Department of Physics and Astronomy
University of Heidelberg

Master thesis in Physics
submitted by
Jakob Jens Kaiser
born in Waldshut
2020

Implementation of Large Scale Neural Networks on the Neuromorphic BrainScaleS-1 System

This Master thesis has been carried out by
Jakob Kaiser
at the
Kirchhoff-Institute for Physics
under the supervision of
Dr. Johannes Schemmel

Abstract Neuromorphic hardware has made great advances in recent years; consistently implementing a larger number of neurons and synapses. With the increasing complexity of such systems the demands on the accompanying software increase as well. This thesis contributes to the operating system of the wafer-scale BrainScaleS-1 neuromorphic system with the aim to facilitate the implementation of large scale neural networks.

We use a synchronous firing chain to demonstrate the challenges which arise during the implementation of neural networks on neuromorphic devices and present the emulation of a functional chain with 190 links and 19 000 neurons. More than 70 000 analog neuron circuits and 1.4 million synapses are involved in modeling the dynamics of this chain.

A high number of repeaters is needed to ensure the accurate transmission of spikes across the wafer. These repeaters recover a timing reference from incoming signals. Not all repeaters derive the correct timing in a single try. By repeatedly putting these repeaters in reset, the total number of repeaters which recover the correct timing can be increased. By implementing such a routine in the operating system, we paved the way for the successful emulation of large scale neural networks.

Zusammenfassung Eine immer größere Anzahl an Neuronen und Synapsen kann durch neuromorpher Hardware implementiert werden. Um diese zunehmend komplexen Systeme effektiv betreiben zu können, muss die dazugehörige Software stetig weiterentwickelt werden. Diese Arbeit widmet sich der Weiterentwicklung des Betreibssystems des BrainScaleS-1 wafer-scale Systems mit dem Ziel, die Implementation großer neuronaler Netzwerke zu ermöglichen.

Mithilfe einer Synfire Chain untersuchen wir, welche Herausforderungen die Implementation von neuronalen Netzwerken auf neuromorpher Hardware birgt. Weiterhin präsentieren wir die Emulation einer Synfire Chain mit 190 Gliedern und 19 000 Neuronen. Mehr als 70 000 analoge Neuronen-Schaltkreise und über 1,4 Millionen Synapsen werden benötigt, um diese Kette erfolgreich zu emulieren.

Eine große Anzahl an Repeatern stellt sicher, dass Spikes zuverlässig auf dem Wafer weitergeleitet werde. Die Repeater erhalten kein globales Taktsignal, sondern leiten eine Periode aus den empfangenen Signalen ab. Nicht alle Repeater finden die richtige Periode im ersten Versuch. Indem diese Repeater wiederholt zurückgesetzt werden, kann die Anzahl an Repeatern mit einer korrekten Perioden erhöht werden. Dieses wiederholte Zurücksetzen wurde in das Betriebssystem integriert und ebnet damit den Weg für die erfolgreiche Emulation großer neuronaler Netzwerke.

Contents

1 Motivation

Playing music, drawing pictures, solving puzzles, building cars: the human brain is capable of performing a variety of tasks. But how does the brain work? How is information represented, processed and stored? In an effort to answer these questions the anatomical and electrical properties of the brain have been studied for decades. This lead to the development of a broad range of different neuron, synapse and network models. These models range from complex, biologically realistic models such as the Hodgkin-Huxley model to more simplified models which are easier to analyze analytically.

Due to its size and complexity, he extent to which the brain is accessible by electrophysiological methods is limited. It is difficult to study the interactions between a high number of neurons directly in biological tissue. As a result, software simulations have been becoming increasingly more important to study the dynamics of larger networks. They allow to study almost arbitrary neuron and synapse models as well as network topologies. This flexibility comes with the cost of a high energy consumption and long simulation times.

Dedicated integrated circuits which mimic the dynamics of neurons and synapses, so called neuromorphic hardware, can lead to a speed up of emulations while being more energy efficient at the same time. Such neuromorphic circuits have been developed in Heidelberg for almost two decades: starting from single chips which host hundreds of neurons to the Brain ScaleS 1 (BSS-1) system which is able to implement more than a hundred thousand neurons on a single wafer.

In order to emulate neural networks on neuromorphic systems, dedicated software is needed which translates the biological description to a hardware representation and configures the hardware accordingly. With an increasing size of networks the mapping and setup becomes increasingly more difficult. Advances in neuromorphic hardware have always go hand in hand with improvements of the accompanying software.

This thesis uses a benchmark network, a synchronous firing chain, to illustrate the challenges which arise during the implementation of large scale neural networks on the BSS-1 system. Changes to the BSS-1 operating system aim to make the hardware operation faster and more robust, facilitating the emulation of large scale neural networks.

2 Introduction

This thesis contributes to the operating system and commissioning efforts of the BSS-1 system to enable the emulation of large scale neural networks. The improvements are demonstrated on the example of a synchronous firing chain which is easily scalable to form large networks.

Section 2.1 provides a general overview over the BSS-1 neuromorphic system and describes for this thesis relevant parts in more detail.

The last section quickly introduces the used neuron model and the structure as well as dynamics of our benchmark model.

2.1 The BrainScaleS-1 System

BSS-1 is a mixed signal neuromorphic system. While analog circuits are used to emulate biological neurons and synapses, digital logic is responsible for spike transportation.

In the first two sections we give a broad overview of the system before covering the for this thesis relevant components in the subsequent sections.

2.1.1 The HICANN Chip

The High-Input Count Analog Neuronal Network Chip (HICANN) chip is the fundamental building block of the system, Fig. 2.1. The two synapse arrays of the symmetric chip occupy the largest area. A single array implements $220 \times 256 \approx 100\,000$ synapses.

256 dendrite membrane (DenMem) circuits are located next to each array towards the center of the chip. These circuits implement the AdEx neuron model, c.f. Section 2.2.1. The intrinsic time constants of these circuits lead to a speed up factor of 10 000 compared to biological real time [1, 2]. Parameters of each neuron are configurable and stored on floating gate cells on the chip [1, 3].

Each DenMem receives inputs from one column of synapses of the corresponding synapse array. 32 circuits on the top and 32 circuits on the bottom are combined to form a neuron block. Multiple of two DenMems in a neuron block can be connected to form a single neuron such that the total number of possible synapses increases to a maximum of 14 080 per neuron [3, 2].

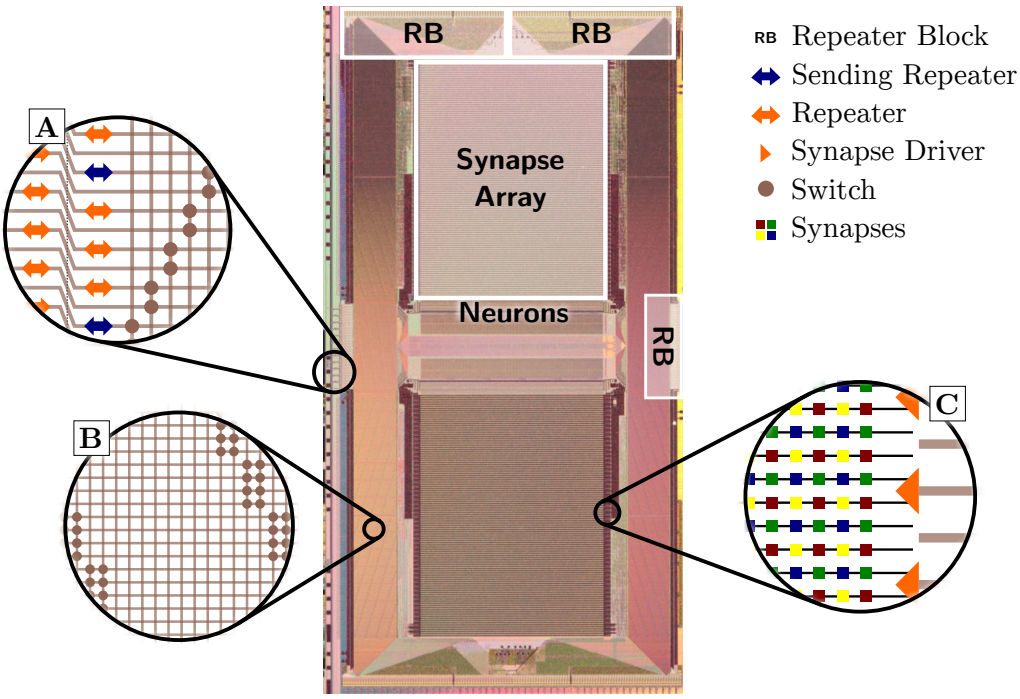


Figure 2.1: HICANN Chip – As the chip is symmetric we only mark the repeater blocks, synapse array and neurons on the top (right) part. **(A)** Repeaters are located at the edges of the HICANN. They occupy every second on-waffer communication lane on a single chip such that one repeater is on each line at the boundary of two HICANNs. Sending repeaters (blue) are used to inject events from the neurons and external events in the L1 network. To assure that not all sending repeater inject in the same horizontal lines, they are shifted by two at the edges of the chip. In the right a part of one of the two sparse crossbars is visible which are used to connect horizontal and vertical lines. **(B)** Sparse synapse switches allow to relay signals from vertical lines to synapse drivers. **(C)** One synapse driver is responsible for two lines of synapses. The synapse driver decodes the two most significant bits of the 6-bit address and determines which of the four sets of synapses (mark by color) receive the incoming signal. Synapse rows which are not connected to drivers in this picture are controlled by drivers on the other side of the synapse array.

Each HICANN contributes 256 vertical and 64 horizontal lines to the on wafer communication network, visible as H in Fig. 2.1 [3]. Sparse connection matrices at the center of the chip allow to connect specific vertical and horizontal lines, cf. Fig. 2.1 (A). At both sides of each synapse array sparse synapse switches are used to connect vertical lines to synapse drivers, sub figure (B).

Spikes from the DenMem circuits as well as external spikes are injected via eight specialized sending repeaters on each chip, sub figure (A). This is done in form of a six bit neuron address, compare Section 2.1.3.

To connect two neurons on the chip the switch matrices have to be set in the according way. This allows to connect the output of one sending repeater to one synapse driver. Using the two most significant bits of the address the driver decides which set of synapses receives the input signal, compare Fig. 2.1 (C). Each synapse stores a 4-bit address as well as a 4-bit weight. The 4-bit address is compared with the remaining 4 bits of the signal and a pulse modulated by the weight is forwarded to the neuron which lies at the bottom of the column. With this space/address coding scheme arbitrary neurons on the wafer can be connected.

2.1.2 Wafer Scale

One of the distinguishing features of BSS-1 is the wafer scale integration: wafers are not cut into single chips after production but stay intact. As a result chips can be directly connected on the wafer. This allows for high connection densities and an energy efficient, low latency communication of spike events [4, 5].

Due to the high resolution needed during lithography, only a restricted area of about $2 \times 2\,\mathrm{cm}$ – a reticle – can be processed at once during wafer production [4]. While the eight chips on a single reticle are therefore connected, the connections between different reticles have to be established in a post-processing step [1]. A single wafer is made up of 48 reticles and therefore of 384 chips [2].

The wafer is connected a printed circuit boards which provides connections to other boards such as field programmable gate arrays (FPGAs) boards (for communication between host and HICANN) or power boards.

2.1.3 Repeater

Several repeater ensure that signals stay accurate enough to travel long distances on the wafer. Repeaters are placed on alternating lines at the top/bottom (left/right) of the HICANN, Fig. 2.1 (A). Each repeater belongs to one of six repeater blocks. Repeaters in a single block share some common settings/registers such as the delay-locked loop (DLL)-reset or test ports.

Providing each repeater with a clock signal would be space and energy consuming [6]. Therefore, DLLs are used to recover a timing reference from received signals. A repeater

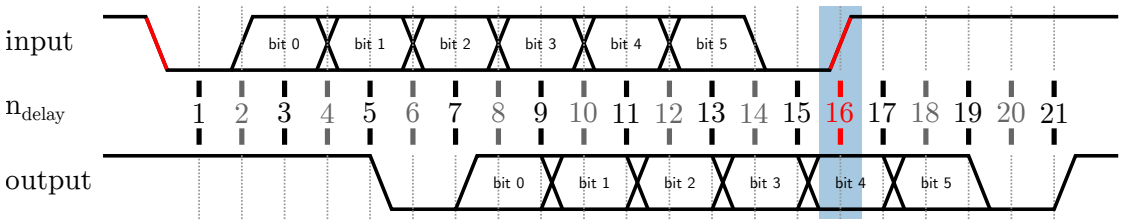


Figure 2.2: Function Principal of Repeaters – The repeater receives a serialized six bit address embedded in two framing bits. The entire frame is split into 16 equally sized bins. When receiving the falling edge of the start bit (red) the repeater sends a signal along an internal delay line; each element delaying its input by the same time Δt. Depending on the position of the signal in the delay line different actions are triggered. At the third delay the value of the incoming signal is captured in parallel memory, every second next bit is used to capture another value. Starting from delay five the output line is driven; starting with the start bit. Every second delay one of the previously captured bits is forwarded followed by a stop bit after the 21^{st} delay. For a small region around the 16^{th} delay (blue region) an edge detection mechanism is active and tries to capture the rising edge of the stop bit (red). This edge is aligned with the delay line signal after 16 delays to update the duration of a single delay Δt.

is called "locked" if it deduced the correct timing from the incoming signals.

Hock [6] and Schemmel et al. [4] describe in detail how locking of repeaters work. Each repeater deserializes the six bit address, stores it and serializes it again. Receiver, parallel memory and driver each perform one of these tasks.

Each six bit neuron address is embedded in framing bits, cf. Fig. 2.2. The falling edge of the start bit and the rising edge of the stop bit are used to determine the duration of a 8-bit (start/stop bit and six address bits) which is then represented as a voltage V_{CTRL} stored on a capacitor.

Locking

The receiver detects the falling edge of the start bit and sends a signal along an internal delay line; each element in the delay line delays its input by the same time Δt which is controlled by the voltage V_{CTRL}. The aim is now to align the signal of the rising edge of the start bit after the 16^{th} delay with the fall falling edge of the stop bit; thus dividing the input signal in 16 equally sized bins (Fig. 2.2). In order to prevent locking to rising edges of address bits the input signal is masked. Only a small region (one timing bin) around the 16^{th} delay is visible to edge detecting mechanism.

To establish an initial lock zeros are sent to the repeater and the capacitor is pre-charged with a voltage V_{dllres}. V_{dllres} has to be chosen such that the unmasked area lies near the expected rising edge of the stop bit; this prevents locking to multiple of the frame length. Zeros are sent such that no rising and falling edges except for the ones of the stop and stop bit are present in the received signal.

After the initial lock was successful addresses can be relayed via the L1 network. The

repeaters are now able to use these signals to adjust the voltage on the capacitor and therefore compensate leak currents or temperature deviations.

Synapse drivers have receiving circuits which are identical to the ones in the repeaters and therefore locking works in the same way.

Signal Forwarding

In order to store and forward an incoming signal the delay line is used to trigger the capture of the input signal: after the 3^{rd} delay address bit 0 is captured, after the 5^{th} address bit 1 and so on. All bits of the address are stored in a parallel memory. At the 5^{th} delay the driver starts reading this memory and forwards the address in a serialized form.

Repeating is expected to delay the signal by about 2.3 ns at each repeater [4].

Test Ports

Each repeater has a test data output on which it can send decoded parallel addresses to the associated control logic. All repeaters in one repeater block are connected to two data outputs in an alternating fashion such that the events of two repeaters can be read put in parallel. After enabling the reading of test data in the control logic the addresses as well as the arrival times (in number of clock cycles) of the first three received events are stored in the memory of the control logic.

By sending a known sequence of addresses with a known delay between them this allows to test whether the repeater locked correctly. Background generators on the HICANN allow to send address zero events at fixed time intervals.

2.1.4 Synapse Controller

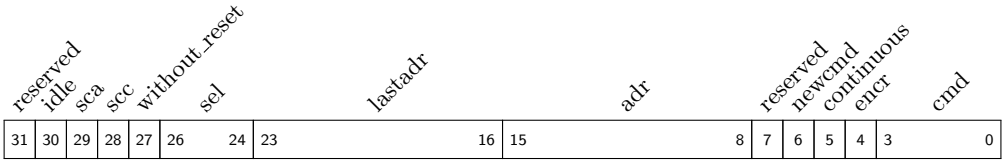
In hardware a controller is responsible for a specific part of the hardware; it provides input and output registers for communication with other parts of the hardware. The synapse controller is a finite state machine which is responsible for a single synapses array on the HICANN chip: it takes care of weights, decoder addresses and locking of synapse drivers.

The responsibility for weights also includes the evaluation of the correlation circuits and automatic weight updates which can be used to implement spike time dependent plasticity (STDP) [7]. Since this feature is currently not implemented in the software stack of BSS-1, we will ignore all functionality related to correlation measurements.

Registers relevant for this thesis are the control register (*CREG*), configuration register (*CFGREG*), status register (*STATUS*) as well as the transfer registers *SYNIN* and *SYNOUT* which are used as a temporary storage of weights/decoder addresses. While the *SYNIN* and *SYNOUT* are made up of four 32-bit registers, which are each divided in eight 4-bit words, the other registers are single 32-bit registers with special meaning for each subgroup of bits.

We display the control register as an example, Register 2.1. Information for the other registers can be found in *Neuromorphic Platform Specification* [8] and the thesis of Friedmann [7].

Register 2.2: Control Register of the Synapse Controller – *encr*, *continuous*, *lastadr*, *without_reset*, *scc* and *sca* are used to control cross-correlation evaluation and STDP-functionality [7, 9]. *cmd* stores the command the controller is supposed to perform; the different commands can be found in *Neuromorphic Platform Specification* [8]. The synapse controller only performs the command specified in *cmd* if the *newcmd* bit is set to true. *adr* determines on which synapse row the operation should be performed. The column set is selected with the field *sel*, see text for more information. *idle* is set if the controller does not perform a command and idles.



Reading of Weights

This section presents a common operation of the synapse controller: reading weights. A more detailed version can be found in Friedmann [7]. It is used as an example to illustrate the purpose of the different registers and the operations of the synapse controller.

Each synapse array is horizontally split into four blocks, i.e. synapse columns 0 to 63 belong to group 0, 128 to 191 to group 1 and so on. The blocks are further divided in eight column sets each such that a single column set contains eight synapses per block and 32 synapses in total, compare Fig. A.1. In order to save bit lines not all weights in a single row are read at once. The read operation is performed for one column set after another; the 4-bit weights of eight synapses in the first column set are stored in the first 32-bit of *SYNOUT*, the weights of the second column set in the second 32-bit register and so on.

Algorithm 2.1 outlines a possible host side implementation for reading an entire row of weights. Fig. 2.3 illustrates what happens internally in the synapse controller during these operations. Operational parameters such as the different waiting times during stages can be set in the configuration register.

For a read of the SRAM cells in the synapse array the bit lines have to be driven with the value stored in the cells. This process is initiated by the `START_READ` command, Line 3. First the controller enters the state *READ_WEIGHTS_PRE*; it pre-charges the bit lines and waits for c_{predel} cycles. In the next state, *READ_WEIGHTS_EN*, pre-charging is disabled and the wordline of the desired row (set in Line 2) is activated. After waiting c_{endel} cycles the state *ROW_SELECTED_WEIGHT* is entered: the bit lines should now have reached a stable value such that the input to the multiplexer

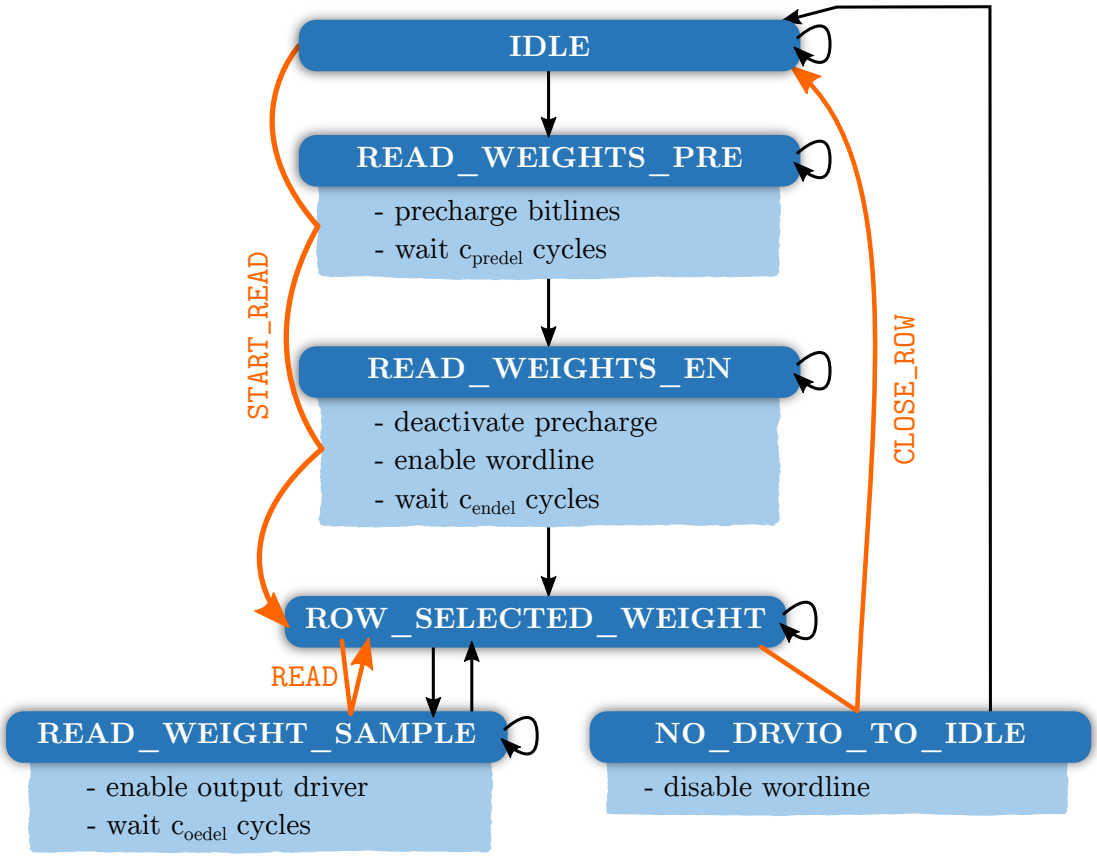


Figure 2.3: Reading Weights – States of the synapse controller which are related to reading weights. Each state is represented by a box. The header gives the name of the state and the body lists actions which are taken in the state. Black arrows indicate transition between states. Orange arrows show which states are entered if a specific command is given to the controller. For detailed explanation see text.

which reduces the $4 \cdot 256 = 1024$ bits to the size of the *SYNOUT* register (128-bit) is stable, compare Fig. A.1. The settings of the multiplexer are determined by the column set stored in the field *sel* of the control register, compare Register 2.1. While the controller performs the tasks initiated by `START_READ` the host computer has to wait; this is done by reading the status register and waiting till the busy bit is unset, Line 5.

Now the registers *SYNOUT* have to be set according to these values; this is done with the command `READ` (Line 11). The synapse controller enters the state *READ_WEIGHT_SAMPLE*: the multiplexer is configured, the output driver is enabled and control is remained for c_{oedel} cycles before transitioning back to *ROW_SELECTED_WEIGHT*. The read weights are now available in the *SYNOUT* register.

In order to read the weights of an entire synapse row these tasks are repeated for each column set (Line 9) and the contents of *SYNOUT* are stored in an output data structure (Line 17). After each command, which is sent to the synapse controller, the host software has to wait until the controller finished the task.

Finally, we can close the row (Line 18): the synapse controller enters the state *NO_DRVIO_TO_IDLE* where the wordline is disabled and then transitions back to *IDLE*.

2.1.5 Off Wafer Communication

Communication of configuration and spiking data between a computing host and the chip is mediated by FPGAs. This results in two communication streams: host ↔ FPGA and HICANN ↔ FPGA. Each FPGAs is connected via a $1\,\text{Gbit}\,\text{s}^{-1}$ link to the host computer, each HICANN uses a $2\,\text{Gbit}\,\text{s}^{-1}$ link to connect with the FPGA [10].

Spike data from the host is stored in the playback memory of the FPGA and released during the experiment; the timing is specified by a time stamp. Spikes from the HICANN are stored in memory and can be read by the host after the experiment is completed. During configuration of the HICANN configuration data sent from the host are immediately forwarded to the chip.

The input rate of external spike events is limited by the speed of the FPGA and the bandwidth between FPGA and HICANN. External spikes can be grouped in so called "pulse groups"; each group can be made up of up to 184 pulses. To start the processing of a pulse group the FPGA needs six clock cycles. Every additional pulse can be processed in one cycle.

The FPGA organizes the pulses from the playback memory in FIFOs of size 16. From there up to two spikes are combined in a packet and sent to the HICANN. The HICANN immediately forwards the pulses in the on-wafer network. According to Klähn [11] the link between FPGA and chip is limited to $17.8\,\text{MEv}\,\text{s}^{-1}$. Dividing this rate by the speed up factor of 10^4 yields about two events every millisecond. For higher input rates, spikes are shifted or even lost if the FIFOs are full.

For the links between FPGA and host/HICANN automatic repeat request (ARQ)

Algorithm 2.1: Possible host side implementation for reading weights of an entire synapse row. For a detailed explanation see text.

```
Data: Hardware address of synapse row row, vector w to fill with weights from
      hardware
Result: Updated vector w
ControlRegister creg;
// Open row
creg.adr ← row;
creg.cmd ← START_READ;
creg.newcmd ← true;
write_to_hw(creg) // Write content to hardware
// Wait till row is opened
repeat
    busy ← status_register.slice_busy
until busy == false;
// Read weights from different column sets
foreach sel in column sets do
    creg.sel ← sel;
    creg.cmd ← READ;
    creg.newcmd ← true;
    write_to_hw(creg)
    // Wait until weights are read to SYNOUT
    repeat
        busy ← status_register.slice_busy
    until busy == false;
    // Store weights in output structure
    save_weights_colset(w, synout_register, sel);
// Close synapse row after reading
creg.cmd ← CLOSE_ROW;
creg.newcmd ← true;
write_to_hw(creg) // Wait until the row is closed
repeat
    busy ← status_register.slice_busy
until busy == false;
```

(HostARQ and HicannARQ) mechanisms are used to ensure reliable communication[1], for details see Müeller [10]. The expected round-trip time on the HostARQ depends on the traffic and is expected to be around 100 µs if the queue is empty and in the range of 100 ms if the queue is full. For the HicannARQ the round-trip time is expected to be around 1 µs in case of low traffic and several tenth of microseconds if the queue is full[2].

2.1.6 Software

Dedicated software had to be developed in order to operate the BSS-1 system. In this section we give a quick summary of some of the layers of the BSS-1 software stack. For details see Müller et al. [12].

Halco

As seen in previous sections the BSS-1 system is made up of many components. In order to configure each component it is important that the programmer can specify each component uniquely. For this purpose a coordinate system, called halco, was introduced.

Halco provides a coordinate for each building block of the system, e.g. wafer, repeater or synapse array. It further provides relations between these components such that appropriate coordinates can be transformed in each other.

Halbe

The halbe is responsible for low level hardware access. It tries to provide dedicated data structures, called container, for configuration data of each component of the system. Reading and writing from/to the hardware is also handled by halbe.

Sthal

The sthal uses halbe containers to represent the current state of the hardware. Furthermore, the read/write functions of halbe are used to configure the system. For this purpose sthal implements different configuration routines called "configurators". Configurators take care that all relevant values are set in the correct order.

Marocco

Marocco takes the high level description of PyNN and translates it to a valid hardware configuration. It places the neurons on different neuron circuits on the wafer and realizes connections between them by setting the synapse switches accordingly. In addition, it

[1] For the HICANN ↔ FPGA connection only configuration data are sent via the ARQ stream.
[2] Estimations by E. Müller.

translates parameters such as reset potential or synaptic time constants to hardware values.

PyNN

BSS-1 provides a PyNN API such that neural networks can be described in a high level abstraction in the Python scripting language. Allowing for high level description of neuronal experiments which can be executed on BSS-1 aims to enable non-expert users to use the system without a profound knowledge of the underlying hardware and software.

Besides BSS-1 PyNN is supported by a number of different simulation backends such as NEST[3], Brian[4] and NEURON[5] [13].

2.2 Neuroscience

The brain is made up of a high number of small processing units, neurons, which interact in a complex manner. One aim of neuroscience is to find adequate models which are able to describe the behavior of the neurons but are at the same time simple enough to simulate and analyze. In the first section the popular leaky integrate and fire (LIF) neuron model is introduced.

In the second part the network topology of a synfire chain is presented. This structure was first introduced by Abeles [14] to explain repeating spatiotemporal spiking patterns. As synfire chains are easily scalable to form large neural networks and their dynamics are robust against parameter variations, the render ideal benchmark models for neuromorphic hardware.

2.2.1 Neuron and Synapse Model

BSS-1 implements the dynamics of the adaptive exponential integrate and fire (AdEx) neuron model which is able to mimic the spiking behavior of a broad range of neuron types [15, 16]. The AdEx model can be simplified to a simple LIF neuron which we will use in this thesis.

It models the neuron as a membrane with a capacitance C_m which is connected to a leak potential E_L. Incoming spikes from other neurons cause a current flow I_syn on the membrane. The evaluation of the membrane potential V_m is given by:

$$C_\mathrm{m}\frac{\mathrm{d}V_\mathrm{m}}{\mathrm{d}t} = -g_L\left(V_\mathrm{m} - E_L\right) + I_\mathrm{syn}, \tag{2.1}$$

[3] https://nest-simulator.readthedocs.io
[4] https://briansimulator.org/
[5] https://neuron.yale.edu/neuron/

where g_L is the leak conductance [17].

Whenever the membrane potential crosses a threshold V_{thres} from below a spike is emitted and the membrane potential is set to a reset value V_{reset}. Afterwards the temporal evaluation of the membrane potential is again governed by Eq. (2.1).

We use conductance based synapses to describe the current I_{syn} which is caused by incoming spikes of other neurons:

$$I_{\text{syn}} = -\sum_{\text{synapses } i} g_i(t)\left(V_{\text{m}} - E_i\right), \tag{2.2}$$

with synaptic conductance $g_i(t)$ and synaptic reversal potential E_i.

The dynamics of the synaptic conductance can be modeled as an exponential decay with a synaptic time constant τ_i and a base conductance g_i^0:

$$g_i(t) = \sum_{t_0 \in T_i} g_i^0 e^{-\frac{t}{\tau_i}} \Theta(t - t_0). \tag{2.3}$$

Here T_i represents the times of all spikes which are transmitted by the synapse i. The base conductance g_i^0 is used to encode the strength of the synaptic connection and is also called synaptic weight.

2.2.2 Synchronous Firing Chain

Synchronous firing chains (short synfire chains) were introduced to explain spatiotemporal firing patterns. In its original form a synfire chain consists of several chains/groups of neurons which are connected in an converging-diverging manner, Fig. 2.4 (A). A synchronous activity in a group causes a synchronous response in the following group; allowing to transmit synchronous signals along the chain. Depending on the synchronicity of the initial input the signal may be transmitted along the chain or die out after a few links, Fig. 2.4.

As the signal is propagated by groups of neurons, the network is robust to the defect of single neurons and synapse loss. This renders it a suitable network for hardware emulations where parameters may vary from neuron to neuron and synapse loss may be present due to the restricted number of connections between parts on the hardware.

In order to classify which input leads to a propagation, "pulse packets" can be used to describe the temporal spread and strength of the input [19, 18]. A pulse packet of strength a and temporal spread σ (short (a,σ)) is a group of a spikes with spike times drawn from a Gaussian distribution with standard deviation σ, Fig. 2.5.

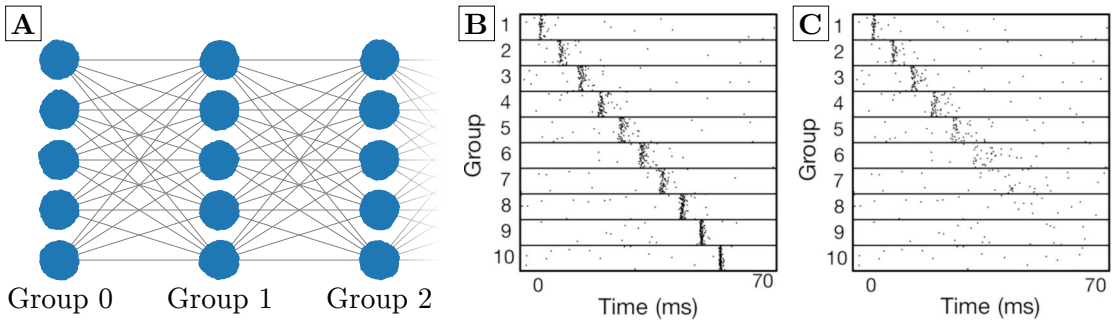


Figure 2.4: Connectivity and Pulse Propagation in a SynFire Chain – **(A)** In a synfire chain neurons in different groups are connected in a converging-diverging scheme: each neuron connects to a high number of neurons in the following group and receives inputs from a high number of neurons in the preceding group. Here groups of five neurons with all to all connectivity are shown. **(B,C)** The propagation of a pulse depends on its synchronicity and strength. The spiking activity of ten groups which each consists of 100 neurons is shown. Each neuron in the fist group receives a pulse packet input (a,σ). **(B)** For a sufficiently strong input (50,0) to each neuron the pulse packet propagates along the chain. **(C)** If the input strength is lightly lower (48,0) the packet may survive the first few groups but finally dies out before reaching the final group. **(B,C)** Taken from Diesmann et al. [18] (Fig. 3).

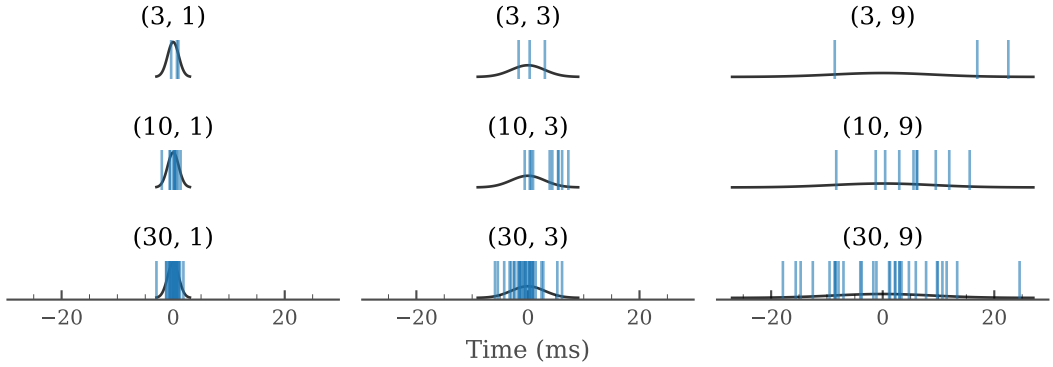


Figure 2.5: Gaussian Pulse Packets – Example for different pulse packets (a,σ) drawn from a random distribution. The spikes are presented as vertical lines. In the background is the Gaussian distribution from which the spike times are drawn (lines extend to 3σ).

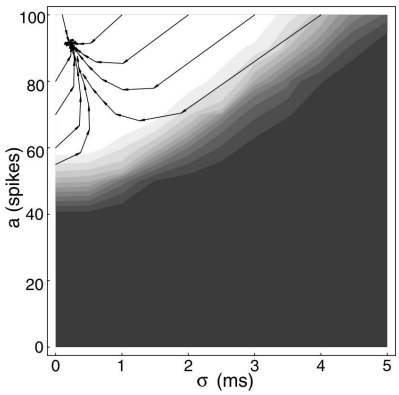


Figure 2.6: Phase Space of a SynFire Chain (without feed forward inhibition) – The contour lines mark the probability of the initial pulse packet (a,σ) reaching the final group. High propagation probabilities are colored light while the regions of unsuccessful propagation are black. The success depends on both strength and synchronicity of the initial input. In the region of successful propagation all trajectories converge in a common attractor. Some of these trajectories are displayed in the upper left corner of the plot. Taken from Gewaltig et al. [20] (Fig. 7).

Simple Feed Forward Structure

Diesmann et al. [18] started by investigating the response of a single LIF neuron to pulse packet input. These studies were later extended by Gewaltig et al. [20] who investigated the dynamics of an entire synfire chain.

The network structure is similar to the one illustrated in Fig. 2.7 but without inhibitory populations in each group. Each neuron received independent background input from inhibitory and excitatory Poisson spike sources which mimic the activity in a cortical network. All neurons in the first group received a pulse packet (a,σ) as stimulus input. The authors performed several trials to determine the networks response to the different input stimuli.

Propagation success depends on both strength and temporal spread of the input, Fig. 2.6. A higher strength can compensate a broader temporal distribution and vice versa.

A critical strength of about 50 spikes are at least needed to transmit even the most synchronous pulse packets, compare Fig. 2.4. The minimum amount of spikes depends on the neuron parameters and the topology of the network [18].

In the region of successful propagation all trajectories converge in a single attractor. Therefore, we will also call this region the "basin of attraction". The attractor is characterized by a high output strength and a low temporal spread of $\sigma < 1\,\text{ms}$. We have to note that the simulations did not include variations in neuron parameters and synaptic delay. Therefore, the temporal spread is expected to be higher in real networks. Fur-

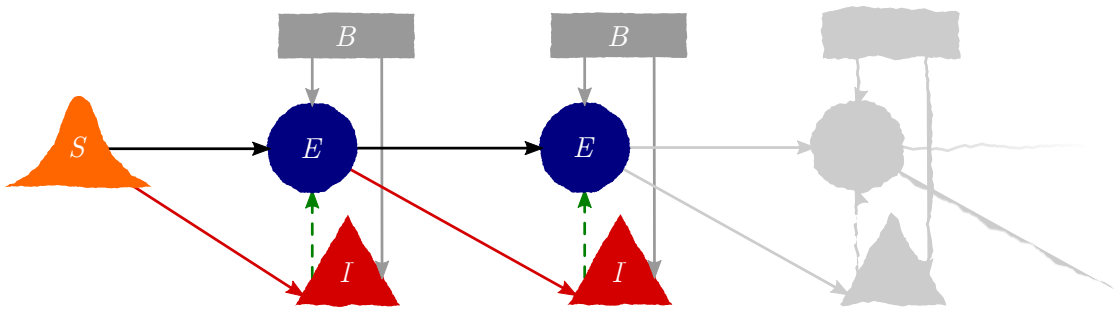


Figure 2.7: Synfire Chain with Feed Forward Inhibition – A synfire chain consists of a stimulus population (S) and several chain links (also called groups). Each group is made up of an excitatory (E), inhibitory (I) and background (B) population. Each stimulus neuron injects spikes in form of a Gaussian pulse packet, cf. Fig. 2.5. Neurons in the excitatory (E_i) and inhibitory (I_i) population connect to a fixed number of neurons in the preceding excitatory group (E_{i-1}) or in case of the first group to the stimulus. Inhibitory neurons only project in the excitatory population within the same group. The background input is shared between inhibitory and excitatory neurons in a group. Without inhibitory neurons this network reduces to a simple synfire chain without FFI.

thermore, the attractor will no longer be represented by a sharp point in the phase space but by a blurred out one.

After the pulse reached the attractor the dynamics of the network are quite robust. Small perturbations may shift the system in a nearby state. As long as this state is in the basin of attraction the network will move back towards the attractor.

The convergence speed towards the attractor depends on the initial starting point in the phase space. While states starting near the separatrix need around 10 to 15 chain links to reach the attractor, states in the core of the basin of attraction reach the attractor after about five chain links [20].

As mentioned above a lower synchronicity of the input can be compensated by a higher input strength. This makes the simple synfire chain susceptible to asynchronous activity: random background activity may initiate a pulse packet. In order to increase the filtering for synchronous input feed forward inhibition (FFI) can be introduced.

Feed Forward Inhibition

A simple synfire chain transmits a large range of input stimuli and therefore may also be activated by random background activity. In order to reduce the integration window of the neurons inhibition can be incorporated into the network [21, 22]. The idea is that inhibition follows shortly after excitation. This limits the integration time of the neuron to the range of the delay between excitation and inhibition; making it harder for asynchronous inputs to travel along the chain [22].

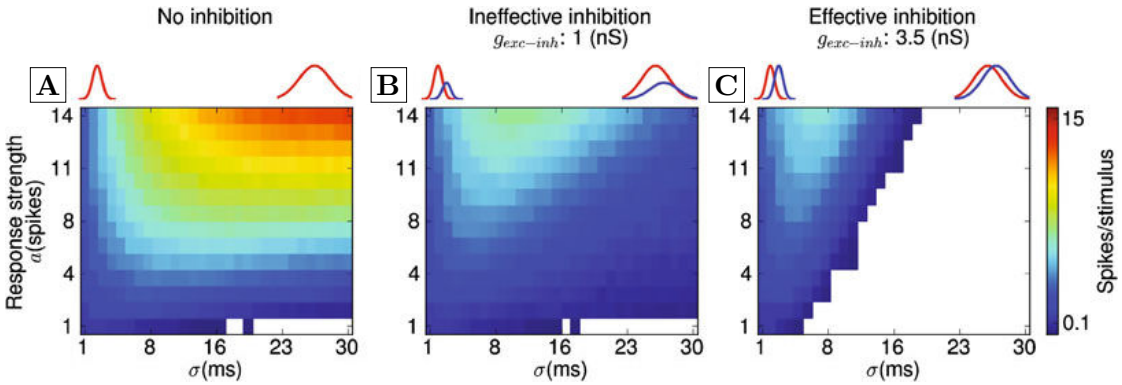


Figure 2.8: Phase Diagram of a Synfire Chain with Feed Forward Inhibition – Output strength versus input strength and standard deviation. Signals with $a_{\text{out}} < 0.1$ are not displayed. Note that the input is given as the spiking behavior of a single neuron in the stimulus population and not as the input neurons in the first population receive. Each neuron in the first population has 60 incoming stimulus connections. **(A)** Without inhibition propagation is possible over a large range of the phase space. For strong and asynchronous inputs each neuron in the last group spikes more than 10 times. **(B)** Introducing inhibition can effectively reduce the output strength in the last group. The filtering properties of the network are not improved. **(C)** For effective inhibition the network was less susceptible to asynchronous inputs. Picture taken from Kremkow et al. [22] (Fig. 2).

Feed forward inhibition can be implemented by introducing an additional inhibitory population in each group, Fig. 2.7. This population receives the excitatory input of the previous population, such that it is activated at approximately the same time as the excitatory population. It then connects to the excitatory population of the same group via inhibitory synapses.

Figure 2.8 illustrates the effect of inhibition. For the simple feed forward structure the network transmits a large range of input stimuli. Only the weakest packets are not propagated if the temporal spread is too high. In all other cases the strength is sufficient to allow for transportation of asynchronous packets. The activity in the final group is made up of several pulse packets if the input pulse is sufficiently strong and not too synchronous: neurons are reactivated several times after their refractory period.

In case of ineffective inhibition the filtering for synchronous input is not improved, Fig. 2.8. Strong but asynchronous input stimuli are still propagated along the chain. However, the output strength in the final group is reduced such that for a large part of the phase space only a single packet is forwarded to the last group.

Increasing the synaptic strength between excitatory and inhibitory populations allows for a more reliable activation of the neurons in the inhibitory population. As a consequence of this increased activity they are able to effectively forward spikes to the excitatory population in the same group. This increased inhibition leads to an improve-

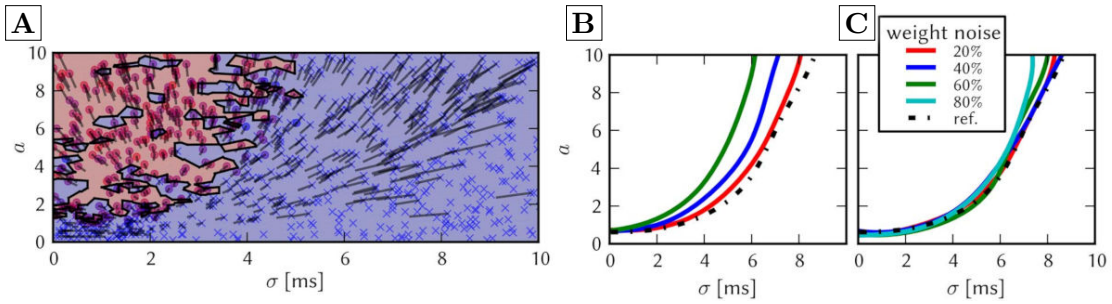


Figure 2.9: Influence of Weight Noise on a Synfire Chain – Each weight w was drawn from a Gaussian distribution with mean w and standard deviation $w \cdot p$ where p is the percentage of noise. **(A)** Without compensating for the weight noise the phase diagram is patchy. The separation between successful propagation and failure is not sharp anymore. **(B)** Fitting curves to the separation line indicates that the region of successful propagation decreases with increasing noise. For $p = 80\,\%$ the functionality breaks down completely. **(C)** Compensating the weight noise (see text) can almost restore the separation line. Only small differences for large input strengths can be observed. Taken from Petrovici et al. [24] (Fig. 17).

ment of the filtering property of the network: the input has now to be much more synchronous to elicit pulse packets in the final group.

Simulation with the Executable System Specification

Synfire chains have already been implemented on Spikey, a predecessor of the HICANN chip, and simulated with a executable system specification (ESS) of the BSS-1 system [23, 24]. The ESS takes the digital communication infrastructure as well as the analog circuits into account and replicates their dynamics in software. Variations of neuron parameters, synapse weights or the fixed on-wafer communication bandwidth can for example be simulated with the help of the ESS. In addition, the ESS allows to monitor state variables which are not accessible during the operation of the BSS-1 system This makes it possible to analyze distortions induced by the hardware in more detail.

Petrovici et al. [24] simulated a synfire chain with the help of the ESS. They analyzed the influence of weight noise, synapse loss and fixed synaptic delays on the network dynamics.

Figure 2.9 shows the effect of synaptic weight noise on the network dynamics. The region of successful propagation becomes more patchy and no sharp separatrix can be distinguished, sub figure (A). Increasing the weight noise reduces the basin of attraction (B). For a weight noise of 80 % successful propagation is no longer possible.

The authors provided background inputs via a single synapse to each neuron in the network. All other connections consist of at least 25 synapses such that the weight variations are not as significant as for the background input. While it is possible to

use more synapses to provide random background activity, the authors chose to reduce the weight of the single synapse and increase the resting potential of all neurons in the network. As a result it was possible to maintain the mean membrane potential of the original network and compensate weight noise of up to 80 %, Fig. 2.9 (C).

As the network is locally feed forward the synapse loss is low if the hardware resources are not restricted by blacklisting and becomes only relevant for large networks. Synapse loss of up to 50 % can be compensated by increasing the synaptic weight of the remaining synapses.

Kremkow et al. [22] used the synaptic delay between the inhibitory and excitatory population of the same group to adjust the position of the separatrix. On BSS-1 the synaptic delay can not be adjusted and is in the order of 1 ms to 2 ms. Petrovici et al. [24] showed that a delay of 1.5 ms is still sufficient to allow for pulse propagation. Furthermore, they altered the synaptic strength $g_{I_i \to E_i}$ between inhibitory and excitatory population to control the size of the basin of attraction.

They also observed that dense pulses are broadened due to the limited bandwidth of the on-wafer communication network. This broadening had no visible effect of the position of the separatrix.

3 Software Improvements

In order to operate the BSS-1 system not only have the hardware components to work together but also dedicated software is needed which allows to easily configure the hardware as desired. As part of this thesis some parts of the BSS-1 operating system were improved in terms of stability, performance and speed.

At first the software representation of a specific hardware component was improved. This helped to clarify the routine which is responsible for the locking of repeater and synapse driver, also see Section 2.1.3. It also lays the basis for introducing a repeated locking of repeaters.

Not all repeaters can be locked successfully within one try. By putting unlocked repeaters in reset and retrying the locking can effectively increase the number of locked repeater, Section 3.2.

Finally, the access speed for synaptic weights and decoder addresses was decreased, Section 3.3.

3.1 Synapse Controller Class

Several changes to the software stack aim to improve the representation of the hardware implementation in software while also providing additional convenience of high level functions. The changes described in this section build up on the work by S. Schmitt and concern the representation of the synapse controller, Section 2.1.4.

Apart from introducing specific data containers to represent hardware parts, some code restructuring was done in order to clarify the locking of repeaters and synapse drivers. Furthermore, a synchronization of command buffers after each configuration stage was introduced.

Finally, tests which ensure the functionality of the software are presented.

3.1.1 Previous State of Software

The content of the control and other registers was put together manually, see Listing 3.1, and afterwards written to hardware.

Listing 3.1: Prior to the introduction of a specialized data structure for the control register of the synapse controller the content of the register was put together manually. Example for a read operation on column set 6 in row 14.

```
1 // h is a HICANN handle
2 SynapseControl& sc = reticle.HICANN[h.jtag_addr()]->getSC(HicannCtrl::SYNAPSE_TOP);
3 uint32_t read_command = facets::SynapseControl::sc_cmd_read |
4                          (6 << facets::SynapseControl::sc_colset_p) |
5                          (1 << facets::SynapseControl::sc_newcmd_p) |
6                          (14 << facets::SynapseControl::sc_adr_p); // address in hw
                           ↪  coordinates
7 c.write_data(facets::SynapseControl::sc_ctrlreg, read_command); //issue read command
```

Pulling and releasing of the DLL-reset was done within a single halbe function for repeaters and synapse drivers respectively. `lock_repeater_and_synapse_driver()`, regardless of the name, was only responsible for locking repeaters. A command to pull the reset was immediately followed by the command to release the reset. Locking of synapse drivers was performed in the same function which also writes the general configuration of synapse drivers to hardware, namely `set_synapse_driver()`.

Sthal provided the two stages `LOCK_REPEATER` and `LOCK_SYNAPSEDRIVERS` were these functions were called.

3.1.2 Software Changes

This section displays some key features of the new software implementation and provides some examples of their usage.

Specialized Data Structures

By implementing dedicated data structures for configuration data it can be ensured that only valid values are written to hardware. In addition, it makes the adjustment of settings more convenient and easier to read/understand.

The control register for example is a 32-bit register which stores several settings at different bit positions, cf. Register 2.1. While the single bit values such *sca*, *scc*, *without_reset*, *newcmd*, *continuous* and *encr* are stored as Boolean member variables of the `SynapseControlRegister` class, the other bit values are represented by specialized data structures.

The column set *sel* for example is now represented by a ranged integer `SynapseSel`. This ensures that the member variable only holds values in the allowed range. Other examples are the member variables representing *lastadr* and *adr* which make use of halco coordinates such that software developers can use a single coordinate system in the entire

Listing 3.2: Setting the contents of the synapse control register after introduction of a dedicated data structure and a setter function. Compare with the old implementation Listing 3.1.

```
1 // syn_ctrl is of type SynapseControlRegister
2 syn_ctrl.cmd = SynapseControllerCmd::READ;
3 syn_ctrl.sel = SynapseSel(6);
4 syn_ctrl.newcmd = true;
5 syn_ctrl.row = SynapseRowOnArray(209);
6
7 // h is a HICANN handle
8 set_syn_ctrl(h, SynapseArrayOnHICANN(0), syn_ctrl);
```

software stack. The translation to hardware coordinates happens in the getter/setter functions, cf. Section 3.1.2.

These changes allow for a more readable and less error-prone definition of the contents of the control register. Another advantage is that members which are not actively set conserve their previous state. In Listing 3.2 this for example ensures that settings such as *sca* or *continuous* keep their intended values and are not zeroed out as in Listing 3.1.

Similar data structures were introduced for the others registers and settings related to the synapse controller.

Optional Values

Some bits are read-/write-only on hardware, e.g. the idle bit in the control register (Register 2.1) or the whole status register of the synapse controller. Variables representing these bits are of type `boost::optional<base_type>` in software. This has the advantage that containers which are used to write data to hardware and containers which are filled with values read from hardware are easily comparable.

In case of the *idle* bit the member variable in the `SynapseController` is of type `boost::optional<bool>`. This member variable is unset when initializing a new object of type `SynapseControlRegister`, lets call it `creg_write`. After writing `creg_write` to hardware and saving the read contents of the register in `creg_read` the *idle* member variable is set. The comparison of `SynapseControlRegister` instances is implemented such that the comparison of `creg_write` and `creg_read` evaluates to true. In case both *idle* variables are set a standard comparison in performed.

Setters and Getters

For each register getter and setter functions are provided to facilitate communication with the hardware. Inside these functions the custom data structures are transformed

to bit patterns which the hardware can understand or vice versa. It is also the place where the transformation between software and hardware addresses happens.

Furthermore, the programmer only has to provide a handle to a HICANN and the synapse array on which the corresponding register/controller is located. Therefore, a `SynapseControl` object does not have to be initialized manually and the user does not have to know the hardware address of the corresponding register, cf. Listing 3.1 and Listing 3.2.

Restructuring

As seen in Section 3.1.1 the locking of repeaters and synapse drivers was tightly integrated in halbe functions. This not only contradicts the encapsulation of responsibilities but also leads to a constant relocking of synapse drivers when `set_synapse_driver()` is called.

Consequently, the locking functionality was removed from the halbe functions and moved one layer up to sthal.

In order to have all repeaters and synapse drivers in reset at the beginning of the hardware configuration the corresponding data structures initialize DLL-reset to be set.

The `ParallelHICANNv4Configurator` provides several configuration stages in which the different reticles are configured in parallel. When writing the initial configuration to hardware, in the configuration stage `TIMING_UNCRITICAL`, all repeaters and drivers are in reset. In a following stage, `LOCKING_REPEATER_BLOCKS`, reset of the repeaters is released. After that we expect a reliable signal at the synapse drivers and they can leave the DLL-reset; this is done in `LOCKING_SYNAPSE_DRIVERS`.

The `ParallelHICANNv4SmartConfigurator` was also altered. Users can now choose actively how they want to handle repeater and synapse driver locking: locking can be skipped, enforced or done smartly. In the smart configuration the configurator decides, based on the previous hardware configuration, if locking is needed. The configurator decides as follows: if the hardware was not configured beforehand locking is always done; in case of changes to the L1 route relocking of repeaters is done. Synapse drivers are relocked if repeaters are relocked or if there have been changes to the synapse drivers themselves.

Synchronization of Command Buffers

While working on the restructuring of the locking scheme a synchronization of the command buffers after each configuration stage was introduced in the `ParallelHICANNv4-Configurator`. This ensures that all HICANNs receive all commands before the next stage is entered.

The communication path from the host to a single HICANN is split in two ARQ-streams. One from the host to a FPGA and one from the FPGA to the HICANN,

compare Section 2.1.5. If one wants to be certain that no commands are pending both streams have to be emptied.

For HicannARQ the function `flush_HICANN()` was already implemented in halbe. A read command is sent from the FPGA to the HICANN and the execution of code is stopped until the FPGA receives the result of the read command.

A similar function was implemented for the first half of the communication path: `flush(FPGA_handle_t& f)` checks if the host has sent all packages. If there are still remaining packages in the queue the functions waits 10 ms and then checks again for unsent packages. This is done until the queue is empty or a timeout is reached.

`sync_command_buffers(FPGA_handle_t const&, HICANN_handles_t const&)` in the `HICANNConfigurator` first calls the flush function of the stream from host to a FPGA and afterwards the flush function for the communication path between FPGA and each HICANN. This function is called after every configuration stage of the `ParallelHICANNv4Configurator`.

3.1.3 Tests

Digital Blacklisting Tests

Digital blacklisting tests are normally used to check the functionality of digital parts of the hardware. A random but legal configuration is written to hardware. Afterwards the values are read back and compared with the original values. If there is a mismatch between the written and read value the hardware component is assumed to be defect and is blacklisted such that it will not be used during operation of the system.

The digital blacklisting was extended for the synapse control register and the synapse configuration register, see Table A.5 By using the blacklisting test on registers which are expected to work correctly the implementation of setters and getters was tested.

The blacklisting test does not protect us from making a mistake in translating the software configuration to hardware and then doing the reverse mistake in the back translation. However, the software has to pass at least this test to be functional.

Running the test on a few selected registers on wafer 33 was successful. Since the tests did not throw any errors and the changed software was in usage for several month, we are confident that the implementation is correct.

Locking Test

In order to confirm that the locking is still working after restructuring a random network was created and tested with different versions of the software stack. For the creation of the network a set of HICANNs was selected and the following scheme was executed:

1. select random seed,

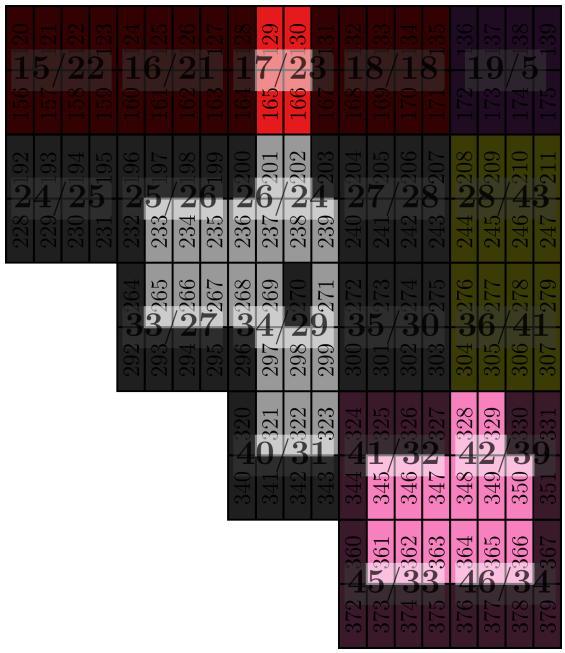


Figure 3.1: Locking Test - Used HICANNs on Wafer 33 – South west corner of wafer 33. HICANNs which were unused during the locking test on wafer 33 are blacked out. In total 39 HICANNs were selected for the test.

2. place a population of 120 neurons on a random HICANN (remove HICANN from list),

3. place a population of 120 neurons on a random HICANN (remove HICANN from list),

4. connect population to previous population, and

5. repeat Item 3 and Item 4 until all HICANNs are used.

Wafer 33 was used to run the test with five different seeds. Since the post processing of wafer 33 connects L1 lines at the boarder of the wafer to unused reticles, the outermost HICANNs were not selected. Taking blacklisting into account 39 HICANNs in the south-east corner of the wafer were selected, cf. Fig. 3.1. This resulted in networks with 1089 to 1348 repeaters.

In order to test the influence of the different changes the test was run for several versions of the software stack. The change set related to the synchronization of the command buffer was merged on November 12, 2019. Changes concerning the `SynapseController` class and restructuring were submitted on November 26, 2019. The versions

Table 3.1: Synapse Controller Class - Locking Test – Mean value and standard deviation of locked repeaters for locking test on wafer 33. The version numbers of the software stack are given in the header. In the last column the total number of repeaters in each network is displayed.

Seed	2019-11-11-1	2019-11-13-1	2019-11-25-1	2019-11-27-1	Number of Repeaters
	Unlocked Repeaters (%)				
6	0.5(2)	0.5(1)	0.5(2)	0.4(3)	1348
35	0.8(2)	0.9(3)	0.8(3)	0.3(2)	1089
42	0.4(2)	0.4(1)	0.3(1)	0.5(2)	1249
94	0.3(2)	0.3(1)	0.3(1)	0.9(6)	1103
128	0.7(3)	0.6(2)	0.6(2)	1.3(5)	1203

of the software stack were chosen such that there is always a test with a version before and after the changes were introduced.

After introducing synchronization of command buffers the mean locking rate remains basically the same for all seeds, cf. Table 3.1. This is also evident in Fig. 3.2. The median value as well as the 25 % and 75 % quantiles stay approximately the same.

In order to confirm that no changes to the software stack affected the locking success after the synchronization was introduced a test with an additional software version was done. The version from November 25, 2019 represents the software just before the merge of the `SynapseController` class. Table 3.1 and Fig. 3.2 verify that the locking success is the same for the two versions of the software.

After restructuring the `SynapseController` the distribution of unlocked repeaters extends to higher percentages and no network was locked completely in any run, Fig. 3.2. Table 3.1 reveals that this is mainly due to a drop in the locking success for the networks with seed 94 and 128. The percentages of unlocked repeaters for these two seeds and the last version of the software stack show a relatively high standard deviation to the other values. Nevertheless, the mean values are still within 1 to 2 standard deviations across the different versions of the stack. Since no systematic decrease for all seeds is observed the locking scheme is assumed to work correctly.

3.2 Relocking of Repeaters

As described in Section 2.1.3 several repeaters on the wafer ensure that spike signals can be transmitted from one HICANN to another. Just pulling and releasing the DLL-reset of these repeaters at the same time does not lead to completely locked routes if the network is complex. By allowing for a repeated reset of unlocked repeaters the total number of locked repeaters can be increased. Once again the changes presented in this section are based on work by S. Schmitt.

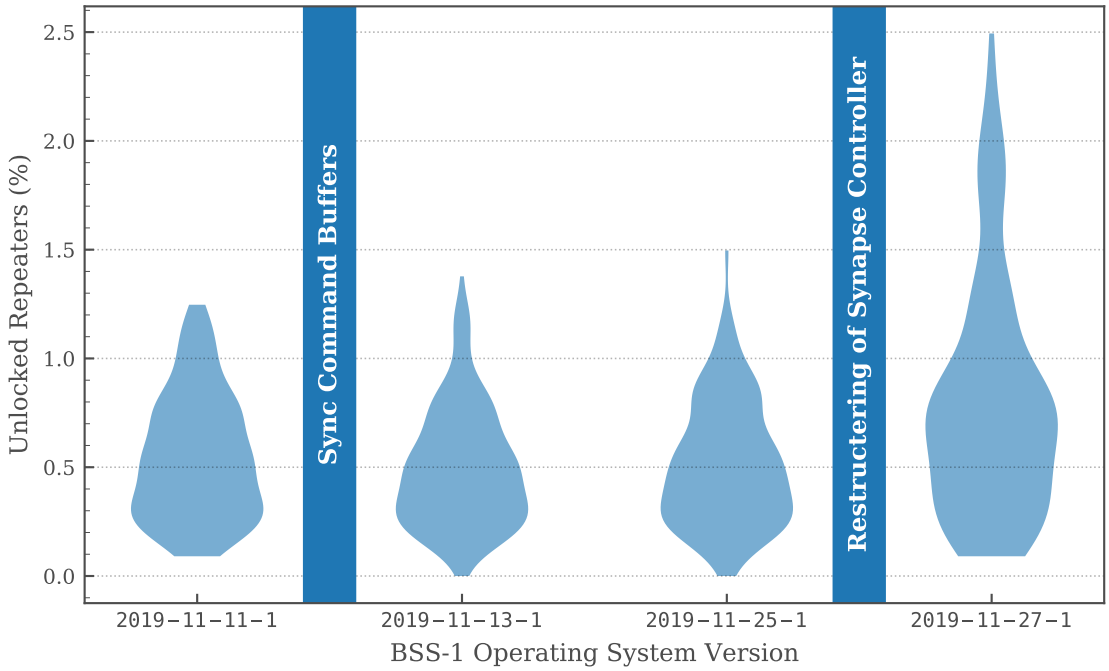


Figure 3.2: Synapse Controller Restructuring - Confirm Functionality of Repeater Locking – Locking success for different verisons of the BSS-1 software stack. The test was performed 50 times for each of the different seeds, compare Table 3.1.

3.2.1 Implementation

In an attempt to increase the total number of locked repeaters, the locking of repeaters which are still unlocked after each try is redone. Repeaters can not be put in reset individually since they are organized in blocks; DLL-reset can only be pulled/released for an entire repeater block. Therefore, the routine was implemented such that whole blocks get relocked if a single repeater failed the locking test:

1. lock of all repeaters,
2. test locking of all repeaters,
3. collect all repeater blocks with unlocked repeaters,
4. redo locking of collected repeater blocks,
5. repeat 2 to 4 a predefined number of times,
6. relock synapse drivers, and
7. return result of final locking test.

Marocco already provides a routine which checks whether all used repeaters are locked, `marocco::ReadRepeaterTestdata::check()`. This function makes use of the test ports to read out the received addresses and delays between them, see Section 2.1.3. We integrated the relocking algorithm into this function.

Specialized Configurator

While the isolated relocking of synapse drivers can be realized with the `Parallel-HICANNv4SmartConfigurator` a specialized configurator had to be introduced for the relocking of specific repeater blocks. The `ParallelOnlyRepeaterLockingNoResetNoFG-Configurator` takes a set of *RepeaterBlockOnWafer* coordinates and retries the locking for all repeaters in them.

The configurator implements the following tasks in the sthal configuration stages:

TIMING_UNCRITICAL Set selected repeater blocks into reset

LOCKING_REPEATER_BLOCKS Release reset of selected repeater blocks

LOCKING_SYNAPSE_DRIVERS Skip

NEURONS Skip

Functions concerning the configuration of FPGAs, e.g. `config_FPGA()` and `start_-systime_counter()`, are overwritten such that the FPGAs are not reconfigured.

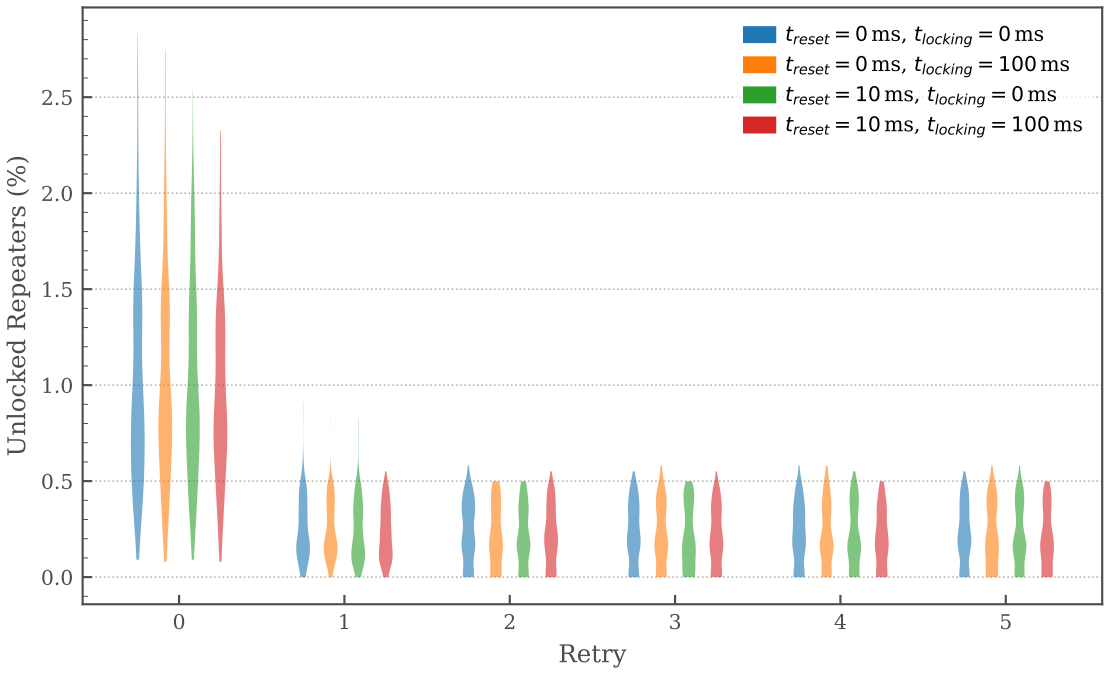


Figure 3.3: Relocking Test – Locking tests on wafer 33 for five different seeds with 50 runs each. The percentage of unlocked repeaters drops after the first retry and stays basically the same afterwards. Different waiting times seem to have no effect on the success rate of locking.

3.2.2 Test

The test introduced in Section 3.1.3 was used to investigate the effect of repeated locking.

Once more wafer 33 was used to perform the locking experiments. The list of HICANNs stayed the same as in Section 3.1.3. We chose the same seeds as given in Table 3.1 and run the tests for each seed 50 times. The waiting time after the stage `TIMING_UNCRITICAL` $t_{\text{reset}} = 0\,\text{ms}$ and 10 ms, which determines how long the repeaters are in reset, and the waiting time after the locking of repeaters $t_{\text{locking}} = 0\,\text{ms}$ and 100 ms were altered.

Figure 3.3 shows that the different waiting times had no effect on the locking success. The number of unlocked repeaters drops after locking is done again and additional relocking attempts do not increase the number of locked repeaters. A low percentage of repeaters can not be locked.

In Chapter 4 we improve the locking test and our evaluation methods. We want to find out if the limited locking success can be attributed to a small set of repeaters which do not lock or forward the wrong addresses.

3.3 NOP-Waits

The synapse controller is responsible for writing/reading weights and decoder addresses of synapses, c.f. Section 2.1.4. During this process specific waiting times have to be met such as pre-charge times or enable delays. The reading of weights is illustrated in Algorithm 2.1. While performing reads and writes the controller sets a busy bit in the status register. By reading this bit the software can check if the controller is ready to receive a new instruction.

Reading this bit takes at least the round-trip time (in the order of 100 µs (empty queue) 100 ms (full queue)) and is therefore quite time consuming compared to the waiting times which are in the order of tens of nanoseconds. The aim of the changes presented in this section is to introduce predefined waiting times for the different actions of the synapse controller after which it is save to assume that the controller finished its task. As these waiting times are significantly lower than the round-trip time a speed up is expected. Since the waiting is realized by sending instructions to the HICANN which do not change the state of the hardware, this waiting scheme was named "NOP-wait"[1] This form of waiting can also be applied to the SRAM controller of the synapse drivers.

Changes presented in this section build up on the work by S. Schmitt and A. Grübl.

3.3.1 Previous Implementation

Before the implementation of the NOP-waits so called "busy waits" were used. Readiness of the synapse controller was ensured by reading the status register continuously and waiting while the busy bit is set, Algorithm 2.1. Since the controller performs it tasks faster than the time it takes until the information of the status bit is available in software the waiting time is in the order of the round-trip time.

3.3.2 Implementation

Pre-charge and delay times are not fixed but can be set for each synapse controller individually. The times are stored in the configuration register in number of HICANN clock cycles. Taking state transitions in the controller and these waiting times into account one can determine how many clock cycles n_{cycles} the synapse controller needs to perform a given task.

An example for the different states and waiting times involved in reading weights is given in Section 2.1.4. Figure 2.3 illustrates the states during this process.

Instructions for the synapse controller are sent in configuration packets to the FPGA from where they are forwarded to the HICANN.

[1] NOP stays for stands for "no operation" and is a common operation in computer instruction sets which does not change the state of the hardware.

After sending a configuration packet with the read/write instruction to the HICANN, we want to occupy the communication channel for the number of clock cycles it takes the controller to finish the task. To do so we write data in the configuration register of the synapse controller. In order to know how many such write instructions we need to send, we have to determine the time between configuration packets in number of HICANN clock cycles.

The clock frequency at which the core logic of the HICANN operates is derived from a phase-locked loops (PLLs) which receives it input from an external clock signal. The output frequency of this PLL can be tuned in a range of 100 MHz to 250 MHz. Since the HICANN core logic operates at a quarter of this frequency the maximal cycle length yields $1/25\,\mathrm{MHz} = 40\,\mathrm{ns}$.

Due to a limited communication bandwidth and a fixed size of configuration packets at most every 80 ns a packet can be sent to the HICANN. Therefore, at least two clock cycles pass during the sending process of a single packet; we have to write the configuration register a maximum of $n_{\mathrm{packets}} = n_{\mathrm{cycles}}/2$ times.

The timing information for the synapse controller is stored in the configuration register, which contents are available in software. A. Grübl determined the constant numbers of state transitions needed for each operation of the synapse controller. Therefore, all information needed to determine the expected execution time are available in software.

NOP-waits are implemented as a simple for loop which runs over the specified number of NOP instructions needed and sends a write instruction of the configuration register in every iteration.

3.3.3 Speed Test

The digital blacklisting test, Section 3.1.3, was once more altered to test the functionality of the changes as well the speed improvement. Each row of synaptic weight (decoder addresses) was read (wrote) 100 times and an average access time was determined, Table 3.2. The same procedure was used for the synapse driver tests: each driver was accessed 100 times.

Synapse Array

For reading we have to open a row, issue eight read instructions (one per column set) and close the row, see Section 2.1.4. After every instruction we have to wait; this sums up to ten waits in total. In comparison reading of a row just takes eight instructions and therefore two less waiting steps. If we divide the difference in access time of the corresponding rows in Table 3.2 by the number of waits we get a speed up per wait of about 130 µs. This is in the order of the expected round-trip time if the traffic on the communication channel from host to FPGA is low, c.f. Section 2.1.5. For a busier link between host and FPGA an even higher speed up is to be expected.

Table 3.2: NOP-waits - Speed Tests – Measurements of synapse array as well as synapse driver access for busy waits and NOP-waits. In case of the synapse array times are given for the access of an entire row (weights) or two entire rows (decoder addresses). Each row/synapse driver was accessed 100 times and the time was averaged over all rows. Synapse array measurements were performed on HICANN 4 on wafer 30 for two different seeds; synapse driver measurements on HICANN 2 on wafer 24.

Operation	Busy Waits (µs)	NOP-Waits (µs)
writing weights	1046 ± 101	15 ± 8
reading weights	3365 ± 173	2079 ± 149
writing decoder addresses	2090 ± 162	26 ± 9
reading decoder addresses	6724 ± 243	4176 ± 221
writing synapse driver	815 ± 3	10 ± 1
reading synapse driver	813 ± 3	850 ± 2

The waiting time was the dominating factor in the time it takes to perform a write. As a consequence writing speeded up by a factor of about 70. The relative speed up in reading was much lower.

Synapse Driver

In the routine which is responsible for writing synapse drivers seven busy waits were replaced by the new waiting scheme. Table 3.2 shows that this operation increased by about 715 µs. Dividing this time difference by the number of replaced waits puts us again in the range of the expected round-trip time.

Reading of synapse drivers slowed down compared to the previous implementation. This is due to the fact that no waits were previously performed. In contrast the hardware specification states that waiting is needed after each read access [8]. As a consequence we introduce a NOP-wait after every read; three waits were added in total. This resulted in the increase execution time.

4 Extended Locking Test

In the Section 3.2 we saw that relocking of repeaters can increase the total number of locked repeaters in the network. Nevertheless, it was rarely possible to lock our tested networks with around 1200 repeaters completely; a low number of repeaters stayed unlocked in each retry.

Our aim is to extend the previous developed test to analyze if the unsuccessful locking can be attributed to a small set of repeaters. To do so we will extract the locked and unlocked repeaters in each try and provide evaluation methods to determine repeaters which are unlocked frequently. In addition, we provide a visualization of the routes which contain unlocked repeaters in order to check dependencies between different repeaters.

We change the structure of the test such that it easily be executed on different wafers in the BSS-1 system. Furthermore, we introduce the variation of the voltage which is used to reset the DLL circuit in the repeaters as well as the clock frequency which is used for the on-wafer communication.

4.1 Test Routine

The routine is oriented on the test described in Section 3.1.3. This time we do not place the populations randomly but establish connections between random populations.

The user provides a number of populations N_{pop} and their neuron size N_{neurons}. If the user provides an optional list of HICANNs the populations are manually placed on these chips, otherwise the standard placement algorithm of marocco is used. We now create a randomly ordered list of these populations and create all-to-all connections between adjacent list items. A seed provided by the user is set before performing the randomization of the list.

Before executing the experiment on the wafer, low level parameters are set. These parameter include: sleeps after configuration stages as in Section 3.1.3, reset voltage V_{dllres} and PLL-frequency ν_{pll}.

A bash script is used to define a set of hardware settings which should be tested. It runs over all possible combinations, tests the locking for this configuration several times and saves the result in a folder with a unique id for the set of experiments (date and

time). The logging output of the BSS-1 operating software is filtered for information relevant for the determination of unlocked repeaters. Wafer statistics such as voltages and temperatures are saved in a JSON file.

4.2 Experiment Results

An initial test in Section 3.2.2 already suggested that the sleeping times have no effect on the number of locked repeaters. In the first section we want to confirm this for the new test.

Afterwards, we test how the reset voltage of the DLL circuits and the operation frequency ν_{pll} influence the number of locked repeater.

4.2.1 Varying Sleeps

Experiment Parameter – ID: `2020-06-27 09h52m`

- Wafer: 24
- N_{pop}: 50
- N_{neurons}: 40
- t_{reset}: 0 ms and 10 ms
- t_{locking}: 0 ms and 100 ms
- V_{dll}: 200
- ν_{pll}: 125 MHz

As a start we want to confirm with the new test that the waiting time after pulling/releasing the reset has no clear effect of the number of locked repeaters. We use the same sleeping times and seeds as in Section 3.2.2 but this time we run the tests on wafer 24 and allow for a maximum of ten locking retries.

We chose the same seeds as in the previous tests. This time our networks were made up of about 600 to 900 repeaters. The tests confirmed the previous findings: the total number of locked repeaters can be increased by repeatedly putting the repeaters in reset and the varying sleeps had no clear effect on the locking success Fig. 4.1 (A). Nevertheless, just two networks could be locked completely in all runs (and at least for all but one combination of sleeping times).

The locking success of the different networks, subfigures (B) to (D), show some differences for different combinations of sleeping times. However, no combination seems to be systematically superior. This is also evident in subfigure (A).

In case of seed 35 the network already shows a low percentage of unlocked repeaters in the first try. After the third try the network was locked completely regardless of the waiting times. This network included the lowest number of tested repeaters. For the other network which could be locked completely in most of the runs, seed 94, considerably more retries are needed until the number of unlocked repeaters drops to zero.

All other networks stayed unlocked in a high number of runs. The number of locked repeaters increases at first but then stays roughly constant. No specific sub set of "bad" repeaters could be determined and also no single repeater block seemed to be responsible for the unsuccessful locking.

We observed that the locking gets occasionally stuck in loops: routes A, B, C, D are unlocked in retry 2; E, F in retry 3; A, B, D in retry 4; E, F, G in retry 5 and so on. The routes present in these loops were similar in each run (but different for different seeds) and the algorithm was rarely able to escape these loops. One reason for the occurrence of this periodic behavior is that entire repeater blocks have to be put in reset such that a few initially locked repeaters may lose the correct timing.

Such loops did also occur in the network with seed 94 which could be locked after many retries. Here the algorithm was able to escape the loop in a low number of retries. A further increase of the maximum number of retries is only partly practical. While a single retry takes only about half a second for the given network, the time can reach several seconds if the networks are bigger.

4.2.2 Varying ν_{pll} and V_{dllres}

Experiment Parameter – ID: `2020-06-28 23h21m`

- Wafer: 24
- N_{pop}: 150
- N_{neurons}: 40
- t_{reset}: 0 ms
- t_{locking}: 0 ms
- V_{dll}: 0 to 1000 in steps of 100
- ν_{pll}: 100 MHz, 125 MHz and 200 MHz

We now want to vary the frequency at which the on-wafer communication network is operated. A PLL is used to derive a frequency ν_{pll} which is then used during the encoding of the injected signal. Repeaters and synapse drivers then derive the timing from the frame length which is given by $1/\nu_{\text{pll}}$.

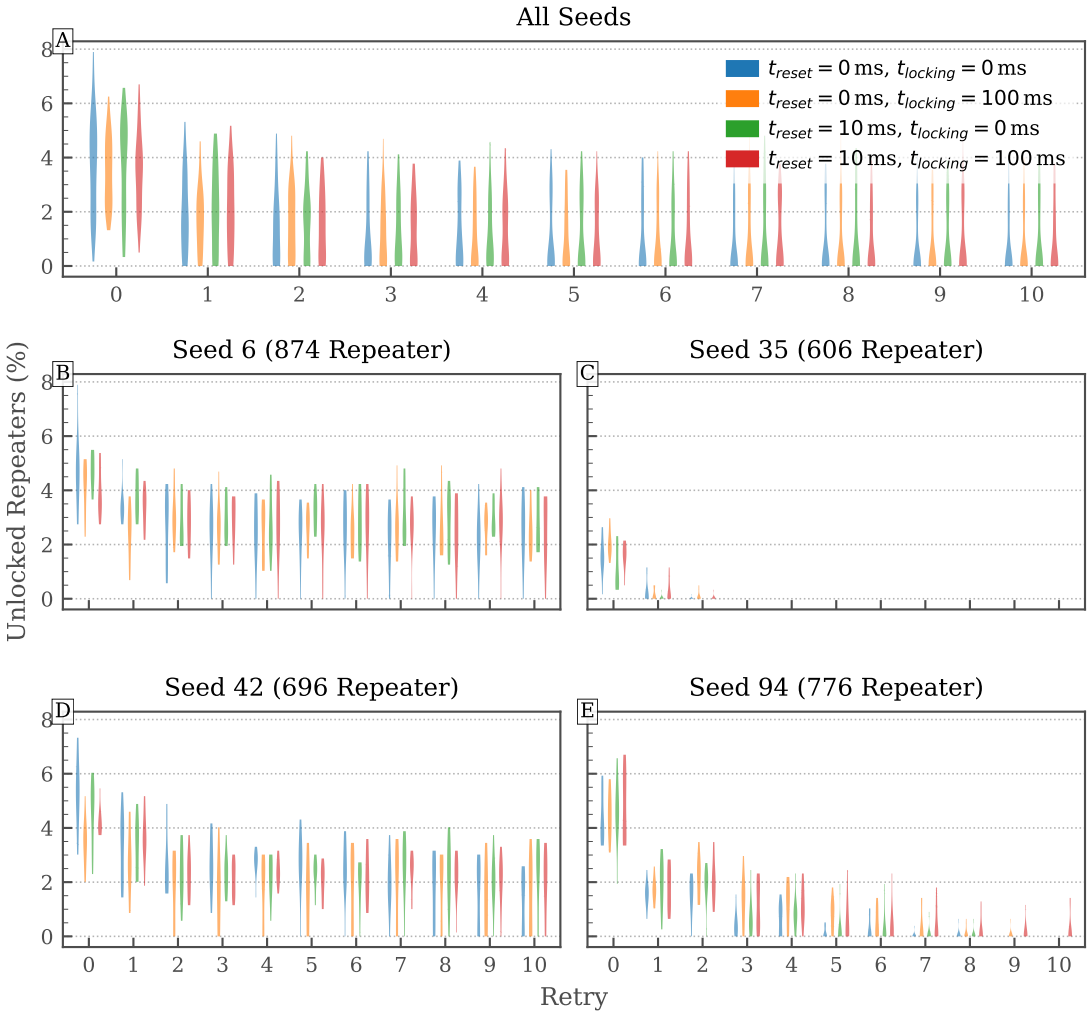


Figure 4.1: Locking Test - Varying Sleeps – Locking test for five different seeds with ten runs each. A maximum of ten retries were done in each run. The clock frequency was set to $\nu_{\text{pll}} = 125\,\text{MHz}$ and the reset to $V_{\text{dllres}} = 200$. **(A)** Violin plot of repeater locking success for all seeds combined. **(B-E)** Locking results for a selected subset of seeds. Only two networks could be locked in most of the runs. For the other networks relocking has a positive effect but a small number of repeaters stay unlocked.

The reset voltage V_{dllres} should be chosen such that the rising edge of the stop bit lies in the detection region (blue region in Section 2.1.3) [6]. We therefore have to adjust the reset voltage to the clock frequency. The voltage can be set via a 10-bit digital-to-analog converter (DAC) and is stored in floating gates at the center of the chip. All values of V_{dllres} given in this thesis describe the input value of this DAC.

As preliminary results showed that the networks in the previous test can be locked easily when the clock frequency is increased, we expanded our networks to include 150 populations. This resulted in networks with 3600 to 3900 repeaters.

The mapping of one of these networks is illustrated in Fig. 4.2. About 50 HICANNs are used to place the neuron populations and three times as many to establish connections between the different populations. Marocco uses chips without neuron populations on them to redirect routes if a more direct path is not available. This is visible in the figure where a few routes go south at first, then to the left/right and finally back north. Since blacklisted HICANNs can not be used for routing, these routes bypass the center part of the chip were a lot of chips are unavailable.

Figure 4.3 summarizes the results for all three tested frequencies. At the minimum frequency which can be chosen for the operation of the on-wafer communication circuits, $\nu_{\text{pll}} = 100\,\text{MHz}$ a large number of repeaters stay unlocked, subfigure (A). Even at the optimal value of the reset voltage about a fifth of all repeaters deduced a wrong timing from the incoming signals.

While relocking increases the number of locked repeaters around the optimal reset voltage, relocking has a negative effect when the reset voltage is non-optimal. This behavior can be observed for all tested frequencies.

At the frequency which is used as a default for the BSS-1 system, $\nu_{\text{pll}} = 125\,\text{MHz}$, the percentage of unlocked repeaters falls under five percent. Nevertheless, no network configuration was locked successfully in any run. The optimal reset voltage seems to lie somewhere between the default of 200 and 300.

For the highest tested frequency of $\nu_{\text{pll}} = 200\,\text{MHz}$ all but two networks could be locked successfully in each run. Complete locking of the networks could be achieved within one retry when the reset voltage was equal to 300 or above.

The unsuccessful locking of the two networks could be traced back to a small set of repeaters. In both networks at most two routes were responsible for the failed locking. The first unlocked repeaters in these routes were `H179VR080`[1] (seed 35) and `H179VR087` (seed 94). For a seed of 94 `H179VR69` was also occasionally unlocked; interestingly repeaters which followed later in the route were locked according to the test. All repeaters belong to the same repeater block (`H179RB1`[2]). Furthermore, both receive inputs from repeaters which lay on the same repeater block on the preceding HICANN (`H215RB1`).

[1] The first number encodes the HICANN on which the repeater is located, 179. "VR" stands for vertical repeater ("HR" for horizontal repeater). The last number encodes the number of the repeater on the HICANN.

[2] "RB" stands for repeater block.

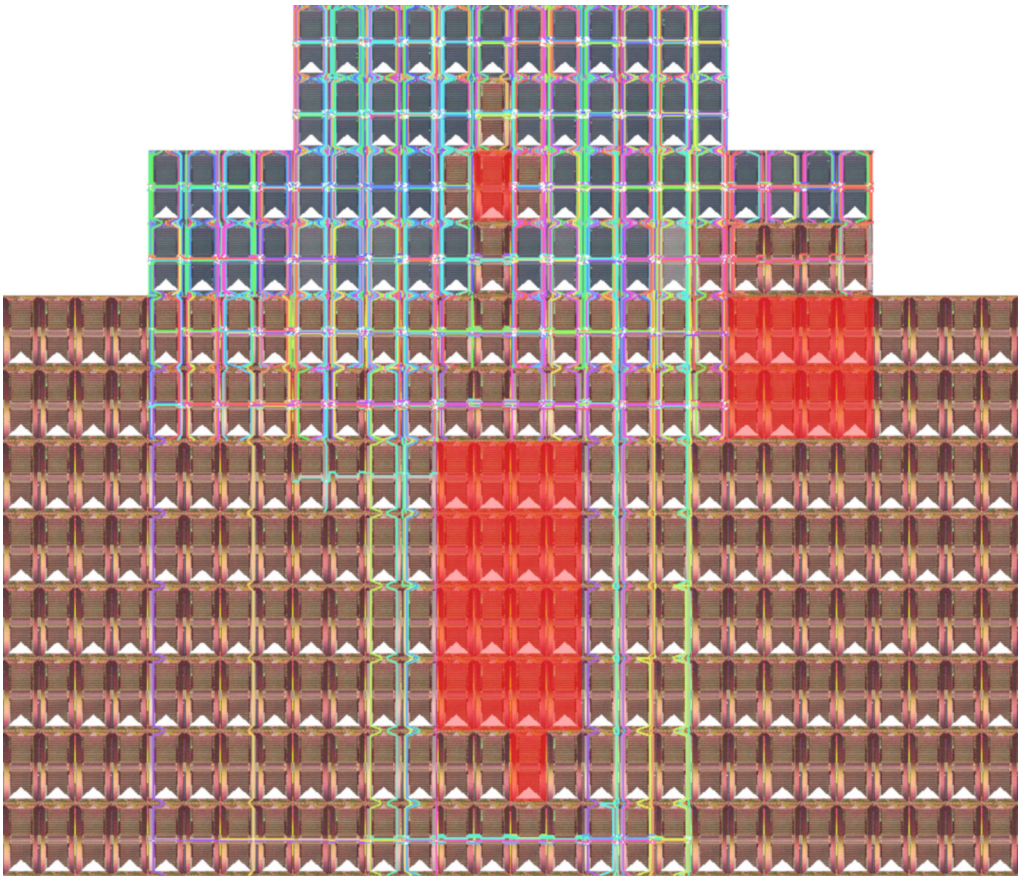


Figure 4.2: Locking Test - Mapping Result Seed 6 – Mapping result for the network with seed 6 and $N_{\text{pop}} = 150$ and $N_{\text{neurons}} = 40$. Neurons are marked by a blue color. The higher the opacity the more neurons are placed on the HICANN. Neurons are placed from left to right. Each HICANN hosts a little more than 100 neurons. Fully blacklisted chips are colored red. Colored lines represent the connections between sending repeaters and synapse drivers. A total of 3939 repeaters were used in this network.

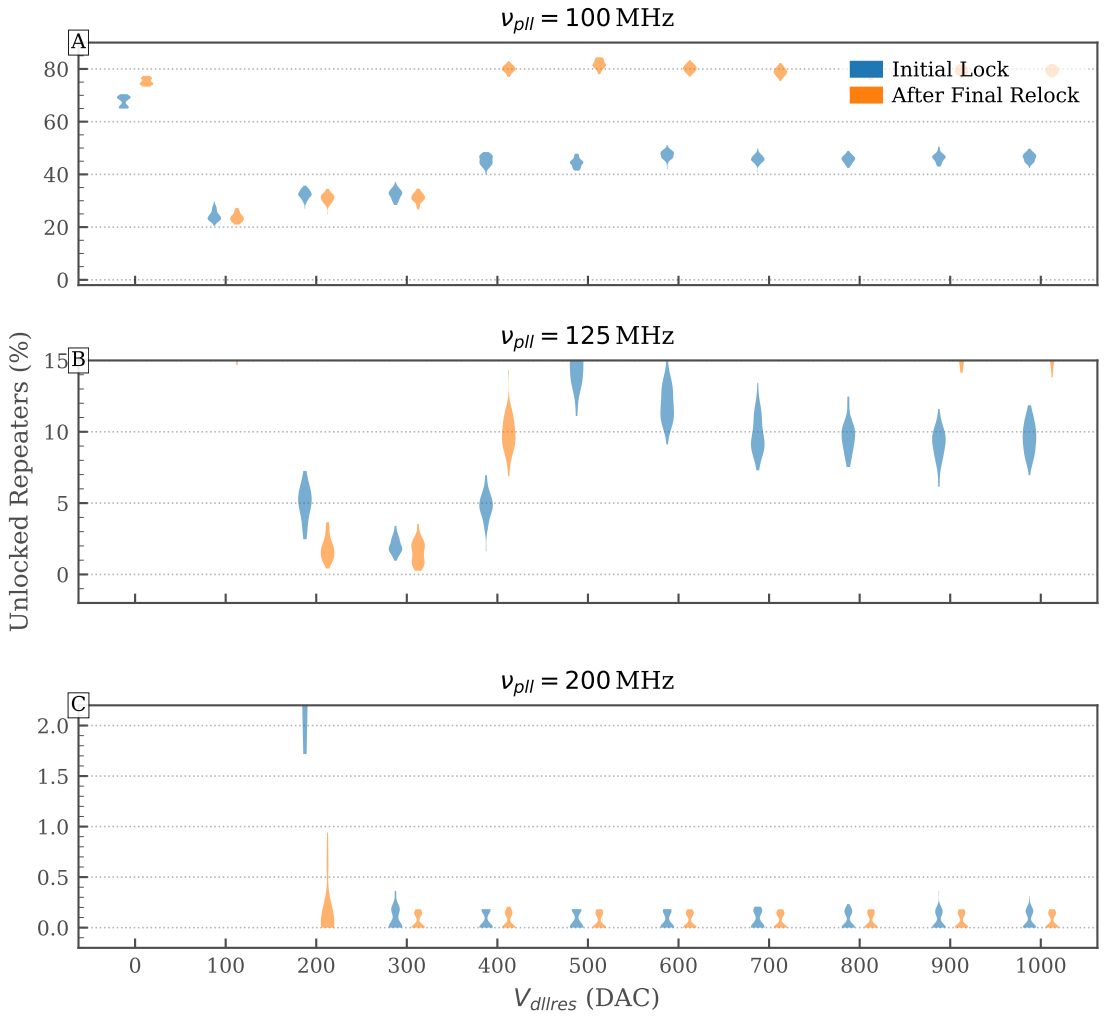


Figure 4.3: Locking Test - Varying $\nu_{\mathbf{pll}}$ and $V_{\mathbf{dllres}}$ – Locking test for five different seeds with ten runs each. A maximum of ten retries were done in each run. **(A)** The number of unlocked repeaters is high for the lowest frequency at which the on-wafer communication can be operated. In case the reset voltage is optimal, relocking has a small positive effect on the number of locked repeaters. Otherwise, the number of locked repeaters decreases when relocking is done. **(B)** At the default frequency of on-wafer communication of the BSS-1 system the number of unlocked repeaters decreases dramatically. The optimal reset voltage is in a small range around 200 to 400. **(C)** A even higher frequency decreases the number of unlocked repeaters even further and the locking success is less dependent on the reset voltage.

The repeater blocks in question were not used in any other networks and in no other routes than the ones which could not be locked. Further tests with more seeds and other population sizes revealed a few more repeaters on `H179RB1` which can not be locked. However, the test also revealed one repeater on this block which could be locked successfully in all tries. The number of routes which were routed over `H179RB1` was to low to determine whether the repeater block might be defect. More tests with different seeds are needed or our test should be extended such that specific repeater blocks can be targeted.

A look at the blacklisting data reveals that the repeaters of HICANN 215 were not tested during the digital test and that a relatively high number of synapse drivers are blacklisted on HICANN 179. It is not clear whether there is a connection between the bad locking performance and the results from the digital blacklisting.

We conclude that an increased operation frequency of the on-wafer communication circuits has a positive effect on the locking success. Additional tests have to determine if signal encoding and decoding are still reliable for higher frequencies. Tuning the reset voltage of the DLL can be beneficial at low frequencies while the circuits are more robust in variation of the reset voltage if the frequency is high.

5 Synchronous Firing Chain

In this chapter we want to investigate what challenges arise during the emulation of networks which span over multiple HICANNs on the BSS-1 wafer system and if the in Section 3.2 introduced relocking facilitates the implementation of these networks. For that purpose we use a synfire chain with FFI which is easily scalable from samll to large network sizes.

At first, we want to confirm the functionality of our implementation and to get a better understanding of the network dynamics. Therefore, we start by simulating a synfire chain in NEST; this also allows us a comparison with hardware result later on. For the initial transfer to hardware we want to look at a less complex network such that we also run simulations for a synfire chain without FFI and background input.

We initially run simulations/emulations of chains with less than ten chain links. This allows for an easier transfer to hardware since only a small part of the wafer is used. Using this simplified, small network we want to investigate the influence of hardware induced restrictions such as limited communication bandwidth, limited number of synapses and parameter variations.

Finally, we increase the chain length and look at the implementation of larger scale networks on the BSS-1 system.

5.1 Experiment Description

We use the PyNN implementations of NEST and BSS-1 to describe the biological properties of the network such as populations, projections and neuron parameters. This description is handled in the dedicated class `SynFireChain`, Section 5.1.1.

Another python script `run.py` takes care of experiment control; for example it places the populations on pre-selected chips or saves the experiment results. A third script allows to define a set of biological values for which the network should be simulated. It takes care of submitting the jobs to the compute cluster and the bookkeeping of results. In the last subsection we quickly describe which evaluation routines we use and why, Section 5.1.4.

Table 5.1: Synfire Chain - Populations – An illustration of the network can be found in Fig. 2.7. Neuron parameters of the excitatory and inhibitory populations are given in Table 5.2. The rate of the background neurons r_B will be determined during experiment. The structure of the stimulus spike train is described in the text.

Population x	Size N_x	PyNN type
Stimulus (S)	$= N_E$	`SpikeSourceArray`
Excitatory (E)	100	`IF_cond_exp`
Inhibitory (I)	25	`IF_cond_exp`
Background (B)	32	`SpikeSourcePoisson`

5.1.1 Biological Description - `SynFireChain.py`

The class `SynFireChain` takes care of the biological description of the network. Upon construction it takes a instance of PyNN (either the NEST or BSS-1 implementation) and a set of network parameters such as the number of groups or the sizes of the different populations. We will use the topology and neuron parameters described in this section as base parameters for our experiments.

Populations

Each group consists of an excitatory, inhibitory and background population, compare Fig. 2.7. The properties of the network were passed to the class upon construction. Calling the build member function results in creation of the network.

The sequence of creation influences the neuron placement on the wafer and therefore might have an effect on the topology of the mapped network as well as on the dynamics of the chain. We create one group after another and within each group excitatory neurons before background neurons before inhibitory neurons.

Excitatory as well as inhibitory populations consist of LIF neurons (`pynn.IF_cond_exp`), see Eq. (2.1). In the base configuration they share the same parameters, Table 5.2. Background neurons are simulated as Poisson spike sources (`pynn.SpikeSourcePoisson`) with a rate r_B.

The analog circuits on hardware are subject to fixed pattern variations caused by the manufacturing process. Floating gates also contribute to neuron to neuron variations on hardware. In order to account for these variations we allow to include these variations in NEST simulations. If a simulation is run with distributed neuron parameters each neuron parameter is drawn from a Gaussian distributions with a standard deviation of 5 % (with respect to the mean value). The mean values are, if not stated otherwise, taken from Table 5.2.

After all groups have been created we deal with the stimulus input. The stimulus is

Table 5.2: Synfire Chain - Neuron Parameters – In the default configuration neurons in the excitatory and inhibitory populations share the same neuron parameters. For the meaning of the variables see Section 2.2.1.

Parameter		Default Value
Membrane Capacitance	C_{m}	0.29 nF
Excitatory Reversal Potential	$E_{\text{rev-exc}}$	0.0 mV
Inhibitory Reversal Potential	$E_{\text{rev-inhib}}$	−75.0 mV
Constant Current Offset	I_{offset}	0.0 nA
Membrane Time Constant	τ_{m}	10.0 ms
Refractory Time Constant	τ_{ref}	2.0 ms
Excitatory Synaptic Time Constant	$\tau_{\text{syn-exc}}$	1.5 ms
Inhibitory Synaptic Time Constant	$\tau_{\text{syn-inhib}}$	10.0 ms
Reset Potential	v_{reset}	−70.0 mV
Rest Potential	v_{rest}	−70.0 mV
Spiking Threshold	v_{thres}	−57.0 mV

provided by a group of $N_S = N_E$ neurons; the spikes of each neuron resemble a Gaussian pulse packet $(a_{\text{in}}, \sigma_{\text{in}})$, compare Fig. 2.5. We want to test response of the synfire chain to a large number of different input stimuli. To speed up the experiment we combine all stimuli, which we want to test, in a single spike train. To do so we introduce inter stimulus inhibition: each neuron receives a stimulus input followed by an inhibition after $t_{\text{post-stimulus}}$ ms and the next stimulus after waiting an additional $t_{\text{pre-stimulus}}$ ms. The output spike train represents a single draw of the input stimuli, Fig. 5.1.

In order to account for statistical variations in the drawn spike times, we present several draws to the network; each time we redraw the spike times of the stimulus from the same Gaussian distributions. Numpy's random module is used to draw the spike times from normal distributions and the different draws are then "glued" together to form a single spike train which represents one trial, Fig. 5.1.

Projections

Projections between inhibitory, excitatory and background population all have a fixed number of pre synaptic connections. The number of connection, the weights and delays are summarized in Table 5.3.

In our implementation we take the weight $w_{E_i \to E_{i+1}}$ as a reference weight and scale the other weights by the factors $g_{E_i \to I_{i+1}}$, $g_{I_i \to E_i}$ and $g_{B_i \to E/I_i}$. The weight of the connections between stimulus and the first population are the same as in between groups, i.e. the connection from the stimulus to the excitatory population has a weight of $w_{E_i \to E_{i+1}}$ while the weight to the inhibitory population is further scaled by $g_{E_i \to I_{i+1}}$.

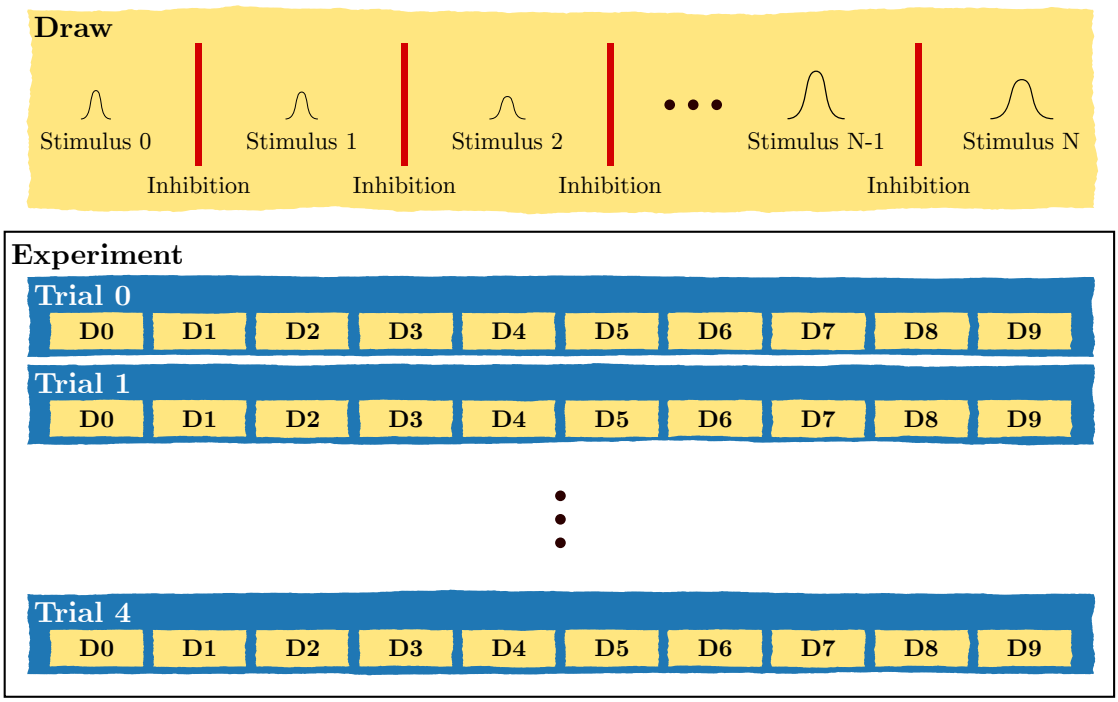


Figure 5.1: Structure of a Single Experiment – In each draw we test several different input stimuli. In order to "reset" the neurons before a new stimulus input arrives, we send inhibitory spikes between different stimuli. For every draw we redraw the spike times in the stimulus packets. The total experiment is made up of several trials which consist of several draws each. Before each trial we configure the hardware such that the neuron parameters and weights may vary slightly from trial to trial (due to floating gate variations).

Table 5.3: Synfire Chain - Projections – The capital letters denote the type of neuron population: stimulus (S), excitatory (E), inhibitory (I) and background (B). The group to which the population belongs is indicated by the subscript. All connections are realized with a fixed number of pre-synaptic neurons (`pynn.FixedNumberPreConnector`). $w_{E_i \to E_{i+1}}$ is used as a reference weight and all other weights are scaled accordingly The weight between stimulus and an excitatory neuron in the first group is the same as the weight between excitatory populations in different groups (1 nS). For the connection to inhibitory neuron the same holds true, i.e. the weight is 3.5-times higher than the reference weight. Compare Fig. 2.7.

Connection x	Number of Connections N_x	Delay D_x (ms)	Weight w_x (nS)	Scaling g_x
$S \to E/I_0$	$= N_{E_i \to E_{i+1}}$	20	same as inter-group weight	
$E_i \to E_{i+1}$	60	20	1	-
$E_i \to I_{i+1}$	60	20	-	3.5
$I_i \to E_i$	25	5	-	2
$B_i \to E/I_i$	8	2	-	1

The specified delays are only relevant for NEST simulations as delays can not be adjusted on BSS-1. They are determined by the hardware components and yield around 1.2 ms to 2.2 ms (biological time) [24].

Again the order of projection definition matters for the mapping done by marocco. At first the connections between the stimulus and the first group are established. After that we once more iterate over the different groups where we first create $E_i \to E_{i+1}$, followed by $B_i \to E_i$ and finally the connection to the inter stimulus inhibition. For projections involving inhibitory populations the order of creation is: $E_i \to I_{i+1}$, $I_i \to E_i$, $B_i \to I_i$ and inter stimulus inhibition.

Again we allow for variations in the synaptic weight and delay to account for variations on hardware. While the delay is set to a mean value of 1.5 ms and has a standard deviation of 5 % for simulations which include parameter variations, the synaptic weight is taken to have a standard deviation of 50 %.

Experiment Data

We use neo to store experiment results [25]. The class `SynFireChain` provides functions to create a `neo.Block` with experiment relevant annotations such as the number of groups or the list of input stimuli. Another function provides a `neo.Segment` with all recorded spike trains. Each spike train is named by its neuron id and annotated with its group and type ("exc" or "inhib"). Neurons in the stimulus population have a label "stimulus" which is set to true.

5.1.2 Experiment Control

Another python script `run.py` handles simulator specific settings and saves experiment data to files.

Command line arguments are used to provide the script with all necessary information such as the simulation backend, random seeds and parameters of the synfire chain.

Depending on the simulation backend different steps are taken. In case of NEST we set the random seeds, the simulation time step as well as the maximum synaptic delay. For BSS-1 we have adjust a larger number of parameters.

Settings for BSS-1

In case of a hardware emulation we set several low level settings. We choose the `ParallelHICANNv4Configurator` to configure the chips[1]. Calibration and blacklisting data is loaded if provided by command line arguments.

Manual placement of excitatory and inhibitory populations as well as of the stimulus input is possible. The user can provide for each type of population a separate list of HICANNs. In case of excitatory and inhibitory populations a single population is placed on each HICANN in the list. For the stimulus input we loop over all neurons and place one neuron at a time on a chip. If we reach the end of the list we restart at the beginning. This is done until all neurons are placed.

A command line argument determines if mapping[2] is performed or if the mapping results should be loaded from file. In case mapping is done results such as synapse loss and mapped weights are stored in text files.

Experiment Run

After all setup steps are completed the experiment is run. The user can provide the number of trials via a command line argument. The scripts loops over the trials and adds the spike trains of each trial to a different segment of a single `neo.Block`. At the end of the experiment this block and some meta information such as trial start and end times are saved in a folder provided by the user.

5.1.3 Scanning of Parameters – `scan_param_range.sh`

In order to allow scanning of a large range of neuron and network parameters we provide a bash script. In the script sets of values for different parameters such as chain length N_{group}, weights or number of neurons are defined.

[1] The smart configurator was not able to record spikes correctly after a `pynn.reset()` was performed. Debugging is still ongoing.

[2] Mapping refers to the transformation of the biological network description to a hardware representation.

The script creates a base folder with an unique id for the set of experiments (date and time). It then loops over all possible parameter combinations and submits a job to the computing cluster. While doing so it enumerates each combination[3], creates dedicated folders for each experiment's results and creates an overview file over all configurations as well as within each configuration an overview file for each weight combination. The storage location of the results and some additional information can be found in Appendix A.1.

We use a custom build of the BSS-1 software stack to run the experiments, container image `2020-06-07-4740-1`, Table A.1. This image includes a new mapping routine of synaptic weights which is developed by M. Wehrheim and S.Schmitt and is not yet part of the production code [26].

In case the experiments are run on the wafer the script also saves wafer statistics such as voltages and temperatures via the Graphite interface.

5.1.4 Evaluation

We use pulse packets to classify the response of the network. Therefore, we want to extract these packets from the recorded spike trains and determine their strength and temporal spread.

The spike trains are saved in such a way that we can easily extract all trains which are related to a specific group and neuron type. Furthermore, we know the input time of each stimulus and the time which was waited before and after this input. Therefore, we can easily extract all spikes which fall in the time range of a single stimulus and belong to the same population. The spike times of these spikes are then combined in a list from which we want to extract the Gaussian pulse packets.

If the time of the first and last spike is less than 10 ms apart, we assume that the neurons do not spike spontaneously and assign all spikes to the pulse packet. The strength of this packet is determined by the total number of spikes; the temporal spread is given by the standard deviation of spike times.

In case the spikes are distributed over a longer time range, we organize the spikes in a histogram and fit (after some checks) a Gaussian to the histogram. For details see Appendix A.4.

5.2 Short Chains

In this section we look at a relatively short chain with six links. This chain length is comparable to the ones reported in Kremkow et al. [22] and Petrovici et al. [24].

[3] We combine parameter combination in so called "configurations" which share all parameters but the weights. So if we loop over $N_{\text{group}} = \{6, 20\}$ and $w_{E_i \to E_{i+1}} = \{1.0, 1.2\}$ nS we get two configurations `C0/C1` and within each configuration two weight variations `W0/W1`.

We started by simulating the network with inhibition and background in NEST. For the transfer to hardware we chose a less complex network by excluding FFI and background input. This network was simulated in NEST, Section 5.2.2, before emulated on hardware, Section 5.2.4. Finally, we emulate a version with FFI but without background on BSS-1, Section 5.2.6.

5.2.1 NEST with Inhibition and Background

Experiment Parameter – ID: `2020-06-01 08h10m`

- Chain Length: 6
- Structural Changes (compared to Table 5.3 & Table 5.1):
  - $w_{E_i \to E_{i+1}}$: 1 nS and 2 nS
  - $g_{E_i \to I_{i+1}}$: 2.0, 3.5
- Neuron Parameter Changes (compared to Table 5.2)
  - r_{B}: 200 Hz, 300 Hz and 400 Hz
- Other Changes: None

Parameters similar to the ones in Petrovici et al. [24] and Kremkow et al. [22] were chosen to verify the correctness of the network description. They are the same as described in Section 5.1.1.

In total three different configurations with different background rates were simulated. Each configuration was simulated for two weights $w_{E_i \to E_{i+1}}$ and two weight factors $g_{E_i \to I_{i+1}}$. For each network and weight configuration a single trial was run. A single trial consisted of ten draws and took 53 min on average[4].

Spiking Behavior

While all runs show spiking activity, reliable pulse propagation is not possible for all configuration and weight combinations, Table 5.4. "Reliable propagation" means that synchronous activity in the last group is only initiated by a sufficiently synchronous stimulus input; it is not caused by background fluctuations. For a weight of $w_{E_i \to E_{i+1}} = 1$ nS reliable propagation is possible for the two lowest background rates. In case of the highest background rate the fluctuations are too strong such that synchronous firing in the final group may be initiated by random spiking, Fig. A.2.

For a higher base weight propagation for the lowest background rate is not reliable anymore. Pulse packets are introduced by random background noise and the behavior

[4] Single thread execution on `Intel Xeon CPU E5-2643 v2`.

Table 5.4: NEST with Inhibition and Background - Reliable Propagation – Marks indicate if propagation of input stimuli was reliable (✓) or the activity in the last group was affected by random background activity (✗).

r_B (Hz)	$w_{E_i \to E_{i+1}}$	1 nS		2 nS	
	$g_{E_i \to I_{i+1}}$	2	3.5	2	3.5
200		✓	✓	✗	✗
300		✓	✓	✓	✓
400		✗	✗	✓	✓

is comparable to Fig. A.2. When increasing the background rate the activity of the inhibitory neurons increases, Fig. A.3. This leads to a lower mean of the membrane potential. Therefore, background spikes are less likely to cause a threshold crossing and propagation is reliable again.

Figure 5.2 displays a few example spike trains. In case of (1,1) a single excitatory pulse packet is initiated in each group. The inhibitory group spikes slightly earlier due to the higher weight of incoming excitatory synapses. Broad pulse packets may cause several spike packets in the first group. Subsequent groups filter out multiple packets such that after a few groups only one packet is left, subfigure (B).

If the strength is not sufficient or the temporal spread to high, the packet dies out. This is illustrated for the packet (1,4). The inhibitory population in the first group shows strong activity for a few milliseconds which leads to an inhibition in the excitatory population and only a few excitatory neurons spike. Even though the response to (8,9) has a high temporal spread in the first group, it still able to initiate synchronous spiking in the second group. But the strength of this pulse is not strong enough and decreases in the next groups even further such that the signal finally dies out in group 3.

To determine the group to group delay of pulse packets we only want to consider responses which are made up of a single packet. Therefore, we filter for $0.1 < a < 1.5$ and $\sigma < 3$. We then measured the delay between adjacent groups and averaged over all values. This yields a group to group delay of about 11 ms and is therefore dominated by the synaptic delay $D_{E_i \to E_{i+1}} = 10\,\text{ms}$. A dependency on the weight between excitatory populations could not be observed as the temporal spread of the pulse packets is below the standard deviation of the determined delays.

Phase Diagram

Highly synchronous pulse packets (small σ_{in}) are transported along the chain while broad packets die out if their strength (a_{in}) is not sufficiently large, Fig. 5.3. This is similar to the observations by Kremkow et al. [22] and Petrovici et al. [24], compare Fig. 2.8

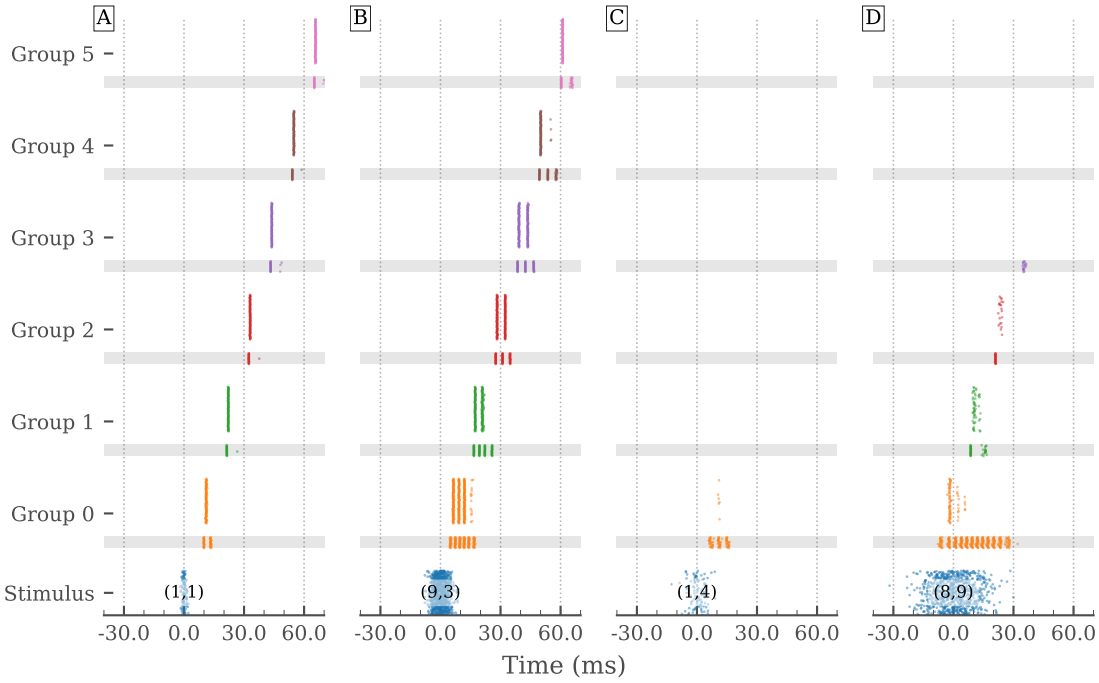


Figure 5.2: NEST with Inhibition and Background - Spike Trains – Simulation with base parameters (Section 5.1.1) and $r_B = 200\,\text{Hz}$. Spikes of neurons in the inhibitory populations are plotted on a gray background. **(A)** A highly synchronous input only initiates one excitatory pulse packet in each group. **(B)** Broader inputs may cause multiple packets which die out in the subsequent groups. **(C,D)** If a packet dies out depends on strength and temporal spread. Simulations with base parameters (Section 5.1.1) and $r_B = 200\,\text{Hz}$.

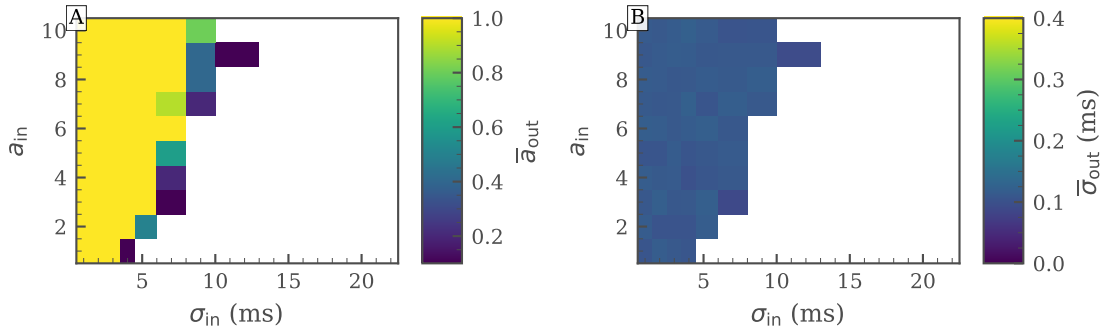


Figure 5.3: NEST with Inhibition and Background - Phase Diagram – Simulation with base parameters (Section 5.1.1) and $r_B = 200\,\text{Hz}$. Responses in the final group averaged over all ten draws. Average strengths smaller 0.1 are not printed. **(A)** When transmission is successful the activity is homogeneous in the last group. Only near the separatrix the success depends on the exact input spike times and transmission fails for a few draws. **(B)** The standard deviation of the output packet is low and almost homogeneous in the region of successful transmission.

(C). A lower synchronicity can, to some extent, be compensated by a higher strength: while (1,4) dies out before reaching the final group the less synchronous packet (10,7) is propagated successfully.

In the region of the phase space where transmission is possible the strength of the output is identical: the activity is independent of the initial input packet and consists of a single packet with strength $a_{out} = 1$. Only near the separatrix the output strength in the final group is reduced. Here the pulse packet is successfully propagated along the chain in a fraction of draws: success depends on the exact distribution of the incoming spikes. In case a pulse packet reaches the final group the strength of the output is again $a_{out} = 1$.

The standard deviation also shows an almost binary response. In the basin of attraction the standard deviation assumes a low value of about 0.1 ms. This is expected since the simulation does not include any variation in neuron parameters or transmission delay such that the temporal spread is small and homogeneous. A different set of incoming synapses for each neuron and the background are the only sources of variability.

At low rates the spontaneous activity of the neurons was relatively low such that evaluation method 2 was used to evaluate successful transmitted packets ($a_{out} > 0.1$) in the last group; all spikes are assigned to the pulse packet and the standard deviation is determined from their time, compare Appendix A.4. For higher background rates the activity in the network increased and evaluation method 5 was used more often to extract the pulse packets in case of successful transmission or a Gaussian was fitted to the spike time histogram. In case that the pulse died out there are no spikes in the final group for a low background rate and evaluation method 1 was used. When the

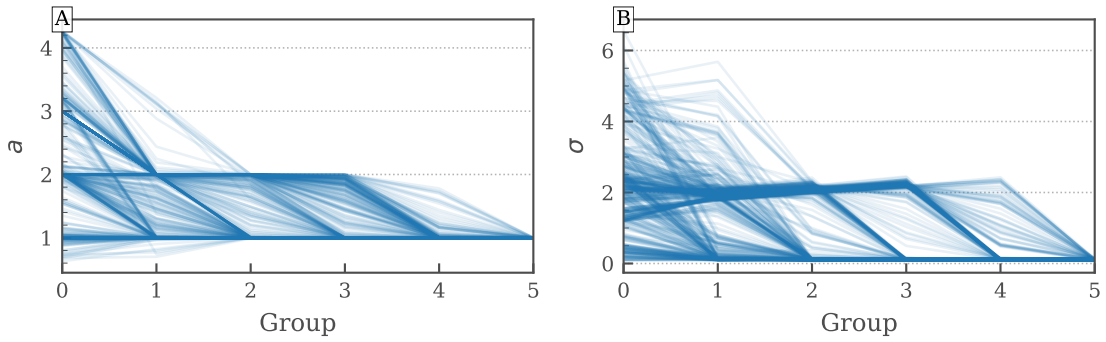


Figure 5.4: NEST with Inhibition and Background- Evolution Along the Chain – Evolution of strength and standard deviation for all tested (100 different input pulses and ten draws per stimuli leads to 1000 tested pulses) pulses which reached the final group ($a_{\text{out}} > 0.1$). **(A)** The spread in strength is the highest in the first group. A single pulse packet can initiate more than 4 spikes per neuron. The strength of the packets approach 1 while traveling along the chain. **(B)** The standard deviations of the pulse packets also converge towards a common value. Parameters: $r_{\text{B}} = 200\,\text{Hz}$, base parameters as in Section 5.1.1.

spontaneous activity was higher methods 0, 4 and 3 were used to determine the strength of weak pulses in the last group.

Evolution Along the Chain

Figure 5.3 already indicated that the activity in the final group tends to be homogeneous for successfully transmitted pulse packets. This convergence towards a fix value is illustrated in Fig. 5.4. The spread in strength is relatively large in the first group and depends on the initial strength and temporal spread, compare Fig. A.4. In the subsequent groups the strength of "too strong" pulse packets diminish while the strength of weaker packets increases to 1. For our chosen set of input stimuli the output strength is identical for all packets which reach the final group. In case of even stronger input strengths this might not hold true and more chain links would be necessary to unify the final response.

The same convergence can be observed for the standard deviation. In the first group the temporal spread still depends on the input packet; in the last group all packets share a common value (ignoring small deviations due to connectivity and background noise). We observe that the change of temporal spread is not monotonic for all curves. The temporal spread may increase at first but still converge in later groups towards a common value. This is in accordance with observations for simple synfire chains without FFI [18, 20].

5.2.2 NEST without Inhibition and Background

Experiment Parameter – ID: `2020-06-01 22h51m`

- Chain Length: 6
- Structural Changes (compared to Table 5.3 & Table 5.1):
  - $w_{E_i \to E_{i+1}}$: 1 nS and 2 nS
  - N_I: 0
  - N_B: 0
- Neuron Parameter Changes (compared to Table 5.2):
  - V_{rest}: -70 mV and -65 mV
- Other Changes: None

The shape of the basin of attraction depends on the delay between excitation and inhibition. Since the delay is not adjustable on hardware we chose to implement a network without inhibition as an initial test. Furthermore, we exclude background activity to simplify the dynamics of the network. At first, we run NEST simulations of this network in order to have a comparison to the hardware results which follow in the next sections.

Excitatory background activity increases the mean value of the membrane potential. To account for the missing background input, we tested two configurations with different rest potentials V_{rest}. In addition, we scanned over three different weights between excitatory groups.

For each combination of reset potential and weight we performed one trial with ten draws each. The average runtime per trial increased to around 130 min[5].

Spiking Activity

The network with the base configuration (without inhibition and background activity) was already able to transport pulse packets along the chain, Fig. 5.5. Propagation for the weakest and most synchronous pulse packet did not change, Fig. 5.2. It still results in single pulse packets in all subsequent groups.

This holds not true for (9,3): as before more than one pulse packet is initiated in the first group but this time two of these survive to the last group. Neurons are excited again after their refractory period. Propagation looks stable from the third group onward.

A similar behavior can be observed for the broadest packet displayed in Fig. 5.5. Several packets are present in the first group; three of them reach the final group. So unlike the network with inhibition the broad input can cause a response in the last group.

[5] Again single thread on `Intel Xeon CPU E5-2643 v2`.

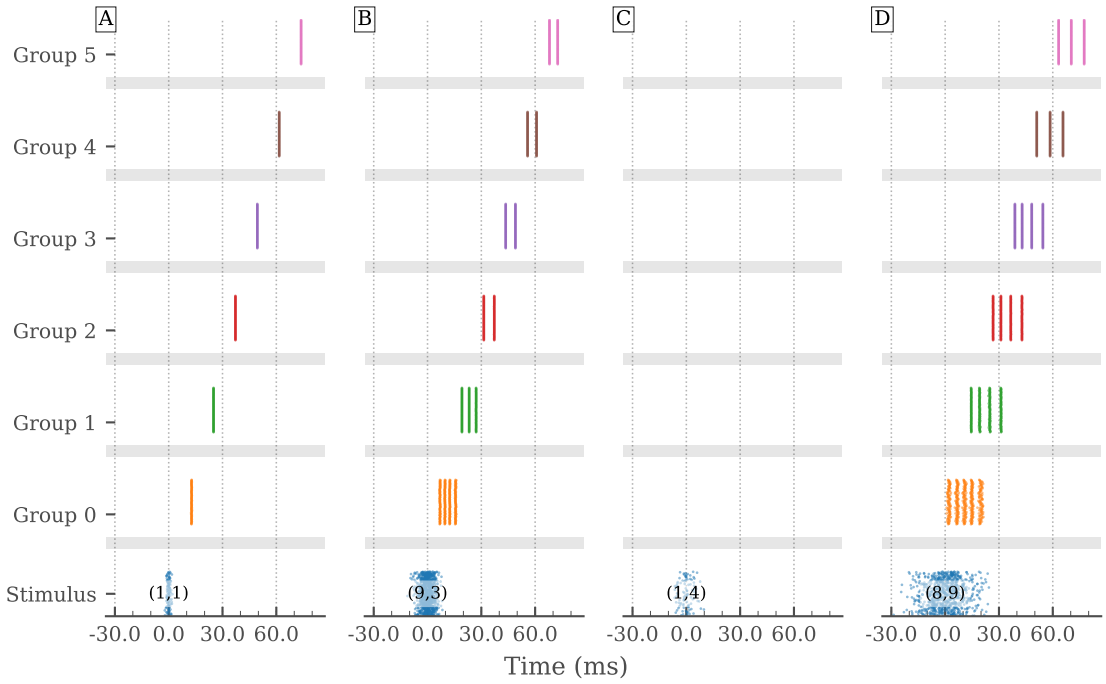


Figure 5.5: NEST without Inhibition and Background - Spike Trains – **(A)** Highly synchronous input pulses are still propagated along the chain and cause a single pulse packet in each group. **(B)** As before four pulse packets are invoked in the first group. The network is again able to reduce the number of pulse packets. Nevertheless, more then one packet is excited in the last group. **(C)** A relatively asynchronous and weak pulse still dies out. **(D)** In the network with inhibition the strong but asynchronous packet introduced only a weak and temporal broad activity in the first group. This time a number of pulse packets are triggered in the first group. Some of these packets survive to the last group. Simulation with base parameters, Section 5.1.1, but without inhibition and background activity.

Packet (1,4) is still not sufficiently synchronous and does not even cause any activity in the first group for this network configuration.

Phase Diagram

Figure 5.6 shows that the area for which propagation is possible increases compared to Fig. 5.3: packets with a temporal spread of $\sigma_{\text{in}} > 20\,\text{ms}$ can now be propagated to the last group. The filtering ability for synchronous input is highly reduced but is still present for low input strengths.

The response in the final group is not binary anymore but depends on input strength and standard deviation. For strong stimuli the output strength increases at first with temporal spread as the neurons are reactivated more often after being refractory in-

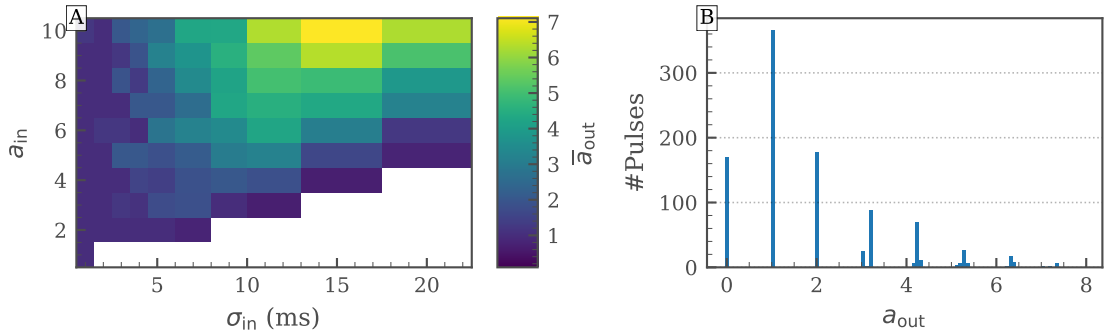


Figure 5.6: NEST without Inhibition and Background - Phase Diagram – (A) Responses in the last group averaged over all ten draws. Strengths smaller 0.1 are not printed. Pulse propagation is possible for a large range of input values. The response in the final group is not binary anymore but depends on the input. **(B)** The response strengths are near multiple of one; indicating that each response is made up of several pulse packets with strength 1. Our evaluation method seems to overestimate the output strength if the response consists of several pulse packets. Base parameters (Section 5.1.1) without inhibition and background.

creases. At a critical spread the output response decreases again in strength. The input spikes are now so far apart that the membrane potential has enough time to approach the rest potential such that a threshold crossing becomes less likely. For your given set of parameters the critical standard deviation is around 15 ms for inputs with $a_{in} > 7$. Furthermore, for fixed temporal spreads the output strength increases with input strength.

A strong and broad stimulus packet can cause several packets in the last group, see Fig. 5.5 (D). This is evident in the histogram, Fig. 5.6 (B): the output strengths are near multiple of 1. The deviations from exact multiple appear for values larger two and are due to the fact that we change from evaluation method 2 (direct calculation of standard deviation and strength) to a Gaussian fit (method 5), Appendix A.4. We overestimate the strength of the pulse packets by about 10 %.

Since most responses are made up of several pulse packets, the calculated standard deviation of the responses do no longer represent the temporal spread of a single packet. Therefore, we will not analyze the standard deviations for the given network.

Near the separatrix the phase diagram shows regions with $a_{out} < 1$. Here the propagation of the pulse once again depends on the exact distribution of the input spike times.

Networks with a higher synaptic strength between excitatory groups or a higher reset potential showed the same overall behavior with an even stronger response in the final group.

5.2.3 NEST without Inhibition and Background, with Parameter Variations

Experiment Parameter – ID: `2020-07-10 12h21m`

- Chain Length: 48
- Structural Changes (compared to Table 5.3 & Table 5.1):
  - $w_{E_i \to E_{i+1}}$: 1.4 nS and 1.6 nS
  - N_I: 0
  - N_B: 0
  - All delays are set to 1.5 ms with a deviation of 5 %, weights have a variation of 50 %
- Neuron Parameter Changes (compared to Table 5.2):
  - Parameter variations of 5 %
- Other Changes: None

The analog circuits on hardware can not be manufactured identically but only with small variations. As an result neuron and synapse properties time-independent inhomogeneities across the wafer. Neuron parameters are stored on floating gates which is an additional source of of variation which changes for every reconfiguration of the hardware.

The effects of these variations on a balanced network have been studied by Schwarzenböck [27]. Here we will assume that all neuron parameters as well as the synaptic delays follow a Gaussian distribution with a standard deviation of 5 %, Section 5.1.1. The synaptic weights are drawn from a Gaussian distribution with a standard deviation of 50 %. As discussed previously, the synaptic delay can not be adjusted on hardware. Therefore, a value of 1.5 ms is taken as the mean delay for all synaptic connections.

We will now perform five trials; for each trail we redraw the neuron and synapse parameters from Gaussian distributions.

Spiking Activity

The successful transmission of pulse packets is still possible, Fig. 5.7. Due to the minimized synaptic delay, transmission along the chain is faster. Pulse packets are no longer as sharp as before and show a clear temporal spread. Weak, asynchronous packets, such as (1,4), cause a relatively broad response in the first group. This response is synchronized along the chain until it is comparable to the response of (1,1) in the final group, compare subfigures (A) and (C).

For strong, asynchronous packets the response in the first group is strong, subfigures (B) and (D). There seems to be some kind of pulse packet structure similar to the

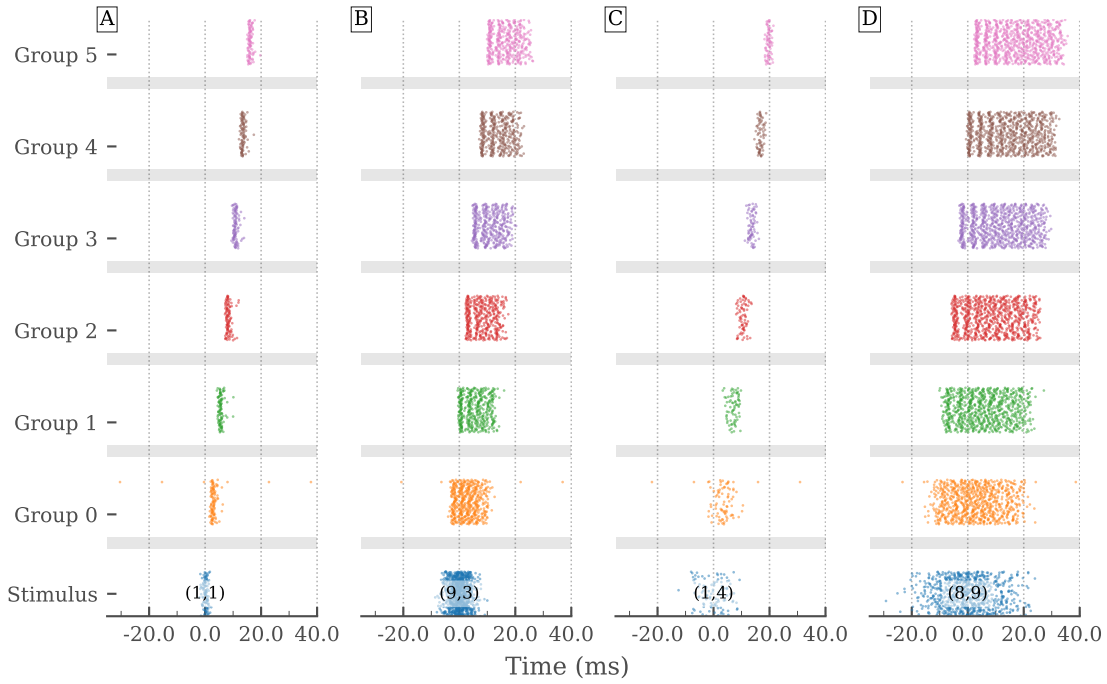


Figure 5.7: NEST without Inhibition and Background, with Parameter Variations - Spike Trains – Simulation with base parameters (Section 5.1.1) and $w_{E_i \to E_{i+1}} = 1.6\,\text{nS}$ but without inhibition and background. Neuron and synapse parameters are drawn from Gaussian distributions. **(A)** Highly synchronous input pulses are still propagated along the chain and cause a single pulse packet in each group. The temporal spread is higher than before **(B)** An asynchronous pulse packet initiates a strong response in the first group which seems to show some kind of packet structure. This is similar to the simulation without variations, Fig. 5.5. **(C)** A relatively asynchronous and weak pulse is synchronized along the chain. The response in the final group looks similar to (A). **(D)** The response in the first group looks random. In later groups some kind of pulse structure becomes visible.

simulation without variations, Fig. 5.5. Even though the structure appears to become more pronounced in later groups, the single packets are hard to tell apart because of their larger temporal spread. The temporal spread of the final response seems to be comparable to the initial spread of the stimulus in case the input strength is strong.

The rest potential V_{rest} of one neuron in group 0 is higher than the spiking threshold V_{thres}. As a consequence, the neuron spikes periodically; this is visible in the subfigures (A) to (C).

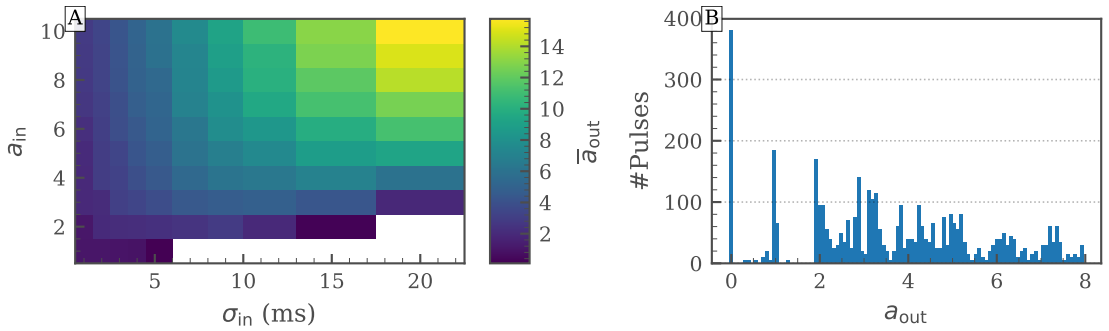


Figure 5.8: NEST without Inhibition and Background, with Parameter Variations - Phase Diagram – (A) Responses in the last group averaged over all five trials with ten draws each. Strengths smaller 0.1 are not printed. Pulse propagation is still possible for a large range of input values. The response in the final group depends on the input stimuli. **(B)** The output strength seems to peak near multiple of one but the peaks are not as sharp as for the simulation without variations, Fig. 5.6. Base parameters (Section 5.1.1), $w_{E_i \to E_{i+1}} = 1.6\,\text{nS}$, without inhibition and background. Neuron and synapse parameters are drawn from Gaussian distributions.

Phase Diagram

The response in the final group still depends on the input stimuli, Fig. 5.8 (A). As we increased the weight in comparison to the previous plot, Fig. 5.6, the area of successful transmission increases and the overall strength is higher.

Subfigure (B) shows that the response strength in the final group still peaks near multiple of 1 and therefore still seem to consist of multiple pulse packets with a strength of 1. However, the peaks are less sharp, compared to Fig. 5.6, and a whole range of different output strengths are present in the histogram. This is in agreement with the displayed spike trains in Fig. 5.7 which looked like a weak packet-like structure is superimposed by random activity.

5.2.4 BSS without Inhibition and Background - Default Mapping

Experiment Parameter – ID: `2020-06-05 15h25m`

- Wafer: 24
- Chain Length: 6
- Structural Changes (compared to Table 5.3 & Table 5.1):
  - $w_{E_i \to E_{i+1}}$: 1 nS to 2 nS in 0.2 nS-steps

  - N_I: 0
  - N_B: 0
- Neuron Parameter Changes (compared to Table 5.2):
  - V_{rest}: $-70\,\text{mV}$ and $-65\,\text{mV}$
- Other Changes: None

We start with a simplified network which does not include inhibition and background activity. This allows us to analyze hardware induced distortions more easily.

Floating gates are used to store neuron parameters, Section 2.1.1. The exact value stored in these gates varies slightly with each write. As a consequence the neuron parameters change from trial to trial. To account for this variation we run five trials with ten draws of each input stimuli. The runtime per trial was on average 16 s[6]. In this time the configuration of the hardware is included. The bare emulation time should be around 100 ms as the simulation spans over 700 000 ms in biological time. The emulation on hardware was about 500 times faster than our non-optimized NEST simulation.

Mapping

The mapping result is illustrated in Fig. 5.9. Neurons are placed from left to right. As each HICANN provides space for a little more than 100[7] neurons five and a half chips are needed to place all 600 excitatory neurons. As a consequence different populations may share the same HICANN.

The external stimulus input is placed near the neurons of the first group since no other population is connected to the stimulus. Placing all external input on a single HICANN might not be optimal since the input rate of external events is limited, compare Section 2.1.5.

In addition to the rate limitations from FPGA to HICANN, the input rate of external events in the on-wafer network is limited by the input rate of the sending repeaters. The frame of a six bit address is 8 ns[8] long. After each sent address the repeater waits one frame before sending the next address. Therefore, about every 0.16 ms (biological time) a spike can be injected in the on-wafer network. All eight sending repeaters in total can relay about 50 spikes each millisecond.

[6] This does not include mapping from the biological network to the hardware which needs to be done for the first trial (and is loaded afterwards for the remaining trials) and takes about one minute. We did also not perform a verification of the hardware configuration. This would take about 15 s in addition, cf. Section 5.2.5.

[7] The exact number depends on the number of blacklisted DenMem circuits and the neuron size. We use four DenMem circuits per neuron, such that a fully functional chip could implement 128 neurons.

[8] For a PLL frequency of 125 MHz.

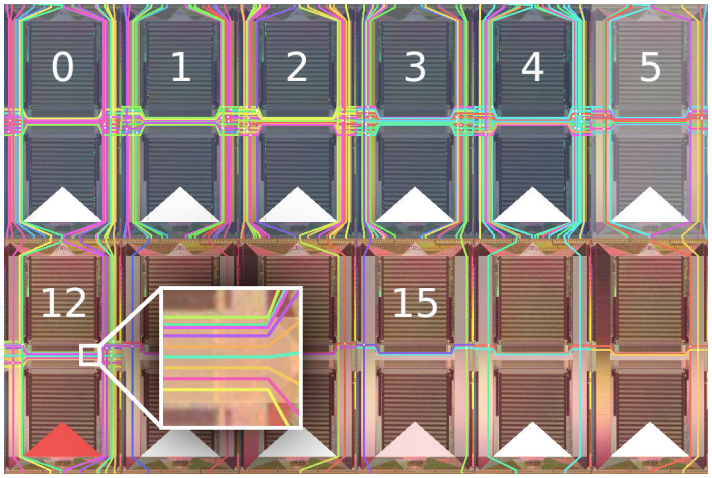


Figure 5.9: Result of the Default Mapping – Mapping result for the default marocco placement. Neurons are marked by a blue color. The higher the opacity the more neurons are placed on the HICANN. External inputs are resembled by the red triangle in the lower part of the chip. The opacity once more encodes their amount. Neurons are placed from left to right. Each HICANN hosts a little more than 100 neurons. On HICANN 12 all 100 stimulus neurons are placed. Inter stimulus inhibition is injected on chip 15. Colored lines represent the connections between sending repeaters and synapse drivers. Inset: eight buses are used to inject the stimulus (one bus is used by the inter-stimulus inhibition).

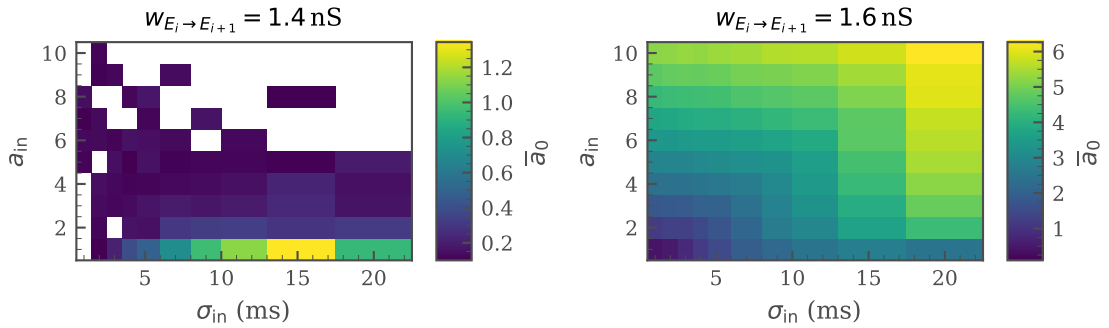


Figure 5.10: BSS-1 without Inhibition and Background - Default Mapping - Phase Diagram of the First Group – Values are averaged over all five trials and ten draws. Strengths smaller 0.1 are not displayed. Note that the z-axis is not aligned. **(A)** Only for small input strengths and an intermediate temporal spread notable activity can be observed in the first group. **(B)** For a slightly higher weight the activity in the first group is considerably higher all inputs packets cause a response. All parameters are set as described in Section 5.1.1 but without inhibition and background input. Due to a bug in the code, the spikes were not reset after each trial such that the strength presented here is higher than the actual strength.

If we assume the spikes of an input packet (1,1) are evenly distributed over $2\sigma = 2\,\mathrm{ms}$ this rate would be just sufficient to inject the spikes of 100 stimulus neurons. We test up to ten times higher input strengths with not evenly but normal distributed spike times. The high density of spikes can lead to a shift or even loss of spikes. This could already be observed in the FPGA were spikes were shifted since pulse groups were full, compare Section 2.1.5 and Fig. A.7.

Inter-stimulus inhibition is shared between all neurons and is therefore injected near the center of all populations.

Due to the simple feed forward structure the synapse loss is below 0.5 % for each projection.

Activity

Regardless of the reset potential there was no activity in the final group for the lowest three weights: activity died out after the first group.

With decreasing weight the response in the first group changes abruptly, Fig. 5.10. While we observed a strong activity in the first group for a weight of $w_{E_i \to E_{i+1}} = 1.6\,\mathrm{nS}$, a slightly lower weight leads only to a weak response. We do not expect such a drastic change for a small change in weight. Furthermore, the strength of the response in the first group should not decrease with input strength.

During experiment execution a lot of spikes were shifted while filling the pulse groups, compare Section 2.1.5 and Fig. A.7. This unexpected behavior might be connected to

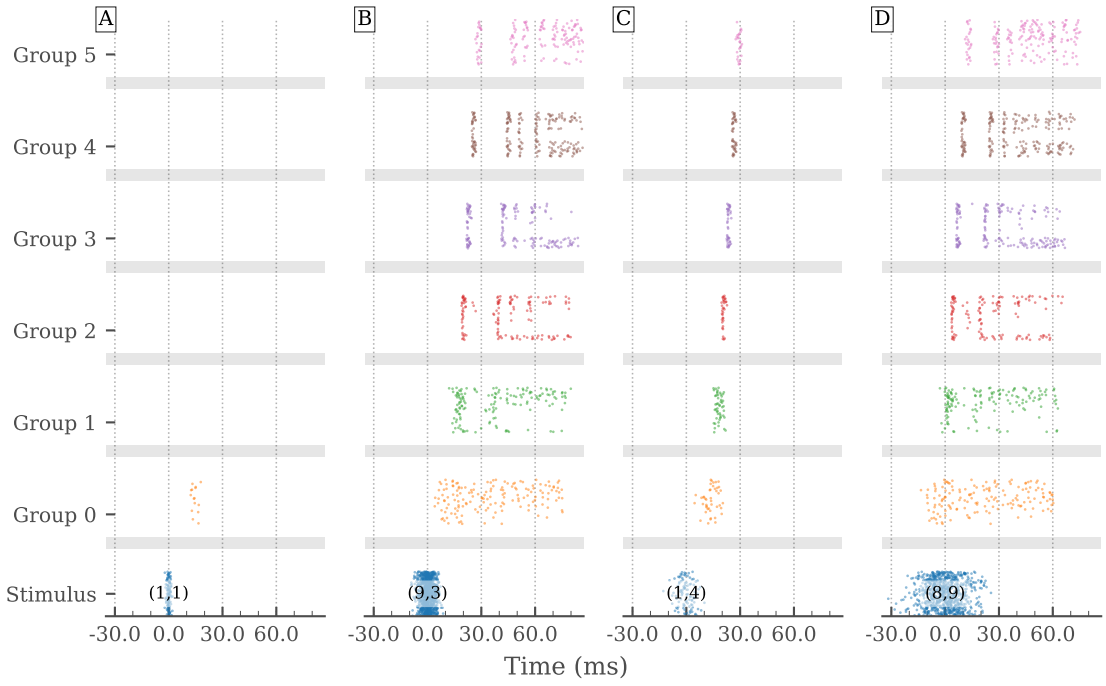


Figure 5.11: BSS-1 without Inhibition and Background - Default Mapping - Spike Trains – Due to the high number of shifted spikes the spike trains have to be interpreted with care. Vertical gaps in the spike response are caused by a limited readout, Footnote 9. **(A)** The propagation of a weak but highly synchronous input dies out after the first group. **(B)** A stronger and slightly more asynchronous packet introduces a high activity in the first group which is mediated to the last group. **(C)** A more asynchronous pulse, compared to (A), causes a response in the last group. This is not expected and is caused by hardware effects, see text. **(D)** Similar to (B) an asynchronous but sufficiently strong input is propagated to the last group. Emulation with base parameters (Section 5.1.1), no inhibition, no background and $w_{E_i \to E_{i+1}} = 1.6\,\mathrm{nS}$.

these shifts and we will postpone an in-depth analysis of the network dynamics to the next section where we will place the stimulus input manually. Here we will just have a look at some example spike trains for the first weight for which successful propagation is possible, Fig. 5.11.

While packet (1,1) dies after the first group, the broader pulse packet (1,4) leads to a single packet in the final group. This is not expected for a synfire chain which shows higher propagation probabilities for more synchronous inputs [18, 20, 22]. We assume that several spikes are dropped due to high input rate such that the actual input strength of (1,1) is considerably lower than $a_{\mathrm{in}} = 1$. Furthermore, measurements indicated that the synaptic input current saturates for too strong/synchronous input, Appendix A.5.

Later NEST simulations showed a similar effect, Fig. A.5, which could be attributed to the extinction of pulse packets along the chain, Fig. A.6. Therefore, the observation in Fig. 5.11 might indicate that the strength between excitatory groups is too low and all packets would eventually die out; this effect might be enhanced by the characteristics of the hardware.

For (1,4) the packet is synchronized along the chain and shows a low standard deviation of about one millisecond in the last group. This synchronization is expected for a synfire chain and was observed in NEST simulations, Section 5.2.2.

Similar to the NEST simulation broad, strong input stimuli cause a strong response in the first group. While the simulation showed multiple, distinctive pulse packets the emulation on BSS-1 shows an enhanced activity over a broad time frame. The readout of spikes from the system is limited by the bandwidth of the link between the chip and the FPGA. This may result that spike times are slightly shifted or lost during readout. The loss of spikes is visible as horizontal stripes with no activity[9], as in group 1 of packet (9,3).

The bandwidth of the on-wafer network is considerably higher such that most of the spikes, which are lost during readout, are forwarded on the wafer [11].

In later groups there seems to be a pulse packet response followed by no activity for about 20 ms and some more pulse packets which fade out until the activity looks random. The large pause after the first packet might be caused by the shifted input spikes.

5.2.5 BSS without Inhibition and Backkround

Experiment Parameter – ID: `2020-06-14 14h27m`

- Wafer : 24
- Chain Length: 6
- Structural Changes (compared to Table 5.3 & Table 5.1):
  - $w_{E_i \to E_{i+1}}$: 0.8 nS to 1.6 nS in 0.2 nS-steps
  - N_I: 0
  - N_B: 0
- Neuron Parameter Changes (compared to Table 5.2): None

[9] The readout prioritizes DenMem on the left side of the HICANN. As a consequence neurons which are placed later on the HICANN have priority over neurons which were placed before. For mapped network a neuron population may be spread over two chips. The first few neurons have priority on the first HICANN. Then placement on a new chip is started and spikes of these low priority neurons are dropped. As soon as the DenMem is far enough to the right, spikes are recorded again. This results in horizontal patches of no activity within a single population.

- Other Changes:
  - External stimulus input distributed over 24 HICANNs on six FPGAs

The default placement algorithm of marocco placed all 100 stimulus neurons on a single HICANN. This resulted in large shifts of input spike times, Fig. A.7, and probably a significant spike loss for dense pulse packets. In this set of experiments we will place the stimulus on several different chips to reduce spike loss and shifting of spike times.

We once again performed five trials with ten draws each. This time we included the verification of the hardware configuration such that the average runtime increased to 31 s per trial which is till about 260 times faster than the simulation of an identical network in NEST.

Mapping

We distribute the stimulus neurons over 24 different chips which are located on six different reticles, Fig. 5.12. At the maximum tested input strength of 10 each chip has to forward about 40 spikes per stimulus input. Therefore, we still expect some of the input spike to be dropped or slightly delayed due to the FIFOs in the FPGA, see Section 2.1.5, if the inputs are densely packed. In our setup each FPGA handles four HICANNs and therefore 160 spikes. The number of FPGAs was chosen such that all spikes fit in a single pulse group and spikes are not shifted due to the limited group size.

Figure 5.12 displays the mapping result. The stimulus input was placed on the HICANNs in the lower left corner. From here marocco routes the inputs horizontally to a HICANN which is in the same column as the target HICANN. After that the input is forwarded straight upwards to the target chip.

Since we only placed the external input manually, the remaining mapping stayed the same as in Fig. 5.9.

Due to the increased number of entry points, more L1 buses and synapse drivers are needed to rely the external input to the first group. This resulted in a synapse loss of about 5 % for connections between stimulus and the first excitatory group. Because of the local feed forward structure, other connections were not affected and the synapse loss stayed the same as in case of the manual mapping.

The 255 repeater in the network could be locked in the first try.

Spiking Activity

The propagation of incoming pulse packets to the last group is possible for all weights. Depending on the weight the network shows other filtering properties and synchronization properties, Fig. 5.13.

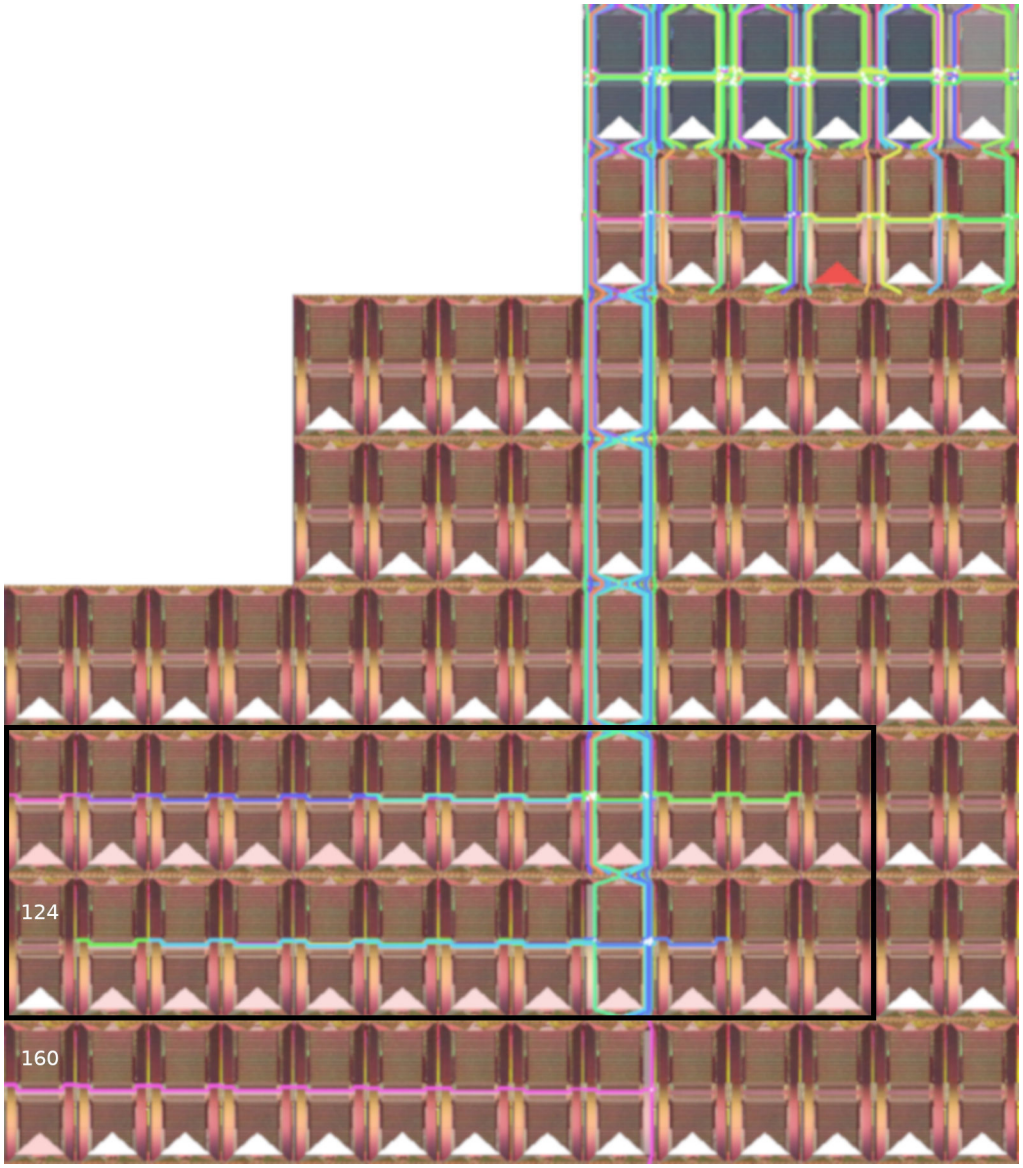


Figure 5.12: Short Chain - Manual Mapping – The number of neurons on each chip is encoded by the opacity of the blue area on the synapse arrays. External inputs are marked with a red triangle in the lower part of a chip. The higher the number of external input neurons, the higher the opacity. Routes between sending repeaters and synapse driver are marked by colored lines. Input stimuli were manually placed on HICANNs in the lower left part of the figure (marked by a black rectangle). HICANN 124 threw errors during configuration such that HICANN 160 was used as a replacement. Marocco uses a single sending repeater for all input on a single chip. The inputs are at first forwarded horizontally and then routed downwards to HICANN 0.

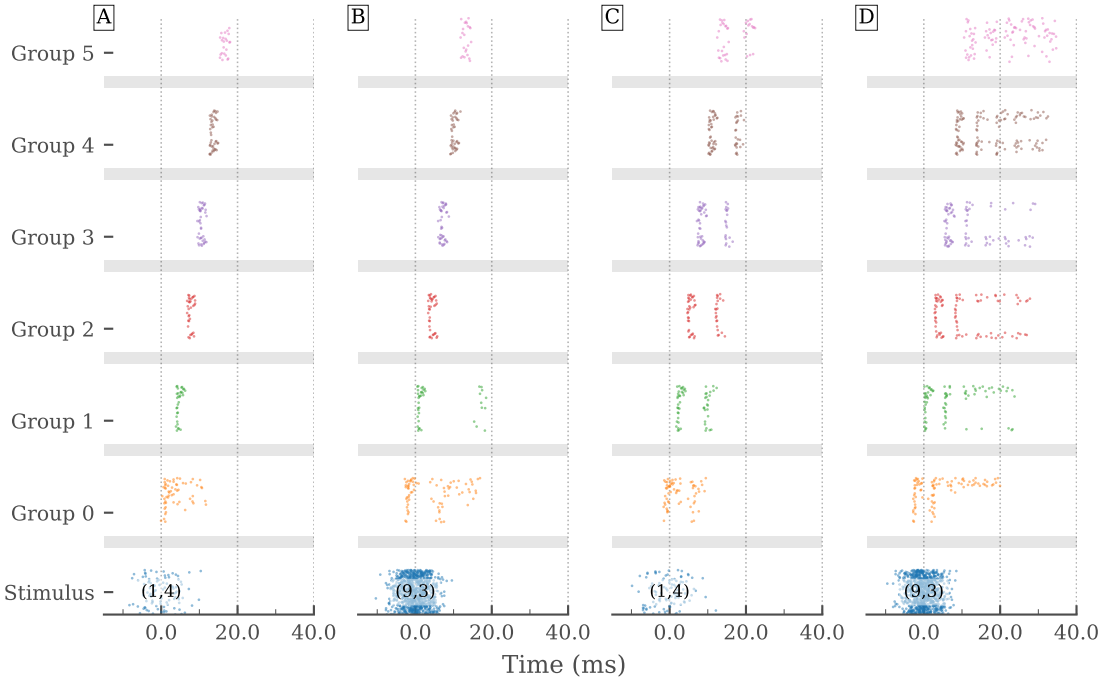


Figure 5.13: BSS-1 without Inhibition and Background - Spike Trains – Emulation without inhibition and background activity, base parameters (Section 5.1.1) and $w_{E_i \to E_{i+1}} = 1.4\,\text{nS}$ (A,B), $w_{E_i \to E_{i+1}} = 1.6\,\text{nS}$ (C,D). **(A)** A weak and relatively asynchronous packet can propagate along the chain. The response in the first group is rather high and broad. It synchronizes along the chain such that a single sharp packet arrives at the last link. **(B)** For a stronger pulse packet the activity in the first group is stronger and broader. The response in the second group consists of two pulse packets one of which survives till the last group. **(C)** For a higher weight between excitatory populations the response in the first group is stronger and two packets are able to travel along the chain. **(D)** A similar but stronger response is visible for a input with more spikes. The activity shows vertical areas of missing spikes due to the limited readout rate, compare Footnote 9.

For $w_{E_i \to E_{i+1}} < 1.6\,\text{nS}$ asynchronous input causes a pulse packet followed by some broad activity in the first group, subfigures (A) and (B). While the packet is able to propagate along the chain, the temporal broad response dies out after a few groups.

For higher weights we see a similar response in the first group, (C) and (D). An initial pulse packet is followed by some broader activity. Due to the higher weight the rather asynchronous activity does not die out but initiates a pulse packet in the following group.

(1,4) causes two pulse packets with a distance of around 10 ms in the second group. As in the NEST simulations multiple pulse packets reach the final group, Section 5.2.2.

For an even stronger input of (9,3) the activity in the first group is even stronger. Two pulse packets with a small delay between them are distinguishable before we see the effect of the limited readout rate and some spikes are lost (see Footnote 9 on Page 65). The distribution of the spikes which could still be readout looks random and no pulse packets can be distinguished. In the following groups the response seems to consist of multiple pulse packets but they are hard to tell apart because of the missing spikes. The response in the final group does not show any pulse packet structure but looks like increased, random spiking activity for a fixed period of time.

This random looking activity combined with some pulse like structure is in agreement with our NEST simulations, Fig. 5.7. However, the overall spiking activity seems to be considerably higher in the NEST simulation.

In the first group all displayed spike trains show an initial pulse packet response followed by some other activity. The delay between this first pulse packet and the additional activity depends both on the input stimulus as well as on the weight. For more dense packets (compare (9,3) and (1,4)) the delay between both responses is shorter since a second threshold crossing is initiated more quickly. Similarly, the delay decreases with a higher weight as the number of incoming spikes needed to cause an output spike decreases. Furthermore, the delay was quite high in case of the default placement, Fig. 5.11. We assume that this was caused by the shifted spike times.

A look at the delay between packets in adjacent groups revels that an increasing weight leads to a faster propagation, Table 5.5. Such a dependency of the group to group delay was not evident in the NEST simulations (without variations), Section 5.2.1 since the temporal spread of the pulse packets was with around 0.1 ms considerably smaller.

The standard deviation of the delays measured for lower weights are higher since the effect of parameter variations is enhanced. For variations in the distance of reset V_{rest} and threshold potential V_{thres} for example, it is more likely that the number of spikes needed to cause an outgoing spike stays the same for high weights while the number of required spikes changes more easily for low weights (the post synaptic potential (PSP) induced by a lower weight is lower and more spikes are needed to cross the threshold). This can result in different group to group delays which depend on the exact neuron parameters.

For the highest weight the increased standard deviation is caused by the low number of samples. We only consider relatively synchronous spikes and try to avoid responses

Table 5.5: BSS-1 without Inhibition and Background - Propagation Delay – Group to group delay of a pulse packet which is transmitted along the chain. Only packets in adjacent groups with $0.1 < a < 1.5$ and $\sigma < 3\,\text{ms}$ were considered. The delay was then averaged over all trials, draws and stimuli.

$w_{E_i \to E_{i+1}}$ (nS)	Delay (ms)	Samples
0.8	3.4(3)	19 203
1.0	3.2(2)	22 390
1.2	3.0(2)	21 565
1.4	3.0(1)	21 701
1.6	2.8(3)	3370

with multiple packets in them. For the highest weight we have many responses which are strong and broad, cf. Fig. 5.13, resulting in the low number of samples.

Phase Diagram

The spike trains displayed in Fig. 5.13 already showed a synchronization of the pulse packets along the chain for small weights $w_{E_i \to E_{i+i}}$ only a single pulse packet reached the final group. This is also visible in Fig. 5.14: the activity in the final group is almost binary for low weights. In the basin of attraction the strength is with 0.2 rather low and could lead to an extinction of the pulse packet in later groups; emulations with longer chains are needed to determine if the packets can survive, Section 5.3.1. The low weight might be an effect of the parameter variations present on hardware. Emulations of longer chains show that the output strength changes from group to group and can be considerably higher than 0.2, Fig. 5.25.

With increasing weight the area for which the input pulses are successfully propagated increases. The network with the lowest weight between excitatory neurons shows good filtering properties and relays only packets which are rather synchronous. This changes for higher weight where also asynchronous packets are transmitted along the chain.

For the highest tested weight all pulse packets reach the final group. The activity is no longer binary but depends on the initial stimulus. Broad and strong input packets cause a higher activity in the last group. This is in agreement with the NEST simulations.

Figure 5.15 shows the comparison of the BSS-1 emulation with the NEST simulation which included parameter variations. We observed in the previous plot, Fig. 5.14, that the output strength is only around 0.2 and therefore only about a fifth of the value which one would expect from simulations. After scaling the z-axis by this factor, the phase diagrams, subfigure (A) and (C), show the same overall behavior. While in simulation weak, asynchronous packets are not transmitted along the chain, all packets reach the

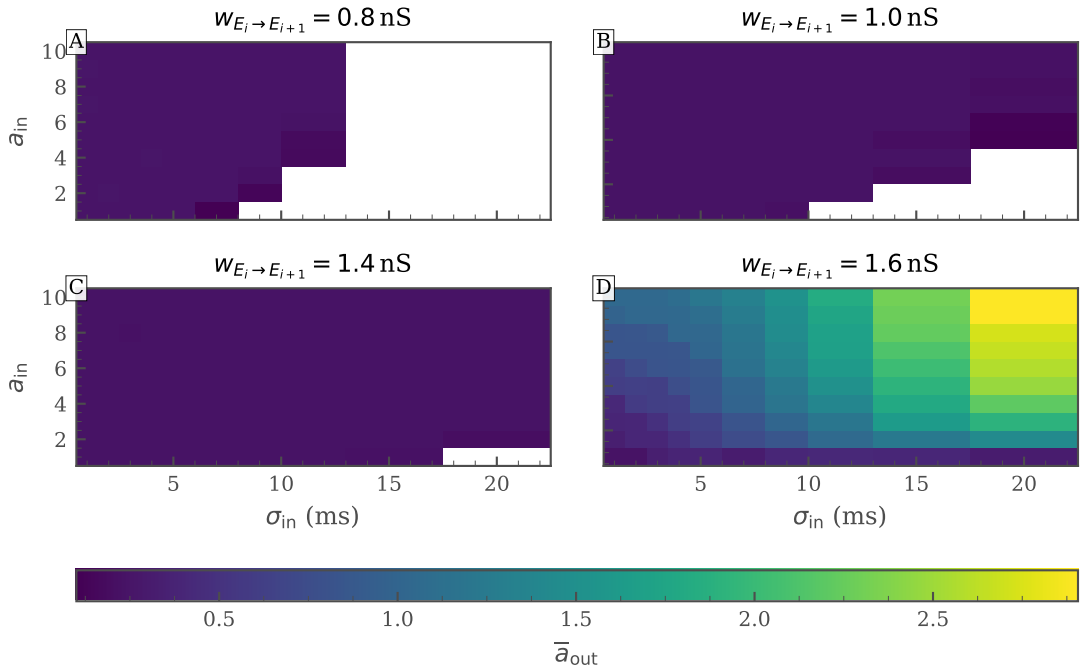


Figure 5.14: BSS-1 without Inhibition and Background - Phase Diagram – Phase diagram of the strength in the final group. Values are averaged over all trials and draws. $a_{\mathrm{out}} < 0.1$ are not displayed. Emulation with base parameter (Section 5.1.1) without inhibition and background activity. **(A-C)** For low weights the response in the final group is binary. The basin of attraction increases with increasing weight. **(D)** All pulses are propagated to the last group. The strength of the response depends on the properties of the input packet.

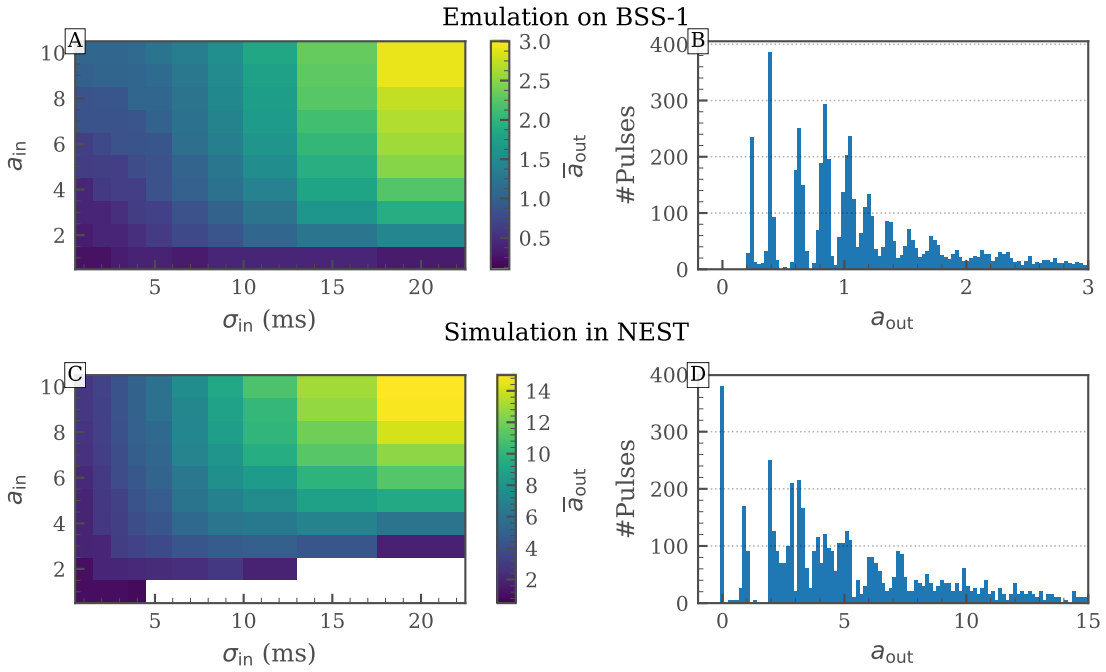


Figure 5.15: BSS without Inhibition and Background - Strength Distribution in the Final Group – Emulation of the simplified network (no inhibition and background activity) with base parameters, Section 5.1.1, and a weight of $w_{E_i \to E_{i+1}} = 1.6\,\text{nS}$. For the phase diagram the values are averaged over all trials and draws. **(A)** The output strength depends on the properties of the input stimulus. Several neurons are active more than once. **(B)** Distribution of the output strength in the final group. The number of pulses peaks at strengths near multiple of 0.2. The distance between spikes decreases with increasing strength. Furthermore, the peaks are less sharp for higher strengths. **(C,D)** Results of NEST simulation with neuron parameter, weight and delay variations, same as in Fig. 5.8; the axis/color bar is scaled by a factor of five compared to the plots in (A,B).

final group when emulated on BSS-1.

The distribution of the output strength, Fig. 5.15 (B) and (D), peaks at certain strengths. These peaks are more pronounced in the hardware emulation than in simulation. In both cases the peaks are not as sharp as for the NEST simulation without variations, Fig. 5.6, and broaden for larger strengths.

For the hardware emulation, the distribution indicates that responses in the final group are made up of several packets with a strength of around 0.2. In contrast, Fig. A.8 shows that the number of active neurons increases to the top right of the phase space and exceeds 20 active neurons. So it might be a combination of packets stronger than $a = 0.2$ and additional packets with a strength of 0.2 which are caused by the highly active neurons which did also spike for lower weights.

For low weights evaluation methods 2 and 5 were used to determine the properties of strong ($a_{\text{out}} > 0.1$) pulses in the last group, see Appendix A.4. Broad and strong activity was observed for the highest weight such that a fit to the histogram of the spike times was performed in some cases (method 0). Weak pulses were in all cases detected by methods 1 and 2.

Evolution Along the Chain

For small weights $w_{E_i \to E_{i+1}}$ we observed a binary response in the final group. This is a sign that the pulse packets synchronize along the chain.

In order to investigate this behavior we plot the output strength and temporal spread averaged over all trials, draws and stimuli which resulted in a response in the last group ($a_{\text{out}} > 0.1$), Fig. 5.16.

In the first group the mean of the strength increases at first with the weight, before it approaches a fixed value. In group 1 the difference of the mean values between the weights is small and vanishes completely in group 2. The value range of the strength (shaded areas) also shrinks from group to group but still not all responses share the same strength in the final group.

Overall, the strength of the response decreases from the first to the second group. Then it starts to increase slightly till group 4 and drops abruptly in the last group. This gives the impression that the pulse packets do not gradually die out but that the excitability of the neurons in the last group is reduced[10]. Longer chain length have to be emulated to test whether this low strength is still able to ensure the propagation of the pulse packets, Section 5.3.1.

The standard deviation shows a high variability in the first few groups. This is due to the distribution of the spike times which is no longer Gaussian but may consist of several pulse packets or a high rate over a fixed period of time, see Fig. 5.13. As a result

[10] In agreement with this assumption a similar drop was also evident in the evaluation of the strength of the highest weight $w_{E_i \to E_{i+1}} = 1.6\,\text{nS}$.

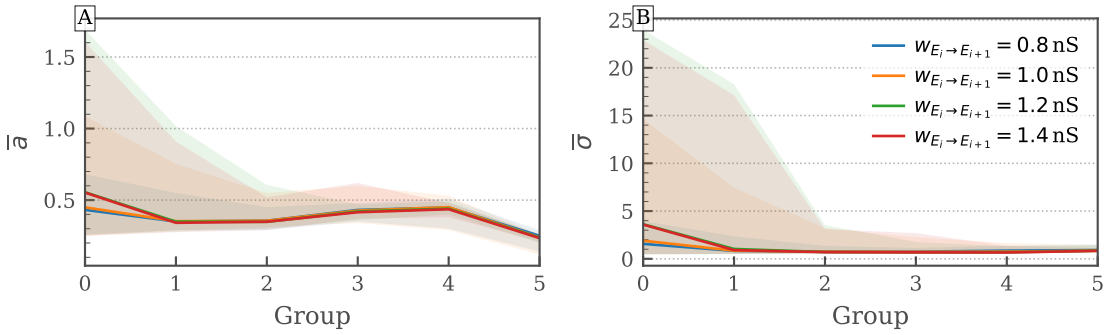


Figure 5.16: BSS-1 without Inhibition and Background - Evolution Along the Chain – The solid lines represent the mean of all trajectories which end in a strength $a_{\text{out}} > 0.1$. The shaded area around these lines represent the maximum/minimum response in each group (with $a_{\text{out}} > 0.1$). **(A)** All weights share the same mean output strength from the second group onward. The mean strength stays almost constant after the second group but drops in the final group. The deviation between the trajectories is the highest in the first few groups. **(B)** The standard deviation of the responses in each group show a similar trend as the strength. It converges towards a low value of about 1 ms.

a Gaussian does no longer describe the spike response appropriately and the determined standard deviation has to be interpreted with care.

Nevertheless, we can observe the synchronization along the chain: the mean values for the different weights approach each other from group to group and reach a low value of about one millisecond from the third group onward. The temporal spread is, as expected, higher than in the NEST simulations. This is caused by the variation in neuron parameters, synaptic strength and synaptic delay which are present on hardware. Another source of broadening can be the limited readout rate of spike events [24]. Other effects of the limited readout rate were already visible in the spike trains (Fig. 5.13).

Activity in the First Group

The activity in the first group gives some insight in hardware effects, Fig. 5.17.

In the beginning the number of active neurons increases with temporal spread, subfigure (B). This is not expected and could be caused by spike loss. If we assume the spikes of the input population are evenly distributed over 2σ, spikes for the most dense input, (1,1), would be separated by 0.02 ms. A single HICANN would have to emit a spike every 0.48 ms and a FPGA would have to forward a spike every 0.12 ms (as we distribute the input over 24 HICANNs on six FPGAs). While a HICANN can handle this rate, the FPGA is too slow (Section 2.1.5).

For $\sigma_{\text{in}} < 5$ ms the number of active neurons is almost independent of the weight. This indicates that the synaptic input current behaves nonlinear. We assume that the

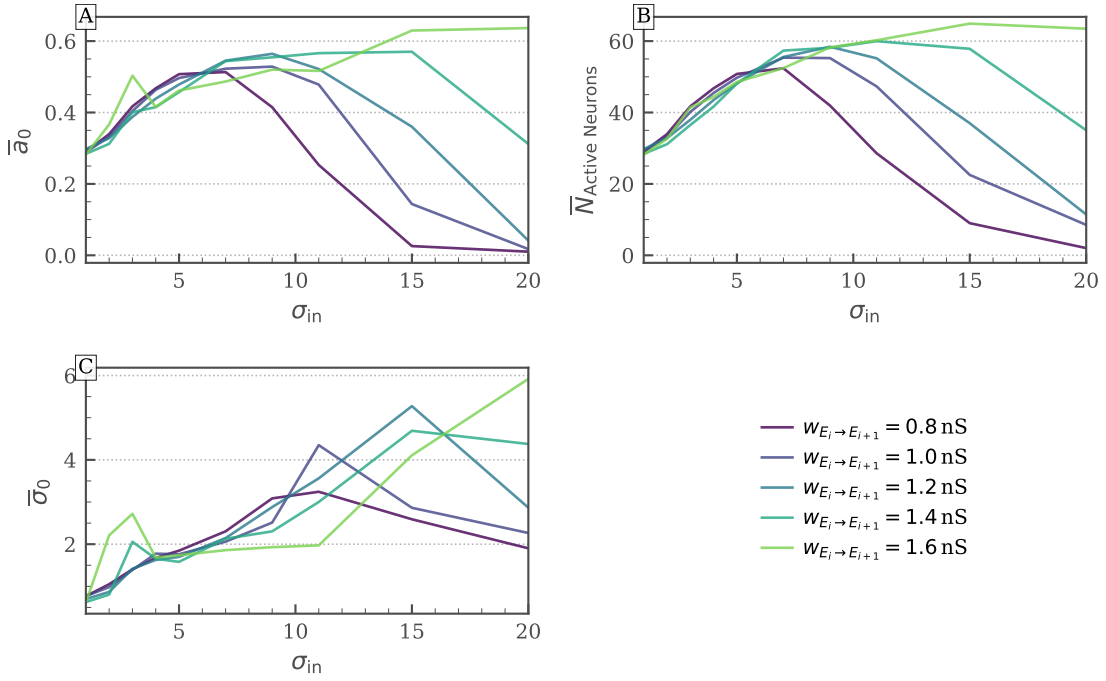


Figure 5.17: BSS-1 without Inhibition and Background - First Group Activity – Values are averaged over all trials and draws. Network with base configuration (Section 5.1.1) but without inhibition and background. The curves show the response to an input strength of $a_{\text{in}} = 1$. **(A)** The strength of the packets in the first group is similar for small standard deviations for the three lowest weights. The response is higher for the two highest weights. The strength rises till a critical standard deviation and then drops. From this point onward it is lower than the strength of a higher weight. **(B)** The number of active neurons (at least one spike) follows a similar trend as the strength. **(C)** The temporal spread of the responses in the first group increase at first with the standard deviation of the input. The standard deviation for the two lowest weights starts to drop again as the number of spikes per packet becomes quite low for an input with a high standard deviation. Note that the lines of the maximum weight have to be interpret with caution as they are affected by the limited readout speed, compare Fig. 5.13 and Footnote 9.

synaptic conductance saturates if spikes are to dense or their weight is too strong; which in turn leads to a saturation in the synaptic current. Appendix A.5 covers this nonlinear behavior in some more detail.

At a critical standard deviation (dependent on the weight) of the input stimuli the number of active neurons starts to drop. Since the input spikes are further apart for large standard deviations the membrane potential has more time to relax towards the rest potential V_{rest} and incoming spikes are less likely to induce a threshold crossing. For the highest weight this critical value might just be reached at 15 ms and a slight decrease can be observed for higher standard deviations.

For the three lowest weights the response is similar to the number of active neurons and most active neurons only spike once, subfigure (A). The curve for the higher weights has to be interpreted with care as some spikes are lost due to the limited readout rate, compare Fig. 5.13 (D). In addition, the evaluation method is changed at about 3 ms which results in a visible peak.

The determined output strength is up to 26 % smaller than the number of spikes per neuron, compare Fig. A.9. This high difference between the number of spikes per neuron and determined output strength can be attributed to evaluation method 5, Appendix A.4. If the responses consist of more than one packet and some packets are affected by spike loss, method 5 only detects the strongest packet and disregards the other packets. Since we are more interested in qualifying single packets and strong responses are often affected by spike loss, we do not change our evaluation method but keep in mind that we have to interpret the results with care if the strength of the response is strong.

Figure 5.13 showed that the response in the first group is made up of an initial pulse packet response followed by some more activity. The activity can not be described by a Gaussian and can only give an estimate of the temporal spread of the present spikes. The standard deviation of the response in the first group increases with the standard deviation, subfigure (C). It starts to drop when the output strength becomes small.

5.2.6 BSS with Inhibition, without Background

Experiment Parameter – ID: `2020-06-14 16h06m`

- Wafer : 24
- Chain Length: 6
- Structural Changes (compared to Table 5.3 & Table 5.1):
  - $w_{E_i \to E_{i+1}}$: 0.8 nS to 1.6 nS in 0.2 nS-steps
  - $g_{E_i \to I_{i+1}}$: 3.5 and 5

  - $g_{I_i \to E_i}$: 2 and 5
  - N_B: 0
- Neuron Parameter Changes (compared to Table 5.2):
  - $E_{\text{rev-exc}}$: $-85\,\text{mV}$ and $-75\,\text{mV}$
- Other Changes:
  - External stimulus input distributed over 24 HICANNs and six FPGAs

FFI can reduce the susceptibility of a synfire chain to asynchronous input, Section 2.2.2. This allows to embedded synfire chains in biologically realistic, balanced networks [21, 22].

For the BSS-1 system we already saw that the response in the last group consists of a single pulse packet if the weights between excitatory connections are low. In case of the lowest weight the network already showed good filtering properties. In this experiment we want to use FFI to further improve the filtering for synchronous input.

Mapping

Compared to the previous experiment we now also have to place an inhibitory population of 25 neurons in each group. Furthermore, these neurons connect to the excitatory population in the previous group and to the excitatory neurons in the same group; this results in 4000 additional synaptic connections in each group.

The synapse loss for connections which were present in the previous experiment stayed basically the same. Only derivations in the sub percent range are induced since the exact placement on hardware differs now slightly[11]. From the excitatory population outgoing connections have a low synapse loss of less than 0.5 %.

The connection from the stimulus to the inhibitory population of the first group shows the same loss as the connection to the excitatory population, about 5 %. Connections within a single group show the highest synapse loss. Here the loss ranges from 4 % to 12 % and is not homogeneously distributed between different groups. This can be accounted to the fact that a single group is spaced over several HICANNs; which could also result in a different amount of incoming inhibitory synapses for neurons within the same excitatory population.

As the number of connections increased the number of used repeaters increased as well: 427, excluding sending repeaters, repeaters are now active in the network. The locking algorithm was still able to lock all repeaters in the first try.

[11] Inhibitory population are placed between excitatory populations.

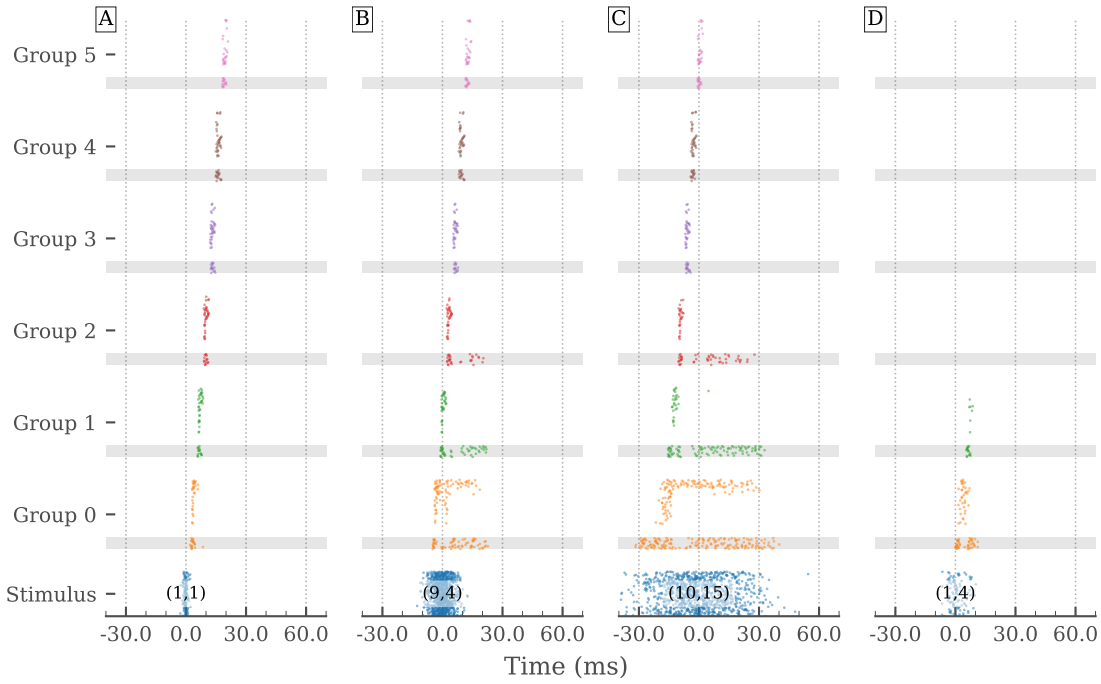


Figure 5.18: BSS-1 with Inhibition, without Background - Spike Trains – Spiking behavior for a network with FFI but without background activity and parameters as in Section 5.1.1. Inhibitory spikes are plotted on a gray background. **(A)** A weak, synchronous pulse packet causes a single pulse packet in each group. Inhibitory and excitatory neurons seem to spike at the same time. **(B)** A broader pulse input causes a higher activity in the first group. Nevertheless, only a single pulse packet is present in the final group. **(C)** For an even broader packet the earlier onset of the inhibitory neurons is visible. Even though the inhibitory neurons are highly active, the asynchronous pulse packet is still propagated along the chain. **(D)** The network is able to prevent the transmission of asynchronous packets if the strength is weak enough.

Spiking Behavior

For all but the lowest weight between excitatory populations, some pulse packets were able to reach the last group. As an example we display some spikes trains for the configuration with the base weight (see Section 5.1.1) in Fig. 5.18.

The weakest and most synchronous pulse causes a single sharp pulse response in each group. However, the activity of the excitatory neurons in the last group is low and it seems like the pulse packet would die out. In each group we now also have inhibitory neurons. They spike at approximately the same time as the excitatory neurons; an earlier onset as in the NEST simulation, Fig. 5.2, is not distinguishable. Due to the stronger connection to the preceding excitatory population, $g_{E_i \to I_{i+1}} > 1$, and no incom-

ing inhibition the inhibitory neurons are more frequently activated. As a consequence the response of the inhibitory neurons in the last group is considerably stronger.

For broader input stimuli the earlier onset of the inhibitory activity is visible. This earlier onset is as well caused by the stronger synaptic weight and the absence of incoming inhibition.

Even though the inhibitory neurons are highly active during a broad stimulus input they fail to prevent the transmission of asynchronous pulse packets. The inhibitory synaptic current on the excitatory neurons is too weak.

As for the network without inhibition, Fig. 5.13, weak, asynchronous inputs die out after the first few groups. The transmission delay from group to group is also in the same range as without inhibition and still depends on the weight of connections between excitatory populations.

Phase Diagram

For the chain without inhibition we already observed a binary response in the last group if the weights were low enough, Fig. 5.14. In case of the lowest weight the chain even showed filtering for synchronous input.

Figure 5.19 shows the phase diagrams of networks with inhibition in comparison with the previous tested networks. For the lower displayed weight the introduction of FFI shrinks the basin of attraction and improves the filtering for synchronous input. Nevertheless, strong, asynchronous inputs are still propagated along the chain.

In case of the highest tested weight we saw that the output in the final group was not binary. The response depended, in agreement with the NEST simulations, on the properties of the input packet and several neurons in the last group contributed more than one spike. As for Kremkow et al. [22] including FFI results in a more uniform response in the final group. Once more, the basin of attraction only shrinks slightly.

That the network still propagates asynchronous input is a sign of ineffective inhibition, compare Fig. 2.8 (B). In order to improve the filtering properties we try to increase the weight of incoming synapses to the inhibitory population, (E) and (F). No improvement can be observed.

Figure 5.18 shows that the inhibitory neurons are effectively activated for asynchronous input but are still not able to prevent the spiking of the excitatory population. We conclude that the weight of the incoming excitatory synapses, $g_{E_i \to I_{i+1}}$, is sufficient but the induced inhibitory current on the excitatory population is too weak.

Increasing $g_{I_i \to E_i}$ did not show an effect on the filtering properties. We assume that the synaptic conductance might saturate again, Appendix A.5. Therefore, we decrease the reversal potential of inhibitory synapses $E_{\text{rev-inhib}}$ to increase the synaptic current, (G) and (F).

A tiny decrease of the region of successful propagation can be observed for both weights. Smaller reversal potentials lead to a more pronounced improvement of the

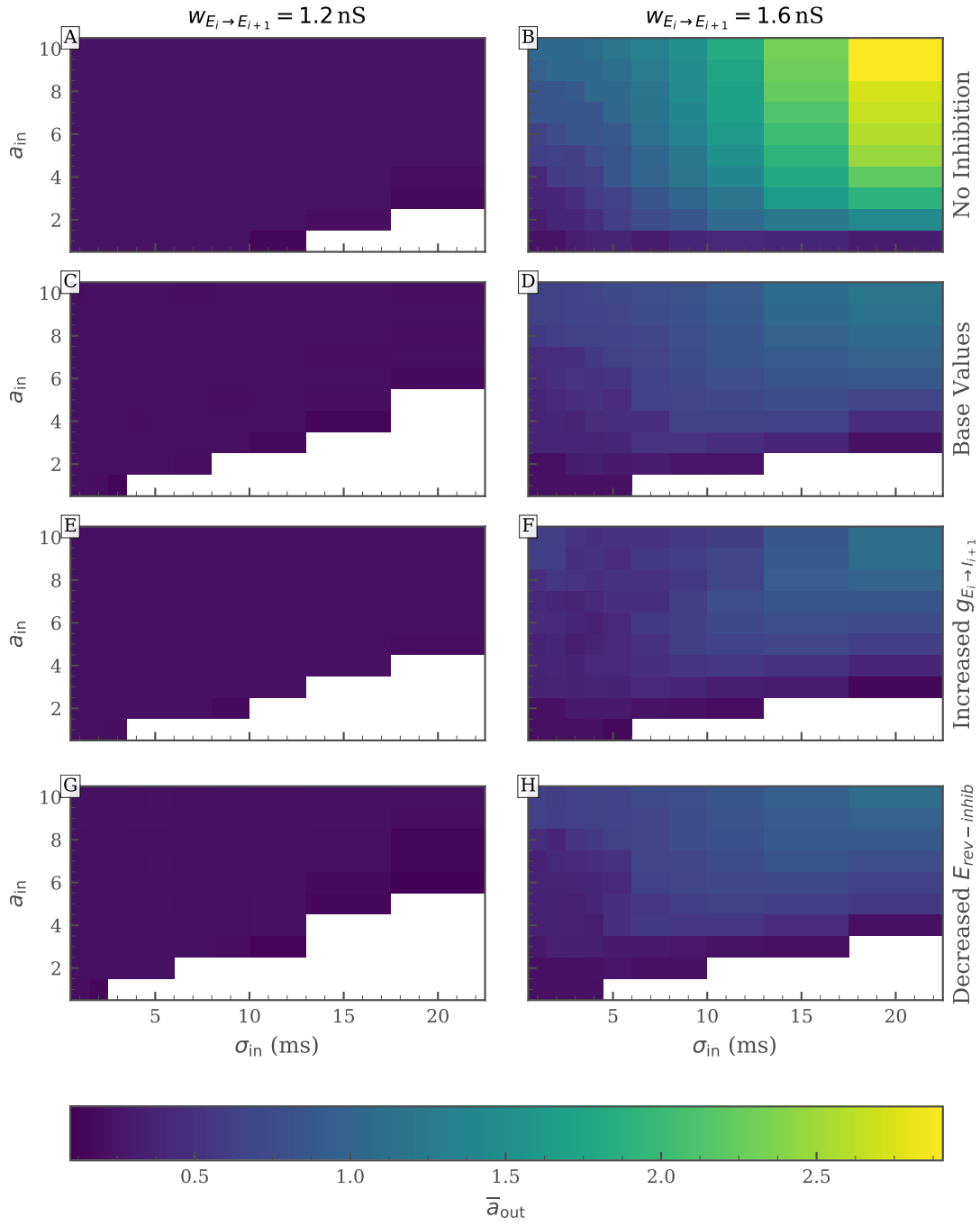


Figure 5.19: BSS-1 with Inhibition, without Background - Phase Diagrams – Response in the final group averaged over all trials and draws. Parameters are given in the figure. All other values are the same as in the base network, Section 5.1.1. **(A,B)** Network without inhibition. Same as in Fig. 5.14 **(C,D)** Network with base values. Inhibition is able to reduce the basin of attraction and to make the response in the final group more uniform. **(E,F)** $g_{E_i \to I_{i+1}} = 5$: the filtering properties are not improved. **(G,H)** $E_{\text{rev-inhib}} = -85\,\text{mV}$: the basin of attraction shrinks slightly. For more detail see text.

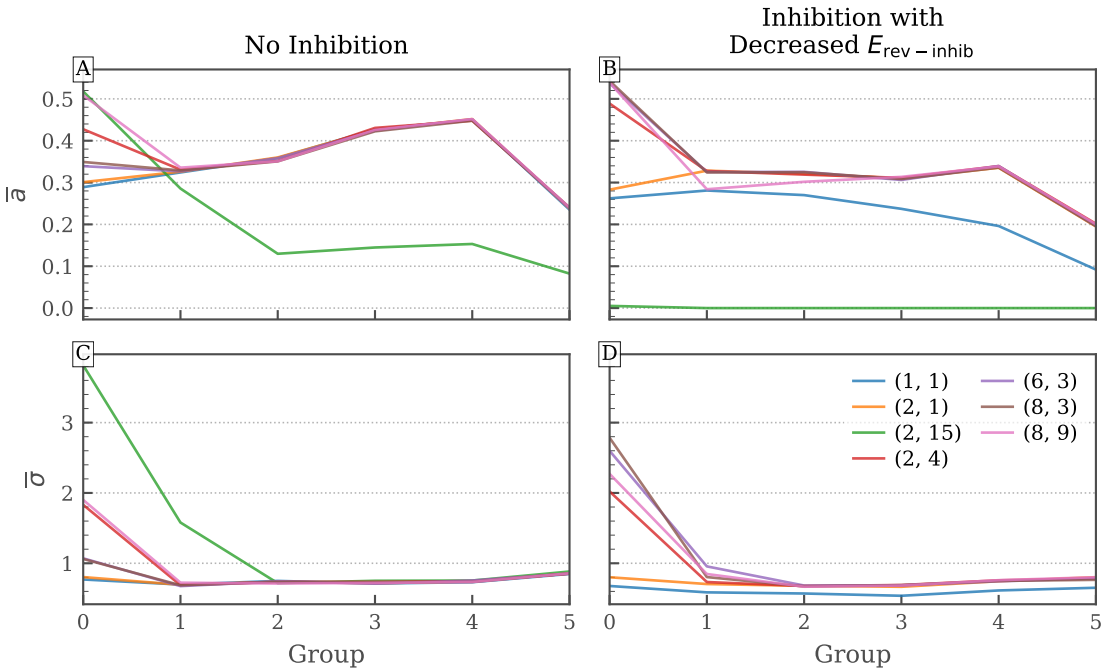


Figure 5.20: BSS-1 with Inhibition, without Background - Evolution Along the Chain – Evolution of strength and standard deviation for a number of selected input stimuli. The values are averaged over all trials and draws. Base configuration, Section 5.1.1, with no background activity and $E_{\text{rev-inhib}} = -85\,\text{mV}$ for (B,D). **(A,B)** For both network configuration the response strength converges towards zero or a common value. The strength drops in the final group. **(C,D)** The standard deviation of the response decreases from group to group after the third group most of the packets show the same low temporal spread of less than one millisecond.

filtering properties, Fig. A.10.

Evolution Along the Chain

Figure 5.20 compares the activity along the chain for some selected input stimuli. Without inhibition the output strength of all but (2,15) synchronize in the second group and stays the same for all packets in the following groups, subfigure (A). The activity slightly increases to the penultimate group and finally drops in group 5. This trend was already visible in the network without inhibition, Fig. 5.16. The strength of (2,15) shows an decrease from group to group and it seems that the pulse packet would finally die out.

With inhibition a similar trend can be observed for the evolution if the strength, subfigure (B). A subset of the packets synchronize after a few groups and share a common output strength and the strength of the others is zero or weakens from group to group. (2,15) already fails to invoke a response in the first group this time. The network with

inhibition seems to suppress asynchronous input more effectively. While in case of no inhibition (1,1) shows the same activity as the other packets which reach the last group, the strength is lowered with inhibition. It looks like the packet would die out in longer chains.

Overall the strength from group 1 to 4 is reduced and more flat for the network with inhibition. As a consequence the drop in the last group is less pronounced.

For both networks the standard deviation of almost all packets is unified in the third group and stays low at about one millisecond, subfigures (C) and (D). Due to the weaker response in the first group, (1,1) shows a slightly lower standard deviation than the other pulse packets in case inhibition is present. It seems like the pulse packets need longer to synchronize but have a smaller standard deviation if inhibition is present.

5.3 Long Chains on BSS-1

The previous section showed that the BSS-1 system is able to emulate the dynamics of a synfire chain with FFI. Up to now we only used a small number of HICANNs in a part of the wafer where few defects are present. How does the mapping and with it the network dynamics change if we use larger parts of the wafer?

We also observed a drop in the output strength in the final group in our previous emulations, compare Fig. 5.20. Will the signal strength still be sufficient to excite a synchronous response in the next group? Can the synfire chain compensate neuron to neuron variations which are inherent on hardware? By extending the chain to include up to 40 links we try to answer these questions in this section.

5.3.1 Default Mapping

Experiment Parameter – ID: `2020-06-21 00h05m`

- Wafer : 24
- Chain Length: 20 and 40
- Structural Changes (compared to Table 5.3 & Table 5.1):
  - $w_{E_i \to E_{i+1}}$: 1.4 nS and 1.6 nS
  - N_I: 0 and 25
  - N_B: 0
- Neuron Parameter Changes (compared to Table 5.2): None
- Other Changes:
  - External stimulus input distributed over 24 HICANNs and six FPGAs

In our first set of experiments we use marocco's default placement algorithm to place all populations but the stimulus. The tested changes are now so long that the mapped neuron populations enter regions of the wafer where more blacklisted components are present. How will marocco deal with more restricted resources?

As the verification of the hardware configuration does currently not exclude blacklisted components, we will skip the verification step during the emulation of the synfire chains. Consequently, the average runtime per trial dropped to around 16 s.

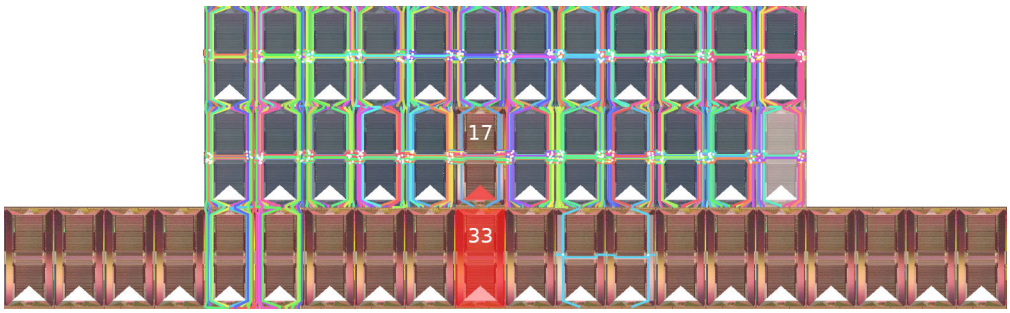


Figure 5.21: 20 Groups - Default Mapping Result – Mapping result for the default marocco placement for a chain with inhibition and a chain length of 20. Neurons are marked by a blue color. The higher the opacity the more neurons are placed on the HICANN. External inputs are resembled by the red triangle in the lower part of the chip. The opacity once more encodes their amount. Neurons are placed from left to right. Each HICANN hosts a little more than 100 neurons. Fully blacklisted chips are colored red. Colored lines represent the connections between sending repeaters and synapse drivers. Inter-stimulus inhibition is injected in the center of the second row. As in Fig. 5.12 the stimulus input is placed in the lower left corner (not visible in this figure).

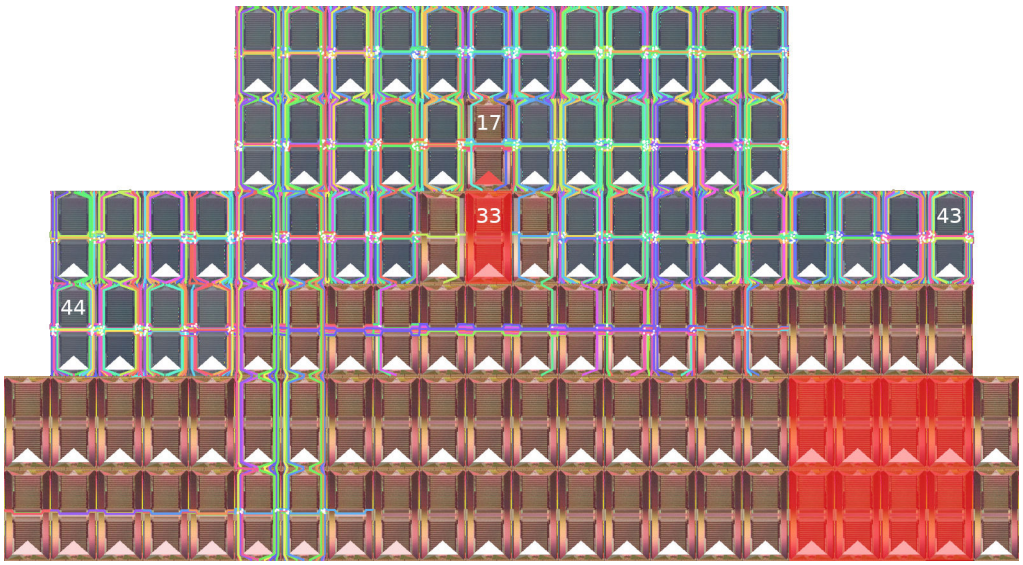


Figure 5.22: 40 Groups - Default Mapping Result – Mapping result for the default marocco placement for a chain with inhibition and a chain length of 40. Neurons are marked by a blue color. The higher the opacity the more neurons are placed on the HICANN. External inputs are resembled by the red triangle in the lower part of the chip. The opacity once more encodes their amount. Neurons are placed from left to right. Each HICANN hosts a little more than 100 neurons. Fully blacklisted chips are colored red. Colored lines represent the connections between sending repeaters and synapse drivers. Inter-stimulus inhibition is injected in the center of the second row. As in Fig. 5.12 the stimulus input is placed in the lower left corner.

Mapping

The mapping results are displayed in Figs. 5.21 and 5.22. In both cases stimulus and inter-stimulus inhibition are placed as before. HICANN 33 is blacklisted such that no neurons are placed on it. The neighboring chips do not host any neurons since buses which are needed to rely events in the on-wafer communication network are blacklisted due HICANN 33.

As the projections from inhibitory to excitatory neurons $I_i \rightarrow E_i$ were defined last, the loss is most serious for these connections, Fig. 5.23. Until group 15 the loss is below 20 %; from group 15 to group 16 about half of the synapses are lost. This high loss is due to an empty HICANN between the two parts of the group, Fig. 5.21: about half of the excitatory neurons of group 15 are placed on the HICANN 16 and the other half on HICANN 18. Due to this empty chip the horizontal lines on which the inhibitory neurons want to inject their spikes are already occupied by excitatory inputs from group 13 and the inhibitory connections can not be realized[12].

The synapse loss shows a weak periodic behavior with a period around 5 to 6. This oscillation of the loss is caused by the splitting of the populations over several chips. Each chip can implement about 100 logical neurons such that the ratio between the parts of the groups which are placed on different chips changes from group to group. The occurrence of the empty chip breaks this periodic behavior.

The neurons are again placed from left to right. When the boundary of the wafer is reached placement continues on the left; at this point the routes have to cross the entire width of the wafer. This leads to a high usage of horizontal lines.

A first effect of this congestion on horizontal lines can be observed for a chain with 20 links. One route from the inhibitory population on HICANN 20 to the excitatory population on HICANN 19 uses buses of chips in the row below, blue route in Fig. 5.21.

More severe effects can be seen in Fig. 5.23 (B) where lots of synapses are lost for connections involving group 35 or 36. Group 35 is placed on the right side of the wafer while group 36 is split over HICANN 43 and 44, compare Fig. 5.22. Due to a blacklisted chip[13] in the third row a high number of horizontal buses in the fourth row is already used and no connections from the right to the left can be realized for the mentioned groups: the synfire chain is cut in two. In addition, a whole reticle is blacklisted in the lower right corner in Fig. 5.22. Therefore, marocco is not able to establish a straight

[12] At each chip boundary the horizontal lines are shifted by two, Fig. 2.1, such that on every fifth HICANN sending repeaters inject events on the same physical lines. As we have 125 neurons per group and about 100 available neurons on each HICANN, a single group is almost always distributed over two chips. HICANN 16 is shared between group 14 and 15. Group 14 receives excitatory input from neurons in group 13 (on HICANN 14). The empty chip in between causes the clash of these excitatory inputs and the inputs of the inhibitory population of group 15, Fig. A.11. Note that all eight lines on which sending repeaters are placed are used up by the excitatory connections.

[13] No routes run through a blacklisted HICANNs such that we already use row four to redirect routes which would normally run through HICANN 33.

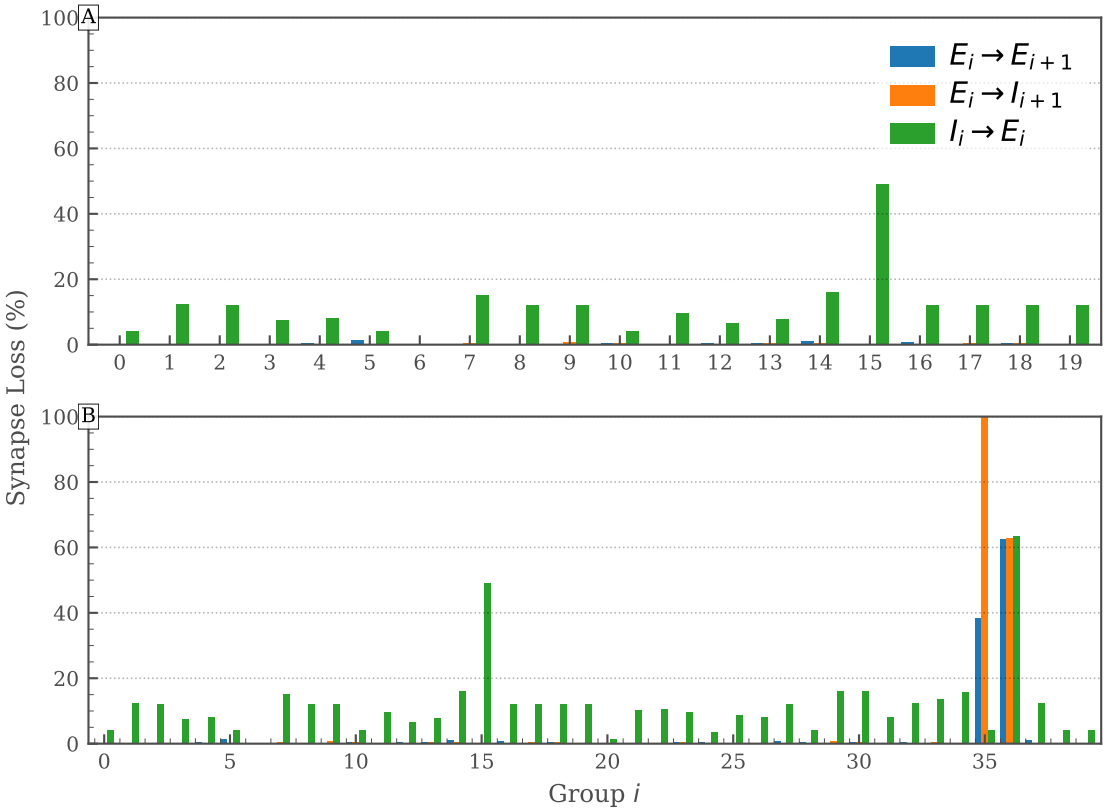


Figure 5.23: Long Chain - Synapse Loss with Default Mapping – Synapse loss between different populations for a chain length of 20 (A) and 40 (B). **(A)** Only for connections within a single group significant loss can be observed. The peak at $i = 15$ can be explained by a chip which was left empty during placement, compare text. **(B)** The in-group loss is similar to the one in the shorter chain. Between group 35 and 36 the neuron placement changes from the right to the left side of the wafer which results in a high loss and the chain is cut in two parts at this point.

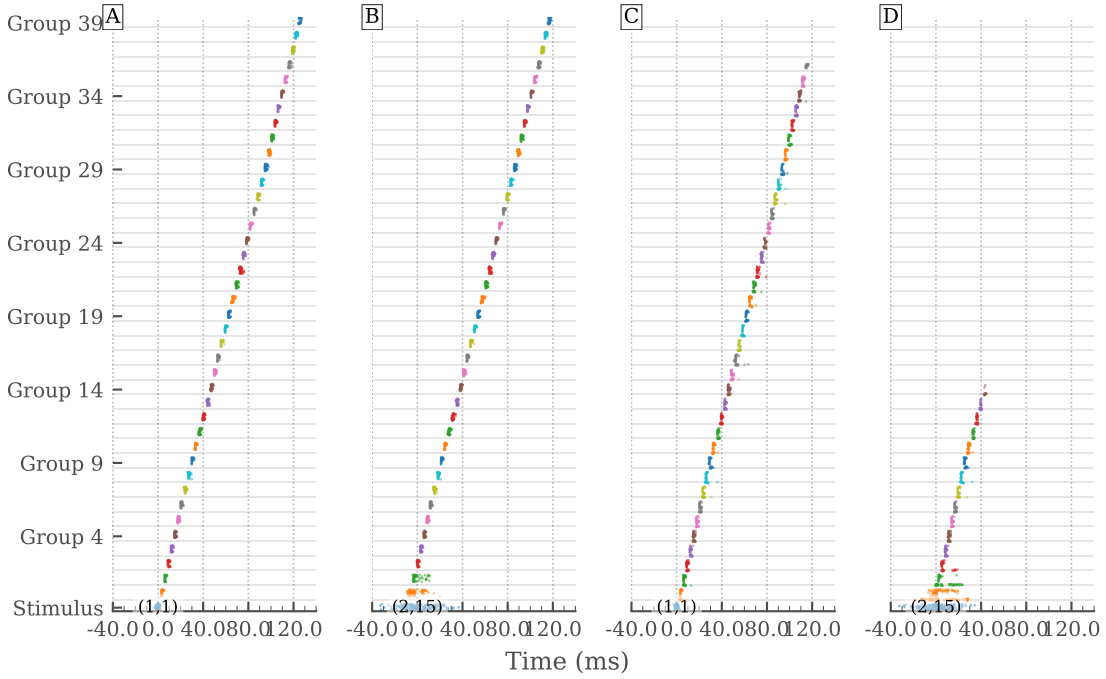


Figure 5.24: 40 Groups - Spike Trains – Spikes of neurons in the inhibitory populations are plotted on a gray background. Trains for a chain with 40 links. Base parameters (Section 5.1.1) with weight $w_{E_i \to E_{i+1}} = 1.4\,\text{nS}$. Sub figures (A) and (B) are simulations without inhibition. **(A)** A weak but synchronous input reaches the final group. **(B)** A broad stimulus also leads to activity in the last group. The response looks similar to (A). **(C)** In case of inhibition the pulse does not reach the final group. This is caused by the high synapse loss, Fig. 5.23. **(D)** The broad stimulus now vanishes before the last group; inhibition improves the filtering properties of the network.

route southwards, go to the left and then to the north to arrive on HICANN 44. We conclude that the left-to-right placement is not optimal for a synfire chain, especially in the presence of blacklisted components.

The number of repeaters increased to about 480 for the shorter chain without inhibition and up to 1355 for the longer chain with inhibition. Locking was rarely successful in the first try and up to three retries were necessary to lock all repeaters.

Spiking Activity

The spike trains of the chain with a length of 40 and some selected input stimuli are displayed in Fig. 5.24. Without inhibition we do not reach the bottleneck between HICANN 43 and 44 – the chain stays intact. Propagation of input pulse packets to the last group is possible, subfigure (A) and (B). The chain is also susceptible to asynchronous

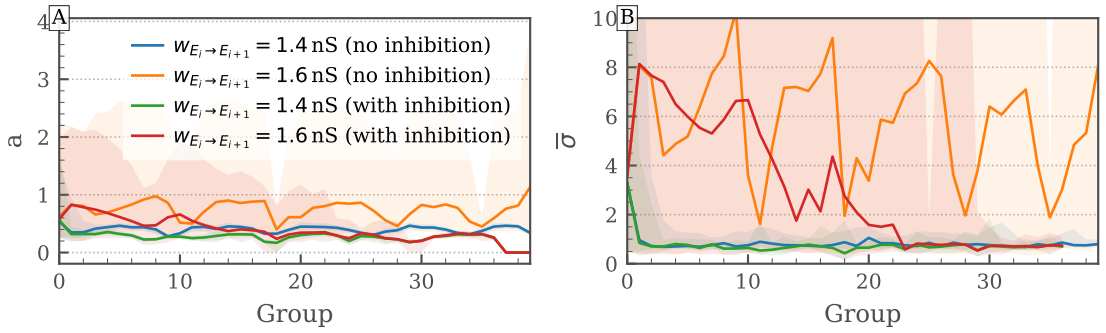


Figure 5.25: 40 Groups - Evolution Along the Chain – Base parameters (Section 5.1.1) with weight given in the figure and without background stimulus. **(A)** The strength of the packets varies from group to group as the neurons and synapses are not identical on hardware. For all but the highest weight without inhibition the strength gets more unified in higher groups. The periodic behavior in the orange trace can be attributed to the placement of the groups and the spike loss induced by a limited readout rate, compare text. **(B)** A similar trend as for the strength is visible. Strong responses with a high temporal spread have to be interpreted with care as the response can no longer be described by a Gaussian.

stimuli and the trains responses look similar after a few groups. This unification along the chain will be discussed in a later section.

Figure 5.23 showed that the chain is split in two if inhibitory populations are present. The cut was between group 35 and 36. Consequently, the spike trains die out at this point, Fig. 5.24 subfigure (C). In case of asynchronous input the signal may vanish before this cut and the network shows some ability to filter for synchronous input. This was already observed in chains with fewer links, Fig. 5.19.

Evolution Along the Chain

In case of shorter chains, Section 5.2.6 we already observed that the synfire chain can unify the response in the final group. For weights lower $w_{E_i \to E_{i+1}} = 1.4\,\mathrm{nS}$ the activity in the final group is binary regardless of the inclusion of FFI: packets died out before reaching the final group or initiated a response of about (0.2,1) in the last group. When increasing the weight, the response in the final group depended on the input properties of the initial packet. FFI could make the final response in the basin of attraction more uniform but a slight dependency on the stimulus input was still observed, Fig. 5.19.

These trends are still present when emulating longer chains, Fig. 5.25. In case of the lower weight the response are getting more uniform from group to group. The exact strength and standard deviation changes slightly in each group due to the variation of neuron and synapse parameters.

Strength and standard deviation are slightly lower in case inhibition is included. In

addition, the propagation does, as expected, die out after 37$^{\text{th}}$ group due to cut in the synfire chain, see Fig. 5.23 (B).

A periodic behavior of strength and standard deviation can be observed if the weight is higher and no inhibition is included. The period of the oscillations seems to be around eight but does not seem to be fixed. We assume that this oscillation can be attributed to the missing spikes due to the limited readout rate, compare Fig. 5.13. As each HICANN hosts a bit over 100 neurons, the populations are not always distributed in the same way over multiple chips; which in turn leads to a periodic behavior.

Furthermore, the strength shows a higher mean and the deviation between the strength of the packets is considerably higher than for all other configurations, shaded area in Fig. 5.25. This behavior is also visible for the determined temporal spread of the packets. As mentioned earlier the response is no longer given by a single packet and a Gaussian does no longer adequately describe the data such that the determined temporal spread should be interpreted with care.

Overall the means of strength and standard deviation also show for the highest weight without inhibition a slight decrease as the packets reach later groups. In the last group both values increase since it is less affected by the limited readout rate.

With a length of 40 the chain is now long enough to unify the signals which travel along the chain if inhibition is present. The mean values of standard deviation and strength drop from group to group. This also holds true for the variation in these two values which becomes less and less at the same time and seems to be gone after group 31.

For the lower weight the propagation success shows a higher trial to trial variation. The packets have to be transmitted successfully in between each group to be able to reach the final link. Due to manufacturing the neuron and synapse properties vary from group to group. If these properties are unfavorable in just one group the pulse may die out. The lower weight of 1.4 nS seems to be near the minimum weight for which most neuron populations on the wafer are still able to forward incoming packets.

5.3.2 Manual Mapping of a Wafer-Scale Chain

Experiment Parameter – ID: `2020-07-07 08h28m`

- Wafer : 24
- Chain Length: 190
- Structural Changes (compared to Table 5.3 & Table 5.1):
  - $w_{E_i \to E_{i+1}}$: 1.4 nS and 1.6 nS
  - N_E: 80

  - N_I: 20
  - N_B: 0
- Neuron Parameter Changes (compared to Table 5.2): None
- Other Changes:
  - External stimulus was, as before distributed over several FPGAs, and each neuron population was placed manually as well

In the previous section, where we used the default mapping algorithm, the synfire chain was cut in two parts due to a high synapse loss. We will now not only place the stimulus but all populations manually. A similar approach was already taken by Jeltsch [3] to place a so called "Hellfire chain" on the wafer. As before we perform five trials with ten draws each and do not verify the hardware configuration, compare page 83. The average runtime increased to 26 s since the locking of repeaters takes longer.

Mapping

In order to simplify the manual mapping we decreased the number of excitatory and inhibitory neurons such that each group consists of 100 neurons. Most HICANNs offer space for a little over 100 logical neurons and we can therefore place an entire group on a single chip. If the number of available neurons was less than 100, we excluded the chip from the manual mapping. HICANNs with fewer than 400 DenMem circuits, marked orange in Fig. 5.26, were excluded from the beginning. For all other chips we tried to place 100 neurons and, if it failed, excluded the chip.

The stimulus input is distributed over 24 chips on 7 FPGA. We use two HICANNs on the same reticle as the first population to inject stimulus spikes since they could not implement desired number of neurons and could therefore not be used to place a group on it.

In the previous section the chain as cut in two since the mapping jumped form the right to the left side of the wafer. To avoid this we will go to the south when reaching an obstacle, wafer edge or blacklisted reticle/chip, and then continue in the other direction. This results in a zig-zag pattern which starts in the north and goes to the south of the wafer.

We place the first group on HICANN 30 and then continue to the left until we reach the edge of the wafer. The chip in the south is then used for the next group. From there we continue to the right until we come near a fully blacklisted reticle, HICANN 58. As the chips next to blacklisted HICANNs have a minimized number of horizontal/vertical buses, we leave a margin of one around blacklisted chips. Consequently, we continue to the south and go left again.

Since the middle part of the wafer is blacklisted, we travel down on the left side of the wafer. When we come near the bottom, we go to the right side of the wafer and place the groups in a spiral path to the top. As a result we have a zig-zag path on the left seide which goes from the north to the south and then continues on the right side back to the north of the wafer.

All connections from the inter-stimulus inhibition were lost in our network configuration. The synapse loss for from the excitatory population outgoing connections was overall below 5 % while the projection from the inhibitory to the excitatory population showed losses of about 10 %, only twice all synapses were lost.

The whole network consisted of 19 000 neurons and about 1.4×10^6 synapses; 230 HICANNs were used for the implementation. Half of the time all 903 used repeaters could be locked in the first try; for the other half one additional try was necessary.

Spiking Behavior

Figure 5.27 shows the propagation of one pulse packet which reached the final group for each weight. In both cases the propagating pulse packet leads to an almost straight diagonal in the figure; only small deviations are visible where the group-to-group delay is slightly increased/decreased. Table 5.5 already revealed that a higher weight leads to a faster propagation along the chain. This is also visible in Fig. 5.27 where the pulse packet reaches the final group about 10 ms earlier if the weight is higher.

As before the transmission success shows a large trial-to-trial variability for the lower weight such that successful transmission of a large number of input packets was only possible in one trial.

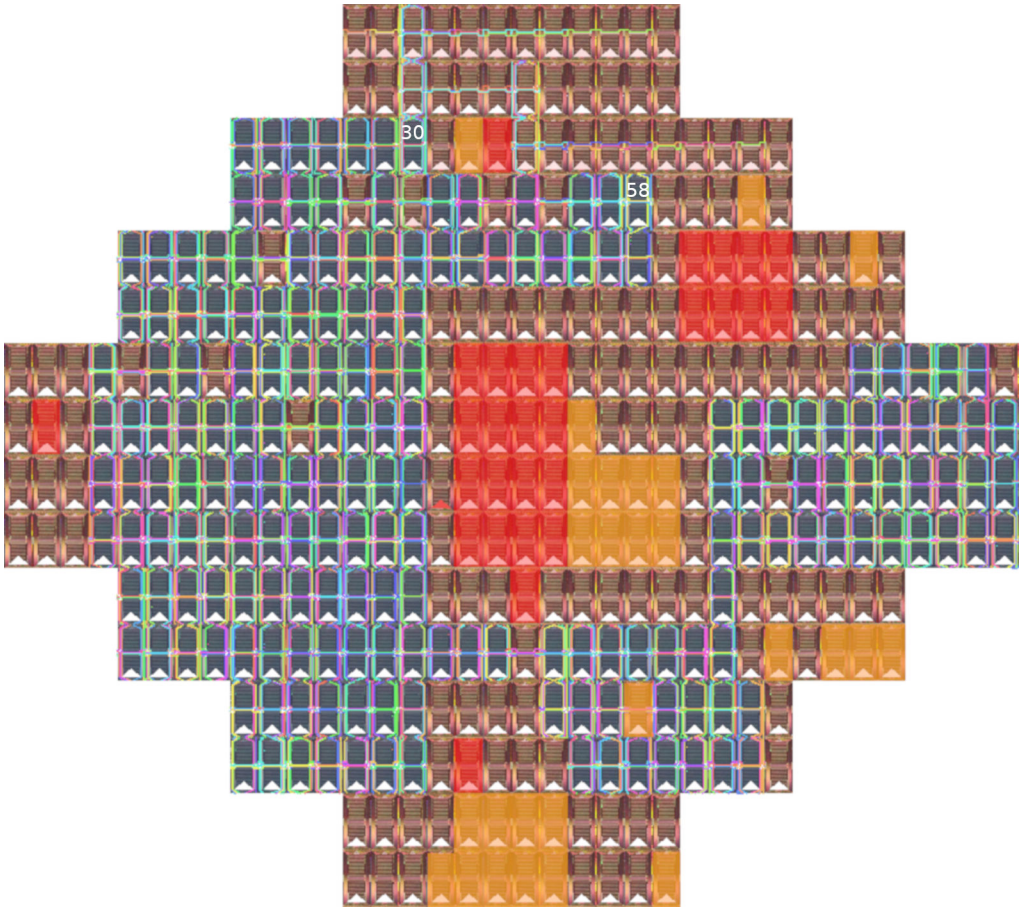


Figure 5.26: 190 Groups - Manual Mapping Result – Manual mapping of a synfire chain with feed forward inhibition and 190 chain links. Neurons are marked by a blue color. The higher the opacity the more neurons are placed on the HICANN. External inputs are resembled by the red triangle in the lower part of the chip. The opacity once more encodes their amount. Neurons are placed from left to right. Each HICANN hosts a little more than 100 neurons. Fully blacklisted chips are colored red while HICANNs with less than 400 non-blacklisted DenMem circuits are marked orange. Colored lines represent the connections between sending repeaters and synapse drivers. Inter-stimulus inhibition is injected slightly left to the center of the chip. Stimulus input was placed manually in the north of the wafer.

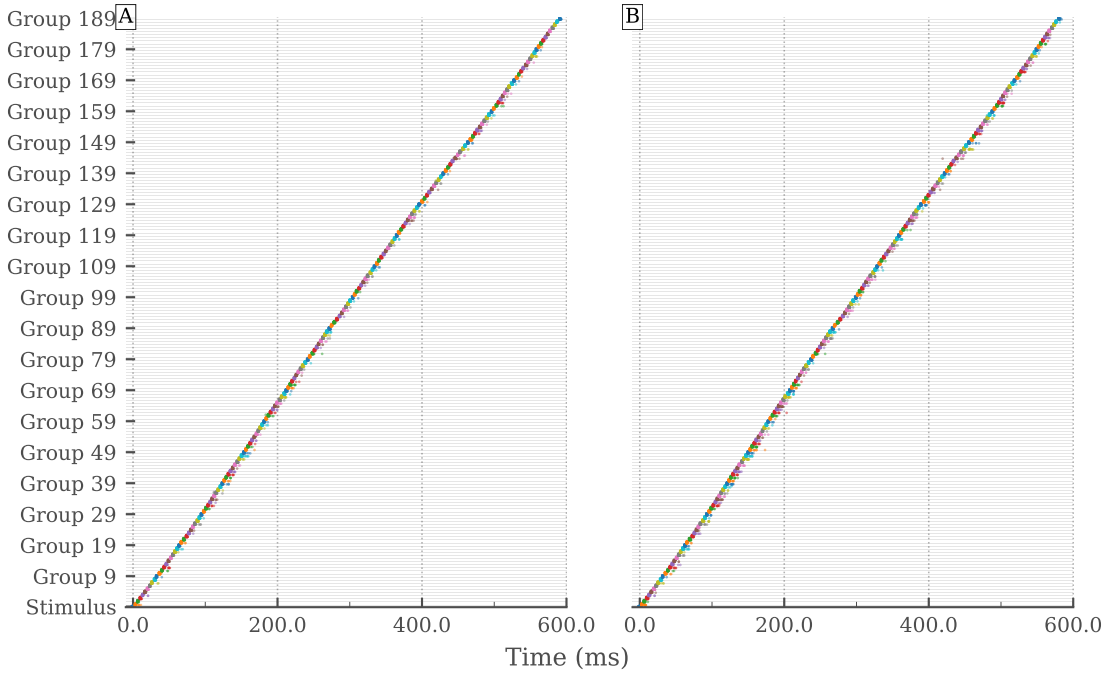


Figure 5.27: 190 Groups - Spike Trains – Spikes of neurons in the inhibitory populations are plotted on a gray background. Emulation with base parameters (Section 5.1.1) but without background stimulus and $w_{E_i \to E_{i+1}} = 1.4\,\text{nS}$ (A) $w_{E_i \to E_{i+1}} = 1.6\,\text{nS}$ (B). Responses to an input packet of (1,1). The packets reach the final group slightly earlier if the weight is higher.

6 Conclusion and Outlook

We showed the successful implementation of a large scale neural network: a synfire chain with 190 chain links, Section 5.3.2. In total 230 HICANNs were involved in mimicking the behavior of 19 000 neurons and over 1.4 million synapses.

The successful emulation of such a large network was enabled by a new locking scheme of the repeaters which are part of the on-wafer communication network, Section 3.2. Unlike before the locking of these repeaters was not only tried once but the reset was pulled several times in case the repeater did not lock. Tests showed that the total number of locked repeaters can be increased with this new routine; this allows to successfully setup networks with a large number of repeaters.

We tested the locking behavior for different operation frequencies of the on-wafer communication network, Chapter 4. It turned out that the locking success is not optimal for the current default frequency of 125 MHz and that the percentage of locked repeaters is increased with higher frequencies. It was not tested if the communication is still reliable, i.e. if the repeaters still decode and encode the incoming signals correctly. Further tests are needed to determine whether a higher frequency is beneficial.

In case of the highest tested frequency of 200 MHz the unsuccessful locking could be traced back to a small set of repeaters. It could not be determined whether these repeaters were defect and additional tests are needed. In future defect repeaters could be blacklisting; this would allow to successfully lock even larger networks.

The time needed to configure the BSS-1 system before experiment start could be reduced by shortening the waiting times during writing/reading of synapse weights and decoder addresses: access of a single row in the synapse array was speeded up by about 1 ms, Section 3.3.

Before implementing the long chain with 190 links, we simulated shorter chains in NEST and emulated them on hardware. We discovered that the input rate of external events reaches it limits for the dense input stimuli tested in our simulations. Distributing the stimulus neurons over several HICANNs on different FPGAs allowed to inject a higher density of spikes. Nevertheless, we still expect a high number of spikes to get lost when injecting the most dense packets (1000 spikes drawn from an Gaussian with a standard deviation of 1 ms). The implementation of drop counters in the FPGA is on

its way to determine the exact number of drops.

Some experiments indicated that asynchronous pulse packets are sometimes propagated further along the chain than more synchronous ones. In NEST simulations a similar effect was observed, Fig. A.5, and could be attributed to the gradual extinction of the input pulses, Fig. A.6. On hardware this effect seems to be enhanced by spike loss and a possible saturation of the synaptic input current, Appendix A.5.

In emulations on hardware the output strength in the different groups was relatively low compared to the strength in NEST simulation. This is assumed to be caused by parameter variations on hardware and a more systematic investigation is needed.

At the transition to longer chains, it turned out that the default mapping algorithm of marocco is suboptimal for synfire chains. Jumping from the right side of the wafer to the left side can cause a high synapse loss if blacklisted components are present. In our case this lead to a cut of the chain and input signals could not be propagated to the end of the chain. Furthermore, we observed for low weights that the propagation probability showed a higher trial-to-trial variability due to parameter variations on hardware. Inhomogeneities of neuron and synapse parameters are more likely to lead to an extinction of the pulse packet.

By placing the neurons manually on the wafer, a functional synfire chain with 190 links could be implemented which utilize a large fraction of a BSS-1 wafer.

A | Appendix

A.1 Additional Information to Used Software and Data Storage

If not stated otherwise we used a custom build version of the software stack including a new weight calibration developed by M. Wehrheim and S. Schmitt for experiments related to synfire chains: `nmpm_software/2020-06-07-4740-1`. This software version was build with the singularity image `/containers/stable/2020-05-29_1.img` and the commit ids of the different repositories are given in Table A.1.

The extended locking test were performed with the most recent version of the software stack. A log of the commit ids of the different repositories can be found in the experiment folder.

For all runs on wafer 24 we used the `/wang/data/commissioning/BSS-1/rackplace/-24/calibration/2020-05-11-1` as calibration and `/wang/data/commissioning/BSS-1/rackplace/24/derived_plus_calib_plus_layer1_plus_custom_blacklisting/WIP-2020-05-19` for blacklisting.

Experiment data are saved in `/loh/users/jkaiser/DataMasterThesis`. The sub directories for each experiment are given in Table A.2. For the extended relocking tests, Section 4.2, and the synfire chain, Chapter 5, these folders contain sub folders named by the unique experiment id (data and time).

Table A.1: Storage Locations of Experiment Results – Commit ids of the custom software version `nmpm_software/2020-06-07-4740-1`.

Repository	`Commit ID`
repo_db	`2729e2d58355c8fd48774dda735725961413b23e`
adc-calib	`cae0e1eeb8d2d5f63bbc78acd78c3080325c30ac`
bitter	`cd78f4ee9bdfc0ee785c76f43e677576ff089958`
cake	`0e75959129bef5e9d7fc97ac3597d466b2006c6d`
calibtic	`b1ce7f7e608b51a76bf3aa19fb4504b46aeb5d6b`
cd-denmem-teststand	`49c208262d5f59425d005395d2cd03fa060ba674`
chip-teststand	`500f30a6f2b00678f71744c38d8c21060eaf1d0e`
code-format	`6b7ea4c595d7e0bb3655bacee9bf13c8fb8fd175`
euter	`838adf2e97735919a3ece946dd9e90ef57989d1e`
halbe	`d4fb86ade32246cbb6cb32bef93c66513673c594`
halco	`23d6906b93805ffee8c80696d34404a3169a9dd4`
hate	`66e2f8321778b4379612ec2564a43aa0d015e2f9`
hicann-system	`e19f953f9bf94b3fee380f30dde367c22ae06c46`
hmf-fpga	`708514c4e0d7584e69281689ff1b01bdc7592edb`
hmf-fpga-test	`80b8fc93498557722344d1f164d95e84168b9a88`
hwdb	`7d1d60978d9f773f7ce67b654725110078605b1e`
lib-boost-patches	`2d7e07d4e74827c42d9e1a51f8d180af9907f7cb`
lib-rcf	`1b681d745b071653b98ac8893ef7c123fe258662`
lib-vhdl-utils	`3f8f1f18e3b3d06f80bf2f07fed8991057ec403d`
logger	`24fa8fc38d7861ea57f6b91e478753c3108a1b0e`
marocco	`b6efa3508aaf3e14438a486d340f3a78680cb9fe`
odeint-v2	`9473d7f067b9c4b56862de644898b0ac99676075`
pygccxml	`acd80b8e816091d13807fba77c158f9971ed2db6`
pyhmf	`26fcf61775f46f4b48474f57f328577109ed9fcd`
pyplusplus	`8ca5344fdc39007f5b117fabf4fd9adecb9ad43d`
pythonic	`e9628388d2eb0ce34db770c660ea37718d97dd3d`
pyublas	`fb538e8c313a3f04d1a5b77200d192fece3ea901`
pywrap	`550051ab0faad678e58cb456079b1ba45ad2230a`
rant	`856963edcdaf43d86220309d5a84e80ffd1d077b`
redman	`ba69cfbc95850cd2188be577f37acd11f364b6e6`
sctrltp	`1cf4d47a37c9466350be3a08d7606be3516424ab`
sthal	`daf83160c104a4a7f4c944ff560d0ab1a6b8d8bf`
systemsim-stage2	`b2aca9af60203f58f9d5edb3d76ec3360374ac69`
visions-slurm	`06b298835f6fafd79455cbc98da44adcd5af3fea`
vmodule	`2eb1aeadd2338f61c7098273dff95fbc2eb1e5d7`
ztl	`d900ab073f6aa8df4bf7f187bdbb65f1f6cac2f6`

Table A.2: Storage Locations of Experiment Results – Paths are given relative to `/loh/users/jkaiser/DataMasterThesis`.

Experiment	Section/ Chapter	`Folder`
Locking Test	3.1.3	`synapse_controller_confirm_locking`
Relocking Test	3.2.2	`relocking_test`
NOP-waits	3.3.3	`nop_waits_speed_test`
Extended Locking Tests	4	`repeater_locking_test`
Synfire Chain	5	`synfire_chain`
Saturation Test	A.5	`saturation_test`

Table A.3: Extended Relocking - Experiments – Overview over the experiments performed with the extended relocking test in Chapter 4. The start time of the experiment is used as a unique ID. Patch sets (PS) are saved in CS 11290, compare Appendix A.2.

Experiment Name	Section	`ID`	PS
Varying Sleeps	4.2.1	`2020-06-27 09h52m`	1
Varying ν_{pll} and V_{dllres}	4.2.2	`2020-06-28 23h21m`	1

Table A.4: Synfire Chain - Experiments – Overview over experiments done in Chapter 5. The start time of the experiment is used as a unique ID. Patch sets (PS) are saved in CS 10338, compare Appendix A.2.

Experiment Name	Section	`ID`	PS
NEST with Inhibition and Background	5.2.1	`2020-06-01 08h10m`	32
NEST without Inhibition and Background	5.2.2	`2020-06-01 22h51m`	32
NEST without Inhibition and Background, with Paramater Variations	5.2.3	`2020-07-10 12h21m`	48
BSS without Inhibition and Background - Default Mapping	5.2.4	`2020-06-05 15h25m`	32
BSS without Inhibition and Background	5.2.5	`2020-06-14 14h27m`	36
BSS with Inhibition, without Background	5.2.6	`2020-06-14 16h06m`	36
BSS Long Chain - Default Mapping	5.3.1	`2020-06-21 00h05m`	37
BSS Long Chain - Manual Mapping	5.3.2	`2020-07-07 08h28m`	42

A.2 Where to Find the Code?

The Electronic Vision(s) group uses the code review system gerrit[1]. Changes to the software stack are collected in so called change sets (short CS), reviewed by other members of the team and then integrated in the production code. When modifications to a change set are required, a new patch set (PS) can be uploaded. A patch set can be thought of as a commit to the change set. Most of the software changes presented in this thesis were a change set before being merged in production code, Table A.5.

A dedicated repository was created in order to save all code which is related to the synfire chain: `model-hw-synfire`.

In addition, Jenkins-jobs for the extended repeater locking test and the synfire chain are currently under development.

Table A.5: Overview over Change Sets – Change sets which were created or used in/for this thesis. Chapter 5.

Description	Section/ Chapter	Change Sets
Restructuring of Synapse Controller	3.1	2939, 2941
Synchronization of Command Buffers	3.1	7798, 8235
Digital Test of Synapse Controller	3.1.3	10507
Relocking of Repeaters	3.2	3948, 4048, 4046, 8634, 8635
NOP-Waits	3.3	7318, 9315, 9397, 9398, 9399, 9481, 9482, 9483, 10289, 10985
Extended Locking Test	4	11290
Synfire Chain	5	10338 (4047 weight calibration)
Saturation Test	A.5	11369

[1] https://www.gerritcodereview.com/

A.3 Additional Figures

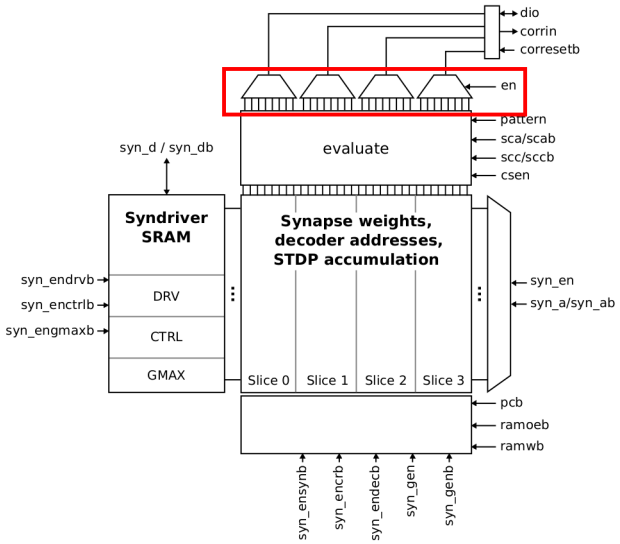


Figure A.1: Multiplexer Used for Synapse Array Access – In order to save bit lines not all weights/decoder addresses are accessed in parallel. The synapse array is organized in four slices and eight eight column sets. Eight synapses in each slice belong to one column set. Therefore, each slice is also split in eight parts (visible as vertical buses which enter the four multiplexers). Access is always performed one column set; accessing eight synapses in each slice. Four multiplexers (highlighted by a red box) select the values of a column set with the help of the enable signal `en`. Taken from Friedmann [7].

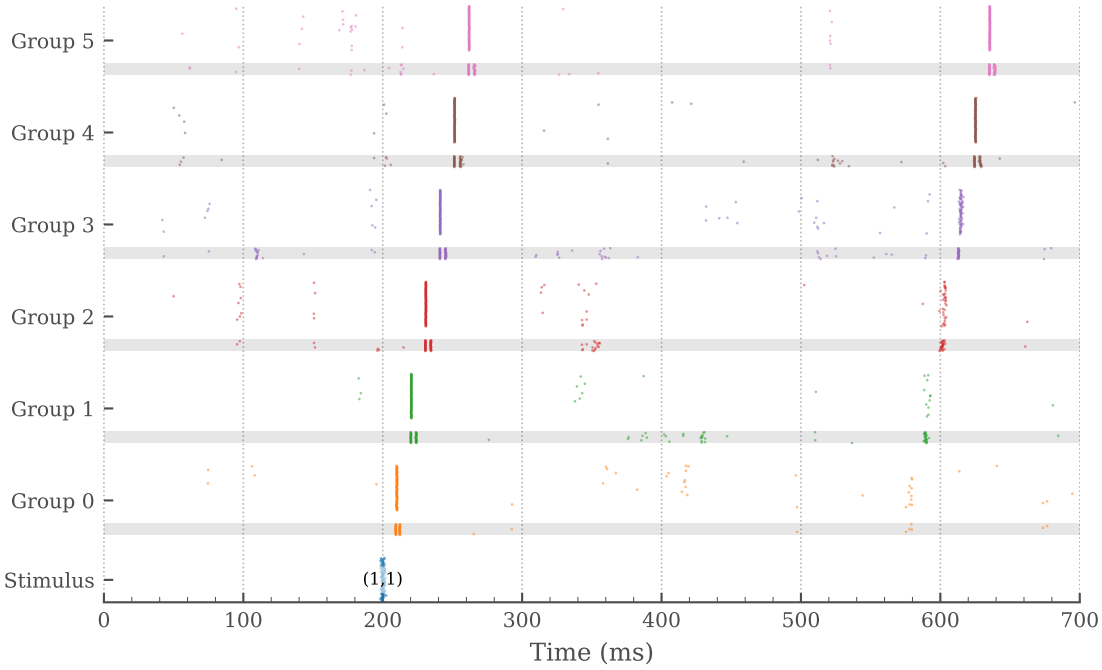


Figure A.2: NEST with Inhibition and Background - Background Induced Pulse Packets – For $r_B = 400\,\text{Hz}$ and base parameters (Section 5.1.1) synchronous spikes may be introduced by random background spikes. A synchronous response in a lower group caused by background fluctuations can travel to the final group. This happens in this example in group 0 at around 580 ms. The synchronous activity in the last group is not anymore only caused by synchronous stimuli input.

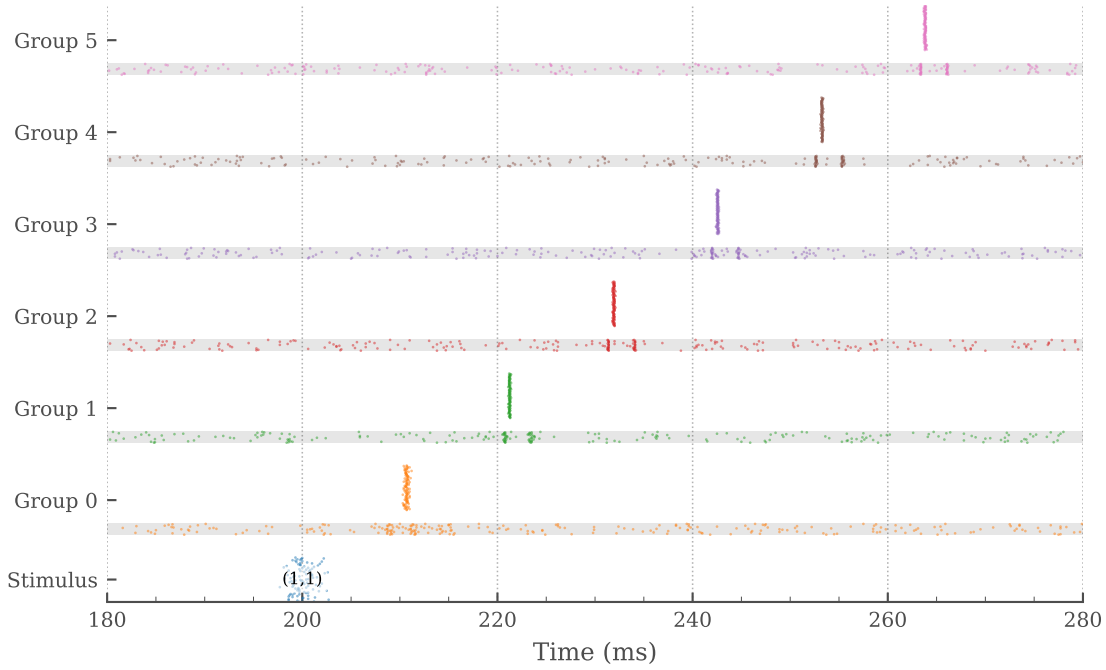


Figure A.3: NEST with Inhibition and Background - Strong Weights Reduce Spontaneous Firing – Doubling the base weight of Fig. A.2 increases the activity of the inhibitory neurons dramatically. This allows for reliable pulse propagation again. All other parameters as described in Section 5.1.1 and with $r_B = 400\,\text{Hz}$.

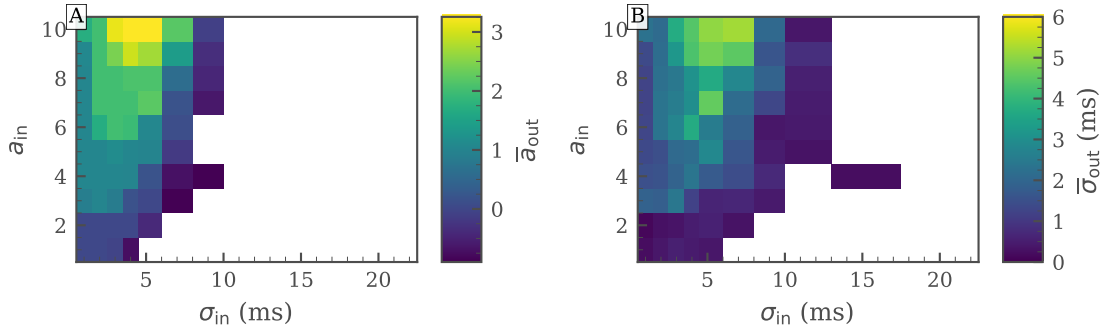


Figure A.4: NEST with Inhibition and Background - Response in the First Group – Simulation with base parameters (Section 5.1.1) and $r_B = 200\,\text{Hz}$. Responses in the first group averaged over all ten draws. Strengths smaller 0.1 are not printed. Both strength and standard deviation in the first group depend on the initial pulse packet.

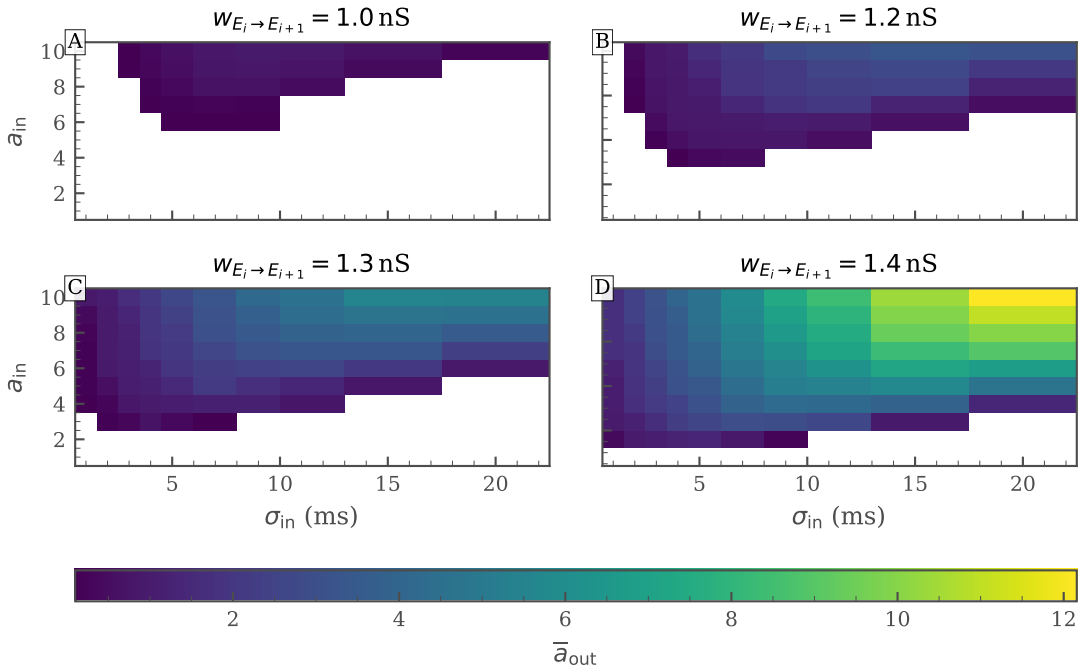


Figure A.5: NEST without Inhibition and Background, with Variations - Phase Diagram – Simulation with base parameters (Section 5.1.1) and weight as given in the figure (Experiment ID: `2020-07-08 09h46m`). Responses in the last group averaged over all draws. Strengths smaller 0.1 are not printed. In case of low weights the propagation probability can decrease with synchronicity. This effect is most prominent in subfigures (A) and (B) where the most synchronous input never reaches the final group. This is an effect of the gradual extinction of the incoming pulse packets, compare Fig. A.6.

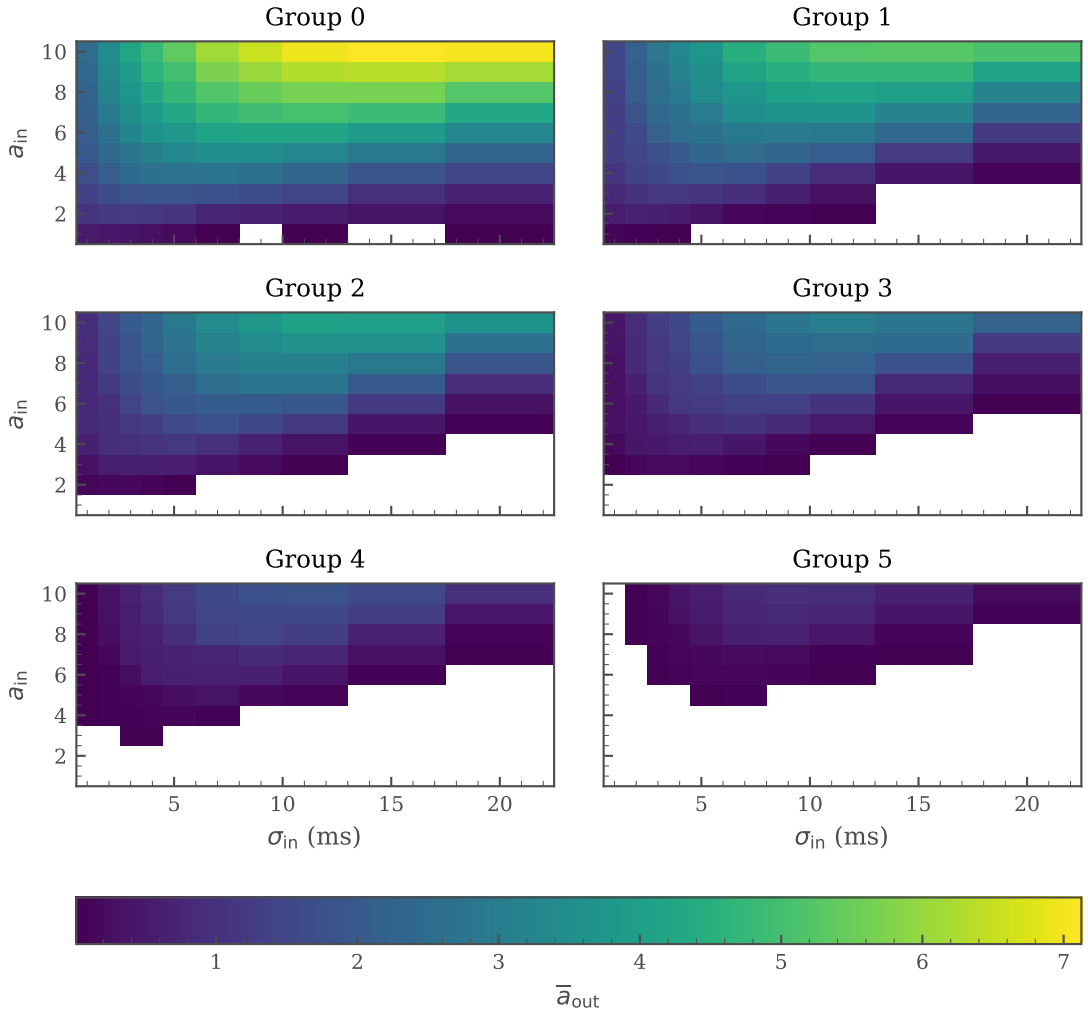


Figure A.6: NEST without Inhibition and Background, with Variations - Phase Diagram in Different Groups – Simulation with base parameters from Section 5.1.1 (Experiment ID: `2020-07-08 09h46m`). Responses are averaged over all ten draws and strengths smaller 0.1 are not printed. A increasing number of pulse packets die out when traveling along the chain and the region of successful transmission shrinks from group to group. It is assumed that the weight between excitatory populations is to weak and that all packets would eventually die out in longer chains.

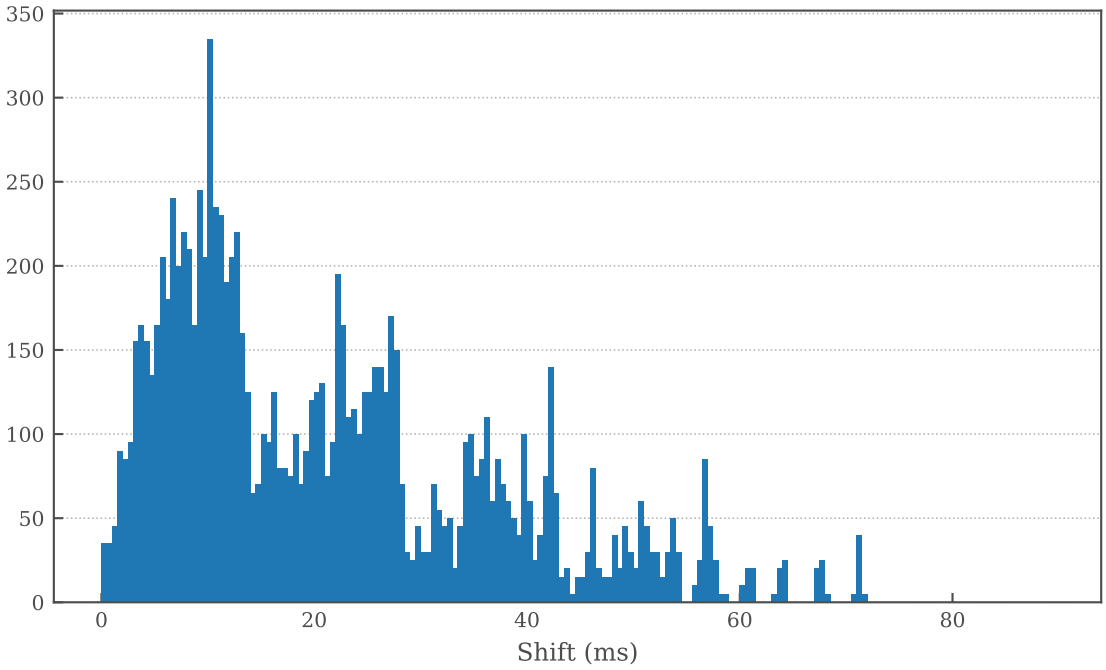


Figure A.7: Shifts due to Limited Group Size – The playback memory in the FPGA organizes spike in groups of maximal 184 spikes. If a new group is started and the delay between the first spike of the last group and the first spike of the new group is smaller than six FPGA clock cycles, the input times of all spikes in the new group are shifted. Here we show the shifts for the default mapping of a simplified network, Section 5.2.4. Data from experiment set `2020-06-05 15h25m` (C0, T0)

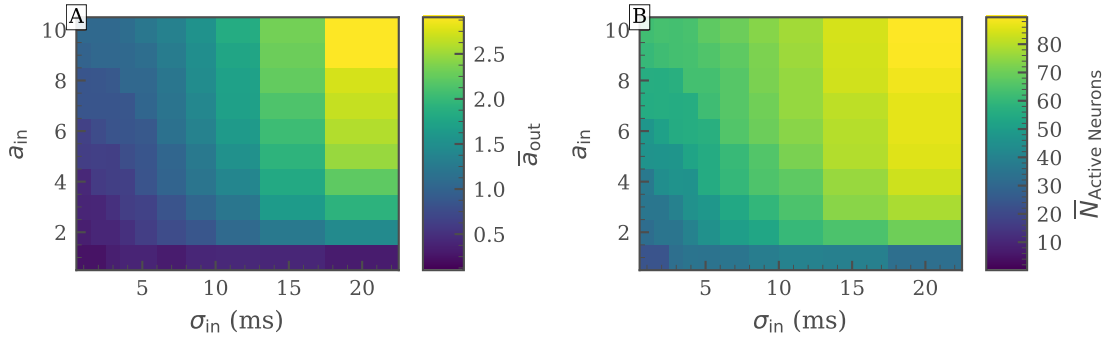


Figure A.8: BSS without Inhibition and Background - Active Neurons in Final Group – Emulation of the simplified network (no inhibition and background activity) with base parameters, Section 5.1.1, and a weight of $w_{E_i \to E_{i+1}} = 1.6\,\text{nS}$. **(A)** The output strength depends on the properties of the input stimulus. Several neurons are active more than once. **(B)** Similarly to the output strength, the number of active neurons depends on the input stimuli. For a broad and strong input almost all neurons in the final group can be activated.

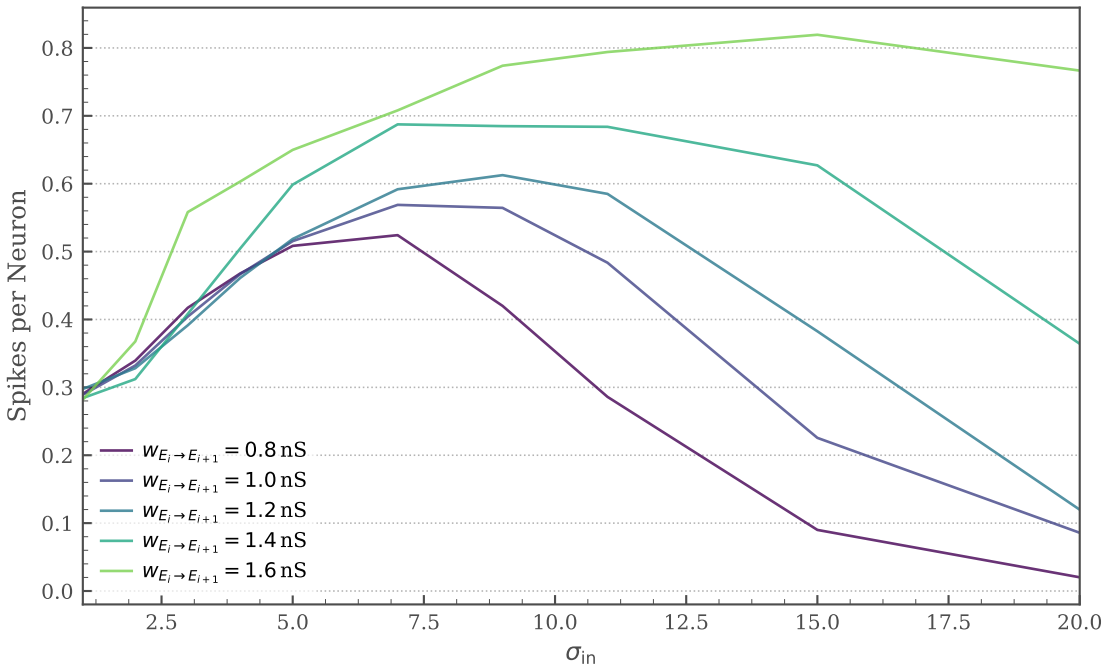


Figure A.9: BSS without Inhibition and Background - Spikes per Neuron in the First Group – Emulation of the simplified network (no inhibition and background activity) with base parameters, Section 5.1.1, and a weight of $w_{E_i \to E_{i+1}} = 1.6\,\text{nS}$. The number of spikes per neuron is considerably higher than the determined output strength for broad inputs, compare Fig. 5.17 and text in Section 5.2.5.

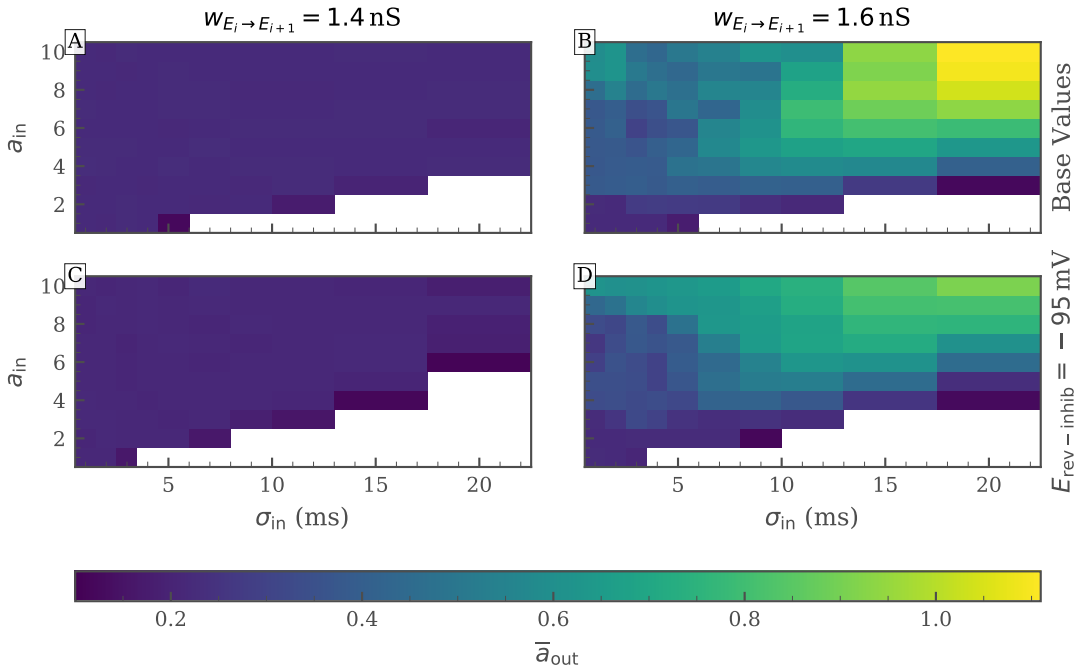


Figure A.10: BSS-1 with Inhibition, without Background - Improved Filtering by Decreased Synaptic Potential – In an attempt to improve the filtering properties of the network we decreased the inhibitory reversal potential, Fig. A.10. Here we decreased the potential even further to $E_{\text{rev-inhib}} = -95\,\text{mV}$. For both tested weight a decrease in the inhibitory synaptic reversal potential leads to smaller basin of attraction.

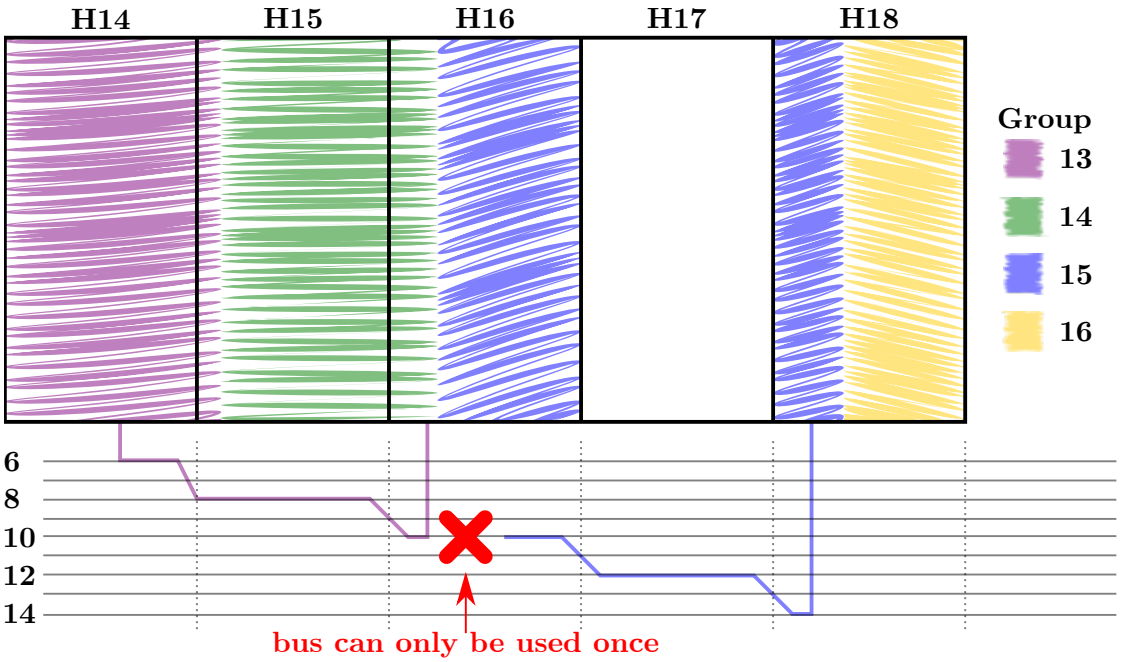


Figure A.11: Increased Synapse Loss due to Empty HICANN – Illustration why an empty HICANN increases the synapse loss. Group 13 establishes excitatory connections (red line) to group 14. Group 15 wants to connect the inhibitory population to the excitatory population in the same group. Horizontal lines shift by two at the edge of each HICANN and a sending repeater is placed on every eighth line, compare Fig. 2.1 (A). Due to the empty HICANN 17 which is not used for placement, the routes collide and the inhibitory projection can not be realized.

A.4 Evaluation

Our data are saved such that we can easily extract all spikes time of one neuron type in a single group. Let $T = \{t_0, \ldots, t_n\}$ be the set of these spike times and N_{neurons} the number of neurons. How do we now extract the pulse packet response to an input stimulus?

First of all we organized our experiment in such a way that we know the input time of the stimulus packet t_s and therefore a time frame in which the spikes should be investigated. We extract all spikes which are within this time frame:

$$T' = \{t \mid t \in T \text{ and } t_s - t_{\text{pre-stimulus}} < t < t_s + t_{\text{post-stimulus}}\}\,. \tag{A.1}$$

A.4.1 Evaluation Methods

Now we run through several conditions and evaluate the data accordingly. For each evaluation method we return the mean spike time μ within the pulse packet, the strength a and standard deviation σ as well as the mean spiking frequency ν of all neurons (excluding the pulse packet). To distinguish between the different evaluation methods we also label each method.

Less Than Two Spikes – Label 1

If there are less than two spikes evaluating the spikes makes no sense. We return that no pulse packet was found and that the mean rate is zero.

Narrow Time Frame – Label 2

We assume that the neurons are not spontaneously active if the first and last spike time are less than 10 ms apart. Since we are mostly interested in synchronous packets and do only need a rough estimate of the strength if the response is made up of more than one packet, we chose the relatively low time period of 10 ms. A higher time frame would lead to a better estimation of the output strength of responses which are made up of several pulse packets, compare Fig. 5.6. But it would make the standard deviation less accurate in case of synchronous pulse packets and a low number of spontaneous spikes.

With no spontaneous activity in the network we can assign all N spikes to the pulse response, $a = \frac{N}{N_{\text{neurons}}}$. The mean and standard deviation is determined by the spike times:

$$\mu = \frac{1}{N}\sum_{t\in T'} t, \qquad \sigma = \sqrt{\frac{1}{N}\sum_{t\in T'} (t-\mu)^2}. \tag{A.2}$$

The mean firing rate is assumed to be zero $\nu = 0$ (excluding the pulse packet).

Not Enough Spikes – Label 3

We now assume that the distribution is rather broad or that it is embedded in spontaneous activity. Therefore, we want to organize the data in a histogram and fit a Gaussian to it.

In order to get a reliable fit we need a minimum number of samples. Here we took the limit to be 20. If we have less than this number of spikes we again assume that there is no pulse packet present.

We return zero strength and not determined standard deviation/mean spiking time. The mean firing rate is given by the number of spikes N divided by the time frame of the stimulus $\Delta t = t_{\text{pre-stimulus}} + t_{\text{post-stimulus}}$ and the number of neurons N_{neurons}:

$$\nu = \frac{N}{\Delta t \cdot N_{\text{neurons}}}. \tag{A.3}$$

Maximum Activity to Near to the Edge – Label 4

The data is now organized in a histogram with bin size 1 ms.

For the following evaluation methods we need to evaluate a high number of samples around the bin with the most samples in it. We have chosen out timings such that the response packet should be well within the stimulus time frame. It is therefore justifiable to disregard all sets T' where the highest bin is in the first or last 30 bins.

We return zero strength and a mean rate as in method 3.

Narrow Distribution – Label 5

Due to the bin size of 1 ms narrow distributions with a standard deviation in the low millisecond range are hard to fit to the histogram. We use the number of spikes around a small region of the histogram to determine if the distribution is narrow.

First we estimate a mean rate by excluding a region of 30 bins around the highest bin. Then we determine the number of spikes N'_{center} in the highest bin and its two neighbors. We also calculate the number of spikes N'_{broad} in the center bin and ± 30 bins around it. From both values we subtract the expected number of spikes due to the spontaneous activity using the previous calculated mean firing rate; this gives us N_{center} and N_{broad}.

For a standard deviation of 3 ms (2 ms) a fraction of around $a = 0.38(0.55)$ spikes lies in a region of ± 1.5 ms around the center. If we assume the remaining spikes are in the broad region defined by ± 30 bins, we can use the ratio $\frac{N_{\text{center}}}{N_{\text{broad}}}$ to determine if a pulse is narrow.

We say a pulse is narrow if:

$$\frac{N_{\text{center}}}{N_{\text{broad}}} > 0.4. \tag{A.4}$$

0.4 was chosen to be near the expected fraction of spikes for $\sigma_{\text{ref}} = 3$ ms.

If Eq. (A.4) evaluates to true, we take all spikes which are within a region of $2\sigma_{\text{ref}} = 6\,\text{ms}$ around the center of the highest bin minus the mean background as strength. Mean firing time μ and standard deviation σ are also determined from this set of spikes, compare Eq. (A.2).

Gaussian Fit – Method 0

In case none of the previous methods was used we use SciPy to fit a Gaussian distribution, Eq. (A.5), to the histogram.

We divide the number of background spikes (which is given in Hz) by 1000 since we fit the data to a histogram with bin size 1 ms:

$$f(x) = \frac{a \cdot N_{\text{neurons}}}{\sqrt{2\pi\sigma^2}} \cdot e^{\frac{-(x-\mu)^2}{2\sigma^2}} + \frac{\nu \cdot N_{\text{neurons}}}{1000}. \tag{A.5}$$

A.4.2 Check of Evaluation Routine

In order to confirm the validity of our evaluation routine we use NumPy to generate Gaussian pulse packets of a given strength and temporal spread $(a_{\text{in}}, \sigma_{\text{in}})$. We embed these packets in spontaneous Poisson activity, rate r_s, generated with the help of elephant. Afterwards, we use the evaluation routine described in this section to evaluate these pulse packets.

The results are shown in Fig. A.12. The routine is able to determine the correct strength and temporal spread of spike packets even in the case of high spontaneous activity. For large temporal spreads the determined values are less accurate.

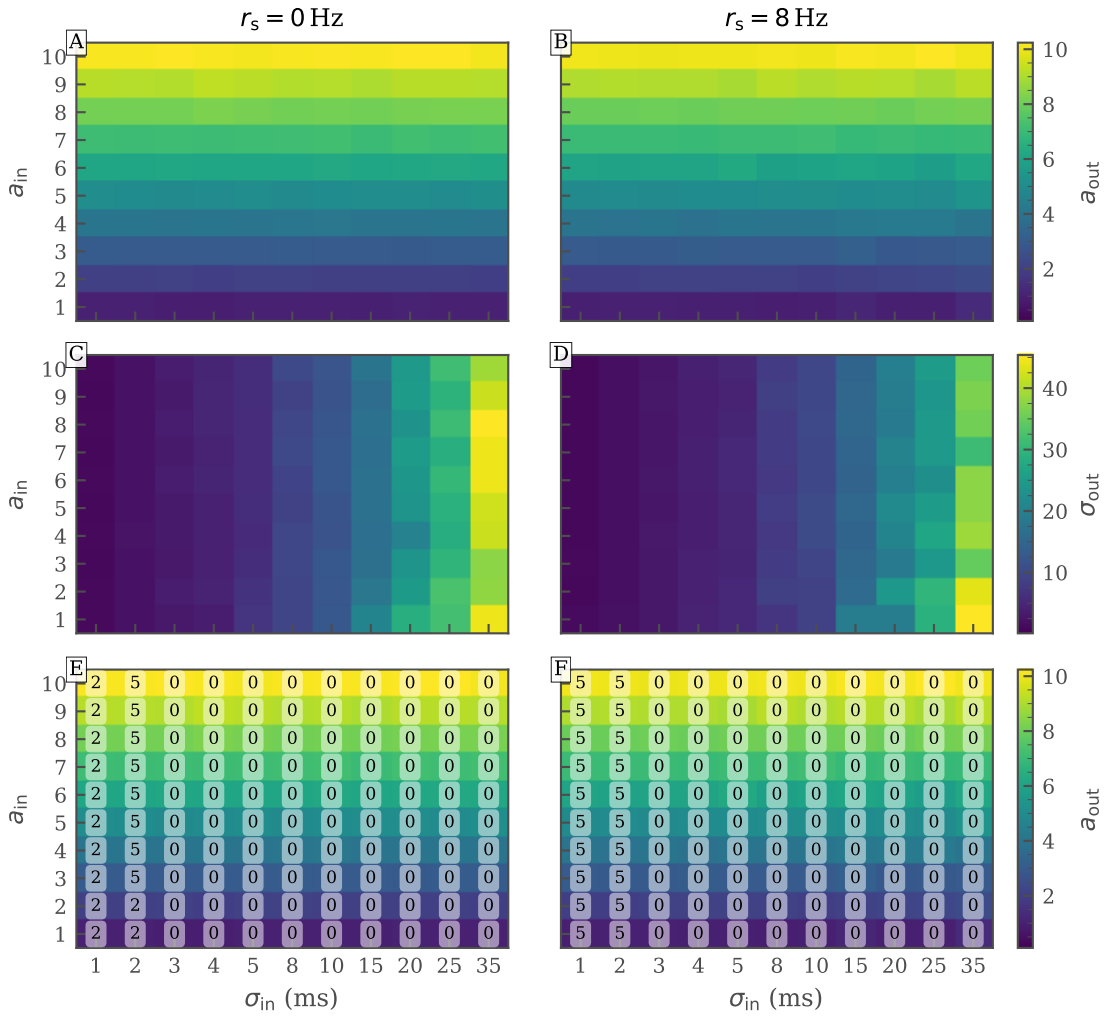


Figure A.12: Verification of Evaluation Routine – We generated for $N_{\text{neurons}} = 100$ neurons Gaussian pulse packets and embedded them in random spontaneous activity. The spontaneous activity is Poissonian and the rate is given per neuron. **(A,B)** Determined strength of the packets. For both activity rates the strength can be determined accurately. Slight variations for broad packets are visible. **(C,D)** Overall the evaluation routine can determine the temporal spread good. For large standard deviations the determined values are not as accurate as for lower standard deviations **(E,F)** In case of no spontaneous activity method 2 for the most synchronous packets, before method 5 was used and finally the fitting method 0 for broad packets. In case of a high spontaneous activity narrow pulse packets are reliably detected (method 5) and a fit is used for standard deviations larger 3 ms.

A.5 Saturation of Synaptic Current

In our analysis of synfire chains we repeatedly observed that a group of spikes with a small temporal spread causes less outgoing spikes than a group were the spikes are less densely packed. This effect can in some extend be attributed to the limited communication bandwidth and the spike loss which is caused by it. However, a constant number of active neurons by increasing synaptic strength, Fig. 5.17, suggested that the synaptic current might saturate.

To test this hypothesis we run a simple test in this section, CS 11369. A single neuron is placed on a HICANN chip and connected to ten stimulus neurons. Each of this stimulus neurons is placed on different reticles and only spikes once such that the input is not affected by the limited bandwidth between FPGA and HICANN. The neurons inject spikes with fixed delays in between them, e.g. for a delay of 0.5 ms the spike times would be given by a fixed offset plus $0.5, 1, 1.5, 2, \ldots$ ms.

We combine input packets with different delays in a single spike train and place inhibition between the different packets, compare Section 5.1, and measure the membrane potential of the target neuron. For each delay we determine the height of the PSP (maximum value of the membrane voltage, subtract the mean of the potential and divide it by the maximum potential measured for any delay. We call this value the relative height.

Figure A.13 and indicates that the height of the PSP at first increases with the temporal spread of the input. At a delay of 1 ms between incoming spikes the height of the PSP starts to decrease again.

Assuming that all spikes reach the target neuron and that the delays induced by different paths on the wafer[2] do not influence the measurement significantly, this suggests that the synaptic current saturates in case the synaptic input is too strong. From the measurements it is not clear if this is a "hard" saturation or if the synaptic current scales non-linear with the synaptic input. Further investigation is needed to determine where this saturation happens.

[2] Each repeater leads to a delay of about 2.3 ns in real time which corresponds to about 0.02 ms in biological time. At most 13 repeater were used; therefore, it is unlikely that the delays induced by the hardware compensate the delays of the injected spikes.

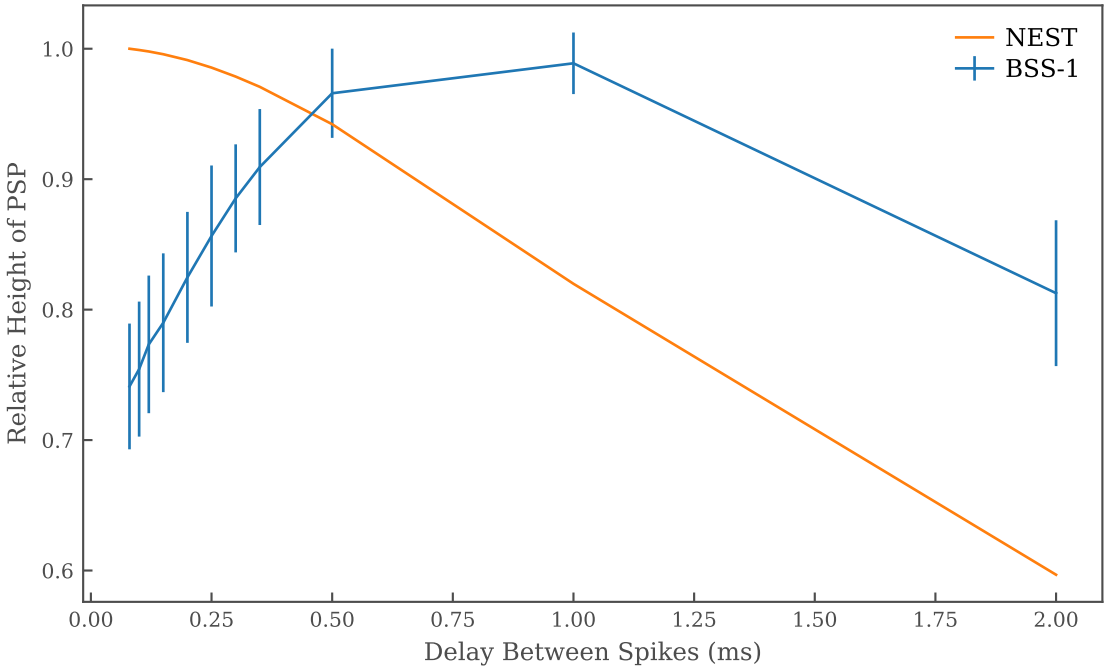


Figure A.13: Saturation of Synaptic Current – Relative height of PSPs for different delays between ten incoming spikes. The measurements were performed on eight different HICANNs. On each HICANN the logical neuron was placed at 16 different locations. The curve indicates the mean of the measured PSPs while the bars encode their standard deviation.

List of Figures

List of Tables

Bibliography

[1] 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"10)* (2010), pp. 1947–1950.

[2] Sebastian Schmitt et al. "Neuromorphic Hardware In The Loop: Training a Deep Spiking Network on the BrainScaleS Wafer-Scale System". In: (Mar. 6, 2017). DOI: 10.1109/IJCNN.2017.7966125. arXiv: 1703.01909 [cs.NE].

[3] Sebastian Jeltsch. "A Scalable Workflow for a Configurable Neuromorphic Platform". PhD thesis. Universität Heidelberg, 2014.

[4] Johannes Schemmel, Johannes Fieres, and Karlheinz Meier. "Wafer-scale integration of analog neural networks". In: (2008). DOI: 10.1109/ijcnn.2008.4633828.

[5] Johannes Fieres, Johannes Schemmel, and Karlheinz Meier. *Realizing biological spiking network models in a configurable wafer-scale hardware system*. 2008. DOI: 10.1109/ijcnn.2008.4633916.

[6] Matthias Hock. "Test of Components for a Wafer-Scale Neuromorphic Hardware System". In: (2009).

[7] Simon Friedmann. "A new approach to learning in neuromorphic hardware". PhD thesis. Heidelberg, Univ., Diss., 2013, 2013.

[8] HBP SP9. *Neuromorphic Platform Specification*. Human Brain Project, Sept. 18, 2018.

[9] Tobias Nonnenmacher. "Characterization of Spike-Timing Dependent Plasticity in Neuromorphic Hardware". Masterarbeit. Universität Heidelberg, 2015.

[10] Eric Müeller. "Novel Operation Modes of Accelerated Neuromorphic Hardware". PhD thesis. Universität Heidelberg, 2014.

[11] Johann Klähn. "Training Functional Networks on Large-Scale Neuromorphic Hardware". Master. Universität Heidelberg, 2017.

[12] Eric Müller et al. "The Operating System of the Neuromorphic BrainScaleS-1 System". In: (Mar. 30, 2020). arXiv: 2003.13749 [cs.NE].

[13] Andrew Davison, Daniel Brüderle, Jochen Eppler, Jens Kremkow, Eilif Muller, Dejan Pecevski, Laurent Perrinet, and Pierre Yger. "PyNN: a common interface for neuronal network simulators". In: *Frontiers in Neuroinformatics* 2 (2009), p. 11. ISSN: 1662-5196. DOI: 10.3389/neuro.11.011.2008. URL: https://www.frontiersin.org/article/10.3389/neuro.11.011.2008.

[14] Moshe Abeles. *Local Cortical Circuits*. 1982. DOI: 10.1007/978-3-642-81708-3.

[15] Romain Brette and Wulfram Gerstner. "Adaptive Exponential Integrate-and-Fire Model as an Effective Description of Neuronal Activity". In: *Journal of Neurophysiology* 94.5 (Nov. 2005), pp. 3637–3642. DOI: 10.1152/jn.00686.2005.

[16] Richard Naud, Nicolas Marcille, Claudia Clopath, and Wulfram Gerstner. "Firing patterns in the adaptive exponential integrate-and-fire model". In: *Biological Cybernetics* 99.4-5 (Nov. 2008), pp. 335–347. DOI: 10.1007/s00422-008-0264-7.

[17] Wulfram Gerstner, Werner M. Kistler, and Richard Naud. *Neuronal Dynamics*. Cambridge University Press, Jan. 18, 2019. 590 pp. ISBN: 1107635195. URL: https://www.ebook.de/de/product/22190732/wulfram_gerstner_werner_m_kistler_richard_naud_neuronal_dynamics.html.

[18] Markus Diesmann, Marc-Oliver Gewaltig, and Ad Aertsen. "Stable propagation of synchronous spiking in cortical neural networks". In: 402 (1999), pp. 529–533. ISSN: 0028-0836. DOI: 10.1038/990101.

[19] A. Aertsen, M. Diesmann, and M. O. Gewaltig. "Propagation of synchronous spiking activity in feedforward neural networks". In: 90 (1996), pp. 243–247. ISSN: 0928-4257. DOI: 10.1016/s0928-4257(97)81432-5.

[20] Marc-Oliver Gewaltig, Markus Diesmann, and Ad Aertsen. "Propagation of cortical synfire activity: survival probability in single trials and stability in the mean". In: 14 (2001), pp. 657–673. ISSN: 0893-6080. DOI: 10.1016/s0893-6080(01)00070-3.

[21] A. Kumar, S. Rotter, and A. Aertsen. "Conditions for Propagating Synchronous Spiking and Asynchronous Firing Rates in a Cortical Network Model". In: *Journal of Neuroscience* 28.20 (May 2008), pp. 5268–5280. DOI: 10.1523/jneurosci.2542-07.2008.

[22] Jens Kremkow, Laurent U. Perrinet, Guillaume S. Masson, and Ad Aertsen. "Functional consequences of correlated excitatory and inhibitory conductances in cortical networks". In: *Journal of Computational Neuroscience* 28.3 (May 2010), pp. 579–594. DOI: 10.1007/s10827-010-0240-9.

[23] Thomas Pfeil et al. "Six Networks on a Universal Neuromorphic Computing Substrate". In: *Frontiers in Neuroscience* 7 (2013). DOI: 10.3389/fnins.2013.00011.

[24] Mihai A. Petrovici et al. “Characterization and Compensation of Network-Level Anomalies in Mixed-Signal Neuromorphic Modeling Platforms”. In: *PLoS ONE* 9.10 (Oct. 2014). Ed. by Gennady Cymbalyuk, e108590. DOI: 10.1371/journal.pone.0108590.

[25] Garcia S. et al. “Neo: an object model for handling electrophysiology data in multiple formats”. In: *Frontiers in Neuroinformatics* 8:10 (Feb. 2014). DOI: 10.3389/fninf.2014.00010.

[26] Malte Wehrheim. “Reconstruction of Synaptic Weight on the Neuromorphic BrainscaleS-1 System”. Bachelorarbeit. Universität Heidelberg, 2019.

[27] Quirinus Schwarzenböck. “Towards Balanced Random Networks on the BrainScaleS I System”. Bachelor. Universität Heidelberg, 2019.

Acknowledgements

I want to thank:

- Sebastian for answering countless questions, helping me solving many problems and providing ideas as well as guidance throughout this thesis. Even on the weekend or in the evening you were there to help. Your efforts go way beyond what one expects of a supervision. Thank you!
- Johannes for providing me with the opportunity to write the thesis in the Electronic Vision(s) group.
- Eric for taking time to explain even the most simple programming principles to a novice in software development.
- Andreas for providing me with detailed knowledge about the hardware.
- Hartmut and Jose for discussions about the BSS-1 system and providing blacklisting/calibration data.
- Lukas for reading the thesis and correcting a large number of mistakes.

The work carried out in this Master's Thesis used systems, which received funding from the European Union's Horizon 2020 Framework Programme for Research and Innovation under the Specific Grant Agreements Nos. 785907 and 945539 (Human Brain Project, HBP).

Declaration

Erklärung:

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

Heidelberg, den (Datum) .....