RUPRECHT-KARLS-UNIVERSITÄT HEIDELBERG

Andreas Hartel

Improving and Testing a Mixed-Signal VLSI Neural Network Chip

Diplomarbeit

HD-KIP-10-18

KIRCHHOFF-INSTITUT FÜR PHYSIK

Department of Physics and Astronomy

University of Heidelberg

Diploma thesis

in Physics

submitted by

Andreas Hartel

born in Mannheim, Germany

May 2010

Improving and Testing a Mixed-Signal VLSI Neural Network Chip

This diploma thesis has been carried out by
Andreas Hartel
at the

Kirchhoff Institute for Physics

University of Heidelberg

under the supervision of

Prof. Dr. Karlheinz Meier

Abstract

Improving and Testing a Mixed-Signal VLSI Neural Network Chip

This thesis presents improvements and testing of a neuromorphic mixed-signal VLSI ASIC. These include changes in both, the digital part and the analog part. Improvements to the former part that have been carried out by other group members are reported and partially experimentally verfied. These changes are the exchange of the standard cell library, an increase of the communication bandwidth and changes that account for the prerequisites of multi-chip operation in an isochronous gigabit transport network. The analog part of the chip has been extended by readout circuits to allow for calibration of parameter currents generated by the on-chip digital-to-analog converter. By extending the verification scheme of the analog full-custom circuits by an automatic parasitics extraction workflow, changes in the readout priority encoder for neural events, that were crucial for realistic experiments on the chip, could be verified in simulations. The improvements could also be successfully verified experimentally and calibrations with the analog current readout circuits have been carried out and are presented.

Verbesserung und Test eines neuronalen Netzwerk-Chips in mixed-signal VLSI Technologie

Die vorliegende Arbeit fasst Verbesserungen und Tests an einem hochintegrierten neuronalen Netzwerk-Chip in analoger und digitaler Mikroelektronik zusammen. Die präsentierten Veränderungen beziehen sich sowohl auf den analogen als auch auf den digitalen Teil des Chips. Änderungen an letzterem, die von anderen Gruppenmitgliedern durchgeführt wurden, werden aufgeführt und teilweise experimentell verifiziert. Dabei handelt es sich um den Austausch der verwendeten Standardzellen-Bibliothek, eine Erhöhung der Kommunikations-Bandbreite des Chips und Änderungen am Event-Transport, die für den Multi-Chip-Betrieb in einem isochronen Gigabit-Netzwerk nötig sind. Der Analog-Teil des Chips wurde um Auslese-Schaltkreise zur Kalibration und Verifikation des auf dem Chip befindlichen Digital-Analog-Konverters erweitert. Durch die Erweiterung der Verifikations-Methoden um eine automatische Berechnung der parasitären Effekte konnten Änderungen an den Prioitäts-Encodern der neuronalen Events durch Simulationen bestätigt werden. Letztere waren von großer Bedeutung für die Durchführbarkeit von Experimenten auf dem Chip. Die durchgeführten Änderungen konnten auch experimentell bestätigt werden und es wurden Kalibrationen mit den analogen Auslese-Schaltungen der konfigurierbaren Ströme durchgeführt. Diese werden ebenfalls präsentiert.

Contents

Chapter 1

Introduction

One might call science and reason the skills that single out the human race from the multitude of other living beings. It is the capability of the human brain to wonder about the world, how it works, where it comes from and which role we play in it. The human brain has also brought up the question about its very own nature, leading to an endeavor called neuroscience. But it is not only for philosophical reasons that one is interested in the essence of the human brain, it is also for more practical reasons, such as cures for neurological diseases and new paradigms for information processing. Nowadays however, neuroscience is still far from providing a complete understanding of the brain.

The FACETS project [24], in whose context the work presented in this thesis has been carried out, is a European scientific collaboration that aims at finding answers to these questions. FACETS stands for "Fast Analog Computing with Emergent Transient States". But what does that mean? Emergence is a phenomenon that has already been known to philosophers of the ancient world. It describes the fact that complex behaviour arises from a collection of simpler building blocks, of which no single one has properties that would by itself explain such behaviour. In the case of the brain it is the rather limited properties of its basic building blocks to exchange electrical signals that, when connected to large nets containing billions of these units, result in capabilities such as sensation, emotion and intelligence. It was around 1900 that the first pioneers of neuroscience stated that these building blocks be neurons and their synaptic connections.

The principle of emergence can also be applied to larger scales, making individual intelligence collective intelligence. The latter being responsible for important achievements in fields such as mathematics, physics and neuroscience. And of course, to physicists emergence is well known as the basic paradigm of statistical mechanics.

Nowadays, much of the endeavor in natural sciences involves the use of computational resources. Even though computer science has seen vast improvements concerning computational power in the last 40 years, many scientific calculations, especially simulations of complex systems such as large collections of neurons and synapses, still require very much time. This is due to the fact that general purpose computers obey the von-Neumann paradigm which involves serial processing of instructions, contrary to the parallel nature of neural information processing.

A solution to this problem, that is proposed by the FACETS project, involves very-large-scale integration (VLSI) of microcircuits that are designed to emulate the behaviour of biologically inspired silicon neurons and synapses. Information processing takes place simultaneously in every neuronal circuit, making it much faster than simulations of neuronal networks on general purpose computers. The FACETS hardware systems that implement these circuits are designed at TU Dresden[1] and the University of Heidelberg[2].

The FACETS hardware, parts of which will be described in this thesis, is a platform for neuroscientific experiments. It is built to supply a flexible and fast emulation tool for scientists that

[1] Highly-Parallel VLSI-Systems and Neuro-Microelectronics of TU Dresden, `http://hpsn.et.tu-dresden.de/`

[2] Electronic Vision(s) group of the Kirchoff Institute for Physics Heidelberg, `http://www.kip.uni-heidelberg.de/cms/groups/vision/home/`

intend to discover the computational paradigms that form the basis of information processing in the brain. Since it is biologically inspired, such hardware is often referred to as *neuromorphic hardware*. The development of neuromorphic hardware based on very-large-scale integration is a process that takes time. From the preliminary design, in terms of circuit schematics, to the final chip, built of silicon and metal, it may take months to years. A first prototype always has to be improved in later revisions, which may again take several months to years. One reason for this long-winded process is that the involved circuits are themselves outcome of scientific engineering research. The second reason is that, as stated above, simulations of complex systems, in this case systems of millions of active and passive electronic devices, are not feasible on normal computers. Therefore, full-chip simulations can not be done with reasonable effort. The process of the improvement of the VLSI ASIC[3], that is the central building block of the FACETS Stage 1 Hardware System, is the content of the thesis at hand.

Outline

After this introduction, the second chapter will introduce the FACETS Stage 1 Hardware System (FHW-1)[4]. Its core part is the *Spikey* ASIC which existed in its third revision in 2009. Chapter 2 will give an overview of the different parts of FHW-1, especially of the Spikey chip in this state. Chapter 3 will then describe the most important improvements that have been made on the Spikey chip. These resulted in the production of the fourth revision of the ASIC. Included in these improvements was the introduction of improved current monitoring devices for the parameter currents generated by the on-chip digital-to-analog converter by the author. However, not all the changes that have been made will be presented in this thesis. The interested reader may refer to [13] for a detailed listing of all the changes between Spikey's third and fourth revision. The fourth chapter outlines the experimental setup that was necessary to verify the basic functionality of the newly produced chip and to check the impact of the implemented improvements. The results of these measurements will then be presented in chapter 5.

[3] Application-specific Integrated Circuit

[4] FACETS Hardware Stage 1. This notation was introduced by Dr. Daniel Brüderle in his PhD thesis [4].

Chapter 2

The FACETS Stage1 Hardware System

This chapter will introduce the reader to the FACETS Stage1 hardware system which has been developed during the last six years within the FACETS project.

There is a popular computer book series called "...in a Nutshell", sending the reader on a journey from outside a subject to the very core of it, i.e. the most important information that defines the subject. Even though the core of a nut is mostly approached from outside the nut, by first taking care of the shell, we will introduce FHW-1 the other way around in this chapter. First of all, the purpose of the system will shortly be introduced, giving an idea of the mathematical background of the used neuron model. Afterwards, the implementation of the FACETS Stage1 Hardware System will be explained from inside to outside.

2.1 Purpose

The research in the field of neuroscience can be divided into two main branches. The first one of these is the bottom-up approach, consisting of biological experiments on the cell and network level. The output of this research serves as input to the second branch, the top-down approach, which involves modelling neural networks on different scales, both in time and space, i.e. on the cell and network level. The success of this latter task depends strongly on the choice and capabilities of the computational tools that are used because of its mathematical complexity. It is one aim of the FACETS project to supply the international neuroscientific community with such tools. These are implemented as neuromorphic hardware devices, i.e. electronic circuits that emulate the behaviour of biological neural nets. The FACETS hardware systems are implemented in terms of VLSI[1] circuits. Therefore, the size of the implemented circuits is measured in scales of μm. Two of the most important advantages of neuromorphic hardware of this type over software-based simulators, that are executed on general-purpose computers, are less power consumption and higher simulation speed.

The framework of which structure and recent improvements will be discussed in this thesis is called the *FACETS Stage 1 Hardware System*. Its structure will be explained in detail in the following section. Its core elements are highly accelerated (compared to biological timescales) and configurable neuronal and synaptic circuits which have been implemented as leaky integrate-and-fire neurons with synaptic circuits that feature two different types of plasticity. Schematics of the neuronal and synaptic circuits of the system are depicted in figure 2.1. The correlation between the parameters that are shown in this figure can also be described in terms of the differential equation 2.1. It describes the change of a neuron membrane potential with time, dependent, on the membrane capacitance C_m, the leakage Potential E_l, the inhibitory and excitatory potentials E_i and E_x and the conductances that control the current flowing from and

[1] Very-large-scale integration

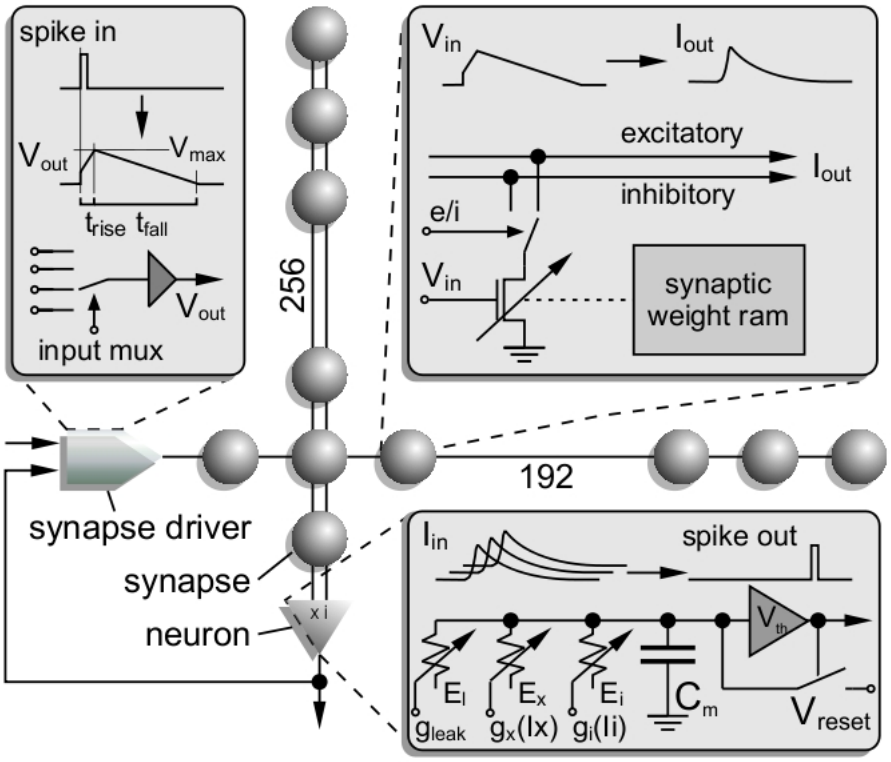

Figure 2.1: Operating principle of the spiking neural network (taken from [25]). The boxes contain schematics of the synapse drivers, the synapses and the neurons of the FHW-1.

to membrane potential.

$$-C_m \frac{dV}{dt} = g_l (V - E_l) + \sum_k p_k(t) g_k(t) (V - E_x) + \sum_j p_j(t) g_j(t) (V - E_i) \tag{2.1}$$

This conductance-based leaky integrate and fire model allows for the inclusion of plasticity mechanisms, i.e. changing synaptic behaviour with time, by varying the values of the inhibitory and excitatory conductances g_j and g_k with time. The values of the conductances $g_{j,k}$ are given by equation 2.2.

$$g_{j,k}(t) = w_{j,k}(t) \cdot g_{j,k}^{max}(t) \tag{2.2}$$

There are two implemented synaptic plasticity mechanisms on the chip: Spike Timing Dependent Plasticity (STDP), also referred to as Long Term Plasticity, and Short Term Plasticity, consisting of synaptic depression and facilitation.

2.2 Structure

2.2.1 The Spikey chip

In the following, the FACETS Stage 1 Hardware System will be referred to as FHW-1.x, where x labels the revision of the Spikey chip which is the core element of the system.

This main building block of FHW-1 is a mixed-signal VLSI neural network ASIC that is fabricated in a 180nm CMOS[2] process by *UMC*[3]. It has been developed by Schemmel[4] et al.

[2]Complementary Metal-Oxide-Semiconductor
[3]United Microelectronics Corporation, Taiwan
[4]Member of the Electronic Vision(s) group

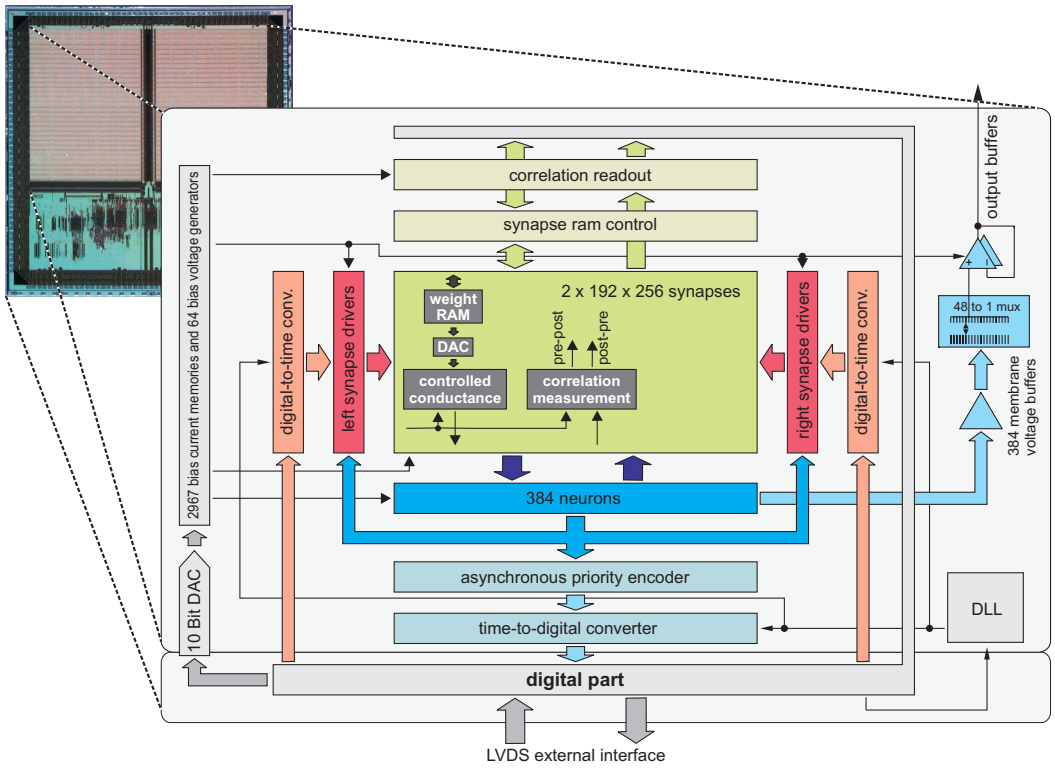

Figure 2.2: Spikey chip with focus on the analog part (taken from [14])

([27], [26], [25]) and Grübl[4] ([14]) from 2003 on and had been available in its third revision as of April 2008. During this thesis it has been improved by the aforementioned individuals and the author.

Spikey contains 384 leaky integrate-and-fire neurons and 512 synapse drivers, split into two blocks of equal size. A *synapse driver* is a circuit that generates action potentials, also called "spikes", which can then be redirected to the neuronal circuits via a synapse array. As can be seen in figure 2.1, the neurons are located at the bottom of a synaptic array, i.e. one neuron is placed at the end of every row of synapses. On the other hand, every row of synapses gets its input from a synapse driver. Every neuron of one block can thus be connected to every synapse driver in the same block via synapses with a digitally adjustable weight with a resolution of 4 bits.The conductance based synapses feature two plasticity mechanisms: Synaptic depression and facilitation and Spike Timing Dependent Plasticity (STDP), which will not be described in more detail.

Block diagrams of the complete chip are depicted in figures 2.2 and 2.3. Both figures show the same chip, only differing in that they emphasize different parts of it. The first one concentrates on the analog part which is a full custom design unit, while the second figure shows the details of the digital part.

The analog part comprises the neuronal and synaptic circuits which build a configurable spiking neural network. Most of the parameters that control the behaviour of the individual neurons and synapses are analog voltages between 0 and 1.8 volts or currents between 0 and 2.5 μA (one exception is the synaptic weight, that is encoded as a 4 bit digital value). These currents and analog voltages are digitally stored in a parameter RAM in the digital part of the chip. The current range of 0 A to 2.5 μA had been chosen as a result of a multi-objective trade-off between low power consumption and good accuracy amongst others. There are also circuits to monitor the adjustable parameterized voltages and currents contained in the analog part of the chip, some of which have been developed during this thesis. It was the main goal of this thesis to improve both, the current readout facilities of the Spikey chip and the voltage

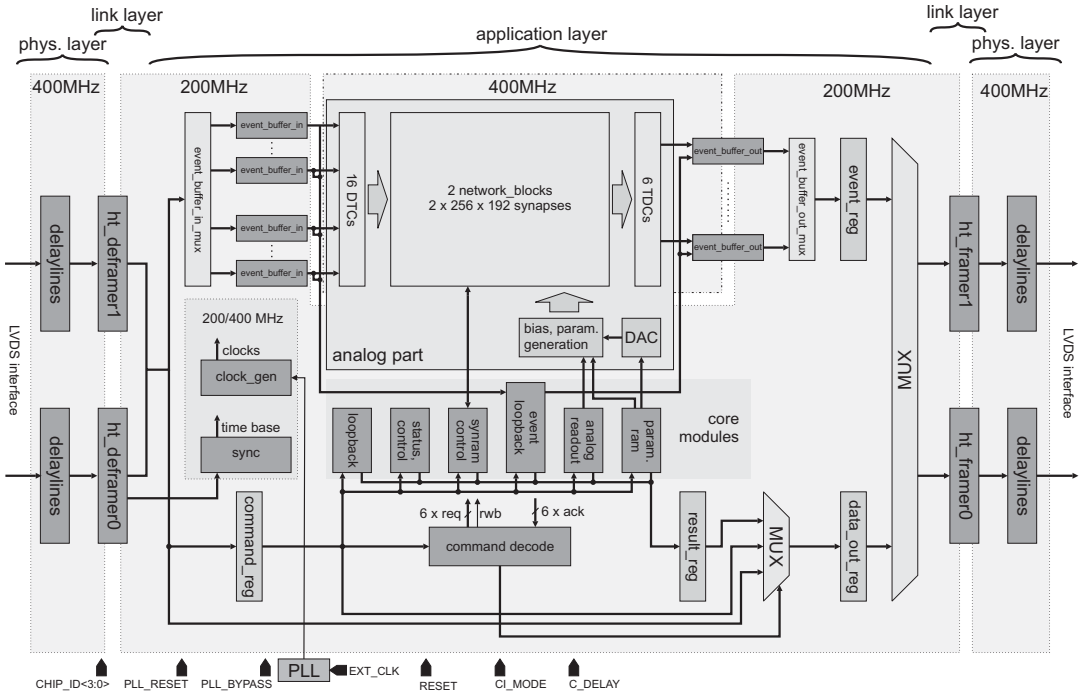

Figure 2.3: Spikey chip with focus on the digital part (taken from [14])

generating cells in the analog part of the chip.

In order to individually program the analog voltages and current parameters of the 384 neurons and 98304 synapses with minimal die area usage, an approach with only one current DAC (digital-to-analog-converter) and multiple analog current memories has been chosen. Most of the programmable parameter voltages and currents are used to adjust the behaviour of the individual neurons and synapses and the way the spikes are generated. Some of the parameters that are stored in the parameter RAM will be described in more detail in chapter 3.

When a neuron spikes, it generates a digital event, consisting of a timestamp and the neuron's number. These events can afterwards be transported off chip or fed back into the synaptic input circuits of the same chip. The digital part of the chip performs this task of spike transportation as well as all duty that has to do with chip control and configuration. It can be split up into the three following layers, of which the functional blocks are shown in figure 2.3.

The Physical layer converts the incoming data from the Low Voltage Differential Signal (LVDS) lines to the CMOS levels of the chip and vice versa. The differential signal transmission is implemented according to the HyperTransport[5] Standard which requires source-synchronous data transmission. Therefore, a clock signal is transmitted by every transmission unit along with the data signals. The Physical Layer also comprises configurable delay lines, both at the receive and at the transmit side of the chip, that allow the user to adjust the phases of the input lines of one bus relative to each other.

The signal transmission on the lines of a data bus between a sender and its receiving counterpart is subject to signal skew[5]. This is because of the different lengths of the connecting lines on the boards between the Nathan module and the network chip. To compensate for this effect in the Spikey chip, delay lines had been implemented on every signal line of the receive and the transmit side of the Physical layer. The delay of one line is adjustable in 8 steps. A schematic of such a delay circuit is shown in figure 2.4. Via a 3-bit delay selection line, one can activate one of seven tri-state buffers or deactivate all of them. In the diagram, they are denoted by numbers from 0 to 7. The 3-bit address decoder

[5]Time difference between to signal events that would ideally happen simultaneously.

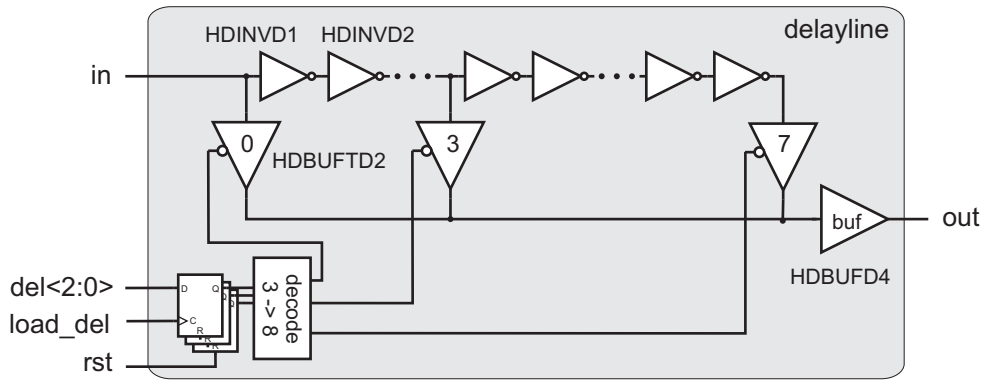

Figure 2.4: Schematic of a delay line as implemented in the physical layer (taken from [14])

is depicted in the lower left corner. The outputs of the buffers are connected to a common output net. Between two of such buffers are two inverters, which have been selected to have a summed delay of approximately 80*ps* in the typical process corner. The reason for using two identical inverters is to have the same delay for the rising and falling edge of the signal, which would not be given for a single inverter. These delay lines have been improved during the development of FHW-1.4, as will be explained in detail in chapter 3.

To maximize the *data valid window*[6] of the two input buses of the Spikey chip, the delay lines can be used to adjust the arrival time of the signal edges such that they arrive at the same time at the DDR input registers of the link layer. The clock phase relative to the signal phase at the output of the controlling programmable logic device can be adjusted additionally. This mechanism will be described in detail in section 5.1.

After the delay lines the double data rate (DDR) signals are split up to two registers of a width of nine bits each. Afterwards, the content of these registers can be read by the Link Layer with single data rate.

The Link layer decodes the packet information from the clock domain of the physical layer, operating at twice the clock speed, to the clock domain of the chip, operating at up to 200 Mhz, and vice versa. This process is called (de-)framing. To de-frame an incoming packet, the signal width is doubled while dividing the clock frequency by two. The Link Layer will then transmit the received data to the Application Layer. Analogously, the Link Layer converts data that is intended to be sent off chip in a process called framing. However, in order to be able to test the validity of the links that connect the chip and configure the delay elements of the Physical Layer, it also implements a bypass mode, called `CI_MODE`. In this mode the Link Layer bypasses the Application Layer while still de-framing the packet information received from the Physical Layer and forwarding it directly to the transmission unit.

The Application layer distinguishes between *event packets* and *control interface packets*. The former contain spike timing information that can be sent to the analog part, the latter control and configuration information. The Application Layer is connected to the analog network block that contains the actual neuronal and synaptic circuits and a DAC that generates the parametrized currents and voltages.

The functionality of the parameter RAM module of the digital part of the chip and the DAC in the analog part will be explained in the following because it is crucial for the understanding of some of the improvements in FHW-1.4 that were carried out by the

[6]Time during which a signal (bus) is in a stable logical state and can be sampled by a receive unit upon occurrence of a clock edge.

author and that will be explained in chapter 3. A schematic of the mentioned modules is depicted in figure 2.5. The main task of the parameter RAM module is to refresh every programmed current value in a periodic way, such that the chip can work with only one DAC while keeping the current values in the current memory cells constant over time. These current memory cells are used to allow different parameters for every synapse driver and neuron circuit.

Another important task of the Application Layer is readout and packing of spike event data. After a spike has been digitized by the Time-to-digital-converter (TDC) it is buffered in the *event_buffer_out* module acting as a FIFO that stores digital spike events. Up to three of these events can be combined to a 64-bit event packet by the subsequent *event_buffer_out_mux* module. These event packets will then be sent off the chip to the controlling programmable logic device, that will be explained in the next subsection.

The following paragraphs will introduce the details of the parameter voltage and current generation on the chip. This is an important feature because it makes the chip configurable and versatile and allows to implement neurons and synapses with different behaviours with one hardware system. The core part of the parameter voltage and current generation block is a 10-bit DAC that receives its reference current I_{refdac} from outside the chip and its digital input signal from the parameter RAM controller. The reference current is generated on the Recha board (as explained in the next subsection). The parameter RAM is shown in the *core modules* block of figure 2.3. This digital module controls the DAC input and determines which one of the 2967 current memory cells on the chip will be activated and act as a sink (or source) to the current controlled by the DAC. To this end, the parameter RAM module has a 12-bit address output. Since there exist both, current memory sources and sinks on the chip, the DAC has two current output terminals, one for each type of current memory. Beyond these two normal current outputs, there is one current output that is able to source up to the decouple of the input current I_{refdac}. For simplicity, on the schematic shown in figure 2.5 only this output and the current source output is shown, the current sink terminal is omitted.

There are also two arrays of additional current memories. One containing nine current memory sources and one containing nine sinks. Both arrays are connected to the according current output terminals of the DAC, and can be activated by the parameter RAM module in order to be able to apply the tenfold value at the DAC input, resulting in the normal current value (before multiplication) from a point of view of the target current memory cell. The configuration can also be seen in figure 2.5. This feature has been implemented to avoid large steps at the DAC output. When a large input value was applied to the DAC's input and a very small value shall be output by the DAC afterwards, this change in the output current would take several tens of microseconds. To prevent the DAC from having to switch the capacitive load at its output from a large current value (during the write process of n-th parameter value) to a very small value (during the write period of the (n+1)-th parameter value), the additional current memories can be activated so that the DAC input can be kept at a large value that is scaled down afterwards. This feature is called *boost*.

The parameter RAM module can make use of 16 look-up-table (LUT) entries. These contain the current write timing information and can be referred to by the parameter information. Each of the LUT entries holds 4 pieces of information:

1. Number of clock cycles during which the programmed current value is applied, stored as exponent of two.

2. Number of clock cycles during which the boost switch is activated, stored as exponent of two.

3. Number of repetitions of the write times in this LUT with the same DAC input value.

4. Step size of current memory address increment between two successive repetitions.

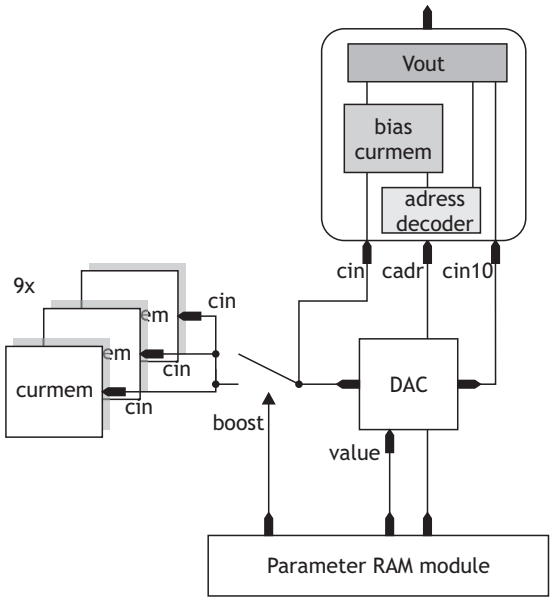

Figure 2.5: Digital-to-analog converter with controlling digital block and example target parameter voltage cell

With these LUT entries the user can control which current value will be applied for how long to the according current memory cell. If in the same LUT entry the normal write time and the boost write time value are greater than zero, the programmed DAC input value will be multiplied by a factor of ten during the boost period and afterwards the normal 10-bit value will be setup during the normal write time.

If, however, the normal write time exponent is set to zero, while the boost write time exponent is greater than zero, the digital control module applies the programmed current value to the DAC without multiplying it by a factor of ten while activating the nine supplementary current memory cells, thus effectively dividing the current value applied at the DAC input by a factor of ten.

After the chip has been configured by the controlling host PC, the voltages and currents are stored in a single ported static memory cell by the parameter RAM module. For the reasons just mentioned, one programmable current actually involves three pieces of information: a 10-bit value applied to the DAC, a 12-bit address value determining which current memory cell to enable and a 4-bit Look-up-table address.

Apart from the current memory cells, there are also two voltage generating blocks, called *curmem_vout_block*, containing 24 *vout* cells each. Their functionality will be explained in the following paragraph and can be seen in figures 2.5 and 2.6. Theses cells can be programmed by the parameter RAM module, too. The *cin* input pin is connected to the normal current source output of the DAC. The current applied to this terminal will be saved in the current memory cell included in each *vout* cell. This current serves as bias current for the operational amplifier (op-amp) buffering the output voltage. The other current input terminal, called *cin10* generates a voltage via a $10k\Omega$ resistor, which is then saved on a capacitance. This is the actual voltage that serves as a parameter to the neuronal and synaptic circuits. Switch S_1 is controlled by an address decoder that is controlled by the address bus of the parameter RAM module. This implies that only one *vout* cell at a time is enabled. During one write procedure to a *vout* cell, switch S_2 will be closed first. After the write time is finished, switches S_1 and S_2 will be closed while S_1 will be opened, such that the voltages on C_3 and C_2 can adjust.

For monitoring purposes, a separate analog output pin, called `IBTEST`, is available in the Spikey chip up to FHW-1.3. Internally, it is connected to the so called *analog readout chains* of each *curmem_vout_block* and to one of the current memories in the so called *outamp_block*. An

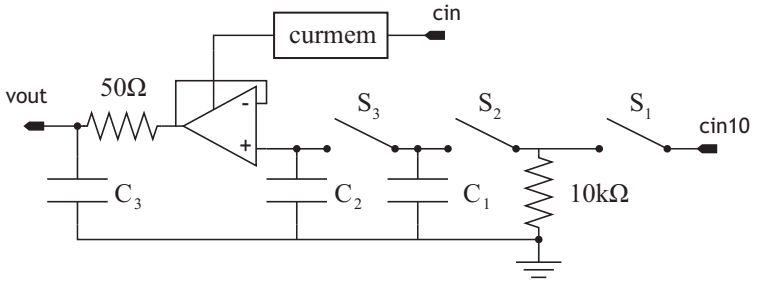

Figure 2.6: Schematic of the configurable, voltage generating cells in FHW-1.4

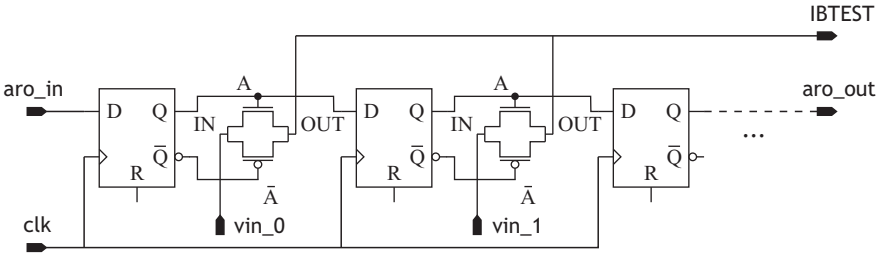

Figure 2.7: Schematic of the analog readout chain

analog readout chain consists of a set of flip-flops connected to build a 1-bit shift register, while each one of these flip-flops is connected to a transmission gate that controls the connection of one *vout* cell to the output pin `IBTEST`. This way, only one cell at a time can be connected to the output pin if the digital analog readout chain controller feeds the chain input accordingly. A schematic of this shift register can be seen in figure 2.7.

The digital module controlling the analog readout chain consists of a 24 bit register that can be accessed by the controlling software. After having set this register to a value containing only one single logic high value and having set which of the two analog readout chains to program the module consecutively applies the bits stored in its register to the input of the analog readout chain.

The other cell that is connected to the `IBTEST` output net in FHW-1.3 is a current memory that belongs to the *outamp_block*. This block contains nine separately addressable current memory cells, eight of which supply the bias currents for the operational amplifiers that buffer neuron membrane voltages of a neuron, which can be selected by the user.

The ninth current memory cell had been implemented for testing purposes only. The changes that have been performed in this block will also be described in chapter 3.

2.2.2 Supporting hardware

The Recha Board is a printed circuit board (PCB) on which the Spikey chip is directly bonded. It serves as a carrier board for the chip and supplies voltages and currents that are necessary for its operation. The current version of this board is a conjoint work of Dr. Andreas Grübl and Boris Ostendorf[7] [20]. It also features the following support circuits:

- A four-channel 10 bit serial DAC that generates the reference voltage V_{casdac} for the DAC inside the Spikey chip, as well as the parameter voltages V_m, V_{REST} and V_{START}. It is controlled by the FPGA on the Nathan module which will be introduced in the next subsection.

[7]Former member of the Electronic Vision(s) group

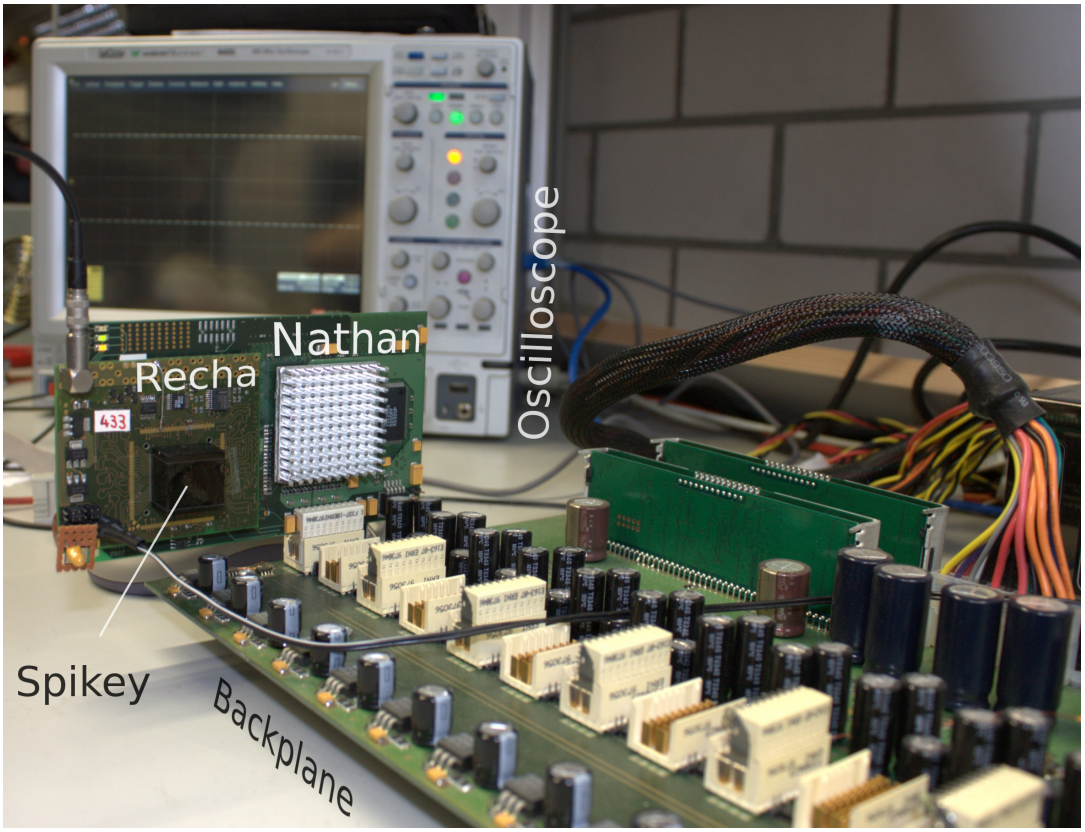

Figure 2.8: Photograph of the experimental setup, showing Backplane, Nathan module with mounted Recha PCB, Spikey chip and oscilloscope in background.

- A current source that generates the reference current for the DAC inside the Spikey chip I_{refdac}. It receives its control voltage from a DAC on the Nathan PCB.
- Nine lemo jacks that can be connected to an oscilloscope. Eight of these are hard-wired to the eight neuron membrane potential pins of the Spikey chip.
- A multiplexer that selects which one of the 9 analog voltage output pins of Spikey will be multiplexed to the first of these lemo jacks, that is mounted in one corner of the Recha board. Since this multiplexer can easily be controlled by the FPGA on the Nathan board, and thus by the host PC, all the analog output pins of the Spikey chip can be monitored by using only one single oscilloscope channel.
- A four-channel 10 bit serial analog-to-digital-converter ADC, to enable the software framework to directly read back the voltage that has been selected by the multiplexer. This way, one can read parameter voltages and currents both, digitally via the ADC and the software framework and directly via an oscilloscope connected to the test pin.

The Nathan Module was designed as controller of the Analog Neural Network (ANN) ASIC that can be hosted on the Nathan board and as a connector module for the distributed operation of mixed-signal Analog Neural Network ASICs. The main building blocks of a Nathan module are a printed circuit board with an FPGA module and a connection socket for an ANN ASIC. In the case of FHW-1 a Recha board containing a Spikey chip is mounted on this socket. It is possible to setup large scale neural networks consisting of up to 16 Spikey chips per backplane (c.f. following paragraphs). These chips communicate via the FPGA on the Nathan module by exchanging digital events over high-speed serial links. The main tasks of the programmable logic device on the Nathan module are:

Slow-Control This module allows for real-time access to the chip via the modules *spikey-sei* and *spikey control* and for access to the DDR-SDRAM attached to the FPGA, as can be seen in figure 2.9. If the user wants to send packets to the chip with a specified timing, i.e. with a controlled delay between the packets, this can only be realized with the so called *Playback memory* feature.

Playback memory Packets that have been saved in the FPGA's RAM modules can be sent to the chip in the programmed order and with specified delay values. Therefore, this mode of operation can guarantee a well-defined spike input rate to the Spikey chip.

Other tasks of this module are:

- The operation of the digital-to-analog-converters that generate some of the control voltages for the chip.
- The readout of the analog-to-digital-converter on the Recha board.
- A temperature control mechanism

The Backplane serves as connection device and physical support. It establishes physical connections between up to 16 Nathan Modules, that can be plugged onto the backplane. A controlling host PC can be connected via an SCSI cable and a PCI card or, based on work lately accomplished in the Electronic Vision(s) group, via a Gigabit Ethernet link. A programmable logic device on the backplane controls the communication between the host PC and the Nathan modules. A Multi-Class Gigabit network architecture (c.f. [21]) that connects the Nathan Modules, implemented using the multi-gigabit transceivers of the FPGAs, is used for the exchange of spike events (c.f. [8]). For the work that has been carried out during this thesis, the SCSI connection via the PCI card in the host PC was used exclusively because the Ethernet link software and hardware implementation had not been available with sufficient stability at the beginning of the thesis. The connection

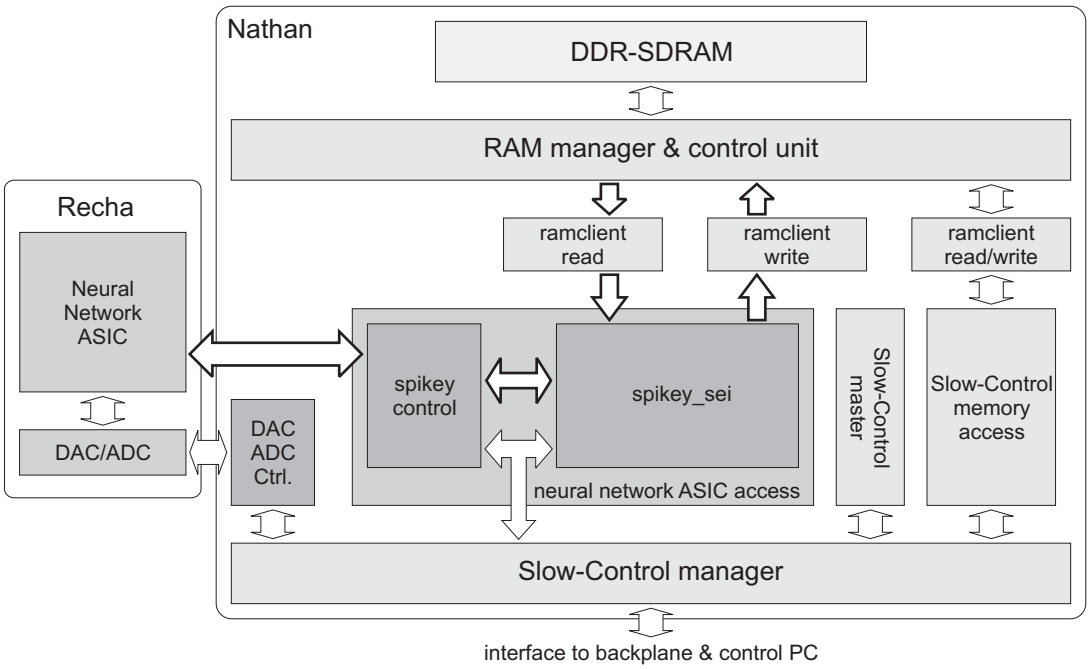

Figure 2.9: Block diagram of the functional modules of the Nathan and the Recha modules (taken from [14]) as used for single-chip networks.

between the individual Nathan modules and the controlling host PC is established via a token ring protocol.

Apart from its role as mere physical support and power supply, the backplane also provides the clock signals for the Nathan and Recha boards, i.e. for the Nathan FPGA and the Spikey ASIC and hosts a programmable logic device for connection purposes. The clock signals can be chosen to originate from either two different internal clock sources or an external clock source, that can be inserted via a LEMO jack or an S-ATA cable. The latter could also be used to supply two backplanes with identical clock signals. Because the internal clock sources operate at fixed clock frequencies of 100 MHz and 156.25 MHz respectively, these frequencies will be investigated in more detail in chapter 5. Since the Physical Layer of the Spikey chip operates at double data rate, these clock frequencies result in a clock frequency of the link layer of 200 MHz and 312.5 MHz respectively.

2.2.3 The Control PC and its software

The controlling host PC is connected to the backplane via an SCSI cable and a PCI card. The hardware access is encapsulated in the *Darkwing server* (*dwserver*) software framework that allows simultaneous access of multiple software clients to the same backplane.

Using the *dwserver* interface to the hardware, a software framework allows for control of the Nathan FPGA and the Spikey chip on different levels of abstraction. Enabling the user on the one hand to configure the Spikey chip completely, i.e. all of the 2967 current and voltage parameters and all synaptic weights, with one single configuration file while on the other hand giving register level access to the chip. Analogously, it gives the user the choice to send single spike event packets or to send complete sequences of spikes (*spiketrains*). The software structure and improvements that have been performed in the past 12 months will be explained in more detail in chapter 4.

Chapter 3

Improvements in the hardware design

As mentioned earlier, the first revision of the Spikey chip (FHW-1.1) has been finished in 2003. Since then, many improvements had to be made, which is common practice in the development process of a mixed-signal ASIC, as can be seen in figure 3.1 ([9]). It shows the typical steps of an Integrated Circuit (IC) design process. The work that has been done for this thesis in the past year has been a walk along this line of IC development. Before the design of FHW-1.4 the Spikey chip was in the state of "Test & Evaluation". Because of the shortcomings of the chip that have been found in the past years, changes had to be made both, to the "Preliminary Design" and the "Layout". "Preliminary Design" means circuit schematics in the case of analog devices or source code in a Hardware Description language in the case of digital devices. The "Layout" step is the actual implementation of these devices in terms of MOS transistors or passive electrical components.

For example, the modifications that needed to be done to add better current monitoring devices to the analog part of the chip however, also involved changes in the circuit schematics and simulations thereof. Not all the changes between FHW-1.3 and the current version of the hardware system have been done by the author, therefore the responsible person(s) will always be named together with the changes themselves.

In this chapter, the improvements in chip design are grouped by logical design units on the chip. Some of the changes not carried out by the author will be mentioned for completeness and for reference to give an overview of the improvements between FHW-1.3 and FHW-1.4.

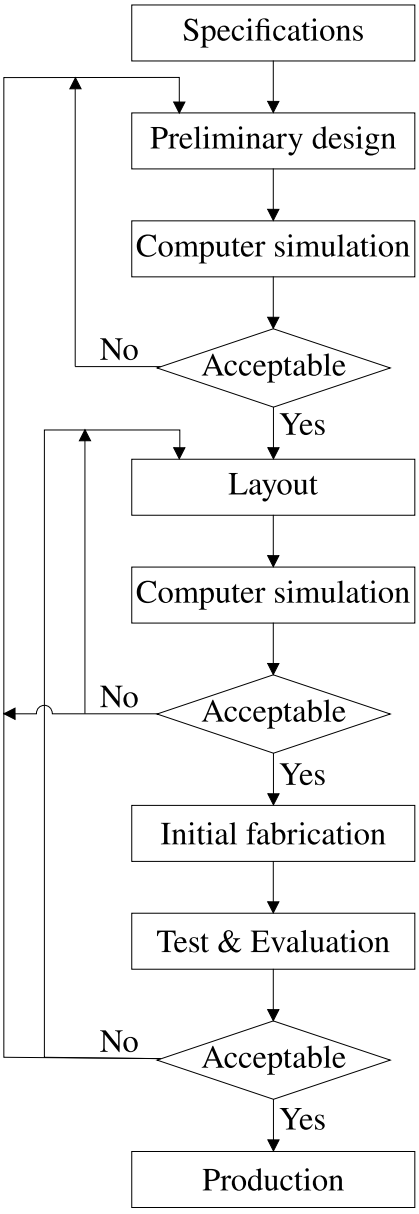

Figure 3.1: VLSI chip design workflow, as shown in [9], page 14

Some of the shortcomings of a mixed-signal VLSI ASIC are discovered only when the chip has been produced and can be tested in the lab. This is grounded in the fact that, due to the high integration density, the chip's design cannot be simulated in reasonable time. A simulation of that kind would only make sens when it would include a complete netlist of parasitic[1] circuit elements, making the netlist to be simulated too large for the available simulation software and hardware. This problem will be discussed in detail in appendix A.

3.1 The digital part of the Spikey chip

The changes to the hardware that are described in this section have been carried out by Dr. Andreas Grübl. The most important and fundamental improvement was the conversion of the standard cells library provided by *Virtual Silicon Technologies* (VST) to a standard cell library provided by *Faraday*. This change was necessary because *UMC* had cut the support for the VST library. Only due to a special permission, it was possible to keep the IO ring cells of the VST library, because the according Faraday cells are not compatible.

Furthermore, the *Verilog*[2] description of the digital part has been improved to shorten the data paths on the chip, which was also done by changes in the chip's floorplan.

Another improvement concerns the power distribution in the digital part. The voltage drop of the power lines has been analyzed and improved with a tool called *VoltageStorm* by *Cadence Design Systems* that can be included in the *SoC Encounter* workflow. The procedure is called IR-drop analysis, and makes use of the power consumption information that is supplied together with the standard cell library. It involves the following steps: At first a simulation of the digital part is done that has to simulate a realistic worst-case scenario. From this simulation, an activity profile is generated containing information about the usage of the different states of each standard cell. From this activity information and the power consumption values for different states of a cell, a voltage drop profile for the chip's floorplan can be generated, which is then symbolized in terms of a color map across the chip's floorplan. As a result of this analysis, some of the power distribution nets could be widened selectively, thus guaranteeing a reliable power distribution across the digital part of Spikey.

3.1.1 Improvements in the Physical Layer

As an improvement for FHW-1.4, the delay lines have been placed manually on the chip such that the time delay of a signal line has a linear dependence on the number of activated delay elements. To this end, the signal lines between two subsequent delay elements of one delay line have been implemented equidistant. This placement had been done automatically by the *Cadence Encounter* software for FHW-1.3 and has not yielded such linear dependence due to the random nature of the auto-placement algorithm. The improvements can be seen in figure 3.2 which was exported from the digital implementation and analysis tool *Cadence SoC Encounter*. Furthermore, the two different inverters that made up a delay step on FHW-1.3 have been substituted by two instances of one single inverter from the standard cell library to make sure that the rising and falling edges of the delayed signal experience the same amount of delay.

3.1.2 Improvements in the Application Layer

The depth of the `event_buffer_out` FIFO[3]s in the Application layer (c.f. figure 2.3), which buffer the outgoing events from the network block, has been made configurable to range from 0 to 127. In FHW-1.3 this FIFO had a depth of 127 events, but, as Simon Friedmann has shown in his diploma thesis [8], this value is too large for multi-chip operation. The time it takes for

[1] Circuit elements that are not required for proper functionality and who's existence is not intended by the designer

[2] Hardware description language by Cadence Design Systems

[3] First in first out

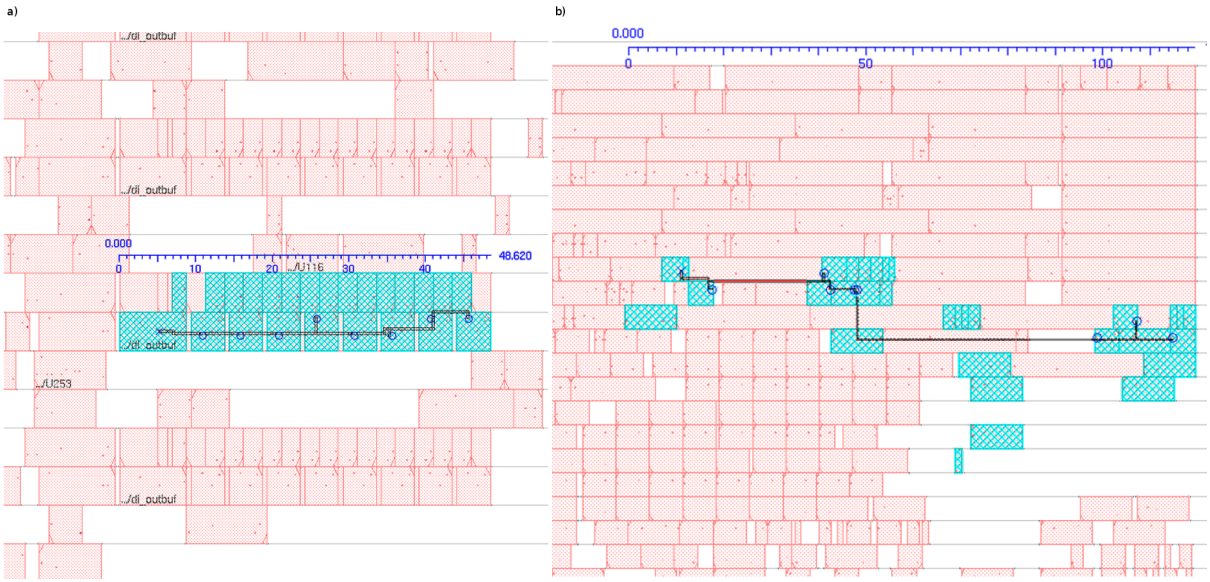

Figure 3.2: Placement of input delay line #8 of the CAD1 input bus. The cells that belong to this delay line are shown in darker tones. Distances in μm. a) Placement of delay line elements in FHW-1.4 is identical and compact for every delay line, b) Placement of delay line elements in FHW-1.3 was done with the automatic placement algorithm of the *Cadence Encounter* software suite.

an event to travel through a FIFO of that depth would unnecessarily increase the latency of the event packets. To this end the so called *control register* on the Spikey chip had to be augmented by 7 bits, thereby making necessary some minor changes to the controlling software, which have been carried out by the author.

3.1.3 Errors during changes in the design

As will be explained in section 3.2.3, the *curmem_vout_block* had to be augmented by one cell. Thus, the analog readout chain, consisting of the serial interconnection of the flip-flops in the vout cells, has also been incremented by one. Unfortunately, the size of the input register of the analog readout controller in the digital part was not incremented by one. Therefore a workaround had to be figured out by the author, to be able to set all values in the analog readout chain correctly. This workaround will be presented in the next chapter.

Also, a mis-assignment of the current memory addresses of the left block by the parameter RAM controller has accidentally been introduced in FHW-1.4. To provide a sufficient signal slew rate between the parameter RAM module and the current memory cell's address inputs, the number of the address registers was doubled. This has become necessary because the driver strengths of the used standard cells of the *VST* library were more than a factor of two stronger than those of the *Faraday* library.

In FHW-1.3, only 12 drivers where used to set the 12-bit address bus that was connected to all the current memory cells in both chip blocks. In FHW-1.4, the current memory address register was doubled in order to supply each half of the chip with its own 12-bit current memory address bus. In *Verilog*, this was implemented using the concatenation operator provided by the language. It allows to duplicate a single bit vector by appending its value to its end, thus yielding a register of twice the size containing the same string of bits twice. However, the bit string that results from the address calculation by a simple subtraction of a 32-bit integer has a width of 32 bits, of which only the lower 12 bits contain valid information. In the previous version of the Spikey chip this has not been a problem, since it was assigned to a 12-bit bus

which simply didn't take care of the upper 20 bits. Thus, the concatenation resulted in a 64 bit wide address register that, in the current revision of the chip, is assigned to a 24 bit wide bus. This means that only 12 bits of this bus obtain a valid address information while the other half of the bus reads nothing but zeros. As mentioned in the introduction of this chapter, a mixed-signal simulation, i.e. a simulation of the digital part and the complete analog part, was not possible due to insufficient computational resources. Therefore the above described problem was not discovered during simulations. It can also be considered a shortcoming of the *Verilog* HDL that registers of different sizes can be connected to each other without resulting in a warning by the compiler.

This problem has already been fixed in the code, so that a potential fifth revision of the Spikey chip will not suffer from this problem anymore. For the current work with FHW-1.4 (including the thesis at hand) the left block of the chip cannot be used for experiments because its voltage and current parameters can't be controlled at all.

3.2 The analog part of the Spikey chip

3.2.1 Improving the neuron readout priority encoder

FHW-1.3 has imperfections which are partially based on parasitic effects. One of these was a malfunction in the neuron readout priority encoder. This problem has already been located by Dr. Daniel Brüderle [4] and Johannes Bill [1].

- "The current version of the FACETS Stage 1 Hardware (Spikey III) reveals errors, when multiple neurons of the same block of 64 neurons are recorded and emit spikes simultaneously: The priority encoder can enter a locking state and stops further spike output."

  From: [1], page 23

- "When recording from more than the last n_{rec} neurons of a neuron block, output spikes from different neurons that are generated temporally close can result in a total spike readout deadlock, i.e. no spike is recorded from this block anymore. [...] For all FHW-1 systems, the critical number n_{rec} has been observed to vary from 5 to 9 [...]"

  From: [4], page 9

The spike readout deadlock problem is also illustrated in figure 3.3. It shows that, when two neurons from the critical block of neurons are chosen and a deadlock occurs, the observed neuron continues to emit spikes, but the event readout priority encoder prevents the neuron from sending spike information to a Time-To-Digital Converter (TDC).

The digital spike readout is a crucial feature of the chip. Since the digitization of the neuronal spikes is done by one single TDC per 64 neurons it has to be assured that only one spike at a time activates the TDC input, because otherwise one would not be able to distinguish two or more successive spikes from different neurons of the same block. Therefore, a neuron readout priority encoder has been implemented by Dr. Johannes Schemmel in order to process simultaneous action potentials in direct succession. This circuit sends an inhibition signal to each neuron's output circuit, telling it to hold its last spike if the action potential of a neuron of higher priority is still being processed. The readout priority is determined reciprocal to the neuron number which is arranged in ascending order from inside to outside in both blocks of the chip.

The "hold" signal is implemented as a latch, i.e. a circuit similar to a flip-flop that has no clock signal and is only input-driven, in each neuron's readout circuitry. When a spike is triggered the latch output is set to a logical high value and it will only be reset when the inhibition signal has become low, i.e. no neurons of higher priority are being processed, and when the TDC has affirmed the processing of the held action potential by raising a clear signal.

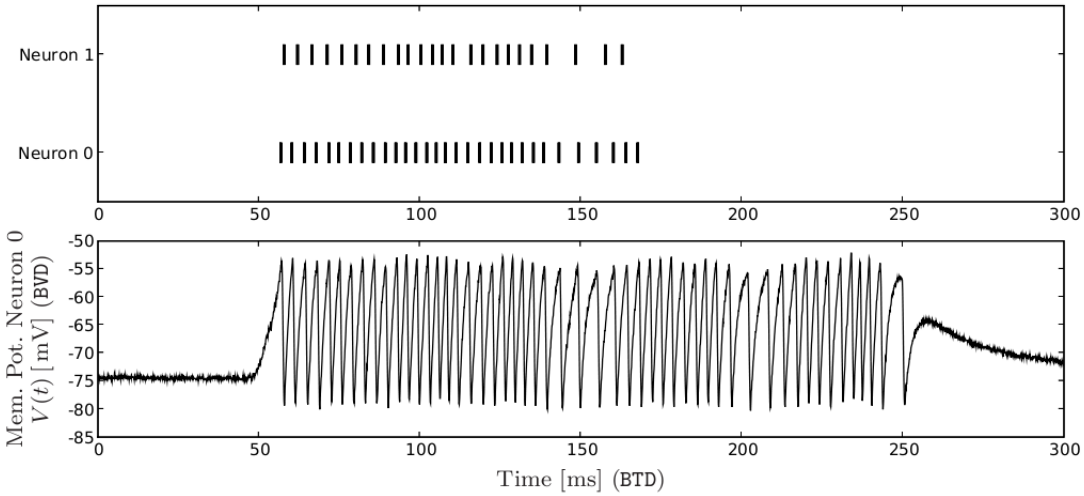

Figure 3.3: Rasterplot and membrane potential showing the deadlock in the digital event readout priority encoder. The lower graph shows the membrane potential of Neuron 0, indicating that this neuron should still emit spike events at times larger than 170 ms biological time. Taken from [4], page 98.

After the neuron readout latch has been cleared, another action potential can be received and held by this particular neuron's readout circuit.

The clear signal could be identified as the source of the problem described above. Because of the large capacitive load that has to be driven by an inverter inside the TDCs, the signal does not reach the logical high value threshold during the time of the digital pulse at the input of the neuron readout circuits. Therefore, the circuits are not cleared and thus locked after having held an action potential until the next chip reset. Only the first 5 to 9 neurons[4] of each block can be used without invoking a readout deadlock, because the signal lines of the clear signal to these neurons has a lower capacitance compared to the neurons with higher indices. The only workaround that exists to bypass this problem in FHW-1.3 is to send a neuron reset signal every time a neuron deadlock occurs. This signal resets the neuron readout circuitry of every neuron, forcing it to reset the event holding latch. The problem has now been solved by adding a signal buffer for every block of 8 neurons.

The reason why this effect has not been discovered in simulations after the "Preliminary Design" step, is that it is caused by parasitic effects. These parasitics are circuit elements that are not required for the circuits proper functionality nor is their existence intended by the designer. Their presence however, is unavoidable and it does affect the proper functionality of the circuit. Parasitics can appear as capacitances, resistances, inductors and even as unwanted bipolar transistors in a CMOS process, when n-wells or p-wells are involved. In the above mentioned case, the capacitances and resistances of the signal lines had to be taken into account to determine the signal levels at the neuron readout circuit's input to proof the changes that have been done to solve the problem. For the extraction of the resistive and capacitive parasitics, the *Calibre* IC verification software by *Mentor Graphics* was used by the author. The extracted parasitic values have then been inserted into a testbench that was created by the designer.

The work-flow of this software is described in Appendix A. It explains how the parasitic circuit elements are extracted out of a GDS II[5] file by the software. These parasitic elements are not contained in the simulation models that are supplied by the foundries, because they depend on the actual spacial relations of the devices to each other.

[4] According to [4], page 96

[5] A standard file format for IC layouts that is supported by many IC design software suites and IC foundries.

After the parasitics extraction had been carried out successfully, the testbench circuit was simulated using the *Cadence UltraSim Full-Chip* simulator, to verify the proper functionality of the improved priority encoder in all process corners.

3.2.2 Improving parameter voltage generation

The parameter voltage generation in FHW-1 has always been a problem for many experiments, especially those involving synaptic plasticity.In his diploma thesis, Johannes Bill showed that for a decent operation of the STP mechanism on Spikey, it is crucial to be able to configure the necessary voltages to values below approximately 0.6 V ([1]):

> "More restricted are the internally generated voltages vouts: The current revision (Spikey III) allows for a vout-range from 0.6 V to 1.6 V. The exact interval is determined by the individual vout. Especially, the lower bound prevents reasonable STP-parameters as revealed in section IV.5."

The reason for this behaviour was that the dynamic range of the operational amplifier in the voltage generating cells (called `curmem_vout` cells) of FHW-1.1 to FHW-1.3 was limited to approximately 0.6V to 1.6V. However, Sebastian Millner[6] had designed an op-amp with "rail-to-rail input and output voltage range, low quiescent current, 1.8 V power, a high bandwidth and unlimited stability" in his diploma thesis [18] for the FACETS stage 2 hardware.

During this thesis this improved operational amplifier was inserted into the `curmem_vout` cells of the Spikey chip, by including the layout of the op-amp into in the layout of the voltage generating cells. Afterwards, simulations had to be done to verify the behaviour of the new cells. Because the operational amplifier had to be simulated in a realistic context, a circuit of a capacitor in parallel to a series of a resistor and another capacitor was used. The capacitances have been chosen to 1 pF and the resistance has been chosen to 200 Ω. These were connected between the output of the op-amp and a pulsed voltage source, acting as a load. The results can be seen in figures 3.4, 3.5 and 3.6. The figures show an AC simulation of the new op-amp in a unity gain buffer configuration, for different common mode voltages. It can be seen that the old amplifier has a dynamic range of 0.7V to 1.6V. At 0.6V and 1.7V the amplifier is not capable anymore of supplying a unity gain over a sufficient frequency range. The new op-amp however, is able to buffer voltages ranging from 0.1V to 1.7V with a frequency of up to 10MHz. The experimental verification of this improvement will be discussed in section 5.2.

3.2.3 Adding current monitoring devices

In FHW-1.3 and its predecessors, the only possibility to directly monitor the parameter current generation was given by measuring the current flowing out of a current memory cell at the top of the chip. Since the measured currents were in the order of magnitude of $10^{-3}\mu A$ to $2.5\mu A$, especially low currents could not be measured with sufficient accuracy. This is due to the fact that $0.01\mu A$ result in a voltage drop of merely 5 mV with the included measurement resistance of 500 kΩ on the *Recha* board (c.f. section 2.2.2). Voltages in this order of magnitude cannot be measured with sufficient accuracy considering the existing noise background. Besides, the currents that need to be measured are biased by the input bias current of the op-amp on the Reach PCB that buffers the output voltage at the 500 kΩ resistance and by the current flowing through the ESD[7] diodes. This current however, lies between -10pA and 10pA in all possible process corners as obtained from simulations of the analog output pad used in the Spikey chip and is therefore 2 orders of magnitude smaller than the currents that were to be measured.

More reliable current monitoring devices have thus been developed by the author. To this end, two different implementations have been realized, they are shown in figure 3.7. The first

[6] Member of the Electronic Vision(s) group

[7] Electrostatic discharge

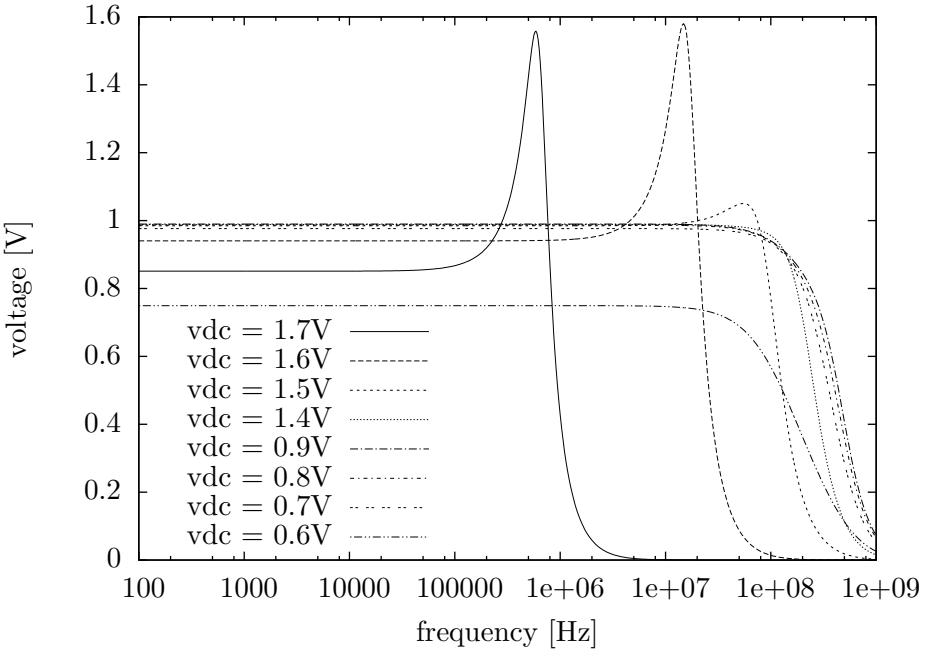

Figure 3.4: AC analysis of op-amp in FHW-1.3 for different common mode voltages from 0.7V to 1.7V. The common mode voltages from 1.0 V to 1.3 V are not shown because their shape is very similar to that of the 0.9 V line. It can be seen that a unity gain can only be supplied for frequencies up to 10 MHz for common mode voltages between 0.7 V and 1.5 V.

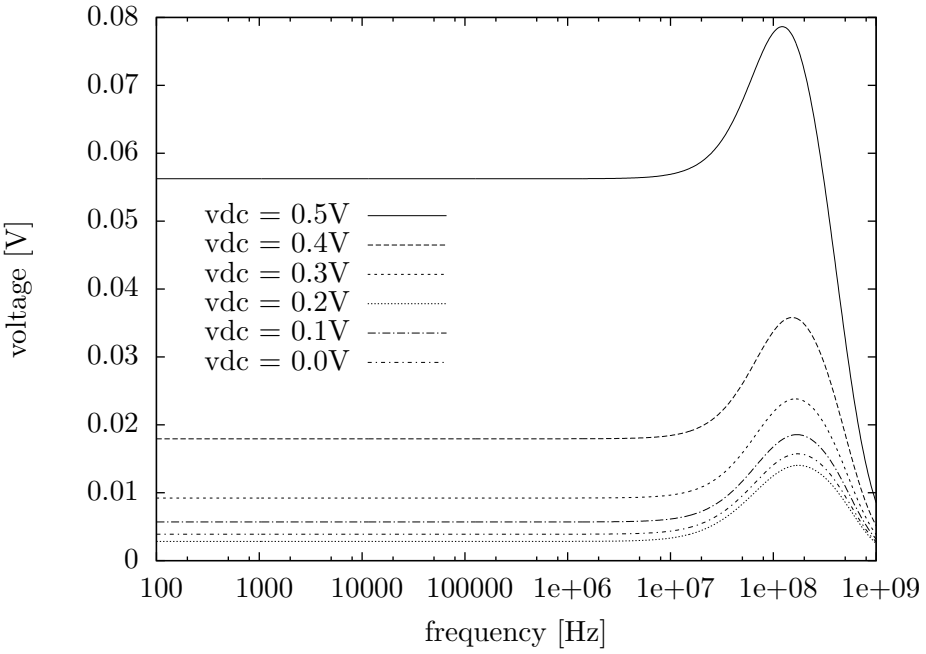

Figure 3.5: AC analysis of op-amp in FHW-1.3 for different common mode voltages from 0.1V to 0.6V. For these common mode voltages a gain of one cannot be supplied, independent of the frequency.

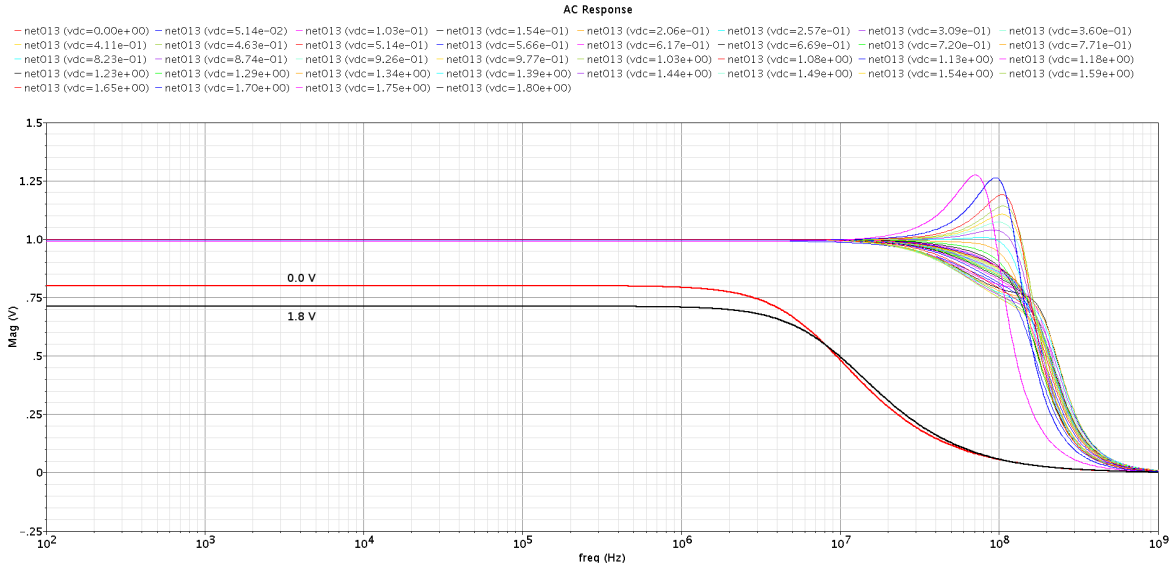

Figure 3.6: AC analysis of op-amp in FHW-1.4's parameter voltage cells for different common mode voltages from 0.0V to 1.8V in steps of 0.05V. It can be seen, that a unity gain can be supplied for frequencies up to 10 MHz for common mode voltages from 0.05 V to 1.75 V.

implementation is a current mirror connected to a current memory cell at the top of the die, it is sketched in the left part of the schematic. A second implementation that allows to measure the current indirectly via a measurement resistance and a *curmem_vout* cell, is shown in the right part of the schematic. The objective of this improvement was to be able to measure the currents generated by the on-chip DAC (introduced in section 2.2.1). The following paragraphs will describe these implementations in more detail.

As mentioned above, in FHW-1.3 there was only one possibility to measure the currents generated by the DAC via a dedicated current memory cell, located in the *outamp_block* at the top of the die area. Since currents between a few nA and $2.5\mu A$ are to small to be directly measured, a current mirror with an amplification factor of 64 was designed by the author. This current mirror was then connected to the output of the current memory cell, thus sourcing a much larger current to the `IBTEST` pin of the Spikey chip. A schematic can be seen in the upper left corner of figure 3.7. It consists of an NMOS current mirror and a PMOS current mirror, both multiplying the input current by a factor of 8. This multiplication factor was chosen because it could best be implemented using the common-centroid layout. This layout technique compensates for mismatch due to gradients in the doping concentration, because it arranges devices symmetrically. Actually, every other type of inhomogenities, like for example thermal or stress induced gradients, that is linear in distance can in principle be compensated for with this layout, even non-linear gradients can be counterbalanced if the size of the layout is small compared to the distances affected by the gradient, a principle well known as linearization. The book "The art of analog layout" by A. Hastings [15] was very helpful on this subject during the design phase of this current mirror.

The device can further be divided into two parts, an NMOS input stage that serves as a current sink controlling the current flowing through a PMOS output stage, which, on his part, sources a current to the `IBTEST` pin. In the layout, both stages consist of two identical transistor devices with a length of $400nm$ and a width of $1\mu m$. The two compound transistors of one stage have 6 and 48 fingers, respectively, which results in a current amplification by a factor of eight. Furthermore, both stages have two axes of symmetry, due to the common-centroid configuration.

Simulations have been carried out to verify the functionality of the current mirror. Figure 3.8 shows a corner analysis that has been carried out with the designed current mirror. The

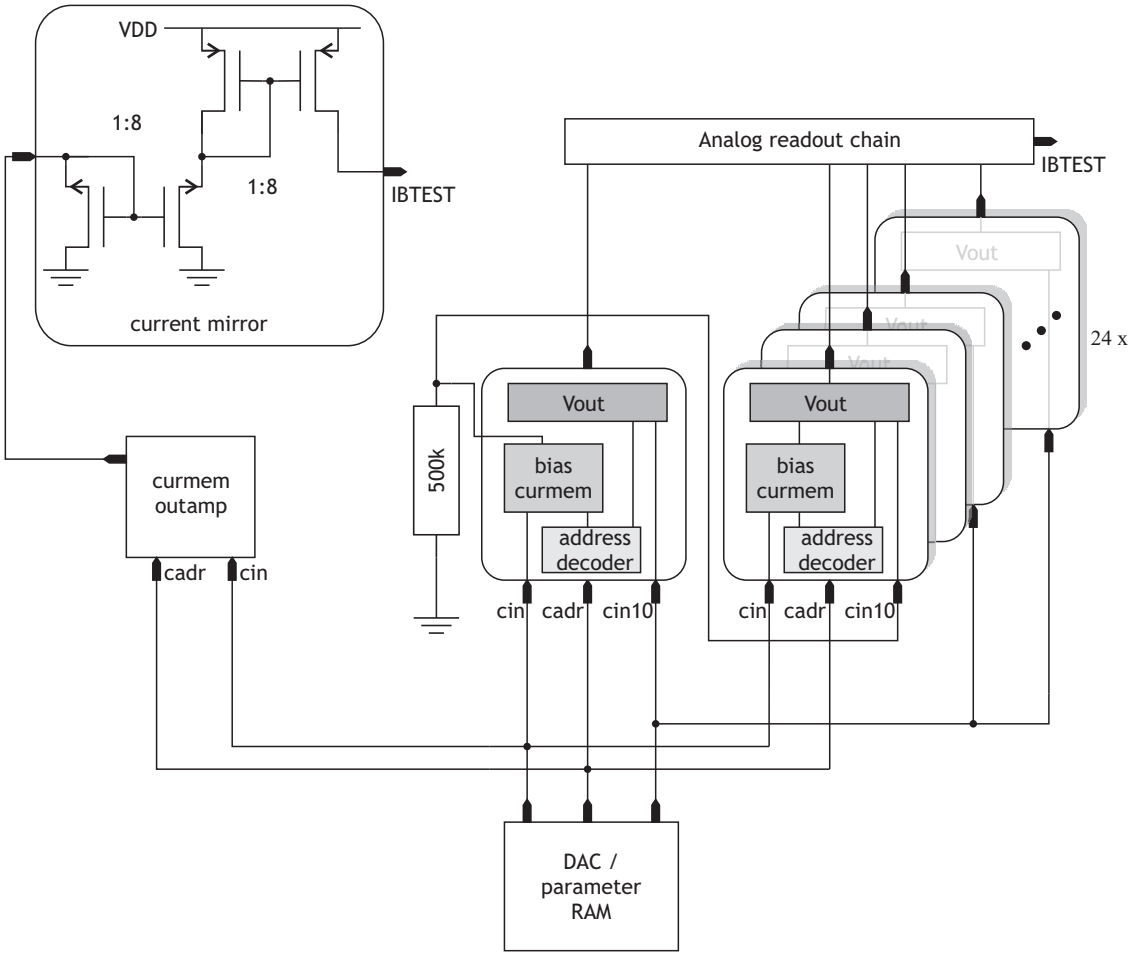

Figure 3.7: Block diagram of the current readout mechanisms of FHW-1.4. On the left side the newly implemented current mirror can be seen which is connected to the ninth *outamp* current memory cell. The right side of the schematic emphasizes another method to indirectly measure the currents that are generated by the DAC via a supplementary *curmem_vout_cell*.

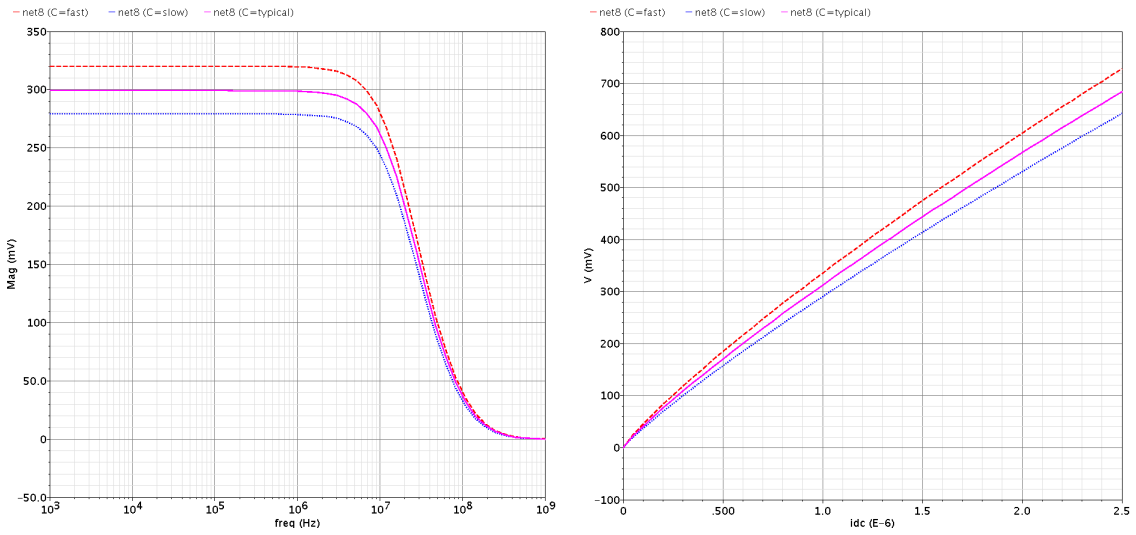

Figure 3.8: Corner analysis of the current mirror, showing an AC analysis on the left side and a DC analysis on the right side. The lines represent the fast, typical and slow corners from top to bottom. The lines for the corners *slow-n-fast-p* and *fast-n-slow-p* have been omitted because they are similar to the typical corner. The x-axis shows the frequency and the DC input current, respectively. On the y-axis the voltage at a resistance of 3 kΩ, that is connected between the current mirror output and ground, is shown.

left graph shows an AC analysis of the circuit with an input current of 0.5 μA. The resulting output value (shown in mV along the y-axis) is the voltage drop from the output of the current mirror to ground over a 3 kΩ resistor. It can be seen that the current mirror is operable up to a frequency of 1 MHz. The right part of figure 3.8 shows a DC analysis of the circuit with an input current ranging from 0 to 2.5 μA for different process corners. Along the y-axis the current flowing out of the current mirror through a 3 kΩ resistance to ground is plotted. This resistance has also been used on the Recha board for the measurements presented in chapter 5.

The second method that was introduced to measure the current in an indirect fashion can be seen on the right side of figure 3.7. It makes use of the *curmem_vout_block* which was introduced in section 2.2.1. This block consisted up to FHW-1.3 of 24 current memory cells, of which four were not used for parameter generation in the network block. Three of these were used to buffer internal chip voltages making use of the unity gain op-amp inside the voltage generating cell and its analog readout chain. For this task the input circuits connected to the *cin10* pin were not used. The last *curmem_vout* cell in each block was not used at al in Spikey's third revision. Every voltage cell has two analog current input terminals, one for the op-amp bias current and one for the current controlling the programmed voltage. The former current is saved in a current memory cell. By adding another *curmem_vout* cell to each block it was possible to measure the current saved in the current memory of the 25th vout cell by feeding its voltage drop over a 500 kΩ high resistance poly silicon resistor into the positive input of the op-amp of the 24th cell.

After having added another *vout* cell the address decoder block had to be redesigned. The address decoder consists of a series of transistors that implement a so called *dynamic logic*[8] block. Therefore, the 6 bit input address bus, going from the parameter RAM controller to each vout block, both, as positive and inverted signal, has to be connected to the address decoder transistors of each vout cell such that the current input terminals only get activated when the correct address is applied. The connection of the address bus to the cell block has already been implemented in former revisions of the chip by Dr. Andreas Grübl in terms of a cell view containing metal vias, which can be instantiated in the *curmem_vout_block* with the Cadence

[8] also known as *clocked logic*, as opposite to *static logic*

design software. This software makes use of a scripting language called *Skill*. By using a Skill script to place the necessary vias in a cell view, this rather tedious task was easier to carry out and enabled the author to place the connection vias relatively easy by only modifying the generating Skill script.

3.2.4 Changes in the current parameter assignment

For reasons of documentation, this section will name a change in the network block parameter assignment. The upper four parameters shown in table 3.1 are stored for every synapse whilst the lower two parameters can be adjusted for each neuron. In FHW-1.4 the neuron threshold comparator bias current has now been set to a fixed value. Instead, the same current memory address can now be used to adjust a particular neuron's refractory time, i.e. the time during which the neuron's membrane potential is pulled to the reset voltage.

Address	Parameter
0x0	synapse output driver
0x1	delay element before synapse driver
0x2	pulse shape rise time
0x3	pulse shape fall time
0x0	neuron membrane leakage (g_{leak})
0x1	neuron V_{th} comparator speed

Table 3.1: List of analog parameters available in each column and row circuit in FHW-1.3. All parameters are bias currents supplied by a current memory. The actual physical addressing is given in Appendix A of [14].

Chapter 4

Experimental setup

This chapter outlines the experimental setup that was for the measurements presented in this thesis. At first, the measurement devices that are supplied by the supporting hardware that belongs to the FACETS Stage 1 Hardware System are introduced. This description will be followed by a listing of the additional measurement devices that have been used. The last section of this chapter will describe the software framework that supports the FHW-1.

4.1 Supporting hardware

The *Recha* PCB, serving as carrier board for the Spikey chip, contains an analog-to-digital converter (ADC) and multiplexer (MUX) since the second revision of the board that was designed as a conjoint work of Dr. Andreas Grübl and Boris Ostendorf. These enable the user to measure the voltages of the 8 neuron membrane potential readout pins and of the `IBTEST` measurement pin via the Stage 1 software framework to which the aforementioned current and voltage parameter monitoring devices are connected. A schematic of this readout circuitry is depicted in figure 4.1. It shows that the eight neuron membrane potential pins can be switched to a net called `TESTPIN` to which the net `IBTEST` is also connected using a unity gain buffer. The latter net was of special interest for the work presented in this thesis, since it is used for the output of the analog monitoring devices that have been designed by the author. The multiplexer is controlled by the Nathan FPGA via so called *sideband data packets*, i.e. data packets sent to the FPGA by the software of the controlling host PC that do not affect the direct communication with the Spikey chip. Other packages of this type include the control of the digital-to-analog converters that generate some external voltages necessary for a proper operation of the chip and information that is read from the on-board temperature sensor or the ADC. This analog-to-digital converter can also be seen in figure 4.1. It is used to measure the parameter voltages on the chip to allow for automatic calibration of the voltage parameter cells.

The 50 Ω resistance that is connected to the LEMO jack has to be kept in mind, as well. It causes a voltage division by a factor of two together with the 50 Ω input resistance of the oscilloscope. The resistance at the input of the unity gain buffer has been selected to a value of 500 kΩ by the designer, but had to be exchange in order to allow for accurate current measurements, as will be explained in more detail in section 5.2.

4.2 Measurement devices

4.2.1 Measurements with LVDS lines

The Tektronix TDS 725 oscilloscope was used to perform the measurements of the Physical layer of the Spikey chip. It has an analog bandwidth of 2.5 GHz and a maximum sample rate of 20 GS/s. It was used with the P7330 active differential probes which have a bandwidth of 3.5 GHz to measure LVDS signals.

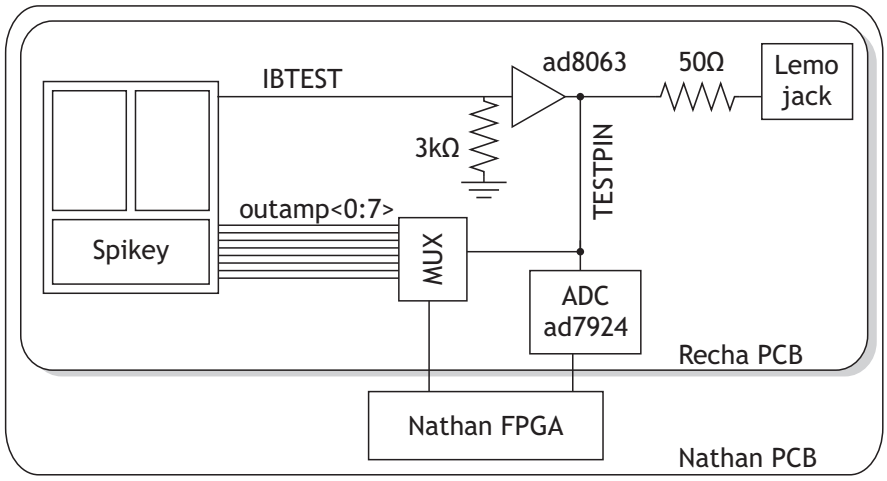

Figure 4.1: Block diagram of the readout circuitry on the *Recha* PCB. The nets `outamp`<0:7> can also be directly accessed via separate LEMO jacks which are not depicted. The schematic and the complete layout of the *Recha* board can be seen in appendices B and C of [20]

The LeCroy Serial Data Analyzer 6000A was also used to perform some of the measurements of the digital part's Physical Layer. The oscilloscope has an analog bandwidth of 6 GHz and has 4 input terminals with a sampling rate of 10 GS/s. The input resistance of the terminals is 50 Ω. It was used with the *WaveLink 600* 6 GHz differential probes.

The Tektronix AWG2041 arbitrary waveform generator was used to generate clock signals with arbitrary frequency for the measurements in section 5.1. It was connected to the external clock input of the backplane. The waveform generator features 1 GS/sec with 2 V peak-to-peak output amplitude.

4.2.2 Analog measurements

The LeCroy waveRunner 44Xi oscilloscope has a bandwidth of 400 MHz and has 4 input terminals with a sampling rate of 5 GS/s, resulting in a maximum time resolution of 200ps. The input resistance of the terminals was chosen to be 50 Ω. Therefore, the measured voltages, have to be doubled to yield the correct results when the oscilloscope is connected to the *Recha* PCB via the lemo jack, because the jack also has a termination of 50 Ω.

The Keithley 2100 Multimeter is a 6 1/2 digit[1] USB digital multimeter. It features standard measurement methods such as DC measurement, AC measurement and resistance measurement. The most important reason why it was chosen for the measurements in this thesis is its capability for USBTMC[2] readout. This specification defines a communication protocol as a vendor-independent standard for USB-based instruments, published in 2002. For the readout of the instrument, a Linux Kernel driver by *Agilent Technologies* was used, which is described in [28]. A C++ class for the readout of DC voltages, written by Sebastian Millner, already existed and could easily be included into the Stage 1 test software. According to the data-sheet [16], the multimeter has a resolution of 1μV in the used voltage range of 1V. The accuracy is given by ±(0.0045% of reading +0.0008% of range). This means the measured values are subject to an absolute error of 0.8mV and an relative error of 0.0045%.

[1] Of the 7 available digits the most significant digit can only be 1 or 0
[2] USB Test and Measurement Class

The AD7924 ADC was used for some of the measurements in section 5.2. It is depicted in figure 4.1. The used component is an *Analog Devices* analog-to-digital converter, which has a specified differential non-linearity[3] (DNL) of -0.9/+1.5 in units of the maximum value of the LSB[4] [6]. Since the specified maximum reference voltage is 2.5 V and the resolution of the ADC is 12 bit, the expected measurement error is approximately -0.55/+0.92 mV.

4.3 The FACETS Stage 1 software framework

The software framework of FHW-1 has originally been designed by Dr. Johannes Schemmel and Dr. Andreas Grübl. It has since been improved and extended by many other people and has become part of a much more comprehensive software suite that allows for automatic mapping of neuro-scientific experiments (described in *PyNN*[5]) to the multi-chip hardware system. The core part of this software, which has also been used in this thesis, is written in C++. Around this core software, an interfacing software layer, written in Python, acting as a translation layer between *PyNN* and the C++ low-level software has been developed by Dr. Daniel Brüderle and Eric Müller.

The Electronic Vision(s) group has lately seen increasing activity towards a unification of the software development efforts of its members. This includes better communication, non-regular meetings about software organization, the use of a software management system (called *indefero* [17]) and the unification of the build process. The complete software framework also has been integrated into the so called *symap2ic* software project which stands for "Neural Architectures with Synaptic Plasticity Mapped to ICs". The changes in the used build process have also affected the Stage 1 low-level software which now goes by the name *SpikeyHAL* (*Spikey* Hardware Abstraction Layer) and is built with the *waf*[6] build system. This build system offers more flexibility (because the build scripts are plain python scripts) and allows for automatic configuration of the compilation prerequisites. It replaces the formerly used, rather static, *make* build system. The transposition of the build system of *SpikeyHAL* has been realized by Eric Müller and the author.

The *Spikey* test software itself consists of a program called *createtb* that allows for the execution of so called *testmodes* which are pieces of C++ code, each testing a certain functionality of the *Spikey*chip. The output of the software are data packets that are sent to the Nathan FPGA via a PCI card or a Gigabit Ethernet connection. These packets can either be event or control interface packets, intended for the communication with the chip, or sideband data packets that contain information about the periphery on the Nathan and *Recha* PCBs. The control interface packets are necessary to configure the *Spikey* chip. They can be generated by low-level interfaces, allowing the user to set every individual bit of the according registers on the chip, but they can also be automatically generated by higher level functions that read so called *spikeyconfig* files containing a complete chip configuration. These different levels of abstraction are depicted in figure 4.2.

The readout functionality of FHW-1 is also controlled by the FACETS Stage 1 software framework. Thus, before the parameter voltage cells on the *Spikey* chip could be verified, a workaround for the control of the analog readout chain had to be implemented. As described in section 3.1, the analog readout input register has not been incremented, although the analog readout chain has been augmented by one. In FHW-1.3, it was possible to write the values of the register in question with one single write statement. In the latest version of the chip however, two write statements have to be issued to assure that only one parameter voltage cell's analog output is connected to the `IBTEST` net. This fact can be explained by the two exemplary use cases that are shown in figure 4.3. In the diagram, the size of both elements has been reduced

[3] the difference between the measured and the ideal 1 LSB change between any two adjacent codes in the ADC
[4] Least significant bit
[5] A simulator-independent language for building neuronal network models.
[6] "Waf is a Python-based framework for configuring, compiling and installing applications." C.f. [23]

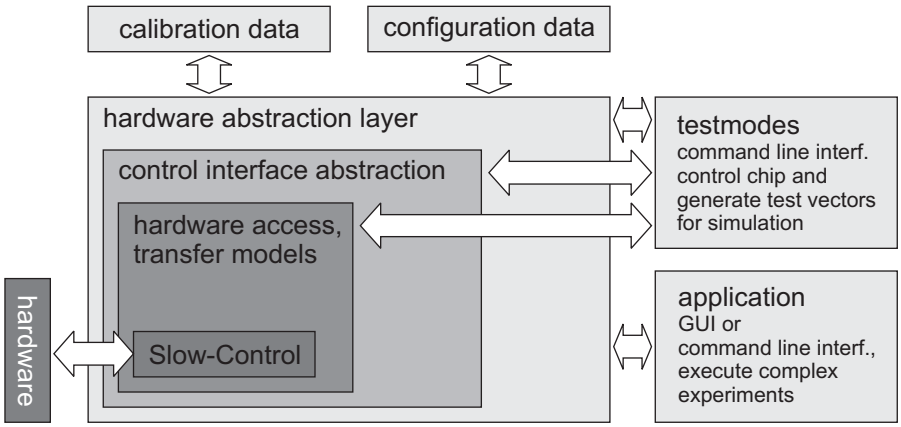

Figure 4.2: Different levels of abstraction of the FHW-1 software framework. Taken from [14], page 131.

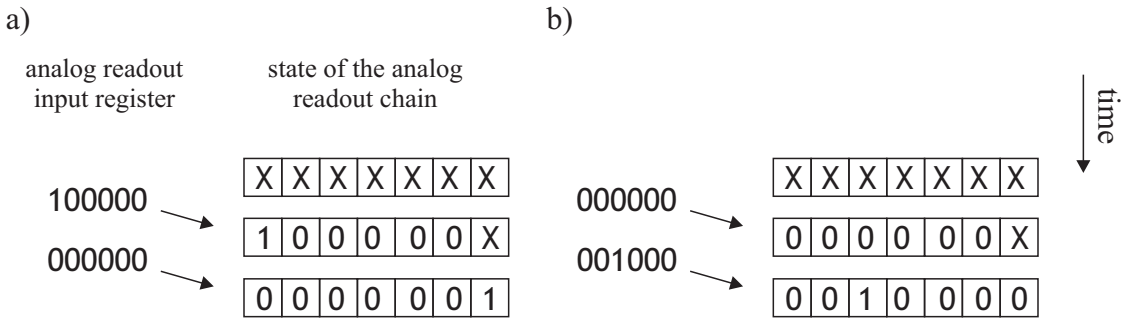

Figure 4.3: Analog readout input register (in the digital part of the chip) and state of the analog readout chain with time. The sizes of both registers have been reduced for better readability. '1' denotes an activated analog readout transmission gate, '0' a disabled one. 'X' symbolizes an unknown state. **a)** Shows the two bit strings that have to be written to the analog readout input register in order to activate the last cell in the analog readout chain. **b)** Shows those for the activation of the third cell.

for better readability, the crucial difference in size of one bit has been kept. On the chip the analog readout input register has a size of 24, whilst the analog readout chain has a size of 25.

The schematic shows that, in order to activate only the last cell in the readout chain and given the fact the state of the readout chain beforehand is unknown, one has to activate the first voltage cell first and move the activation bit to the end of the chain afterwards. For the case in which one intends to activate one of the first 24 cells, only zero values have to be written to the chain followed by a high value at the correct position.

4.3.1 PyNN and FHW-1

PyNN is a simulator-independent language for building neuronal network models. It provides a Python API that allows to describe neuronal networks for experiments with software simulators or neuromorphic hardware. These descriptions can then easily be mapped to different software simulators or the FACETS Stage 1 hardware system [3] and can be shared amongst neuroscientists. Since PyNN scripts are valid Python code, one can include every other Python module to post-process the experimental data. This may also include visual presentation of data, using rasterplots.

Chapter 5

Experimental Verification

5.1 The Physical layer

This section describes the characterization of the delay lines in the Physical layer.

First, the step size of the delay line of output signal `CAD1_OUT<3>` was measured. Therefore, the falling edge of the signal was monitored on an oscilloscope while triggering on the rising edge of its according clock signal `CLK1_OUT`i and recording both in infinite persistence mode. In this mode the oscilloscope remembers every measured signal and continues displaying it on the screen. The number of active delay elements was then stepwise incremented by one. The results can be seen in figure 5.1. From this measurement, the delay step size was obtained, it can be seen in figure 5.2. The resulting line, which was only drawn for better readability, has a slope of 82 ps.

These measurements have only been carried out on a single chip which obviously imposes a lack of sufficient statistical measures. This was mainly because most of the available Stage 1 hardware had to be prepared for external presentation of the group's work in the last weeks before this thesis was submitted. For the same reason, a sufficient analysis of the maximum operating frequency of FHW-1.4 has not been carried out. However, tests of the link capacity of a single sample chip have been carried out which have showed that it is possible to operate the latest version of the chip with a clock frequency of 312.5 MHz.

There have also been implemented automatic methods into the software framework that can adjust the optimum delay values for each data input link. The functionality of these will be explained in the following paragraphs. However, there is up to now no functionality implemented in the software framework that allows to save and load optimum delay values for a single chip in a configuration file. This should be done in the near future.

The delay values of the input data buses of *Spikey* can be measured and adjusted to maximize the data valid window with a testmode. It exploits the features of the *Digital Clock Managers* that come with the *Virtex II Pro XC2VP7* FPGA, the core unit of the Nathan module. These clock managers allow for a phase shifting of the signal clock relative to the signal data lines with an accuracy of "clock period divided by 256" with a minimum step size of 30 ps[1]. In the environment in question, the clock period varies between 5ns (200MHz) and 2.5ns (400 MHz), resulting in a theoretical phase shift step size of approximately 20ps to 10ps. It turned out that the actual phase shift was indeed 20ps (in the case of a 200MHz clock) instead of the minimum step size of 30ps, as stated in [29]. The oscilloscope screenshot in figure 5.3 shows the clock signal of the CAD1 input bus operating at 200 MHz (brown trace) and the CTL bit of the same bus (blue trace) in infinite persistence mode. The oscilloscope has been adjusted to average over 50 trigger events. In infinite persistence mode, different states oft the same input bit can be seen and compared at the same time. Between two positions of the blue trace, the DCM phase register has been increased by 20. The blue vertical cursors show a time difference of 400 ps, thus resulting in a phase shift step size of 20ps per DCM step. For a clock frequency of

[1] According to the Platform Handbook [29], page 48 (speed grade 6)

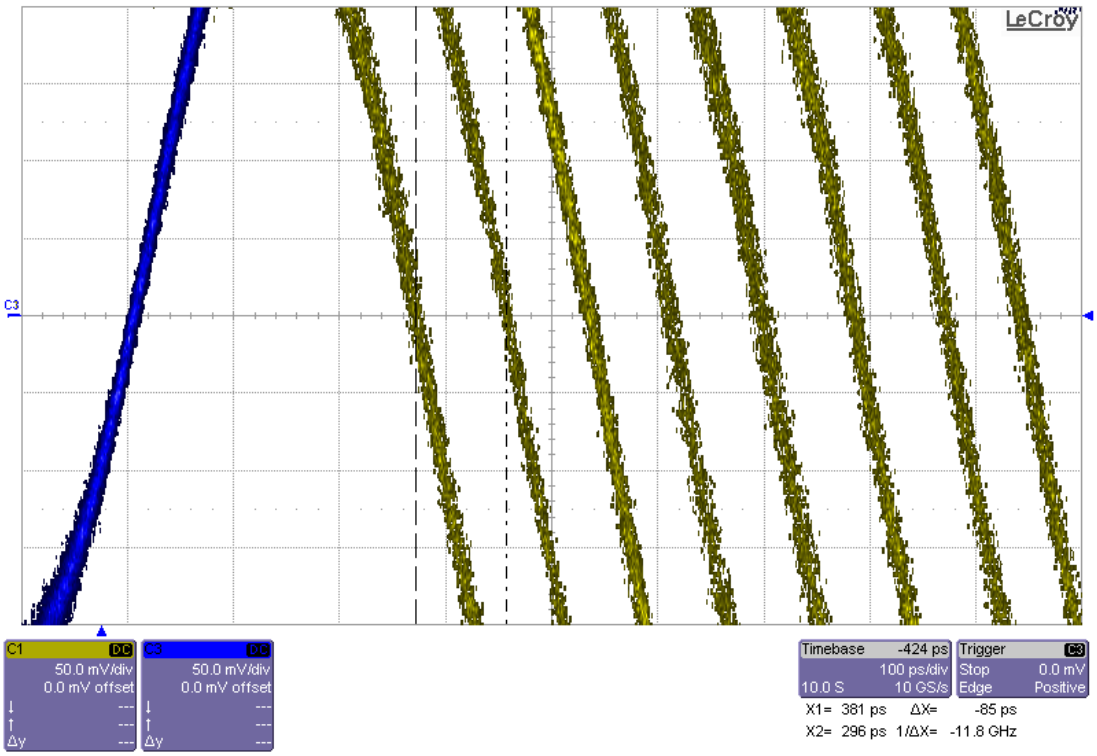

Figure 5.1: Clock signal `CLK1_OUT` (blue, leftmost line) and data signal `CAD1_OUT<3>` (yellow, right lines) in infinite persistence mode at 200 MHz clock frequency. Between two positions of the yellow trace the number of active delay elements in the transmission unit of the Physical Layer was incremented by one.

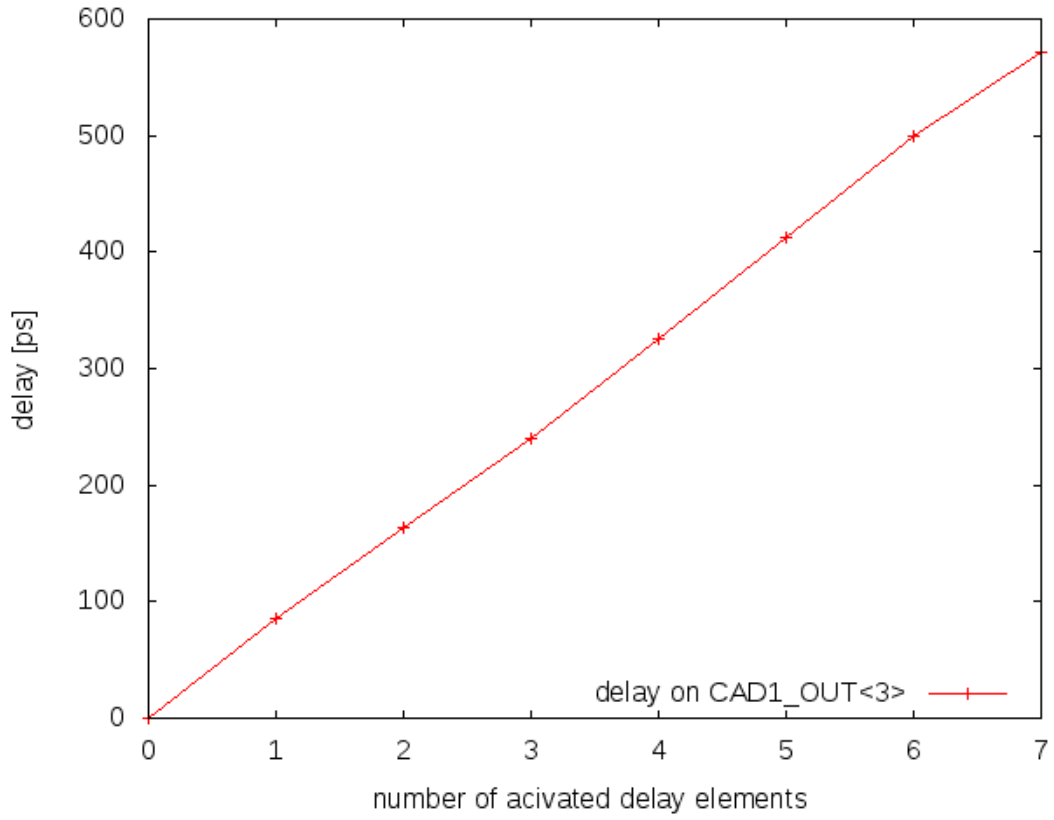

Figure 5.2: Delay values of data signal `CAD1_OUT<3>` and relation to time delay in ps relative to according clock signal at 200 MHz. Values taken from measurement as described in figure 5.1.

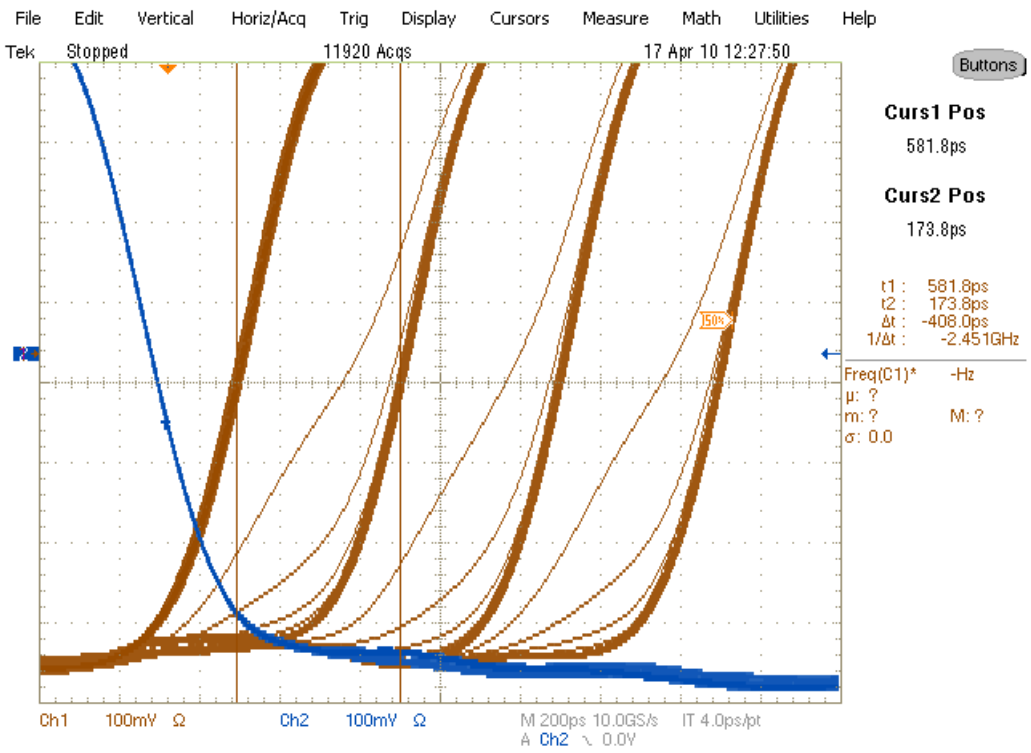

Figure 5.3: Clock signal of input bus CAD1 (blue) and CTL bit of CAD1 (brown) in infinite persistence mode at 200 MHz. Between two positions of the brown trace the DCM phase shift has been incremented by 20 steps. The oscilloscope averages over 50 trigger events. The fine brown traces correspond to intermediate traces that have been recorded between two DCM phase shift values and can still be seen because of operation in infinite persistence mode.

312 MHz the expected phase shift step size of approximately 13 ps could also be verified in the measurements shown in figure 5.4. In this case the cursors indicate a time difference of 240 ps per 20 steps in the DCM register which results in a step size of 12 ps per DCM increment.

This fact in mind, it was possible to measure the time shift of one delay line step via the software framework. These values were obtained by shifting the input link clock relative to the data lines until the chip was no longer able to sample each line at its input registers. After all the input lines were no longer readable, the delay values were incremented simultaneously for each input line and the clock shifting was repeated. It was then possible to measure the difference between the maximum phase shift value at which one line was still readable for every pair of two adjacent delay values. This argument implicitly rests on the assumption that the transmission unit of the *Spikey* chip is working correctly and has itself correctly adjusted delay elements that allow optimum sampling results by the FPGA.

After having determined the delay step size for each signal line pair of the input bus, the software can measure the phase shift of every input line relative to the according clock signal. It can then adjust the delay values accordingly, such that signals that would arrive earlier can be adjusted to an arrival time closer to that of the other signals. Afterwards the clock phase is adjusted to a value that maximizes the data valid window, i.e. to a phase shift value that corresponds to the mean value of the signal that arrives latest and that arriving at first.

Of course it should be noted, that the time shift of the data bit relative to the clock signal is only about one seventh of the expected time delay. Therefore a measurement with this method is not very accurate, but the accuracy is sufficient to implement a software algorithm that can automatically adjust the delay values of each input line by carrying out the above mentioned steps.

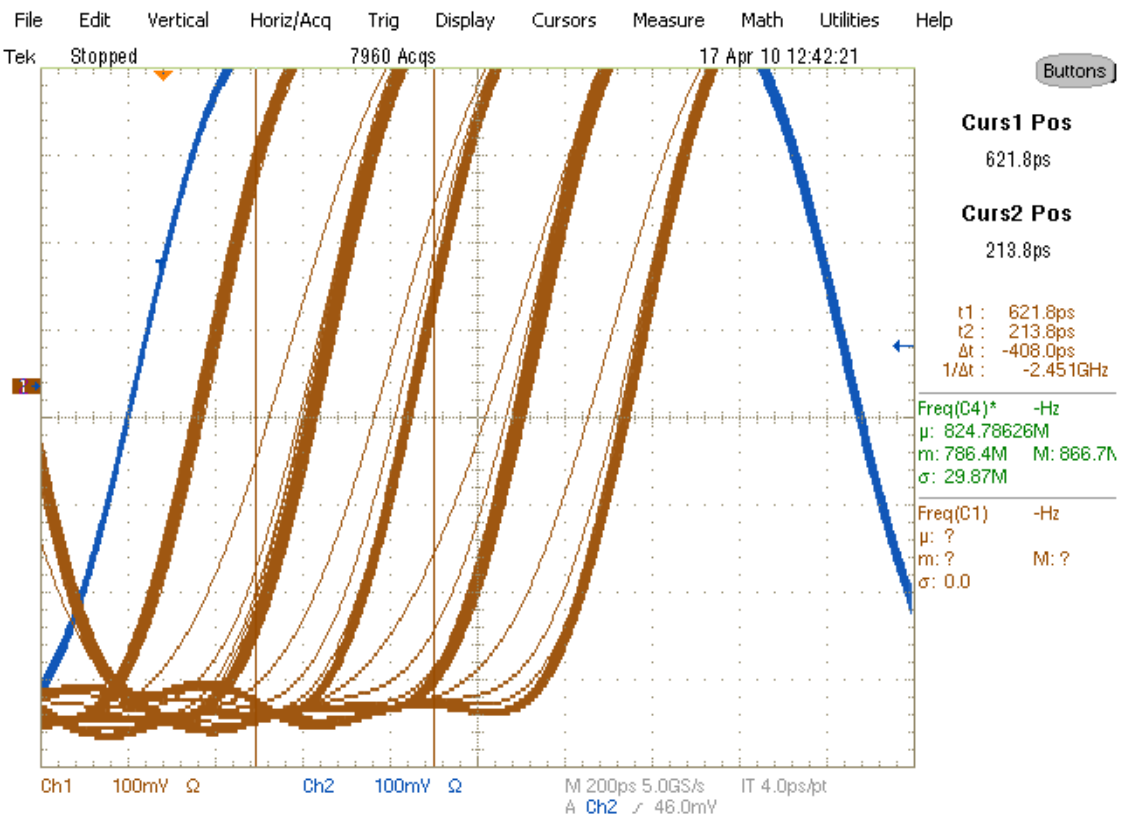

Figure 5.4: Clock signal of input bus CAD1 (blue) and CTL bit of CAD1 (brown) in infinite persistence mode at 312 MHz. Between two positions of the brown trace the DCM phase shift has been incremented by 20 steps. The oscilloscope averages over 50 trigger events. The fine brown traces correspond to intermediate traces that have been recorded between two DCM phase shift values and can still be seen because of operation in infinite persistence mode.

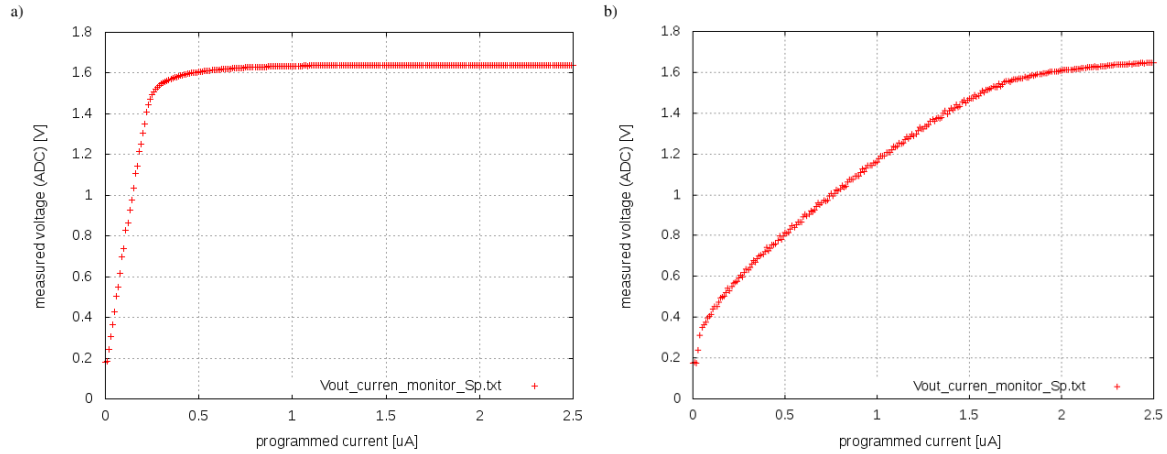

Figure 5.5: Indirect DAC current measurement via parameter voltage cell and 500 kΩ resistor. Single measurement. a) without boost and I_{refdac}=25 μA b) Same measurement but with boost enabled

5.2 Parameter value generation

The investigation of the parameter current generation on the Spikey chip was a central objective of this thesis. As stated in chapter 5, FHW-1.4 is the first revision of the hardware system that supports reliable current readout mechanisms. These have been tested for correct functionality and have been exploited to improve the parameter refresh timing in order to supply reliable current and voltage values while keeping the parameter refresh cylce as short as possible. Strictly speaking the latter criterion is a requirement for the validity of the former. The measurements that have been carried out in this context will be explained in the first subsection.

The second subsection describes the experiments that have been carried out to verify the (improved) functionality of the voltage generating cells.

5.2.1 Parameter currents

During the first measurements that have been carried out with the indirect current readout functionality of FHW-1.4 (introduced in section 2.2.1) it has been found out that in a first naive try the currents generated by the DAC were approximately tenfold larger than their expected values. This can be seen from figure 5.5a. The graph shows the voltage resulting from the current that is sourced by the DAC flowing through a 500kΩ resistor plotted against the programmed current values as translated by the control software. The voltages were measured via the ADC on the *Recha* board (as described in section 3.2.3). The current values are translated by the software, resulting in 10-bit DAC values that are then sent to the parameter RAM controller. One would expect a voltage range of 0 V to 1.25 V for a current range of 0 A to 2.5 μA, however, it turned out that a voltage of 1.25 V has already been reached at an intended current value of 0.2 μA.

The translation between the current values the user wants to program onto the chip and the actual 10-bit DAC input values takes place inside the software. The DAC input value is calculated by dividing the current to be programmed by the maximum current that the DAC can supply and multiplying the result by 1023 (of course, the result will have to be mapped to the set of natural numbers).

$$DAC_{in} = I_p/(I_{refdac}/1024 \cdot 1023) \cdot 1023 \tag{5.1}$$

The same behaviour could be reproduced using the current mirror output, as can be seen in figure 5.6. Both lines show the voltages measured at a 3 kΩ resistor on the Recha PCB,

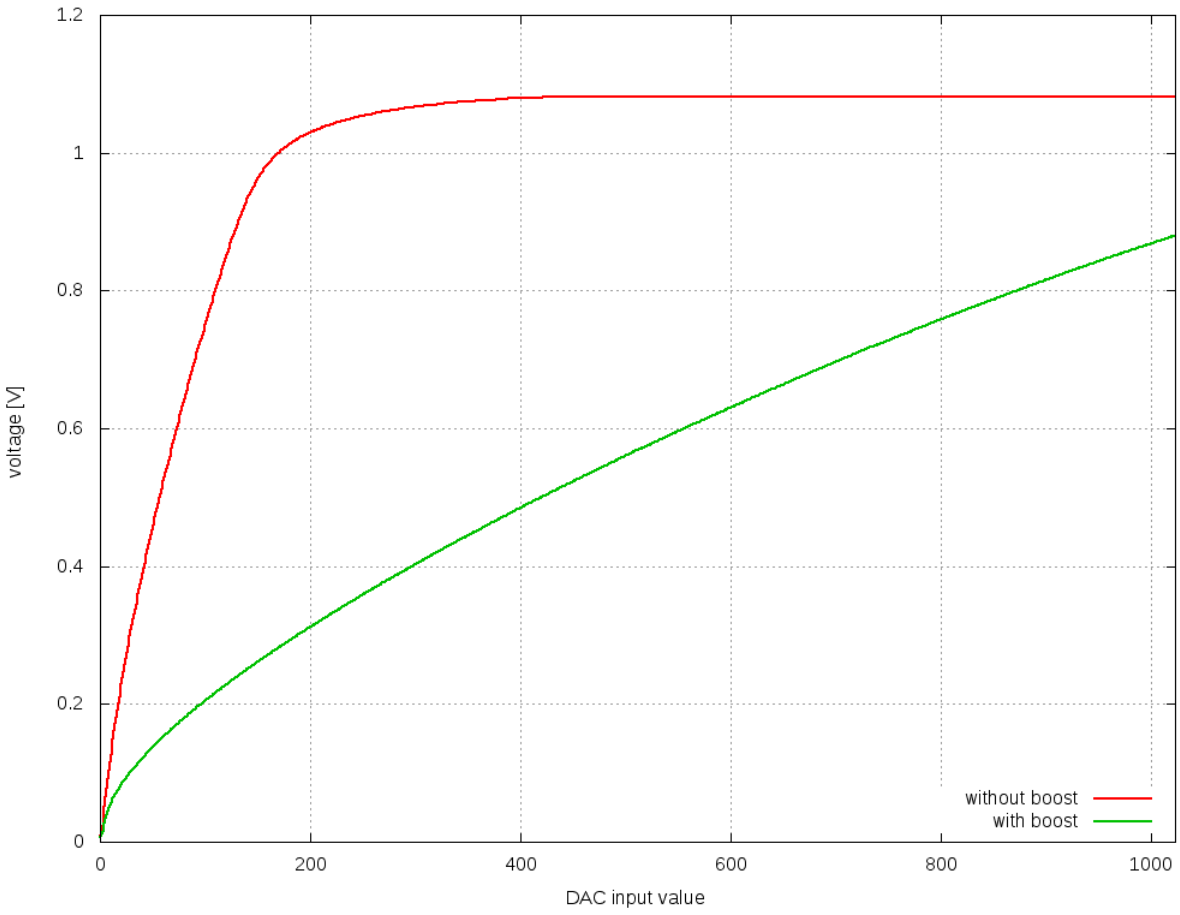

Figure 5.6: Direct DAC current measurement via on-chip current mirror. Previous parameter RAM value was 0. The measurement was repeated 10 times, mean value is shown, error bars have been omitted because the standard deviation is 2 to 3 orders of magnitude smaller than the actual values. Upper line: without boost and I_{refdac}=25 μA, Lower line: Same measurement but with boost enabled

caused by the current that is sourced by the current mirror on the *Spikey* chip. On the x-axis, the programmed DAC input value can be seen. The upper line was measured without the additional 9 current memories that can be activated via the boost signal. The lower lines was measured with active boost signal. Depicted is the mean value of ten repeated measurements, the standard deviation has been omitted because it is 2 to 3 orders of magnitude smaller then each according mean value.

Further investigation of this problem showed that the reference current of the DAC had been set to 25 μA in the default setup that has always been used for experiments on FHW-1.3. As described above this reference current can be divided on a 10-bit basis to control the current supplied by the DAC resulting in a current range of 0 to 25 μA. Thus, as the upper line in figure 5.6 shows, the programmed DAC input values intended to control a current range of up to 2.5 μA actually resulted in a current range of up to 25 μA. Probably, the value for the DAC reference current has been chosen to yield optimal performance for the parameter voltage cells, because they were the only measurable parameters on the chip up to FHW-1.3. Since these voltages are generated with a 10 kΩ resistor, a current of at least 180 μA is necessary to generate a voltage drop as large as the positive supply rail. For the DAC to source a current of this size with its tenfold output, a reference current of at least 18 μA would be necessary. Of course, this problem wouldn't have appeared if the conversion method in the software had used a correct reference current value for its calculations, but this, on the other hand, would have resulted in a lower resolution of only 7 bit. In the documenting texts of previous work that has been done to improve the calibration of the parameter voltage cells ([19] and [20]), no hint on why a reference current of 25 μA was chosen could be found. The erroneous usage of the on-chip DAC could only be attributed to communication problems.

A solution to this problem could be found by making use of the boost mechanism in the DAC block. Therefore, the look-up-tables of the parameter RAM controller were configured such that always when writing current values to the analog part, the controller would activate the nine supplementary current memories while not multiplying the intended 10-bit DAC input value by ten. As described in section 2.2.1, using the boost mechanism in a LUT that also foresees some write cycles without boost would lead to a multiplication of the DAC input value by a factor of ten during the boost write time. This can be avoided by only setting the register corresponding to boost write time to a non-zero value.

This workaround made it possible to operate the chip decently. Figure 5.5b shows a first try of an indirect current measurement via the vout cell in the right block. These results are still not optimal because one cannot tell whether small currents can simply not be generated by the DAC or the according voltages are out of the dynamic range of the op-amp inside the vout cells (in the case of figure 5.5). Besides, the accuracy of the DAC output depends on the preceding value that has been output by the DAC and the time for which a new current value is applied to the DAC input. In these first experiments these parameters were not controlled because they were set up by reading a full configuration file into the software and sending a complete parameter set to the chip. Because the software at this point only made use of one single look-up-table (determining the write time) and sorted the values in ascending order before sending them to the parameter RAM controller.

The following paragraphs describe the measurements that have been performed to optimize the write timing of the current parameters. The aim was to find a configuration that would result in reliable current values over the whole range of currents up to 2.5 μA while keeping the total time for one full parameter set refresh cycle as small as possible. The latter objective resulted from the fact that for too large refresh cycle times the parameter voltages showed a large drift, which will be discussed in detail in the next section. For the following measurements the current mirror readout mechanism was used, because the dynamic range of the operational amplifiers in the voltage generating cells is not sufficiently large to accurately buffer voltages in the order of millivolts.

First of all, it has been found during the experiments, that in order to be able to measure

small currents, the on-board ADC is not a good choice, because the voltage measured by the ADC is buffered by an operational amplifier of the type AD8063[2]. As can be seen in the data sheet [7], the input bias current has, in the typical corner, a value of 3.5 μA, i.e. a current of this magnitude flows into the input transistors of the operational amplifier. This makes a measurement of currents in this order of magnitude impossible. Therefore the operational amplifier had to be disconnected from its input net and the ADC could not be used in this case (for further information about the readout circuitry on the *Recha* PCB, refer to section 4.1). To avoid manual and cumbersome measurements that involve visual metering, the Keithley 2100 multimeter (c.f. section 4.2.2) has been used. This device features a USB connection port that can be connected to the host PC with a Linux Kernel module supported by Agilent Technologies. The USB functionality of the measurement device has already been made use of by other group members while working with the prototype of the FACETS Stage 2 Hardware System. Therefore, it was possible for the author to integrate the instrument into the software framework with reasonable effort.

In order to yield reasonable results for measurements with the newly implemented current mirror, the measurement resistance on the *Recha* board had to be exchanged. This was because the current range at the current mirror input was 1 nA to 2.5 μA, resulting in a current magnitude ranging from approximately 0.1 μA to 250 μA. But, a voltage drop of 1.8V over a 500kΩ resistance will already be generated by a current of 3.6 μA flowing through the resistance. To give best results over the whole voltage range, a resistance of 3 kΩ was chosen.

With a particularly implemented testmode, DAC input values from 0 to 1024 could now be programmed to the current memory that is connected to the newly implemented current mirror, while being in control of the value that has been set up by the DAC directly before this one. Moreover, the write timing and whether to use the boost mechanism or not could also be controlled for every value that was written to the chip, and without making use of higher level software objects. This way, it could be assured that the user was in full control of all the values that are written to the parameter RAM controller and their write timing in terms of look-up-tables.

Figures 5.7, 5.8 and 5.9 show the resulting voltage measured at the 3 kΩ resistor on the *Recha* board for the programmed DAC input values shown on the x-axis and different write timings. The write timings are given as numbers of cycles in powers of two during which the intended DAC value is applied. In the case of the upper graph the value that has been written before the current value in question during a refresh cycle of the parameter RAM controller has always been 0. The lower graph of figures 5.7, 5.8 and 5.9 show the according measurements with a preceding value of 512. From figure 5.7, it can be seen that currents below 10/1024 of 2.5 μA can only be reliably written when the DAC is given a time of at least 2^{11} cycles of the 100MHz clock. This shows the strong dependence of the time the DAC needs to set up a new value from the previously written DAC value.

The two graphs in figure 5.8 show the same measurements but for DAC input values from 50 to 1023. The graphs show that for a previous DAC input value of 0, only very large values of at least 600 can be written without risking an error of about ten percent.

The graphs in figure 5.9 cover a range of input values from 600 to 800. They show that for a previous value of 0 and an actual value of larger than 600, a write time of 2^7 cycles results in an error of smaller than two percent compared to the result of a write time of 2^{15} cycles. In the case of a preceding value of 512 any value larger than this can be written in 2^5 cycles. This will differ in an error of smaller than five percent compared to a write time of 2^{15} cycles. An error that can be accepted considering other sources of error on the chip and a write time that has been reduced by a factor of 1024.

It is crucial to investigate the accuracy of the DAC in different contexts because in realistic experiments the DAC will have to write currents ranging from 0 to 2.5 μA, reliably. The software will send the parameter RAM entries in a sorted fashion, but this sorted array of DAC input

[2]by Analog Devices

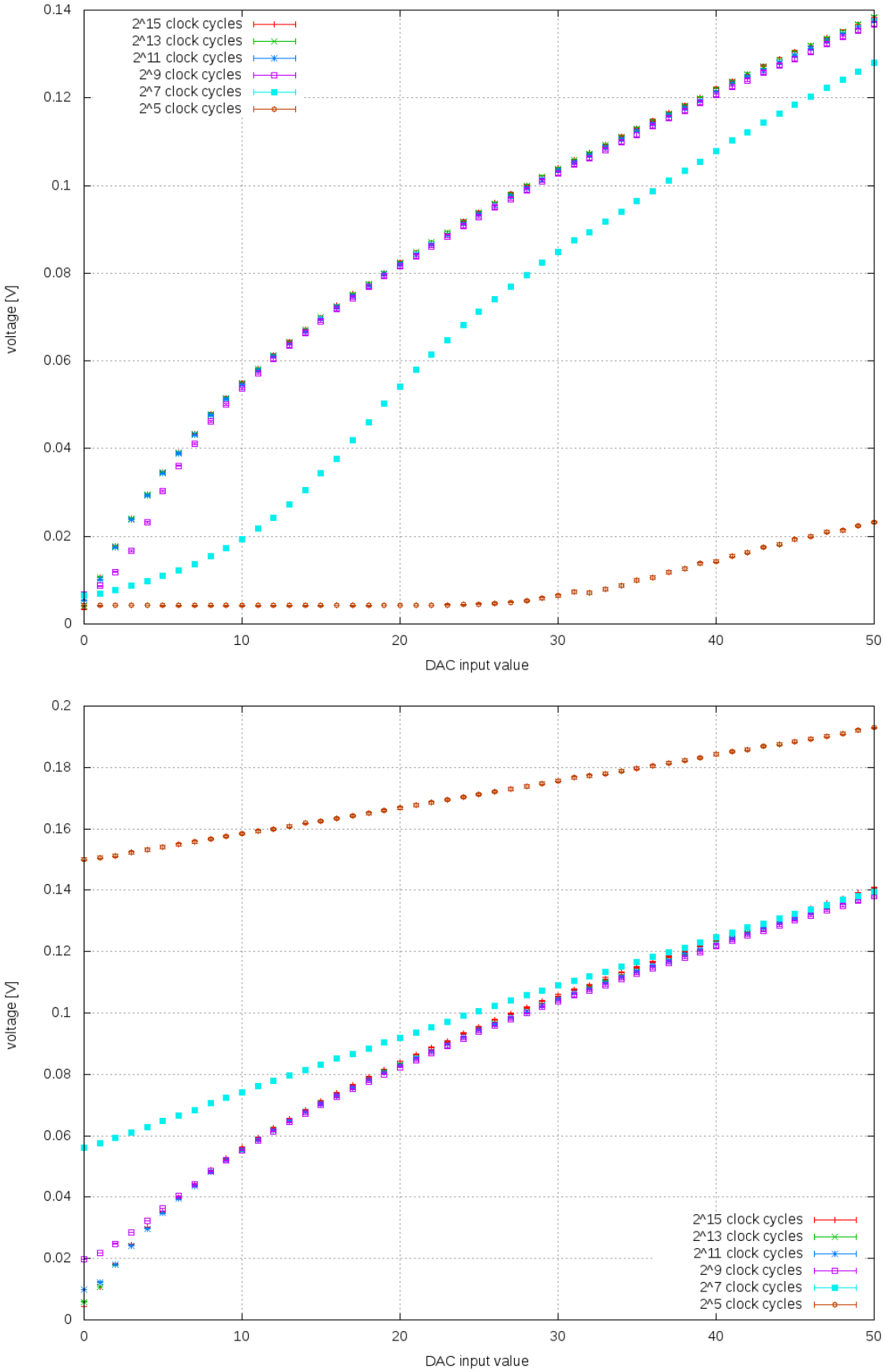

Figure 5.7: Voltages measured with 3 kΩ resistance on the *Recha* board for different currents sourced by the on-chip output current mirror for different DAC input values and different write timings. In the upper image, the preceding parameter RAM value was 0. The lower graph shows the same measurement but with a preceding value of 512. The measurements were repeated ten times, the graphs show the mean value and the standard deviation.

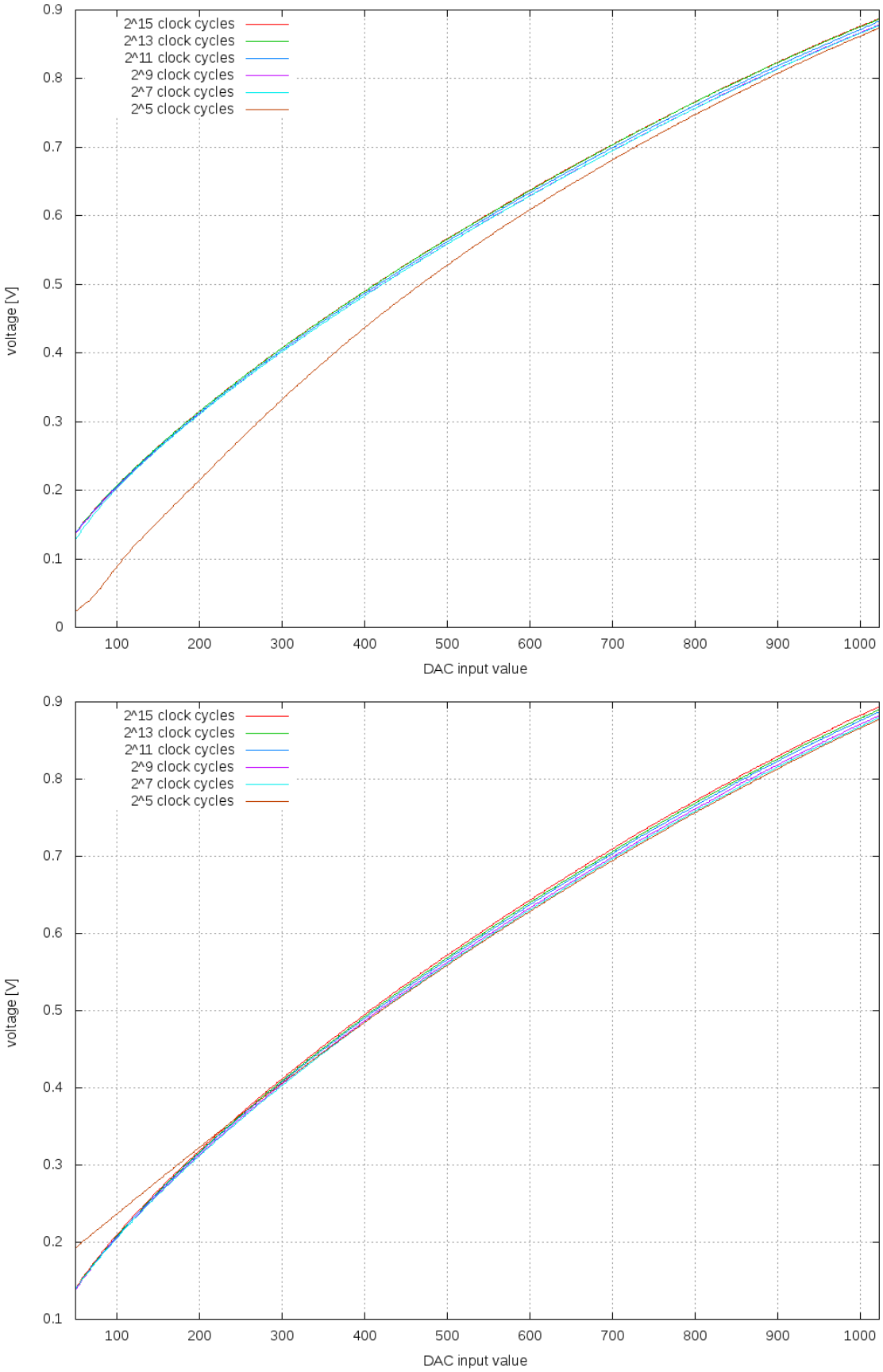

Figure 5.8: Voltages measured with 3 kΩ resistance on *Recha* PCB for different currents sourced by the on-chip output current mirror. X-axis shows DAC input values. Different lines belong to different write timings. In the upper graph, the preceding current value in the parameter RAM was chosen to 0, in the lower graph it was chosen to 512. The measurement was repeated ten times, the graph shows the mean value. Standard deviation has been omitted for better readability.

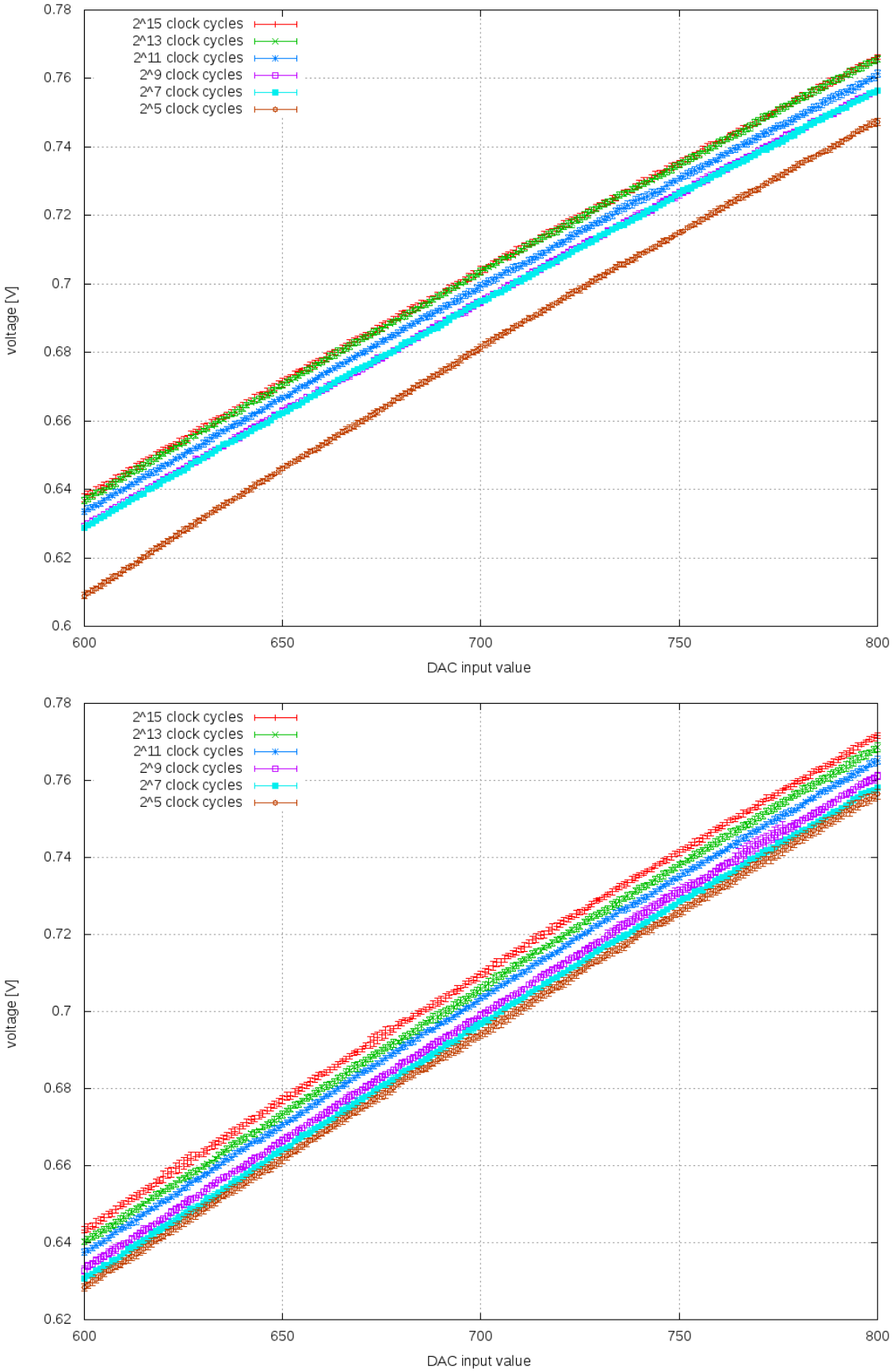

Figure 5.9: Voltages measured with 3 kΩ resistance on the *Recha* board for different currents sourced by the on-chip output current mirror for different DAC input values and different write timings. In the upper image, the preceding parameter RAM value was 0. The lower graph shows the same measurement but with a preceding value of 512. The measurements were repeated ten times, the graphs show the mean value and the standard deviation.

values will still contain steps that have to be compensated by giving the DAC enough time to reload the capacitive load at its output.

From the results of the above mentioned measurements, the parameter write timing as shown in table 5.1 has been chosen. It lists the number of cycles a DAC input value has to be applied at the DAC input depending on the preceding value in the parameter RAM refresh cycle.

actual value	preceding value	number of cycles
< 15	previous value 0	2^{11}
	previous value greater than 0	2^{13}
15-100	all	2^9
100-300	all	2^7
300-600	step larger than 300	2^9
	step smaller than 300	2^7
600-1023	greater than 512	2^5
	else	2^{11}

Table 5.1: Write timing values depending on current value to be applied to the DAC input and current value applied in the previous step of the parameter cycle

These write timing conditions have been implemented into the software framework by the author. This resulted in the following parameter conversion flow:

1. Load parameters from configuration file. Current parameters are given in μA, voltage parameters in volts.
2. Sort parameters by value. This is done by a stable sort algorithm after all parameters have been converted to DAC input values.
3. Apply triangle sort, i.e. every second value (at position n) will be moved to position *size_of_array - n*.
4. Change look-up-tables according to context (as shown in table 5.1)
5. Write parameter data to chip

To test the improved write timings, a modified version of the testmode described above was used. It has been modified to write every DAC input value to the current memory connected to the readout current mirror, while writing a random value with a gaussian distribution of a width of 100 to the current memory that was written just before. A single run can be seen in figure 5.10. Here, the randomly chosen preceding value has been selected from a gaussian distribution with a width of 100 around the value that is intended to be written. It can be seen that every value can be written, forming a continuous line independent of the preceding value. Figure 5.11 shows the resulting graph for 10 runs with mean value and standard deviation.

As a final result it could be shown that with the above mentioned parameter RAM timing, the cumulative time for one refresh cycle could be reduced by more than a factor of two, compared to the default software configuration that had been used before this work has been carried out. And as the measurements show, the currents from 0 A to 2.5 μA can be written in a steady fashion independent of the preceding parameter value.

5.2.2 Parameter voltages

As described in section 3.2.2, the voltage generating cells have been improved during this thesis by integrating a new operational amplifier. Therefore, it has to be verified that theses changes result in the expected larger dynamic range of the cells. These measurements will be described in this section. But before the verification of the improvements, two figures concerning the drift

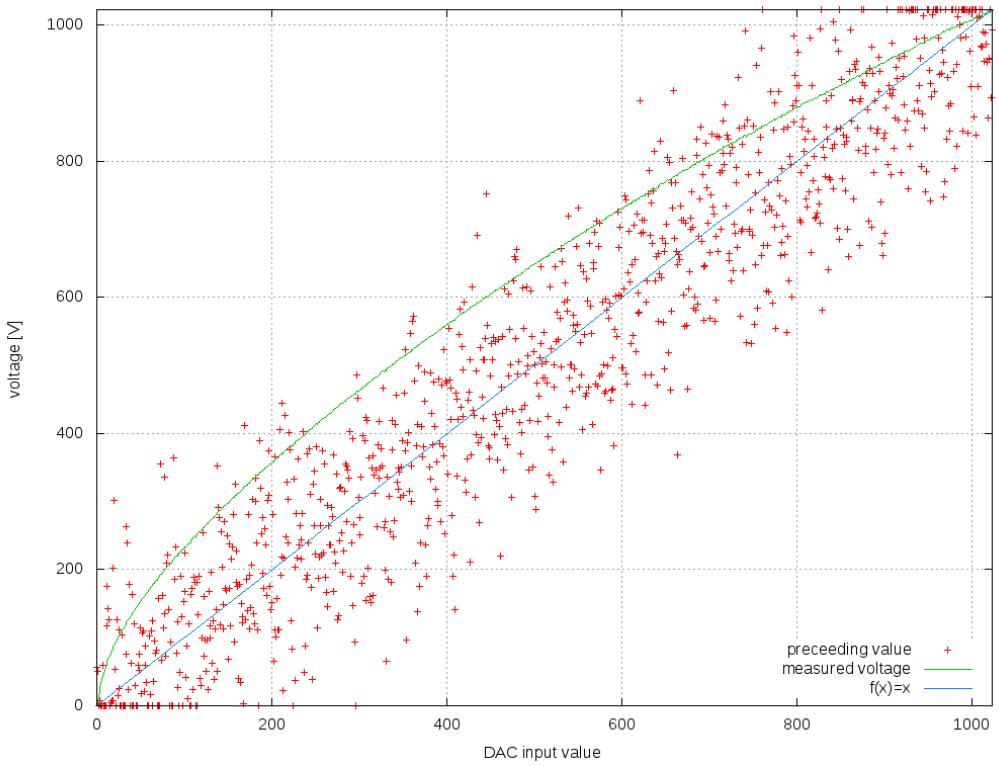

Figure 5.10: Voltage measured on `IBTEST` plotted over programmed DAC input value for out-amp[8]. Random preceding DAC input in parameter RAM refresh cycle. Green: resulting voltage. Red: preceding DAC input value (can be compared to thin blue line showing a line with unity slope going through the origin)

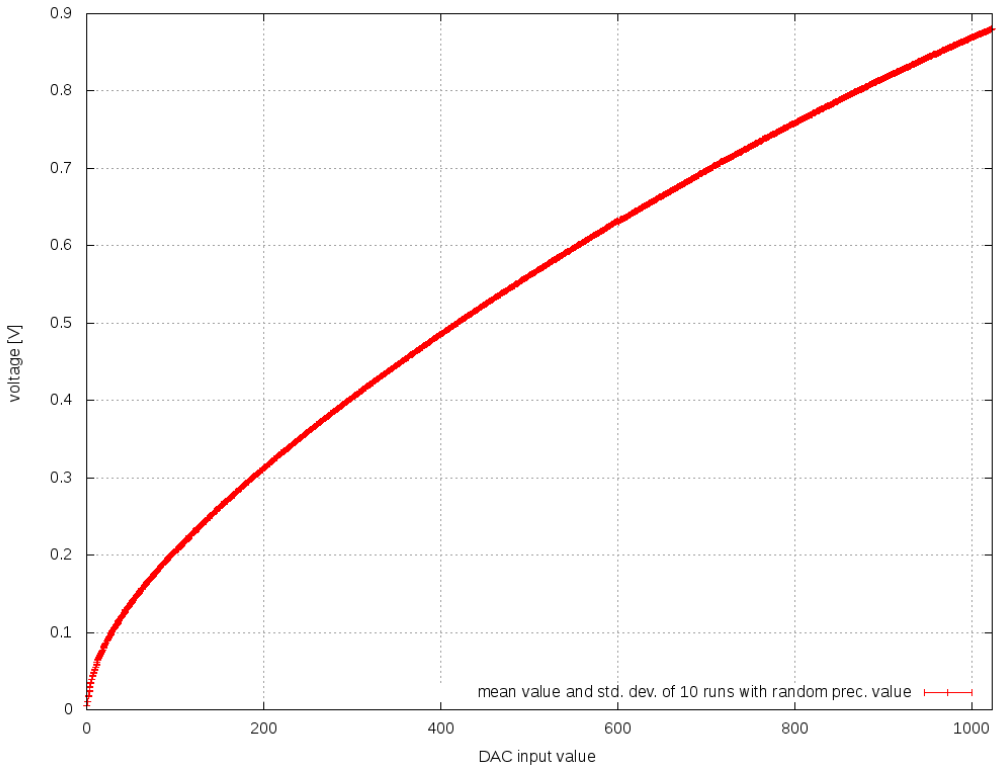

Figure 5.11: Voltage measured on `IBTEST` over programmed DAC input value for outamp[8]. Random preceding DAC input in parameter RAM refresh cycle. Ten repeated measurements, mean value and standard deviation are shown.

of the voltage cells with time will be discussed. As can be seen from figures 5.12 and 5.13, the peak-to-peak drift can be divided a factor of two by dividing the refresh period by four. The figures show a measurement of the voltage output of the first vout cell of the right block, recorded with an oscilloscope. The peaks result from the refresh of the voltage on the capacitance inside the cell by the parameter RAM controller. The zero line of the grid is at an absolute voltage value of approximately 860 mV in the first screenshot, whilst in the second screenshot it is at approximately 870 mV. The actual voltages at the output of the chip are twice as large because the 50 Ω resistances at the output of the *Recha* PCB and at the input of the oscilloscope result in a voltage divider.

This rather qualitative analysis of the voltage drift intends to show that the overall refresh time it takes the parameter RAM controller to refresh all its values once is an important criterion, as has already been stated in the previous section.

To measure the voltages that are generated by the vout cells, the analog readout chain (as described in section 2.2.1) and the analog-to-digital-converter on the *Recha* board (as described in section 4.1) were used. In figure 5.14 the first vout cell of the right block of FHW-1.4 has been programmed to different voltage values between 0.0 V and 1.8 V in steps of 50 mV. The analog readout chain, controlling which vout cell's output will be connected to the `IBTEST` pin of the chip, was programmed accordingly. The voltages were programmed in increasing order from 0.0 V to 1.8 V, while measuring the output voltage via the on-board ADC of the *Recha* board every time a new voltage was set up. This was repeated 50 times. Afterwards the average value and its standard deviation were calculated.

Moreover, a linear fit of the voltage ramp is also depicted in figure 5.14. Its slope has been calculated to 1.04 and its offset to 0.05 V with the help of gnuplot, a free, interactive plotting software which uses "an implementation of the nonlinear least-squares (NLLS) Marquardt-

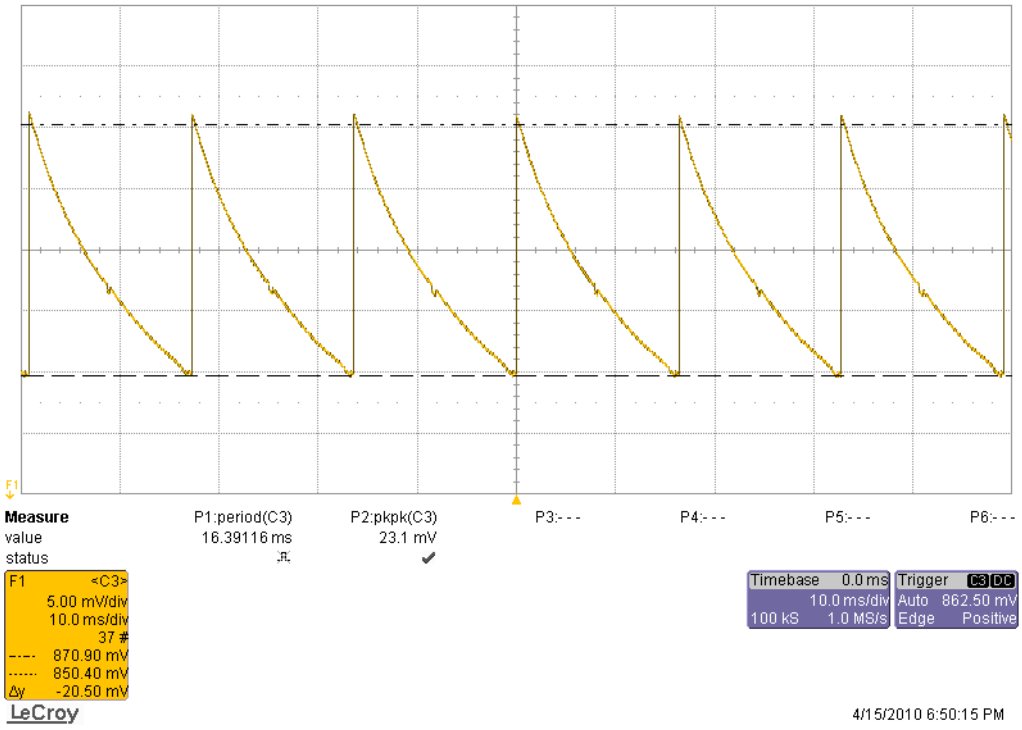

Figure 5.12: Voltage drift on the first vout cell of the right block for a parameter refresh period of approximately 16 ms. Programmed DAC value 620.

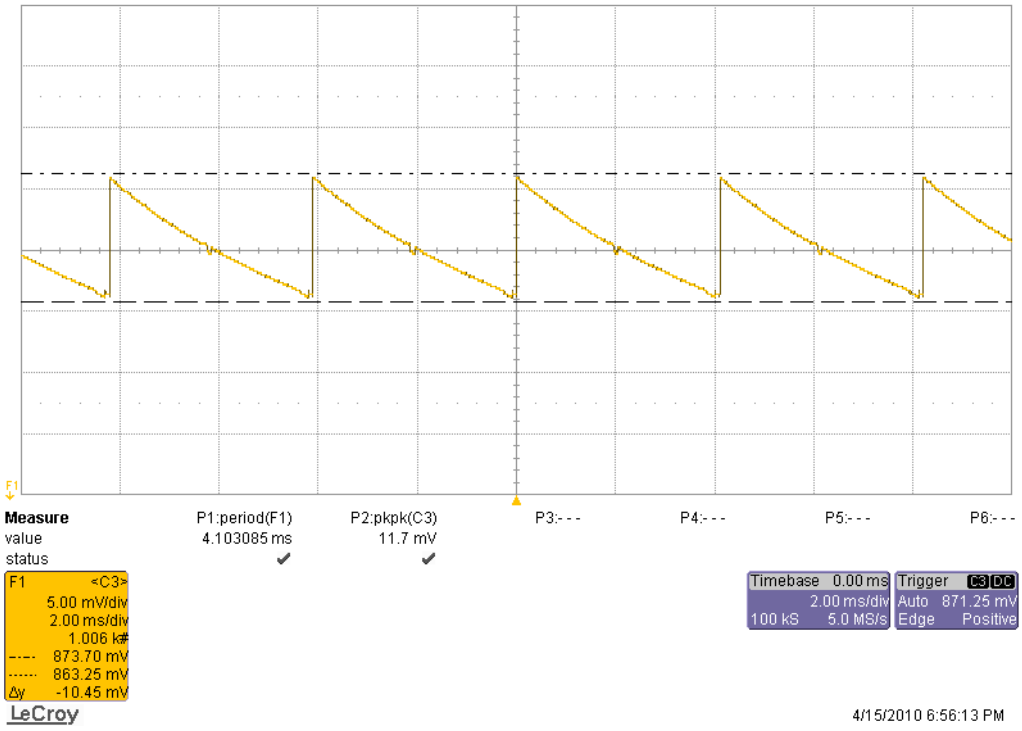

Figure 5.13: Voltage drift on the first vout cell of the right block for a parameter refresh period of approximately 4 ms. Programmed DAC value 620.

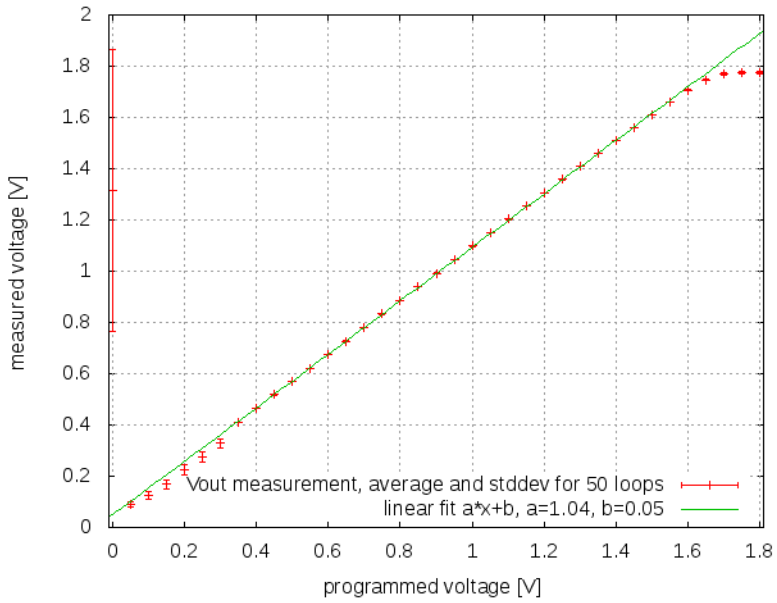

Figure 5.14: Vout measurement with FHW-1.4, Vout cell 0 of right block, 50 loops, average values and standard deviation

Levenberg algorithm"[3] to fit a given function to an arbitrary data set.

It can be seen that values larger than 1.65 V show a plateau and thus a deviation from the linear relation that should ideally be fulfilled by the voltage cells. The smallest value of 0.0 V is also beyond the dynamic range of the operational amplifier inside the vout cells, as has already been expected from simulations.

In comparison, figure 5.15 shows the same graph for FHW-1.3. Here, the dynamic range of the op-amp is much narrower, a fact that has been expected from simulations and former experiments by other members of the Electronic Vision(s) group. The slope of the linear fit has been calculated to 0.89 and the offset is 0.18 V.

All in all, the results shown here represent a remarkable step from the previous revision of the Spikey chip to the current. The linear fit has almost unity slope and a very small offset even though it has not yet been calibrated. The software framework allows for automatic calibration of the voltage generating cells by exploiting the same features of the supporting hardware that have been used here. It was implemented by Boris Ostendorf and improved by Eric Müller during their diploma theses.

5.3 Event readout priority encoder

A PyNN script that has been implemented by Daniel Brüderle was used to verify the functionality of the improved event readout priority encoder. It sets up a population of 192 neurons that receive input from 208 excitatory and 48 inhibitory neurons sending poisson spike-trains. Such a spike-train consists of a random number of neuronal events that are randomly and independently distributed with a given mean rate. In this case, the spike-trains are generated with a rate of 5 Hz in the biological time domain and the excitatory neurons are connected with a threefold higher input weight. The duration of the experiment was chosen to 3 seconds. The resulting raster plot is shown in figure 5.16 and is meant to qualitatively demonstrate that the readout deadlock problem has successfully been overcome.

[3] gnuplot official documentation [22]

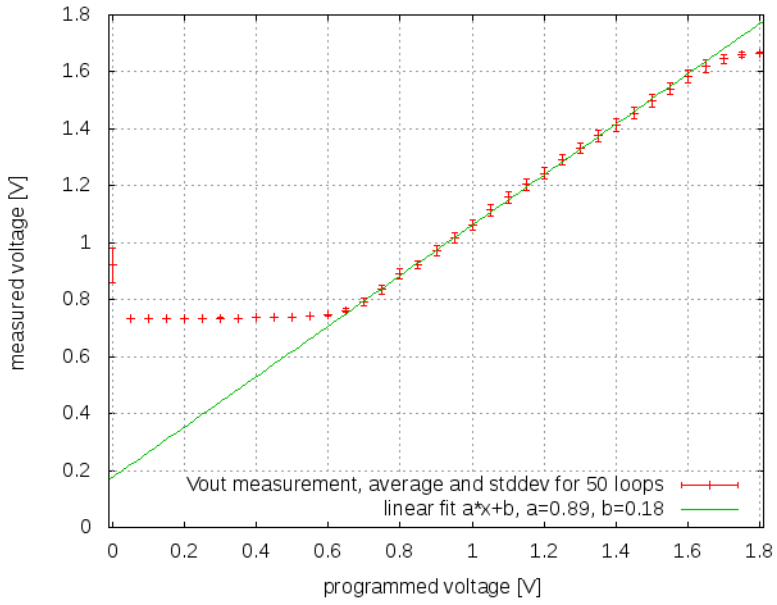

Figure 5.15: Vout measurement with FHW-1.3, Vout cell 0 of right block, 50 loops, average values and standard deviation

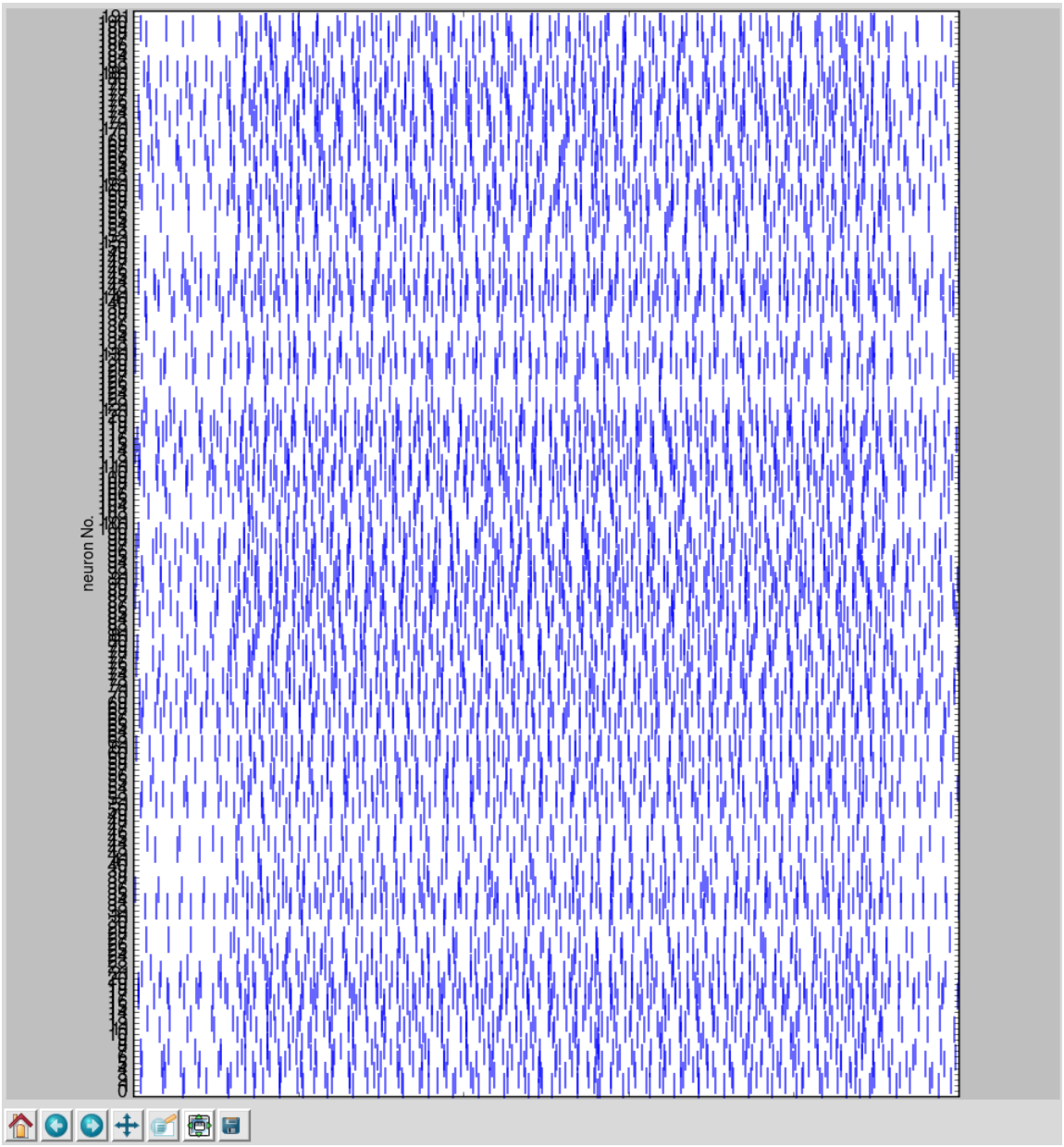

Figure 5.16: Raster plot of neuronal events in the right block of FHW-1.4. Recorded with SpikeyDemo.py.

Chapter 6

Conclusion and Outlook

The goal of the work that has been presented in this thesis, was to contribute to the improvement of a mixed-signal VLSI Network Chip and to verify the changes after the chip has been produced. The focus of the presented work was on the enhancement of the parameter voltage and current generation for the synaptic and neuronal analog circuits that have been designed by the author. Other changes that have been carried out by Dr. Johannes Schemmel and Dr. Andreas Grübl have been named and verified.

It is now possible to calibrate the parameter current generation via the `TESTPIN` of the *Spikey* chip. To do so with sufficient accuracy required changes to the design of the readout circuits on the *Recha* PCB. For more convenient measurements in the future, an ADC with a CMOS input stage and a $3\,\mathrm{k}\Omega$ resistor with a jumper will be included in a future carrier board that is worked on at the moment.

In neuroscientific simulations at present, one has always to deal with a trade-off between simulation speed and accuracy of the involved model. The main advantage of the FACETS Stage 1 hardware is that it emulates neural nets with a high speed-up factor of 10^4 compared to biological real-time. On the other hand, working with analog circuits in a $180\,\mu$m CMOS process always involves inaccuracies and mismatch between each and every neuron. It can be said that the current revision of the *Spikey* chip was a step in the direction of a more accurately operable emulation device, while keeping the advantage of high acceleration. With the event readout deadlock problem solved and by extending the dynamic range of the parameter voltage cells, real experiments have now become possible.

The different aspects of the work that has been carried out by the author are those typical for the design of a VLSI ASIC, as can be seen in 3.1. This work gave a broad and in some aspects deep introduction to the craft of microcircuit design to the author. Almost all the aspects of this trade have been covered during the presented work, from the design of full-custom analog circuits via the verification of larger units of circuits in simulations to the measurements on a chip in the laboratory. Quite some work has also been carried out to improved and extend the software that is necessary for the operation of a hardware device such as FHW-1. It turned out that the existence and maintenance of a flexible and mature software framework must not be underestimated. Therefore, a more extensive simulation and verification setup should be elaborated in the future that includes a behavioural model of the controlling FPGA. This should be implemented in an appropriate hardware description language. Modern simulation environments, like for example *UltraSim*, support the simulation of analog circuits together with digital model description in different hardware description languages and also C++.

It would also make sense to implement a fifth revision of the *Spikey* chip once the implemented STDP mechanism has been sufficiently explore in experiments. The problem that emerged during the exchange of the standard cell library, making the use of the left chip half impossible, has already been fixed in the source code and will therefore allow full-chip operation in the future.

Final remarks

Neuroscience has reached a point where a lack of sufficient computational power prevents the scalability of the systems under investigation. The aim of understanding information processing in biological systems involves the use of mathematical models of synaptic and neuronal behaviour. These models can be very complicated and can involve the use of many numerical parameters. As mentioned above, the simulation of such models, when carried out in larger networks of several hundreds or thousands of neurons and an even larger number of synapses takes much time or lacks accuracy. It is a widely known fact that number of transistors of computers obeying classical von-Neumann architectures has approximately doubled every two years in the past four decades. There are however limits to applying the standard paradigm of a processor that executes predefined instructions and operates on a separate memory when it comes to solving differential equations. Parallelization of numerical calculations on these architectures often has to deal with a bottleneck in memory bandwidth.

It is a promising approach to try to solve this problem by using dedicated hardware devices that implement neurons and synapses in an analog and parallel fashion. In these circuits every neuron and most synapses operate independently and massively parallel. There are many models of neural information processing units available that need to be tested and verified or falsified against nature. Trying to understand nature one has to repeat its behaviour in a controlled environment by doing experiments. Re-building the brain in silicon or in massively parallel operating computer architectures is a bit like building cathedrals was in the middle ages. There exists no blueprint, all there is are prototypes in form of living, thinking beings. Like it often happened with cathedrals that imploded after having been build up for years, it can happen to neuromorphic hardware projects that they fail in answering all questions about the nature of the brain. However, one should invest the effort into building biologically realistic models of the brain, be it in silicon or software.

Since dedicated hardware highly accelerates the process of building up a model and verifying it against nature, it seems to be the most promising approach at present. With the FACETS Stage 2 hardware system the FACETS project will show that it is possible to build a wafer-scale systems containing more than 100,000 *adaptive exponential integrate-and-fire* [2] neurons that operate at a speed-up factor of 10^3 to 10^4 compared to biological real-time. This system does not yet exist as a whole but all parts of it and their interconnection have already been prototyped and are being tested. By coordinating international efforts in this field and developing communication infrastructure, one can even speed up the development process of such hardware systems in the far future.

Of course, such efforts require faith in science and the power of the human brain to finally and completely understand itself. This may or may not succeed and it may or may not succeed by means of neuromorphic hardware. But no single cathedral would be visible today if nobody had believed in their fulfilling a higher purpose and the power of statical calculations in the first place.

Appendix A

A brief description of the Calibre xRC workflow

A.1 The different steps in parasitics extraction

The parasitics extraction toolchain used in this work is part of the *Calibre* IC verification software by *Mentor Graphics*. This software implements a workflow that consists of the following steps (c.f. figure A.1, which was taken from the Calibre® xRC™ User's Manual [11]):

1. During the first step a Persistent Hierarchical Database (PHDB) is created which contains information about layout connectivity and devices.
2. In the second step, the Parasitics Database (PDB) is created, containing information about the parasitic models of each net.
3. The third step generates a netlist containing the parasitic elements of each net and device.

A more detailed description of each of the above mentioned steps is given below.

The Persistent Hierarchical Database contains hierarchical geometries and the results of connectivity extraction and device recognition. It stores the information in terms of a hierarchical geometry (i.e. primitive device placements) and the so called hierarchical connectivity model (i.e. the nets connecting the devices). The recognized devices determine current flow and their placement determines where nets end.

As an input to the process of device and connectivity extraction the program requires an SVRF[1] rule file, a layout database and a source netlist. The latter is only required if the user wishes that the nets and devices in the PHDB be named according to the source netlist. These requirements are also depicted in figure A.1. An SVRF file is a set of instructions that control the devices and nets extraction process, the parasitics extraction and other verification tasks, such as Design Rule Checks (DRC) and Electrical Rule Checks (ERC).

An exemplary rule file is given in the next section. The basic principles of device recognition and connectivity extraction will be mentioned in the following paragraphs.

In a simple NMOS process, as it is assumed in the example rule file (c.f. listing A.1), a transistor is made up of two n^+ diffusion areas separated by an area of un-doped substrate that is covered by a poly-silicon gate. To be able to recognize such a device, one has to define first of all the layers that are used in the process and that are relevant for the design. Theses layers are defined using the *LAYER* keyword, as can be seen in lines 9-13 of the example rule file. Furthermore, the relevant polygons of a transistor are defined by boolean

[1] Standard Verification Rule Format

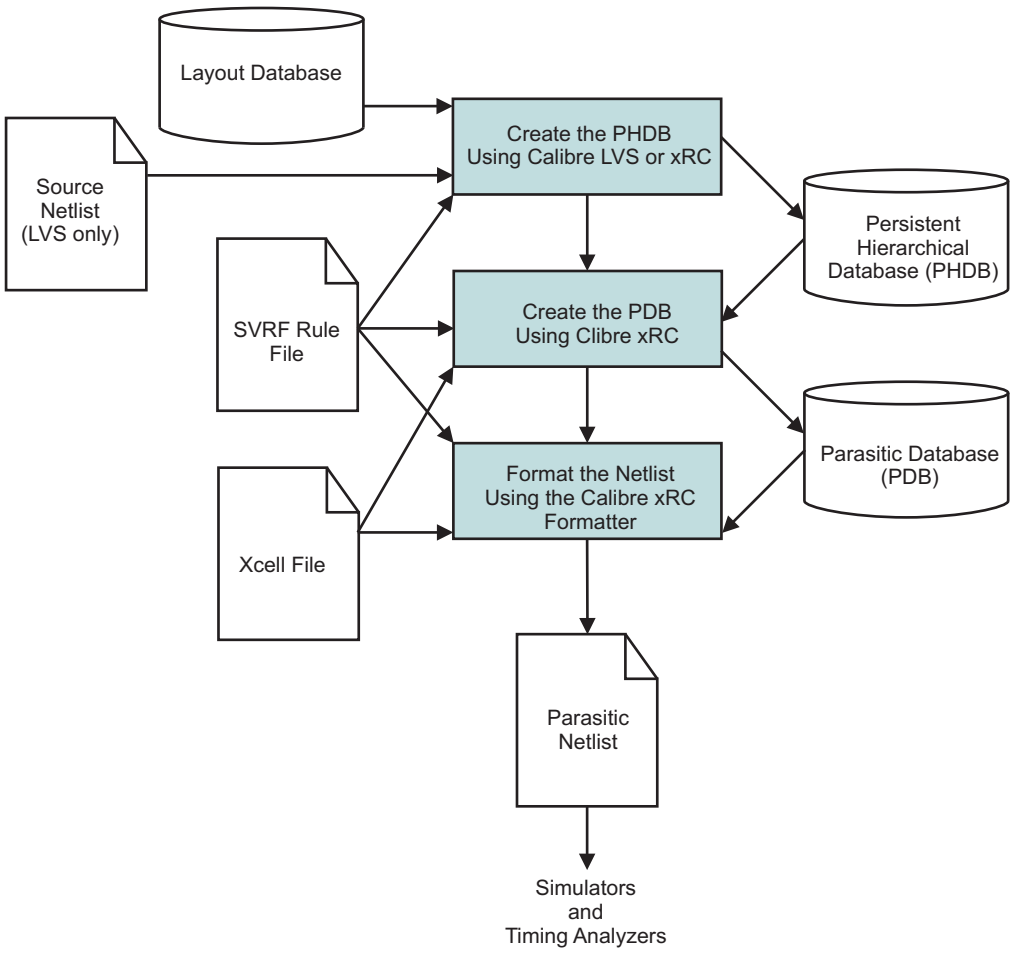

Figure A.1: Calibre® xRC™ workflow taken from [11]. It shows the three steps that are carried out by the software to extract parasitics from a GDS II layout file and the necessary input files for each step.

operations on the shapes that belong to the relevant layers. In our case the n-channel gate is defined as a diffusion shape that is covered by a poly-silicon area. The instructions, that are used to define such composite polygons, are simple boolean operators, such as *AND* or *NOT*. But there is also the possibility to use more complicated layer combinations such as *layer1 COINCIDENT EDGE layer2*. In an SVRF rule file one has to assign a name to such derived polygon layer that can be used as if it was a physical layer in the course of the file.

Once the layers and the derived polygon layers have been recognized, the software continues to apply connectivity instructions to extract the nets of the design. This is done by the CONNECT statement. In our example we connect the metal layer to the derived drain, source and gate polygons by a layer called *contact*. This way, the extraction software can identify connections between different devices that have been interconnected by routing on the metal layer.

After having prepared the derived polygon layers and having established the electrical connections between them, one can proceed to recognizing devices in the layout. This is done via the *DEVICE* statement which uses the (derived) layers, that are important for the definition of the device and its pins as its arguments. In the case of the example rule file in the next section, one would like to define an NMOS transistor that has its gate pin on the poly-silicon layer, its drain and source pin on the corresponding derived polygon layers and its bulk contact on the *pwell* layer. It is necessary to define a layer that contains the shape which serves to recognize the device. In the exemplary case one would want to use *ngate* shape. There is also the possibility to give so called auxiliary layers as a parameter to the device statement. These can only be used to perform property calculations in the statement block that follows the device statement line, as can be seen in lines 73 of the example rule file. Such calculations include width and length of the active transistor and area calculations.

The property calculation block, which is embraced by square brackets, is composed of calculations (determining for example the width and length of the device) preceded by a *PROPERTY* statement that defines which properties the device has. In our case we would like to calculate the width and length of the transistor gate and the area, perimeter and number of resistor sheets of the device's drain and source. The example rule file contains two different device descriptions.

The Parasitics Database will be described in the following paragraphs.

There are five types of parasitic capacitances that can be calculated by the Calibre xRC tool. They can further be classified in two groups.

1. The intrinsic capacitance a layer has with respect to the substrate
   - Capacitance Intrinsic Fringe
   - Capacitance Intrinsic Plate
2. The crossover capacitances of a layer with respect to other layers next to or above it
   - Capacitance Crossover Fringe
   - Capacitance Crossover Plate
   - Capacitance Nearbody

The first four of these parasitic capacitances are shown in figure A.1 together with an illustration of the shielding effect, which is caused by crossing metal lines. The Nearbody capacitance is calculated by taking into account two shapes that pass by each other on the same height with a certain distance.

The parasitic capacitances and resistances are calculated from the *PHDB* by using the geometrical properties that have been calculated by the inner block of the *DEVICE* statement.

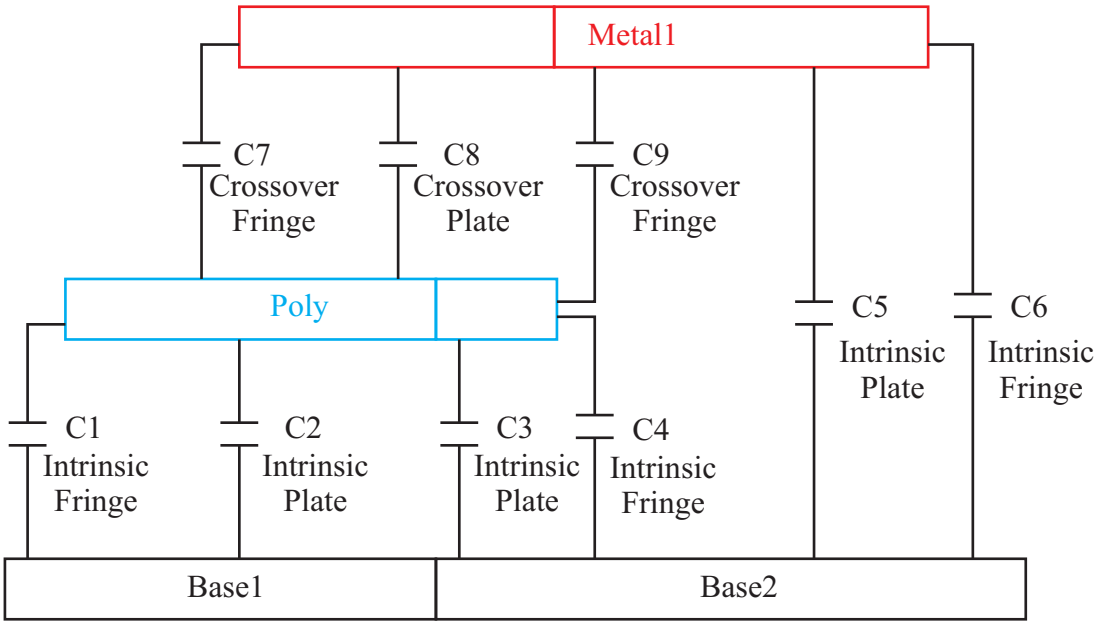

Figure A.2: Cross-Sectional view of an arbitrary semiconductor device showing to demonstrate vertical shielding, taken from [10]. It shows four different types of parasitic capacitances that can be extracted with the *Calibre Parasitics Extraction* tool.

Therefore, the widths and other physical properties like sheet resistance and permittivity of the different layers have to be given, which is normally done in a separate SVRF rule file by the foundry.

A parasitics netlist is created in the last step of the extraction tool chain. It contains a back-annotated netlist of the circuit, i.e. a netlist into which all the parasitic capacitances, inductances and resistances that have previously been extracted are included. The pins of the circuit under consideration are kept as the circuit is processed by the parasitics extraction software. Therefore, one can include the back-annotated netlist into a testbench that has already been designed before and can simulate it with a circuit simulator. However, this procedure becomes practically impossible with full-chip back-annotated netlists. The number of devices in such a file exceeds the memory resources of computers that are normally available. This is the reason why the analog part of the *Spikey* chip has never been simulated completely with parasitic devices and in conjunction with the digital part. At the moment, it is only feasible to extract parasitics partially for particular circuits and simulate these in a dedicated testbench.

A.2 Example SVRF Rule File

This example rule file was taken from [12] and is explained in more detail in section A.1. It is used to demonstrate the basic steps of device recognition and characterization and connectivity extraction.

Listing A.1: Example SVRF rule file

```
1  ////////////////////////////////////////////////////////////
2  //      Example rule file for Computing Device Parameters
3  //                NRS, NRD, AS, AD, PD, PS, L, W
4  ////////////////////////////////////////////////////////////
5  //------------------------------------------------------------
6  //This section contains example layer definitions, layer
7  //derivations, and connect statements.
8  //------------------------------------------------------------
9  layer ipoly       4     // The numbers refer to the layer numbers
10 layer diff        5     // of the GDS II input file.
11 layer contact     6
12 layer metal1      8
13 layer pwell       10
14 ngate    = ipoly AND diff
15 nsd      = diff NOT ngate
16 connect metal1 nsd ipoly by contact
17 connect pwell
18 nsd_rs1 = nsd not contact
19 nsd_rs2 = nsd_rs1 coincident edge ngate
20 nsd_rs3 = nsd_rs1 coincident edge contact
21 nsd_rs = int nsd_rs2 nsd_rs3 < 100 parallel opposite region
22 //Note, instead of 100, the above command should use the
23 //actual largest distance a source/drain contact would ever
24 //be from the edge of a gate in your process.
25 ////////////////////////////////////////////////////////////
26 //Note, there are two device descriptions below. The first
27 //addresses ordinary device configurations. The second
28 //addresses more exotic device configurations, particularly:
29 // a. Devices in which both source and drain regions have no
30 //      contacts. (example: three or more devices in series)
31 // b. Devices in which no contact resides within 100u (or
32 //      whatever distance you specify) from gate edge.
33 // c. Devices in which no contact has any edge facing the
34 //      gate edge. If you find any computation wherein you
35 //      require more accuracy, contact Mentor Graphics
36 //      Customer Support.
37 ////////////////////////////////////////////////////////////
38 ////////////////////////////////////////////////////////////
39 //                    Device Description Example 1
40 //
41 //Note, the auxiliary layers "diff" and "nsd_rs" are added
42 //to this first device description; use diff
43 //in the AS, AD, PS, PD property computations below to
44 //account for shared source/drain; use nsd_rs in the
45 //NRS, NRD property computations.
46 ////////////////////////////////////////////////////////////
```

```
47 device mn ngate ipoly(G) nsd(S) nsd(D) pwell(B) <diff>
48     <nsd_rs>
49 [
50     property W, L, AD, AS, PD, PS, NRS, NRD
51     bend_effect = 0.5
52     //-------------------------------------------------------------
53     //This section of the property computations measures gate
54     //length and width. The "if" clause accounts for any bends
55     //that can exist in the gates.
56     //-------------------------------------------------------------
57     W = perimeter_coincide(ngate, nsd) / 2
58     L = (perim(ngate) - perimeter_coincide(ngate, nsd)) / 2
59       if (bends(ngate) > 0)
60       {
61         if (W > L)
62           W = W - bend_effect * bends(ngate) * L
63         else
64           L = L - bend_effect * bends(ngate) * W
65       }
66     //-------------------------------------------------------------
67     //This section of the property computations measures Area of
68     //Source and Area of Drain, even in cases of shared
69     //source/drain. Note, because the capacitance effects of AS
70     //and AD are a function of source/drain area and perimeter, AS
71     //and AD are not affected by bends in the source/drain regions.
72     //-------------------------------------------------------------
73     AS = area(S) * (W / perimeter_inside(S, diff))
74     AD = area(D) * (W / perimeter_inside(D, diff))
75     //-------------------------------------------------------------
76     //This section of the property computations measures
77     //Perimeter of Source and Perimeter of Drain, even in cases
78     //of shared source/drain. Note, since the capacitance effects
79     //of PS and PD are a function of source/drain perimeter, PS
80     //and PD are not affected by bends in the source/drain regions.
81     //-------------------------------------------------------------
82     PS = perimeter(S) * W / perimeter_inside(S, diff)
83     PD = perimeter(D) * W / perimeter_inside(D, diff)
84     //-------------------------------------------------------------
85     //This section of the property computations measures Number of
86     //Resistance Squares in Source and Number of Resistance
87     //Squares in Drain, in terms of a first order approximation.
88     //Note:
89     //1. The following calculations use edges of contacts
90     //      instead of centers of contacts. That is,
91     //        NRS = average distance from gate to contact's nearest
92     //        edges / width of gate.
93     //2. The following calculations fully account for relative
94     //      placement of contacts to gate and to each other, with
95     //      the single exception that contacts that have no edges
96     //      that face the gate edge are not involved in the
97     //      calculation.
98     //3. Calculations assume all contacts are equally sized.
```

```
99      SUM_S_LENGTH = perimeter_inside(nsd_rs, S) - perimeter_coincide(nsd_rs, S)
100     COUNT_S = trunc((count(nsd_rs) * perimeter_coincide(nsd_rs, S)
101                     / perimeter_coincide(nsd_rs, G)) + 0.5)
102       if (count_S != 0) {
103           NRS = SUM_S_LENGTH / COUNT_S / W / 2
104       }
105       else {
106           NRS = AS / (W * W)
107       }
108     SUM_D_LENGTH = perimeter_inside(nsd_rs, D) - perimeter_coincide(nsd_rs, D)
109     COUNT_D = count(nsd_rs) - COUNT_S
110       if (COUNT_D != 0) {
111           NRD = SUM_D_LENGTH / COUNT_D / W / 2     }
112       else {
113           NRD = AD / (W * W)
114       }
115 ]
116 ////////////////////////////////////////////////////////////
117 //                    Device Description Example 2
118 //
119 //This second device description should be used along with
120 //the first if your design has any of the following device
121 //configurations:
122 // a. Devices in which both source and drain regions have no
123 //        contacts (example: three or more devices in series).
124 // b. Devices in which no contact resides within 100u (or
125 //        whatever distance you specify) from gate edge.
126 // c. Devices in which no contact has any edge facing the
127 //        gate edge example: contact resides in source/drain
128 //        "dog-leg."
129 //Note, the auxiliary layer diff is added to this
130 //device description; use diff in the AS, AD, PS, PD
131 //property computations below to account for shared source/drain.
132 ////////////////////////////////////////////////////////////
133 device mn ngate ipoly(G) nsd(S) nsd(D) pwell(B) <diff>
134 [
135   // property W, L, AD, AS, NRD, NRS, PD, PS
136   //The line above is commented out until NRD and NRS computations are added
137   //to prevent syntax error upon compiling rule file.
138   property W, L, AD, AS, PD, PS, NRS, NRD
139   bend_effect = 0.5
140   //-------------------------------------------------------------
141   //This section of the property computations measures gate
142   //length and width. The if clause accounts for any bends
143   //which can exist in the gates.
144   //-------------------------------------------------------------
145   W = perimeter_coincide(ngate, nsd) / 2
146   L = (perim(ngate) - perimeter_coincide(ngate, nsd)) / 2
147     if (bends(ngate) > 0)
148     {
149         if (W > L)
150           W = W - bend_effect * bends(ngate) * L
```

```
151           else
152             L = L - bend_effect * bends(ngate) * W
153       }
154     //-------------------------------------------------------------------
155     //This section of the property computations measures area of
156     //source and area of drain, even in cases of shared
157     //source/drain. Note, since the capacitance effects of AS and
158     //AD are a function of source/drain area and perimeter, AS
159     //and AD are not affected by bends in the source/drain
160     //regions.
161     //-------------------------------------------------------------------
162     AS = area(S) * (W / perimeter_inside(S, diff))
163     AD = area(D) * (W / perimeter_inside(D, diff))
164
165     //-------------------------------------------------------------------
166     //This section of the property computations measures
167     //perimeter of source and perimeter of drain, even in cases
168     //of shared source/drain, Note, since the capacitance effects
169     //of PS and PD are a function of source/drain perimeter, PS
170     //and PD are not affected by bends in the source/drain regions.
171     //-------------------------------------------------------------------
172     PS = perimeter(S) * W / perimeter_inside(S, diff)
173     PD = perimeter(D) * W / perimeter_inside(D, diff)
174     //-------------------------------------------------------------------
175     //This section of the property computations measures number of
176     //resistance squares in source and number of resistance
177     //squares in drain. Note, in the case where neither the source
178     //nor drain has any contacts,
179     //NRS = area of source / width / width
180     //That is, average length of AS = AS / W
181     //number of resistance squares = average length of AS / W
182     //-------------------------------------------------------------------
183     NRS = AS / (W * W)
184     NRD = AD / (W * W)
185   ]
```

Bibliography

[1] Johannes Bill. Self-stabilizing network architectures on a neuromorphic hardware system. Diplomarbeit, Universität Heidelberg, 2008.

[2] R. Brette and W. Gerstner. Adaptive exponential integrate-and-fire model as an effective description of neuronal activity. *Journal of Neurophysiology*, 94:3637–3642, 2005.

[3] D. Brüderle, E. Müller, A. Davison, E. Muller, J. Schemmel, and K. Meier. Establishing a novel modeling tool: a python-based interface for a neuromorphic hardware system. *Front. Neuroinform.*, 3, 2009.

[4] Daniel Brüderle. Neuroscientific modeling with a mixed-signal vlsi hardware system. Dissertation, Universität Heidelberg, 2009.

[5] The Hypertransport Consortium. `http://www.hypertransport.org`, 2010.

[6] Analog Devices. *AD7924 Data Sheet*. 2009. `http://www.analog.com/static/imported-files/Data%5FSheets/AD7904%5F7914%5F7924.pdf`.

[7] Analog Devices. *AD8063 Data Sheet*. 2010. `http://www.analog.com/static/imported-files/data%5Fsheets/AD8061%5F8062%5F8063.pdf`.

[8] Simon Friedmann. Extending a hardware neural network beyond chip boundaries. Diplomarbeit, Universität Heidelberg, 2009.

[9] R. L. Geiger, P. E. Allen, and N. R. Strader. *VLSI - Design techniques for analog and digital circuits*. McGraw-Hill, 1990.

[10] Mentor Graphics, 2007. SVRF Capacitance and Resistance Statements — Concepts and Usage.

[11] Mentor Graphics, 2008. Calibre® xRC™ User's Manual.

[12] Mentor Graphics, 2008. Calibre® Verification User's Manual.

[13] Electronic Vision(s) group of the Kirchoff Institute for Physics Heidelberg. Listing of latest changes made in the design of the spikey asic. `http://www.kip.uni-heidelberg.de/repos/VISION/project/spikey/revision.txt`.

[14] Andreas Grübl. Vlsi implementation of a spiking neural network. Dissertation, Universität Heidelberg, 2007.

[15] A. Hastings. *The ART of ANALOG LAYOUT*. Prentice Hall, 2001.

[16] Keithley Instruments. *2100, 6 1/2-Digit USB Digital Multimeter*. 2010. `http://www.keithley.de/data?asset=50757`.

[17] Céondo Ltd. Indefero code management system. `http://www.indefero.net/`.

[18] Sebastian Millner. An integrated operational amplifier for a large scale neuromorphic system. Diplomarbeit, Universität Heidelberg, 2007.

[19] Eric Müller. Operation of an imperfect neuromorphic hardware device. Diplomarbeit, Universität Heidelberg, 2008.

[20] Boris Ostendorf. Charakterisierung eines neuronalen netzwerk-chips. Diplomarbeit, Universität Heidelberg, 2007.

[21] Stefan Philipp. Design and implementation of a multi-class network architecture for hardware neural networks. Dissertation, Universität Heidelberg, 2008.

[22] Gnuplot 4.4 An Interactive Plotting Program. `http://www.gnuplot.info/docs%5F4.4/gnuplot.pdf`, 2010.

[23] The WAF project. Waf - the flexible build system. `http://code.google.com/p/waf/`.

[24] The FACETS project website. `http://facets.kip.uni-heidelberg.de/public/`, 2010.

[25] J. Schemmel, D. Brüderle, K. Meier, and Ostendorf B. Modeling synaptic plasticity within networks of highly accelerated i&f neurons. In *Proceedings of the 2007 IEEE International Symposium on Circuits and Systems (ISCAS'07)*, page 3367–3370, 2007.

[26] J. Schemmel, A. Grübl, K. Meier, and E. Müller. Implementing synaptic plasticity in a vlsi spiking neural network model. In *Proceedings of the 2006 International Joint Conference on Neural Networks (IJCNN'06)*, pages 1–6, 2006.

[27] J. Schemmel, K. Meier, and E. Müller. A new vlsi model of neural microcircuits including spike time dependent plasticity. In *Proceedings of the 2004 International Joint Conference on Neural Networks (IJCNN'04)*, volume 3, pages 1711–1716, 2004.

[28] Agilent Technologies. *Using Linux to Control USB Instruments.* 2010. `http://www.home.agilent.com/agilent/redirector.jspx?action=ref&cname=AGILENT%5FEDITORIAL&ckey=1189335&lc=eng&cc=US&nfr=-33861.0.00`.

[29] Xilinx. *Virtex-II Pro and Virtex-II Pro X Platform FPGAs: Complete Data Sheet.* 2007.

Acknowledgements

I wish to express my appreciation and gratitude to all persons who contributed in any way to this thesis.

All members of the Electronic Vision(s) group for an incredible work atmosphere, general support and openness.

Prof. Dr. Karlheinz Meier for having given me the opportunity to work in this field of research and his commitment to the FACETS project.

Dr. Johannes Schemmel for supervision and for his invaluable advice and knowledge.

Prof. Dr. Norbert Herrmann for providing a second opinion for this thesis.

Dr. Andreas Grübl for patient explanations, introducing me to the tricks of the trade and his sense of humor.

Eric Müller and Markus Dorn for helping me on innumerable computer problems.

Ralf Achenbach for his support with lab facilities.

All Hardies and Softies for their patience and general support.

Kathrin Aguilar Ruiz-Hofacker and Simone Schumacher at INI Zürich for making it possbible for me to submit this thesis on time.

Linn for her love and general support during the developpement of this thesis.

My family.

Statement of Originality (Erklärung):

I certify that this thesis, and the research to which it refers, are the product of my own work. Any ideas or quotations from the work of other people, published or otherwise, are fully acknowledged in accordance with the standard referencing practices of the discipline.

Ich versichere, dass ich diese Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

Heidelberg, 03.05.2010

.....
(signature)