \( O \) with \( \hat{O} \) and \( I \) with \( \hat{I} \). (q.e.d.)

A more precise normalization can be achieved if \( O \) is replaced in (2.14) with the expression in (2.23) that is on average equal (and not just close) to the scalar product \( \hat{w}^T \hat{I} \) of the weight and the input-vector. We will call that expression \( \hat{O} \) and define it as

\[
\hat{O} = (\hat{M} + \hat{w}^T \hat{X})O \tag{2.26}
\]

\( \hat{O} \) is a ‘sample’ of \( \hat{M} \) at times \( t_k \) plus the synaptic contribution that pushed \( M \) over the firing threshold, multiplied by a delta function \( \delta(t - t_k) \). In other words it is the reset term of the membrane dynamics (2.16).

The resulting second variant of the Riccati rule for spiking neurons will be called SRR2 and it is now defined by:

\[
\frac{d}{dt} \hat{w} = \alpha \hat{I} - \beta \hat{w} \hat{O} \tag{2.27}
\]
2.3. INTEGRATE AND FIRE NEURONS

Theorem 4 If \( \hat{\mathbf{w}} \) is the average weight-vector (averaged during interval \( (t_k, t_j] \) between two APs) of a spiking neuron (processing and emitting spikes) that changes its weight according to the second variant of the Riccati rule for spiking neurons (SRR2) (2.27) and the changes during the averaging interval are small, then the attractor \( \hat{\mathbf{w}}^* \) has the same direction as the average input-vector \( \hat{\mathbf{I}} \) (also averaged over the time-period \( (t_k, t_j] \) ) (2.28). If the neuron is an iEF neuron with parameter \( \tau = \infty \), then the length \( ||\hat{\mathbf{w}}^*|| \) of \( \hat{\mathbf{w}}^* \) is \( \sqrt{\frac{\alpha}{\beta}} \) (2.29).

\[
\hat{\mathbf{w}}^* = \frac{||\hat{\mathbf{I}}||}{||\hat{\mathbf{w}}^*||} \hat{\mathbf{I}}
\]

The proof is exactly the same as for theorem 1, substituting \( \hat{O} \) by \( \hat{\mathbf{w}} \) with \( \hat{\mathbf{w}} \) and \( \frac{d\hat{\mathbf{w}}}{dt} \) by its average \( \frac{d\hat{\mathbf{w}}}{dt} \). \( \hat{O} \) is the scalar product \( \mathbf{w}^T \hat{\mathbf{I}} \) (see 2.26 and 2.23). The assumption that the changes of the weights are small permits the approximation \( \mathbf{w}^T \hat{\mathbf{I}} \approx \mathbf{w}^T \hat{\mathbf{I}} \).

This new rule causes the weights to converge so that the average weight-vector length is precisely the predicted value. This property is illustrated by a simple simulation shown in figure 2.3.

In biological neurons, however, it is not very plausible that one could know about this value \( \hat{O} \) from signals observable at the synapse. But recent observations [68] show that the action potential in a biological neuron is actively propagated back through the dendrites to the synapses. So the signal \( O \) can be observed at the synapses. In that sense the approximation that is the rule SRR1 (2.14) is closer to what could happening in biology than rule SRR2 (2.27). In natural brain cells the single synaptic weights are rather small, roughly around 3% of the difference between resting potential and firing threshold. For such small weights the functions \( O \) and \( \hat{O} \) become more similar to one another.
2.4 Synaptic correlation signals

We now introduce a variable $\tilde{C}$ to replace $\tilde{I}$ in (2.27) such that the scalar product $\tilde{w}^T \tilde{C} = \tilde{O}$, independent of $\tau$ and thus the weight-vector will be normalized also for $\tau < \infty$ (see section 2.5, theorem 5). The resulting learning rule will express Hebbian behaviour (see section 2.6.1) and will have additional interesting properties. It will for example resemble learning behaviour observed in real neurons (section 3.3).

First we define a vector $\tilde{C}$ with elements $c_i$ that reflect some information about recent presynaptic activity:

$$\frac{d}{dt} \tilde{C} = -\frac{1}{\tau} \tilde{C} + \tilde{I} - (\tilde{C} + \tilde{x}) \tilde{O} \quad (2.30)$$

$\tilde{C}$ is derived from $\tilde{C}$ in the same way as $\tilde{w}$ is derived from $\tilde{w}$ as defined in (2.15). The first term of (2.30) results in an exponential decay. The second is an increment of 1 that occurs with every presynaptic spike. So if there was recent presynaptic activity at synapse $i$, then $c_i$ (i'th element of vector $\tilde{C}$) will tend to be high, and if the synapse is quiet, $c_i$ will decay to zero. The third and last term resets $\tilde{C}$ when APs occur. Since $\tilde{O}$ is a sum of delta functions, the latter term is only non-zero at times when APs occur. Then $\tilde{C}$ will be decremented by $\tilde{C}$ and, in addition, if there was an input at the same time, the increment of that input $\tilde{I}$ will also be eliminated right away by the term $\tilde{x} \tilde{O}$.

$\tilde{C}$ multiplied by its weights is the contribution to the membrane potential made by the individual synapses as defined in (2.16). For an example of the evolution of that signal $\tilde{C}$ refer to figure 2.5.

If we assume that the weights do not change in between postsynaptic spikes then the membrane potential is the scalar product of this vector $\tilde{C}$ and the synaptic weights $\tilde{w}$.

$$M = \tilde{w}^T \tilde{C} \quad (2.31)$$

Analogous to $\tilde{O}$ we also define a $\tilde{C}$. $\tilde{C}$ is a 'sample' of $\tilde{C}$ at times $t_k$ plus the increment of the synaptic contribution that pushed $M$ over the firing
threshold, multiplied by a delta function $\delta(t - t_k)$. $\tilde{C}$ is equal to the reset term in the dynamics of $\tilde{C}$ (2.30).

$$\tilde{C} = (\tilde{C} + \chi)O$$  \hspace{1cm} (2.32)

We named $\tilde{C}$ ‘correlation signal’ because it reflects the activity levels of the synapses at times of postsynaptic APs, and therefore it is an indicator for activity correlations. But it does not merely indicate activity correlations but more precisely it reflects causal relationship between inputs to a synapse and the APs.

$\tilde{O}$ is the scalar product of the weight-vector and the ‘correlation signal’ if the weights do not change between APs.

$$\tilde{O} = \tilde{w}^T \tilde{C}$$  \hspace{1cm} (2.33)

In a biological neuron the carrier of the signal $\tilde{C}$ could be for instance the calcium concentration in a synaptic spine. That concentration is increased by presynaptic activity temporarily and it is localized enough to represent the recent presynaptic activity of individual synapses [75].

## 2.5 The modified Riccati rule (MRR)

If we now substitute in the original Riccati rule for spiking neurons (2.14) $\tilde{I}$ with $\tilde{C}$ and $O$ with $\tilde{O}$

$$\frac{d}{dt} \tilde{w} = \alpha \tilde{C} - \beta \tilde{w} \tilde{O}$$  \hspace{1cm} (2.34)

then the result is a learning rule that only updates the weights at times of postsynaptic spikes. The condition for (2.33), that the weight must not change in between APs, is thus satisfied and $\tilde{O}$ is at all times the scalar product of $\tilde{w}$ and $\tilde{C}$. In the same way as for the original Riccati rule, weight-vector normalization in the steady state can be proven (equation (2.3)). And we state

**Theorem 5** If $\tilde{w}$ is the weight-vector of a spiking neuron (processing and emitting spikes) that changes its weight according to the MRR (2.34), then a
Figure 2.5: A snapshot of the simulation variables involved at one MRR synapse ($\tau = 0.8$s, $\alpha = 0.4$, $\beta = 0.2$). The signal names refer to formula (2.34), (2.17) and (2.18). The pre- and postsynaptic spikes determine the events $t_{i,j}$ and $t_k$, where $k$ numbers the postsynaptic spikes, $i$ is the synapse number and $j$ counts occurrences of presynaptic spikes. Whether the weight increases or decreases with a postsynaptic spike depends on the current size of the weight, and on the size of the correlation signal at that moment (see (2.34)).
Figure 2.6: The convergence of the length of the weight-vector using the MRR. The curves show the length of a weight-vector of a simulated neuron with two synapses that each receive Poisson distributed spike inputs with constant average frequency, evolving over time in two different trials. The upper, somewhat more irregular trace, is derived from a trial in which the synapses are changed according to a variant of the MRR, where $O$ was used instead of $\hat{O}$ in (2.34). That variant would be biologically more plausible than the original MRR but does not perfectly stabilize the weight-vector's length at $\sqrt{\frac{\alpha}{\beta}}$. For the second trial the original MRR (2.34) was used and the result is cleaner. The time-constant $\tau$ of the membrane and of the correlation signal leakage was 100 steps. The input of synapse 1 is 0.12 (spike-probability per time-unit) and the input of synapse 2 is 0.2. The parameters $\alpha$ and $\beta$ are both 0.01. The initial weights are 0.6.
fixed point $\vec{w}^*$ has the same direction as vector $\vec{C}$.

$$\vec{w}^* = \frac{||\vec{C}||}{||\vec{w}^*||} \vec{C} \quad (2.35)$$

If the neuron is an i&f neuron, then the length $||\vec{w}^*||$ of $\vec{w}^*$ is $\sqrt{\frac{\alpha}{\beta}}$.

$$||\vec{w}^*|| = \sqrt{\frac{\alpha}{\beta}} \quad (2.36)$$

The proof is exactly analogous to the one for theorem 1. So equation (2.3) is valid and additionally we can adapt (2.2) to (2.35). Which is to say the attractor is actually:

$$\vec{w}^* = \frac{\alpha}{\beta \tilde{O}} \vec{C} \quad (2.37)$$

(q.e.d.)

Note that formula (2.35) is not a closed form for $\vec{w}^*$, since $\vec{C}$ in general is not independent of $\vec{w}^*$. We still know that the length of any attractor is constant (formula (2.36)) but to determine its direction (the same as the direction of $\vec{C}$) is more difficult and not directly given by (2.35). Temporal correlations influence the direction of $\vec{w}^*$, because they influence the direction of $\vec{C}$. It is not even clear that there should be an attractor. In the following sections we will determine the average direction of $\vec{C}$ as the input and output comply with certain conditions, and for the most restricted case even compute the attractor.

As has been mentioned, in biology the signal $\tilde{O}$ can probably not be observed or reconstructed at individual synapses, but $O$ is observable there. As in SRR1 we can also replace $\tilde{O}$ with $O$ in the MRR. The normalization will no longer be perfect. This effect is illustrated in figure 2.6.

## 2.6 ‘Rate behaviour’ of the MRR

The dynamics in the MRR are much more difficult to analyze than in the RR or SRR. For example there is not necessarily only one attractor. Although
the length of all fixed points can be computed (equation (2.36)), the direction (equation (2.35)) is still dependent on the variable $\vec{\mathcal{C}}$. The average of that variable is dependent on the temporal correlations of input and the output spike-trains and those are generally dependent on the firing mechanism and the weights. Making some assumptions about these temporal correlations, the average $\tilde{\mathcal{C}}$ of $\mathcal{C}$ will be computed in the following and some intuition will be given for the general case. We will also show that if the product $\hat{O}_T$ is small the MRR shows 'Hebbian' behaviour in a rate encoding sense provided that there are no temporal correlations.

2.6.1 Independent Poisson-distributed input and output spike-trains

The strong assumption that there are no temporal correlations between the spike-trains what-so-ever makes expressing $\tilde{\mathcal{C}}$ in terms of average frequencies relatively easy. The direction of $\tilde{\mathcal{C}}$ is independent of $\tilde{w}$ and therefore there is a single attractor for the average dynamics for constant input, and we will compute it. We assume that all input spike-trains are independently Poisson distributed, as is the output. So we can actually not use an i&f model, since there would always be temporal correlations, e.g. between the output spikes and the inputs that pushed the membrane voltage over threshold. But we will still consider that the dynamics of the membrane potential is governed by equation (2.16).

In the following we will compute the expected level $E(l)$ to which the contribution of an input spike $I_i$ to the correlation signal $C_i$ will have decayed when an action potential is elicited. For this we first want to know the probability density $f_T(T)$ of an arbitrary presynaptic spike hitting a postsynaptic interval of duration $T$. A randomly chosen input-spike can be anywhere in the observed time with equal probability. A random point in time will occur in a postsynaptic interval of duration $T$ with a probability proportional to the expected fraction of the total time that is occupied by intervals of length $T$. The expected time covered by intervals of duration $T$ is proportional to their length and the probability density of their occurrence $\hat{O}e^{-\hat{O}T}$ (probability density of the length of an interval in a Poisson distributed spike-train). The product of these two must be multiplied additionally by $O$ to obtain a
Figure 2.7: Comparison of the evolution of the weights using the MRR with independent Poisson-distributed spike inputs and independent Poisson-distributed spike output in a spike simulation (solid line) (2.34) and a rate based simulation (dashed line) (2.43). The dotted lines are the predicted weights of the fixed point. Parameters were $\bar{I}=(20\text{Hz}, 40\text{Hz}, 60\text{Hz}, 70\text{Hz})$, $\tau=0.01$, $\alpha = \beta =0.01$, $\hat{\sigma}=50\text{Hz}$, initial weights: $\bar{w}=(0.6, 0.6, 0.6, 0.6)$. Choosing $\alpha$ and $\beta$ smaller would permit the weights from the spike simulation to follow the ones from the rate-simulation more smoothly.
Figure 2.8: Comparison of the evolution of the difference of the weight-vector norm and the predicted norm using the rate behaviour of equation (2.43) (dashed line) and a simulation of the MRR with independent Poisson-distributed spike inputs and a constant independent Poisson-distributed spike output (solid line). The traces are computed from those of figure 2.7. The norm of the weights from the spike-simulation (solid line) saturates at a minimum that is greater than zero. Choosing smaller $\alpha$ and $\beta$ would reduce the saturation minimum.
Next we compute the probability density \( f_{L|T}(l, T) \) (\( L \) for ‘leak’) of the contribution of a presynaptic spike to the correlation signal \( C_i \) having decayed to level \( l \) by the time of the next AP, given that it finds itself in an AP interval of length \( T \). The time \( s \) of the spike can with equal distribution be anywhere in that interval. If we choose the time at the beginning of the interval to be the origin 0, then \( f_s(s) = \frac{1}{T} \) if \( s \in [0, T] \) and 0 otherwise. Thus \( F_s(s) = \frac{s}{T} \) for \( s \in [0, T] \) and \( F_S(s) = 0 \) before that interval and \( F_S(s) = 1 \) afterwards. Because \( l = e^{-\frac{s}{\tau}} \) and the inverse function is \( s = -\tau \ln(l) \), \( F_{L|T} \) can be computed as \( F_{L|T} = -F_s\left(\frac{-\tau \ln(l)}{T}\right) \) for \( l \in [e^{-\frac{T}{\tau}}, 1] \). Finally we can compute \( f_{L|T} \) by differentiating \( F_{L|T} \):

\[
   f_{L|T}(l, T) = \begin{cases} \frac{T}{\tau l} & \text{if } l \in [e^{-\frac{T}{\tau}}, 1] \\ 0 & \text{otherwise} \end{cases} \tag{2.39}
\]

The probability density \( f_L(l) \) of the contribution of a presynaptic spike to the correlation signal having decayed to \( l \) by the time of the next AP is therefore

\[
   f_L(l) = \begin{cases} \int_{-\tau \ln(l)}^{\infty} f_{L|T}(l, T)f_T(T)dT \\ \frac{T}{l} \int_{-\tau \ln(l)}^{\infty} e^{-\hat{\Omega}T}dT \\ \tau \hat{\Omega} l \hat{\Omega}_{\tau-1} \\ 0 \end{cases} \text{ if } l \in [0, 1] \tag{2.40}
\]

So the expected value to which a single presynaptic contribution to the correlation signal has decayed is

\[
   E(L) = \int_0^1 l f_L(l)dl = \tau \hat{\Omega} \int_0^1 t \hat{\Omega}_{\tau} dt = \frac{\hat{\Omega}_{\tau}}{\hat{\Omega}_{\tau} + 1} \tag{2.41}
\]

and the \( \hat{C} \) can thus be expressed as

\[
   \hat{C} = \hat{I} E(L) = \hat{I} \frac{\hat{\Omega}_{\tau}}{\hat{\Omega}_{\tau} + 1} \tag{2.42}
\]
The direction of \( \hat{C} \) is the same as that of \( \hat{I} \) and therefore the fixed point of the dynamics for a constant input \( \hat{I} \) is the same as for the RR (See figure 2.7). Only the speed with which that point is approached is now dependent on the term \( E(L) \) which is monotonous in \( \hat{O} \). Under these conditions and if the product \( \hat{O}_\tau \) is small, then the behaviour of the MRR is Hebbian, since the term \( E(L) \) increases monotonically with \( \hat{O} \), and that means that with changing input patterns \( \hat{I} \) the direction of the weight-vector \( \hat{w} \) follows the input-vector more quickly if that input causes a higher output-activity. To complete the equation for the the dynamics we assume that equation 2.16 that describes the membrane dynamics for an i&f neuron is valid, although the moment of the firing is not given as for i&f neurons by the equation 2.19. Then we can use equations 2.33 in the MRR 2.34 and thus the dynamics (example in figure 2.7) can be expressed as

\[
\frac{d\hat{w}}{dt} = E(L)(\alpha \hat{I} - \beta \hat{w} \hat{w}^T \hat{I})
\] (2.43)

In the limit \( \hat{O}_\tau \to \infty \) equation (2.43) becomes the Riccati rule with the output \( \hat{O} \) being the scalar product \( \hat{w}^T \hat{I} \):

\[
\lim_{\hat{O}_\tau \to \infty} \frac{d\hat{w}}{dt} = \alpha \hat{I} - \beta \hat{w} \hat{w}^T \hat{I}
\] (2.44)

This is actually always true, independent of the presence of temporal correlations in the signals, or the nature of the firing mechanism. That is because if \( \hat{O}_\tau \) is big, the correlation signal \( \hat{C} \) has no time to decay between output spikes. Thus \( \hat{C} \) acts as a reliable counter of input spikes and its average is therefore the same as the average input frequency \( \hat{I} \).

\[
\lim_{\hat{O}_\tau \to \infty} \hat{C} = \hat{I}
\] (2.45)

In the limit \( \hat{O}_\tau \to 0 \) equation (2.43) becomes the well known Oja’s rule (the Riccati rule multiplied by \( \hat{O} \)) as described in [41]:

\[
\lim_{\hat{O}_\tau \to 0} \frac{d\hat{w}}{dt} = (\tau \alpha \hat{I} - \tau \beta \hat{w} \hat{w}^T \hat{I})\hat{O}
\] (2.46)

Oja’s rule (2.46) is Hebbian since the incrementing term contains the product of input activity and output activity.
2.6.2 Independent Poisson-distributed inputs and an i&f neuron

As now the assumption of temporal independence of the neurons output with each input is dropped by using an i&f neuron, computing $\tilde{C}$ gets more complicated. It has been previously mentioned that in the limit $\dot{\Omega} \tau \to 0$ the MRR behaves like the original Riccati rule (equation (2.45)) independent of temporal properties and of the firing mechanism. In the general case however the dynamics now has multiple attractors. The initial weights influence the outcome of a stimulation with a fixed input-vector.

The output $O$ is no longer Poisson distributed, as can be seen in the extreme case of very small synaptic weights (small as compared to the firing threshold) and no decay ($\tau = \infty$) of the membrane voltage $M$. In that case the output will be very regular, since many presynaptic spikes need to sum up to reach the threshold. So the incoming Poisson signal is averaged over a long period. And there is always one input spike that drives the neuron over threshold. That is a clear temporal correlation between that input and the output spike. The strength of that temporal correlation is also dependent on the synapse's weight, since synapses with higher weight cause APs more often. The contribution of the input spike that triggers an AP to the correlation signal has no time to decay. For these two reasons the former derivation of $f_L(l)$ (2.40) is no longer valid. $f_L(l)$ is now individual to each synapse and dependent not only on the input-vector but on the weight-vector too. Because the correlation signal contribution of the input that fires the cell does not decay, the strong synapses get an edge over weaker ones and the weight-vector will not exactly follow the input-vector but will further emphasize the strength of the stronger synapses and the weakness of the weaker synapses (See figure 2.9). This advantage of synapses that already are strong also leads to a hysteretic behaviour and multiple attractors of the dynamics. This effect becomes more pronounced as $\tau$ gets smaller relative to the average AP interval $\frac{1}{\dot{O}}$. In other words as $\tau$ decreases the learning gradually becomes a hysteretic winner take all (WTA) mechanism.

Computing the probability density of the membrane potential $M$

To compute a rate based model in this scenario (Poisson-distributed inputs and an i&f neuron) we need other means of estimating $\tilde{C}$ (for a constant
Figure 2.9: Comparing the evolution of the weights using the MRR with independent Poisson-distributed inputs and an i&f neuron in a spike simulation (solid line) (2.34) and a rate based simulation (dashed line) (using (2.58) to compute the average $\hat{C}$). The dotted lines are the predicted weights of the fixed point. Parameters were: $\hat{I}=(20\text{Hz}, 40\text{Hz}, 60\text{Hz}, 70\text{Hz}), \tau=0.01s, \alpha = \beta =0.001$, initial weights: $w=(0.6, 0.6, 0.6, 0.6)$. The resulting weights are more widely distributed than in figure 2.7, because of the ‘winner take all’ tendency of the synapses.
Figure 2.10: Comparing the evolution of the difference of the weight-vector norm and the predicted norm in the rate model (dashed line) using (2.58), and in a spiking simulation (solid line) of the MRR when the inputs are independent Poisson-distributed spikes and the input/output relationship is defined by the i&f mechanism. The traces are computed from the ones in figure 2.9. The norm of the weights out of the spike-simulation saturates at a minimum greater than zero. Choosing smaller $\alpha$ and $\beta$ would reduce the saturation limit.
average input and given weights). We tried various simplifications of the problem for this but most of them produced results that were too inaccurate by comparison with the spike simulation. However, a numerical computation of the probability density of the membrane-potential $M$ and deriving $\tilde{C}$ from that function was successful.

Probability density functions of membrane potentials in i&f neurons have been estimated before in simplified neuronal models, where the neuronal weights were considered small compared to the threshold and the input spike intervals were considered small relative to the neuron's time constant $\tau$. The current flowing onto the membrane was thus assumed to be composed of a mean current plus Gaussian white noise and so the evolving of the membrane voltage can be modelled as a diffusion process [72, 46, 45]. In that case the membrane's probability density can be computed by analytically solving the Fokker-Planck equation [12], for certain boundary conditions.

For our purpose we chose a numerical approach that is able to deal with the kind of i&f neuron defined in this thesis without simplifying or restricting assumptions. First we arbitrarily assume a probability density $f_M(M)$ for the membrane potential $M$. The function is discretized into $n$ bins ($f_M^i$):

$$f_M^i(M) = f_{M,i} \text{ if } M \in [M_{i-1}, M_i]$$

$$M_i = \frac{i}{n}$$  \hspace{1cm} (2.47)

If the time is discretized too, we can ask: if the membrane potential was distributed this way, what would be the expected distribution in the next time-step? Since we know the probabilities of state transitions $P_{i,j}$ from one membrane potential bin ($j$) to another ($i$) we can actually compute this: the probabilities of the state transitions will be written into a matrix $P$. Multiplying this matrix with the assumed probability density $f_M$ we get a probability density transition for the next time step. We repeat this until the probability density converges to its fixed point.

So what are those transition probabilities $P_{i,j}$? We handle two causes of state transitions sequentially: First the decay with time (matrix $P^{\text{decay}}$) and then the transitions due to synaptic input (matrix $P^{\text{synapse}}$).

$$P = P^{\text{synapse}} P^{\text{decay}}$$  \hspace{1cm} (2.48)
In one time-step $T$ the membrane potential $M$ will have decayed to $Me^{-\frac{T}{\tau}}$. The contents of bin $j$ will end up in the interval $[M_{j-1}e^{-\frac{T}{\tau}}, M_j e^{-\frac{T}{\tau}}]$. Or, put the other way round, the contents of bin $i$ originate from the bins within $[M_{i-1}e^{\frac{T}{\tau}}, M_i e^{\frac{T}{\tau}}]$

$$P_{i,j}^{\text{decay}} = \begin{cases} 
1 & \text{if } [M_{j-1}, M_j] \in [M_{i-1}e^{\frac{T}{\tau}}, M_i e^{\frac{T}{\tau}}] \\
(M_j - M_{i-1}e^{\frac{T}{\tau}}) & \text{if } M_{i-1}e^{\frac{T}{\tau}} \in [M_{j-1}, M_j] \text{ and } M_i e^{\frac{T}{\tau}} > M_j \\
n(M_i e^{\frac{T}{\tau}} - M_{j-1}) & \text{if } M_{i-1}e^{\frac{T}{\tau}} < M_{j-1} \text{ and } M_i e^{\frac{T}{\tau}} \in [M_{j-1}, M_j] \\
n(M_i e^{\frac{T}{\tau}} - M_{i-1}e^{\frac{T}{\tau}}) & \text{if } [M_{i-1}e^{\frac{T}{\tau}}, M_i e^{\frac{T}{\tau}}] \in [M_{j-1}, M_j]
\end{cases}$$

(2.49)

In the following illustrative example, as in all other computations of $f_M(M)$ in simulations, the time-step $T$ was chosen such that all the elements in the highest numbered bin were projected to the second highest numbered bin ($\Leftrightarrow T = -\tau \log(1 - \frac{1}{n})$). This way of defining $T$ ensures that no bins are skipped as $M$ decays. If the time-step is longer, severe artifacts of the discretization of the time are observed.

Under this condition, the construction of the matrix can be described in a much simpler way, as might be deduced from the following example.

Example for $\tau = 0.01$, $\bar{w} = (0.25, 0.65)$, $T = 0.001$ s, $n = 10$:

$$\begin{bmatrix}
1.00 & 0.11 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0.89 & 0.22 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0.78 & 0.33 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0.67 & 0.44 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0.56 & 0.56 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0.44 & 0.67 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.33 & 0.78 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.22 & 0.89 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.11 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}$$

The transitions due to synaptic input can be expressed in a matrix too. Since a Poisson process can be modelled with a small probability of spiking
per time-step, the membrane potential will be shifted by the synaptic weight with that probability. If it would exceed the threshold (become greater than one), it is reset to zero.

Again for a better overview we compose that matrix from three components: one containing the increments, one containing the transitions that cause a firing and a reset of the membrane, and one that contains the probabilities of the membrane staying in its current state.

\[
\mathbf{p}_{\text{synapseinc}}^{i,j} = \sum_k \begin{cases}
  p_k(j + w_k \cdot n - (i - 1)) & \text{if } i - 1 \in [j + w_k \cdot n - (i - 1), j + w_k \cdot n] \\
p_k(i - ((j - 1) + w_k \cdot n)) & \text{if } i \in [j + w_k \cdot n - (i - 1), j + w_k \cdot n] \\
0 & \text{otherwise}
\end{cases}
\]  

\(2.50\)

Example for \(\bar{w}=(0.25, 0.65), \bar{p}=(0.0211, 0.0843)(\Leftrightarrow \hat{f}=(20\,\text{Hz}, 80\,\text{Hz}))\), \(n=10\), where \(\bar{p}'\) is the vector of components \(p_k\). \(p_k\) is the spiking probability of synapse \(k\) per timestep:

\[
\begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0.01 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0.01 & 0.01 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0.01 & 0.01 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0.01 & 0.01 & 0 & 0 & 0 & 0 & 0 & 0 \\
0.04 & 0 & 0 & 0.01 & 0.01 & 0 & 0 & 0 & 0 & 0 \\
0.04 & 0.04 & 0 & 0 & 0.01 & 0.01 & 0 & 0 & 0 & 0 \\
0 & 0.04 & 0.04 & 0 & 0 & 0.01 & 0.01 & 0 & 0 & 0 \\
0 & 0 & 0.04 & 0.04 & 0 & 0 & 0.01 & 0.01 & 0 & 0 \\
0 & 0 & 0 & 0.04 & 0.04 & 0 & 0 & 0.01 & 0.01 & 0
\end{bmatrix}
\]

\[
\mathbf{p}_{\text{synapsefire}}^{i,j} = \sum_k \begin{cases}
p_k & \text{if } ((j - 1) + w_k \cdot n) > n \\
p_k((j + w_k \cdot n) - n) & \text{if } n \in [(j - 1) + w_k \cdot n, j + w_k \cdot n] \\
0 & \text{otherwise}
\end{cases}
\]  

\(2.51\)
Example:

\[
\begin{bmatrix}
0 & 0 & 0 & 0.04 & 0.08 & 0.08 & 0.09 & 0.11 & 0.11 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\end{bmatrix}
\]

\[
P_{\text{synapse stay}} = 1 - \sum_{k} p_k
\]  
(2.52)

Example:

\[
\begin{bmatrix}
0.89 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0.89 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0.89 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0.89 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0.89 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0.89 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0.89 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.89 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.89 \\
\end{bmatrix}
\]

\[
P_{\text{synapse}} = P_{\text{synapse inc}} + P_{\text{synapse fire}} + P_{\text{synapse stay}}
\]  
(2.53)

Example:
Now if we start with an arbitrary discretized distribution \( f_M \) we can iteratively compute the expected distribution by iterating

\[
\hat{f}_M = P^{\text{synapse}} P^{\text{decay}} f_M = P f_M
\]  

(2.54)

This is equivalent to computing the eigenvector for eigenvalue 1 of the matrix \( P \) and normalizing it such that its sum is \( n \). The distribution corresponding to the example is shown in figure 2.11. The same distribution but with a higher voltage resolution can be seen in figure 2.12.

Now \( \hat{O} \) can be computed with \( f_M \), the firing probabilities \( p_k \) and the weights \( w_k \) and the probabilities \( P_{\text{fire}|(M,k)} \) of the neuron firing when the membrane voltage is \( M \) and then neuron is receiving an input from synapse \( k \). The expected \( \hat{O} \) is

\[
P_{\text{fire}|(M,k)} = \begin{cases} 
1 & \text{if } M_{i-1} + w_k \geq 1 \\
0 & \text{if } M_i + w_k < 1 \\
(n(M_i + w_k - 1) & \text{if } 1 \in [M_{i-1} + w_k, M_i + w_k]
\end{cases}
\]

(2.55)

\[
\hat{O} = \sum_k p_k \sum_{i=1}^n (M_i + w_k) P_{\text{fire}|(M,k)} \hat{f}_{M,i} \frac{f_{M,i}}{n}
\]

(2.56)

To compute \( \hat{C} \) we assume that the contributions of a synapse \( k \) to the membrane potential are proportional to the synaptic weight \( w_k \) and the probability \( p_k P_{-\text{fire}|k} \) of a presynaptic event occurring which does not cause an
Figure 2.11: The membrane potential probability distribution as computed in the example in the text. $\tau=0.01\text{s}$, $\bar{\nu}=(0.25, 0.65)$, $T=0.0011\text{s}$, $\bar{\varrho}=(0.0211, 0.0843)(\leftrightarrow \hat{f}=(20\text{Hz}, 80\text{Hz}))$, $n=10$
Figure 2.12: The same distribution as in figure 2.11, but computed with a much higher spatial resolution.
AP \((p_k P_{\text{fire}}(k) = p_k(1 - P_{\text{fire}}(k)))\). This is slightly incorrect, since bigger membrane potentials are more likely to have been built up by recent inputs of strong synapses. This results in a slight deviation of the rate based prediction as compared with a spike based simulation of the dynamics (compare especially the lower weights in figure 2.9).

\(P_{\text{fire}}(k)\) can be computed as

\[
P_{\text{fire}}(k) = \sum_{i=1}^{n} P_{\text{fire}}(M, k) (M_i, k) \frac{f_{M,i}}{n}
\]

and \(\hat{C}_k\) is

\[
\hat{C}_k = \sum_{i=1}^{n} \frac{f_{M,i}}{n} \left( p_k P_{\text{fire}}(M, k) (M_i, k) + P_{\text{fire}}(M) \frac{1}{w_k} M_i \frac{p_k P_{\text{fire}}(k) w_k}{\sum_k p_k P_{\text{fire}}(k) w_k} \right)
\]

Knowing \(\hat{C}\) and \(\hat{O}\) permits one to simulate the average dynamics as depicted in figure 2.9. The attractor of the dynamics can be computed in a recursive way using equation (2.37)

### 2.6.3 Temporally correlated inputs and an i&f neuron

The situation is even more complex, when the input spike-trains are temporally correlated. The 'winner' can now be a group of synapses that tend to get input-spikes together. Coincident inputs are thus rewarded (figure 2.13).

### 2.7 Single neuron properties

The MRR and similar spike based learning rules in general, in combination with i&f neurons, prefer synapses that receive spike-trains with partially coincident spikes (figure 2.13). This property can be exploited, for instance, to select synapses that are triggered by correlated events. The neuron associates these correlated events (figure 2.14 A). Normalizing the weight-vector at length one ensures that, after learning, all of the associated synapses must be stimulated together in order to trigger an AP. So after a while the neuron
Figure 2.13: Stimulation pattern and evolution of the weights in a computer simulation. A i&f neuron is equipped with four MRR synapses that all receive Poisson distributed spike-trains of the same average frequency. But two of these spike-trains are actually the superposition of two Poisson processes one of which is shared by both of them. So forty percent of all their spike input is simultaneous (Top two traces, synapses one and two). The MRR is capable of preferring those two, although there is no difference among the four synapses in terms of frequency. The weights of synapse one and two gain an advantage in the competition for synaptic weight (lower graph).
Figure 2.14: Examples for delay line schemes that let a MRR neuron learn temporal correlations in its inputs. Three neurons are shown that receive delayed spike-events from sources 1, 2 and 3 at synapses a, b and c. The delay is proportional to the drawn axon length. A MRR neuron prefers synapses that tend to be stimulated simultaneously (figure 2.13) and since it normalizes the weight-vector’s length, that length can be chosen such that after learning all of the preferred synapses must be stimulated together to produce an AP. So in A the neuron would select for example the synapses a and b if the sources 1 and 2 tend to be active simultaneously. In B the neuron may select the same two synapses a and b if 2 tends to be active a certain time before 1 such that the difference in the transmission delay match that time difference. In C only one source of spike events is present. The first two synapses might become strong if source 1 often fires pairs of spikes that are exactly so far apart that the first spike reaches synapse b at the same time as the second spike reaches synapse a.
will only react to the coincident stimulation of these synapses, independent of their number. By including different delays in the transmissions of these events, neurons can learn to associate events that happen in sequence (figure 2.14 B). If we just consider one source of events that reaches a neuron via different delay-lines, that neuron can become responsive to a particular temporal pattern of events from that source. A particular pattern would result in the simultaneous arrival of different spikes at the synapses, the differences in delay at each synapse matching the pattern specific inter-spike intervals (figure 2.14 C).

2.8 Network properties

2.8.1 Network stability during learning

Stabilizing network activity in a recurrent neural network is not trivial. If the connection weights are allowed to be changed by learning the problem is even more difficult. One way of putting an upper bound on network activity is to guarantee that activity cannot be self sustaining. For example when making sure that the loop gain for every neuron is smaller than 1, network activity will decrease when not driven from an external source. Weight-vector normalization can be used to achieve this. The maximal output frequency for a given input-vector length $\|\hat{I}\|$ and a given weight-vector length $\|\hat{w}\|$, when the output is the scalar product (2.6) and weights and inputs are positive, is reached when the two vectors point in the same direction. That makes the output $\hat{O}$ at most equal to that value:

$$\hat{O} \leq \|\hat{w}\| \|\hat{I}\|$$ (2.59)

A weight-vector normalizing learning rule, like MRR, can put a limit to the gain with respect to the input-vector length of a single neuron. Nevertheless, in a recurrent network consisting of neurons with this gain $\|\hat{w}\|$ smaller than one, the loop gain can still be larger than one and the network activity can increase. To illustrate this, let us consider the geometric mean $\|\hat{O}\|$ of all neurons' activities $\hat{o}_i$ in a network of $n$ neurons as a measure for the network activity. Assume a fully connected network in a time-discrete model where the vector of all outputs is at the same time the input for all neurons $\hat{I}(t) = \hat{O}(t - 1)$. 
The loop gain of that network's activity $\| \hat{I}(t) \| = \| \hat{O}(t - 1) \|$ is defined in (2.60).

$$\frac{\| \hat{O}(t) \|}{\| \hat{I}(t) \|} \leq \sqrt{n} \| \vec{w} \|$$  \hspace{1cm} (2.60)

Therefore if we normalize the weight-vector according to (2.61) we can suppress self sustained activity by having the loop gain of the network activity smaller than one.

$$\| \vec{w} \| < \frac{1}{\sqrt{n}}$$  \hspace{1cm} (2.61)

The maximal loop gain of $\sqrt{n} \| \vec{w} \|$ is only reached if all of the input-vectors point in the same direction as the weight-vectors. For this to be the case in the fully connected network, where all input-vectors are the same, also all the weight-vectors must be the same. Therefore all activity levels which form the input-vector would be equal and so all weights would be equal too. So we can state:

**Theorem 6** In a fully connected network of excitatory neurons (the outputs of which are the scalar product of their input-vectors and their weight-vectors) in which the weight-vector length for every neuron is constant, the loop gain of the net-activity is maximal when all weights are the same.

In more desirable states of the network, the loop gain will decrease. That makes recurrent computations increasingly more difficult. So this method of keeping a network from self-sustaining and increasing activity is not always convenient. When the network is computing something sensible, not all the weights will be the same. The recurrent loop gain will be considerably smaller than one and therefore activity is initiated by external input almost exclusively. Recurrent computations are close to impossible. A bigger loop gain with asymmetric weight distributions is desirable. If the restriction on the weights is relaxed, another possibility of keeping the network from self-sustaining activity is to restrict the length of the input-vector. Several mechanisms are applicable here, such as refractory periods and global inhibition.

**Theorem 7** In a fully connected network of RR-neurons (sharing the parameters $\alpha$ and $\beta$, and the outputs of which are the scalar product of their weight-vector and their input-vector), which is capable of self sustained activity and
does not get external input, the only fixed point occurs if all weights of non silent neurons are identical.

Proof: In a fully connected network all neurons share the same input-vector. RR-neurons do not change their weights any further when the input-vector and the weight-vector point in the same direction, and when the weight-vector length is $\sqrt{\frac{\alpha}{\beta}}$. Thus if the output of a RR-neuron is the scalar product of the input and weight-vector, all activity levels are the same. Therefore the input-vector is uniform and therefore all weights are uniform. (q.e.d.)

For this reason one should be very careful not to let a RR, SRR or MRR fully connected network become self-sustaining at high frequencies. At high frequencies ($\dot{\tau}$ is big) (see (2.44)) they all behave like an RR network and will level out the internal weights. In particular, while there is no external input, the synapses receiving external input will decrease their weights.

2.8.2 A coding scheme for spiking networks

Although the neuroscience community by now (almost) agrees that temporal codes are an important component of neural processing, the concrete nature of these temporal codes is as yet not understood. Artificial sensory motor systems that encode information in spike-patterns have not yet been built. As a first step in the direction of constructing such systems, we propose a concrete encoding scheme, which will be established by spike based learning, given a certain network structure and sensory input [26].

We will exploit the property described in section 2.7 to establish a particular code in a network of MRR neurons. Spikes represent some abstract event. Coming from a sensory neuron they represent a system-external event, coming from a non-sensory neuron they represent a system internal event. Such spike-events also form the input of other MRR neurons that, as described in section 2.7, become selective to inputs that tend to be active together, therefore binding simultaneous events into a new event. Including temporal delays in the transmissions of the spike-events (by axonal delays and multi-synaptic transmissions) enables the binding of sequences and temporal patterns of events. For example synfire-chain like structures [1] may form.

So leaving the learning issue aside at first, what does the encoding look like? How are features and objects represented in the network activity and how are whole sequences and causalities of observations stored?
Figure 2.15: Event group coding for object/feature binding. When an object appears in the receptive field of some receptor it will trigger a spike response (These are photo-receptors in this example but the same principle applies for other receptor types.). Simultaneous activity of receptors that are aligned indicate the presence of a bar. This can be detected by a neuron whose receptive field covers these aligned photo-receptors. Applying this same principle repeatedly, neurons will represent objects of increasing complexity. Note here that being active simultaneously refers to single spike activity within a sliding time window of approximately the duration of one neuronal membrane timeconstant.

Feature and object binding

A bar moving across a retinal visual field will simultaneously trigger a group of optical sensors. Bar sensitive neurons that are tuned to that group of sensors will be stimulated (figure 2.17 A), forming groups themselves. Other neurons can then be sensitive to particular groups of these 'bar-neurons', and represent an object outline, as depicted in figure 2.15. In general feature binding can be coded in this way. These scheme resembles a simplified version of a neocognitron [21, 22], a hierarchical pattern recognition system. A major difference is again the use of spike signals in the model proposed here: simultaneity is
Figure 2.16: An illustration of the predictive ability of synfire chain like models. The communicating elements in this graph should be thought of as groups of neurons that are not necessarily element-wise distinct. The spread of spike activity symbolized by the arrows follows a hypothetical sequence of sensory percepts. It might decay away if not confirmed by sensory input on the way and another hypothesis might be favoured, or it might continue without external confirmation. It might trigger appropriate motor actions along the way.

not given by time discretization in our model but simultaneity is a gradual property defined by the membrane's and the correlation signals time constant. Possibilities of merging the two models might be worth investigating, but this has not been attempted for this thesis.

**Binding temporal sequences**

Temporal correlations between events can be represented in a synfire chain like manner. Some group of neurons might respond to the first bar sensitive group, representing the fact that a bar has just passed. Some of those might not fire without additional input from optical sensors that are slightly removed from the original bar-position and so become direction selective (figure 2.17
B). So more generally some group of neurons representing an observed event might trigger a whole (synfire) chain of other events, some needing sensory confirmation just representing an expectation of what happens next, some performing a prediction without sensory confirmation. Figure 2.16 illustrates this.

2.8.3 Learning a temporal code

This coding scheme can be expressed when exposing a neural net via sensors to an environment, when using the MRR. The MRR, or a similar spike based learning rule, is able to learn such a coding in a network with some jitter in the transmission delays between neurons. Neurons will become sensitive to events they experience repeatedly. Sequences that present themselves to the net will be remembered. This can be demonstrated in a simple example.

A simple example: direction and orientation sensitivity

Although we do not (yet) have a strong claim to biological plausibility, we tried to use physiologically realistic parameters when possible in the following simulations.

We use leaky integrate-and-fire neurons with a firing threshold at 1. We chose a short membrane leakage time constant of 1.6ms. Therefore our neurons act like coincidence detectors. This can be justified for average cortical neurons if one assumes a constant background activity that acts to increase the neurons' membrane voltage baseline above their reversal potentials. Then the time constant to bring a neuron back to that level is much reduced. Modelling of in vivo conditions in general show shorter effective time constants than in vitro experiments would suggest [7, 42].

A two layer network architecture with 9 neurons per layer is used. Four different bar stimuli are moved in a random sequence past the input layer. Neurons in that layer react to the passing of the bar by a single spike release. The bars are parallel to the diagonals of the square formed by the nine input neurons and are moved orthogonally to their orientation. The bars are presented during 50 seconds of simulation time in 0.2 second intervals. The speed of the bars was such that the transit time from one line of neurons to the next matched the average axonal transmission delay. This maximizes
Figure 2.17: An example of an orientation and a direction selective cell in a simple model. The grey levels of the cells indicate how recently they have been active. The darkest cells are spiking now. The lighter a cell is, the longer it has not spiked. The activations in the first (left) layer are caused by a passing bar-stimulus. (A) depicts an orientation selective cell in the second layer, marked with an ‘O’. The strongest connections are feed-forward connections from aligned input cells. In (B) a direction selective cell is shown, marked with a ‘D’. It receives inputs from aligned cells from the input layer too, but additionally from an orientation (or direction) selective cell that was active approximately one transmission delay earlier. This is a simple example of binding events that appear in sequence. Note that in order for this model to work, the connection weights must be such that input from all connections are necessary to trigger an action potential, regardless of their number. This can be ensured by normalizing the weight-vector to one. So in our particular example for the orientation selective cell in (A) only two inputs are needed, whereas three simultaneous inputs trigger the direction selective cell in (B). A way to make this possible is to learn the weights with the MRR, which normalizes the weight-vector.
the probability of obtaining direction selective neurons. If the speed is reduced, the numbers of direction selective cells will gradually decrease (figure 2.18 C). The input layer is fully connected to the next layer via learning feed-forward connections. In this next layer all neurons are connected to each other. Connection delays were randomized in an interval of 10ms±3ms in the first experiment and 3ms±3ms in a second simulation. The second setting is closer to biology (The delay of the earliest polarisation in cortex after thalamic stimulation has been estimated to be between 1 and 5ms [18].). Initial weights were 0.45. An additional inhibitory neuron received inputs from all neurons in the second layer (weight = 0.45), and reduced all the neurons' membrane voltages by a fixed amount of -10 when active. This prevented the network from self sustained firing at the beginning of the experiment, when the learning had not yet normalized the weight-vector.

A neuron in the second layer becomes tuned to a stimulation pattern that results in simultaneous arrival of presynaptic spikes at several of its synapses. Note that in contrast to time discrete Hebbian learning, here the term simultaneous is fuzzy and not biased by borders between time-slots. A neuron's choice of a stimulus depends on the set of stimuli and on the random offsets in the transmission delays from both layers. The preferences of a cell may therefore change when the preferences of others change, and neurons tend to choose similar stimuli, e.g. the example of a direction selective cell in figure 2.17 B is dependent on the existence of a cell that is selective to the same orientation. Still the coupling was not so strong as to always prevent the emergence of different orientation preferences in one run. In a bigger network with more local instead of full intra-layer connections several preference regions might develop, such as are observed in visual cortex for example.

With the shorter axonal base delay (3ms) signals from subsequent bar positions could arrive simultaneously in the second layer. Therefore, as opposed to the example in figure 2.17 B, cells can become direction selective without another cell being orientation selective. Also the spike density is increased, which affects the coincidence detection property of the neurons and self sustained activity was more probable. More cells remain non-selective in graph 2.18 B than in 2.18 A.

Mismatches between bar travel time and axonal base delay do not immediately destroy the network's ability to produce direction selective cells (figure 2.18 C, solid line). Even when the bar speed is reduced such that a direct input to the second layer must be faster than a two synaptic one from the
Figure 2.18: A&B: The numbers of members of selectivity-classes over 60 simulations. The first column shows the number of cells that were non-selective (N) because they responded to more than two stimuli, the second bar represents the cells that did not respond to any of the bars (S for silent). The third column shows the number of cells that were sensitive to orientation only (O; responding to two parallel stimuli) and the third is the number of direction selective neurons (D; responding to only one or, if the cell is direction sensitive without being orientation selective, to two orthogonal stimuli). A is the outcome of the simulation with 10ms base axonal delay. For B that base was 3ms. C depicts the influence of the bar transit time between rows of input neurons (x-axis) on the number of cells belonging to the above described classes (y-axis). The dotted line represents the numbers the non-selective cells, the dash-dotted line the numbers of silent ones, the dashed line the numbers of neurons sensitive to orientation and the solid one the numbers of direction selective cells. The sums over 10 experiments are shown. The axonal delay was 10ms

previous bar position, one can still not rule out the possibility of a coincidence of a triple synaptic input from that former bar position with a direct one, though the probability is very low. The number of orientation selective cells is independent of the bar speed (figure 2.18 C, dashed line). The number of silent cells (figure 2.18 C, dash-dotted line) increases with decreasing bar speed up to a point where only the silent cells and the orientation selective cells remain, whereas the number of non-selective cells declines to zero.
2.9 Conclusion

We defined a spike based learning rule (MRR) that is driven by temporal correlations in a neuron's incoming and outgoing spike signals. Causal relationships between input and output are rewarded and the weight-vector is implicitly normalized. The behaviour of the MRR can be described in terms of rates, when the inputs are independently Poisson distributed spike-trains. Then under certain conditions the MRR expresses the behaviour of the original Riccati rule, or Oja's rule, or it performs some hysteretic soft winner take all competition among the synaptic weights. If the input spike-trains do show temporal correlations, the MRR can learn to become sensitive to coincidence, temporal or spatio-temporal patterns in the input spike-trains.

For the computation of the rate defined dynamics in the case of Poisson distributed inputs to an i&f neuron, a numerical method to compute the exact membrane potential probability distribution of an integrate and fire neuron has been introduced. In other publications (e.g. in [12]) analytical solutions of the Fokker-Planck equations have been used to approximate that probability density under some restricting assumptions. For the method described in this work however, no assumptions on the size of the synaptic weights or on the ratio between the membrane time constant and the average AP intervals have to be made.

In this chapter it is shown how weight-vector normalization sets the maximal gain of a neuron with respect to its input-vector and how that can contribute to limit network activity. If the weight-vectors are normalized to a length that would ensure that the network activity cannot explode, recurrent computations become difficult. Therefore other means of limiting network activity such as refractory periods and global inhibition are necessary in addition in a network of MRR neurons.

An encoding for sensory spike events in a network of spiking i&f neurons is proposed that can bind simultaneously experienced features to objects and that can bind sensory events that happen in sequence. That kind of encoding can be learnt by MRR, as is demonstrated in a basic example. Applying the same learning scheme to a more extensive task however would require some elaboration of the network structure. For instance decorrelation of the single neurons responses should be ensured through some inhibiting interaction.

Simulations of more realistic problems or even applying the MRR to real-world problems will surely require larger numbers of artificial neurons. Simu-
lating these on a serial processor the processing speed would decrease at least linearly (neglecting issues of communication) with the number of neurons. Realtime processing would be hard to impossible to achieve. A possible solution to that dilemma is given in the next chapter, by implementing the model in neuromorphic aVLSI.
Event based\(^1\) or even time-continuous simulations of a reasonably sized network of MRR neurons (of for example more than 100 neurons with 20% connectivity) on a serial digital computer would clearly challenge the program optimization skills of a computer scientist, if it is required to run fast enough to interact with a real environment via sensors and effectors. Above a certain network size the task would become impossible. Therefore we developed a number of neuromorphic analog VLSI models that approach the behaviour of the MRR. This analog hardware models consist of synapse and neuron elements that all run independently in parallel and therefore (neglecting problems of communication) the processing speed is independent of the number of elements. The most recent implementation is described and analyzed in detail in this chapter.

3.1 Analog parameter storage on floating gates

A central problem in implementing adaptive algorithms in analog VLSI, is how to represent the adapting analog parameters. For the neuronal aVLSI model

\(^1\)In event based simulations states of simulation variables are only updated with e.g. spike-events
Figure 3.1: Some examples of labels that occur in the electronic circuit schematics in this thesis. Starting from the left the first one names a node so that it can be referenced in the caption. The next three symbols are examples of I/O nodes. They also label the corresponding nodes in a circuit's symbol that may be used in other circuits. The two final labels on the right are global nodes, providing constant parameter voltages (originating off chip or from an analog memory cell). Voltages on a labeled node will be referred to as $V_{\text{labelname}}$, and current flowing in or out of a labeled node as $I_{\text{labelname}}$. Label names that start with '/' name digital active low signals and are the inverse of nodes labeled with the same name without '/'.

\[ /AP \quad \text{psU} \quad \text{weight} \quad \text{soma} \quad \text{ap\_bias} \quad \text{HV} \]
Symbol Schematics

Figure 3.2: This memory core cell consists of circuits proposed in [17] (symbol and schematics). The floating gate's charge (node FG) is decreased by the up and increased by the /down signal (the latter is active low). The signal names refer to the output (Vout) of the memory cell, which is inverted with respect to the FG. The up signal activates the hot electron injection via a special pbase NFET transistor on our 2 μm Orbit process chip (via MOSIS). We use a voltage slightly higher than 5V (6V) for this hot electron injection, simply by powering the memory at 6V (Vhote.Vdd). The /down signal switches the tunneling voltage between 33V and 20V by using a high voltage switch described in figure 3.3. The speed of changing the content of such a memory cell is typically between a few volts per second and a few volts per minute. That speed depends on the operating point of the FG, which is influenced by Vfg.amp.bias, and the applied injection voltage Vhote.Vdd and tunneling voltage VHV.
Figure 3.3: A high voltage switch that is controlled by a normal digital input and switches roughly between 33V and 20V. This element was described by Diorio et al. in [17]. Two of the NFET transistors in the switch are special high voltage transistors: since the drains of these high voltage transistors are n-well instead of n-active, the transistors have a breakdown voltage greater than 40V.
Figure 3.4: Schematics of the FGaRAM as it has been used on several chips to hold analog parameters and biases. It consists of an array of memory cells as described in figure 3.5 and a demultiplexer (see also figure 4.4 for a more detailed description of a demultiplexer). Each memory cell’s content can be changed and accessed while applying its address through signals $A(0..4)$. The digital control signals now affect that cell: $U$ (up) increases and $D$ (down) decreases the voltage, while the read out line accesses the voltage stored in that cell. The cells’ outputs are also connected directly to the locations on the chip, where they provide the bias parameters for the silicon neuron (direct out). Five biases are needed to operate the memory: three high voltages ($HV$ and $HV\_protect$ in figure 3.3), one only slightly higher than $Vdd$ ($hote\_Vdd$ in figure 3.2) and two ‘normal’ ones ($fg\_amp\_bias$ and $hv\_bias$ in figure 3.2). A total of 13 chip pins are needed to generate 32 parameters. More generally one needs $log_2(n) + 8$ pins for $n$ analog voltage biases. The actual layout area of the whole block is $343 \times 1865 \lambda^2$ ($\lambda = 1 \mu m$ in a 2.0 $\mu m$ process, and $\lambda = 0.6 \mu m$ in a 1.2 $\mu m$ process ).
Figure 3.5: Schematics of a single floating gate analog memory cell as it is used in the array of memory cells in figure 3.4. The sel signal from the demultiplexer, enables the cell’s controls (U and D) and places the cell’s value on a common read out line. In the center is the memory core cell, which is described in figure 3.2.
non-volatile analog storage on floating gates (FG) was chosen [38, 48, 17] (see also in section 1.3), because learning by the MRR is a relatively slow process, so the slowness of that particular memory is not a hindrance and because the accurate long term storage ability of FG memory lets the chip not forget what it has once learnt, even if the chip is idle or turned off. Fowler Nordheim tunneling [20] is used to charge a FG and hot electron injection discharges it [17].

We use the same ‘memory core cell’ for parameter storage in a memory array (FGaRAM) and for variable storage, i.e. to store the adapting synaptic weights. The core cell is described in figure 3.2 and consists of a ‘high voltage switch’, a tunneling structure, a hot-electron injection transistor and a two transistor amplifier. It interfaces to other circuitry by its digital up and down signals and by its analog voltage output.

3.1.1 Reliability

We quantified the long term reliability of the floating gate memory cells on a chip containing a silicon neuron (figure 3.25) [31]. A FGaRAM (as shown in figure 3.4) was used to hold the silicon neuron’s parameters. We measured the cells’ storage capacity in two ways: during a 12 hour recording with a sampling interval of 30 minutes; and with two measurements taken six days apart, during which period the chip was switched off. We used a 12 bit ADC from Maxim (MAX180ACPL) for the measurements. From that ADC chip’s specifications, we could expect an accuracy in the order of ± 5mV. We verified the accuracy of our measurement setup by taking 30 measurements of 3V directly generated by a Keithley 230 Programmable Voltage Source (accuracy ± 11mV over 1 year according to specifications), which we used later to provide the four bias voltages required by the floating gate memory. We observed a standard deviation of 3.1mV, which corresponds approximately to a 10 bit resolution. The two recordings from the memory cells showed standard deviations of the same order (2.2mV for the 12 hour recording and 3.4mV for the two measurements 6 days apart) and there was no drift apparent. That means that the error is at most as big as the accuracy of the measurement setup and a resolution of 10 bits over the measurement periods is guaranteed. That stability however is sensitive to the set of the five memory-parameters: most importantly fg_amp_bias should be chosen such that the FG’s operating point is low enough so that no residual hot electron injection through the amplifier’s PFET transistor is possible. (In our experience above threshold
3.2 MRR WITH NON-VOLATILE WEIGHT STORAGE

the voltage difference between channel and gate has more influence on the injection rate than the channel current.) That level is also dependent on the memory's Vdd (hote-Vdd), which sets $V_{DS}$ for the hot electron injection.

The accuracy obtained proved to be quite sufficient for our requirements. The neuron operated with unchanged qualitative behaviour during 2 days of testing. Even after leaving it without power for another 6 days, no adjustment to the parameters was necessary to get it operating again.

3.2 Learning synapses with non-volatile weight storage

Several versions of aVLSI models approaching the MRR were developed and tested during this work [29, 27, 28, 30, 31]. Only the latest will be described here in detail.

3.2.1 Theoretical analysis

The i&f neuron that we use is described in figure 3.13 and fires when the soma voltage $V_{soma}$ falls below the digital switching threshold $\vartheta$ of an inverter (e.g. around 2V). Note that the soma voltage is inverted as compared to a biological neuron: low voltage means a high degree of activation. The synaptic current flows through a transistor the gate voltage of which is $V_{weight}$. The decrease of the soma voltage $V_{soma}$ per input spike is furthermore dependent on the soma capacitance $C_{soma}$ ≈ 200fF, the duration of the presynaptic spike $T_{spike}$ and the constants $V_T$, $\kappa$ and $I_0$ that describe the transistor. That change in somatic voltage per input spike defines the synaptic weight $w$. On the chip described here only $V_{weight}$ is directly observable, so the synaptic current was computed with constants measured from previous chips from the same process. These constants were $I_0 = 3 \cdot 10^{-15}$A and $\frac{\kappa}{V_T} = 23.7$. To ensure that the input spike has a fixed pulse width $T_{spike}$, we filtered the input using a monostable circuit (figure 3.6) that produces a pulse of fixed width in response to the rising edge of the input.

$$\Delta V_{soma} = w = \frac{T_{spike}}{C_{soma}} I_0 e^{\frac{V_T}{V_T} V_{weight}}$$  \hspace{1cm} (3.1)
Figure 3.6: A monostable circuit (symbol and schematics) that responds to a rising edge on its input with a pulse of well defined and adjustable width. The input pulse to the monostable can be shorter or longer than its output pulse. We developed this circuit to capture the incoming presynaptic spike which comes from off chip and the width of which might not be controllable or not even constant. The monostable's output pulse, which does have an adjustable constant width, is then connected to the actual synapse ($V_{ps}$ in figures 3.8 and 3.9). The monostable's core is an active high RS-flipflop (compare with figure 3.7). It is set by the input (node $inedge$) and reset as soon as the capacitances on node $R$ have charged to the digital switching threshold. Note here that if the input pulse is longer than the output, temporarily both $R$ and $S$ are high, in which case $Q$ and therefore the output pulse will be low (according to figure 3.7) as required. The charging of $C1$ starts with $\overline{Q}$ going low and is current-limited by $M2$. As soon as the flipflop is reset, $C1$ is discharged to $Gnd$. The second, smaller capacitance $C2$ is a fail-safe against meta-stable states of the flipflop when it is reset. The flipflop could enter a meta-stable state when $R$ is rising in voltage and therefore $\overline{Q}$ starts to rise too. This turns on $M3$ which starts discharging $R$ before the reset is accomplished. So $\overline{Q}$ would fall again and $R$ would then rise again, and so on. The coupling capacitance gives $R$ some extra momentum (positive feedback) in that critical moment and so avoids that meta-stable, oscillating state.
3.2. MRR WITH NON-VOLATILE WEIGHT STORAGE

Figure 3.7: An active high RS-flipflop (symbol, schematics and truth table). In this work an RS-flipflop is sometimes used in what is normally an ‘illegal’ state, namely with both R and S active (figures 3.6, 3.13). Generally an RS-flipflop’s output is not defined in this state. Here however this RS-flipflop has been implemented using two NOR gates, so we know that this otherwise illegal state results in both outputs being low.

The synaptic weight is contained in a FG-memory core cell as described in figure 3.2. We will call that voltage $V_{weight}$ (equivalent to the voltage $V_{out}$ on node out in figure 3.2). Since the voltage on the floating gate ($V_{FG}$) hardly moves, because the memory cell’s output $V_{weight}$ is vastly amplified ($=V_{out}$ in figure 3.2), we assume the injection and tunneling rates (which actually depend on the voltage $V_{FG}$) to be constant. Therefore the change of $V_{weight}$ is proportional to the duration of the tunneling ($T_{tunneling}$) or injection ($T_{injection}$) with proportionality constants $K_{tunneling/injection}$.

$$\Delta^- V_{weight} = K_{tunneling} T_{tunneling}$$
$$\Delta^+ V_{weight} = K_{injection} T_{injection}$$  \hspace{1cm} (3.2)

According to the MRR, the weights are only changed when the neuron fires. On such occasions two circuits (figures 3.11 and 3.12) emit pulses ($T_{tunneling}$, $T_{injection}$) that turn on tunneling or injection for the duration of the pulse. The durations of the pulses are implemented to model the incrementing and decrementing term in formula 2.34.

The decrementing pulse is produced by a circuit that consists of an AND gate that receives both the AP and its inverse /AP (figure 3.12). It can only go high when both the AP and /AP are high which of course should
Figure 3.8: An analog VLSI neuron circuit including synapses modelling the MRR (symbol and schematics). It shows a neuron with 5 learning excitatory MRR synapses (figure 3.9) and two non-learning synapses (figure 3.10), an excitatory and an inhibitory one. The learning synapses are described in figure 3.9. The box labelled $\Sigma$ is the soma (figure 3.13), integrating synaptic currents and firing an AP after reaching the threshold.
Figure 3.9: The MRR-synapse (symbol and schematics). The two boxes with the plus (in more detail in figure 3.11) and the minus sign (figure 3.12) control pulses that are triggered by an AP and are variable in duration. Those pulses increase and decrease $V_{weight}$ stored in the memory cell (described in figure 3.2).

not be possible. But in this circuit we delay $/AP$ by putting a capacitance $C_{id} \approx 164 \mu F$ on the inverted node and limiting the current to it. The current limit on its charging line ($M_{ld\_rise}$) only prevents glitches on the AP’s falling edge and is biased such that it does not seriously delay the rise of $/AP$. But the current-limit on its discharging line ($M_{ld\_fall}$) effectively sets the width of the output pulse. Since we put $V_{weight}$ on the source and the bias $V_{ld\_fall}$ on the gate, the current $I_{ld}$ is given by

$$I_{ld} = I_{0} e^{\frac{1}{V_{T}}(\nu V_{G} - V_{S})} = I_{0} e^{\frac{1}{V_{T}}(\nu V_{ld\_fall} - V_{weight})}$$  \hspace{1cm} (3.3)

The duration of the pulse $T_{tunneling}$ is therefore the time needed to discharge the capacitance $C_{id}$ from $Vdd = 5V$ to the switching voltage of an inverter ($\vartheta \approx 2V$).

$$T_{tunneling} = \frac{(Vdd - \vartheta)C_{id}}{I_{ld}} = \frac{(Vdd - \vartheta)C_{id}}{I_{0} e^{\frac{1}{V_{T}}(\nu V_{ld\_fall})}} e^{\frac{1}{V_{T}}V_{weight}}$$  \hspace{1cm} (3.4)

Using the definition of $w$ in equation (3.1) and defining $\beta$ as

$$\beta = \frac{1}{V_{T}} \kappa K_{tunneling} \frac{(Vdd - \vartheta)C_{id}C_{soma}}{T_{spike} * I_{0}^{2} e^{\frac{1}{V_{T}}(\nu V_{ld\_fall})}}$$  \hspace{1cm} (3.5)
Figure 3.10: The basic excitatory synapse (A) used in the learning synapses and in the MRR neuron (figure 3.8) which is also equipped with an inhibitory synapse (B). The non learning excitatory synapse consists of two NFET transistors in series, controlled by $V_{weight}$ and the incoming pulse $V_{ps+}$ respectively. The effect of the inhibitory synapse is designed to outlast the presynaptic spike, just as in GABA B synapses in real neurons. This can be used effectively to shunt the soma for a while.
Figure 3.11: The circuit responsible for weight increments. An action potential (nodes AP and /AP), back-propagating from the soma, will trigger a pulse ($V_{\text{learn\_up}}$) that increments the weight. The pulse width ($T_{\text{injection}}$) is dependent on the voltage $V_{\text{corr}}$ across $C_{\text{corr}}$, which by itself is dependent on the recent presynaptic activity (compare equation (3.9)).
Figure 3.12: The weight decrementing circuit. This circuit outputs a pulse \( V_{\text{learn\_down}} \) of a certain width \( T_{\text{tunnel}} \) when the soma produces an AP. The width \( T_{\text{tunnel}} \) of this pulse is exponentially dependent on the momentary voltage \( V_{\text{weight}} \) on the weight node (compare equation (3.4)). A non-ideality of this circuit is the fact that \( V_{\text{weight}} \) is affected capacitively by the switching of the tunneling. It is decreased by a fixed offset of about 1V in the present implementation while the tunneling is switched on. This by itself is not a problem since one can set the parameter \( V_{\text{id\_fall}} \) to account for this offset. The real problem however occurs when the range of \( V_{\text{weight}} \) includes values that are smaller than that offset. Since 0V is the lower bound for \( V_{\text{weight}} \), the pulse width is almost constant for values smaller than that offset. This placed severe restrictions on the range of the parameters for the learning mechanism and made it more difficult to tune.
Figure 3.13: The schematics of the soma. This particular implementation of an i&f neuron was developed and chosen over possible simpler ones, because there was no need to minimize the number of parameters (because of the on chip FGaRAM) and therefore it uses a parameter $V_{ap.bias}$ to set the AP width. The firing mechanism is well decoupled from synaptic currents, which therefore cannot influence the width of the AP and the membrane is completely reset when an AP is released. $M_{soma.leak}$ implements the membrane leakage. The actual soma voltage $V_{soma}$ is stored on a capacitance which is discharged by synaptic currents. As the soma reaches the digital switching threshold $\bar{\vartheta}$ (e.g. $\sim 2.0V$) of the inverter N1 a RS-flipflop is set. $\bar{Q}$ goes low and its inverse $AP$ goes high. $\bar{Q}$ goes through another inverter the rising output $R$ of which is delayed by a capacitance and a current-limiting transistor $ap.bias$. As soon as $R$ reaches the digital threshold, the flipflop gets reset. In the meantime the soma has been reset via $M_{reset}$. Note that the flipflop’s outputs are both low, when $R$ and $S$ are high (compare figure 3.7). So $R$ necessarily remains high until the soma and the flipflop are reset. So even if the soma would not be reset instantaneously the AP will end after the set period. A flaw in this circuit though is the possibility of meta-stable states of the AP-output when trying to reset it. The same measure of capacitive positive feedback as used in the monostable circuit (figure 3.6) will correct that problem in future versions. In addition the signal starting the charging of the reset-capacitance could be the doubly inverted $\bar{AP}$ instead of the direct output $\bar{Q}$. That would further reduce the possibility of the mentioned meta-stable state.
Figure 3.14: A timing diagram of the relevant voltages in a learning synapse. $V_{ps}$ is the presynaptic spike. It enables current flow from the synapse (figure 3.9) to the soma (figure 3.13) where it increases the voltage $V_{soma}$ across $C_{soma}$ and it also charges up $C_{corr}$ (increasing $V_{corr}$, figure 3.11). If $V_{soma}$ reaches the digital switching threshold, an AP ($V_{AP}$) is emitted. At the same time $V_{corr}$ starts to leak away and the learn up pulse ($V_{learn\_up}$) lasts for as long as the voltage $V_{corr}$ remains greater than $V_{corr\_thold}$ (compare figure 3.11) which is set as close to ground as possible. The learn down pulse ($V_{learn\_down}$) also starts with the AP. Its duration is correlated with the current weight (see figure 3.12). During the 'learn up pulse' charge flows from the floating gate, therefore increasing the weight (compare figure 3.2). This is counteracted during the 'learn down pulse' by charge flowing to the floating gate (decreasing the weight). Finally the sum of the two charges determines if the weight is increasing or decreasing.
and assuming that \( \kappa \approx 1 \), then according to (3.2) the decrement per AP \( \Delta^{-} V_{weight} \) becomes:

\[
\Delta^{-} w \approx \frac{dw}{dV_{weight}} \Delta^{-} V_{weight} = \frac{1}{V_T} \kappa w K_{tunneling} T_{tunneling} = \beta w^2
\]  

(3.6)

This term is very similar to the decrementing term in the MRR (2.34), just additionally multiplied by \( w \).

As in the MRR we use a 'correlation signal' \( c \) to determine the increment per AP \( \Delta^{+} V_{weight} \) (figure 3.11). \( c \) is proportional to the voltage \( V_{corr} \) on a capacitance \( C_{corr} \). The leakage current from of \( C_{corr} \) can be set by a bias \( V_{corr, leak} \). \( V_{corr} \) is incremented with a single presynaptic spike by \( \Delta V_{corr} \).

\[
\Delta V_{corr} = \frac{T_{spike} I_{corr, inc}}{C_{corr}}
\]  

(3.7)

Where \( I_{corr, inc} \) is the current flowing through \( I_{corr, inc} \) during the presynaptic spike. In the MRR \( c \) is incremented by one per presynaptic spike. To match the circuit onto the MRR we define the proportionality constant between \( c \) and \( V_{corr} \) by

\[
c = \frac{V_{corr}}{T_{spike} I_{corr, inc}} C_{corr}
\]  

(3.8)

The incrementing part also produces a pulse, the duration (\( T_{injection} \)) of which is proportional to the voltage \( V_{corr} \). An AP enables a comparator which emits the incrementing pulse as long as \( V_{corr} \) is above the parameter \( corr.thold \). At the same moment \( C_{corr} \) starts to discharge with a constant current \( I_{corr,reset} \) through transistor \( M_{corr.reset} \). The pulse ends when \( V_{corr} \) falls below \( V_{corr.thold} \).

\[
T_{injection} = \frac{V_{corr} - V_{corr.thold}}{I_{reset}}
\]  

(3.9)

Choosing \( \alpha \) as

\[
\alpha = \frac{1}{V_T} \kappa K_{injection} \frac{T_{spike} I_{inc}}{C_{corr} I_{reset}}
\]  

(3.10)

and setting \( V_{corr.thold} \) effectively to zero, we get a match with the incrementing term in the theoretical formula (2.34) multiplied by \( w \):

\[
\Delta^{+} w \approx \frac{dw}{dV_{weight}} \Delta^{+} V_{weight} = \frac{1}{V_T} \kappa K_{injection} T_{injection} = w \alpha c
\]  

(3.11)
Summarizing the incrementing and decrementing terms:

\[ \Delta w = \Delta^+w - \Delta^-w = w(\alpha c - \beta w) \]  

If we think of the change in weight at the moment of an AP as instantaneous we can write down the dynamics in a very similar fashion to the MRR (compare equations 2.32 and 2.34):

\[ \frac{dw_i}{dt} = w_i(\alpha c_i - \beta w_i O) \]  

The resulting dynamics has the same attractive fixed point as the MRR. Due to the additional multiplier \( w_i \) the dynamics are 'accelerated' and more turbulent with higher weights. And \( \bar{w} = 0 \) is an additional fixed point, though not an attractor.

### 3.2.2 Empirical analysis

The tests described in this section were conducted on a Orbit 2.0\( \mu \)m bi-CMOS chip (figure 3.15), that was designed to plug into SCX (section 1.4 and chapter 4). It contained 10 MRR neurons with four learning synapses each. It will be referred to as the SCX multi neuron chip (SCX MNC). All Parameters were stored in FGaRAM cells. Spike input and output were conveyed by AER as described in chapter 4. The weights were computed from the measured \( V_{weight} \) as the current through a NFET CMOS transistor with gate voltage \( V_{weight} \) according to equation (3.1). The parameter \( \kappa \) was measured on a previous chip from the same production process. All constant factors were eliminated by dividing the currents \( \bar{w} \) by the average over all stimulation patterns of the norms \( ||\bar{w}|| \).

**Emulating the Riccati rule for spiking neurons (SRR1)**

At first three synapses on one neuron were stimulated with Poisson distributed spike-trains. The time constants of the correlation signal and of the soma were kept large compared to the stimulation frequency in order to approach the behaviour predicted by the Riccati rule: In addition to the weight-vector's norm approaching a constant value, the weight-vector's direction should follow the input-vector.
Figure 3.15: Transceiver 3.0: Layout of an Orbit 2.0\(\mu\)m bi-CMOS chip. It is referred to as the SCX multi neuron chip (SCX MNC). The lower half of the core contains ten MRR neurons, each with four learning synapses.
Figure 3.16: Data of three synapses of one MRR neuron are shown. The synapses were stimulated in 64 trials with all possible combinations of four frequencies (12.5Hz, 25Hz, 50Hz and 100Hz). The inputs were Poisson distributed spike-trains and lasted 5 minutes for each trial. The weights were all reset to \( V_{weight} = 0.7 \) between trials. The three groups of three traces, shifted with their respective baselines are as follows. The topmost values, shown as squares, are the input frequencies. The second group of traces (o’s on a dashed line) is the theoretical weight of that synapse according to the fixed point of the RR (equation (2.4)) for \( \alpha = \beta \). The lowest traces are the synaptic weights measured on the chip and normalized by the average weight-vector length (averaged over the 64 different input patterns). The time constants of the soma-membrane and the correlation signal were chosen to be large such that the neurons behaviour should match the Riccati rule and the measured MRR weights should in average match the theoretical RR weights. Standard deviations from the theoretical Riccati rule values: synapse 1 (lowest trace): 0.1540; synapse 2: 0.1203; synapse 3 (top trace): 0.1962.
Figure 3.17: This graph is derived from the data in figure 3.16. For every trial and synapse the expected weight versus the real weight is plotted. synapse 1 (the lowest in figure 3.16): ‘+’; synapse 2: ‘◊’; synapse 3: ‘□’. The identity line is also shown. Ideally all points should fall on that line.
Figure 3.18: This graph is derived from the data in figure 3.16. It shows the weight-vector norm computed from the trials described in that figure (solid line), which according to theory should constantly be 1. Standard deviation: 0.1570. In particular the weight vector length should be independent of the length of the input vector (dashed line, values are divided by the average over all input patterns). The covariance of the two is only 0.0541, the correlation-coefficient is 0.4972.
The stimulation frequencies, the predicted attractor according to formulas (2.35) and (2.36), and the measured weights are shown in the figure 3.16. The measured weights are compared to the normalized input frequencies in figure 3.17, which according to the RR should match. Standard deviations are 0.1540, 0.1962 and 0.1203 for synapses 1, 2 and 3 respectively.

All the weights are slightly too low in the lower middle range and slightly too high in the upper middle range, which is an indicator of the behaviour complying with the MRR with a long time constant rather than complying perfectly with the RR.

The norm plotted against the input frequency patterns in figure 3.18 is noisy with a standard deviation 0.1570. This noise can be explained by the fact that the dynamics are more prone to deviate from the average dynamics predicted by the RR for large weights because of the additional multiplier \( w_i \) in equation (3.13). Still the weight-vector is bound within a narrow range and can neither grow infinitely nor vanish.

**Emulating the modified Riccati rule for spiking neurons (MRR)**

The same experiment was repeated, but this time the leakage currents from the soma and the correlation signal were turned on (figures 3.19, 3.20 and 3.21). \( \tau \) could not be measured directly from the chip and was estimated to be roughly 1ms to compute the theoretical values.

The relative strength of the synaptic weights match well with the theory but the weight-vector normalization of the measured weights is poor (figure 3.21). The major problem was that the leakage from the soma and the leakage from the correlation signal were dependent on different biases and were not well matched. The membrane voltage was not observable on the chip and therefore the leakage from the membrane could not be determined directly. For the recorded experiments presumably the time constant of the leakage from the soma was longer than the one of the correlation signal. Then if the input-vector length was small the output frequency would be higher than it would have been if the time constant of the membrane had been as big as the one of the correlation signal. Therefore the decrementing term in the learning rule was too big and the resulting weight-vector too short. Therefore, as one can see in figure 3.21, the length of the input-vector and the weight-vector are correlated.
Figure 3.19: Similar data plot as in figure 3.16 but with the soma leak and the correlation signal leak turned on. The theoretical (middle graph) traces are computed with $\tau=0.001$s, $\alpha = \beta$ and initial condition $w = (0.6, 0.6, 0.6)$. As compared to the RR peaks and valleys are more pronounced. The relative strength of the synapses is correct but the weight-vector normalization in this example is poor.
Figure 3.20: This graph is derived from the data in figure 3.19. For every trial and synapse the weight as predicted by the theory (with \( \tau = 0.001 \) and initial weights \( \bar{w}_{in} = (0.6, 0.6, 0.6) \)) versus the weight of the chip are plotted. Synapse 1 (lowest trace in figure 3.19): '+'; synapse 2: '◊'; synapse 3: '□'. Standard deviations of the measured to the theoretical weights are (0.5282, 0.5012, 0.3116) for synapse 1 (lowest trace), 2 and 3 (topmost trace), respectively.
Figure 3.21: The weight-vector norms derived from the trials described in figure 3.19 (solid line) and the length of the input-vector (dashed line, values are divided by the average over all input patterns). Presumably because the time constants for the leakage of the correlation signals were shorter than the membrane time constant there is an undesirable strong covariance (0.3151) between the two. If the time constant of the membrane had been shorter than those of the correlation signals, a negative covariance between weight-vector length and input-vector length would have been expected instead. Only if the time constants were identical, could a better normalization be expected. Standard deviation of the measured norm is 0.8101. The covariance between the input vector length and the weight vector length is 0.3151 and the correlation coefficient 0.5612.
Independent of that effect, the mismatch between synapses are observable: synapse 3 is always weaker than the others for the same subjective stimulation pattern. The leakage transistors are an additional source of mismatch and the efficacies of hot electron injection and especially the efficacy of tunneling changed with use, so that their mismatches got worse over time too (figures 3.23 and 3.24).

The theory also matched the measurement poorly when two or three input frequencies were the same. Since the initial conditions of the theoretical computation were all weights being the same, there was no way to break that symmetry in the computation of the fixed point. On the chip however, the intrinsically strongest synapse broke the symmetry and determined to which attractor the weights converged.

The standard deviations from the prediction with the MRR for the original data are (0.5282, 0.5012, 0.3116). The relative strength of the synapses however was accurate. If the measured weight-vector was explicitly normalized, the standard deviations of the synapses dropped sharply to (0.1410, 0.1405, 0.1454).

The normalization is rather crude. Standard deviation from perfect normalization was 0.8101.

Preference of synapses with temporally correlated inputs

Synapses that received a certain percentage of coincident input spikes got stronger as when they were stimulated with the same frequency but without coincident spikes. Since the intrinsic strength of the synapses did not match up well though, it was not always the case that they would become stronger than all other synapses with the same stimulation frequency. An example of the effect of partially coincident spike stimulation is shown in figure 3.22.

3.2.3 Practical problems of the implementation

Synapse mismatches

The dynamics of individual synapses did not match well. Some clearly tended to be stronger than others. The major sources of mismatches were the tunneling and injection rates of the FGs (figures 3.23 and 3.24). These mismatches,
Figure 3.22: The graph illustrates the effect of partially coincident spikes on the learning outcome. The resulting weights after 5 minutes of stimulation of four synapses of a neuron are shown. All synapses receive Poisson distributed spike-trains with 25Hz average frequency, except for synapse 3, which receives 33Hz input. The other differences in synaptic strength to the left are due to inherent differences in the circuits. Moving to the right in the plot, the percentage of coincident spikes for the second and the fourth synapse was increased. As predicted by the theory of the MRR in chapter 2.7 the neuron starts to favour those synapses more and their strength is increased. The difference in strength between synapses 2 and 4 is due to intrinsic synaptic mismatch.
Figure 3.23: The rate of decrease of the contents of the memory cells' that were holding the synaptic weights when the tunneling was turned on. The rate shown is the inverse of the time it took to move the memory cell voltage from three to two volts. The coefficient of variance (CV) among the cells was 1.17. The most intensively used cells (1-4 and 13-16, weights of neurons 1 and 4) are worn out most and the tunneling is not efficient anymore. The memory parameters were: $V_{HV}=25\text{V}$, $V_{HV\text{-protect}}=21\text{V}$, $V_{HV\text{bias}}=0.68$, $V_{hote\_Vdd}=6.2$, $V_{fg\_amp\_bias}=0.85$. 
Figure 3.24: The rate of increase of the contents of the memory cells that were holding the synaptic weights when the hot electron injection was turned on. The rate shown is the inverse of the time it took to move the memory cell voltage from two to three volts. The coefficient of variance (CV) among the cells was 0.13. The memory parameters were: $V_{HV}=25\,\text{V}$, $V_{HV\_protect}=21\,\text{V}$, $V_{HV\_bias}=0.68$, $V_{hote\_Vdd}=6.2$, $V_{fg\_amp\_bias}=0.85$. 
especially in the tunneling rates, became even worse over time, since the tunneling and injection efficacies are decreased with use due to trapped charges in the oxide [17].

The capacitive coupling of the high voltage node and the synaptic weight

The weight decrementing circuit (figure 3.12) is non ideal. When the learn down pulse switches on the tunneling, the memory cell’s content \( V_{weight} \) drops sharply due to capacitive coupling of the tunneling voltage to the FG. That drop is rather big (up to 1 V). Since the duration of the learn down pulse \( T_{tunnel} \) is dependent on \( V_{weight} \), \( V_{id\_fall} \) has to be adjusted to make up for that drop. A problem occurs when the drop would be larger than the voltage \( V_{weight} \) since then it would always drop to zero and \( T_{tunnel} \) would become independent of the actual weight. To avoid this, the weights had to be kept larger which limits the choice of parameter settings, and makes their tuning difficult.

Noise due to meta-stable states of the integrate and fire mechanism

Learning trials with more than one neuron stimulated failed. Closer investigations revealed that if there was an action potential, the content of all memory cells, independent of the distance from the active cell, experienced a downward shift of about 100 mV. On the latest chip the i&f soma’s (figure 3.13) spike output falls into a meta-stable state such that the action potential remains at the digital switching threshold after firing. This has also been observed sometimes on smaller chips, but never with such consistency. Perhaps a small initial glitch on \( V_{ap\_bias} \) caused by crosstalk of the digital AER periphery makes it easier for the AP to fall into that meta-stable state. Due to the meta-stable state the learn_down pulse also (figure 3.12) moves to a stable non-digital voltage at the switching threshold. On this chip, the memory cells containing the synaptic weights are changeable not only by the learning circuitry but also by off chip intervention. That is why the learn down pulse does not lead directly to the HV-switch. Instead, it passes another digital gate that is powered by \textit{hote}._\textit{Vdd} first (see figure 4.8). These events showed a strong correlation with that downward shift in the memory cells. We concluded that the amount of current drawn by the digital gates with non digital inputs is big enough to cause a level shift in \( V_{dd} \) and \textit{hote}._\textit{Vdd}, the latter
being responsible for the shift in the memory cells, since it powers the two transistor amplifiers that amplify the floating gates' voltages. A voltage drop on \textit{hotc.Vdd} was not observable outside the chip, but a large increase in current could be observed. These shifts in the memory cells and therefore in the learning parameters produced very unstable and irreproducible results. Experiments with single neurons were not severely affected since parameters that were important during an AP could be chosen to compensate for this offset. If there was another cells active though, these shifts could be caused by the AP of the second neuron and affect the parameters at random times also for the first neuron, which greatly disturbed its behaviour.

These metastable states could be avoided in the same way as it has been done in the very similar monostable circuit of figure 3.6, by applying positive capacitive feedback from \(Q\) to \(R\).

3.3 An aVLSI model of biological LTP and LTD

We also combined an earlier version of a learning synapse (figure B.1) with non-volatile storage and an FGaRAM with a more detailed model of a neuron [31], a so called silicon neuron [54]. In this way we could emulate biological long term potentiation (LTP) and long term depression (LTD) (figure 3.26).

3.3.1 The 'silicon neuron'

A silicon neuron is an aVLSI circuit that models a number of variables observed in a biological neuron, i.e different ion currents and conductances, concentrations of elements, ions and molecules. It can be divided into compartments to reflect some spatial locality to these variables, for example as they vary along the length of the dendrites. The silicon neuron used here consists of a single compartment soma with a leakage conductance, the sodium and potassium spike conductance, a high-threshold calcium conductance and a calcium-dependent potassium conductance [64, 63, 54]. The operation of these conductances is described in [63]. Almost all parameters (28) are set by cells in the FGaRAM (with the exception of four that draw more current than our cells can provide without changing their voltages) and define the time and voltage dependence of the ionic conductances.
Figure 3.25: The schematics of the silicon neuron that was used for the AP traces in figure 3.26 A and B (courtesy of Christoph Rasche). The membrane potential is the voltage on $C_m$. The passive leakage is represented by the current $I_{\text{leak}}$. The currents $I_K$ and $I_{Na}$ are responsible for the action potentials. In addition the output rate attenuates with neuron activity, because capacitor $C_C$ is charged and an additional leakage current ($I_{AHP}$) is increased. For more details, refer to publications [64, 63, 54]. For an example of the output, refer to figure 3.26 C.
Figure 3.26: Stimulation patterns that lead to LTP and LTD of our AMPA synapse and their effect on subsequent excitatory post synaptic potentials (EPSPs). A and B show two stimulation patterns that lead to LTP and LTD respectively. The presynaptic spikes are digital pulses delivered to one of the learning synapses (dashed lines), whereas the silicon neuron’s output resembles closely a biological train of action potentials (AP) (solid lines). The postsynaptic APs have been forced by stimulating a non-learning synapse that produced single EPSPs that were large enough in amplitude to make the silicon neuron fire. We ensured that the learning synapses EPSP remained below the soma’s firing threshold, even when it was increasing. Similar to experiments described in [57], we stimulated pre- and post-synaptically with bursts of four spikes with 50ms (100ms in [57]) inter-spike interval. We applied 15 such bursts with 200ms (4s in [57]) inter-burst intervals. The pre- and postsynaptic spikes were shifted by 10ms with respect to each other. When the presynaptic spike preceded the postsynaptic spike (A) the subsequent EPSPs were higher in amplitude (C, dot-dashed line) and if the presynaptic spike followed the postsynaptic one (B), the synaptic efficacy decreased (C, dashed line). Before the experiments the synaptic weight was always brought to the same baseline. The solid line in C is an example of a single EPSP recorded before an experiment.
Additionally, the soma has four learning synaptic conductances. They are simple excitatory synaptic conductances [64] that behave similar to biological AMPA synapses and will be referred to as AMPA synapses in the following. One of the AMPA synapses’ parameters sets the amplitude of its EPSP. We call that parameter synaptic weight. The learning mechanism described in section 3.3.2 and figure B.1 controls these four synaptic weights which are stored on four FG memory cells.

3.3.2 Regulation of synaptic efficacy

The learning synapse is an earlier version than the one described in section 3.2. It is described in figure B.1. The change in synaptic weight modulates the EPSP amplitude of the four AMPA synapses.

Our learning rule shows an approximation of the behavior described in [57, 14]. In the 'in vitro' setup described in [57], stimulation of two interconnected cells with spike bursts of 10Hz and a phase of 10 ms, causes the synapses to the cell which is stimulated later to become stronger and the ones towards the other cell to become weaker. We did not interconnect two silicon neurons but stimulated a single cell in the required manner. The resulting behaviour is very similar to the one in the physiological experiment, causing LTP and LTD respectively. For details refer to figure 3.26.

3.4 Conclusion

A neuromorphic aVLSI chip has been built, that approaches the behaviour of the spike based MRR to a large extent. Unlike any other on-chip analog learning model of similar complexity it uses FG analog non-volatile storage for its biases and the synaptic weights and therefore does not lose its state when the power is interrupted. Others have used FG analog storage for parameters that are changed from off-chip [38, 48, 17, 32]. Only a few have used them for parameters that are changed on-chip [33, 16] and in only one other recent chip design has it been used for analog storage in a kind of learning [16].

The basic FG memory cell of [17] has been successfully extended in this work to a FG analog random access memory (FGaRAM). Floating gate memory arrays that are not accessed by addressing but that are accessed sequentially by a scanner were implemented in other work [32]. The particular imple-
mentation in [32] has the nice additional feature that the capacitive jump of a FG memory cell when the tunneling voltage is switched on is compensated for. In the present work the difficulty with the 'learn down' circuit described in figure 3.12 could have been avoided from the start with this compensation mechanism.

The aVLSI synaptic model is able to replicate essential features of the MRR such as expressing the rate behaviour of the original Riccati rule and normalizing the weight-vector when the timeconstants of the membrane and the correlation signal are kept long compared to the inter-spike intervals of the output. When these time constants are shortened, the chips changes the relative strength of the weights when receiving rate inputs in accordance with the theory. The absolute strengths however are dependent on the time constants of the membrane and the correlation signal being the same. These time constants were hard to tune though and therefore the absolute weight strengths matched the theory poorly and therefore the weight-vector normalization was poor. Membrane potentials and correlation signals on the chip should be made observable such that this tuning can be achieved more easily. The matching of the transistors and the capacitances that define those time constants should be ensured. The theoretical effects of temporal correlations in the inputs could be observed in the aVLSI implementation as well. Synapses that tended to receive spike input together were enhanced.

Mismatches in general, especially mismatches of the tunneling rate, were a problem. Larger tunneling areas might improve that, but will probably not correct the problem entirely. But a certain extent of asymmetry is desirable in certain learning tasks that do require a symmetry breaking factor.

Multi-unit experiments failed because of an unwanted metastable state of the i&f neuron that had an offset effect on all analog storage cells. The neurons' biases are all either important during an AP or in inter-AP intervals. The biases that are needed during an AP can thus just be programmed accordingly, such that the downward shift during an AP brings them to the appropriate value. But in multi-unit experiments APs from all other neurons interfered with biases that were used in inter-AP intervals in an individual neuron and these interferences had unpredictable effects on the neurons' behaviours. Extensions of the i&f circuit, that would prevent the metastable state are proposed in the text in section 3.2.3 and in the caption of figure 3.13.

A predecessor of the aVLSI FG learning synapse was inserted into a more
detailed neuronal model, a so called 'silicon neuron'. Like this we were able to reproduce to a high level of detail the learning behaviour observed in a physiological experiment. Silicon neurons allow detailed real-time models of neural circuits. The properties that can be emulated now include long term potentiation (LTP) and long term depression (LTD) as well.
Chapter 4

Integrating multi neuron chips into ‘silicon cortex’

In the previous chapter 3 a neuromorphic aVLSI model neuron that learns according to a spike based learning rule has been presented. As a next step towards a neuromorphic system that uses temporal encoding we wanted to use that kind of neuron in a network. One problem in implementing networks of spiking neurons is the communication between multi-neuron chips. As mentioned in the introduction AER offers a solution to this problem.

Fortunately another project under development at the Institute of Neuroinformatics implements a general and configurable AER platform. This system is named ‘silicon cortex’ (SCX) and it allows to configure the network connections between chips that use AER for communication and that comply to the SCX protocol. Using a DSP it maps sender addresses to receiver addresses as illustrated in figure 4.1. During the work for this thesis a standard chip design framework was developed that interfaces a number of circuits that receive and/or emit spike events (neuromorphs) and that are placed within a twodimensional grid in the chip’s core to the SCX system. This so called SCX MNC (multi neuron chip) framework handles the correct communication with SCX. Figure 4.2 depicts the services that a pixel within that framework may use and figure 4.3 shows an example of how a number of artificial spiking neurons may be placed within that framework.

A number of circuits that are part of this framework have been developed
or improved for this thesis. For example a new arbiter circuit was introduced that improves the speed and thus the fairness of the on-chip arbitration (figures 4.13, 4.16, 4.15), and new logic blocks that completely decouple the neuromorphs from the communication protocol (figure 4.19). These blocks provide a clean interface to any circuitry (neuromorph) that produces rising edges (e.g. an AP) as event output.

### 4.1 Address-event representation in ‘silicon cortex’

The chips developed more recently during the work described in this thesis contain multiple neurons. In order to connect them off-chip, address-event representation (AER) (figure 1.2) was used. More specifically they were designed to be connected to the ‘silicon cortex’ (SCX), that is under development as part of another project at the Institute of Neuroinformatics [15]. SCX offers a general AER platform that allows the configuration of the connections between neuromorphs on multiple chips (figure 4.1).

Chips that plug into SCX send and receive address-events on two separate busses (figure 4.1). SCX performs the routing between the two with a DSP
Figure 4.2: The elements of a pixel within the SCX MNC framework that may be interfaced to SCX. Each pixel can receive (symbolized by the circle) and send (symbolized by the triangle) pulses. These pulses are translated into (or from) SCX address-events in the frame surrounding the pixel array. In addition each pixel may contain a memory cell or a probe point that can be accessed by SCX. That means that it can be observed on a global line from off chip and, in the case of a memory cell, can be changed by commands from SCX.
Figure 4.3: An example of five neuromorphs placed into the SCX MNC framework. They are spiking neurons with three synapses each. Since a single synapse covers the chip area of two pixels, only the receiving and storage capability of one pixel in two per synapse is used. The storage cell holds the synaptic weight. The rightmost pixels contain the cell's soma that emits the AP and therefore makes use of that pixel's sending ability. Furthermore a probe point is introduced to make the membrane voltage observable. Note that only those lines are connected that are neccessary to provide the services that are in fact used.
so that any desired connectivity can be programmed.

4.2 Receiving address-events from ‘silicon cortex’

Neuromorphic analog chips are not generally clocked. Therefore these chips and the SCX do not operate synchronously. A handshake can be used to synchronize a chip and the SCX for the duration of one transmission and to ensure that the transmission was accomplished. When multiple chips are receiving however, this is not possible, because there are not two partners for the transmission who could do the handshake: several chips might be interested in a particular address-event. Therefore chips are merely informed of the presence of a valid address by a data valid (DAV) pulse. In the present SCX specifications it is stated that this pulse is at least 200ns wide. Chips must be fast enough to read the address during that pulse, which is not problematic for the chips described in this thesis. A really simple implementation of a receiver (figure 4.4) was sufficiently fast. Only a demultiplexer and enabling gates at its output (opened by DAV, see figure 4.6) were necessary (figure 4.4). Internally the 200ns DAV pulse (on node /rec in figure 4.6) was then widened for our synapses by a monostable circuit (figures 3.6 and 4.8). Other types of synapses however might not require this step.

We also equipped our receiver design (figures 4.4, 4.6) with an additional feature. It supplies not only a /rec line, conveying the /dav pulse to the appropriate synapse (pixel), but also a /memsel line that allows an analog memory-cell (figures 4.10 and 4.8) or a probe point (figure 4.11) pixel to be selected. A load signal from SCX that controls an additional demultiplexer element distinguishes between AER inputs and memory accesses (figure 4.6). When accessing an analog memory cell or a probe point the /memsel line goes active and is latched until another memory cell or probe point at a different address is chosen. The selected pixel is now allowed to put the memory cell’s content or the probed voltage on the shared analog output line alogout that can be observed off-chip. The global lines U (up) and D (down) (figure 4.8) that affect memory cell contents will now only affect the selected cell.

Figure 4.8 shows an extended learning MRR synapse, such that it fits into that receiver framework. (Compare with the basic version in figure 3.8.)
4.2. RECEIVING AE FROM SCX

Figure 4.4: A 3 bit AER receiver to receive address-events from SCX. (Symbol on the left comprised of blocks at right.) If neuromorphs are arranged two-dimensionally, two of these circuits are required. The receiver is also equipped to address analog memory cells. Since SCX supplies no handshake but only a data valid signal (/dav), the basic receiver is quite simple: it is just a demultiplexer that is enabled by the chip select signal (/cs), and the output of which is 'gated' by /dav (see also figure 4.6). The demultiplexer consists of a binary tree of stackable one bit demultiplexers (figure 4.5). The load signal differentiates between address-events (load is low when the /dav pulse is applied) and accesses to the analog memory (load is high when the /dav pulse is applied). For address-events the appropriate receive line (/rec) conveys the /dav pulse. For memory accesses (figure 4.10) or probing of an analog voltage (figure 4.11) the selected /memsel line goes high with /dav and is latched (by the /loaddav pulse which is actually ¬(load ∧ dav) and is computed elsewhere) until another memory access is requested. This will select a memory cell, the content of which drives a common analog output line (not shown), that can be accessed off-chip.
A: demultiplexer positive output
Symbol: \( \text{sel} \)

Symbol Schematics

\[ \text{in} \quad \text{out0} \quad \text{out1} \]

\[ \text{in} \quad \text{out0} \quad \text{out1} \]

B: demultiplexer negative output
Symbol: \( \text{sel} \)

Symbol Schematics

\[ \text{in} \quad \text{in} \quad \text{in} \]

\[ \text{in} \quad \text{in} \quad \text{in} \]

Figure 4.5: The two stackable one bit demultiplexer elements that are used in figure (4.4). They are often used, standard circuits. When selected (\( \text{sel} \) or \( \text{sel} \)) the input \( \text{in} \) and its inverse \( \text{in} \) determine if \( \text{out0} \) (and \( \text{out0} \)) or \( \text{out1} \) (and \( \text{out1} \)) will be active. Since the element with positive (active high) output takes an active low select signal as input and vice versa, one can stack these elements (interleaving the two types) to form demultiplexers of arbitrary size.
4.2. RECEIVING AE FROM SCX

Symbol  Schematics

Figure 4.6: The receiver logic block (symbol (upper left) and schematics), a part of the receiver (figure 4.4). It distinguishes address-event input and memory accesses with an additional 1-bit demultiplexer. Address-events are gated with the /dav signal by a NAND gate, and memory accesses are latched (see also figure 4.7) and held until a next memory access is made and another /memsel line is chosen.

Symbol  Schematics

Figure 4.7: A transparent latch that conveys its input to its output while the clk signal is low and preserves its output when clk goes high.
Figure 4.8: The learning synapse as it was extended from figure 3.9 to form part of the two dimensional SCX-AER framework. Monostables were inserted to make longer, well controlled pulses out of the /DAV pulse (which makes /rec.x and /rec.y for the addressed pixel go low) supplied from off-chip by SCX. The memory cell was extended (figure 4.9) such that it can be affected not only by the learning mechanism via lu and /ld but also by the off-chip signals U and D and so that it can be read via a common line alogout from-off chip. So alogout, U and D are global signals affected by and affecting the memory cell that is currently selected by /memsel.x and /memsel.y.
4.3 Sending arbitered address-events to ‘silicon cortex’

The circuitry required on the sender side imposes greater demands on the chip designer. Since there are multiple senders sharing one bus, there is the question of collision handling, not just between neuromorphs on a single chip, but also among different chips on the SCX-board. The latter is handled by separate dedicated bus arbiter logic that grants bus access to only one chip at a time. Since this is now effectively a many-to-one scenario, a slightly adapted handshake is performed that includes a request for bus access: every chip has its own /bus_req and /bus_ack line (figure 4.4). With the request it does not, as it would in a standard 4-phase handshake, indicate that the data is ready on the bus. It merely indicates that it has the data ready to put on the bus and therefore requests bus access. Only when that access is granted by the acknowledge signal is the chip allowed to drive the bus and place the address it wants to transmit upon it. There is actually a critical timing constraint in this protocol: since after the /bus_ack SCX expects the data to be ready within a certain time (compare figure 4.12). The chip designer must ensure that his chip is fast enough in this respect. After the address is placed on the bus, the chip now withdraws its request, while still holding the address, which is finally withdrawn by the chip after the withdrawal of the SCX’s acknowledge.
Figure 4.10: A memory cell pixel to be used in the SCX MNC to store analog operation parameters. The signals /memsel.x and /memsel.y supplied from the AER receiver circuit are used to connect the content of the memory cell at an address on the chip to the analog output line alogout. alogout and thus the memory cell’s content can then be observed off-chip.
4.3. SENDING AE TO SCX

Symbol

Schematics

Figure 4.11: Circuit to access a probe point within the SCX MNC framework. The signals /memsel.x and /memsel.y are also used to connect the content of a memory cell at an address on the chip to the analog outputline alogout (figure 4.10), but with this simple probe circuit any voltage signal can be connected to alogout and be observed off-chip.

In this work the circuitry in the chip’s core knows nothing of that extended protocol and performs a standard 4-phase handshake (figure 4.12). The translation is handled at the chip’s periphery in a special tristate latch-pad (figure 4.21).

There is not only a competition for bus access among multiple chips, but also within the chip the neurons compete for the possibility of sending an action potential. An arbitration circuit (figure 4.13, 4.15 and 4.16) resolves that competition. The arbitration is not completely fair: if the sender starts to get overloaded, neuromorphs close to the previous winner in address space gain an advantage. This property occurs in other implementations too [10, 9]. It is caused by the tree-structure of the arbiter and the ‘greediness’ of subtrees, i.e. subtrees do not withdraw their request until all their leaves are served. See also figure 4.27 for an illustration of that effect.

A novelty in the particular sender circuit presented here is the distinction between an arb.ack and a pixel.reset signal. In the two dimensional case, this allows for greater speed. In former solutions [10, 11] the arb.ack signal served a double purpose and fulfilled the rôle performed here by pixel.reset too. The arb.ack signal was latched while /bus.ack was active. That meant that if the y-address changed, along the x-dimension the application with the
Standard 4-phase handshake

\[ \text{req} \quad \text{ack} \quad \text{data} \]

SCX-‘handshake’ (multiple senders)

\[ \text{req}_i \quad \text{ack}_i \quad \text{data} \]

individually control signals  max

shared data-bus

Figure 4.12: A normal 4-phase handshake and the adapted protocol for SCX (drawn with active high \textit{req} and \textit{ack} signals, whereas the actual SCX signals are active low). Since SCX receives from multiple senders, there cannot be a conventional two partner handshake.
Figure 4.13: The block diagram of a 3 bit sender (symbol (at left) and schematics). If neuromorphs are arranged two-dimensionally, two of these circuits are required. pu_bias is a global bias. bus_req and /bus_ack are global signals for the off-chip handshake with SCX. bit0 through bit2 connect to latch-pads (figure 4.21) and through them to the off-chip bus. Neuromorphs can request access to the off-chip bus via the /arb_req signals shown entering from the left. If there are several simultaneously active requests, only one at a time gets granted access to the bus by the arbiter tree. The address of the winning neuromorph appears on the internal bus (bit0, bit1 and bit2) which is driven by the common drain transistors depicted as boxes labeled 1 or 0 (figure 4.18) that get activated by the acknowledge from the arbiter tree, and the bus_req signal is sent off chip. When that request is acknowledged by /bus_ack, the neuromorph receives the pixel_reset signal and must withdraw its request. The next neuromorph now gets access to the internal bus. See also figures 4.22 to 4.25 for an explanation of the signal propagation.
Figure 4.14: The i&f soma as it is used on the MRR-chip with the additional interface to the arbiter. The same layout can be used as in the 'standalone' version in figure 3.8. On our chip there were four learning synapses (as in figure 4.8), a non-learning excitatory synapse and an inhibitory synapse connected to each soma.
Figure 4.15: The schematics of the arbiter cell (see also figure 4.13). This is the stackable circuit that exclusively acknowledges one of two incoming requests if its own outgoing request is acknowledged. The two NAND gates on the left are at its core. They actually form a RS flipflop with active low inputs (/S = req0 and /R = req1) (figure 4.17), and as long as there is no request, it is in a state that in general would be indeterminate for a RS flipflop, namely both /S and /R are active. In that state in this particular implementation of a RS flipflop both outputs (Q and /Q) are high (compare truth table in figure 4.17). When now /S or /R becomes inactive (equivalent to req0 or req1 becoming active), the flipflop will be set or reset by the remaining active signal: the output of the NAND whose request signal became active will go low. If now there is a second request, the state of the flipflop will not change, as long as the first request remains. The request (req_out) is passed on higher up the hierarchy of arbiter cells via a NAND gate. If an acknowledge (ack_in) from higher up is returned, it is passed on to the winner of the two inputs (ack0 or ack1).
Figure 4.16: The arbiter logic circuit (see also figure 4.13) shown here (symbol (upper left) and schematics) is the mediator between the neuromorphs, the arbiter-cell-tree (figures 4.13 and 4.15) and the global chip bus. It has a pull up transistor \(\text{arb}_\text{req\_pullup}\) for the \(\text{arb\_req}\) line because in the two-dimensional setting this is a common drain line used by all the neuromorphs along a row to request bus access. If that request gets acknowledged by the arbiter cell tree \(\text{arb\_ack}\), the neuromorph is granted access to the bus and the request is passed on off-chip \(\text{bus\_req}\). The RS flipflop is used in a similar way to the one in the arbiter cell (figure 4.15): it indicates whether the \(\text{bus\_ack}\) signal or the \(\text{arb\_ack}\) signal comes first. (Note: The flipflop is implemented as in figure 4.17. Therefore when its inputs \(\text{R} \) and \(\text{S}\) are both active (low) the outputs are both high.) Only if the \(\text{arb\_ack}\) signal is first is the \text{reset\_pixel} signal passed on to the neuromorph whose address has been sent and so it is told to reset itself and to withdraw its request. Another neuromorph might now get the \(\text{arb\_ack}\) before the \(\text{bus\_ack}\) is withdrawn and will only be reset by \text{reset\_pixel} after the next handshake.
Figure 4.17: An active low RS-flipflop (symbol, schematics and truth table). In this work that RS-flipflop is used in its 'illegal' state, namely with both /R and /S active (figures 4.15, 4.16). Generally an RS-flipflop's output is not defined in this state. Here however, active low RS-flipflops have been implemented using two NAND gates, so we know that what would otherwise be an illegal state results in both outputs being high.

arbiter could only happen after the withdrawal of the /bus_ack signal. This caused a delay in the new request. In the arbiter used in this work however only the pixel_reset signal is latched and the competition for bus access in both dimensions is re-initiated at the moment of the /bus_ack going active. If there is a requesting pixel, everything can be set up for the next transmission, which will occur without further delay when the /bus_ack is withdrawn (see figure 4.23 and 4.26). Recent work by Boahen [9] uses other means to improve arbitration speed. In that work, a row of neuromorphs is latched when it gets selected by the y-arbiter. The x-arbitration (column-arbitration) is then performed within that latched row. The y-arbitration can then restart right away and is already ready when the first row has been served. The reported cycle times are even faster (40-70ns for a 104x96 neuromorphs chip) than the ones achieved with our approach (62.5ns for a 4x16 neuromorphs chip).

Another new detail is a clean interface from the arbiter to any neuromorph that expresses an event by a spike, or just a rising voltage edge (figure 4.19). In former solutions where the neuromorph was more tightly bound into the arbiter, its state could influence the speed of a communication, for example when a pixel cannot be reset quickly because it is receiving strong input or when a pixel illegally withdraws its request without being acknowledged. Now the neuromorph and the arbiter are cleanly decoupled which also makes
Figure 4.18: A Pull down (A) and a pull up transistor (B). They are used to set the address output when sending an address-event (figure 4.13). The logic signals /arb_ack in (A) and arb_ack in (B) are just passed through these blocks.
Figure 4.19: This is a block (symbol and schematics) used to interface between two dimensionally arranged spiking neuromorphs and the arbiter. See figure 4.14 for an example of how it can be connected to a pixel. Other versions, in which the neuromorph itself handles the communication with the arbiter, are possible. They must follow the same protocol as this circuit does. A special flipflop that is set by a rising edge rather than by a level (figure 4.20) captures the spike (using the edge of a spike rather than the level avoids multiple address-events being generated by the same spike) and sets the logic node labeled req which indicates that the pixel wants to transmit a spike. This causes a request (/arb.req.y) to the row arbiter. If the row arbiter acknowledges (arb.ack.y), the circuit then applies a request to the column arbiter (/arb.req.x). The rest of the communication is now performed by the arbiter logic, which also sends the reset.pixel signals, once the address is transmitted off chip. The two reset.pixel signals together reset the flipflop (and therefore the pixel's request) by setting the node served. The circuit is now ready for the next spike. If there have been spikes in the meantime they were ignored.
Symbol Schematics

Figure 4.20: A flipflop that is set by an edge rather than a level signal. The reset however is a level signal. It is used in the circuit that interfaces any spiking neuromorph with an AER sender circuit (figure 4.19).

Schematics

Figure 4.21: The latch-pad is the circuit that translates a normal handshake to the protocol compatible with SCX. It is powered by the pad-Vdd, hence the symbols PVdd. The /bus_ack latches the address_bit (corresponding to one of the bit0 through bit2 in figure 4.13), which is not sustained on the chip from that moment on, and opens the transmission gate such that the chip now drives the SCX-bus which is connected to the node pad. The square is the symbol for a bonding pad in the layout.
4.3. SENDING AE TO SCX

Figure 4.22: This and the three following figures are a sequence of snapshots of the signal activity in the two dimensional arbiter in an example with 2x2 neuromorphs. For simplicity all signals are considered active high. The states when the circuit is waiting for a response from off chip (by setting or resetting the bus acknowledge *back*) are presented. More details on the timing of individual signals is given in figure 4.26. The labels A, B, C, and D in this and the next three figures correspond to the ones labelling the states in figure 4.26. In this first snapshot two neuromorphs, indicated by black dots, are active. Both send their arbiter request signal (*areq_y*) to the row arbiter on the right but only the request from the neuromorph at coordinates (1,1) is allowed to propagate to the column arbiter (*areq_x1*) and finally to put its address on the bus (*adr_y* and *adr_x*) and to request to send it off-chip (*breq*). Now the circuit waits for the transmission to be acknowledged (*back*). Figure 4.23 shows what happens then.
Figure 4.23: The next snapshot of signal activity in the 2×2 arbitration example that starts in figure 4.22. As the first address is acknowledged (back), the \( p_{res} \) signals to the neuromorph that has just transmitted becomes active and so resets pixel (1,1). They remain active until the off-chip bus acknowledge (back) will be withdrawn. As a consequence, that neuromorph's requests are withdrawn and the next neuromorph (0,0) has an opportunity to transmit. (Note that in previous implementations [10, 11] where there was no separation between an arbiter acknowledge (aack) and a reset pixel (p_res) signal, the column address \( adr_x \) could not be ready before the back signal was withdrawn. Then the acknowledge/reset signal was latched during the bus acknowledge. Therefore the switching to a different row and the competition between the columns could only happen after the withdrawal of the back signal.) The new winner in the upper left corner is prevented from generating the breq signal until the back signal becomes inactive, and the former winner is not maintaining breq. So the breq is withdrawn. The new winner has its address ready and can apply its off-chip request as soon as back is taken away (figure 4.24).
Figure 4.24: The scenario of figure 4.23 continued: as back is withdrawn, the new winner who was already ready can instantly request access to the bus. The pixel reset signals to the previous winner \((p\_res\_y1 \text{ and } p\_res\_x1)\) are also withdrawn with the withdrawal of back, allowing that neuromorph to become active again.
Figure 4.25: The last snapshot in the arbitration example. It follows the situation depicted in figure 4.24. The second transmission is acknowledged by back and also the second neuromorph gets reset by the reset signals p.res.y1 and p.res.x1. Its request signals are withdrawn and the bus request breq is also reset.
Figure 4.26: The timing diagram of signals related to figures 4.22 to 4.25. Again for simplicity all signals are considered active high even if in the real circuits they are active low. The states shown in figure 4.22 to 4.25 are marked here with vertical lines labeled A, B, C and D. The diagram shows the theoretical relative timing of events, not the absolute timing, hence no time-scale is given. Arrows lead from the last event that was necessary to cause another, to that next event or those next events. Events may however be dependent on a number of other conditions and previous events. The signals are listed in the order of their first activation.
it easier to reuse the arbiter with different existing neuromorphs, without
changing those neuromorphs. Again one must pay for this in chip area (the
additional block measures 152×61μm in the 2.0μm Orbit process), but one
saves in design-effort. The schematics for the neuromorph/arbiter interface
are shown in figure 4.19. The soma equipped with this interface is shown in
figure 4.14 (Compare with figure 3.8).

4.4 Sender Performance

The most convenient chip for testing the AER circuits that were designed
during this work contains a two dimensional array of 16 × 4 i&f neurons.
They have two synapses each, one inhibitory and one excitatory. We only use
the excitatory synapse in the following experiments. The excitatory synapses
are of the simplest kind (figure 3.10 A): two n-FET transistors in series, one
opened during the incoming spike; the other one (with gate voltage $V_{w+}$)
limiting the current from the soma. (Remember: with our variant of an i&f
soma, the membrane voltage’s sense is inverted (figure 3.13).) One column of
16 neurons shares $V_{w+}$. The synapses receive input-spikes from a monostable
(figure 3.6) that receives its input from an AER receiver (figure 4.4). By
configuring the monostable such that it emits an infinitely long output pulse,
one can turn on a constant current influx to individual neurons. This causes
the neurons to spike at a constant regular frequency, controlled by $V_{w+}$. By
connecting /bus_req directly to /bus_ack on the chip we reached the maximum
output frequency, only limited by the sender’s speed. Thus we were able to
transmit at up to 16MHz, but the addresses were not reliable at that speed
because the charging and discharging of the bus lines took more time than
one half-cycle. We tuned the pull up bias (pu_bias) such that the maximal
transmission frequency slowed down to 10MHz. Thus the speed at which
addresses could be reliably placed on the bus was actually the bottleneck
rather than the arbiter speed. Only with larger arrays would the arbitration-
speed become more important. Because of the binary tree structure of the
arbiter the arbitration speed scales with the logarithm to base 2 of the number
of sending elements or in other words with the number of used bits in the
address space. Our test chip contained 64 neurons. 16MHz corresponds to an
interval of 62.5ns . If that whole time was due to the arbitration, then one
address bit would require approximately 10ns of processing time. So with our
test setup we would expect that communication at 10MHz could be achieved
Figure 4.27: Graphs showing the throughput of spikes through our AE sender as we gradually overload it. We turned up the activity of 16×4 regularly spiking neurons on a AE chip. A raster plot and the individual frequencies are shown.
for chips containing up to 1024 neuromorphs. The raster plots shown in figure 4.27 were produced from the test chip as it was tuned to produce a maximal transmission frequency of 10MHz. We started by having individual neurons fire with a frequency of around 40-85 kHz ($V_{\text{weight}} = 0.48$). Although they fire regularly if turned on individually, due to crosstalk, they do not do so when all are turned on. Their firing frequency is greater since there is additive noise on $V_{w+}$ that is symmetrically distributed around zero. (This noise will have an increasing effect on the current flowing to the soma, because that current is exponentially dependent on $V_{w+}$.) The intervals are also more irregular. The neuromorphs tend to fire clustered in time, but they are still able to transmit at their individual frequencies. The coefficient of variance (CV) of the neuron frequencies is 0.22, since they fire at slightly different frequencies because of device mismatch. The CV is a measure of the diversity of the neurons' firing frequencies. It can be expected that if the neurons start to overrun the arbiter and the transmission slots get arbitered in a regular fashion that the CV will decrease. In the following a smaller CV is considered an indicator of transmissions being restricted to regular time slots and therefore individual spikes may be delayed or even lost.

The individual frequencies were then increased to lie within 80-239kHz ($V_{w+} = 0.50$). The sum of all frequencies was 9.42MHz, very close to the maximal sender frequency. Still there is no obvious pattern visible in the raster plot, except for a weak clustering of activity in time which can be explained by crosstalk among neurons. The address frequencies are still determined by the neurons' firing frequencies (CV=0.21). The timing of spikes is still preserved.

As we now turn up the frequencies further ($V_{w+} = 0.51$), the sum of the neuronal frequencies now exceeds the maximal frequency of the sender. Obviously spikes are lost. The neurons' frequencies can no longer be reliably reconstructed. The patterns occurring in the raster plot are now mainly caused by the arbiter: neurons close to each other in address space are served by the arbiter together. Address activity gets levelled out since basically every neuron gets served in sequence. The address frequencies are now between 96kHz and 194kHz and their CV drops to 0.14.

In the latter three graphs in figure 4.27 ($V_{w+} = 0.52, 0.55, 0.56$) the clustering in address space gradually gets worse. Half of the address space remains dominant for a while until it switches to the other half. The CV of the frequencies rises again in the last two graphs, because of the imbalance between the two halves of the address space. Finally in the last graph the lower half
4.5 Conclusion

A chip design framework has been developed that allows circuits that use temporal codes (neuromorphs) to be used in the silicon cortex system (SCX, a system under development at the Institute of Neuroinformatics). This kind of framework allows the design of chips that contain multiple asynchronous processing and storage elements and that use temporal encoding. Together with SCX these chips form an embedded system and any configuration of the connections among the processing elements can be programmed into SCX. The VLSI design framework contains circuits translating spikes into address-event representation (AER) and vice versa. To be able to use a neuromorph in that framework it only needs to take digital voltage pulses of a specific width as input and it needs to emit digital voltage pulses of any width as outputs. Furthermore the framework provides circuits that control and probe the content of FG analog memory cells or that probe analog voltages anywhere on the chip.

The arbitration among sending neuromorphs is done by an arbiter derived from the ideas in [10, 11]. It contains some improvements that accelerate
the arbitration. In the meantime however other approaches to improve the arbitration speed have been published [9] and can claim to be even more successful.
Chapter 5

Discussion

Approaches to artificial intelligence (AI) can be divided into two classes. There are those researchers who try to understand 'intelligence' and there are those who try to understand the vehicle of intelligence, the brain. The former believe that there must be a simpler way in achieving AI than to copy the brain, the latter believe that copying the brain, to a certain level of detail, is the simplest way. This thesis contributes to the second approach.

It is an ongoing discussion as to what level of detail is necessary in a model of the nervous system to reproduce its basic functional abilities. It is desirable to keep that level of detail low, to facilitate understanding and simulations of neural models. That is why traditional models of neural networks simplify the individual neuron dramatically. For instance they do not communicate by sending nerve pulses but by transmitting analog values that represent an average spiking frequency. The time is often discretized. Although such models do offer some insight on processes in the natural nervous system, they cannot explain it fully. More and more evidence is found that the precise spike pattern is relevant to information processing in the brain. As already mentioned in the introduction (section 1.1.6) reaction times of test subjects are often too fast for averages of spike-trains to be computed [71] and the same highly entropic visual stimulus can repeatedly evoke the almost exact same spike pattern in a neuron in visual cortex [6]. However, it is not yet clear how the nervous system would use temporal coding to processes sensory input that finally results in some motor output. It is only marginally understood.
how information is represented. But it seems unavoidable to include explicit representation of spike events in at least some parts of a working model of the brain.

This thesis answers the question of how spike signals could be used to advantage in processing with an example. An artificial learning mechanism that is based on spike patterns and an encoding that can be learnt by this mechanism are introduced. The learning rule complies with physiological measurements in regard to rewarding causal relationship between pre- and postsynaptic spike with an increase in synaptic efficacy and in regard to heterosynaptic depression as a result of potentiating individual synapses. Thus some aspects of the theoretical rule as yet unconfirmed by physiological experiments may well be found in the behaviour of biological synapses too.

Since neural models of the level of detail of the learning rule introduced in chapter 2 are hard to simulate on a serial computer in real time, the learning rule has been implemented in neuromorphic aVLSI. It has been design to be embedded in a system that is able to arbitrarily connect chips containing multiple elements that process temporal codes. An extension of this complete real time system would well be suited, for example for controlling an autonomous agent in a natural environment. The prototype chip in principle contains all the necessary elements to achieve this. However more developmental work would need to increase the density of the model neurons and the problems with the multi-neuron experiments (section 3.2.3) need to be fixed before the system would be fit to learn to encode relevant sensory information for a foraging robot.

5.1 Contribution in the field of learning theory

The power of encoding using temporal spike patterns has only recently begun to be explored. Even less is known about learning mechanisms that are suited for a system that uses temporal encoding.

This thesis defines and analyzes an example of a spike based learning rule, the modified Riccati rule (MRR). It is an unsupervised, local learning rule that rewards causal relationship between presynaptic and postsynaptic spike by increasing synaptic efficacy. Causality between pre and postsynaptic spike implies activity correlation and thus the MRR can be considered a specialization of a Hebbian learning rule (rules that reward activity correlations). Other
contemporary models [25, 29, 39] and physiological experiments [57, 14] share this property.

This particular learning rule however, lets synapses compete for synaptic strength such that the synaptic weight vector is implicitly normalized. The only signal that needs to be shared among the synapses to achieve such a coordinated behaviour is the time of a postsynaptic firing. Implicit weight vector normalization requires no extra normalization step in the algorithm and ensures that the weights are normalized at all times. Weight vector normalization on one hand is useful to limit the self-amplifying of synaptic weights and on the other hand prevents the neuron from becoming completely unresponsive. Weight vector normalization also sets the maximal gain of a neuron with respect to the length of the input-vector and can thus contribute to stabilize network activity.

The MRR can be used in systems that do use spikes for communication but do not use temporal encoding too: if the inputs and output are deprived of temporal correlations and are independently Poisson distributed spike-patterns, then the rule's behaviour can be described on the average signal rates as being either like the non-Hebbian Riccati rule or like the Hebbian Oja's rule. Which of the two depends on the product of the average output rate $O$ and the neurons membrane and correlation signal time constant $\tau$: If $O\tau$ is big the behaviour is closer to the Riccati rule, if $O\tau$ is small Oja's rule is the more accurate description.

If there are temporal correlations in the input and output spike patterns however, the behaviour changes. If for instance the neuronal model is an integrate and fire model, the output is temporally correlated with the inputs. If one again is to describe the behaviour of the neuron in terms of rate it still depends on the product $O\tau$, and with that product being large the behaviour is still that of the Riccati rule. However with $O\tau$ decreasing the weights behave in the manner of a hysteretic winner take all algorithm. If there are temporal correlations between the inputs too, inputs that tend to fire close in time 'team up' in the winner take all competition. Thus the neuron tends to become selective to synapses that receive coincident inputs. Adding delay lines to the synaptic inputs, coincidences at the synapses can be caused by sequential spike activity in the afferent neurons and so the receiving neuron can become selective to spatio-temporal input spike patterns.
5.2 Contribution in the field of encoding

Significant theoretical research has been conducted on the power of temporal encoding [52]. This thesis approaches that issue from a more practical side, proposing a way of encoding sensory input in a network of spiking neurons.

The issues of encoding and learning are entwined. The motivation for spike based learning is the belief in the importance of temporal encoding for the processing in the nervous system. But the relation between encoding and learning is bidirectional: information about encoding in a system can be gained from the kind of learning that is used. Therefore this thesis gives an example of a concrete encoding scheme that can be learnt by the proposed spike based learning rule MRR. It is shown how this scheme can bind features to objects and how temporal sequences of percepts can be learnt.

5.3 Contribution to neurophysiology

Although there were no neurophysiological experiments conducted for this thesis, it is known from other research that the MRR shows a number of similarities with the learning behaviour of biological synapses. That is in particular the rewarding of causal relationships of presynaptic and postsynaptic spikes [57, 14] and heterosynaptic depression when the efficacy of one or a few synapse is increased [51]. The experiments in this thesis give some examples of what these common properties of biological and MRR neurons could be good for. Other similarities of biology and the theoretical MRR could exist. This might motivate physiologists to check for other properties of the MRR in real neurons. Here are a few questions that could be explored:

1. Does heterosynaptic depression depend on the backpropagating action potential?

2. Although exact weight vector normalization is unlikely, is it still possible that heterosynaptic depression limits the total of the synaptic weights of a neuron in a way that helps to control the maximal gain of a neuron?

3. Do neurons become specific to coincident spike inputs?

Another suggestion is to pay closer attention to temporal patterns instead of frequency in neurons responses to specific stimuli. Already this has been
done as spike-patterns were recorded from a neuron in a monkey’s visual cortex as the animal was presented repeatedly with the same random dot stimulus [6], and it turned out that the temporal pattern of these trains were highly reproducible from trial to trial. It might be possible that frequency responses are but a side-effect of what some neurons really do.

5.4 Contribution to aVLSI engineering

It is the trend in high performance computer development to increase computational power by increasing the number of processing units rather than to increase the power of individual units. Even PCs are nowadays often equipped with multiple processors. The brain in a way puts that trend to its extreme: it consists of a huge number ($\approx 10^{11}$) of simple and slow processing units, the neurons. In massively parallel systems, data communication becomes a speed limiting factor. Keeping the communication localized makes it easier to make communication fast. In cortex for example the major part of a neuron’s synaptic spike input originates in nearby neurons\(^1\). Keeping the data traffic localized requires local means of data storage. The brains long term storage capacity is believed to lie in the synaptic efficacies, so every neuron is equipped with approximately 10,000 storage units. This thesis contributes the chip design framework to an embedded electronic system (a research project of the Institute of Neuroinformatics, ETHZ/UNIZ, Switzerland, named ‘silicon cortex’) that aspires to achieving brain-like computation providing the features mentioned in this paragraph: on-chip distributed processing capacity and distributed analog non-volatile storage on floating gates (though still much more modest in absolute numbers and density than in the brain), and fast asynchronous inter-chip spike communication. Co-localization of storage and processing units and asynchronous spike allows to perform parallel brain-like computation. We call that chips that use that design framework ‘SCX MNC chips’.

SCX MNC chips use arbitered address-event representaion (AER) for the off-chip communication. The arbitration distributes communication time slots among the neuromorphs that want to send a spike. If two neuromorphs intend to transmit at the same moment, one of them is delayed by a few tens or

\(^1\)One example of an estimate of the type of connections to one particular cortical cell type can be found in [3]. The claim there is that only 13% of the connections to a layer 4 basket cell in visual cortex are made by axons originating outside cortex in the thalamus.
hundreds of nanoseconds. It is assumed that this delay is to short to matter to the receiving neuromorph. The building blocks are mainly put together with standard digital logic elements with minimal size transistors. Designs where ratios of transistor sizes matter have been avoided. This makes the design more easily scalable and it can more easily be ported to other production processes. In the present design, the arbitration has been optimized for speed as compared to older designs [10]. However, other groups can claim to have improved the speed of the arbitration even more by other means [9] (see also in section 4.3). There are also AER systems that do no arbitration [61, 60]. Those designs are simpler and consume less chip area, and they can transmit address-events at higher frequencies than the arbitrated AER systems. The trade-off is the possibility of collisions between neuromorphs that want to transmit their address at the same instant. Therefore AER systems with no arbitration use error detecting codes and thus more address bits. In case of a collision, the spike transmission is lost. A recent development is serial AER [62]. Addresses are transmitted serially and therefore the bus bit-width is independent of the number of communicating neuromorphs which makes the system more flexible. Communication speed however, is slowed down.

Non-volatile floating gate analog memory is used in the chip design framework based on a memory cell proposed in [17]. That memory cell was extended to be accessed by addressing. Others have used the same kind of memory cell in a memory array that is accessed by scanning [32]. Scanning requires only two pins for control signals, independent of the number of memory cells but it requires a more sophisticated off-chip control system than addressing. That particular memory array [32] also has a mechanism to compensate for the capacitive shift of the memory cell content that is caused by the switching of the tunneling voltage. Some problems (see section 3.2 and figure 3.12) of the aVLSI MRR-synapse could have been avoided with this mechanism.

A neuromorphic aVLSI model that approaches the behaviour of the MRR has been implemented, and it has been placed into our SCX MNC chip design framework. This allows the neuronal MRR model to process information in real-time. Every synapse and every FG memory cell (containing operation parameters and synaptic weights) can be addressed, synapses to receive a spike event and memory cells to be read and changed. Every AP output produces an address-event that can be observed from off-chip an that can be routed by SCX to any synapse on the same or on another chip. On-chip learning changes the analog synaptic weights that are stored on FG memory cells directly at the synapses. This kind of storage preserves the learning state even when the
power supply is interrupted and permits very slow processes such as learning. Single neuron behaviour is well matched to the theory. The major discrepancies between chip implementation and theoretical learning rule though lie in the mismatches of the intrinsic strength of individual synapses and in the poor weight-vector normalization for short membrane and correlation signal time-constants. Asymmetry in the synaptic strengths can be considered a desirable symmetry breaking factor in some learning tasks. The poor weight-vector normalization is possibly an effect of a bad matching of parameter controlled time constants. These parameters were hard to tune though, because it was neglected to make them directly observable on the present version of the SCX MNC. Single chip multi-unit activity in the present implementation of the MRR-neurons suffers from the i&f mechanism falling into metastable states. This distorts the biases of other neurons by ways described in section 3.2.3. These metastable states should be easily avoidable in future designs by adding the positive feedback that is suggested also in section 3.2.3.
Seite Leer / Blank leaf
Appendix A

Complete schematics of SCX MNC

The chip that was used for most experiments in this thesis is described here in detail.
Figure A.1: Complete SCX MNC
Figure A.2: Magnification of the left pads of SCX MNC
Figure A.3: Magnification of the right pads of SCX MNC
Figure A.4: Magnification of the top pads of SCX MNC
Figure A.5: Magnification of the bottom pads of SCX MNC
Figure A.6: Magnification of the top left corner of SCX MNC
Figure A.7: Magnification of the bottom right corner of SCX MNC
Figure A.8: Magnification of the middle left part of the core of SCX MNC
Figure A.9: Magnification of the bottom left corner of the core of SCX MNC
Figure A.10: The ‘UD latch’ circuit that latches the U and D signals that control the increases and decreases of the memory cells. If both muxU and muxD are high than the circuit moves the selected memory cell’s content logout to match the target voltage alogin.
Figure A.11: Wide follower and wide comparator
Figure A.12: FG memory pixel that is used on the SCX MNC
Figure A.13: soma pixel that is used on the SCX MNC. It actually consists of an i&f soma and an excitatory and an inhibitory synapse.
Figure A.14: Synapse pixel that is used on the SCX MNC
Figure A.15: RS-flipflop pixel that is used on the SCX MNC. It can be used for digital storage and set and reset by two different address events.

Figure A.16: Tristate output pad that is used on the SCX MNC. Its layout does not actually include a pad connection so it has to be connected to a bare pad.
Appendix B

An old version of a plastic FG synapse

The older version of the synapse that was used in chapter 3 in section 3.3, together with a silicon neuron to emulate LTP and LTD.
Schematics

Figure B.1: Schematics of our learning mechanism in the AMPA synapse [64] used on a ‘silicon neuron’. It shares the property of rewarding causal relationships between pre- and postsynaptic spike, but does not perform weight vector normalization. Again a core memory cell is used as described in Figure 3.2. It holds the synaptic weight which sets the EPSP amplitude. The circuit in the trigger icon is a simple positive edge triggered monostable (figure B.2). The output is a pulse, whose width depends on an input parameter (\textit{fall} for trigger 1 and \textit{corr} for trigger 2). Trigger 1 causes a pulse of a fixed width to fire when a presynaptic spike arrives at the synapse. This pulse decrements the \textit{corr} signal. This \textit{corr} signal keeps track of the recent presynaptic activity. It decays back to \textit{Vdd} at a constant rate set by \textit{tau corr}. As in physiological findings [67], the cells action potential (AP) influences events at the synapses. When the silicon neuron releases an AP, the weight is updated. The signal \textit{learn down} is a pulse of fixed width triggered by the rising edge of the AP and decrements the weight, while the pulse output of trigger 2 increments the weight for a time determined by the \textit{corr} signal. The difference of these two pulse widths decides whether the weight is increased or decreased. The \textit{ap-falling} signal is a pulse triggered by the falling edge of the AP and resets the \textit{corr} signal.
Figure B.2: Details on the trigger circuit used in figure B.1. It is a simpler monostable than the one in figure 3.6 that reacts with a pulse to a rising edge. In contrast to the monostable in figure 3.6 its output pulse is at most as long as the input pulse. But if the input pulse is long enough, it also emits a pulse of an adjustable width. The idea is quite simple: an AND gate is fed the input and its inverse. The inverse is delayed by a capacitance and two current-limiting transistors. The rise transistor only prevents glitches on the falling edge of the input but the gate voltage on the fall transistor can adjust that pulse width.
Appendix C

Very basic circuits
Figure C.1: Circuits for NAND NOR and NOT gates.
Bibliography


Index

/l, 111
/memsel.x, 111
/memsel.y, 111
/rec.x, 111
/rec.y, 111
D, 111
E(L), 39
I, 1
I_0, 72
I_{ld}, 76
M, 26
O, 1, 3
O(t), 23
P, 2, 45
P_{decay}, 45
psynapsesfire, 47
psynapsesinc, 46
psynapsestay, 47
psynapse, 47
P_{fire|k}, 51
P_{fire|(M,k)}, 48
S_i, 2
T_{injection}, 82
T_{tunneling}, 76
U, 111
V_T, 72
W, 1, 2
\Delta V_{corr}, 82
\Delta V_{soma}, 72
\Delta^+ w, 82
\Delta^- w, 82
\Delta^+ V_{weight}, 74
\Delta^- V_{weight}, 74
\bar{C}, 31
\bar{D}, 2
\bar{I}, 2
\bar{I}(t), 23
\bar{O}, 2
\bar{\bar{C}}, 31
\bar{\bar{w}}, 24
\bar{\bar{C}}, 31
\epsilon_{i}, 24
f_M, 48
\omega^*, 18
\bar{\omega}, 18
\alpha, 82
\alpha, 18
\beta, 76
\beta, 18
M, 26
\delta, 24
\hat{O}, 18, 20, 27
\hat{O}(t,T), 23
\hat{\bar{I}}, 18, 28
\hat{\bar{I}}(t,T), 23
\hat{\bar{\bar{C}}}, 40
\( \hat{w}^* \), 28
\( \tilde{w} \), 28
\( \hat{O} \), 48
\( \kappa \), 72
\( \mu \), 3
\( \tau \), 26
\( \tilde{O} \), 29
\( \chi \), 26
\( t_k \)
\( \chi_{i,j} \), 26
\( t_{i,j} \)
\( c \), 82
\( c_i \), 31
\( f_L(l) \), 39
\( f_T(T) \), 39
\( f_{LRB30DT} \), 39
\( learn_D \), 112
\( learn_U \), 112
\( lu \), 111
\( t_k \), 20, 27
\( t_{i,j} \), 20
\( w \), 72
\( \bar{x} \), 26
4-phase handshake, 115

accuracy of FG memory, 71
action potential, 5
address-event, 13, 16
address-event representation, 104
AER, 13, 16, 104
AER receiver, 107, 108
AER sender, 112
AI, 135
analog dynamic memory, 9
analog memory cell, 70
analog memory cell pixel, 113
analog non-volatile memory, 10
analog static memory, 11
analog storage, 9

ANN, 2
AP, 5
arbiter, 116
arbiter cell, 118
arbiter logic, 119
arbiter pixel interface, 122
arbitered address-event communication, 114
arbitered AER, 114
arbitration, 114
artificial intelligence, 135
artificial neural networks, 2
asynchronous spiking networks, 12
aVLSI, 6

binding, 57
Binding temporal sequences, 58
capacitive storage, 9
coding, 56
correlation signal, 32
correlation signals, 31
data reduction, 3
delta function, 23
demultiplexer, 70, 108, 109
digital computer, 6
Dirac delta function, 23
direction selectivity, 59
dynamic memory, 9
dynamic storage, 9
dynamic temporal memory, 9
dynamic window, 9

edge flip flop, 123
EEPROM, 11
electron injection, 11
encoding, 138
error backpropagation, 2
error function, 2
event based simulations, 65

FG, 9, 11, 15, 71
INDEX

FGaRAM, 69, 71
floating gate, 11, 15, 71

gradient descent method, 3

handshake, 115
Hebbian learning, 3
high voltage switch, 68
hot electron injection, 11
human learning, 1

i&f, 24–26
i&f aVLSI soma, 80
input-neurons, 2
input-vector, 2
instantaneous frequency, 23
integrate and fire, 24–26
inverter, 164

latch, 110
latch-pad, 123
learn down, 79
learn up, 78
learning, 1
learning rules, 1
local learning rules, 3
long term depression, 17
long term potentiation, 17
loop gain, 54
LTD, 17, 97
LTP, 17, 97

membrane potential, 26
membrane potential probability den-
sity, 41
memory cell pixel, 113
memory core cell, 67
metastable state, 96
MNC, 83, 103
modified Riccati rule, 17, 32
monostable, 73, 161
MRR, 17, 32
MRR aVLSI neuron, 75
MRR aVLSI synapse, 76
MRR-synapse, 111
multi neuron chip, 83, 103

NAND, 164
network stability, 54
neural networks, 2
neuromorph, 12, 103, 104
Neuromorphic aVLSI, 6
neurons, 1
neurophysiology, 138
non-volatile memory, 10
non-volatile weight storage, 72
NOR, 164
normalization, 17
NOT, 164

object binding, 57
Oja’s rule, 40
optimization algorithms, 2
orientation selectivity, 59
output vector, 2
output-neurons, 2

performance, 2
probe pixel, 114

rate based learning, 4
rate behaviour, 35
rate coding, 4
receiver, 107, 108
receiver logic, 110
refresh, 10
reinforcement learning, 2, 3
reliability of FG memory, 71
INDEX

retina, 7
reward signal, 3
Riccati rule, 17, 18
Riccati rule for spiking neurons, 20
RR, 18
RS-flipflop, 74, 120

SCX, 104
SCX MNC, 83
SCX multi neuron chip, 83
self-organization, 4
sender, 112, 116
serial computer, 6
silicon cochlea, 7
silicon cortex, 104
silicon neuron, 8, 97
silicon retina, 7
spike, 1
spike based learning, 1, 5
spiking networks, 12
SRR, 20
SRR1, 24
static memory, 11
supervised learning, 2
synapses, 1
synapses, non-learning, 77
synaptic correlation signals, 31

TDλ-algorithm, 3
temporal coding, 5
temporal sequences, 58
temporally correlated inputs, 92
transparent latch, 110
trigger circuit, 161
tunneling, 11

unsupervised learning, 2, 3

weight-vector normalization, 17
Seite Leer / Blank leaf
Curriculum Vitae

Address

Philipp Häfliger
Institute of Neuroinformatics UNIZ/ETHZ
Winterthurerstr. 190 CH-8057 Zürich
Switzerland

telephone: ++41 1 635 30 23
fax: ++41 1 635 30 53
e-mail: hafliger@ini.phys.ethz.ch

Date and place of birth

18th March 1970, Zürich, Switzerland

Parents

Eduard Häfliger, Diplomierter Elektro Ingenieur ETH (M. Sc. Electrical Engineering), 1941
Monika Häfliger, Teacher, 1942
Nationality

Swiss

Education

Doctor of Natural Sciences (PhD) 4/2000
-Crete Course in Computational Neuroscience 10/1996
-Telluride Workshop on Neuromorphic Engineering 7/1998

Research and teaching assistant at the Institute of Information Systems, ETHZ 4/1995-10/1995

Informatik Ingenieur Diplom ETH (M. Sc. Computer Science) 3/1995
-Member of the winning team in the ETH-internal ACM programming contest 1994
and participant in the western Europe ACM programming contest


Matura Typus B 1989
Kantonsschule Wiedikon, Zürich 1983-89
Primarschule Friesenberg, Zürich 1977-83

Scientific Publications

1. A spike based learning rule and its implementation in analog hardware (Ph. Häfliger); PhD Thesis, 2000
2. Learning a Temporal Code
(Ph. Häfliger); European Symposium on Artificial Neural Networks, 1999, D-Facto publications, Belgium, P. 423-428

3. Floating Gate Analog Memory for Parameter and Variable Storage in a Learning Silicon Neuron

4. Spike Based Normalizing Hebbian Learning in an Analog VLSI Artificial Neuron

5. Weight Vector Normalization in an Analog VLSI Artificial Neuron using a Backpropagating Action Potential
(Ph. Häfliger and M. Mahowald); Neuromorphic Systems, Engineering Silicon from Neurobiology, editors L. S. Smith and A. Hamilton, 1998, P. 191-196

6. A Spike Based Learning Neuron in Analog VLSI
(Ph. Häfliger, M. Mahowald and L. Watts); Advances in neural information processing systems, 1996, vol.9, P. 692-698


Languages

Fluent in German, French and English
Hobbies

Martial arts, skiing, snowboarding, drums, kites, games, reading, origami, climbing, para-gliding, all kinds of sports.