RUPRECHT-KARLS-UNIVERSITÄT HEIDELBERG

Alexander Kononov

Testing of an Analog Neuromorphic Network Chip

Diplomarbeit

HD-KIP-11-83

KIRCHHOFF-INSTITUT FÜR PHYSIK

Faculty of Physics and Astronomy

University of Heidelberg

Diploma thesis

in Physics

submitted by

Alexander Kononov

born in Iskitim, Russia

October 10, 2011

Testing of an Analog Neuromorphic Network Chip

This diploma thesis has been carried out by Alexander Kononov at the

Kirchhoff Institute for Physics

Ruprecht-Karls-Universität Heidelberg

under the supervision of

Prof. Dr. Karlheinz Meier

Testing of an Analog Neuromorphic Network Chip

This thesis describes testing of the prototype system of the neuromorphic hardware for the BrainScales project. The tests were focused on the core mixed-signal component of this wafer-scale system: the HICANN chip.

Particularly the floating gate memory cells that store the analog neuron parameters, as well as the synaptic circuits that emulate behavior of biological synapses, underwent tests concerning their functionality, precision and efficiency.

As a result of testing, the accuracy of subject components was quantified. Also, the methods for optimizing and stabilizing the programming accuracy, speed and voltage range of the floating gate memory cells were developed.

Additionally, the existing test software was extended by several modules, which now provide simple and efficient control over many functions of the chip.

In the course of the testing process, many of the chip's previously untested functionality could be verified and analyzed.

Test eines Analogen Neuromorphen Netzwerkchips

Die vorliegende Arbeit befasst sich mit dem Testen des Prototypsystems der neuromorphen Hardware des BrainScales Projekts. Schwerpunkt der Tests war das zentrale Element des Systems: der HICANN-Chip.

Insbesondere wurden die Floating-Gate-Speicherzellen, die zur analogen Speicherung der Neuronenparameter dienen, und die synaptischen Schaltungen, die biologische Synapsen emulieren, auf ihre Funktion, Präzision und Effizienz untersucht.

Als Resultat der Arbeit wurde die Genauigkeit der untersuchten Komponenten quantifiziert und Methoden entwickelt, um die Präzision, Schnelligkeit und den Spannungsbereich der Programmierung von Floating-Gate-Speicherzellen zu verbessern und zu stabilisieren.

Zudem wurde die existierende Test-Software um mehrere funktionelle Module erweitert, die eine einfache Ansteuerung vieler Chip-Funktionen ermöglicht.

Während der oben genannten Tests wurden einige weitere bisher ungetestete Funktionen des Chips überprüft und analysiert.

Contents

1 Introduction

Over 130 years have passed since Heinrich Wilhelm Waldeyer firstly introduced the term "neuron" and recognized it to be the fundamental element of the nervous system and thus the basic building block of sensory and motivational systems in the animal world [1].

Neuroscience has come a long way since then. The mechanisms of neuronal activity have been investigated: from information acquirement such as in retina of an eye, to its propagation via axons, dendrites and synapses, to its processing via neuronal networks. The first two mechanisms have been closely studied by biologists over the last century and are well known by now. It is the intricate single neuron's design that is making it possible for sunlight to induce an ion flow inside highly electrically polarized rods and cones of retina cells, and for this ion current to be transported over the dendrites to the synapses and eventually to other neurons.

The information processing in neuronal networks, on the other hand, remains a highly mysterious mechanism. Its secrets are believed to lie in topologies of these formations [2]. So uncovering these secrets has become the most important task of neuroscientific research, because this could provide an answer to a question that every human being is wondering about: "Why can I think?"

The human brain contains up to 10^{12} neurons. So it is impossible to just record the behavior of every single one and analyze this data. A logical way to approach this kind of problem would be to construct a mathematical model of smaller brain parts, verify it, and then take on the more complex tasks using previously accumulated knowledge.

In the past decades several single neuron models have been developed, such as Hodgkin-Huxley Model [3] or AdEx[1][4]. These are presently being used for constructing artificial neuronal networks in order to understand their functionality [5]. However, there are many questions on this way, one of which is the implementation method.

The straight-forward solution would be simulating neuronal networks in software. Nowadays there is a variety of software tools [6, 7, 8], designed to build up and simulate artificial neural networks, but all of them share the same weak point: the demand on computational resources for these tools grows enormously with simulated network's size. This is why it is economically unfeasible to perform large counts of simulations with large networks in a short period of time.

BrainScaleS project [9] is an initiative of the European Union, aimed to overcome these difficulties by using neuromorphic hardware[2] as opposed to software simulations. BrainScaleS unites 13 groups of scientists from different fields of study. Neurobiologists who extract biological data from in-vivo neural networks, modelers who develop feasible

[1] Adaptive Exponential Integrate-And-Fire Model.

[2] Semiconductor-based systems, capable of mimicking (emulating) the functioning of neurons and neuronal networks by using analog circuitry to resemble neurons.

mathematical models of these networks, computer scientists and engineers who design and build devices that are capable of reproducing the behavior of biological neuronal networks: the neuromorphic hardware.

The advantages of hardware emulations are low power consumption, high acceleration of experiments and large scalability [10]. There are also disadvantages such as process inaccuracy of semiconductor production facilities. They lead to the necessity of complex calibrations of the new hardware and estimation of the final error of the resulting emulations.

Strictly speaking, biological neurons also possess errors, so one must not try to reproduce exact modeled neurons, but contain the final error in biologically reasonable boundaries. One of thematic priorities of the present thesis is evaluation of the final errors of neuron parameters and synaptic weights. The other one is workflow optimization, taking these errors into account, and error containment.

Outline

Chapter 2 of this thesis handles the essential knowledge. It will briefly present the concept behind neuromorphic hardware and a more detailed view over the single components that were used to carry out the tests.

In chapter 3 the task definitions, concepts and exact methods of testing are depicted. The first part is dedicated to testing of the floating gate memory that functions as analog parameter storage for the neuron circuits. The second part handles tests of the synapse circuits. The work preceding the tests, such as software development and hardware setup, is also described.

Chapter 4 presents the results of single tests in the same order as chapter 3.

The discussion of the experiments and an outlook on improvement of test concepts and the hardware in general are located in the last chapter.

2 Neuromorphic Hardware

This chapter will introduce the reader to the wafer-scale neuromorphic system [11, 10, 12] and its prototype test setup. The purpose of the entire project will also be shortly depicted.

The course of introduction will begin with a general overview of the whole system. As the chapter progresses, the single components will be explained in more detail, focusing on those directly subjected to this thesis.

2.1 Wafer-Scale Neuromorphic System

The BrainScaleS project was initiated by the European Union. Its goal is to find and evaluate novel solutions to existing neuroscientific problems such as the limited resources in computational neuroscience. It is done with an outlook on better understanding the mechanisms of different scales of brain functions: from basic neuron-to-neuron communication to complex high-order network activity. Thus the name BrainScaleS.

The core of the project is the *Hybrid Multiscale Facility* (HMF), which is being developed at the University of Heidelberg and TU Dresden. In its final state the HMF will be able to emulate networks of up to 1.5 Million AdEx neurons (see 2.2.1) with over 350 Million synapses. Remarkable is also the emulation speed which is expected to be between 10^3 and 10^5 times the speed in real biological networks.

Figure 2.1 is depicting the HMF, which will consist of up to 8 *Wafer-Scale Neuromorphic System* modules. Each of those operates an uncut 20 cm silicon wafer, consisting of 384 interconnected mixed-signal ASICs[1] called HICANN[2] chips [13], each of which contains 114,628 physical synapse circuits and 512 so-called *denmems*[3] that emulate the workings of an AdEx neuron. The wafer is attached to a custom-made PCB[4] [14] via elastomer connectors. This PCB provides power and access to the communication devices for the wafer. These communication devices, called DNCs[5] [15] each render the communication between 8 HICANN chips on the wafer and the rest of the System, making a total of 48 DNCs needed to manage the on-wafer data flow. To implement a fast DNC-interconnection, 12 FPGAs[6] are programmed to provide buffering and routing of data packets between DNCs. More information on HMF communication follows in (2.3).

[1] Application-Specific Integrated Circuit.
[2] High Input Count Analog Neural Network.
[3] Dendritic membrane circuits.
[4] Printed Circuit Board.
[5] Digital Network Chips, developed at TU Dresden.
[6] Field-Programmable Gate Arrays.

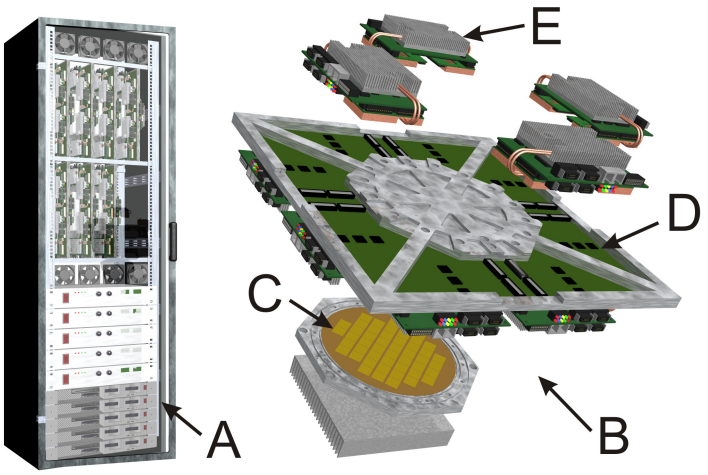


Figure 2.1: Hybrid Multiscale Facility: (A) HMF, assembled in a standard server rack, (B) Wafer-Scale Neuromorphic System, (C) Silicon wafer, containing 384 HICANN chips, (D) PCB and metal mounting bracket, (E) FPGA board with 4 DNCs mounted. Pictures by Dan Husmann de Oliveira.

Depending on the experimental demands, a corresponding neuronal network can be mapped onto the HMF, meaning that single neurons are built by interconnecting the denmems (see 2.2.1) and setting up the communication network on the wafers, DNCs and FPGAs, so that source neuron output data is routed correctly to the corresponding synapse circuits of the target neuron. Also, the neuron parameters have to be programmed so that the hardware neurons behave as intended. For this purposes, a software tool is being developed [16] to make setting up of neuronal networks on hardware user-friendly. Via this software one will be able to set up the hardware network, stimulate it, and read back the response. This data can then be evaluated using common neuroscientific methods.

2.2 HICANN: Mixed-Signal ASIC of the Wafer-Scale System

At the core of the wafer-scale system lies the HICANN chip. It is the smallest part of the framework, which can function as a standalone neuromorphic system. The chip takes up a surface of $0.5\,\mathrm{cm}^2$ and can be roughly divided into 4 main areas, as shown in fig. 2.2: the denmem-circuits [17], the floating gate memory cells [18], the synapse circuits [17] and the Layer 1 (L1) communication bus system [10]. These components are described in more detail in (2.2.1), (2.2.2), (2.2.4) and (2.2.3).

One HICANN chip is able to emulate up to 512 AdEx neurons and possesses enough connectivity resources to build up an almost arbitrary network, overarching many single chips. Also, each HICANN chip will have its own STDP[7]-processor to implement the

[7]Spike-Timing-Dependent Plasticity.

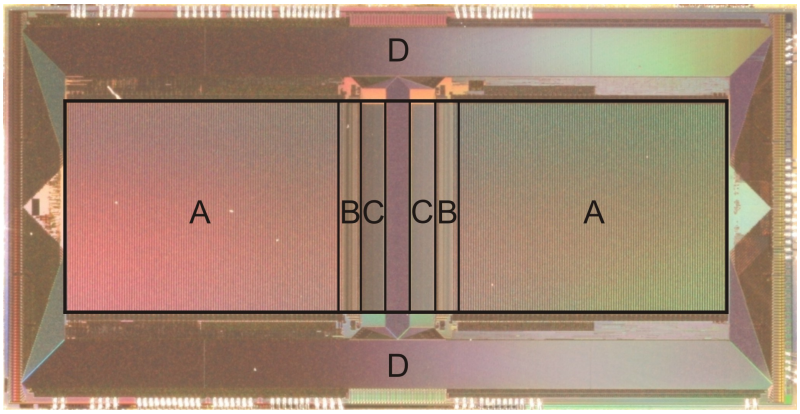


Figure 2.2: Photograph of a HICANN chip: (A) Synapse arrays, (B) Denmem circuits, (C) Floating gate memory cell arrays, (D) L1 network and standard logic. Picture by Electronic Vision(s).

learning mechanism of spike-timing dependent plasticity [19, 20].

One HICANN chip can communicate with others in two different ways, depending on the routing preferences: via DNC and parallel packet-based Layer 2 protocol [21, 15], or directly via differential wire pairs, integrated into the wafer and serial L1 protocol [10]. The characteristics of these protocols are briefly described in (2.2.3) and (2.3).

To the date of issuing of this thesis, two versions of HICANN chips have been produced and tested by Electronic Vision(s) group [22] at the University of Heidelberg. Initial version (HICANN v1, 2009) was improved in 2011 (HICANN v2). The differences that were subjects to the matter of this thesis are described below, for more information the reader is referred to [13]. Also, the results of the tests, carried out on both chips in the course of presented work, are specifically compared to verify the improvements of the second version of the chip as opposed to the first one.

2.2.1 Hardware Neuron

Located in the central part of the chip (fig. 2.2), there are two rows of 256 large circuits called *denmems*, making a total of 512. Each of them is connected to a string of 224 synapse circuits. These denmems, when provided with correct input, can emulate the behavior of AdEx neurons, described by Brette and Gerstner in [4] as follows:

$$C\frac{dV}{dt} = -g_L(V - E_L) + g_L\Delta_T exp\left(\frac{V - V_T}{\Delta_T}\right) - g_e(t)(V - E_e) - g_i(t)(V - E_i) - w \quad (2.1)$$

$$\tau_w\frac{dw}{dt} = a(V - E_L) - w \quad (2.2)$$

$$\text{At spike time} : V \to E_L;\ \ w \to w + b \quad (2.3)$$

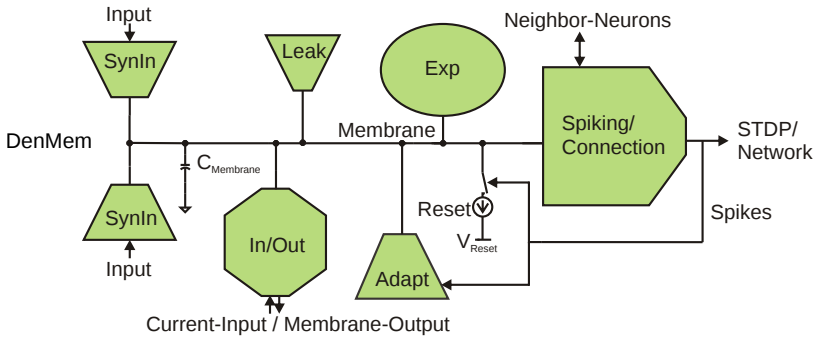


Figure 2.3: Block diagram of a denmem circuit: each unit represents a term in the AdEx equation. Figure taken from [12].

- V - membrane potential
- C - membrane capacitance
- g_L - leak conductance
- E_L - leak reversal potential
- V_T - spike threshold
- Δ_T - slope factor
- w - adaptation current
- τ_w - adaptation time constant
- a - subthreshold adaptation
- b - spike-triggered adaptation

AdEx is derived from standard leaky integrate-and-fire model [23] by adding two extra terms. It provides reasonable description of neuronal activity in mathematical terms by taking into account not only the synaptic and the leakage currents, but also an adaptation mechanism, which is modeled as an exponentially falling current and is adjusted at conditions, described by equations 2.2 and 2.3. One more additional term is resembling the biological spike mechanism and is modeled as an exponential function, applied to the membrane voltage (see eq. 2.1). The synaptic conductances $g_i(t)$, $g_e(t)$ are usually modeled as exponential or alpha-functions[8].

AdEx model has been chosen for use in BrainScaleS hardware because of its relative simplicity and ability to reproduce many of the firing patterns of biological neurons, such as bursting or spike-frequency-adaptation [24]. Also, the amount of chip area, needed to implement AdEx in silicon is fairly feasible compared to more complicated models.

Basically, a denmem circuit consists of several ion-channel emulation circuits, connected to a capacitance that represents the neuron membrane (see fig. 2.3).

Each of the ion-channel circuits stands for a term in the AdEx equation (eq. 2.1). The currents, produced by these circuits, cause the voltage on the membrane capacitance to shift, thusly emulating the workings of a biological neuron.

Depending on the experimental demands, up to 64 denmems can be interconnected to form a neuronal circuit with more than 224 synaptic inputs. This way neurons with up to 14,336 synapses are realizable in hardware.

[8] Alpha function has the form: $g(t) = g_{max} \frac{t}{\tau} exp\left(\frac{\tau - t}{\tau}\right)$.

Additional features of hardware neurons, used for testing purposes, include the ability to connect to a test current generator (which yields the simplest way to stimulate the analyzed neuron), and the ability to read out the membrane voltage by using one of the 2 analog output lines of the HICANN chip. This allows observation of the membrane potential by using an oscilloscope.

As in the model, each neuron is configured by setting certain parameters to meaningful values. These parameters are realized as external voltages and currents, connected to the denmem circuits. They are stored in blocks of *floating gate memory cells* (see section 2.2.2). The parameters of hardware neurons are not directly derivable from AdEx, therefore a series of complex calibrations is required, during which the neuronal behavior in hardware is observed and fitted to the model by varying the hardware parameters.

2.2.2 Floating Gate Memory Cells

To be able to mimic AdEx neurons, each denmem requires 21 adjustable parameters: 12 voltages and 9 currents. Therefore, an entity is needed to store these parameters. This purpose is served by four blocks of analog floating gate memory cells, first described in [18]. They were optimized for HICANN integration in [25]. One memory block contains 24 lines with 129 cells each. Even numbered lines contain voltage parameters, odd lines contain currents. One line stores one parameter, e.g. the firing threshold, for 128 neurons. The remaining 129th value is a global parameter, such as a reset voltage or a bias current, serving to configure other chip components.

The floating gate technology has been chosen for deployment in HICANN chip for several reasons. Two major ones are chip area and power saving. One cell requires less than 200 μm^2 of chip area (less than the space taken up by an 8-bit DAC[9], which could serve as an alternative), and there is no need to refresh the value after it is programmed once, because the lifetime of charge on the floating gate is longer than the foreseeable length of an experiment on the final wafer-scale system.

Functional Principle of a Single Cell

Figure 2.4 shows the schematic diagram of a floating gate voltage cell. The three transistors with interconnected gates form the name-giving floating gate. Two transistors, named CGL[10] and CGS[11], have their sources shorted with drains, which makes them function as capacitors, coupled to the floating gate. CGL is 10 times larger than CGS. Therefore, CGL couples to the floating gate much stronger, thus allowing for the directed programming of the cell. The third transistor sharing its gate with the control gate transistors, is responsible for reading out the cell, as the voltage drop on the fourth transistor (schematized below the third transistor) is proportional to the potential on the floating gate (source follower). The operating point of the source follower is adjusted by applying

[9] Digital-Analog Converter.
[10] Control Gate Large.
[11] Control Gate Small.

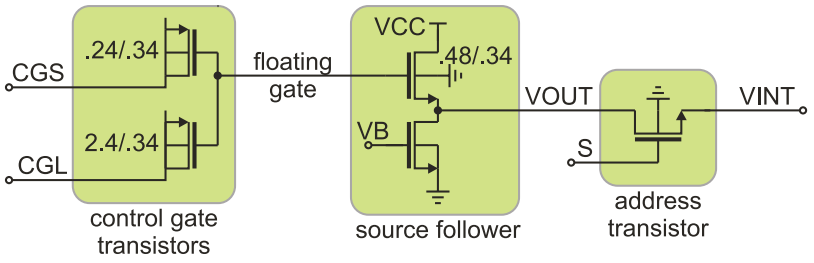


Figure 2.4: Schematic diagram of a floating gate voltage cell. The control gate transistors are used to control the programming process, the source follower transforms the transistor current into voltage. Read out is performed by activating the address transistor. Figure taken from [25].

bias voltage to the VB gate. The fifth transistor switches the output of the cell on and off.

All in all there are two types of cells, categorized by their output: the current cells and the voltage cells. As seen in fig. 2.4, the current through an output transistor of a voltage cell is converted to voltage by a source follower. This value yields a neuron parameter. For the current cells the same output transistor current is applied to a current mirror that employs two outputs. The first output is connected to the corresponding neuron, while the second one multiplies this current by four[12] and channels it through a resistor. The voltage drop over this resistor yields the external output of the cell[13]. The properties of both kinds of cells are different due to design differences, so most of the tests described in section (3.2) are performed separately for each of them.

Cells, deployed in the HICANN chip, are designed to store voltages from 0 to 1.8 V with an accuracy of ± 4 mV. To reach the relatively high voltage of 1.8 V, control voltages of up to 11 V are required. The accuracy is achieved through using series of short pulses to charge the gate. One of the goals of this thesis is to verify the accuracy and to optimize the time, needed to program the cells.

The functioning of a single cell is best explained with an example. To program a cell from 0 V output to a higher value, CGL has to be set to 0 V, CGS to 11 V. The strong coupling between CGL and the floating gate pulls the latter down, so the potential difference between the floating gate and CGS increases to a point where electrons tunnel from the floating gate through the insulator to CGS, charging the floating gate positively and causing permanent potential shift, remaining after CGL and CGS are set to idle (equal) values. After such charging pulse, the readout transistor is switched on and the output voltage is compared to the nominal value. A 10 bit DAC and a comparator are used to perform this task. If the desired voltage is reached, the process is stopped, otherwise another charging pulse is performed.

[12] This is done to ensure faster capacitance charging whenever the cell is read out.

[13] This voltage is e.g. compared to a nominal value during the programming process in order to determine if the cell has reached its desired value. Later in this thesis, this voltage is also used as the readout value of the floating gate current cells. This is why the output values of current cells are voltages.

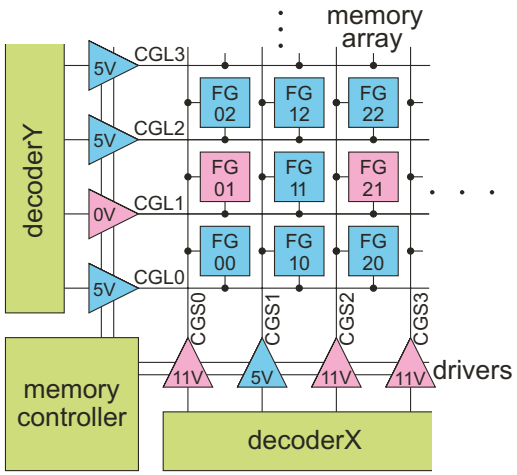


Figure 2.5: Programming of a floating gate array. Address decoders choose the driven wires. Cells are connected to high-voltage drivers line- and column-wise. Writing voltages are 0 and 11 V, the idle voltage is 5 V. Figure re-drawn from [25].

Array of Analog Floating Gate Memory Cells

To be able to operate a floating gate array, a controller has been implemented by Sebastian Millner [13]. Each chip has 4 instances of this controller so that programming can occur simultaneously. The programming is performed one line at a time and one direction at a time, that means one can either increase values of the cells in the line or decrease them. As a result, one full write cycle consists of writing 129 values into the RAM[14] of the controller and sending a *writeDown* command followed by a *writeUp* command. These commands initiate the programming process, described above, in the specified directions. This way, after both commands are executed properly, every cell is programmed correctly regardless of its former value.

In detail, the programming process can be described as shown in fig. 2.5. Each line of cells shares a high-voltage driver, connected to their CGLs, each column shares one as well, but it is connected to the CGS gates. Now, to increase the values of cells in line 1, its CGLs are briefly set to 0 V and CGSs of all the columns are set to 11 V. After this pulse each cell is read out and the values are compared to the nominal values in RAM. If a cell has reached its desired value, it is omitted during the next programming pulse (cell FG11 in the schematic) and CGSs of this column are set to 5 V. This way a maximum difference of only 6 V can be achieved between CGS and CGL of the target cell, which is not sufficient to induce a considerable electron tunneling[15]. This simple programming scheme is looped until every cell reaches its nominal value or the pulse count reaches pre-defined stop value.

[14] Random Access Memory.

[15] However, the measurements have shown that tunneling still occurs and becomes considerable after several write pulses. One of the goals of this thesis is to quantify and limit the potential drift, caused by this effect.

For the readout of a cell all control gates are set to ground and the address transistor of the desired cell is activated. The cell is then connected to the analog output.

The floating gate controller itself also has several parameters to optimize the programming process in terms of writing speed and accuracy. Here is a short description of these parameters and their functions:

Maxcycle: Maximum number of write pulses, applied to a line in case there is a cell that could not reach its desired value. *Maxcycle* has an 8-bit range (possible values from 0 to 255).

Pulselength: This parameter determines the time scale on which the basic operations (e.g. write pulses and read cycles) are performed by changing the controller clock. It is the most crucial parameter when it comes to optimizing the programming time. It has 4-bit range.

Readtime: Time for the voltage at the comparator to settle (in clock cycles). Has a 6-bit range. The time scale is determined by *pulselength*. In between the write pulses the cell values must be compared to the nominal values. To do that, the wire from cell output to the comparator has to be pulled up or down to the momentary voltage. Depending on this voltage and the value of *biasn* (see below), this can take a considerable amount of time. To be sure that the read out value is an actual floating gate cell value, *readtime* must be set as high as possible to allow the output voltage to settle. On the other hand, at 129 comparisons per writing pulse this would strongly affect the total programming time. Hence, the balanced value of this parameter is very important for optimizing the writing process.

Voltagewritetime and ***currentwritetime:*** Basic length of the writing pulses on the time scale, determined by *pulselength*. The range is 6 bits. One parameter only affects the voltage cells, the other one is responsible for the current cells.

Acceleratorstep: The tunneling current gets weaker as the value of the cell increases. This makes it difficult to reach higher values. For this purpose, an accelerator principle has been implemented. If a cell has not finished programming after an adjustable number of pulses, pulse lengths (*voltagewritetime, currentwritetime*) are doubled. However, if the length of the pulse becomes too high, the accuracy of the programming process is impaired. The range of this parameter is 6 bits.

Biasn: Basically, this parameter controls the voltage on the VB transistor gate (see fig. 2.4). The range is 4 bits. The higher this voltage, the higher the voltage drop on VB transistor. This affects the readout time, because the bias point of VOUT shifts and hence the duration of line charging changes too. If the bias point is too low, the readout time for higher values has to be longer. If it is too high, the same applies to lower cell values, so this parameter is also very critical.

Due to mismatches in the course of chip production, the values of different cells will systematically vary under the same conditions. This is not an issue though, because this effect will be compensated by calibrating the neurons. The greater issue is the accuracy of programming, which strongly depends on the parameters, described above. One of the main goals of the present thesis is to find optimum parameters that ensure fast writing speed and high accuracy.

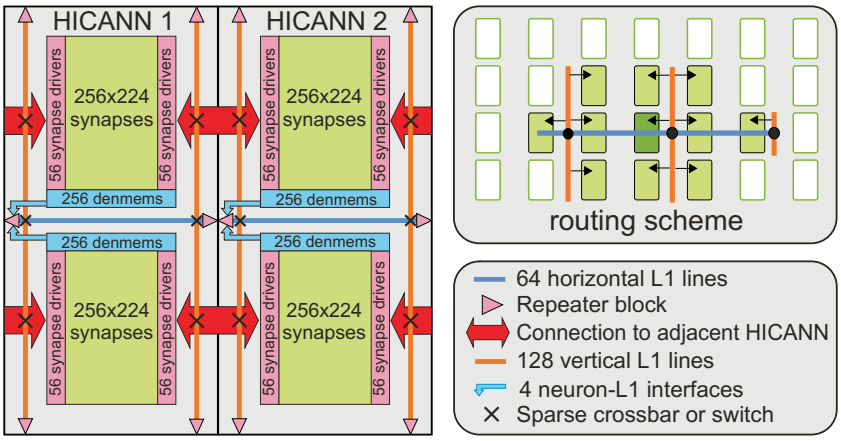


Figure 2.6: Diagram of Layer 1 network. On the right: the routing scheme between the HICANN chips. On the left: major components of the L1 bus system. The signal is injected through the center left repeater block and can traverse the network by using the vertical and the horizontal bus lines, the crossbars, and the switches, until it reaches a synapse driver. Figure partially re-drawn from [11] and [10].

2.2.3 Layer 1 Communication Protocol

To communicate with each other, hardware neurons require an infrastructure that has sufficient resources to implement complicated interconnection patterns, found in biology. The communication structure of the final system is therefore layered: *Layer 2* (L2) protocol (see 2.3) is deployed preferably for inter-wafer communication, while *Layer 1* (L1) protocol covers the intra-wafer routes.

Neuron-to-neuron communication occurs in a manner of source neuron sending its predefined number to a synapse of a target neuron, which, on recognition of the number, emits a current pulse onto latter neuron's capacitance. This 6-bit *neuron number* is transmitted serially over a differential wire pair (referred to as "line" in the following). Therefore a network of such lines with highly adjustable topology is required.

The principle of Layer 1 network is shown in fig. 2.6. It consists of 64 horizontal and 256 vertical differential lines per chip. They are interconnected between the chips and hence traverse the whole wafer (fig. 2.6 on the right), allowing arbitrary connection patterns. The topology is defined by 2 *crossbar switches* and 4 *synapse driver* (2.2.4) *switch matrices* per chip (see below). Due to power consumption limitation, the potential difference in a pair is planned to be quite low, approximately 150 mV, which brings up new issues: high capacities of driven wires (charging the wire takes time) and signal decay (amplitude decreases fast with traversed wire length). The driven capacity issue is solved by reducing the switch connectivity (fewer interconnected transistor gates), which will be discussed later. To overcome the signal decay issue, series of *repeaters* (see below) are embedded into the network structure in such a way that the signal is recovered and retransmitted on each transition form one HICANN chip to another.

The frequency of the clock, used for L1 signal generation, can be adjusted from 50

to 250 MHz. The final system operation is expected to use the clock frequency between 100 and 200 MHz. Higher frequency allows for a higher data rate, but also means higher power consumption and more crosstalk between neighbor lines, leading to higher error rates.

L1 Repeaters

One HICANN chip contains 6 repeater blocks as seen in fig. 2.6. In an alternating nature, every line is repeated either at the incoming or at the outgoing edge of the chip. A repeater block is a complex structure, consisting of 32 or 64 repeaters and a controller that ensures the configuration and operation of the block.

Every repeater is bi-directional and consists of a receiver (with deserializer), a timing restoration unit and a driver (with serializer). It also has a parallel I/O connection to a controller, allowing it to transmit test data, written into controller's RAM or to write the received serial data into this RAM to be able to verify correct functionality. This feature was often used to perform tests in course of present diploma project.

One of the main features of a repeater is that it does not need an external clock to function. It is able to restore the time frame of a packet from the incoming signal alone by using a Delay Locked Loop (DLL) [10]. However, it needs to receive several *zero-packets* first, to accommodate its timing (so-called *DLL lock*). Also, if certain amount of time passes since the last packet, the DLL has to be locked again.

Eight of the repeaters in the center left block are called *sending repeaters*. Their inputs are connected not to the test data output of the controller, but to the 8 outputs of a *neuron control* instance (see 2.3.5), allowing them to inject neuronal events from local neurons, L2 bus (see 2.3), or *background generators* (2.3.5) into L1 network.

L1 Crossbars and Synapse Driver Switches

The packets, transmitted by the sending repeaters, are injected into predefined horizontal lines. To reach its destination, the packet has to be transferred onto a vertical line. With the help of crossbar switches, located on intersections of horizontal and vertical lines, it is possible to connect a horizontal line to a vertical line. Due to a capacitive load of the switching transistors, the switching matrix had to be populated sparsely and hence it is not possible to connect every horizontal to every vertical line. That introduces constraints to network connectivity. For more information on this issue see [13, 11].

Synapse driver switches are located on the left and right sides of the synapse arrays and also have sparse connectivity. They each interconnect 128 vertical L1 lines of the corresponding chip side with 112 horizontal lines, alternately serving local synapse drivers and synapse drivers of adjacent HICANN chip (see fig. 2.6).

2.2.4 Hardware Synapses

As shown in fig. 2.2, there are two blocks of synaptic circuits on the HICANN chip, each containing 256x224 synapses. Left and right from these blocks, there are columns of 56

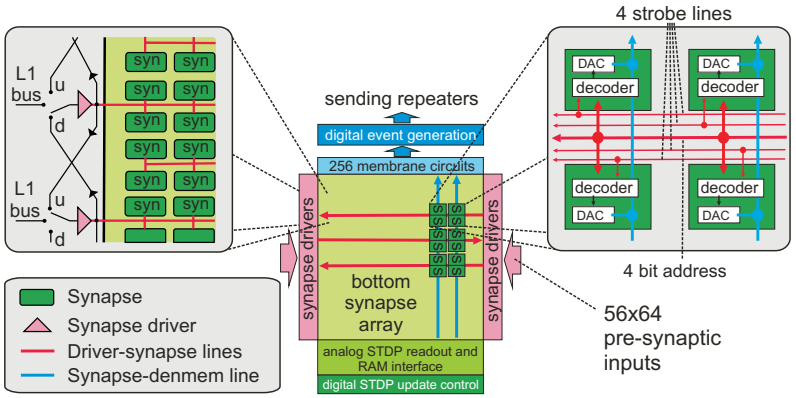


Figure 2.7: Block diagram of a synapse array. On the left: synapse drivers are located alternately on both sides of the synapse array and are controlling 2 lines of synapses each. The drivers can receive their input either from the adjacent L1 bus or from the neighbor driver. In the center: The overall schematic view of a bottom synapse array. Every column of synapses is connected to one denmem. On the right: Driver connection to the synapses. The strobe lines initiate current pulses that are transmitted to the denmems upon neuron address match. Figure partially re-drawn from [10] and [11].

synapse drivers, each controlling 2 lines of 256 synapses. Each column of 224 synapses is connected to one denmem. This constellation is seen in fig. 2.7.

Synapse Drivers

Each synapse driver is connected to an L1 line and thus has an L1 deserializer (identical to those of the L1 repeaters) to receive input from pre-synaptic neurons (*neuron numbers*). The 6 bits of a received number are split into two parts: the upper two are used to select one (or more) of the four strobe lines, and the lower 4 bits are transmitted into the synapse array to be decoded in the synapses (fig. 2.7). One in every four synapses is connected to the same strobe line in a certain pattern. An activated line sends a pulse of an adjustable length τ to all the synapses that are connected to it. But only the ones with correct address decoder values, (matching the 4 lower bits), are activated and transmitting the pulse along to the target denmems. Using the analog output multiplexer of the HICANN chip, one can observe these pulses of the first line of the first driver of each synapse block with an oscilloscope for triggering and debugging purposes.

One more feature of synapse drivers is the ability to interconnect them (fig. 2.7 on the left). This way only one driver has to have access to an L1 line. Other drivers in the chain receive their input from said primary driver. This is an important feature, serving to lower the demand for L1 lines. It was tested in course of this diploma project.

Synapse Circuits

Along with the aforementioned programmable 4-bit address decoder, synapse circuits also contain a programmable 4-bit DAC, responsible for storing of the synaptic weight

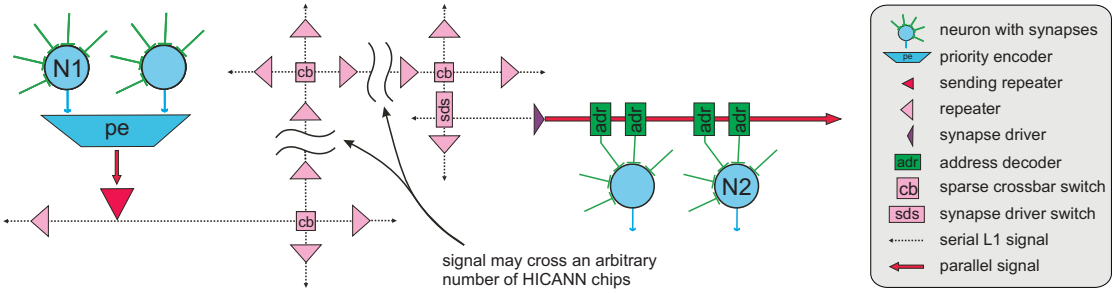


Figure 2.8: Neuron-to-neuron connection over Layer 1 serial bus: Neuron transmits a parallel signal that is serialized and injected into L1 network, then parallelized again before it reaches another neuron in form of a current pulse from a synapse. Figure re-drawn from [10].

and translating it into an output current. So when the address decoder receives the correct 4 bits and the strobe line is also activated, synapse DAC emits a square current pulse of the length τ and amplitude $g_{max} \cdot weight$, where g_{max} is also a value, determined by the synapse driver. This pulse is then used as an input of the synaptic ion-channel circuit of the corresponding denmem [17].

Of course, synaptic circuits have more complicated structure, than depicted here. Description of components, not used in the course of this thesis, were therefore omitted. More detailed information on hardware synapses is available in [20, 10].

2.3 Wafer-Scale Communication Protocol

2.3.1 Layer 1 Communication

A typical path of a neuronal event on the L1 bus is depicted in fig. 2.8. A pre-synaptic neuron emits a digital action potential[16]. If several neurons of the same neighborhood spike in the same clock cycle, a priority encoder (every 64 adjacent denmems share one) determines which neuron transmits its number first, others are put in a queue. The neuron number is forwarded to the sending repeater, which serializes it and transmits it on its L1 line. After traversing an arbitrary number of crossbars and repeaters, the signal reaches a synapse driver switch where it is sent to synapse driver's receiver and then deserialized. If the driver's strobe lines and address decoder of a synapse are programmed to match this particular neuron number, a synapse emits a current pulse onto post-synaptic neuron's ion-channel emulation circuit. That results in membrane potential shift.

[16] Analog of a biological spike, represented by the 6-bit neuron number being sent out to a synapse of the receiving neuron.

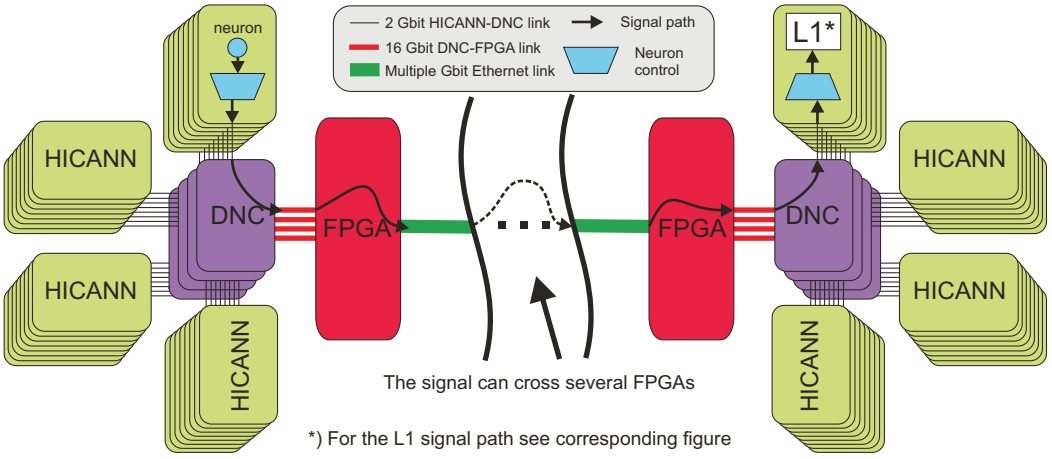


Figure 2.9: Typical signal path over Layer 2 bus. The source signal traverses the neuron control waypoint (fig. 2.10), the DNC, several FPGAs, another DNC and is afterwards injected into the L1 network of another HICANN chip. From here it can be re-routed all over the wafer as shown in fig. 2.8.

2.3.2 Layer 2 Communication

Although L1 protocol is sufficient for neuron-to-neuron communication on the whole wafer, another protocol (Layer 2) has been developed to complement it and make the system scalable. Basically, an outgoing signal can be routed as described above, or it can be redirected to an outgoing DNC link (see 2.3.5) to be routed outside of the wafer (see fig. 2.9).

The L2 protocol specifies one DNC (2.3.3), managing the packet-based communication between 8 HICANN chips and an FPGA [15]. Each FPGA maintains 4 DNC links and a link to other FPGAs. This way the system can contain several wafers, interconnected through L2 bus.

2.3.3 DNC Functionality

As pointed out before, one DNC communicates with 8 HICANN chips using 8 serial LVDS[17] [26] links, representing 8 logical channels for each chip, so when it releases a signal (neuron number) over one of 8 possible channels (see 2.3.5), a packet is built by the DNC, encoding the source address of the spiking neuron, using the internal chip number, channel number and the internal release time of the event. This data packet is sent upwards to the FPGA.

On the other hand, when receiving data packet from an FPGA, the DNC makes sure that, based on the packet content, the correct neuron number is released to the correct channel of the target HICANN chip at the right time.

[17]Low Voltage Differential Signaling.

2.3.4 FPGA Functionality

General-purpose FPGAs are used to control the data flow from/to 4 DNCs. Two general types of packets are designed for transmitting the neuronal event data and the configuration data for the whole system. Also, many vital test functions are integrated into the FPGAs (see below).

The FPGAs play a major role in Layer 2 communication. From each received packet an information on the source is extracted and compared to a routing lookup-table for the target to be determined. In case of multiple receivers the event is duplicated and sent to multiple locations. Also, the release time of the event can be modified to emulate biologically-relevant spike delays in hardware. Depending on the outcome of this process, the event is either immediately transmitted along to its destination, or is buffered to be transmitted later.

As FPGA directly communicates with the software, controlling the system, and is easily augmentable, it provides the best opportunity to implement adjustable test circuits to examine the system on the early development stages, such as:

Pulse-FIFO[18]: A FIFO-buffer for up to 10,000 outgoing neuronal events, to be released at desired times.

Trace-FIFO: A FIFO-buffer for up to 10,000 neuronal events, coming from the connected DNCs and HICANN chips.

Background generator: A circuit, generating random neuronal events, Poisson-distributed[19] on the time scale, and transmitting them to the desired channels of the target HICANN chip (using L2 protocol and DNC). It can also operate in a periodic mode, whereby it generates certain neuron numbers with certain periodicity.[20]

2.3.5 HICANN-DNC Interface

On the logical crossroads between Layer 1 bus, Layer 2 bus, the priority encoders of the neuron block, and the background generators (see below), lies an instance called *neuron control*. It is located on the HICANN chip and is controlling the neuronal event distribution in all possible directions (see fig. 2.10).

A *background generator* (BG) on the HICANN chip has a similar purpose as FPGA BG, but slightly different functionality. There are 8 BGs on every HICANN chip. They can be operated both in Poisson and in periodic mode, but each of them can only transmit one neuron number. BGs of HICANN v1 can only transmit zero-packets (as it is necessary to lock the receiver DLL). In version 2 of the chip, the generated number is adjustable, but it requires a write cycle to change it. Depending on the link speed, it can take up to 100 ms to do that (using JTAG[21], see 2.4). Thus the BG cannot be used to generate arbitrary event patterns because the time scale of the L1 communication lies in nanosecond range.

[18] First-In-First-Out.

[19] Being a digital circuit, BG is completely deterministic, so randomizing is done, using a linear feedback shift register to generate pseudo-random numbers.

[20] The FPGA-code is developed by Stefan Scholze at TU Dresden.

[21] Joint Test Action Group.

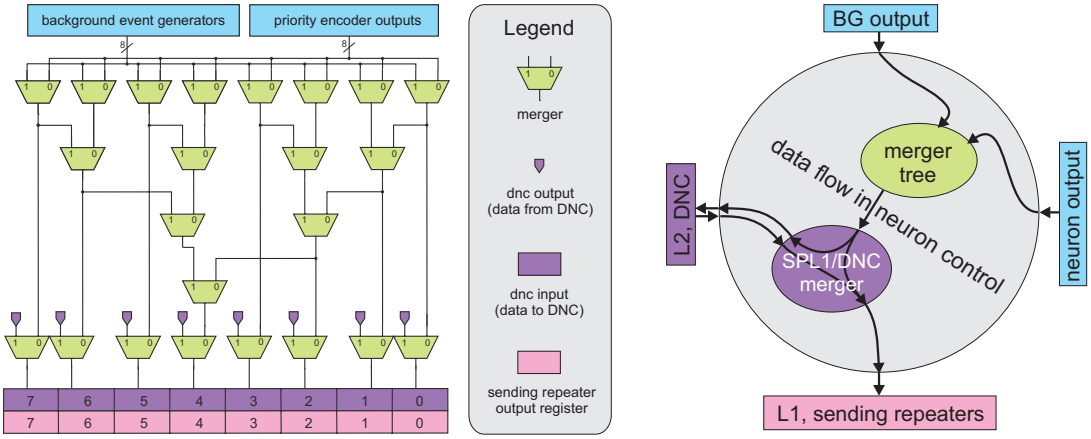


Figure 2.10: Block diagram of neuron control. On the left: schematic of a merger tree with available inputs and outputs. Note, that some input channels can only merge into certain output channels. Also, some input channels can be mirrored onto several outputs. On the right: data flow diagram, depicting neuron control instance as a crossroads between 4 I/O instances. Note, that L2 connections are only potentially bi-directional. Factually, they can only transmit in one direction without being reprogrammed. Figure partially re-drawn from [13].

The outputs of BGs and priority encoders lie side-by side and are connected to a *merger tree*. This tree is shown in fig. 2.10. Each merger in the tree has 2 modes. One is a simple multiplexer mode whereby single input channel is forwarded to the output channel. The other mode "merges" both inputs together, so that all incoming events are passed on to the single output[22]. The 8 outputs of the merger tree are themselves merged with the 8 DNC outputs and forwarded to the 8 L1 channels (i.e. sending repeaters), and in parallel to the 8 DNC inputs (see [13]). Here it has to be said that although all DNC links are per se bi-directional, only one direction can be used momentarily without reprogramming the corresponding entities.

Above mentioned features of neuron control make L1/L2 transition possible and grant many possibilities for testing of the prototype system (see 2.4). For example one can use BG to stimulate neuron(s) through one channel, and use another channel to forward the resulting neuron output to the DNC and FPGA where it can be captured by the trace-FIFO and afterwards read out and analyzed. This and similar methods are used in the course of the following tests.

2.4 Prototype Test Setup

For initial testing of single system components and their combinations (1 FPGA + 1 DNC + up to 8 HICANN chips), a test setup was designed at the University of Heidelberg

[22] This allows, for example, event injection via L2 bus while using the background generators to constantly produce zero-events, thus keeping the DLLs locked.

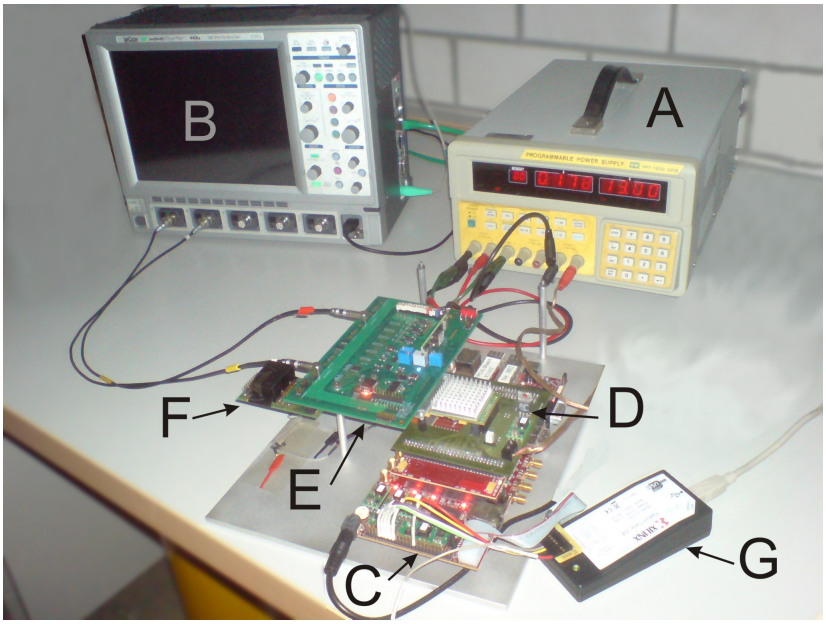


Figure 2.11: Test setup: (A) Power supply, (B) Oscilloscope, (C) FPGA-Board, (D) DNC-Board, (E) System Emulator Board, (F) HICANNmodule with mounted HICANN chip, (G) JTAG-Controller.

and TU Dresden. A photograph of this setup can be seen in fig. 2.11. Following is the detailed description of its components and other instruments used in this thesis.

2.4.1 Test Hardware

System Emulator Board

The **System Emulator Board** (SEB), formerly called iBoard, was developed by Sebastian Millner and Andreas Grübl at the University of Heidelberg for implementing the prototype system with up to 8 HICANN chips. SEB serves as a waypoint between the DNC and the HICANN chips. It also allows to exclude the DNC from the chain in cases, where DNC functionality is not explicitly needed for the tests. This simplifies the data flow and allows for better debugging.

SEB is powered by 13 V supply voltage and uses 3 DC-DC converters and 2 programmable DACs to provide the chips with necessary voltages and currents. It also provides 2 ADCs[23] and 2 Multiplexers to be able to automatically read out (static) analog outputs of the HICANN chip, e.g. values of the floating gate cells. For the read out of non-static analog signals, two LEMO connectors for the oscilloscope are integrated.

[23] Analog-to-Digital Converter.

Power Supply

A **Tektronix PS2520G Programmable Power Supply** has been used for powering the System Emulator Board and the DNC-Board. Also, in some cases certain HICANN chip supply voltages were provided by it instead of using the SEB. This has the advantage of being able to adjust and monitor the current, consumed by the chip, and also eliminate the high-frequency noise, produced by the DC-DC converters, during some sensitive measurements.

Oscilloscope

For the analog signal readout, a **LeCroy SDA6000A Serial Data Analyzer** has been used. This powerful oscilloscope is providing up to 4 input channels, and sampling rates of up to 20 GS per second. SDA6000A also has an Ethernet controller and thus can be accessed and controlled through standard Ethernet network [27], which is very important for long-lasting automated tests.

Even when no data from the oscilloscope is explicitly needed for the test, usually, the oscilloscope is still very useful for debugging, as it can provide information about different HICANN chip components, such as the state of neuron membranes or the repeaters. One can also see if the signal correctly reaches the synaptic drivers. There are many other uses for the oscilloscope readout. They will be described in respective sections of chapter 3.

JTAG Controller

The final wafer-scale system will be accessed and controlled using multi-gigabit Ethernet links. At the time of carrying out of this thesis, the work on Ethernet components was still ongoing. Thus the JTAG connection was used to communicate with the prototype system. HICANN chips and DNCs have JTAG modules integrated in them for the purpose of debugging, the FPGA-Board also has standard JTAG controller. The connection to the host computer was established using **Xilinx DLC10** USB adapter.

In comparison to Ethernet, JTAG is very slow. The main problem for the tests was the long turnaround-time in the JTAG chain. Sometimes up to 20 ms. This severely prolonged some of the tests and in some cases made the gathering of statistical data impossible. Nevertheless, JTAG made these tests possible in the first place.

FPGA Board

General purpose FPGA of the type **Xilinx Virtex-5 LXT** [28] with **LX110T FPGA Board** was used in the prototype system setup. The FPGA code was developed by Stefan Scholze of TU Dresden. Xilinx DLC10 JTAG controller was connected to the FPGA board, making the FPGA first device in JTAG chain.

DNC Board

The DNC board with mounted DNC was developed at TU Dresden and custom-made for the prototype system. It is mounted on top of the FPGA board and connected to the SEB on the opposite side.

HICANNmodule

The HICANNmodule was designed by Sebastian Millner at the University of Heidelberg. A maximum of 2 HICANN chips can be bonded onto one HICANNmodule, which is connected to the System Emulator Board, completing the JTAG chain. Assembled as described, the system is ready for operation.

Additional Used Instruments

- Keithley 2100 - USB-compatible digital multimeter.
- FLUKE 77 series II multimeter.
- LeCroy ZS1000 1 GHz active probe

2.4.2 Test Software

tests2

Tests2 is a software tool, developed at Electronic Vision(s) group for the purpose of testing the prototype system. It is written in C++ [29] and is easily extendable by adding new modules. In the course of this thesis, tests2 was considerably extended (see 3.1). Apart from tests for the present thesis, new modules can be used for developing more complex tests and, potentially, for later use in design of the operation software for the final wafer-scale system.

Tests2 controls the prototype system and is used to gather test data that can then be analyzed with the help of other software tools (see below).

matplotlib

Matplotlib [30] is a powerful Python-based [31] graphic data analysis software. It specializes in 2D-plots and is easily incorporated into python scripts which makes it simple to simultaneously manipulate gathered test data and display it in the form of plots. All 2D plots in this thesis are produced with the help of matplotlib.

gnuplot

Gnuplot is a freeware software tool that can generate plots. Its advantage over matplotlib is that it can produce high quality 3D plots. Its disadvantage is that one cannot easily manipulate data (as with python in the above case) before plotting it, so the data has to be processed beforehand. All 3D plots in this thesis are produced using gnuplot.

3 Tasks and Methods

Following chapter is divided into 3 parts, each of which describes, in chronological order, the proceedings of experimental work, presented in this thesis.

First section is dedicated to preparatory work that had to be carried out before any tests could be performed. It includes setting up parts of the SEB that were not tested before, making sure that they work correctly and writing a software module to control SEB functions. Further, several control modules were written for operating different components of the HICANN chip. Lastly, the new HICANN v2 functionality was included into the existing software.

The second section deals with the extensive testing of the floating gate memory cells. Purposes and methods of testing are explained and expected outcomes are discussed.

Third part of the chapter describes testing of the synaptic circuits and parts of L1 network.

3.1 Extension of the Test Software

The test software is developed and used simultaneously by many people, pursuing two general goals. The first one is simple and efficient testing of the hardware, the second is providing initial reference in further development of software for the HMF.

Basically, there are two types of modules in tests2 software. The *control modules* and the *testmodes*. Both types are derived from a template class and then linked into the main program on compilation. Control modules resemble the hardware circuit structure, i.e. for each component, such as the FPGA or the synapse array, an instance of a control module is created using the memory address range of the desired component. Control modules contain basic control functions and procedures for all hardware constituents that are used. Testmodes are basically test routines, using control modules to communicate with the hardware and get it to perform desired tasks. On program start, tests2 initiates the communication link over JTAG to the FPGA, and runs the specified testmode, which subsequently initializes the necessary control instances and carries out the test routine.

3.1.1 System Emulator Board Control Module

Although not part of the HMF, SEB is a part of the prototype system and has many functions. Therefore a control module for tests2 was needed to access these functions quicker and easier.

This control module was implemented as the first part of diploma project. It is based on example code by Andreas Grübl. The final functionality of the module named **IBoardV2Ctrl** includes:

-Voltage setting: Uses on-board programmable DACs to set 11 voltages on the board to desired values. These voltages are needed for correct functioning of the HICANN chip.

-Voltage readout: This function uses on-board ADCs (see below) to read out momentary values of 11 voltages on the SEB and 2 voltages coming from analog outputs of the HICANN chip.

-Switching multiplexers: The function uses two 8-channel multiplexers for switching analog outputs of up to 8 HICANN chips onto 2 analog ADC inputs and SEB LEMO-connectors.

-Self-configuration: An external xml-file is used to configure the instance of IBoardV2Ctrl to deliver correct output. The configuration data includes: exact resistor values of the particular physical board for proper voltage calculation, correct addresses of DACs, MUXes and ADCs, and the default voltage settings.

Testing of ADC Functionality

As mentioned before, SEB possesses 2 8-channel 10-bit ADCs of the type AD7997 [32] to read out the analog outputs of the chip. Prior to begin of this work, the function of these ADCs has not been tested, so before proceeding with tests, involving them, ADC's functionality was secured and the error was estimated.

The reference voltage of 3.3 V is used for 10-bit ADC, which yields a resolution of ± 1.6 mV for a single measurement [32]. To further reduce this error, an averaging over multiple readouts is applied. For the purpose of measuring ADC readout error, the output OP in the HICANN chip is set to deliver highest possible voltage (approx. 1650 mV on HICANN v1). This is done to obtain the most stable input on the ADC[1].

It has been noticed that in such series of readouts, the first returned value was often false. The cause of this is that the acquisition phase of the ADC lasts exactly 1 μs. Measurements with an oscilloscope have shown, however, that the stabilizing period of an analog output of the HICANN chip can last up to 100 μs and more, depending on the number of the read out cell and other parameters, because the controller is designed in such a way that it has to sequently skip over all preceding cells in order to read out the desired one[2]. Fig. 3.1 shows a voltage trace of the floating gate readout. On the left, the 1st cell of the line is being read out and the output is stable already after 50 ns. On the right, the last cell of the line is tapped. The output is stabilized only after approx. 43 μs. Considering this controller workflow, the first returned ADC value in the series of readouts is always dropped.

Through variation of number of readouts for the average calculation, it was experimentally determined that the error can be reduced to ± 0.5 mV by using an average of 8 measurements. Further increasing of readout count had no effect on the measurement accuracy.

[1] The residual instability originates only from supply voltage oscillations, because the OP is already at its rail voltage and thus the small input distortions do not propagate to the output

[2] This issue has been resolved and the updated readout routine will be implemented in the next HICANN chip version.

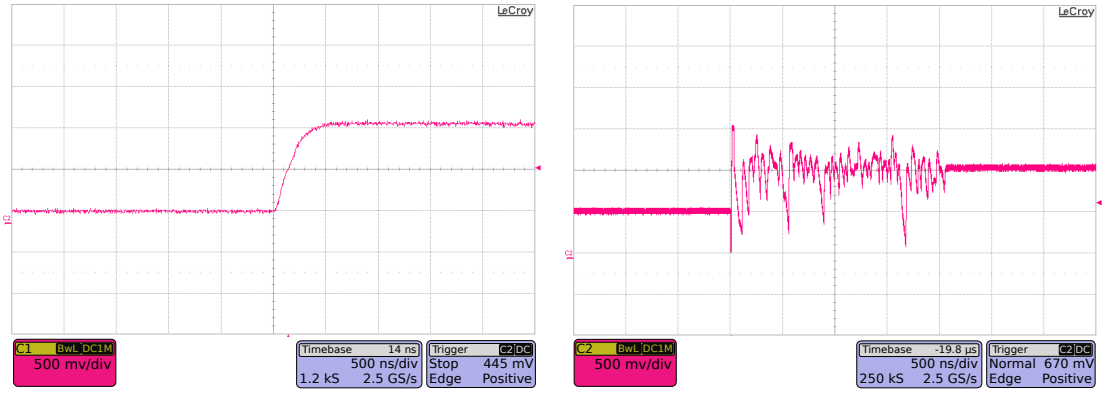


Figure 3.1: Timing of the floating gate cells readout. Left is the readout trace of the 1st cell of a line, right is that of the 129th. Depending on the number of read out cell, the digital operation frequency and the value of *pulselength* parameter, the duration of readout voltage stabilization can take over 100 μs.

The resulting routine using 9 readouts (with the first one being dropped) and returning an average is implemented in IBoardV2Ctrl.

Previously described measurements took into account the statistical error, but did not cover the systematic error. The ADC is calibrated using a potentiometer, regulating the 3.3 V supply on the SEB, and an external calibrated multimeter. Input voltage is set to an arbitrary value (*calibrating point*), which is also observed by the multimeter. Then the potentiometer (*reference voltage*) is adjusted until the read out value of the ADC matches the value on the multimeter. In theory, the quality of calibration should not be affected by choice of calibrating point and $V_{ADC} - V_{multimeter}$ should be 0 for all voltages.

In reality, it has been noticed that $V_{ADC} - V_{multimeter}$, in the first approximation, is a linear function with small but noticeable declination. Because of this, output values of the ADC differ from the exact ones, depending on the calibration point of the ADC. Figure 3.2 shows a measurement where the point of calibration lies at approx. 1.2 V. It depicts the systematic error of the ADC in the range of 0 V to 1.8 V used in following measurements. For the extreme values, it amounts to approx. 1 mV, corresponding to a relative error of 0.06%.

Note, that following tests contain no measurements, involving an ADC, where systematic error of 1 mV is significant for the outcome. More important is the random error of the measured value, which is estimated at $\pm$**0.5 mV**. This is the value of an ADC error, used in the following tests. The systematic error of 0.06% is neglected.

3.1.2 Extension of Further Control Modules

Before the start of this thesis, only two control modules for the HICANN chip components were completed: the *NeuronBuilderControl* that handles the hardware neuron configuration and analog output multiplexers, and *FGControl* that is responsible for the floating gate memory arrays and external neuron stimulation. The following modules'

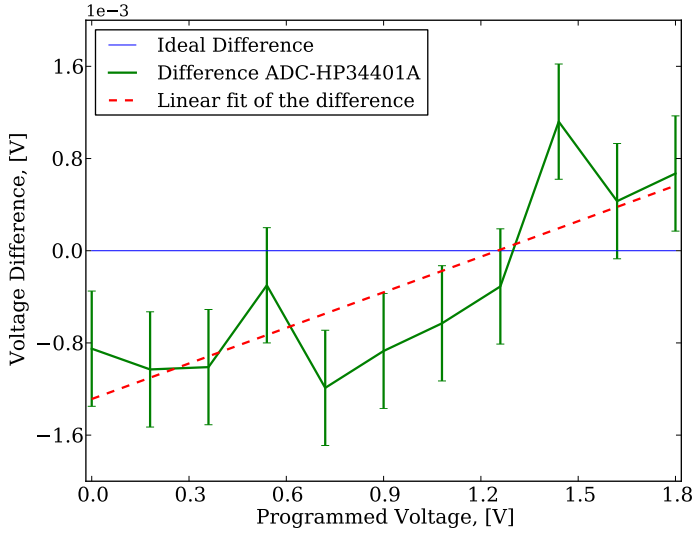


Figure 3.2: Depicts the absolute error of the ADC calibration in the used voltage range. Ideally, $V_{multimeter} - V_{ADC}$ must be zero (thin blue). Using the measurement points, linear regression is performed (dashed red). The fit function approximates the measured curve (thick green) well inside of the error range. Error bars represent the estimated random error of the ADC. The error of calibrated multimeter is unspecified.

functionality was limited to reading and writing of the corresponding instance's memory registers. Listed below are the new functions, implemented in course of this diploma project[3].

RepeaterControl

Each instance controls one block of 32 (in the center of a HICANN chip) or 64 (top and bottom blocks) repeaters.

-Direction setting: The function is used to configure a repeater to drive the connected L1 line directed into the chip, or the one directed outside of the chip. The resp. other line is used to receive input.

-Receiver enable: Switches L1 receiver of a repeater on and off.

-Switch test input: Connects repeater's parallel input to the test data output (TDO) of the repeater block. The function is used to transmit test data packets on an L1 line. All repeaters can receive their input from the controller simultaneously.

-Test data write: Writes the neuron numbers to be transmitted and their corresponding timings to the 3 TDO registers.

-Switch test output: Connects repeater's parallel output to test data input (TDI) of the repeater block. The function is used to receive data packets on an L1 line and read them out digitally. Only one repeater at a time can be used to receive data.

[3]HICANN-Specification [13] and the to-date existing testmodes served as reference for this work.

-Test data read: Reads out the TDI registers of the controller after a repeater filled them with received packets.

-Start/stop TDI/TDO: The functions trigger the controller to start or stop the sending/receiving processes.

-Configure sending repeater: This function uses the above mentioned functions to properly configure a sending repeater for transmitting events on its L1 line. Sending repeaters' parallel input is connected not to the TDO of the repeater block, but to the output registers of neuron control (see 2.3.5).

L1SwitchControl

Each instance controls one crossbar or synapse driver switch.

-Connectivity check: This function verifies if two subjected lines can be interconnected according to sparseness of the switch matrices specified in [13].

-Connect lines: Connects or disconnects a vertical and a horizontal L1 line if connectivity check result is positive.

SynapseControl

Each instance controls a block of 224x256 synapses and 112 synapse drivers.

-Write weights: The programming of the synaptic weights is performed row-wise. The 256 pre-defined values are written into specific registers and the write command is issued for the target synapse row.

-Read weights: Reads out a row of synaptic weights.

-Write decoders: Due to space-saving layout of the synapse circuits, each two rows of synapses share their two rows of address decoder RAMs with each other. This means that decoder data for one row of synapses is partially stored in its neighbor's memory. Thus two rows of address decoders, sharing the same driver, have to be programmed simultaneously. Decoder data of both rows must also be properly altered first to match the data mapping, resulting from the chip layout.

-Read decoders: Reads out two rows of address decoder values.

-Set driver preouts: Configure driver's 4 strobe lines to respond to certain combinations of the upper two neuron number bits, so that when a neuron number is received, the corresponding strobe lines are activated and transmit a pulse to the connected synapses.

-Set driver gmax: Assigns g_{max} values (see 2.2.4) to corresponding strobe lines. This (along with synaptic weights) co-defines the amplitude of the current pulses, induced later in the synapses.

-Configure synapse driver: This function enables the driver and configures its input source (local L1 line or neighbor driver). Also the type of connected synapses is set row-wise to either excitatory or inhibitory.

NeuronControl

This module controls the merger tree and the background generators.

-Configure BGs: This function switches BG mode between Poisson and periodic, and also sets the delay between single events (mean delay in Poisson mode).

-Enable BGs: Switches background generation on and off.

-Configure mergers: By configuring the operation mode (simple multiplexer or merger), sources and timings of single mergers, it is possible to adjust the data flow through the merger tree (see fig. 2.10).

-Enable DNC loopback: For debugging purposes, it is possible to re-route the output of one DNC channel to the input of another one. This function handles 4 possible loopback waypoints.

Accommodating Changes of HICANN V2

The new version of the HICANN chip brought various changes. Some of them (e.g. address space alterations) had to be incorporated into existing software. Most of the changes, such as bit-order reversions and other minor fixes are omitted here. Following are the descriptions of major differences:

-In RepeaterControl, the addressing of some repeater blocks has been changed to satisfy the global L1 addressing convention[4].

-The allocation of switch transistors in both crossbars has changed. This has been incorporated into L1SwitchControl.

-In SynapseControl, much of the code had to be altered due to vast changes: the addressing of drivers and synapses has changed, the hardware controller was completely re-designed, and the read- and write routines have been modified.

-NeuronControl introduced new feature for the BGs: an adjustable neuron number for each BG, which can be changed on-the-fly by utilizing one RAM write cycle (previously only number 0 could be used). Also, the merger tree has been slightly altered.

To work with either version of the chip, the software has to be re-compiled for each of them.

3.1.3 Creating New Testmodes

To carry out the tests, described below, several new testmodes have been created. They contain the procedures and routines that were used.

-tmak_iboardv2: This testmode is based on *tm_iboardv2* by Andreas Grübl. It allows an easy startup of the prototype system after switching the main power supply on. The features include convenient programming of the SEB's DACs, switching the output multiplexers, and resetting of the test logic. Also, it is possible to monitor the voltages on the SEB.

-tmak_fg: This testmode was started as a clone of *tm_fg* by Sebastian Millner, but eventually developed into a separate unit with many unique testing features. All the routines for testing the floating gate memory of the HICANN chip, presented in (3.2), are accumulated in this testmode along with simpler functions, such as programming or reading out of the floating gate cells.

[4]Order of repeater addressing corresponds to the order of L1 line numbering.

-tm_jitter: With the help of this testmode, the tests, described in (3.3), are carried out. The functionality of this testmode incorporates almost all features of the prototype system to the date of writing this thesis.

Short list of tm_jitter's functions includes resetting of each component of the system, configuring the L1 network for event injection to specific synapse drivers, configuring the hardware synapses and neurons, setting up the data flow inside of neuron control instance, controlling the background generators on the HICANN chip and the FPGA, controlling the event FIFOs on the FPGA, stimulating a neuron by an external current and many more. The functionality of this testmode is also used by other members of the work group to carry out their tests of the prototype system.

-tm_quicktests: This testmode was created during the initial test phase of HICANN v2 and is designed to perform quick tests whenever a new chip is commissioned. It helps to determine whether all components are functioning as expected, by issuing a simple command. The functionality includes quick testing of the floating gate memory, repeaters, other parts of the L1 network, background generators, synapse drivers and synapses, HICANN-DNC connection, priority encoders and other parts of the system. The testmode can also serve as a starting point in developing more comprehensive test routines.

3.2 Testing of the Floating Gate Memory

The following section of the thesis describes experiments that were carried out to ensure proper functioning of the floating gate memory cells and evaluate their random and systematic errors, which should be considered while designing the experiments for the final neuromorphic system. The tests are presented in such order that the positive outcome of the former test conditions the viability of the latter one. For example one has to be sure that the cells can hold their charge for a certain period of time, before conducting tests, lasting that long.

3.2.1 Voltage Drift Over Time

Due to the production process, the floating gate of a cell in HICANN chip cannot be completely insulated. Because of that, the charge on the gate is more or less volatile, and, depending on the temperature and insolation, it can slower or faster drift away from the floating gate because of diffusion and leak currents. This process, of course, affects the output of the floating gate cell.

The goal of this test is to quantify the cell output change over time. The motivation for it is as follows: the length of a single experiment, planned to be carried out on the HMF, is expected to be well under one minute[5]. However, one can conduct many experiments in a row with the identical setup, e.g. to gather statistical data. This means that one would program the floating gates once and then run several experiments with the same memory values. The question is: for how long can such series of experiments be run

[5]Which, considering the speedup factor of 10^4, corresponds to several days of real time.

without compromising their accuracy? To answer this question, a measurement on the prototype system has been performed: Ten lines of one chip (5 with voltage cells and 5 with current cells) were programmed to different initial values in the range from 0 V to 1.8 V and left in this condition for 200 hours. During this time, every 20 seconds, every cell was read out once. After that, the mean value and standard deviation for every line was calculated and saved to a file. This allows for reconstructing the drift of each line over time, in order to determine whether all cells in a line drift in the same direction with the same rate. The mean of a line reflects an average cell value, while standard deviation serves as an indicator for the drift speed spread between different cells. The results of this test are presented in (4.1.1).

3.2.2 Stress Test

In a complex system it is crucial that all components are robust and have a long lifetime and low error rate. If a floating gate cell becomes damaged or inaccurate, the neuron, driven by this cell, cannot be used any more. During initial tests, conducted by Sebastian Millner in 2010, several lines of floating gate cells have been corrupted. An assumption arose that the reason for this corruption was the 11 V supply voltage being too high for proper operation of the circuits[6].

This test is designed to evaluate the changes in floating gate memory functionality as the operation time progresses. It simulates operation of the floating gates under stressful conditions, including VDD11 supply being set to 11 V. The measurement method is very simple. Two lines of floating gates (one with voltage cells and one with current cells) are used. On each line, a series of 50 write cycles is performed. One cycle consists of programming the line to 0 V, followed by programming it to approx. 1 V. The value of 1 V was chosen because it is expected to be an average use case for the final system, thus producing a realistic wear-out effect on the cells. After 50 write cycles, the lines are programmed to 0 V and the cells are read out. The mean value and standard deviation for the lines are calculated. Then the lines are programmed to approx. 1 V and read out again. Again, the mean and the STD are calculated and saved to a file. Of course, the single cell values are also saved to be able to find errors (e.g. faulty cells) in case they occur. This procedure is looped until the desired write cycle count is achieved.

Thusly generated data allows us to observe many possible effects that can take place. For example, if a cell would become unstable or damaged, this would lead to an STD increase. If a cell would get "stuck" at one value, the STD would increase and the mean would change. Also, one can observe the mean and standard deviation trends and quantify their change over operation time.

Due to the slow communication speed while using JTAG, this stress test was is limited to 200,000 cycles, which took approx. 7 days to measure. The results can be found in section (4.1.2).

[6] Following this, the 11 V operation voltage, called VDD11, was durably reduced to 10.5 V until further tests, such as this one, could provide more information. For these reasons, all tests, described in this thesis, are carried out with VDD11 set to 10.5 V unless specifically denoted otherwise.

3.2.3 Differential Programming

During the initial tests in 2010 a so-called *sequential* programming scheme was established, which assumed block-wise writing with preceding reset, i.e. all 24 lines are first programmed down to 0 V and then up to their nominal values. This programming scheme has been chosen as being a fail-safe solution, but it is not an optimum one. Often, especially in the test phase, it is sufficient to program only one line, which takes $\frac{1}{24}$th of the time. Also, it is unnecessary to program all the cells to 0 V before writing them up.

A better programming scheme would be programming a line of cells first down to the nominal values (the cells with values lower than desired are not changed) and then up, using the same values (hereby the cells that are already programmed correctly in the downward cycle are not changed). This way all the cells also reach their desired values, but it takes less time, as their floating gates are (in an average case) already pre-charged. This scheme is called *differential*.

Present test compares the results of both programming schemes in case of writing random values into the cells. The method involves generating random numbers (in sets of 129) and programming them to the floating gate memory. After that, each cell is read out and thusly obtained values are compared to the values that were written. This is done to reconstruct the *response function*[7] of the floating gate memory line. A mean of 20 measured *response functions* is calculated to improve the statistics. These *response functions* for both programming schemes were recorded and compared in the course of this test to ensure that the new differential method is applicable. The results are presented in (4.1.3).

Here it needs to be said that the new programming scheme had passed the test and thus was used in all following measurements, involving floating gate programming (unless denoted otherwise as in the case with crosstalk measurements).

3.2.4 Optimization of Programming Speed and Precision

As described in (2.2.2), floating gate controllers have parameters that are used to fine-tune the writing process. This section describes a method for finding the optimum parameter set, combining fast programming speed with high programming precision. To fully understand it, it is necessary to be acquainted with single parameter functions (see 2.2.2).

The core procedure of this method is taking of the floating gate *response function* as described in (3.2.3) and evaluating the absolute measured values as well as their STD. Through analyzing the *response functions* for different values of a parameter, it is possible to determine which value represents an optimum, and use it while sweeping other parameters, thus narrowing down the search range until optimum values for all parameters are found. Very important is the order in which the optimum parameters are searched out. The search begins with all parameters set to their maximum values, except for *voltagewritetime, currentwritetime* and *acceleratorstep* that have to be set in the middle of their available range.

[7] $f(x) = V_{read}(V_{programmed})$, ideally it is an $f(x) = x$ function.

***Readtime/Biasn* sweep:** The programming speed highly depends on these two parameters, so this part of the process is aimed to find the lowest possible *readtime* value and corresponding *biasn*, allowing for stable readout in the whole range of programmable values. As the recording of one *response function* takes approx. 20 minutes, sweeping over all possible parameter values (*readtime* 64x16 *biasn*) would take two weeks, so the sweeping is done sparsely with the step of 8 in *readtime* and the step of 2 in *biasn*, limiting measurement time to 1 day. The results of this and all following measurements of present section are shown and analyzed in (4.1.4).

***Pulselength* adjusting:** By this time, optimum values of *pulselength* and *biasn* are found. As *pulselength* determines the time scale for all operations by changing the clock of the controller, we can further decrease the duration of programming by scaling *pulselength* down by the same factor that we can upscale *readtime* by (equations 3.1, 3.2).

$$readtime_{new} = readtime_{max} = 63 \tag{3.1}$$

$$pulselength_{new} = \frac{readtime_{optimum}}{readtime_{max}} \cdot pulselength_{max} = \frac{readtime_{optimum}}{63} \cdot 15 \tag{3.2}$$

The result is, that the factual readout time does not change (because *readtime* is also scaled by *pulselength*), but due to faster new clock, functions, controlled by other parameters (e.g. *voltagewritetime*), take less time and have better time resolution because of shorter clock cycles. Of course, this expected behavior has to be proven by a control measurement.

***Voltagewritetime/Acceleratorstep* sweep:** After the optimum timing parameters have been found, programming precision and range have to be optimized. *Response functions* are measured while sweeping *voltagewritetime* and *acceleratorstep* with the single parameter step of 6. The measurement with the least STD of the curve is the optimum point.

***Currentwritetime* sweep:** The precision of voltage cells is affected by changing *acceleratorstep* far more than precision of current cells. This is why *voltagewritetime* is sweeped simultaneously with *acceleratorstep*. To choose an optimum value for *currentwritetime* it is not necessary to vary *acceleratorstep*. It is set to the previously found one and a simple *currentwritetime* sweep is performed. The result with lowest STD is then chosen as an optimum value.

Fine-tuning: The above procedures involved parameter sweeping, using steps greater than the minimum step of 1. For these parameters, sweeps with the minimum step are made (in ranges equal to the previously used step around the found optimum point), to fine-tune the setup and reach the maximum precision point.

3.2.5 Crosstalk During Programming

As pointed out in 2.2.2, the programming of one cell can lead to the change of another cell's value because of line- and column-wise applying of CGL and CGS voltages. Also, because of the tunnel currents, flowing not only through the tunnel transistors, but also

through the readout transistors, the current cells are affected more[8]. The goal of this section is to quantify such crosstalk and find a way to reduce its negative effect on the precision of the programming.

To quantify the crosstalk, following routine, involving one block of floating gate memory cells, has been developed: In the beginning, all the cells in the block are programmed to the same value (e.g. 0.1 V), depending on the future change in them (ΔV). Then the cells of one line, called *control line*, are set to a value in the middle of programming range (approx. 1 V), representing an average use case. The cells in the line are read out, mean and STD are calculated. After that, the values of all other lines in the block are sequently changed by a value ΔV, and after each line being programmed, the *control line* is read out again. This procedure is looped with ΔV being increased (or decreased) after each 24-line run. This way the response of the *control line* to the changes in other lines can be reconstructed and the crosstalk can be quantified. The standard deviation of the *control line* values, in this case, indicates whether all cells in line drift in one direction and with the same rate. The results of crosstalk measurements for both a line with voltage cells and a line with current cells are evaluated in (4.1.5).

Looking forward, the results of these measurements suggested (see 4.1.5) that the crosstalk can be reduced by performing several write cycles in a row. As all the cells are already pre-charged after the first cycle, and their values are *almost* correct (all lines but the last one have experienced crosstalk from writing the consequent lines), value changes through crosstalk during the second *full write cycle* (all 24 lines are programmed) are expected to be much smaller and thus the end values are expected to be closer to the nominal, than they were after the first cycle. One can repeat the procedure for several full write cycles to find out how many are needed to fully eliminate the effect of crosstalk.

To verify this assumption, crosstalk test routine has been modified as follows. Start configuration remains unchanged. *Control line* is written and read out once before programming the neighbor lines. Then all 23 neighbor lines are programmed with the change of ΔV and the *control line* is read out once more to capture the change in its cells after one full write cycle. After that, the whole routine is repeated, but without resetting the floating gate block to the original start configuration, so the end configuration of the first cycle becomes starting point of the second one. Again, the *control line* is read out at the end to capture its value change after two full write cycles and so forth. The results of these measurements are evaluated in (4.1.5).

3.2.6 Estimation of Programming Accuracy and Error Sources

One of the major concerns in developing neuromorphic hardware is limitation of result deviation between hardware emulations and the predictions of the underlying model. The goal is to create hardware that produces the exact results of the model. However, there are some unavoidable error sources, such as transistor mismatch during production of semiconductors, instability of supply voltages, or crosstalk, described in 3.2.5. The overall error of the final system has to be estimated. All analog components contribute to

[8] Meanwhile, this issue has been solved by redesigning the controller and will be applied in the next version of the HICANN chip.

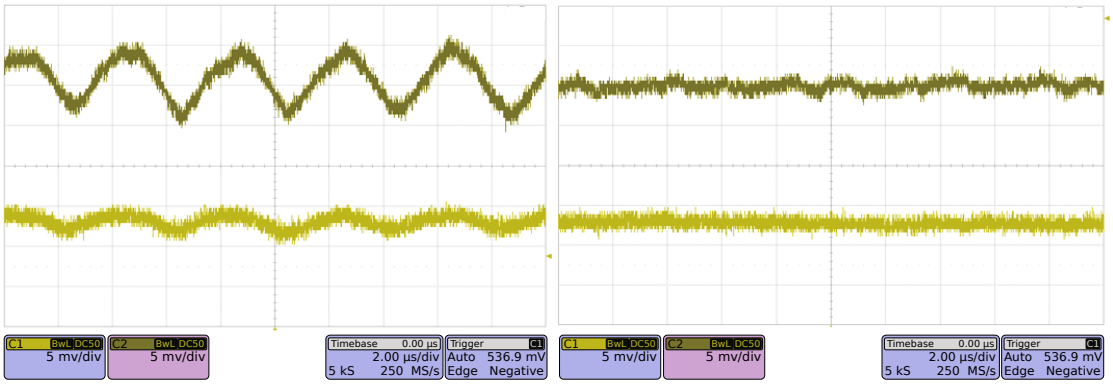


Figure 3.3: Floating gate current cell (top curve) and voltage cell (bottom curve) readout on the oscilloscope. The coupling of an external high-frequency signal is noticeable. On the left is the measurement with internal SEB analog voltage supply, on the right - with external voltage, provided by Tektronix PS2520G power supply.

this error. It also includes the floating gate memory. During the design phase, the error of single floating gate cell programming was expected to amount to 4 mV [25]. Present section focuses on measuring this particular error and evaluating other possible error sources, affecting the accuracy of the floating gate memory output.

To estimate the programming error of the floating gate cells, the results of previous tests, described in (3.2.5) and (3.2.4) are used, meaning that the controller parameters are set to the found optimum, and two write cycles are performed during programming to reduce the crosstalk effect.

The ground measurement of this test is the taking of a *response curve* while programming random values to random cells as described in (3.2.3). To gather sufficient statistics, each possible value (total of 1024) is programmed and read out 80 times for one *response curve*. The STD (or variance) of the read out values is calculated separately for each point.

While conducting these measurements, it has been noticed that the total measured variance of single values originates not only from ADC readout error and the uncertainty in programming the values by the controller, but is also dependent on other factors. Especially the floating gate current cells are affected. So, for example, when the analog output of a current cell is observed on the oscilloscope (fig. 3.3 on the left), it is clear that the read out value oscillates with an amplitude of about 5 mV[9]. The same behavior is also noticed while retrieving the value with the ADC: the variance of repeated measurement on a single cell is much greater with current cells and is value-dependent.

It was found out that this high-frequency signal originates from coupling of the cell output lines with the oscillation of the analog 1.8 V voltage supply of the SEB. Thereupon, said voltage supply was replaced by an external supply, provided by Tektronix PS2520G. After that, the current readout values improved as it can be seen in fig. 3.3 on the right.

[9]The oscillation amplitude is also dependent on the momentary value of the cell.

The voltage cell values are affected much less, probably because of their simpler readout mechanism which is less susceptible to crosstalk. Further tests, regarding other supply voltages, did not show any significant dependence other than on previously mentioned 1.8 V supply. However, STD of single current cell readout (using ADC) still significantly exceeded that of the voltage cells, which was essentially equal to the STD of the lone ADC readout. The reason for this behavior of current cells is the readout value being internally multiplied by 4, leading to a corresponding oscillation increase.

These findings make us define the variance of measured values as $\sigma^2_{tot} = \sigma^2_{adc} + \sigma^2_{prog} + \sigma^2_{read}$. With σ^2_{tot} being the total measured variance, σ^2_{adc} - variance of the ADC, σ^2_{prog} - accuracy of cell programming including the fixed-pattern noise[10], and σ^2_{read} - the readout noise, consisting of e.g. the above described 1.8 V-crosstalk (which has been eliminated by supply replacement), supply voltage oscillations, and other, yet unknown components.

Despite partly unknown origin, σ_{read} can be measured. To do so, the *response curve* of a single cell is taken. As only one cell is involved, the σ_{prog} - component is eliminated and $\sigma^2_{tot\_single} = \sigma^2_{adc} + \sigma^2_{read}$, whereby σ_{adc} is known to be 0.5 mV and range independent.

Knowing σ_{read}, one can easily specify the last unknown: σ_{prog}, which is the most important value, as it defines the floor of the accuracy of the floating gate memory. While components of σ_{read} can be identified and eliminated (as in the case with 1.8 V supply) in the future, σ_{prog} is the value, that cannot be reduced without re-designing the floating gate memory and producing a new chip.

Closer analysis of the data, measured during this test, is presented in (4.1.6).

3.2.7 Table of Test Properties

The following table contains information about chip numbers and floating gate properties used during tests, described above.

Test	HICANN v1			HICANN v2			VDD11	Duration
	Chip	FG Array	Lines	Chip	FG Array	Lines		
3.2.1	-	-	-	15.2	0	0-9	10.5 V	200 h
3.2.2	1.8	2	4-5	15.2	2	4-5	11 V	200 h
3.2.3	1.7	3	8-9	15.1	3	8-9	11 V	5 h
3.2.4	1.7	2	4-5	15.1	2	4-5	10.5 V	70 h
3.2.5	1.9	1/2	9/12	15.1	2	4-5	10.5 V	100 h
3.2.6	1.7	0	4-5	14.2	2	4,7	10.5/11 V	10 h

Table 3.1: Table of Properties of Floating Gate Memory Tests. Test: number of thesis section. Chip: chip label. FG Array: number of used floating gate array. Lines: numbers of used lines. VDD11: setting of VDD11 during test. Duration: test duration in hours, considering pure measurement time for one full test, conducted on one line/array.

[10] Fixed Pattern Noise (FPN) is an effect, originating from transistor mismatch during semiconductor production. Essentially it makes different circuits of identical design produce different output, thus inducing systematic errors in readout patterns.

3.3 Testing of Other Circuitry

Subsequent sections describe testing of the synaptic circuits and calibration of transmission voltages of the L1 bus. These tests, conducted on the prototype system, are for the most part aimed to merely verify the general functioning of the circuits and their compliance with design specifications. The final results of these tests should be acquired, using a wafer-scale system. For example, the calibration of L1 voltages (see 3.3.1) should be carried out on an actual final system to achieve best results. The main goal of measurements, described here, is to develop a calibration method and verify its functionality, so that it could be conducted later on the large scale, whereby more statistics can be gathered and final quantitative results can be derived.

3.3.1 Calibrating L1 Voltages

Serial signal transmitting on the L1 bus is done using LVDS-like technology[11]. It requires a wire pair and transmits signal as voltage difference on it. Both wires have different idle voltages, called V_{OL} and V_{OH}. Basically, depending on these voltages, the transmission quality varies. Also, due to production mismatches, optimum values of these voltages are different for each repeater. To operate a wafer-scale system with lowest possible L1 transmission error rate, one has to find out which voltages represent an optimum operation point for all repeaters in average.

There is one more criterion to be evaluated in this test beside an error rate. The power consumption of such a differential line grows proportionally to $(\Delta V)^2$ ($\Delta V = V_{OH} - V_{OL}$), so one has to make a compromise between the signal-to-noise ratio and feasible power resources. The measurement routine, described below, assists in finding such a compromise. It involves only several repeaters, but it can be easily upscaled to run on larger systems to obtain better statistics.

Firstly, V_{OL} and V_{OH} are set to the designated values. Using a crossbar switch, a sending repeater is connected to another repeater, programmed to save incoming events to the controller's TDI memory. The input of the sending repeater is connected to the background generator. By transmitting neuron number 0, the receiving repeater's DLL is locked. After that, the testing phase begins. The background generator is switched to periodic mode and transmits neuron number n. Subsequently, receiving controller's TDI is read out m times, each time comparing received number to n and counting transmission errors. This is done for all possible $n = 0$ to 63. An error rate of $\frac{en}{64 \cdot m}$ is calculated (with en being total error number) and saved into a file. Also, the error rates for single neuron numbers are saved. The testing phase is finished. Now, previous connection between the two repeaters is cut and the sending repeater is connected to the next receiving repeater. Another test phase is executed. This is done for all possible combinations of sending and

[11] Detailed information on LVDS can be obtained from [33].

receiving repeaters on the whole HICANN chip. Each of the 8 sending repeaters can be directly connected to 8 receiving, making a total of 64 possible configurations.

Now, for the sweeping of V_{OL} and V_{OH}: The test, as described above, is carried out for different values of $V_{CM} = \frac{V_{OH}-V_{OL}}{2}$, but the difference ΔV stays the same during one test run to evaluate error rates at a constant power consumption level.

In the course of this diploma project, test runs with $\Delta V = 50\,\text{mV}$ to $250\,\text{mV}$ have been conducted. Thereby, common mode range of $V_{CM} = 0.55\,\text{V}$ to $1.2\,\text{V}$ has been utilized. Readout count m was set to 45. L1 signal frequency was chosen to be 200 MHz for these measurements to have higher error rates[12]. The results are discussed in section (4.2.1)[13].

3.3.2 Synapse Driver Cascading

Several synapse drivers can be interconnected in a cascading form with the upper driver receiving input signal from its local L1 bus connection and propagating it to its lower neighbor. The lower neighbor, in its turn, forwards the signal to the next lower driver and so forth. This scheme is called *signal mirroring* [11] and is an important component of the L1 routing, as it allows to address several synapse rows using only one L1 line. This can be used e.g. for realizing very strong synapses or increasing synaptic weight resolution by using the same input to operate several synapses in different rows with different g_{max} values.

Due to signal decay, the length of such a driver cascade is limited and depends highly on the clock frequency of the L1 signal. The goal of this section is testing the above constellation and determining how many drivers can be interconnected at a given L1 frequency.

The general approach to the problem is very simple: one successively interconnects an increasing number of synapse drivers n and tests if the last driver in chain is receiving input from its predecessors. Whenever a connection fails, current length n is saved into a file and a new run is conducted, using another set of drivers. After a sufficient amount of runs, distribution of critical chain length n can be analyzed. However simple the idea, its realization requires operation of almost all of chip's components.

Firstly, the background generators are activated in periodic mode to transmit neuron number 0. The merger tree is configured to forward events of 7 BGs and 1 priority encoder to the 8 sending repeaters. 7 of the sending repeaters, connected to the BGs, forward their input into the L1 network, the 8th one receives input from the *indicator neuron* (see below) and is configured to write its input to the controller's TDI memory.

[12] It has been noticed that at lower frequencies, the error rate falls to zero in a wide voltage range, so that the overall error rate/voltage dependence cannot be captured. 200 MHz mode causes higher error rates because of capacitive coupling to neighbor lines. This way the dependence curve in the whole range can be captured and also the worst-case scenario can be evaluated, as 200 MHz is the highest operation frequency planned for the wafer-scale system.

[13] Although the test was designed to utilize all possible 64 repeater combinations, only 8 of them have been used. The reason was the slow readout speed while using JTAG connection. At $m = 45$, total event count amounts to 1,382,400 events per run with 8 repeater combinations, which results in approx. 9 hours of measuring per run. With 64 combinations, the duration of one test run would exceed 3 days, interfering with other scheduled measurements of this thesis.

L1 network is configured in such a way that every sending repeater's output is connected to a different synapse driver. At first, all drivers are disabled and decoder values of all synapses are set to e.g. 15 (binary 1111) to ensure, that no unwanted input is forwarded to the neurons (BGs are transmitting number 0, so all the synapses are in this way set to ignore this input). One of the 64 neurons, connected to the priority encoder in use, is chosen to be *indicator neuron*. It is configured in such a way that every incoming event triggers an outgoing one. Simply put, the firing threshold is set close to the resting potential, and synaptic conductance is programmed to its maximum value.

The whole setup now works as follows: one (*primary*) driver, input of which is connected to an active L1 line, is activated. The decoder value of a synapse, corresponding to both primary driver and the indicator neuron, is set to 0. The weight of this synapse is set to maximum. Now the synapse is periodically stimulating the indicator neuron, which then generates outgoing events that are captured by the repeater controller and can be digitally read out. If this part of the test run succeeds, the synapse is deactivated and another driver is connected to the first one. Now the synapse of the indicator neuron, corresponding to this driver, located below the primary one, is activated. The repeater controller is read out again in order to determine if the indicator neuron is firing, i.e. if the lower driver is functioning as expected. The test is repeated with more interconnected drivers until it fails. The critical length is memorized. To decide if the test failed or succeeded, the TDI memory (which contains 3 last events) is read out three times and the event timings are compared. If the timings have not changed over time, the indicator neuron is obviously not receiving input from the last driver in chain. This indicates test failure. If the timings do change, the test is successful. Also, this timing test is executed each time after one synapse is deactivated but before another one is activated. This confirms that the synapse is indeed off, and prevents measurement errors.

The test, described above, has been carried out for L1 frequencies, ranging from 100 to 250 MHz. The results are presented in (4.2.2).

3.3.3 Linearity of Synaptic Weights

The synapses of the HICANN chip have adjustable weights, meaning that the amplitude of an emitted current pulse can be varied on demand. Synaptic weights are merely numbers that, multiplied with synaptic conductance, alter the strength of the synapse. To have constant transformation factor between model and hardware weights, it is required, that the amplitude of the emitted current pulse is linearly dependent on the programmed synaptic weight value. However, it is not possible to measure the output of the synaptic DAC on the chip directly, and so an indirect method had to be developed.

What can be directly measured in hardware is the membrane potential. In a Leaky Integrate-and-Fire model (LIF), an integral of Post-Synaptic Potential (PSP) over time (after a single synaptic input pulse) is, in the first approximation, linearly dependent on the weight of the firing synapse. To see it, we take the LIF equation:

$$C\dot{U} = -g_{exc}(U - E_{exc}) - g_{inh}(U - E_{inh}) - g_l(U - E_l) = -\sum_i g_i(U - E_i) \qquad (3.3)$$

Where $g_i \subseteq \{g_{exc}, g_{inh}, g_l\}$, $E_i \subseteq \{E_{exc}, E_{inh}, E_l\}$. With g_{exc} - conductance of excitatory synapses, g_{inh}- conductance of inhibitory synapses, g_l - leak conductance, E_{exc} - excitatory reversal potential, E_{inh} - inhibitory reversal potential, E_l - leak reversal potential, C - membrane capacitance, U - membrane potential.

Now, suppose we have the neuron in a stable state (averaged over sufficient time, the mean of the membrane is constant), e.g. no input at all, or even some Poisson-distributed input with unchanging event distribution. We define:

$$U_{eff} = \frac{\sum g_i E_i}{\sum g_j} = const \tag{3.4}$$

Assuming U_{eff} is a constant (averaged over sufficient time), we can write equation 3.3 as

$$C\frac{d}{dt}(U - U_{eff}) = -\sum_i g_i(U - E_i) \tag{3.5}$$

Now rearranging the right half of the equation:

$$C\frac{d}{dt}(U - U_{eff}) = -\sum_i g_i U - \sum_i g_i \frac{\sum g_j}{\sum g_j} E_i = -\sum_i g_i(U - U_{eff}) \tag{3.6}$$

Substituting $U - U_{eff}$ with V and $\sum_i g_i$ with G, we get the equation of the stable state:

$$C\dot{V} = -GV \tag{3.7}$$

Suppose, in this otherwise stable state, one more synapse with weight w fires at some time $t = 0$. An important constraint is that this extra PSP can be seen as a small distortion, i.e. $G \gg g_{exc}$.

$$C\dot{V} = -GV + w \cdot g_{exc}(t)(E_{exc} - V + U_{eff}) \tag{3.8}$$

Substituting $\alpha = \frac{G}{C}$ and $\beta = g_{exc}(t) \cdot \frac{E_{exc} - V - U_{eff}}{C}$ yields:

$$\dot{V} = -\alpha V + w \cdot \beta(t) \tag{3.9}$$

Taking an integral over this one extra PSP:

$$\int_0^\infty \dot{V} dt = -\alpha \int_0^\infty V dt + w \int_0^\infty \beta(t) dt \tag{3.10}$$

The leftmost integral amounts to 0 because $V(0) = V(\infty)$. The second integral reflects the area under the PSP: $\int V dt = A_{PSP}$, and the third one is the contribution from the excitatory conductance $\int \beta(t) dt = B$. Due to the above mentioned constraint, B can be seen as nearly constant compared to the second integral. This gives us:

$$0 = -\alpha \cdot A_{PSP} + w \cdot B \tag{3.11}$$

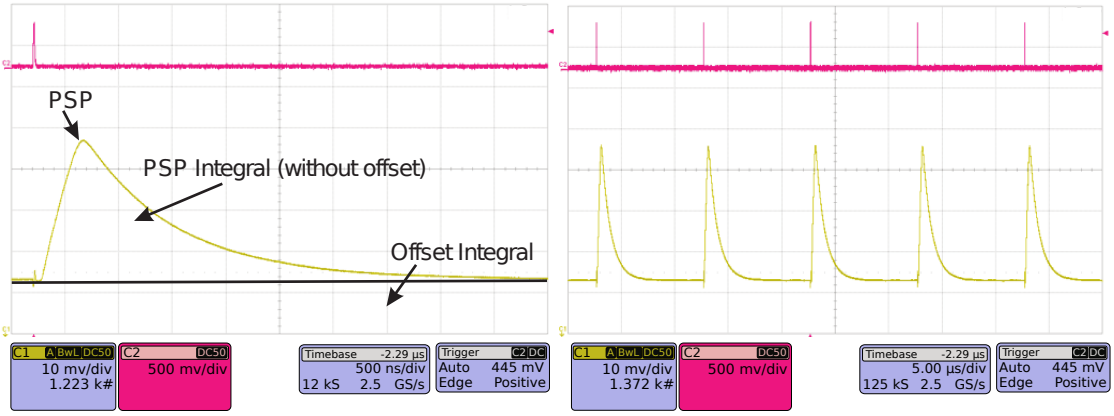


Figure 3.4: Example oscilloscope picture of a PSP. Left: PSP range, used for integral calculation. Difference between total integral and offset is calculated. The top curve is the digital output signal, of the synapse driver used as trigger. Right: Delay between incoming events lets the membrane voltage return to its stable state.

And eventually

$$A_{PSP} = w \cdot \frac{B}{\alpha} \tag{3.12}$$

Which means that the integral over a PSP in a stable state of the LIF neuron is directly proportional to the weight of the firing synapse. We take this approach to our problem of measuring the linearity of synaptic output on the HICANN chip.

For this test we use the first neuron, first synapse driver and the first synapse of the first row[14]. Through adjusting corresponding floating gate parameters, the circuits, controlling exponential and adaptation terms (fig. 2.3, eq. 2.1), are switched off, making the neuron resemble the LIF model. The output of the neuron membrane is connected to one oscilloscope channel, the pulse output of the first strobe line of the synapse driver (see 2.2.4) is connected to the other oscilloscope channel. This way, one can set the oscilloscope to trigger on every pulse of this strobe line that is directly followed by a PSP on the first channel. The time grid of the oscilloscope is adjusted in such way that a PSP completely fits into the picture range (infinite integration in theory is replaced with limited time integration, but the error is negligible due to much greater measurement error). An example of the oscilloscope picture can be seen in fig. 3.4 on the left. Also, the oscilloscope is operated in an averaging mode over 10-20 sweeps to eliminate high-frequency noise on the membrane readout, coming from coupling of supply voltages and clock signals onto the output line.

Background generators, sending repeaters and L1 switches are configured to periodically deliver events to the synapse driver in use. The time between sequent events is set to be much greater, than the decay time of the membrane (fig. 3.4 on the right). This way, the neuron returns to the stable state before the next event arrives.

[14]Due to triggering possibilities while using an oscilloscope.

For the linearity measurement, software by Daniel Brüderle, controlling the LeCroy X-stream interface over Ethernet, has been used. Particularly the function, retrieving momentary voltage trace on the oscilloscope. The measurement course is as follows: firstly, the synaptic input is switched off completely and 10 voltage traces of the inactive state are taken by the oscilloscope[15]. The mean and the STD of the integral are calculated. After that, synaptic input is activated and the weight of the synapse is successively increased. Each time, 10 PSP voltage traces are retrieved from the oscilloscope. Mean integral differences and STDs with error propagation are calculated. The results of this measurement are observable in (4.2.3).

During development of this measurement routine, it was noticed that the linear behavior of the retrieved curves is strongly dependent on two floating gate parameters: the one, regulating the synaptic time constant of the neuron (V_{syntc}) and the one, regulating the amplitude of the synaptic pulse (g_{max}). These dependences were foreseen during chip design [34], however the exact tendencies can only be directly measured. The behavior of synaptic weight linearity was measured in parameter ranges of $V_{syntc} = 1$ to 1.4 V and $g_{max} = 0$ to 0.6 V, that are expected to be adequate for use in the final system. The findings are shown in section (4.2.3).

3.3.4 Fixed-Pattern Noise in Synapses

Because of production mismatches of single synapse circuits, the strength[16] of different synapses with identical configurations is different. To make predictions about the accuracy of the final system, it is important to measure the magnitude of synapse strength mismatch. This is done by evaluating the variation of PSP integrals, introduced in 3.3.3.

Using the same setup and slightly altered routine, PSP areas of a synapse block of 30x30 synapses are measured. Thereby, each synapse column is connected to a different neuron, and each row of synapses is connected to a different synapse driver. Point of interest here is to calculate the variance, coming from the synapse mismatch, but the involved drivers and neurons have mismatches of their own, also affecting the PSP area measurements. The solution lies in evaluation of the measured data.

Calculated over all 900 synapses, the variation σ^2_{tot} is a combination of the driver-to-driver variation, the neuron-to-neuron variation and the synapse-to-synapse variation: $\sigma^2_{tot} = \sigma^2_d + \sigma^2_n + \sigma^2_s$ as in classic case of error propagation. Now, the variance in every synapse column is $\sigma^2_c = \sigma^2_d + \sigma^2_s$ because all of these synapses are connected to the same neuron, and variance in every row is $\sigma^2_r = \sigma^2_n + \sigma^2_s$. Hereby the values of σ^2_{tot}, σ^2_c and σ^2_r are directly derivable from the measured data. So to calculate σ_s one has to solve a simple equation system:

[15] This has to be done to calculate integrals. As the membrane potential is not initially zero, the absolute value of an integral also has offset. Required is the difference between integrals with and without the input to calculate the PSP area.

[16] Amplitude of outgoing current pulse.

$$\begin{aligned} \sigma_{tot}^2 &= \sigma_d^2 + \sigma_n^2 + \sigma_s^2 \\ \sigma_c^2 &= \sigma_d^2 + \sigma_s^2 \\ \sigma_r^2 &= \sigma_n^2 + \sigma_s^2 \end{aligned} \tag{3.13}$$

The solution is:

$$\sigma_s = \sqrt{\sigma_r^2 + \sigma_c^2 - \sigma_{tot}^2} \tag{3.14}$$

Of course, all variances have to have the same magnitude to yield a meaningful result, because if the total error is dominated by only one error source, e.g. $\sigma_n \gg \sigma_s$, it is very probable, that $\sigma_s < \sigma_{\sigma_n}$, meaning that σ_s lies within the error of σ_n and thus the value of σ_s cannot be accurately derived.

Indeed, due to a great number of contributing sub-circuits, the error, originating from uncalibrated neurons is much higher than the other two. To improve this constellation, the neurons have been calibrated by Marc-Olivier Schwartz, meaning that a set of floating gate parameters has been created to even out the behavior of single neurons and reduce the neuron's contribution to the PSP area error to a minimum, originating from inaccuracy of the floating gate programming (see 3.2.6)[17].

The result of this measurement can be observed in (4.2.4).

3.3.5 Table of Test Properties

The following table contains information about chip numbers and other properties, used during tests, described above.

Test	HICANN v1	HICANN v2	Neuron(s)	Synapse(s)	Duration
3.3.1	-	14.2	-	-	400 h
3.3.2	1.8	14.2	4/253	0-223	5 h
3.3.3	1.8	15.1	0	0	20 h
3.3.4	-	15.1	0-30	0-30	20 h

Table 3.2: Table of Properties of Synapse Tests. Test: number of section, where the routine is described. HICANN v1/v2: number of the chip as they are labeled. Neuron(s): numbers of involved neurons. Synapse(s): numbers of used synapses. Duration: test duration in hours, considering pure measurement time for one full test.

[17] At the moment of conducting this measurement, the neuron calibration routine was in the test phase, so a further reduction of σ_n can be expected in the future.

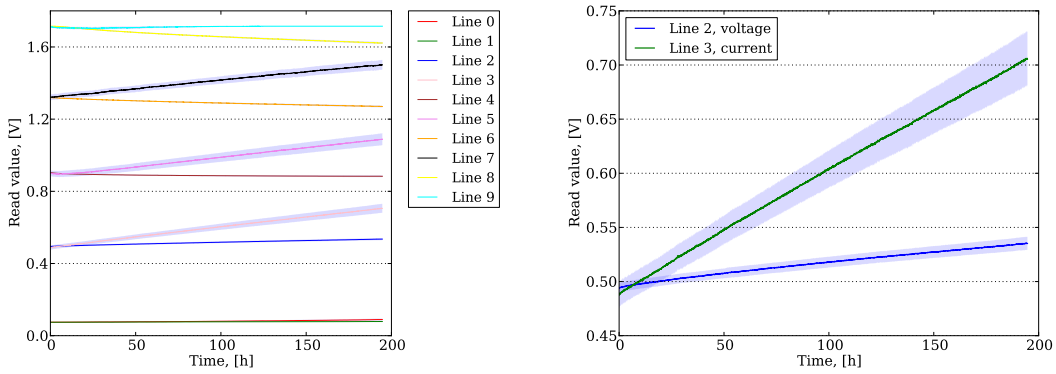
4 Results

This chapter presents results, obtained during previously described tests. The results are given in the same order as test conceptions in chapter 2. The first part handles the testing of the floating gate memory cells, the second part deals with the testing of synaptic circuits and L1 communication.

4.1 Testing of the Floating Gate Memory

4.1.1 Voltage Drift Over Time

This section contains results of the measurements, described in (3.2.1). Figure 4.1 visualizes the readout voltage drift on the floating gate memory cells.



(a) Overview of all 10 tested lines

(b) Zoom to lines 2 and 3

Figure 4.1: Voltage drift on the floating gate memory cells. Solid lines represent the mean of all cells in a line, transparent areas illustrate the corresponding standard deviations. Even lines contain voltage cells, odd lines contain current cells. Average drift rates are 0.2 $\frac{\mathrm{mV}}{\mathrm{h}}$ for voltage cells and 1.0 $\frac{\mathrm{mV}}{\mathrm{h}}$ for current cells.

Figure 4.1(a) shows all 10 involved lines side-by-side, allowing an evaluation of the general drift tendencies. All current cells drift towards a value of approx. 1.7 V and all voltage cells towards approx. 1.0 V. The drift itself originates from the leak tunnel currents, transpiring over CGL, CGS and the readout transistors (see fig. 2.4). The specific resting potentials of 1.7 and 1.0 V are enforced by CGL, CGS, and other transistors serving as potential dividers between 2.5 V (idle voltage on CGS, CGL) and ground on the bias transistor source (see fig. 2.4). Another effect can be seen when considering

lines 0, 1 and 9. These readout values originate not from the respective cells, but from the range limits of the output OP, enforced by voltage drop on the power supply of the OP, changing its rail voltages. These limits constrain the available voltage range to 50 ... 1750 mV on HICANN v2. Due to this effect, it is not possible to track actual cell voltages outside of the aforementioned range.

The drift rates of both cell types are very different. They can be estimated by using plot 4.1(b), showing the zoom to one line of each type. The voltage cells display a drift of approx. 0.2 $\frac{\text{mV}}{\text{h}}$, the current cells of approx. 1.0 $\frac{\text{mV}}{\text{h}}$. The reason for greater drift rate of the current cells is that their output values are internally multiplied by 4 meaning that the actual charge on a current cell, compared to a voltage cell, is much lower. Hence the greater drift rate of an output value at an equal drift rate of an internal value. This is also the reason for many effects, described later, such as higher deviation of the current cells.

Considering an expected programming error of 4 mV for the floating gate memory and experiment durations of several seconds, these drift rates are absolutely uncritical for single experiments, but should be kept in mind while conducting series of experiments, lasting longer than 1 hour. For such series, a periodic re-programming of the memory is recommended.

Also notable is the raise of current cells' STD, which has doubled from 10 to 20 mV during 200 hours of testing. It means that single current cells drift at slightly different rates. However, this drift rate spread is still too small to have an impact on future experiments.

4.1.2 Stress Test

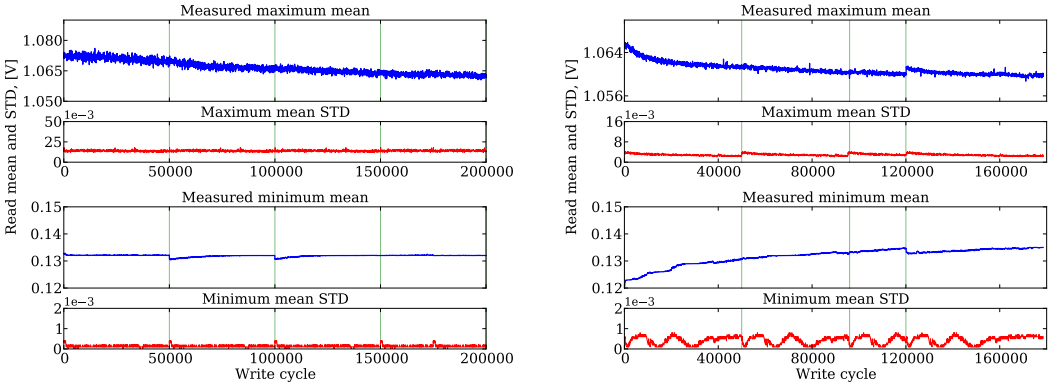
The stress test of the floating gate memory cells has been carried out on both versions of the HICANN chip. Results of HICANN v1 testing are depicted in fig. 4.2, those of HICANN v2 in fig. 4.3.

First of all, it has to be noticed that every plotted curve is a concatenation of 4 sequent measurements with about 50,000 cycles each. This had to be done to prevent data corruption as a result of occasional communication errors[1]. At the transition positions between single measurements (marked by vertical lines), the plot lines often behave unsteadily. This happens because upon testmode restarts, many components of the system are reset. Resulting massive activity on the chip interferes with the analog components by changing the chip's temperature. This results in slightly different read out values (max. 2 mV change of floating gate cell values has been observed), than before the reset. However, this unsteadiness does not disrupt the overall trends that are of interest here.

The *maximum mean* curves show that both current and voltage cell values decrease by approx. 10 mV resp. 5 mV over 200,000 write cycles. At the same time, the *minimum mean*

[1] At the current stage of test software development, data, transmitted after the error, becomes invalid until the testmode is restarted due to shortcomings in CRC check handling. An example of described data loss can be observed in fig. 4.2(b). The 3rd data chunk had to be truncated because it contained invalid data.

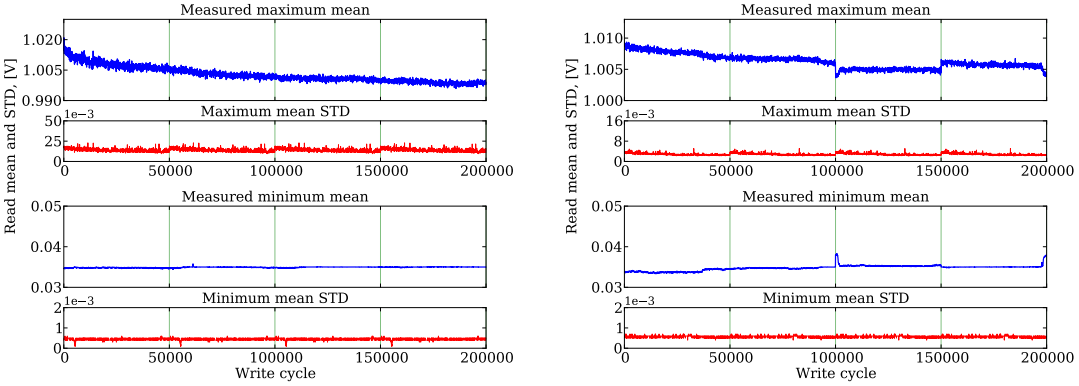
of the *voltage* cells increases by roughly the same amount[2]. Further short measurements were conducted a considerable amount of time later. They have shown that this change



(a) Current cells (b) Voltage cells

Figure 4.2: Stress test of the floating gate memory on HICANN v1. A permanent shift in current cell values of 10 mV over 200,000 write cycles is measured. Voltage cell values shift by 5 mV over the same activity period. Tests were carried out at VDD11 = 11 V.

in cell behavior is permanent. The reason could be the improvement of insulation on the CGS and CGL transistors as a result of eroding the impurities from the insulator layer through high voltages and tunnel currents.



(a) Current cells (b) Voltage cells

Figure 4.3: Stress test of the floating gate memory on HICANN v2.

An important conclusion, resulting from these measurements, is the need for a periodic re-calibration of the neurons (e.g. after every 50,000 write cycles), because the values of

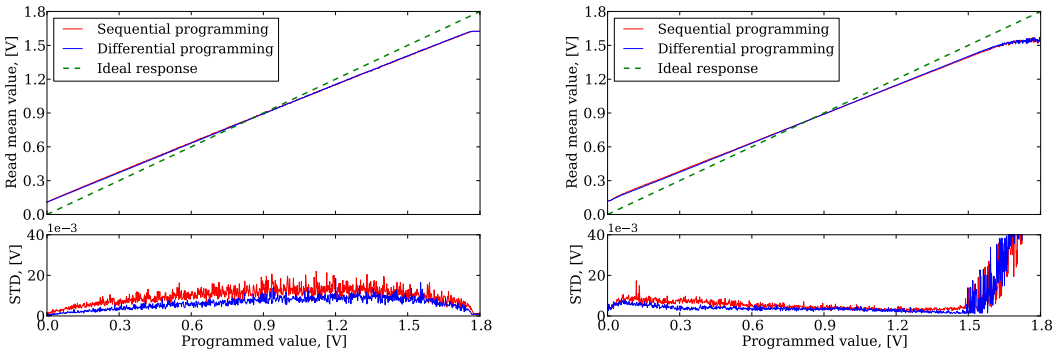
[2]The unchanging values of the *current* cell minima and the very low STDs of these measurements indicate that the real values were outside of the OP range.

single neuron parameters change with passed write cycles. Otherwise, the deviation, accumulated over many neuron parameters, can become a systematic error source.

On the other hand, a very positive result can be noted. The STDs of all measurements have been stable throughout the tests, which indicates that no cells have been damaged, while conducting the tests at VDD11 = 11 V. This means that future operation at 11 V is possible[3]. Also, the general robustness of the cells has been proven. Over the whole test duration, more than 100,000,000 write cycles (counting all 512 involved cells) have been performed and no cell has become defective.

4.1.3 Differential Programming

The results of comparison of two programming schemes, described in (3.2.3), are presented in this section. Fig. 4.4 shows the *response curves*, originating from HICANN v1 measurements, fig. 4.5 features those of HICANN v2 measurements.



(a) Current cells

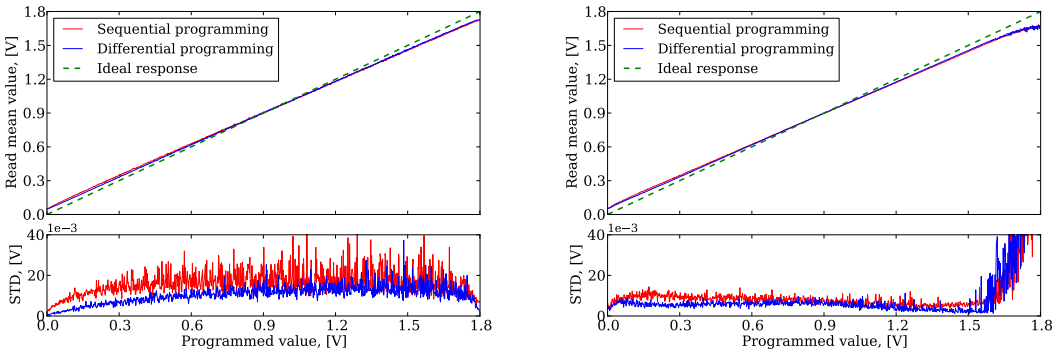
(b) Voltage cells

Figure 4.4: Programming scheme comparison on HICANN v1: sequential vs. differential programming. Due to the crosstalk effect, the sequential programming scheme displays higher deviation. Tests were carried out in VDD11 = 11 V mode.

The plots are split in two parts, each containing two curves, depicting the response behavior of the floating gate memory cells using both programming schemes that were introduced in (3.2.3). The upper part contains means of measured values, the lower one presents the corresponding deviations. Plotted for reference are also the ideal *response functions*.

The exact shapes of the curves can be explained as follows: The overall linear behavior meets the initial expectations. However, the edge points and the declinations of the mean curves differ from the ideal case. There are several reasons for this response.

[3]Operation at 11 V is very important, as the tunnel current is exponentially dependent on the voltage difference. Therefore, setting VDD11 to 11 V improves the programming speed and the available voltage range of the floating gate memory. More information on this topic in available in sections (4.1.3) and (4.1.6).



(a) Current cells

(b) Voltage cells

Figure 4.5: Programming scheme comparison on HICANN v2: sequential vs. differential programming. The results are equivalent to those of HICANN v1.

Firstly, considering the current cells' curves, one may notice that the starting and the end points are not at 0 and 1.8 V as expected, but slightly higher resp. lower. This is a measurement artifact, coming from the limited range of the read-out OPs, already mentioned in previous sections. When comparing respective plots of both chip versions, one can observe an improvement of the range on HICANN v2. This has been achieved through redesigning the power routing of the chip, causing less voltage drop on the OP power supply, so that better rail voltages could be reached.

When considering the voltage plots, it is at hand, that the values do not reach the upper OP rail limit. The curves unexpectedly flatten at the high-value end, and the corresponding STDs increase. The reason for that is that the voltage cells do not reach their expected values at the end of the write cycle. In general, voltage cells need longer and stronger write pulses than current cells to reach their desired values (because, as pointed out before, the internal values of current cells are much lower than those of voltage cells). This issue will be covered more closely in section (4.1.4).

The deviating declinations of the measured curves, as opposed to the ideal one, come from the imprecision of the DACs, used to compare the momentary cell voltage during programming, and the read-out OPs. Obviously, these circuits are not calibrated and have different reference voltages, which results in non-zero linear discrepancy between the values, programmed with the help of the DACs and those, read out by the OPs. This effect, however, yields no issue for the error-free operation of the system, as the neuron calibration does not require calibrated voltage values. More important is the linear dependency, which is clearly observable in the plots.

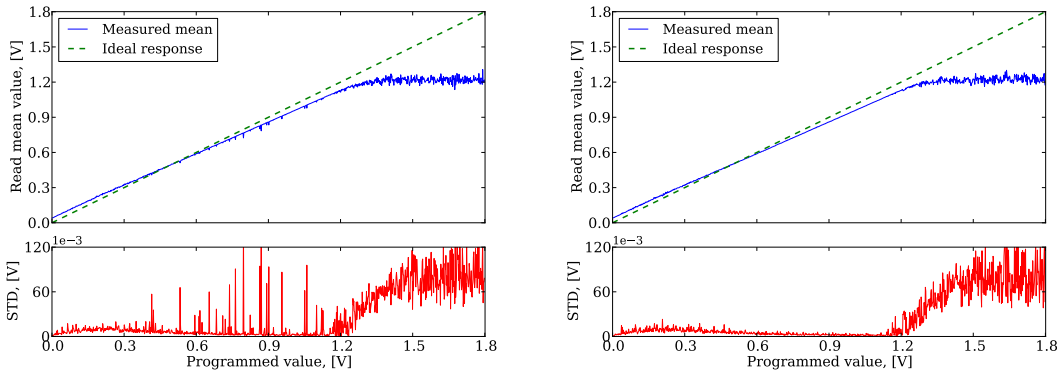
By comparing each pair of curves, taken using both programming schemes, one can clearly see that the resulting values are nearly equal in both cases, with standard deviation of the sequential scheme being slightly higher, than the STD of the differential scheme. Cause of this is the crosstalk between single programmed lines (see (3.2.5) and (4.1.5)).

The linearity of all curves is near to ideal and the programming range is satisfying. This way both evaluated schemes have proven to be applicable for operation on the system. Due to its convenience, the differential programming scheme has been chosen for use in further tests.

4.1.4 Optimization of Programming Speed and Precision

This section handles the process of optimization of the floating gate controller parameters as depicted in (3.2.4).

As the workflow in cases of both chip versions is identical and the resulting plots are very similar, only the plots, originating from the second version's testing, are used in this section with the exception of those, demonstrating functional differences between the versions.



(a) Response curve at *readtime*=29, *biasn*=9

(b) Response curve at *readtime*=37, *biasn*=9

Figure 4.6: Example of cell undercharging due to low *readtime* setting (on the left). The measurement with increased *readtime* shows no such effect.

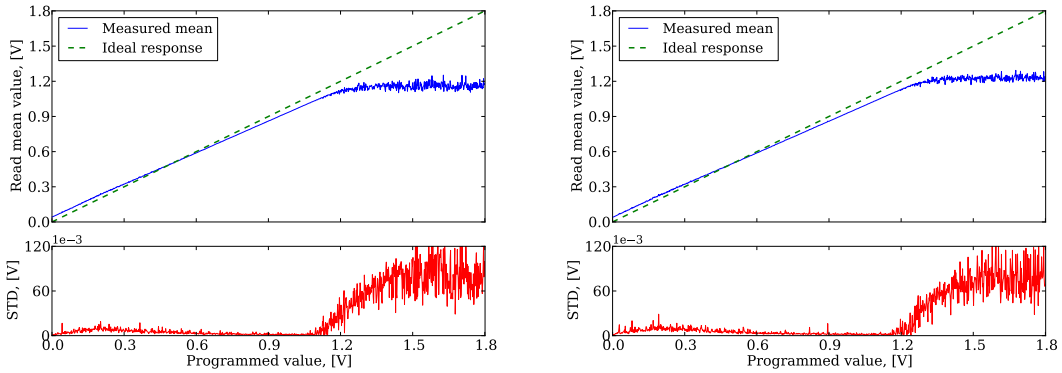
As a result of parameter sweeping, described in (3.2.4), one obtains a set of *write response curves*. Basic forms of those were explained in section (4.1.3). Based on the properties of the curve, corresponding to certain parameter settings, it is possible to choose the optimum of the set. Following subsections explain the criteria, used to select best parameters for the final settings.

It also has to be remarked that presented testing was carried out in VDD11 = 10.5 V mode. The voltage cells in this mode do not reach values, larger than 1.2 V as can be seen in corresponding plots. However, the parameter optimization workflow is not impaired by the choice of VDD11 mode. Further discussion of this issue and an example measurement in VDD11 = 11 V mode can be seen at the end of the present section.

***Readtime/Biasn* sweep:** Fig. 4.6 and 4.7 show typical *response curves*, obtained while sweeping readtime and *biasn* parameters.

Three dependencies can be identified from those plots. Fig. 4.6(a) shows several STD peaks, and corresponding lows on the mean curve. The reason for these is that in some

cases low comparator pre-charge time causes it to incorrectly qualify an undercharged cell as fully charged. Fig. 4.6(b) indicates that setting readtime higher solves this issue.



(a) Response curve at *readtime*=53, *biasn*=1

(b) Response curve at *readtime*=53, *biasn*=13

Figure 4.7: Demonstration of voltage range dependence on *biasn*. Higher *biasn* setting allows for larger available programming range.

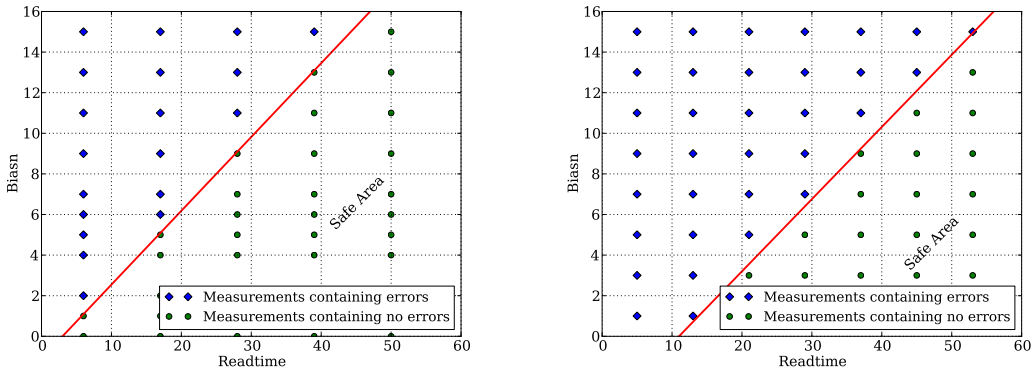
The second dependence is depicted in fig. 4.7. One can see that the available voltage range of the cells is dependent on the *biasn* parameter. This is obvious, because *biasn* controls the bias current through the readout transistor (fig. 2.4), thus conditioning the time, needed for the charging of the comparator capacitance, and, as a consequence, the maximum and minimum achievable values. *Biasn*=1 in fig. 4.7(a) narrows the range down to approx. 1.1 V from 1.2 V in 4.7(b) with *biasn*=13.

The third dependence is the combination of the effects, described above, and can be abstracted from fig. 4.8. The higher the *biasn* parameter becomes, the smaller gets the undercharge-free readtime range, because the time, needed for accurate comparator operation, grows with sinking bias current[4]. Also noticeable is the reduction of the error-free zone of HICANN v2 (fig. 4.8(b)) as compared to HICANN v1 (fig. 4.8(a)). The reason is believed to be the change of voltage supply for the floating gate memory controllers.

The plots in fig. 4.8 are used to choose a parameter pair, lying in the undercharge-free *safe area*, allowing for a satisfactory speedup through *pulselength* reduction, and providing wide programming range for the voltage cells. Also a safety distance to the unsafe area has to be reasonable. As balanced-out values for HICANN v1 have been chosen: *readtime*=32, *biasn*=6, for HICANN v2: *readtime*=38, *biasn*=5.

***Pulselength* adjusting:** As shown in eq. 3.1 and 3.2, *pulselength* and *readtime* parameters have been adjusted to *readtime*=63 and *pulselength*=8 for HICANN v1, and *readtime*=63 and *pulselength*=9 for HICANN v2. Such an adapting of *pulselength* corresponds to reducing the programming time by nearly half. *Biasn* settings have been left unchanged pending the fine-tuning step.

[4] *Biasn* controls the current bias inversely, hence the name bias**n** for "negated".



(a) HICANN v1 (b) HICANN v2

Figure 4.8: Plots, depicting the dependence of cell undercharging on *biasn* and *readtime*. On the right of the experimentally determined red line lies the *safe area*. The parameter tuple has to be located in this area in order to guarantee error-free cell charging.

A control *response curve* measurement has confirmed that performed parameter adjustment has not changed the *response function* in terms of programming range or occurring cell undercharge errors. This way the optimization process can be continued.

***Voltagewritetime/Acceleratorstep* sweep:** In this step, again, two parameters are sweeped simultaneously, producing an array of *response curves*. From these curves two dependencies can be derived. They must be evaluated in order to determine the optimum parameters.

Fig. 4.9 shows these dependencies. 4.9(a) and 4.9(b) depict how the maximum programming range depends on *acceleratorstep*. Lower *acceleratorstep* yields wider programming range, because the writing pulse duration increases quicker. However, if *acceleratorstep* gets too low, the accuracy of the programming suffers as suggested by an increased STD in fig. 4.9(a). This happens because the pulse duration grows too fast and lower values are programmed using high pulse lengths, which leads to overshoots in programmed values, causing inaccuracy.

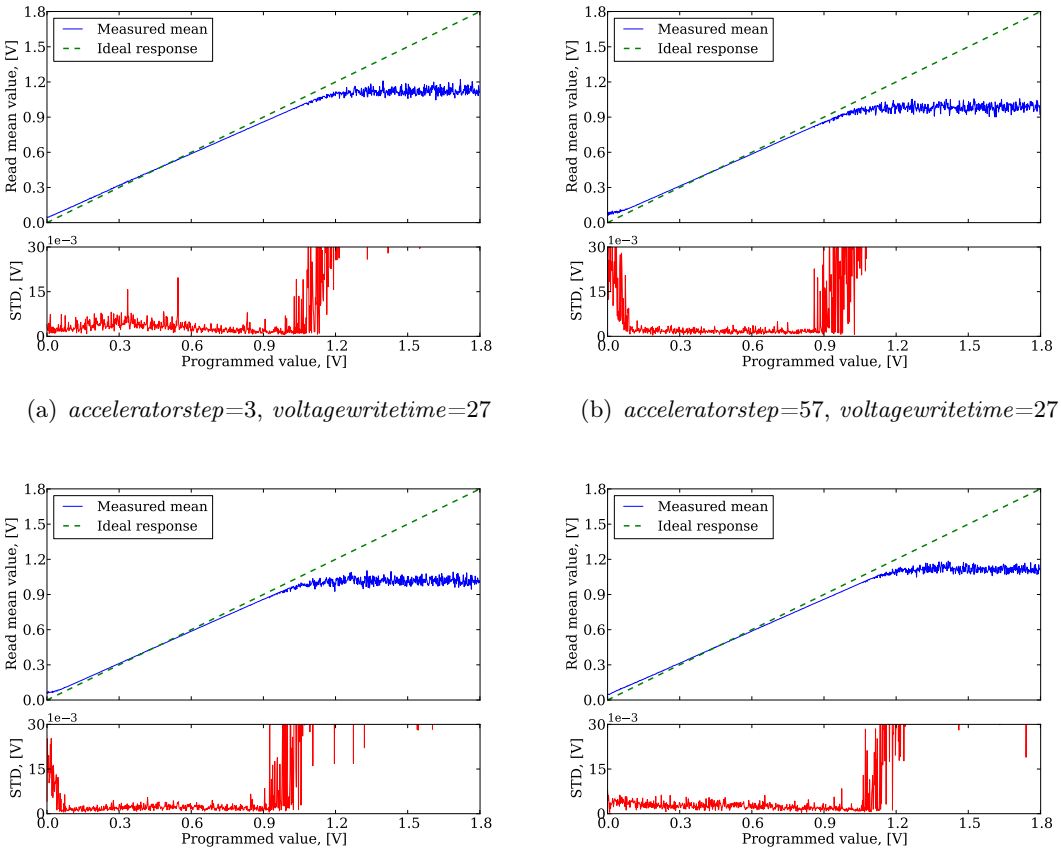
Fig. 4.9(c) and 4.9(d) show the *voltagewritetime* dependence. Basically, it is the inverted *acceleratorstep* dependence: the higher *voltagewritetime*, the higher values are programmed accurately, but if the parameter gets too high, written values become imprecise.

Both parameters are sweeped simultaneously because of their strong effect on resulting accuracy and programming range. Small changes in parameters can have great impact on those properties. This is why both dependencies are best observed at the same time.

Based on *response curve* plots, *voltagewritetime*=21 and *acceleratorstep*=21 have been chosen provisionally for HICANN v1, the second version of the chip takes *voltagewritetime*=15 and *acceleratorstep*=15.

***Currentwritetime* sweep:** After all other parameters have been set, optimum *cur-*

rentwritetime can be determined. Basically, this parameter has the least impact on the programming process and it is possible to find a reasonable setting for *currentwritetime* independently from another parameters, this is why it is handled last.



(a) *acceleratorstep*=3, *voltagewritetime*=27

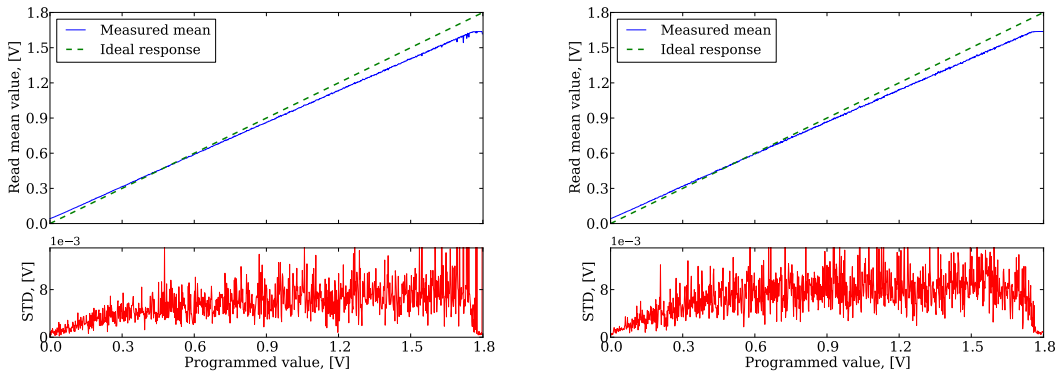
(b) *acceleratorstep*=57, *voltagewritetime*=27

(c) *acceleratorstep*=15, *voltagewritetime*=3

(d) *acceleratorstep*=15, *voltagewritetime*=57

Figure 4.9: Demonstration of programming range and accuracy dependencies on *acceleratorstep* and *voltagewritetime* parameters of the floating gate memory controller. Top subfigures depict *acceleratorstep* variation at constant *voltagewritetime*, bottom subfigures show the effect of *voltagewritetime* variation at constant *acceleratorstep*.

Fig. 4.10 shows two *response curves* of floating gate current cells. It suggests that inaccuracy of lower values grows with increasing *currentwritetime*, while that of higher values sinks, as it is expected. The behavior is analog to that of *voltagewritetime* variation, but the absolute quantitative impact is much smaller, because the internal current cell values are lower than those of voltage cells and therefore require less charging pulses. Thus the whole output range of 1.8 V can already be used with low *currentwritetime* setting. After evaluating all resulting *response curves*, *currentwritetime*=10 was chosen for HICANN v1, and *currentwritetime*=1 for HICANN v2.



(a) Response curve at *currentwritetime*=4

(b) Response curve at *currentwritetime*=58

Figure 4.10: Adjusting *currentwritetime* parameter: due to lower actual values of the current cells, the effect of *currentwritetime* variation is less distinctive, than that of *voltagewritetime* adjustment.

Fine-tuning: In this phase, short sweeps of *biasn*, *acceleratorstep* and *voltagewritetime* parameters have been performed around previously determined provisional optimum points to determine the best settings. Table 4.1 contains the final results.

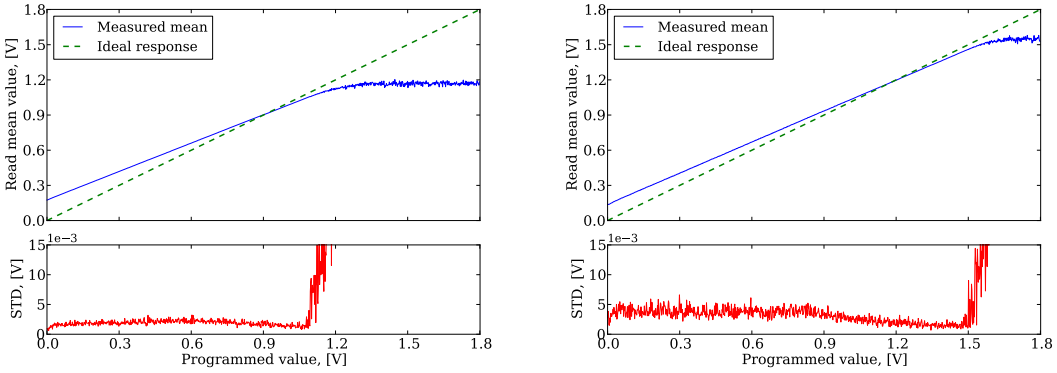
The *response curves*, taken using optimum parameters, can be seen in section (4.1.6) as part of the precision evaluation measurements. Before performing those final measurements, the crosstalk had to be quantified and corresponding measures had to be taken (topic of section 4.1.5).

HICANN	*pulselength*	*biasn*	*acceleratorstep*	*voltagewritetime*	*currentwritetime*
version 1	8	4	20	18	10
version 2	9	5	9	15	1

Table 4.1: Optimum floating gate memory controller parameters for both chip versions.

VDD11 modes and the meaningful voltage range: We define *experimentally meaningful voltage range* as the range between 0 and 1.5 V. Values outside of said range are not foreseen for use in the final neuromorphic system, thus it is sufficient for the optimization to only consider 0 to 1.5 V as desired range and neglect the rest of the original 0 to 1.8 V range.

The VDD11 = 10.5 V mode constricts the maximum reachable voltage vastly. The impact can be observed in fig. 4.11. Optimization process, described here, was carried out in 10.5 V mode, but the obtained optimum parameters can also be used in 11 V mode. In such case, as fig. 4.11(b) shows, the available range is expanded to cover the entire *experimentally meaningful range*. Of course, ideally, the whole optimization process should be repeated for VDD11 = 11 V. The depicted measurement has been



(a) VDD11 = 10.5 V mode

(b) VDD11 = 11 V mode

Figure 4.11: *Response curves* of the voltage cells, taken in both VDD11 = 11 V and VDD11 = 10.5 V modes. At 10.5 V the programming range is constricted, at 11 V the entire *experimentally meaningful range* is covered.

merely carried out to show that the hardware is capable of covering the whole desired voltage range when used with originally projected VDD11 setting.

Validity of the determined parameter sets: As only one controller was used to carry out the measurements, a question arises as to how dependable the determined parameters are if used with another controllers/chips.

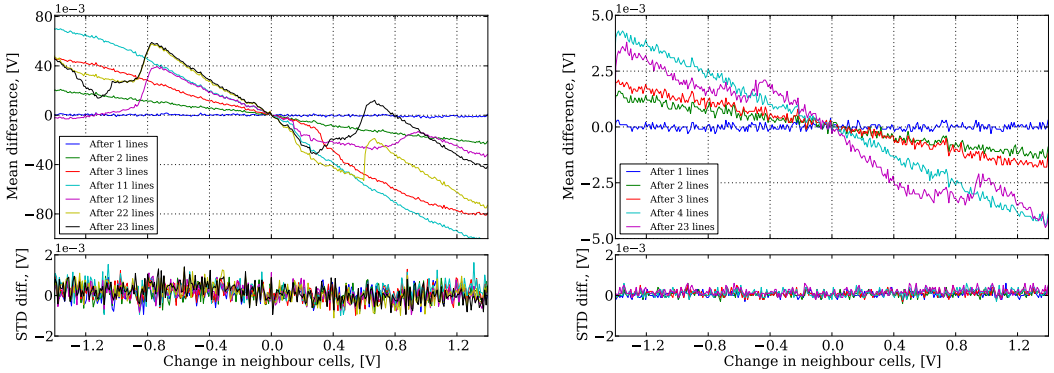
Measurements, involving other controller instances and chips, have shown that available programming voltage range varies from instance to instance. However, the *experimentally meaningful range* is not substantially constricted.

Another important question is the necessity of repeating the optimization process. Obtained results show that optimum settings significantly vary between chip versions, so for every new chip release a new set of parameters is required. Also, judging from the comparison of fig. 4.11(a) and 4.11(b), the STD of the 11 V curve is partially the double of the 10.5 V curve STD, especially at lower values. This certainly indicates that shorter pulses and/or greater accelerator steps are needed at 11 V and thus the parameter values have to be recalibrated for use in VDD11 = 11 V mode.

In general, the optimization process is associated with a tradeoff between programming range and speed. Using yet lower *pulselength* values, the range can be adjusted to an absolute acceptable minimum, yielding faster writing. But then every controller should be calibrated separately and thus each of them will have its own parameters that need to be stored and loaded. Otherwise, one can increase *pulselength*, thus introducing a "safety margin" around the minimum *experimentally meaningful range*, which is sufficient for one set of parameters to be used on all controllers, but this effect will be coupled with some writing speed loss.

4.1.5 Crosstalk During Programming

This section introduces results of crosstalk measurements, described in section (3.2.5). As in the previous section, due to a great similarity of test results, only those of HICANN v2 are depicted here.



(a) Crosstalk on current cells

(b) Crosstalk on voltage cells

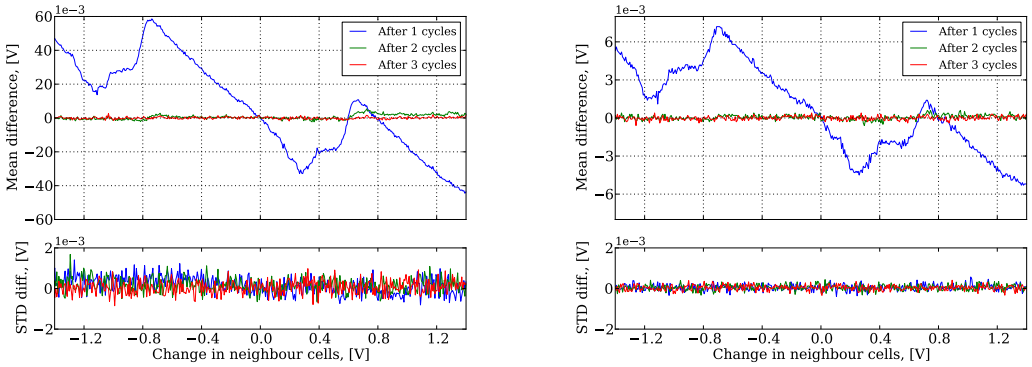
Figure 4.12: Crosstalk measurement using sequential writing. Each curve represents *control line* state after programming of a denoted neighbor line. For better clearance, only those states with most significant changes are plotted. Maximum shift of nominal *control line* values, following the programming of 23 neighbor lines, amounts to 80-160 mV for current cells and 6-12 mV for voltage cells.

Fig. 4.12 presents results of a sequential crosstalk measurement (see (3.2.5)). Plotted are the changes in *control line* voltage against the change in neighbor lines' voltages ΔV. Each curve stands for a neighbor line being written and the *control line* being read out afterwards, measuring crosstalk impact from said neighbor line programming. The change in neighbor lines has been sweeped in the range of $\pm$ 1.3 V, which covers the whole available range in VDD11 = 10.5 V mode. Also plotted is the corresponding STD change of the *control line*, indicating if crosstalk affects all cells of the line equally or if there is a drift speed spread between single cells of the line.

Measurements have shown that the exact shape of mean difference curves varies from chip to chip and even from line to line. STD difference is, on the other hand, stable and is close to zero, suggesting that all cells are equally impacted by the crosstalk and drift with equal rates. In general, it is not necessary to explain exact course of the mean curve. Rather more important are the maximum crosstalk values and the regions, where the crosstalk effect can be neglected. These properties were consistent over all measured chips and fig. 4.12(a) and 4.12(b) are representative examples of crosstalk on current resp. voltage floating gate cells.

Maximum measured crosstalk effect on the current cells was estimated at 80-160 mV, that of the voltage cells lies at about 6-12 mV[5]. These values cannot be tolerated during

[5]The reason for such huge difference is that, due to design specifics, the tunneling on the current cells



(a) Current cells

(b) Voltage cells

Figure 4.13: Crosstalk dependence on number of performed write cycles. Each curve represents the state of the *control line* after denoted number of full 24-line write cycles. 3 write cycles are sufficient to cancel out the crosstalk effect.

final system operation and have to be reduced. When looking closely at the measured mean difference curves, one can notice that they are rather flat in a $\pm 50\,\mathrm{mV}$ region around zero. In this area, crosstalk effect can be neglected. It means that minor changes in neighbor lines do not produce significant crosstalk on the *control line*, and one can use this effect to stabilize the writing process.

The most simple modification to the programming process would be performing multiple write cycles on entire 24 lines until the crosstalk effect is neglectable. During the first cycle the lines are programmed with max. 80-160 mV deviation. It means that during the second write cycle, the cells have to be corrected by maximally this amount. This way each line induces less crosstalk on its neighbors with each write cycle, and ideally, at some point, the resulting crosstalk can be ignored. To test this programing scheme, series of measurements have been conducted. Their results are presented in fig. 4.13. One can see that already after 2 write cycles, the crosstalk is almost nullified. After 3 cycles the *control lines* reach their desired values. Crosstalk effect is canceled out and the writing process is stable.

Results of this section provide very important information regarding floating gate memory use in the final system. First insight is that the programming cycle has to be executed 2-3 times to completely cancel out the crosstalk, the second one is that the writing has to be conducted block-wise. One cannot alter values of one line without changing the others. These facts have to be considered, while designing control software for the final system.

occurs not only at the dedicated CGL and CGS transistors, but also, in an undesired manner, at the readout transistors. This flaw will be corrected in the next chip version.

4.1.6 Estimation of Programming Accuracy and Error Sources

This section presents results of accuracy examinations, conducted on the floating gate memory cells. The measurements are described in section (3.2.6) and already incorporate results of previous sections such as usage of differential programming scheme, optimized controller parameters and two sequent write cycles. Again, only plots from HICANN v2 measurements are used due to great similarity of both chips' results.

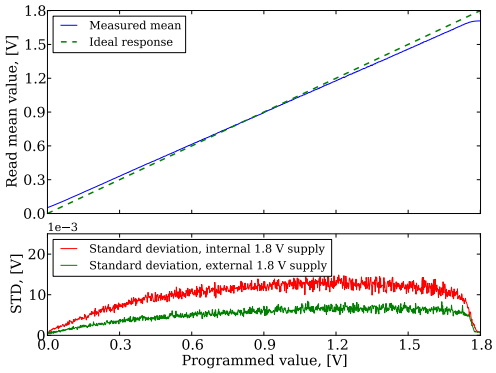


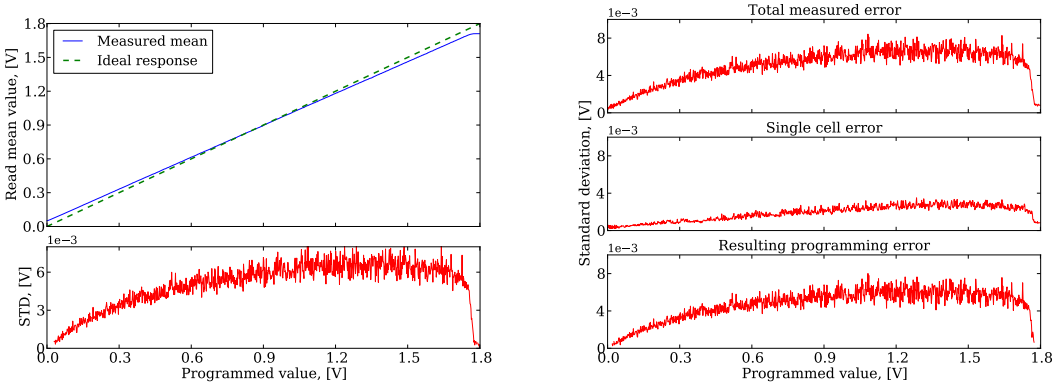
Figure 4.14: Illustration of an STD difference while using SEB's internal and external 1.8 V supply. The measured mean *response curve* is identical in both cases. An unstable internal voltage supply causes the STD of the curve to rise.

Fig. 4.14 shows an example of a *response curve*, measured using both internal and external 1.8 V analog voltage supply. At hand is the vast impact of a less stable internal 1.8 V supply on the curve's STD. It is expected for the supply of the final system to be very stable, so all further measurements are conducted using the external voltage to yield lowest possible deviation.

In (3.2.6) the total variance of a measurement was defined as $\sigma^2_{tot} = \sigma^2_{adc} + \sigma^2_{prog} + \sigma^2_{read}$. Main interest point is the programming accuracy of the floating gate controllers, i.e. σ_{prog}.

Fig. 4.15 shows a *response curve* of the current cells, taken using optimized programming scheme, ergo having the lowest possible deviation at this point of system development. 4.15(a) depicts the *response curve*, while 4.15(b) contains the σ_{tot} components: σ_{tot} itself is on top, σ_{read} in the middle and the resulting $\sigma_{prog} = \sqrt{\sigma^2_{tot} - \sigma^2_{adc} - \sigma^2_{read}}$ at the bottom ($\sigma_{adc} = 0.5\,\text{mV}$ as determined earlier).

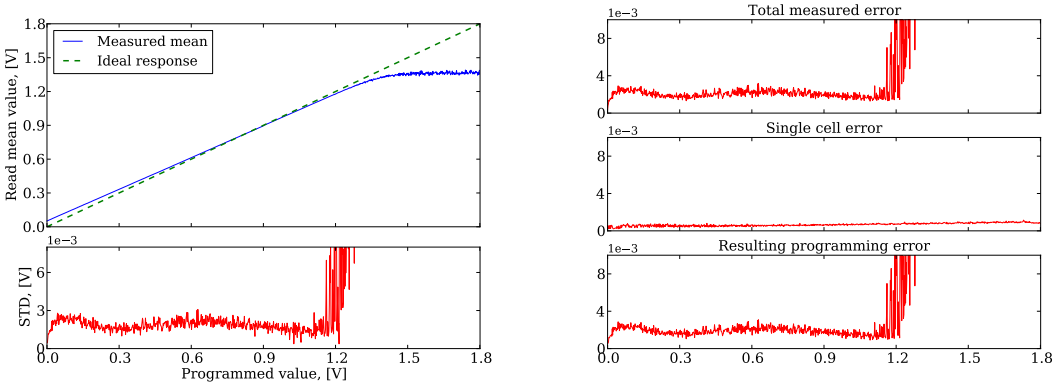
As can be seen from the plot, the expected deviation of 4 mV is exceeded by approx. 2 mV. This indicates that the deployment of the floating gate current memory cells in their present design can be problematic because of the resulting neuron parameter errors. However, design changes, reducing current cell crosstalk, have already been implemented into the next version of the HICANN chip and are expected to diminish the programming error. Also one must consider the fact that the readout values of the current cells are multiplied by four in the voltage-current conversion process. However, the values driving



(a) Optimum *response curve* (b) STD calculation visualization

Figure 4.15: Measurement of floating gate current memory cells' precision. On the left is the optimum measured *response curve* with programming error σ_{prog} as STD. On the right are the variance components, used to calculate σ_{prog}.

the neuron circuits, are not multiplied, therefore the absolute deviation at the neuron is lower. Moreover, the above results represent worst case programming scenario using random programming. In practice, the programmed values are expected to be close to each other, because the neurons of one chip are going to have similar parameters in an average case. This means less crosstalk will take place within the lines and hence the average case deviation will be lower.

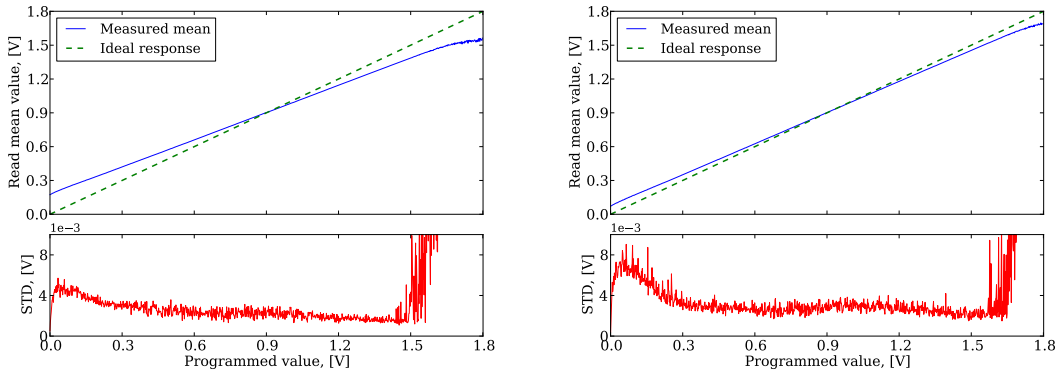


(a) Optimum *response curve* (b) STD calculation visualization

Figure 4.16: Measurement of floating gate voltage memory cells' precision.

Fig. 4.16 depicts the analogous measurement of the voltage cells. Apart from the programming range, constrained by operating in VDD11 = 10.5 V mode, the measured STD values are satisfactory. The error is contained well below the 4 mV limit, meaning

that voltage cells can be unobjectionably used in the final system.



(a) HICANN v1

(b) HICANN v2

Figure 4.17: Measurement of floating gate voltage memory cells' precision in VDD11 = 11 V mode using the optimum parameters, determined for VDD11 = 10.5 V mode. Even without the recalibration the results are satisfactory.

Additionally, measurements on voltage cells at VDD11 = 11 V have been conducted to qualitatively observe the programming error. The results are depicted in fig. 4.17. For these measurements, the controller parameters, optimized for 10.5 V mode have been used, so the deviation is not at its total minimum. Nevertheless, the programming error within the entire *experimentally meaningful range* is acceptable as it is, and is expected to be further decreased once the optimum parameter set for the VDD11 = 11 V mode is found.

4.2 Testing of Other Circuitry

4.2.1 Calibrating L1 Voltages

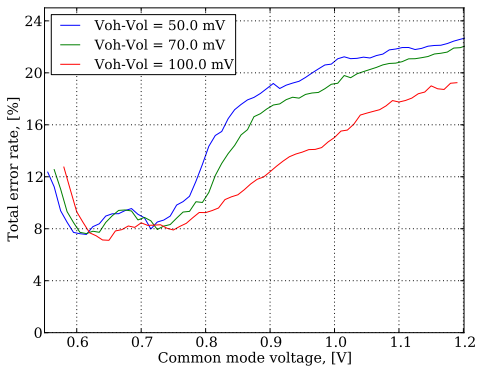
Presented below are the results of L1 communication voltages calibration, characterized in section (3.3.1).

During the design phase, the projected power consumption of the system led to the desired value of $\Delta V = V_{OH} - V_{OL} = 150\,\text{mV}$. Transistor-level simulations have shown that the optimum common mode voltage $V_{CM} = \frac{V_{OH}-V_{OL}}{2}$ should amount to approx. 800 mV at $\Delta V = 150\,\text{mV}$ [35]. To have an extensive overview of the L1 system behavior and be able to confirm the prediction of the simulation, the ranges of V_{CM} and ΔV for the measurements were made large enough to see the overall trends.

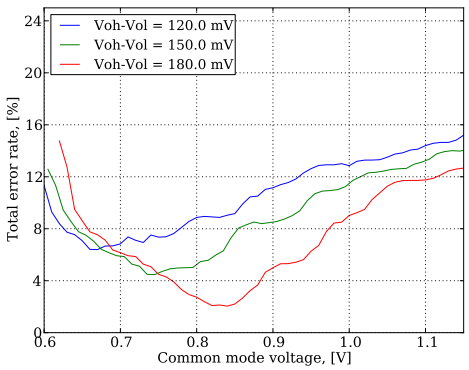
Fig. 4.18 shows error rates in L1 communication while operating at certain common mode and differential voltages. To be able to correctly interpret the results, one has to remember that these measurements were carried out using 200 MHz clock to generate the signal. This frequency is the highest possible setting, considered for the final system. It

has been chosen for testing to artificially make the system produce more errors, because at the lower frequencies the error rate was often 0 and thus the trends were non-observable.

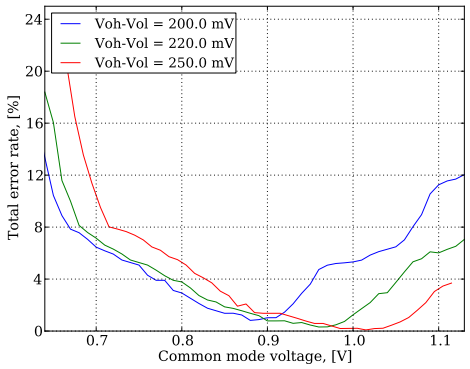
From fig. 4.18(a) one can see, that at low ΔV values the error rates are very high, because the signal-to-noise ratio is low and the decaying signal often gets overwhelmed with the noise while traversing the wires.



(a) ΔV = 50 to 100 mV



(b) ΔV = 120 to 180 mV



(c) ΔV = 200 to 250 mV

Figure 4.18: L1 voltage calibration: error rate dependence on the common mode voltage V_{CM} taken for different values of differential voltage ΔV at 200 MHz L1 clock frequency. Observable is the overall diminishing of the error rate with growing ΔV. An optimum V_{CM} for a given ΔV can be obtained from the plots.

Fig. 4.18(b) presents ΔV region with constantly diminishing error rates as the signal-to-noise ratio improves. One can also see, that the simulation prediction was almost correct, and the optimum V_{CM} for ΔV = 150 mV amounts to approx. 750 mV.

Finally, fig. 4.18(c) shows that at ΔV values of 200 mV and higher, the error rate falls to nearly zero. Which means that at the current state of L1 communication development, system operation at 200 MHz would imply raising ΔV over 200 mV which would increase

the power consumption, limited by the available supply and maximum heat dispersion.

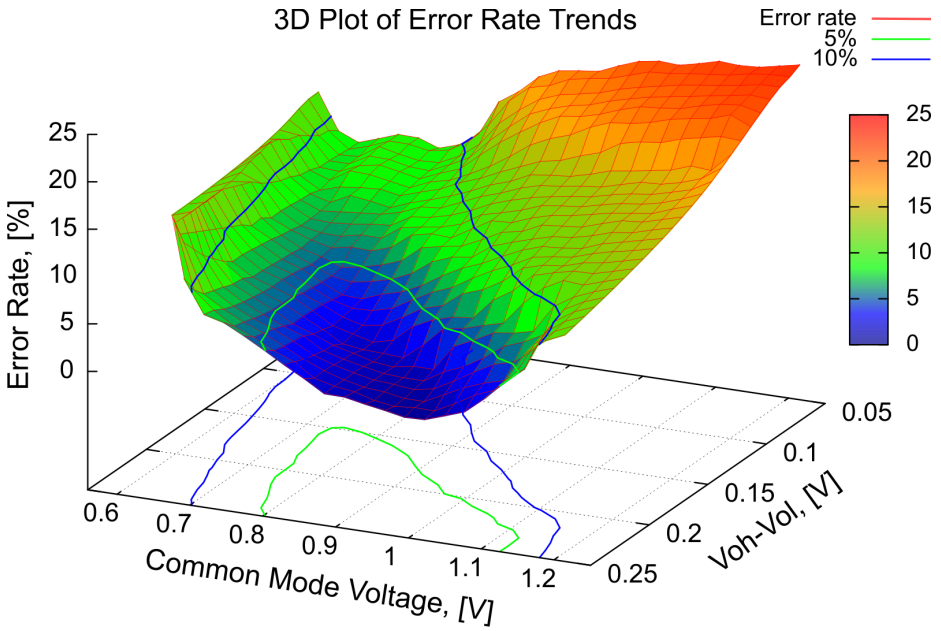


Figure 4.19: L1 voltage calibration: 3D plot, incorporating both V_{CM} and ΔV variation.

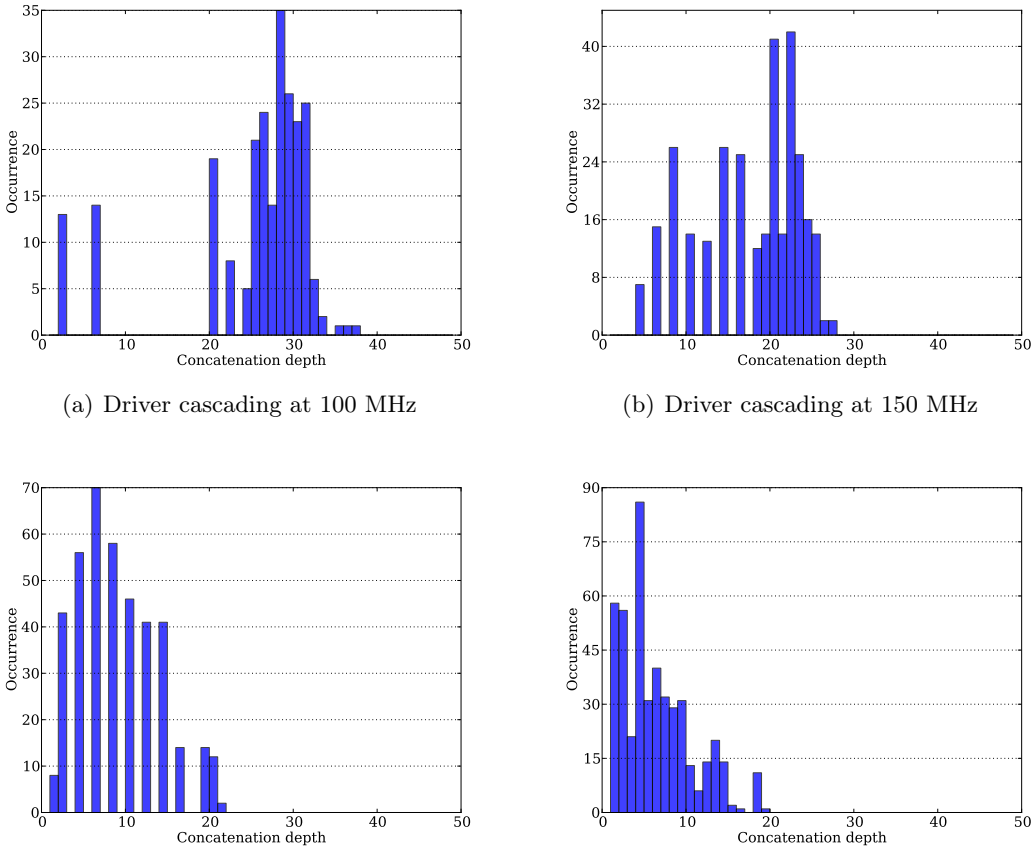
A 3D plot of the described error rate behavior was created for better illustration. It can be observed in fig. 4.19. The borders of 10- and 5-percent error rate areas are drawn to highlight the range with the best results.

The reason why this measurement has to be repeated on an actual system involving far more repeaters, is seen in fig. 4.18(a). The error rates here have two lows, while the expected statistical distribution of the optimum points would lead to a single low point. This indicates that two of the involved repeaters (or two groups of repeaters) had vastly deviating optimum points, (which can be explained, considering the mismatch during production). Due to low total number of tested repeaters, the gathered statistics were insufficient to produce a reliable distribution of expected form. Conducting a measurement with more repeaters would improve the results and help in finding the average optimum point for all repeaters.

4.2.2 Synapse Driver Cascading

This section introduces results of the test, described in (3.3.2). Subject to the measurements was the maximum length of the synapse driver cascade, which was measured on both chip versions and in dependence on the L1 communication frequency. The quality of L1 signal is frequency-dependent, and since synapse drivers contain L1 receivers, identical to those of the repeaters, maximum cascade length should shrink with growing frequency. According to considerations, made during design phase of the system, a length of 4 concatenated drivers is required for normal operation [35].

Fig. 4.20 shows measurement results of the HICANN v1 test at L1 frequencies from 100 to 250 MHz (the 250 MHz measurement was conducted merely to be able to estimate the tendency, maximum planned operation frequency persists at 200 MHz). Considering that productional driver-to-driver mismatch is of statistical nature, one can await the cascade length to be normally distributed, which would result in a Gaussian function. In the first approximation, this assumption is confirmed by the measurements as seen in fig. 4.20(a) and 4.20(b). At higher frequencies the resulting cascade length shrinks so far that the Gaussian form becomes distorted because of non-existent negative lengths.



(a) Driver cascading at 100 MHz

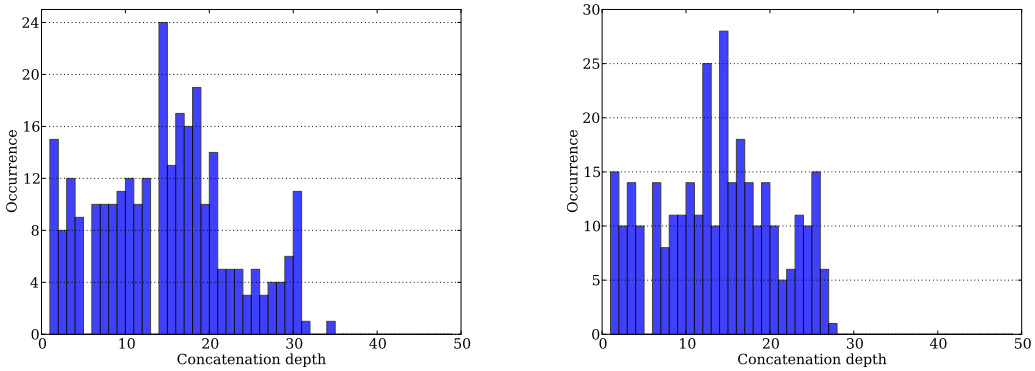
(b) Driver cascading at 150 MHz

(c) Driver cascading at 200 MHz

(d) Driver cascading at 250 MHz

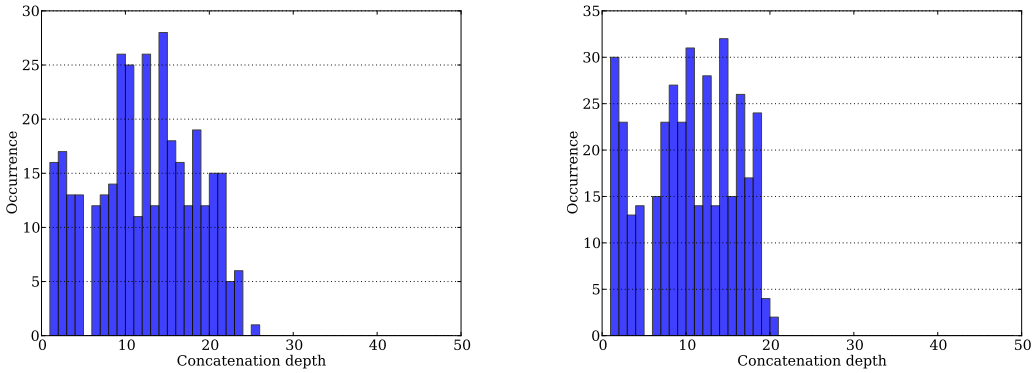
Figure 4.20: Distribution of synapse driver cascade length on HICANN v1. Minimum cascade length deceeds the desired value of 4 up to the frequency of 150 MHz.

All in all the results of testing at 100 MHz and 150 MHz are satisfactory, as the amount of cascades, failed to reach the length of 4, lies below 3%, which can be tolerated. Theoretically, all possible driver cascades can be tested and a list of those causing problems can be generated to coordinate the mapping process in such a way that it would not use the faulty ones.



(a) Driver cascading at 100 MHz

(b) Driver cascading at 150 MHz



(c) Driver cascading at 200 MHz

(d) Driver cascading at 250 MHz

Figure 4.21: Distribution of synapse driver cascade length on HICANN v2.

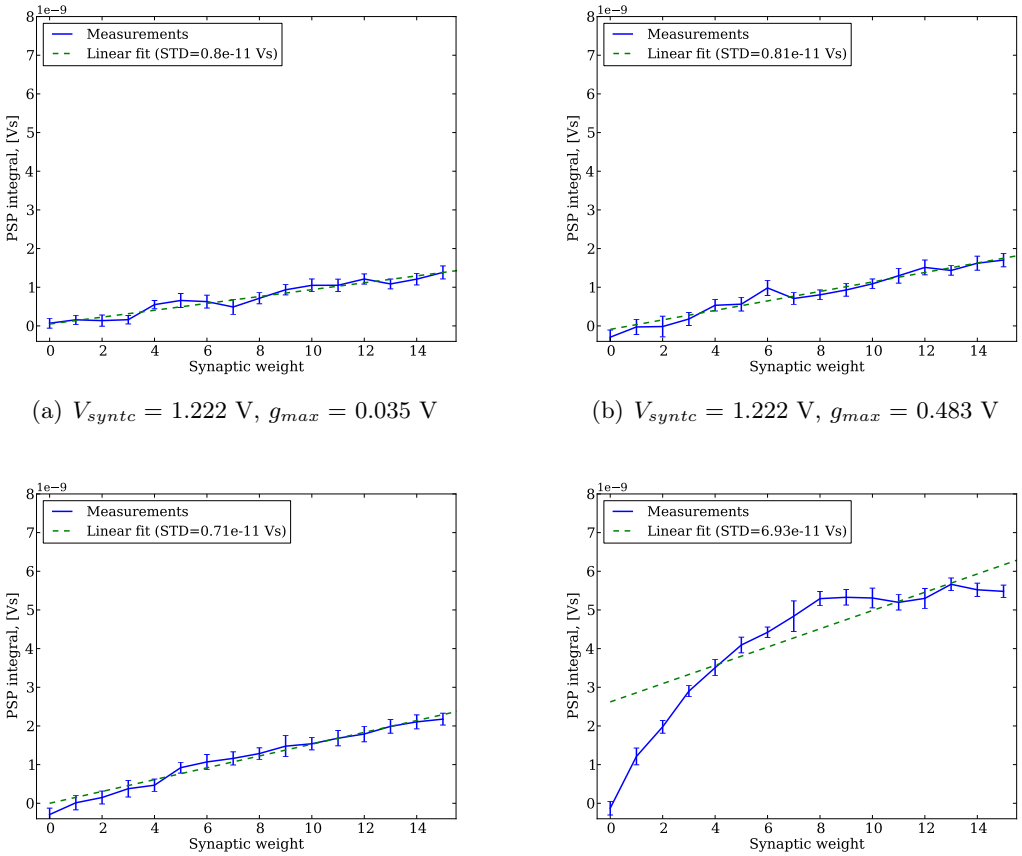
At higher frequencies, the ratio of cascades with length under 4 exceeds 10%, which indicates that operation in these modes is likely to produce errors without improvements in the design of L1 components of the chip.

Results of an analogous test on HICANN v2 are plotted in fig. 4.21. The distributions are much wider, than in HICANN v1 case, and even in 100 MHz mode, the amount of short cascades is quite large, which would make it difficult to operate the final system (assuming that a particular experiment needs many drivers to be concatenated). However, one can also notice that, while the results at lower frequencies are better on HICANN v1, those at higher frequencies are much better on the second version. In fact, the distribution mean on HICANN v2 does not change substantially, only the width changes with frequency.

The reasons for the worsening of HICANN v2 results as compared to HICANN v1 are yet unknown to the date of editing of this thesis and are being investigated.

4.2.3 Linearity of Synaptic Weights

Presented below are the results of measurements, specified in section (3.3.3). Subject of testing was the linear behavior of the synaptic weights and its dependence on neuron parameters.



(a) $V_{syntc} = 1.222$ V, $g_{max} = 0.035$ V

(b) $V_{syntc} = 1.222$ V, $g_{max} = 0.483$ V

(c) $V_{syntc} = 1.327$ V, $g_{max} = 0.035$ V

(d) $V_{syntc} = 1.433$ V, $g_{max} = 0.378$ V

Figure 4.22: Measurements of PSP-areas, plotted against the synaptic weights (solid blue). Error bars denote standard deviation over 10 repetitions. Linear fits (dashed green) serve as reference of linear behavior.

As it is notable from fig. 4.22, the overall linearity of the synaptic weights has been confirmed. However, as already mentioned before, the linear behavior depends on neuron parameters V_{syntc} and g_{max}. Varying of those parameters can lead to the saturation of ion-channel circuits, which means that they physically cannot conduct the demanded amount of current onto the membrane and go into saturation, distorting the linear behavior. This situation is depicted in fig. 4.22(d), where both V_{syntc} and g_{max} are set to very high values.

Both parameters increase the resulting current through ion-channel circuits, and, as

a consequence, the PSP area grows. Fig. 4.22(a) and 4.22(b) show a curve slope advance while increasing g_{max}, comparison of fig. 4.22(a) with fig. 4.22(c) depicts analogous growth while varying V_{syntc}.

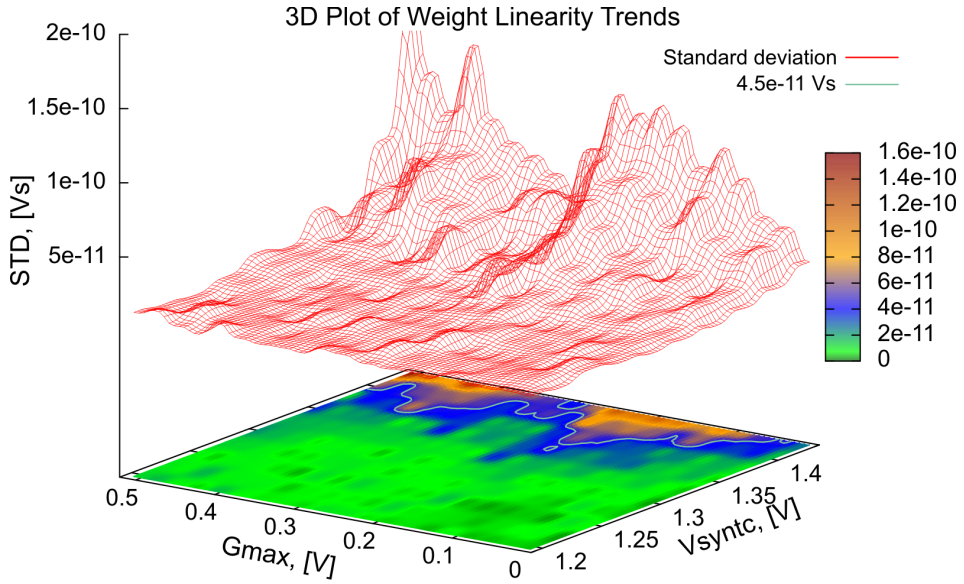


Figure 4.23: 3D plot of linear fit STD, indicating the linearity degree of the measured curves. The linearity depends on floating gate parameters V_{syntc} and g_{max}. The contour line drawn at 45 pVs separates the area with an acceptable linear behavior (green) from the area where resulting curves show saturation effects.

To be able to estimate the range with linear response, both parameters have been sweeped, and a curve has been taken for every sweep point. A linear function has been fitted onto the measured curve using scientific python library scipy [36], and the standard deviation of the fit has been calculated. The STD serves here as an indicator of how linear the curve is (compare e.g. 4.22(a) and 4.22(d)). Judging from many curves, a deviation of 45 pVs has been chosen as the limit of acceptable linear behavior. Fig. 4.23 shows a 3D-plot of the STD against both sweeped parameters. From this plot one can see, that the border between linear and non-linear behavior lies at approx. $V_{syntc} = 1.4\,\mathrm{V}$, which was predicted during design phase [34].

4.2.4 Fixed-Pattern Noise in Synapses

This section contains results, obtained from measurements, described in section (3.3.4). The goal was to quantify the variation of the synapse strength σ_s.

Unfortunately, the measurements have shown that the neuron calibration has to be perfected before the test can deliver proper results. The neuron-to-neuron error σ_n turned out to be enormous because the calibration routine did not yet include the calibration of the synaptic inputs of the neuron. This made calculation of σ_s impossible. However, some preliminary results could be obtained by excluding the σ_n consideration. Each column of

synapses has been evaluated separately and a value of $\sigma_c = \sqrt{\sigma_d^2 + \sigma_s^2}$ for each of them has been calculated. It includes both errors originating from the driver mismatch and from the synapse mismatch. Measurements over 27 synapse columns yielded a provisional result, quantifying said deviation to $\sigma_c = \mathbf{13}\pm\mathbf{11}\,\%$.

This is already a very reasonable error, which can be tolerated during final system operation. Eliminating σ_d in future measurements with calibrated synaptic ion-channel circuits will make the resulting error even smaller, which will let the modelers, who develop experiments for the hardware, construct more precise models and experiments.

5 Conclusion and Outlook

This thesis described testing of the HICANN chip, regarding its compliance with the specifications and evaluating accuracy of many of chips components. In the following the results are shortly summarized and the outlook on improvement options is given.

5.1 Extension of the Test Software

The new control modules and testmodes for the tests2 software, created while working on this thesis, contain many new functions, designed to operate various HICANN chip components and also the whole prototype system. New testmodes allow users to perform comprehensive tests on the hardware in order to observe and quantify many processes, running on the chip, such as the functioning and output of the repeaters, synapses or floating gate memory, but also the entire HICANN as neuromorphic ASIC can be operated.

The new modules can and should be used as reference for further development of the software, as they contain both very simple operations, e.g. switching of the L1 crossbars, and more complicated functions, such as stimulating neurons and reading out the spike traces from the trace FIFO.

Looking forward, one can say that the software has to be further extended by creating more complex functions that can e.g. configure desired amount of neurons to be interconnected and the output of resulting network to be evaluated. Also, the JTAG communication should be replaced by the faster Ethernet to ensure feasible experiment times.

5.2 Testing of the Floating Gate Memory

During this diploma project, the floating gate memory functionality has been thoroughly tested in order to confirm its operational readiness, possibly reveal weaknesses and consider their elimination or reducing their influence on the operation accuracy. The voltage floating gate cells have been found compliant with the specifications and can be utilized during final system operation as they are. The current cells' inaccuracy has been measured to be slightly higher than expected. The cause of this behavior has been identified as an unexpected leak tunnel current. Corresponding circuitry has been improved by Sebastian Millner and will be included in the next version of the HICANN chip.

Also, an optimization of the parameters for balanced programming precision and duration has been performed and a scheme of reducing the crosstalk has been found. While precision is a major issue, the programming time is considered to be less important, given

that the expected network mapping duration will consume several orders of magnitude more time than the floating gate memory programming. Nevertheless, the tests have shown that it is possible to find controller parameters with both reasonable speed and programming range/precision and it is always possible to trade in the speed to have better accuracy.

Extensive testing under realistic conditions have shown that the floating gate memory is able to perform error-free with high control gate voltage set to 11 V. However, it has been found out that the continuous operation produces a wear-out effect that is changing the cells' properties. A periodic recalibration of the neurons every 50,000 write cycles is therefore advised.

In general, the floating gate cells have been found to be a robust and compact analog memory type with a reliable storage time of approx. 1 h. However, they require strict usage conditions. The effects, such as crosstalk and voltage drift, are conditioning this type of memory to have complicated control structures and the storage time is very limited. With possible upgrade of the production process to 65 nm, perhaps it should be considered to use 8-bit digital memory cells and DACs to store neuron parameters because they are easier in usage and do not have a storage time limitation.

5.3 Testing of Other Circuitry

The last part of the thesis was dedicated to the testing of other critical circuits that condition successful wafer-scale system operation. Partly, the results of these tests are preliminary and corresponding measurements have to be repeated using working wafer or improved neuron calibration.

The linear behavior of the synaptic weights has been proven and the linearity range, depending on neuron parameters, has been found satisfactory. Nevertheless, the neuron circuits are being improved by the developers to yield more accuracy and better calibration possibilities for the final system.

The routine for calibration of L1 operation voltages has been successfully tested and the first results have been obtained. The error rate behavior shows expected trends. To achieve better statistical results, the execution of the test routine on a wafer, using more repeaters and total events, is advised. Also, communication over Ethernet should be used, as it was the major bottleneck for this test.

Tests of the synapse drivers have shown that existing circuitry is likely to require improvements, as the lengths of the driver cascades on HICANN v2 chips were not satisfactory. The reason could be the change in power routing as compared to HICANN v1 chips. Exact cause of the performance reduction from the first version of the chip to the second are to be investigated.

Preliminary results of the synaptic strength variation have been obtained. They confirm that the synaptic circuits possess enough accuracy to be used in final neuromorphic hardware. Final results of this test are expected upon the successful calibration of synaptic neuron inputs shortly after submitting of this thesis.

Bibliography

[1] Heinrich Wilhelm Gottfried von Waldeyer-Hartz. Ueber einige neuere forschungen im gebiete der anatomie des centralnervensystems. *Deutsche medicinische Wochenschrift*, 17, 1891.

[2] K. O. Stanley and R. Miikkulainen. Efficient reinforcement learning through evolving neural network topologies. In W.B. Langdon et al., editors, *Proceedings of the Genetic and Evolutionary Computation Conference GECCO 2002*, pages 569–577. Morgan Kaufmann Publishers, July 2002.

[3] Alan Lloyd Hodgkin and Andrew F. Huxley. A quantitative description of membrane current and its application to conduction and excitation in nerve. *J Physiol*, 117(4):500–544, August 1952.

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

[5] R. Brette, M. Rudolph, T. Carnevale, M. Hines, D. Beeman, J. M. Bower, M. Diesmann, A. Morrison, P. H. Goodman, F. C. Harris Jr, M. Zirpe, T. Natschlager, D. Pecevski, B. Ermentrout, M. Djurfeldt, A. Lansner, O. Rochel, T. Vieville, E. Muller, A. P. Davison, S. El Boustani, and A. Destexhe. Simulation of networks of spiking neurons: A review of tools and strategies. *Journal of Computational Neuroscience*, 3(23):349–98, December 2006.

[6] Marc-Oliver Gewaltig and Markus Diesmann. Nest (neural simulation tool). *Scholarpedia*, 2(4):1430, 2007.

[7] M.L. Hines and N.T. Carnevale. *The NEURON simulation environment.*, pages 769–773. M.A. Arbib, 2003.

[8] NeuroTools. Website. `http://neuralensemble.org/trac/NeuroTools`, 2008.

[9] BrainScaleS. Research. `http://brainscales.kip.uni-heidelberg.de/public/index.html`, 2011.

[10] J. Schemmel, J. Fieres, and K. Meier. Wafer-scale integration of analog neural networks. In *Proceedings of the 2008 International Joint Conference on Neural Networks (IJCNN)*, 2008.

[11] J. Fieres, J. Schemmel, and K. Meier. Realizing biological spiking network models in a configurable wafer-scale hardware system. In *Proceedings of the 2008 International Joint Conference on Neural Networks (IJCNN)*, 2008.

[12] J. Schemmel, D. Brüderle, A. Grübl, M. Hock, K. Meier, and S. Millner. A wafer-scale neuromorphic hardware system for large-scale neural modeling. In *Proceedings of the 2010 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1947–1950, 2010.

[13] Johannes Schemmel, Andreas Grübl, and Sebastian Millner. Specification of the HICANN microchip. FACETS project internal documentation, 2010.

[14] Dan Husmann and Holger Zoglauer. A wafer-scale-integration system(WSI). FACETS project internal documentation, 2010.

[15] TU Dresden. DNC specification. FACETS project internal documentation, 2008.

[16] M. Ehrlich, K. Wendt, L. Zühl, R. Schüffny, D. Brüderle, E. Müller, and B. Vogginger. A software framework for mapping neural networks to a wafer-scale neuromorphic hardware system. In *Proceedings of the Artificial Neural Networks and Intelligent Information Processing Conference (ANNIIP) 2010*, pages 43–52, 2010.

[17] Sebastian Millner, Andreas Grübl, Karlheinz Meier, Johannes Schemmel, and Marc-Olivier Schwartz. A VLSI implementation of the adaptive exponential integrate-and-fire neuron model. In J. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R.S. Zemel, and A. Culotta, editors, *Advances in Neural Information Processing Systems 23*, pages 1642–1650. Curran Associates, Inc, 2010.

[18] T.S. Lande, H. Ranjbar, M. Ismail, and Y. Berg. An analog floating-gate memory in a standard digital technology. In *Microelectronics for Neural Networks, 1996., Proceedings of Fifth International Conference on*, pages 271–276, 12-14 1996.

[19] J. Schemmel, K. Meier, and E. Muller. 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)*, pages 1711–1716. IEEE Press, 2004.

[20] J. Schemmel, A. Grübl, K. Meier, and E. Muller. Implementing synaptic plasticity in a VLSI spiking neural network model. In *Proceedings of the 2006 International Joint Conference on Neural Networks (IJCNN)*. IEEE Press, 2006.

[21] TU Dresden. HICANN - Digital Interface Specification. FACETS project internal documentation, 2007.

[22] Electronic Vision(s) – Website. Website. `http://www.kip.uni-heidelberg.de/vision`, 2011.

[23] Wulfram Gerstner and Werner Kistler. *Spiking Neuron Models: Single Neurons, Populations, Plasticity*. Cambridge University Press, 2002.

[24] Richard Naud, Nicolas Marcille, Claudia Clopath, and Wulfram Gerstner. Firing patterns in the adaptive exponential integrate-and-fire model. *Biological Cybernetics*, 99(4):335–347, Nov 2008.

[25] André Srowig, Jan-Peter Loock, Karlheinz Meier, Johannes Schemmel, Holger Eisenreich, Georg Ellguth, and René Schüffny. Analog floating gate memory in a 0.18 μm single-poly CMOS process. *FACETS internal documentation*, 2007.

[26] ANSI/TIA/EIA-644. *Electrical Characteristics of Low Voltage Differential Signalling (LVDS)*, March 1996.

[27] LeCroy. X-stream oscilloscopes - remote control manual. Technical Report Revision D, LeCroy Corporation, 700 Chestnut Ridge Road, Chestnut Ridge, NY 10977-6499, 2005.

[28] Xilinx, Inc., www.xilinx.com. *Virtex-5 FPGA User Guide*, 2009.

[29] ISO/IEC 14882. *Programming Language C++*, July 1998.

[30] John D. Hunter. Matplotlib: A 2D graphics environment. *IEEE Computing in Science and Engineering*, 9(3):90–95, 2007.

[31] Python. The Python Programming Language – website. `http://www.python.org`, 2009.

[32] Analog Devices. *8-Channel, 10- and 12-Bit ADCs with I^2C-Compatible Interface in 20-Lead TSSOP*, 2004. AD7997/AD7998.

[33] National Semiconductor. LVDS owner's manual. LVDS.national.com, 2004.

[34] Sebastian Millner. personal communication, 2011.

[35] Johannes Schemmel. personal communication, 2011.

[36] SciPy. Website. `http://www.scipy.org/`.

Acknowledgements

I would like to express my appreciation to all the people who supported me during this year, in particular:

Prof. Dr. Karlheinz Meier for providing me with the ability to work in such great atmosphere.

Prof. Dr. Ulrich Brüning for the second opinion on this thesis.

Dr. Johannes Schemmel for being a great leader.

Sebastian Millner for supervision and proofreading.

Thomas Pfeil for smuggling me into the group and proofreading.

The rest of the Visions for being a cool bunch.

Alexandra, Svea, Florian, Adrian, Sebastian, David, Hendrik, Thomas, Oliver, Friedrich for being great classmates and friends.

Frank, Alexander, Michael, Sascha, Marco and others, you know who you are... NFL!

Dawid, Lucas, Melanie, Jeffrey, Fynn, and others, cheers!

All my family abroad for being patient when I hesitate to get in touch.

My parents for supporting me throughout all these years in good and bad times.

Meine Oma dafür dass sie die beste Oma in der Welt ist.

Katja, my very special companion and friend, for being who you are ♡.

Sorry if I forgot someone...

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, October 10, 2011

.....
(signature)