Department of Physics and Astronomy

University of Heidelberg

Bachelor Thesis

in Physics

submitted by

Marco Rettig

born in Heppenheim, Germany

May 2019

Characterizing the Event Interface of the HICANN-X

This Bachelor Thesis has been carried out by Marco Rettig at the

KIRCHHOFF INSTITUTE FOR PHYSICS

RUPRECHT-KARLS-UNIVERSITÄT HEIDELBERG

under the supervision of

Dr. Johannes Schemmel

Abstract

In the course of this thesis, the Playback Executor HDL module of the HICANN-X event interface was equipped with functions that enable the time-coordinated sending of events to the HICANN as well as the chronological classification of the events returned by the HICANN. In the main part of the thesis, the event drop rate and the jitter during transport of event data through the event interface depending on the biological input event rate were investigated using HDL simulations. The investigation revealed that the drop rate increases at biological input event rates from 130 kHz. The variation of the time span for the transport of event data from the Playback Executor to the SPL1 links in the HICANN is much smaller than the accuracy of the membrane time constant of the neurons achievable through calibrations. The influence of the event transport on the precision of spike experiments is therefore negligibly small. The event losses can be traced back to an inefficient event packing in the HICANN-X chip and can probably be significantly reduced by an additional event packing in the L2 module of the FPGA.

Im Rahmen dieser Thesis wurde das Playback Executor HDL Modul des HICANN-X Event Interfaces mit Funktionen ausgestattet, die das zeitkoordinierte Senden von Events zum HICANN sowie die zeitliche Einordnung der vom HICANN zurueckgesendeten Events ermoeglichen. Anschliessend wurden die Eventdroprate und der Jitter beim Transport von Eventdaten durch das Event Interface in Abhaengigkeit der biologischen Input Eventrate mittels HDL Simulationen untersucht. Die Untersuchung ergab, dass die Droprate bereits bei biologischen Input Eventraten ab 130 kHz ansteigt. Die Schwankung der Zeitspanne fuer den Transport von Eventdaten vom Playback Executor zu den SPL1 Links im HICANN ist wesentlich kleiner als die mittels Kalibrationen erreichbare Genauigkeit der Membranzeitkonstante der Neuronen. Der Einfluss des Event Transports auf die Genauigkeit von Spike Experimenten ist somit vernachlaessigbar gering. Der Eventverlust ist auf ineffizientes Eventpacking im HICANN-X Chip zurueckzufuehren und kann durch weiteres Eventpacking im L2 Modul des FPGA vermutlich deutlich reduziert werden.

Contents

1 Introduction

1.1 Neuromorphic Hardware

The human brain is characterized by an energy-efficient information processing, a high learning ability and fault tolerance [1, 4]. The neuromorphic computing field of research is dedicated to the investigation of its functioning and the development of a wide range of applications which are based on a recreation of the brain's neural networks. When implementing artificial neural networks, two fundamental principles can be distinguished: The classical method is based on simulating the behaviour of neurons and synapses on digital Von-Neumann computers using numerical models. The Electronic Vision(s) group at the University of Heidelberg, in contrast, follows an alternative approach by implementing a physical neuron model as analog electrical circuits that allow for the emulation of neural networks [7]. The circuits are implemented as very large scale integration circuits (VLSI) and mounted on microchips [2]. The HICANN-X (High Input Count Analog Neural Network) chip represents the most recent version of the neuromorphic hardware developed by the group and is currently in the manufacturing process.

1.2 Motivation

The HICANN-X chip is intended to enable the execution of neuroscientific experiments aimed at investigating the learning processes of the human brain. In these experiments, individual neurons on the chip are stimulated by electrical pulses sent by the user. The sharp increase in the membrane voltage of a neuron during stimulation is also referred to as spike or event and can trigger the stimulation of further neurons depending on the parameters selected in the experiment. The spike behaviour of a neuron is largely determined by the spike or event times of the other neurons which are connected to this neuron. It is therefore of central importance for the execution and analysis of the experiments that the user can stimulate the selected neurons at specified times and knows which other neurons were stimulated at which times by subsequent spikes. Consequently, the event interface of the chip used for the communication between the user and the neurons must allow the transport of the address data of individual neurons as well as the data of their event times. A fundamental characteristic of the analog neuron implementation is that they evolve autonomously in a continuous time domain so that their spike behaviour is not reproducible. If the generated event data is not immediately processed by the event interface, data losses may occur. As the spike behaviour of the neurons is accelerated by a factor of $\approx 10^3$ compared to their biological equivalents, a large amount of events are generated in a short time during the experiments. For this reason, the event interface must enable high data transfer rates while minimizing data losses. In the ideal case, the time required to transport event data to the neurons should remain constant regardless of the event rate. The major goal of this thesis is to characterize the transport of event data through the event interface of the HICANN-X chip by investigating the explained properties. Before that, however, the Playback

Executor HDL module of the interface is extended by missing functions that enable the time-coordinated sending of events to the HICANN as well as the chronological classification of the events returned by the HICANN. As the chip is still in the manufacturing process, the event interface will be tested using HDL simulations.

1.3 Thesis Outline

In the following chapter the general functioning of the communication with the HICANN-X chip via the Playback Executor HDL module as well as the functions of the module that are relevant for the event transport through the event interface are explained. Chapter 3 deals with the actual characterization of the event transport through the event interface. To this end, the analyzed aspects and the approach followed in the investigation are first described. Thereafter, the preparatory steps that are relevant for obtaining valid results as well as the calculation of the quantities considered in the analysis are explained. In the last part of the chapter, the results of the analysis are presented. Chapter 4 summarizes the content of this thesis before the results of the analysis are discussed. Finally, possible further measures will be outlined which could improve the event transport through the event interface.

2 The Playback Executor

2.1 Function Specification

Communication with the HICANN takes place via a specially designed FPGA (Field Programmable Gate Array) containing various interacting modules which are illustrated in Figure 1. The intended function of the Playback Executor is thereby to control the bidirectional data transfer between the user and the chip. It will receive sequences of encoded index-data pairs from the network, which need to be decoded and then either directly forwarded to the appropriate downstream module or modified beforehand. The decoded index of an index-data pair thereby indicates which exact action is to be executed. Conversely, the downstream modules will also deliver data to the Executor that originates from the HICANN. This data is to be encoded and forwarded to the network along with an index that provides information on the type of the data. The Executor must be able to handle the simultaneous arrival of data from multiple modules, taking different data priorities into account.

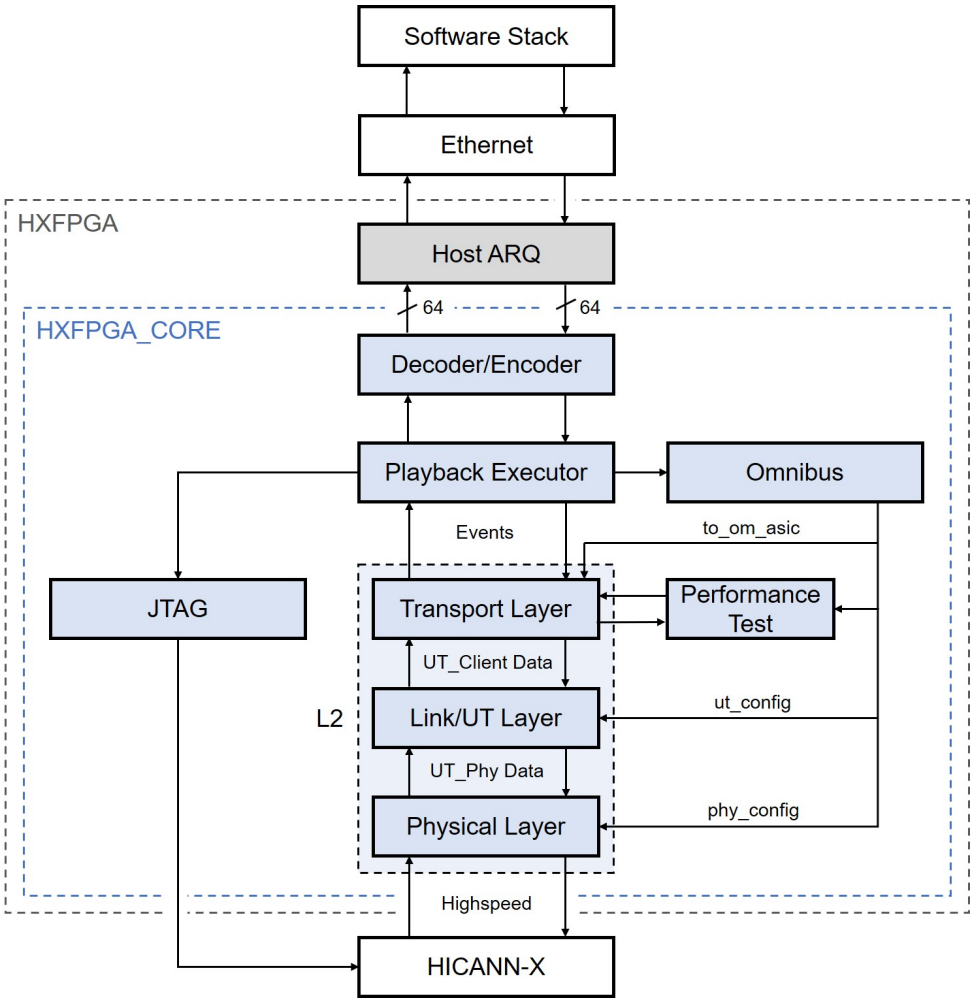

Figure 1: Integration of the Playback Executor into the FPGA communication infrastructure.

Since the HICANN will be used to perform spike experiments where the timing of individual spikes is of central importance, precise time tracking must be ensured. This requires a global time counter as well as a mechanism for the synchronization of the HICANN and FPGA time. The Executor must regularly send an update of the current system time to the user so that he can classify all received data into a chronological order. The reduction of the transmission capacity for payload data by sending system time updates is to be kept as small as possible while at the same time a sufficiently accurate time tracking must be guaranteed. The modification of index-data pairs sent by the network as mentioned in the previous section primarily affects the so-called *fire* index-data pairs, which trigger the spiking of selected neurons on the HICANN. They must be equipped with a current timestamp before forwarding, which is used to align the latency of all *fire* signals. The spike data returned by the HICANN contain timing information as well, which must be retained by the Executor until it can be forwarded to the network.

2.2 Data Traffic Regulation

For simplification, the direction of the data flow from the user to the HICANN is hereinafter referred to as downstream and the reverse direction accordingly as upstream.

2.2.1 Regulation of Downstream Data Traffic

In the direction to the HICANN, the Executor only receives data from the Host ARQ. Each data package is thereby tagged with an index, which corresponds to an instruction that is to be executed. The instructions for downstream data can be classified into five different categories:

1. *To_jtag* instructions: They trigger a simple forwarding of the supplied data to the JTAG interface. The latter transmits data to and from the *ut_jtag_driver* which operates the JTAG TAP (Test Access Port) on the HICANN. Since the *ut_jtag_driver* is able to perform four different operations [6], the Executor must support an appropriate number of instructions. In each case, the received index and data package (if provided) are forwarded to the JTAG interface.

2. *Exec_timing* instructions: They initiate timing related processes inside the Executor. This includes:

   - A *reset_time* instruction: It initiates the reset of a sleep counter needed for the *wait_until* instruction.
   - A *wait_until* instruction: It causes the Executor to pause accepting new data packages from the Host ARQ until the sleep counter reaches a specified value. This instruction is very important for generating spike sequences with defined waiting times between individual *fire* commands.

- A *systime_init* instruction: It triggers the synchronization of the HICANN and FPGA system time. As part of this synchronization, the Executor resets the FPGA system time and sends a signal to the HICANN, which then resets the HICANN internal time and sends a response signal back. The Executor then uses this response signal to align both times. The exact implementation of the synchronization mechanism is described in Chapter 2.3.

3. *Exec_system* instructions: They currently include one instruction that triggers the reset of all modules downstream of the Executor including the HICANN.

4. *To_l2* instructions: They trigger a forwarding of the supplied data to the L2 module and comprise the previously mentioned *fire* instructions. These *fire* instructions contain information about which neurons on the HICANN are to be stimulated and which of the four SPL1 links in the HICANN is to be used for transmitting the *fire* signal to the respective neuron. The implemented FPGA design allows to send up to three *fire* commands per data package. The number of *fire* commands contained in a data package is communicated to the Executor via the supplied index. As a result, three different indices and thus three different *fire* instructions are required. Before the Executor forwards a received *fire* data package to the L2 module, a current timestamp is attached to each *fire* command contained in the package, which is used to minimize the jitter (i.e. variations of the signal transmission times) on the way from the Executor to the HICANN. This is explained in detail in Chapter 2.4.1.

5. *To_omnibus* instructions: They also trigger a simple forwarding to the omnibus tree on the FPGA, which is used to send data to different modules or to read back data from these modules. For this purpose, a data package with the target address of the relevant module and a further bit that signals a read or write operation, is sent to the Executor. In the case of a write operation, a second package with the actual data follows. The Executor forwards the packages to the omnibus tree, which finally transports them to the desired module. This transmission channel is used, for instance, to set up the highspeed connection between the FPGA and the HICANN.

An overview of the described instruction types for downstream data is shown in Figure 2.

If the Host ARQ sends a data package destined for a module which currently cannot accept new data because it is still busy with processing already received data, the Executor must refuse to accept the new data package until the target module has free capacity again. Otherwise the already received data would be overwritten by the new data and would thus be lost. This is avoided by using so-called blocking ut-interfaces which are integrated between the Executor and most of the adjacent modules. In addition to the *idx* and *data* bus for transmitting the indices and data, these ut-interfaces also feature a *valid* and a *next* line which are used by the

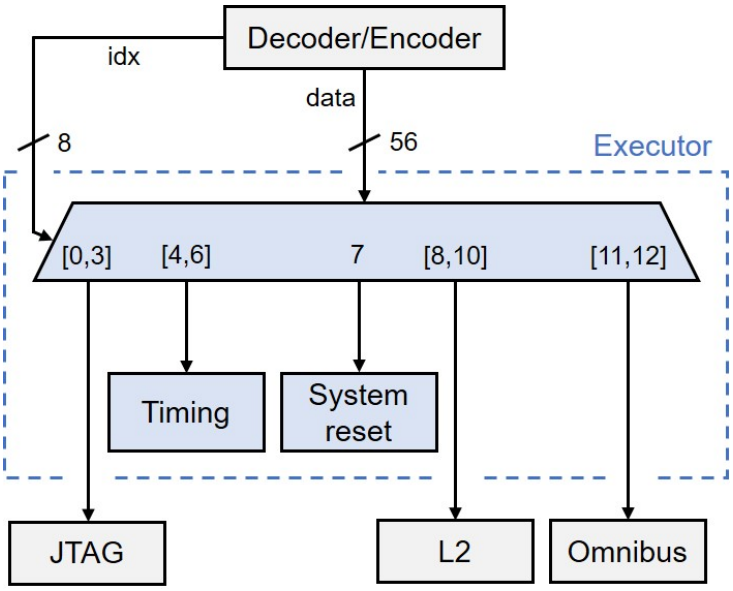

Figure 2: Downstream instruction types.

involved modules to coordinate the data transfer: If the sender module wants to transmit data to the receiver module, it sends a *valid* signal. If the receiver module can accept the data, it communicates this by sending back a *next* signal. Only when this valid-next-handshake is completed, the actual data is transmitted [5]. In the RTL code, the ut-interfaces are connected in a way that the *next* signal of the interface between the Decoder and the Executor is triggered by the *next* signal of the interface between the Executor and the corresponding downstream module. This ensures that the Executor only accepts a new data package intended for a subsequent module if the latter can receive further data.

2.2.2 Regulation of Upstream Data Traffic

In upstream direction the Executor receives data from three different modules. If several modules are sending data simultaneously, the Executor must give priority to one module, since it can only forward one data package per clock to the Host ARQ. The JTAG interface and the omnibus tree on the FPGA exclusively transport data that originates from clocked and thus digital modules. By using blocking ut-interfaces on the entire transport route from these modules to the Executor, it can be ensured that even in case of a congestion no data is lost. In contrast, the spike data is generated by neurons, which are implemented on the HICANN as analog circuits and are therefore not controlled by a clock. If this spike data is not directly forwarded after its generation, it is lost. Even the exclusive use of blocking ut-interfaces could not prevent this, because in case of a congestion in the entire digital part of the transport route, the newly generated spike data would already be lost in the HICANN at the transition from the analog to the digital area. The spike data, which is forwarded to the Executor via the L2 module, is therefore assigned the highest priority in order to minimize the probability of a congestion during transport. The data arriving via the JTAG interface and the omnibus tree are treated secondarily.

However, they have the same priority relative to each other.

The data sent to the Host ARQ must be arranged in a chronological order. The data packages themselves however do not contain any information about the time of their arrival at the Executor: Although the spike data is provided with a timestamp, that timestamp only documents the time of the registration of a spike event at one of the four SPL1 links in the HICANN. Consequently, the Executor must regularly send updates of the current system time to the Host ARQ in addition to the payload data coming from the modules. The sequence in which the various data are sent to the Host ARQ is determined using a finite state machine (FSM) in combination with an arbiter, as illustrated in Figure 3.

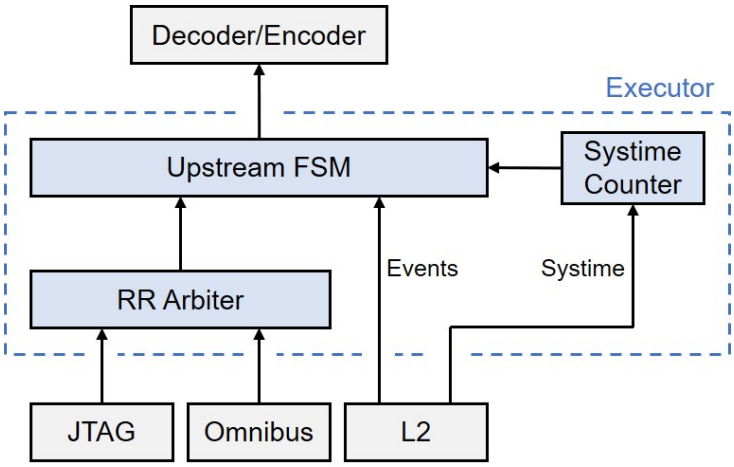

Figure 3: Regulation of upstream data traffic.

If the JTAG interface and the omnibus are sending data at the same time, the arbiter determines in which order the data is forwarded to the FSM. The arbitration happens according to the round-robin method, i.e. the two modules alternately get access to the FSM and are therefore treated equally. The FSM thus receives JTAG or omnibus data transmitted by the arbiter as well as spike data (hereinafter referred to as event data) from the L2 module. When synchronizing the HICANN and FPGA time, the latter also forwards the response signal returned by the HICANN, which is used to adjust the system time counter.

The FSM comprises an *Idle*, a *Time* and a *Data* state. After a system reset and when no upstream data is present at the Executor, the FSM is in the *Idle* state. As soon as at least one of the three input modules is sending data, a transition to the *Time* state is made. In this state, the Executor sends either the full current system time or a system time differene. The latter is determined by means of a separate counter which is reset during a system reset and at each transition of the FSM from the *Data* to the *Idle* state and thus indicates the time span between the last completed data transfer and the transmission of the new data package. The counter is eight bits wide and can therefore measure a time difference of up to 255 FPGA clocks. Unless this maximum value is reached, the system time difference is sent instead of the full system time

as it occupies a smaller number of bits. At the software level, it can be added to the previous system time value to obtain the absolute time of transmission of the new data package. If the counter however has reached the maximum value or no data package has been transferred to the Host ARQ since the system reset, the full system time is sent. As soon as the Decoder/Encoder initiates the transfer of the system time update by sending the *next* signal, the FSM switches to the *Data* state. In this state, the data packages sent by the arbiter and the L2 module are forwarded to the Decoder/Encoder taking their priority into account. Under normal circumstances, one data package is sent every FPGA clock until no more data is available. Consequently, the system time at the software level can be updated by a simple incrementation each time a new data package is received, without having to rely on interim system time updates from the Executor. This keeps the transmission capacity for payload data at a maximum. If not only the L2 module but also the JTAG interface or the omnibus are sending data to the Executor, the event data is always forwarded first due to its higher priority. If the L2 module permanently sent event data, the arbiter data would be held back until the event data stream stops (i.e. until the L2 module does not send any event data for at least one clock). In order to avoid very long waiting times of the arbiter data, an *event_ratio* was introduced, which allows to regulate the ratio of event data to arbiter data. For this purpose, an *event_ratio* counter is incremented after each forwarded event data package. When the value of this counter corresponds to the specified *event_ratio*, the event data stream is interrupted for one clock to transmit an arbiter data package. The counter is then reset and the event data stream is further processed. If no arbiter data is available at the time of reaching the *event_ratio*, the counter is also reset and the event data stream is further processed without interruption. For the currently used FPGA configuration an *event_ratio* of 255 was selected. Once the data packages of all modules have been forwarded (i.e. there is no data at any of the input ports for at least one clock), the FSM returns to the *Idle* state.

In case of a very long coherent event data stream towards the Executor, a congestion above the Executor can occur. It is then no longer possible to send a data package every clock. When at the software level a data package is received again after an interruption due to a congestion, the system time is only increased by one, although the actual system time in the Executor may already have advanced by several clocks. With increasing congestion length the software and hardware system times would diverge further and further. For this reason, a transition of the FSM from the *Data* state back to the *Time* state was inserted: Whenever the Executor wants to send data in the *Data* state (encode.valid), but the Decoder/Encoder cannot receive data due to a congestion (!encode.next), the FSM moves back to the *Time* state and sends a system time update at the next opportunity, before it returns to the *Data* state and continues with the forwarding of event data. This keeps the software and hardware system times synchronous, which, however, reduces the transmission capacity for payload data and leads to an increase in the event droprate. The described state transitions of the FSM are shown again in Figure 4.

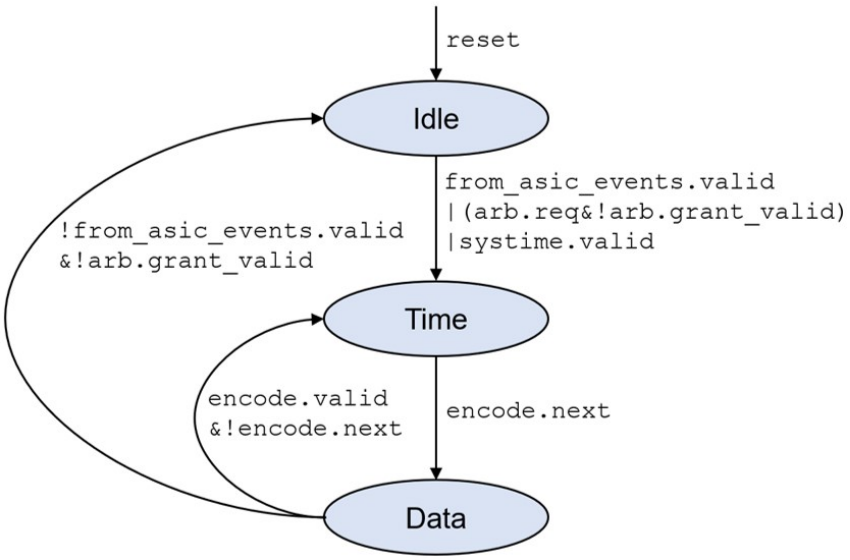

Figure 4: State transitions of the upstream finite state machine.

2.3 Synchronization of HICANN and FPGA Time

The Executor and the digital part of the HICANN are clocked by separate clocks with different frequencies. Both are equipped with counter modules, which are incremented at each clock and thus generate their own system time. In order to ensure an exact timing when stimulating the neurons and to be able to trace their spike behaviour, however, a global system time must be defined from which the FPGA and HICANN system times can be derived. Since communication between the Executor and the Host ARQ happens according to the FPGA time, the latter was selected as the global system time. To determine the HICANN time at a given FPGA time and vice versa, both times must be synchronized. The synchronization process which is triggered by the *systime_init* instruction described in Chapter 2.2.1 is shown in Figure 5a: Before synchronization ($t < t_0$), both system times are in an unknown, arbitrary relation to each other. When the Executor receives the synchronization command from the Host ARQ in the form of the corresponding index at time t_0, it resets the FPGA time and sends a signal towards the HICANN. This signal arrives at the HICANN at time t_1, which then resets its system time and sends a response signal back. Shortly after the reset, the HICANN time is yet smaller than the FPGA time, as the latter was already restarted at time t_0. But unlike the FPGA clock, which has a frequency of only 125 MHz, the HICANN clock runs with a frequency of 250 MHz. Assuming identical signal transmission times Δt from the Executor to the HICANN and the way back, both system times must match exactly when the response signal arrives at the Executor at time t_2, since the FPGA time has been running twice as long, but only half as fast as the HICANN time. If now the FPGA time is halved, this corresponds to a displacement of its reset time from t_0 to t_1, which thus is identical for both system times. Consequently the HICANN time is from now on exactly twice as large as the FPGA time, which allows for an easy conversion.

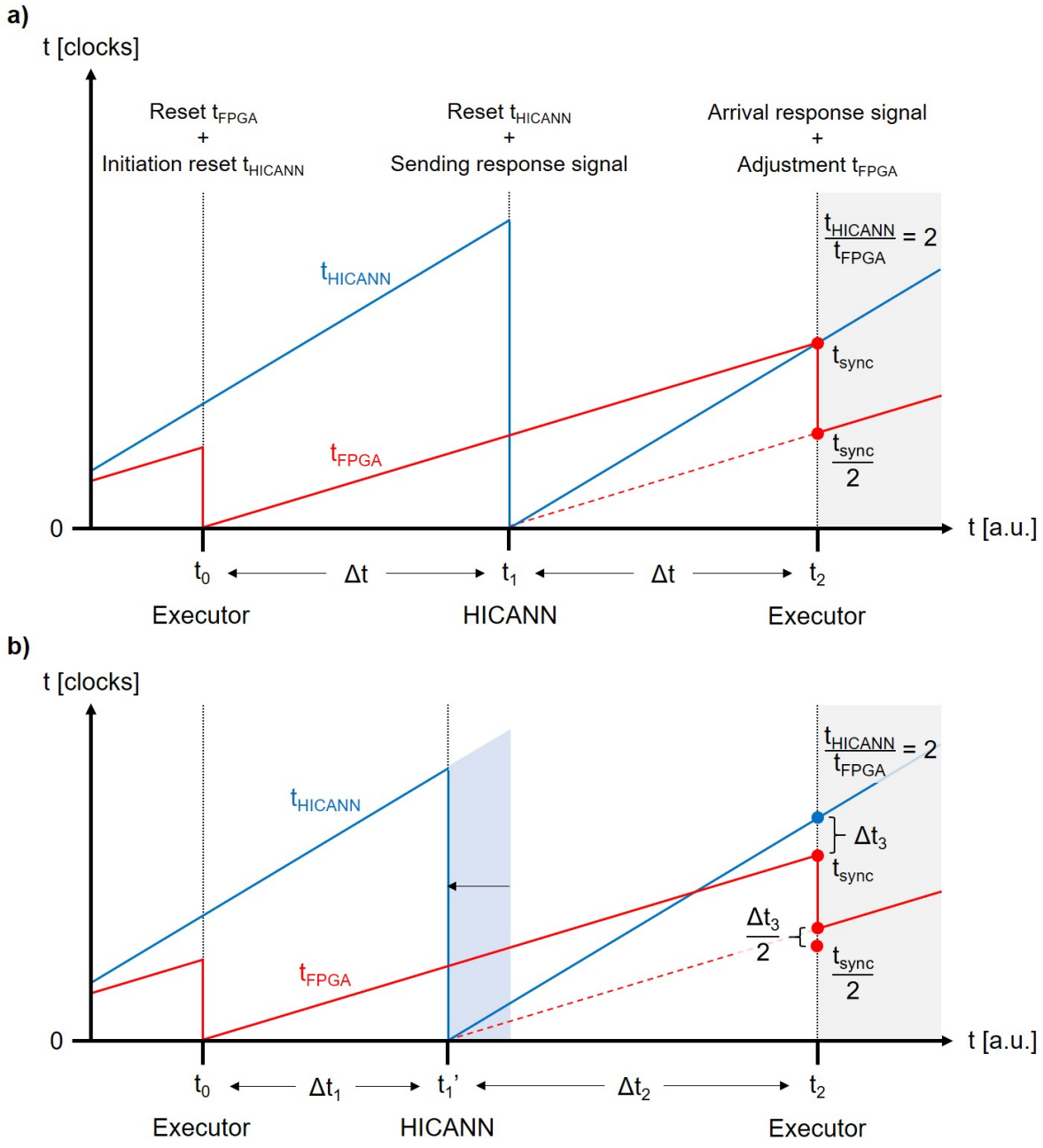

Figure 5: Synchronization of HICANN and FPGA time.

If, on the other hand, the signal transmission times for both ways are different, HICANN and FPGA time will not be identical at time t_2. As it turned out during the implementation of the synchronization process, the signal transmission time from the Executor to the HICANN is smaller than for the opposite direction (Figure 5b). In this case, the HICANN time lies above the FPGA time at time t_2. In order to still obtain a time ratio of exactly two directly after synchronization, half of the difference Δt_3 between the HICANN time and the unmodified FPGA time (t_{sync}) must be added to the halved FPGA time ($t_{sync}/2$). During implementation a difference of $\Delta t_3 = 10$ FPGA clocks could be determined. For the adjusted FPGA time at time

t_2 therefore applies:

$$t_{FPGA}^{adj} = \frac{t_{HICANN}}{2} = \frac{t_{sync}}{2} + \frac{t_{HICANN} - t_{sync}}{2} \tag{1}$$

$$= \frac{t_{sync}}{2} + \frac{\Delta t_3}{2} \tag{2}$$

$$= \frac{t_{sync}}{2} + 5 \tag{3}$$

2.4 Timing of Spike Events

2.4.1 Timestamping of Fire Commands

The transmission time of a *fire* data package from the Executor to the HICANN depends on the data traffic density as well as on the number of *fire* commands contained in the data package. To be able to stimulate the neurons in the HICANN independently of these factors with the time intervals specified via the *wait_until* instruction, all *fire* commands of a data package are equipped with a timestamp in the Executor before forwarding (Figure 6). This timestamp contains the lower eight bits of the current FPGA time converted to HICANN time. Compared to the full FPGA time, this saves 34 bits per *fire* command, allowing the *fire* data package to be transported much faster to the HICANN via one of the highspeed lines. When the package arrives at one of the L2 links in the HICANN at time $t_{receive}^{L2}$, the provided timestamp is used to determine the time t_{send}^{Exec} at which it was submitted at the Executor. To this submission time, a configurable timespan Δt_{spec} is added. This yields an earliest time $t_{release}^{L2}$ at which the *fire* commands contained in the package are forwarded to the corresponding SPL1 link [8]:

$$t_{release}^{L2} = t_{send}^{Exec} + \Delta t_{spec} \tag{4}$$

If the transport time of the package from the Executor to the L2 links in the HICANN is below Δt_{spec}, $t_{receive}^{L2}$ will be smaller than $t_{release}^{L2}$. In this case, the *fire* commands contained in the package are held back until $t_{release}^{L2}$ is reached. Otherwise they are directly forwarded because $t_{release}^{L2}$ is already exceeded at their arrival. The timespan between leaving the Executor and forwarding at one of the L2 links in the HICANN is therefore the same for all *fire* commands regardless of the abovementioned factors, as long as their transport time is less than Δt_{spec}. Δt_{spec} can thus be used to determine to which extent the jitter on the way from the Executor to the L2 links in the HICANN should be reduced: The larger Δt_{spec} is chosen, the lower the jitter will be, but the smaller will also be the maximum data transfer rate, since fewer *fire* commands per time unit can be transported due to the longer delay at the L2 links.

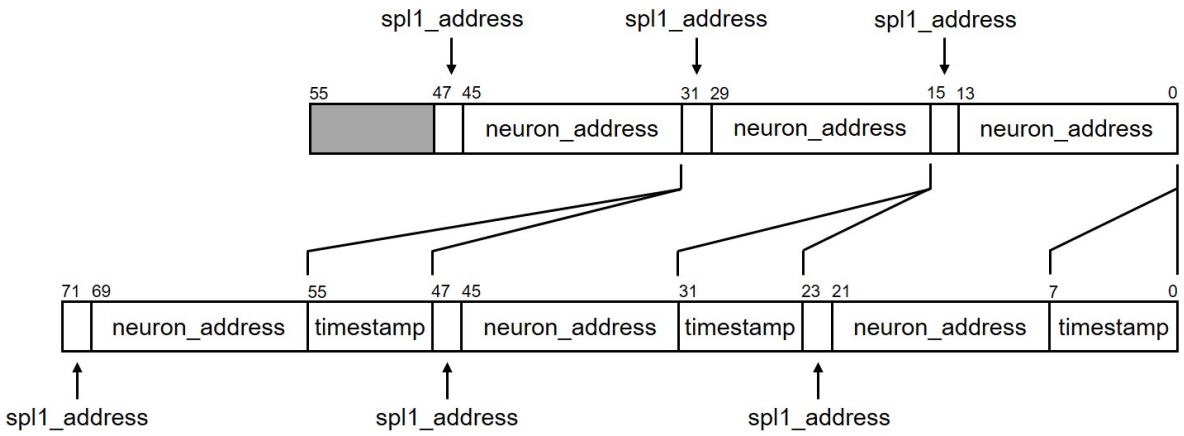

Figure 6: Timestamping of *fire* commands.

2.4.2 Reconstruction of Event Times

To be able to trace the spike behaviour of the neurons, their event data is also timestamped on the way to the Executor when arriving at the SPL1 links in the HICANN. In favour of the data transfer rate, the timestamp again contains only the lower eight bits of the current HICANN time. The time at which the timestamp is attached is hereinafter referred to as event time t_{event} and is of central importance for the analysis of the performed spike experiments. When an event data package arrives at the software level at time $t_{receive}$, the event times of the contained events can be reconstructed using $t_{receive}$ and the attached timestamps. The reconstruction can be understood with the help of Figure 7: The left side of Figure 7a shows as an example the lower bits of an event time in FPGA time. The lower seven bits thereby correspond to the converted timestamp, since the highest bit of the original timestamp is omitted during conversion from HICANN to FPGA time. While the timestamped event data package is transported from the SPL1 link in the HICANN to the software level, the lower bits of the system time change by Δt (Figure 7a on the right). If now the lower seven bits of $t_{receive}$ are replaced by the bits contained in the timestamp, the progress of the system time is undone, resulting in the event time again.

If, however, the event data arrives at the SPL1 links shortly before a rollover of the lower seven bits of the system time (Figure 7b on the left), more significant bits (highlighted in red) will also have changed at time $t_{receive}$. In principle, arbitrary more significant bits of the system time register can change, depending on the bit constellation at time t_{event}. In this case, replacing the lower seven bits with the timestamp is not sufficient to reconstruct t_{event}. Then the rollover must also be undone by changing the more significant bits (i.e. all bits above the seventh bit) so that the number formed from them is one smaller than before the change. Whether this additional operation is required can be determined by comparing the timestamp with the number formed by the lower seven bits of $t_{receive}$: If the timestamp is smaller (see Figure 7a), it is sufficient to replace the lower seven bits. If it is larger (see Figure 7b), a rollover must inevitably

have occurred, which must also be corrected accordingly.

However, the reconstruction using this comparison only works correctly as long as Δt is smaller than Δt_{crit} = 128 FPGA clocks. At longer time spans, which can occur in case of a congestion between the SPL1 links in the HICANN and the Host ARQ, at least one bit above the seventh bit will always change. If then in addition the timestamp is smaller than the number formed from the lower seven bits of $t_{receive}$ (see Figure 7c), the comparison erroneously yields that no rollover has occurred. Consequently, only the lower seven bits of $t_{receive}$ are replaced. In order to avoid these faulty reconstructions, the more significant bits of t_{event} and $t_{receive}$ would also have to be compared. But since the event data packages do not contain any information about the more significant bits of the event time, this is not possible. For this reason, a large congestion between the SPL1 links and the Host ARQ should be avoided if possible.

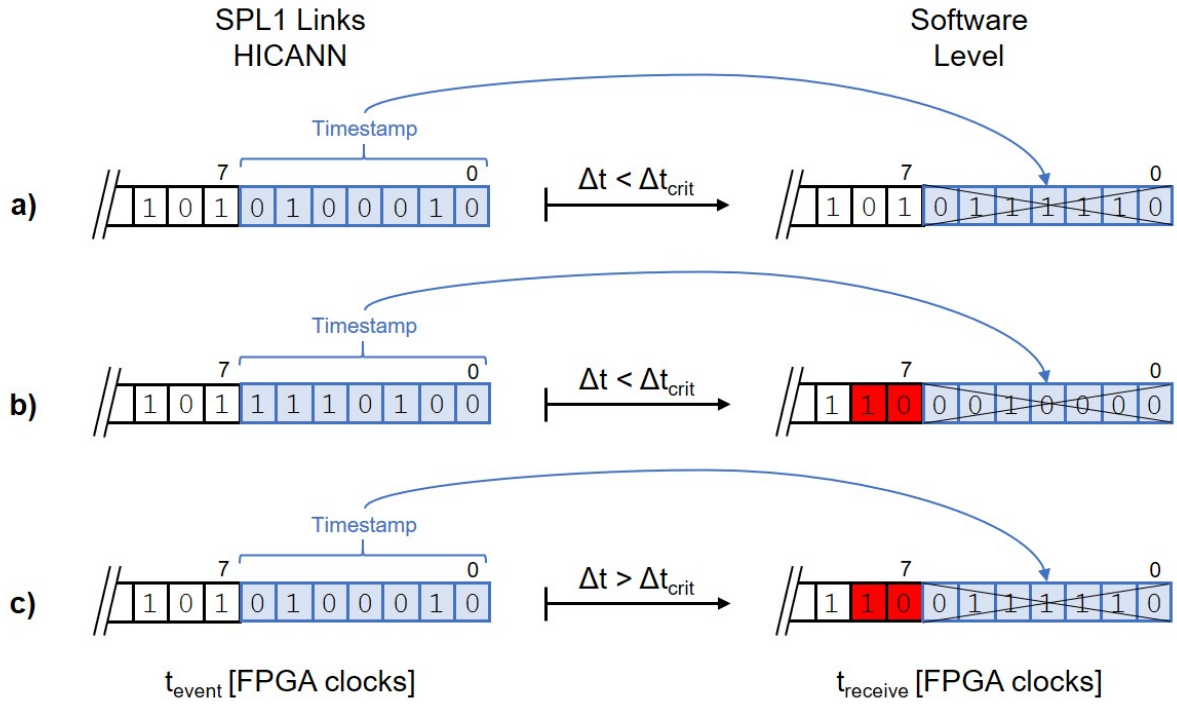

Figure 7: Reconstruction of event times.

3 Characterization of the Event Interface

The Executor was the last missing component of the FPGA communication infrastructure. Its implemented functions now enable in particular the time-coordinated sending of *fire* commands to the HICANN as well as the reception of the returned event data and their classification into a chronological sequence. This allows to investigate to what extent the event interface consisting of the FPGA communication infrastructure and the digital part of the HICANN fulfills the criteria which are relevant for performing spike experiments. This investigation is dealt with in this chapter.

3.1 Content of the Analysis

The goal of the analysis is to gain an exact knowledge of how the droprate and jitter of the system consisting of the FPGA communication infrastructure and the digital part of the HICANN behave at different input spike rates, spike distributions and values of Δt_{spec} (see Chapter 2.4.1). In particular, the maximum data transfer rate that can be achieved with a negligible droprate and jitter is to be determined. For this purpose, all FPGA modules and the relevant digital part of the HICANN are tested using HDL simulations. For the tests, input spike trains are generated at the software level, which contain a large number of *fire* times in ascending order. These input spike trains are then modified according to a certain system in order to adapt them to the specific hardware properties of the FPGA communication infrastructure (see Chapter 3.2.2). At the times t_{send} contained in the modified input spike trains, spikes are then submitted in the form of *fire* data packages (see Chapter 3.2.3). Each spike i is thereby assigned a different neuron address, which corresponds to the position of $t_{send,\,i}$ in the modified spike train. The route of the *fire* data packages through the simulated system is shown in Figure 8: Arriving at the Decoder/Encoder, they propagate through the FPGA communication infrastructure and via the highspeed connection to the HICANN. On the HICANN they pass through the L2 module and the routing interface between the L2 module and the SPL1 level. Behind the SPL1 links they are looped back and timestamped (see Chapter 2.4.2) when passing the links again at time t_{event}. Then they propagate back through the system and arrive at the software level again at time $t_{receive}$.

By comparing the number of sent and received spikes the droprate can be determined (see Chapter 3.2.4). Since the received spikes also contain the neuron address assigned to them during submission in addition to the timestamp, it is moreover possible to check which spikes were lost on the way. Furthermore, each received spike i can be assigned its submission time $t_{send,\,i}$, its event time $t_{event,\,i}$ and its reception time $t_{receive,\,i}$. From these times it is possible to determine the time spans needed to pass through the corresponding sections of the system (see Chapter 3.2.4). By comparing the time spans of different spikes of the same spike train and by comparing the average time spans of spike trains with a different (mean) spike rate, it is finally possible to make statements about the jitter behaviour.

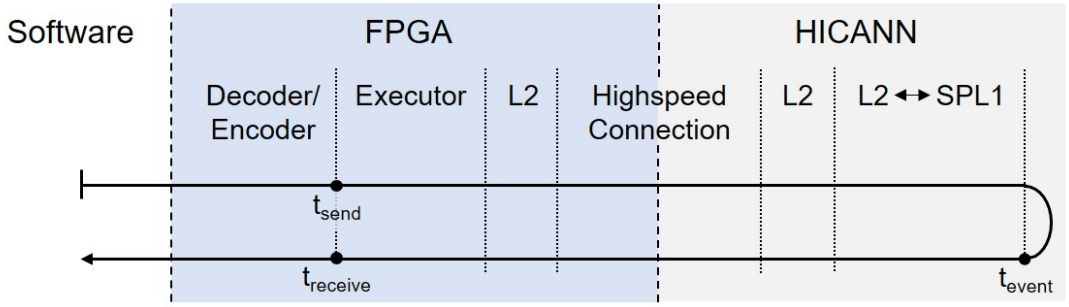

Figure 8: Event tracing.

3.2 Experimental Design

3.2.1 Input Spike Train Generation

For the tests input spike trains with Poisson-distributed ISIs (Inter Spike Intervals) are generated in order to keep the input spike sequences as divers as possible. For comparison purposes spike trains with uniform ISIs are generated as well. In both cases, the structure of a spike train is determined by indicating the (mean) spike rate and the number of spikes. The selected spike rates should cover a broad spectrum and should also contain high rates, which lead to a full utilization of the highspeed connection. In this way it can be determined whether the highspeed connection itself or other sections of the system represent the bottleneck of the communication infrastructure. Since each *fire* data package contains exactly eight further bits in addition to the timestamps, neuron and l1 addresses, up to four bytes are required to transmit the data for one spike. As the maximum data transfer rate of the highspeed connection is 1 GB/s, it can already be saturated from a critical spike rate of

$$R_{crit}^{RT} = \frac{1\,GB/s}{4\,B} = 250\,MHz. \tag{5}$$

However, it must be taken into account that the spike behaviour of the neurons implemented on the HICANN is accelerated by a factor of 10^3 compared to their biological equivalents. Since the future spike experiments will investigate the behaviour of the neurons on a biological time scale, the spike times of the input spikes will be formulated in biological time. For this reason, the spike times and spike rates defined in this analysis are also based on the biological time scale. The following applies when converting the spike rates from real time to biological time:

$$R^{Bio} = \frac{R^{RT}}{1000} \tag{6}$$

The critical spike rate in biological time is thus 250 kHz. Accordingly, the biological spike rate for the tests will be increased from 5 to at least 250 kHz in 5 kHz steps.

For the jitter analysis a maximum number of spikes per spike train of 2^{14} - 1 = 16,383 is

defined, which results from a maximum admissible neuron address length of 14 bits (see Figure 6). In this way, confusions in the allocation of submission and reception times of the spikes due to repeatedly assigned neuron addresses are avoided. For an easier automation of the test execution, the number of spikes of a spike train n is chosen in such a way that for each spike train the same time is needed to send the contained spikes, independent of its spike rate. For n, the following applies:

$$n = R^{Bio} * t^{Bio} = R^{RT} * t^{RT} \tag{7}$$

The duration t^{Bio} is determined by specifying the number of spikes n_{crit} of the 250 kHz spike train. For this a value of n_{crit} = 12,500 was chosen which leads to a duration of:

$$t^{Bio} = \frac{n_{crit}}{R^{Bio}_{crit}} = \frac{12,500}{250\ kHz} = 0.05\ s \tag{8}$$

The number of spikes as a function of the biological spike rate is therefore:

$$n(R^{Bio}) = R^{Bio} * 0.05\ s \tag{9}$$

For the droprate analysis, the number of spikes per spike train is also determined using equation (7). It may also exceed the specified maximum value of 16,383 since the allocation of submission and reception times is irrelevant for determining the droprate. Accordingly, higher input spike rates may also be chosen. Although it can be assumed that at higher input spike rates than the critical spike rate the droprate will increase significantly, the biological spike rate is varied up to 500 kHz in order to illustrate the development of the droprate.

3.2.2 Input Spike Train Modification

The Executor is clocked by the FPGA clock and thus accepts data packages only at positive edges of the clock signal, i.e. every 8 ns (real time) or 8 µs (biological time). The spikes can therefore not be sent exactly at the generated spike times, but only at times that correspond to a multiple of the duration of a clock cycle. In order for the time spans calculated during the analysis to reflect the actual duration of the transport of the *fire* data packages through the simulated system, their submission times contained in the spike trains must be adjusted accordingly. For this purpose, each submission time is rounded down to the next smaller multiple of the duration of a clock cycle in biological time. This step is hereafter referred to as *discretization* and is shown in Figure 9 for the first spikes of a fictitious Poisson input spike train. Each spike is labelled with an index i, which corresponds to the position of its spike time in the original spike train.

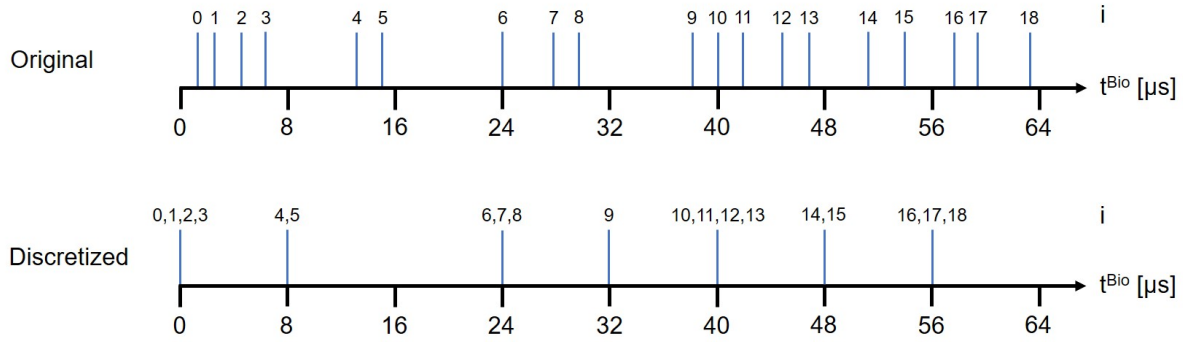

Figure 9: Spike train discretization.

However, at high input spike rates not all spikes will be submitted at the times contained in the discretized spike train. This is for two reasons:

1. The Executor can accept and forward a maximum of three *fire* commands per clock cycle. If, after the discretization, more than three spikes share the same submission time, the excess spikes will inevitably be submitted at later times.

2. The highspeed connection is operated from the FPGA side with a frequency of 125 MHz (real time). From the maximum data transfer rate of the highspeed connection it follows that per clock cycle a data volume D of

$$D = \frac{1\ GB/s}{125\ MHz} = 8\ B \tag{10}$$

can be transferred to the HICANN and thus on average only two spikes per clock cycle can be forwarded from the Executor to the highspeed connection. Though there are buffer stages integrated between the Executor and the highspeed connection, so that short-term exceedances of this average number do not affect the submission times. A longer overrun, however, causes a congestion from the highspeed connection to the Executor, which can then no longer accept new spikes every clock cycle. Then subsequent spikes will be delayed.

For spike trains with uniform ISIs, these delays will only occur above an input spike rate of two spikes per clock cycle, which is equivalent to the critical biological spike rate of 250 kHz. Since, for the jitter analysis, the (mean) biological spike rate is only varied up to 250 kHz, the uniform spike trains will not be subject to any delays. The Poisson spike trains, on the other hand, can show sporadic biological spike rates above 250 kHz already at average spike rates of less than 250 kHz. They may well be subject to delays, which distort the calculated time spans. To prevent this, the spike distributions of the discretized input spike trains are modified in such a way that no delays occur at high spike rates.

With this modification different approaches can be pursued, which differ on the one hand in which spikes are modified. The following two options can be considered:

1. A maximum of two spikes per clock cycle is admitted. The times of all other spikes will be modified.

2. Temporarily up to three spikes per clock cycle are admitted. By specifying a maximum admissible number of spikes within a certain period it is ensured, however, that related to this period on average a maximum of two spikes per clock cycle are submitted. The times of those spikes which are still within the specified period, but would increase the average spike rate to over two spikes per clock cycle, will be modified.

The first option has the disadvantage that the variability of the ISIs of Poisson spike trains with high spike rates will be partly lost, because sporadic spike rates of more than two spikes per clock cycle are suppressed. Therefore, the second option was chosen to determine the spike times to be modified for this analysis. From the size of the buffer stages between the Executor and the highspeed connection, a maximum admissible number of 16 spikes within eight consecutive clock cycles was derived.

Furthermore, there are several ways to modify the affected spikes:

- *Reduction*: The affected spikes are removed from the spike train. This causes some spikes to be lost before submission, but apart from the deviations due to the discretization, the submission times of the remaining spikes correspond to the originally determined times.

- *Cascading*: The affected spikes are shifted to later submission times. Thus no spikes will be lost before submission, but the distribution of the spikes within the spike train becomes more and more homogeneous. In the worst case, the Poisson spike train corresponds to a uniform spike train after modification, which has lost its entire structure.

All modification variants result in biological spike rates of more than 250 kHz being reduced to an average of 250 kHz. With the spike trains modified via the *Cascading* method, however, more spikes are sent over a longer period at this reduced rate, since no spikes are removed during modification, but only shifted to later times. If there are losses when passing through the system, these losses will be higher than with the spike trains modified via the *Reduction* method, since the same loss rate is maintained over a longer period. Thus the *Cascading* variant loses its advantage of a lower spike loss, but still has the disadvantage of a higher structural loss. For this reason the *Reduction* method was chosen for modifications within this analysis.

The detailed modification procedure is shown in Figure 10 for the first spikes of the discretized fictitious spike train. The numbers above the spike indices indicate the number of spikes already contained in the 64 µs interval at the respective time. First of all it is ensured that

a maximum of three spikes share the same submission time. If there are more than three spikes, those three spikes are retained whose times in the original spike train lay before the times of the other spikes. All other spikes are removed. In the second step, the number of spikes contained in the interval is reduced to 16. Of the remaining spikes, those 16 spikes are retained whose times in the original spike train lay before the times of the other spikes. All other spikes are removed.

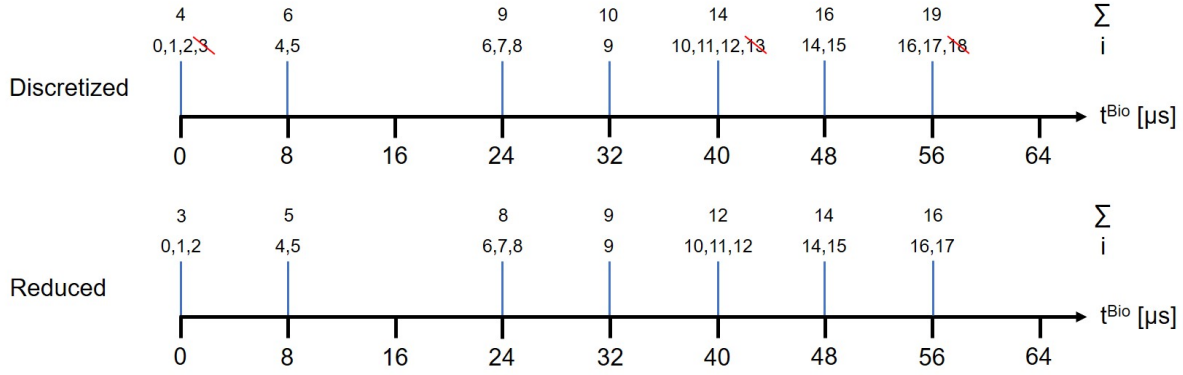

Figure 10: Spike train modification.

3.2.3 Input Spike Train Conversion

The Executor can only further process the received data correctly if the data is sent in the form of the defined instructions (see Chapter 2.2.1). Consequently, the modified spike trains must be converted into sequences of different instructions. The adherence to the defined submission times is realized with the help of the *wait_until* instruction. First a *reset_time* instruction is generated, which causes a reset of the sleep counter of the Executor. In order to send a spike i at the time $t_{send,\,i}$, a *wait_until* instruction is generated before the actual *fire* instruction, which contains the submission time converted to clock cycles. When the Executor receives this instruction, it will only accept the subsequent *fire* instruction when its sleep counter, which is incremented at every positive edge of the clock signal, has reached the communicated time.

During the generation of the *fire* instructions, each spike is assigned a neuron and SPL1 address. The neuron address is selected in such a way that it corresponds to the position of the submission time $_{send,\,i}$ of the respective spike in the modified spike train, in order to be able to determine spike specific time spans for the jitter analysis. The SPL1 address is assigned periodically from zero to three. To be able to send several spikes at the same time, their neuron and SPL1 addresses are grouped together in a single *fire* instruction (see Figure 6). The index of the instruction is selected according to the number of contained spikes (see Chapter 2.2.1).

When sending spikes at consecutive clock cycles it must be taken into account that the Executor can only accept one instruction per clock cycle. Thus, one clock cycle will inevitably

elapse between the acceptance of two consecutive *fire* instructions so that no *wait_until* instruction must be inserted in-between. This would cause the second *fire* instruction to be accepted with a delay of one clock cycle.

The conversion of the already discretized and modified fictitious spike train into a corresponding sequence of instructions is shown in Figure 11. The order in which the instructions are accepted by the Executor is indicated by the numbers on the left hand side of the instruction block. For an easier identification of the single instructions, descriptive names are used instead of the instruction indices.

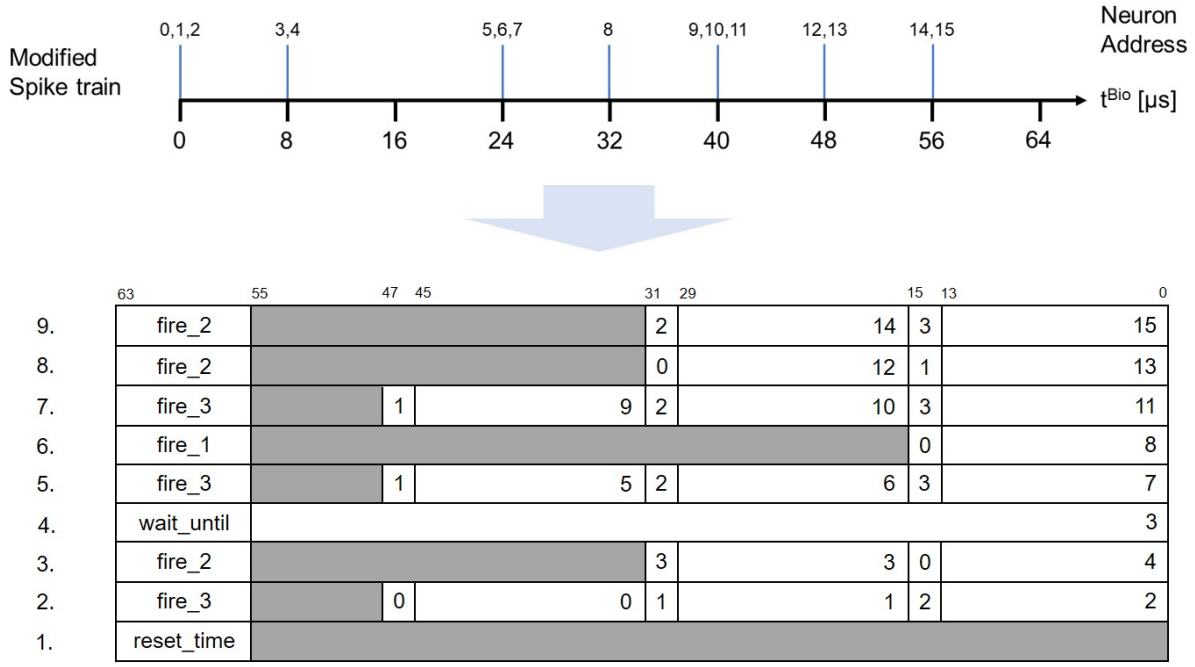

Figure 11: Spike train conversion.

3.2.4 Determination of Drop Rates and Latencies

The spikes contained in an unmodified input spike train can get lost during modification of the spike train and during transport through the system. For a more differentiated analysis, a separate droprate is determined for each loss type. In both cases the number of lost spikes is related to the number of spikes before the loss so that comparisons between different spike rates can be made. The relative droprate $R_d^{mod,\,rel}$, which indicates the spike loss due to the modification of a spike train, is:

$$R_d^{mod,\,rel} = \frac{n_{init} - n_{send}}{n_{init}} \tag{11}$$

Here n_{init} corresponds to the number of spikes contained in the original spike train and n_{send} to the number of spikes contained in the modified spike train. Accordingly, the relative droprate

$R_d^{trans,\,rel}$, which describes the spike loss during transport through the system, is:

$$R_d^{trans,\,rel} = \frac{n_{send} - n_{receive}}{n_{send}} \tag{12}$$

Thereby $n_{receive}$ corresponds to the number of spikes which were received again at the software level. From these two rates the total relative droprate $R_d^{tot,\,rel}$ can be determined using:

$$R_d^{tot,\,rel} = R_d^{mod,\,rel} + (1 - R_d^{mod,\,rel}) * R_d^{trans,\,rel} \tag{13}$$

$$= \frac{n_{init} - n_{receive}}{n_{init}} \tag{14}$$

With the help of the available timing information (see Figure 8), the following latencies Δt_i can be calculated for each spike i that neither got lost during the modification of the input spike train nor during the transport through the system:

- $\Delta t_{Exec/SPL1,\,i}$: The time span between its acceptance by the Executor after a successful valid-next-handshake between the Decoder/Encoder and the Executor and its registration at one of the SPL1 links in the HICANN.
- $\Delta t_{SPL1/Exec,\,i}$: The time span between its registration at one of the SPL1 links in the HICANN and its acceptance by the Decoder/Encoder after a successful valid-next-handshake between the Executor and the Decoder/Encoder.
- $\Delta t_{Exec/Exec,\,i}$: The time needed for a complete traverse of the system, i.e. the time elapsed between the corresponding valid-next-handshakes of the Decoder/Encoder and the Executor.

For the calculation, however, it must be taken into account that the transmission times t_{send} contained in the modified spike trains refer to the value of the sleep counter of the Executor (see Chapter 3.2.3), whereas the event times t_{event} and the reception times $t_{receive}$ refer to the value of the system time counter. Although both counters are clocked by the FPGA clock, they are reset at different times and thus generate different time bases. To be able to calculate the time spans correctly, however, all three times must be expressed in a common time base. For this purpose, it is ensured that the same time Δt_{offset} = 605 clock cycles elapses between the reset of the system time counter and the sleep counter at all tests of this analysis. Using Δt_{offset} the times can be transformed into a common time base.

The time spans are calculated in biological time. For this Δt_{offset} as well as the event and reception times are converted from FPGA time to biological time. For $\Delta t_{Exec/SPL1,\,i}^{Bio}$ thus applies:

$$\Delta t_{Exec/SPL1,\,i}^{Bio} = t_{event,\,i}^{Bio} - t_{send,\,i}^{Bio} - \Delta t_{offset}^{Bio} \tag{15}$$

$$= t_{event,\,i} * 8 - t_{send,\,i} - 4{,}840\ \mu s \tag{16}$$

As the event and reception times refer to the same time base, Δt_{offset} is not required for calculating $\Delta t^{Bio}_{SPL1/Exec,\,i}$:

$$\Delta t^{Bio}_{SPL1/Exec,\,i} = t^{Bio}_{receive,\,i} - t^{Bio}_{event,\,i} \tag{17}$$

$$= (t_{receive,\,i} - t_{event,\,i}) * 8 \tag{18}$$

The time span for a complete traverse of the system $\Delta t^{Bio}_{Exec/Exec,\,i}$ corresponds to the sum of the other two time spans:

$$\Delta t^{Bio}_{Exec/Exec,\,i} = \Delta t^{Bio}_{Exec/SPL1,\,i} + \Delta t^{Bio}_{SPL1/Exec,\,i} \tag{19}$$

$$= t^{Bio}_{receive,\,i} - t^{Bio}_{send,\,i} - \Delta t^{Bio}_{offset} \tag{20}$$

$$= t_{receive,\,i} * 8 - t_{send,\,i} - 4{,}840\ \mu s \tag{21}$$

3.3 HDL Simulation Results

3.3.1 Drop Rate

In the first part of the analysis the behaviour of the drop rate depending on the biological input event rate at different values of Δt_{spec} (see Chapter 2.4.1) was investigated for input spike trains with uniform ISIs (hereafter referred to as linear spike trains) as well as for input spike trains with Poisson-distributed ISIs.

Figure 12 shows the development of the relative drop rate $R_d^{mod,\,rel}$, which indicates the event loss due to the modification of the spike trains, and the relative drop rate $R_d^{trans,\,rel}$, which describes the event loss during transport through the system, for linear (left) and Poisson spike trains (right) at Δt_{spec} = 55.

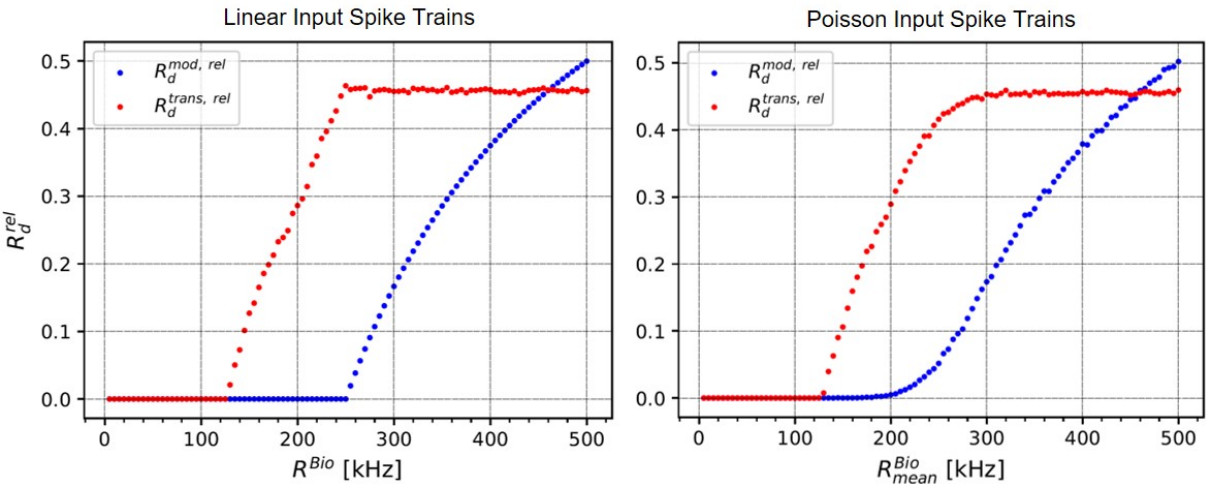

Figure 12: Drop rate development for Δt_{spec} = 55.

Both linear and Poisson spike trains lose events on the way through the system from a bi-

ological event rate of 130 kHz. From this rate on $R_d^{trans,\,rel}$ increases almost linearly with the event rate for both spike train types. For both types, the maximum value of $R_d^{trans,\,rel}$ is about 46 %. This value is reached at 250 and 300 kHz respectively, and is kept constant from then on. This behaviour can be traced back to the modification of the input spike trains: During modification, events are removed from the spike trains before transmission, so that a maximum of three events are contained in one clock cycle and a maximum of 16 events are contained in eight consecutive clock cycles (corresponding to 64 µs in biological time). This leads to modifications of the linear spike trains from event rates above 250 kHz, whereas Poisson spike trains are already modified at slightly lower mean event rates (see Chapter 3.2.2). From a certain mean biological event rate, which is hereafter referred to as saturation rate and which, for linear spike trains, is 250 kHz, for Poisson spike trains is approximately 300 kHz, each 64 µs interval of the modified spike trains will contain 16 events. Since the same spike train length (i.e. the same number of 64 µs intervals) was selected for all input event rates (see Chapter 3.2.1), the total number of transmitted events per spike train is also constant from this saturation rate onwards. Apart from the distribution of the events within the individual 64 µs intervals, the modified spike trains are thus identical from the saturation rate onwards, which also results in identical drop rates $R_d^{trans,\,rel}$. As the total number of events contained in a modified spike train is constant from the saturation rate onwards, but the number of events contained in the unmodified spike trains still increases with the input event rate (see Chapter 3.2.1), the drop rate $R_d^{mod,\,rel}$ must inevitably increase.

The drop rate behaviour at different values of Δt_{spec} is shown in Figure 13. With the linear spike trains the drop rates behave almost identical for all tested values of Δt_{spec}. This also applies to Poisson spike trains up to a value of $\Delta t_{spec} = 60$. At higher values, however, event losses during transport through the system already occur from a mean biological event rate of 65 kHz.

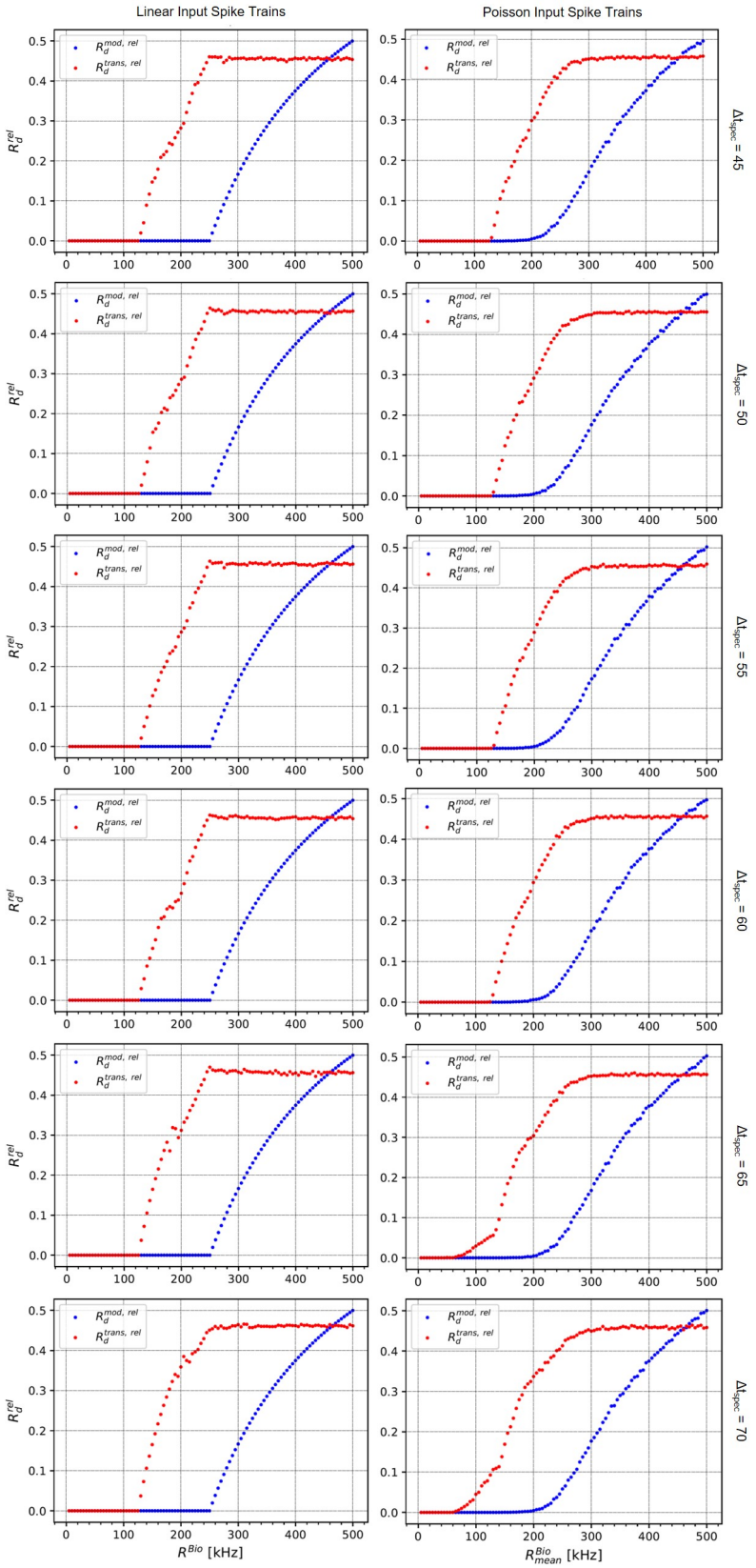

Figure 13: Drop rate development for different values of Δt_{spec}.

3.3.2 Latencies and Jitter

In the second part of the analysis, the time spans required to propagate through the different sections of the system and their variation at different biological input event rates and different values of Δt_{spec} were investigated. The analysis thereby focuses on input spike trains with Poisson-distributed ISIs.

Figure 14 shows the spread of the measured time spans depending on the mean biological input event rate at $\Delta t_{spec} = 55$. On the left side the development of the $\Delta t^{Bio}_{Exec/SPL1}$ latency is illustrated, which corresponds to the time span between the transmission of an event by the Executor and its registration at one of the SPL1 links in the HICANN in biological time. The right plot shows the $\Delta t^{Bio}_{Exec/Exec}$ latency, which indicates the biological time needed to completely traverse the system. In each plot the median ($Q_{0.5}$), the spread of the central 70 % of the measured values as well as the spread of the lower and upper 15 % of the measured values are displayed.

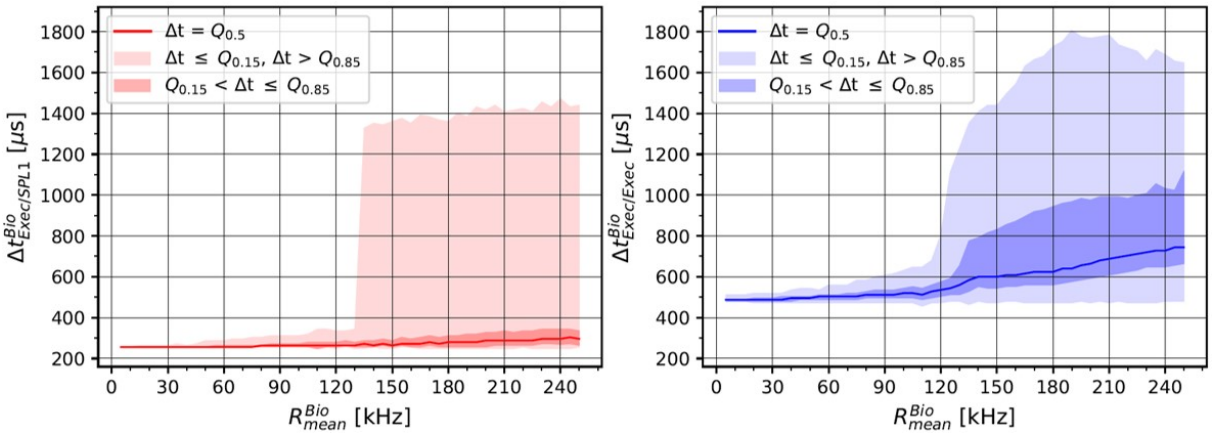

Figure 14: Latency and jitter development for $\Delta t_{spec} = 55$ (Poisson spike trains).

The median of the $\Delta t^{Bio}_{Exec/SPL1}$ values remains constant at 256 µs from 5 to 75 kHz and then increases slightly to a value of 296 µs at 250 kHz. A deviation of the central 70 % of the measured values from the median can only be observed from approximately 100 kHz. Even at 250 kHz, the central 70 % of the values are still within 264 and 344 µs. The spread of the upper 15 % of the measured values increases massively between 130 and 135 kHz, but changes only slightly before and afterwards. This means either that from biological event rates of 135 kHz onwards individual events are delayed on their way from the Executor to the SPL1 links in the HICANN for a very long time or that their time spans have been determined incorrectly. This aspect will be discussed in detail later on.

The median of the $\Delta t^{Bio}_{Exec/Exec}$ values increases only slightly from 488 to 512 µs until 110 kHz. It then increases faster up to a value of 744 µs at 250 kHz. At small biological event rates almost no spread of the measured values is recognizable. The spread of the central 70 % of the measured values increases only slightly until 125 kHz, but significantly between 125 and 135 kHz and moderately afterwards. For the spread of the upper 15 % of the values, a strong

increase can be observed in the range from 115 to 190 kHz. At 250 kHz, the central 70 % of the measured values range between 664 and 1,120 µs. The difference between the minimum and maximum value at 250 kHz is 1,168 µs.

Based on the figure, the following minimum time spans for passing through the different sections of the system can be determined:

- $\Delta t^{Bio}_{Exec/SPL1,\,min}$ = 256 µs (32 clock cycles)
- $\Delta t^{Bio}_{SPL1/Exec,\,min}$ = 224 µs (28 clock cycles)
- $\Delta t^{Bio}_{Exec/Exec,\,min}$ = 480 µs (60 clock cycles)

However, $\Delta t^{Bio}_{Exec/SPL1,\,min}$ and thus also $\Delta t^{Bio}_{Exec/Exec,\,min}$ depend on the selected value for Δt_{spec}, since the latter determines how long the events are held back at the L2 links in the HICANN when propagating from the Executor to the SPL1 links in the HICANN (see Chapter 2.4.1).

The Figures 15 to 20 show the $\Delta t^{Bio}_{Exec/SPL1}$ and $\Delta t^{Bio}_{Exec/Exec}$ latencies of individual events of input spike trains with mean biological event rates in the range between 120 and 135 kHz, in which the spread of the latencies increases most. For comparison the latencies at 5 and 250 kHz are shown as well. The figures also illustrate which events of the spike trains were lost during transport through the system and how many events were removed during modification of the spike trains. This allows to derive correlations between the latency and the occurring event losses. The figures will also be used to explain the massive increase in the spread of the upper 15 % of the $\Delta t^{Bio}_{Exec/SPL1}$ values between 130 and 135 kHz.

At 5 kHz (Figure 15) the $\Delta t^{Bio}_{Exec/SPL1}$ latency of all events is at the minimum value of 256 µs. Apart from three events, the $\Delta t^{Bio}_{Exec/Exec}$ time span also varies between its minimum value of 480 and 488 µs, whereby a regular variation pattern can be seen: All events which were assigned the SPL1 addresses zero or one require exactly one clock cycle more for covering the distance from the SPL1 links in the HICANN to the Executor than those events which were assigned the SPL1 addresses two or three. Since spike train modifications are only made at significantly higher event rates, all generated events are transmitted. During transport no events are lost either.

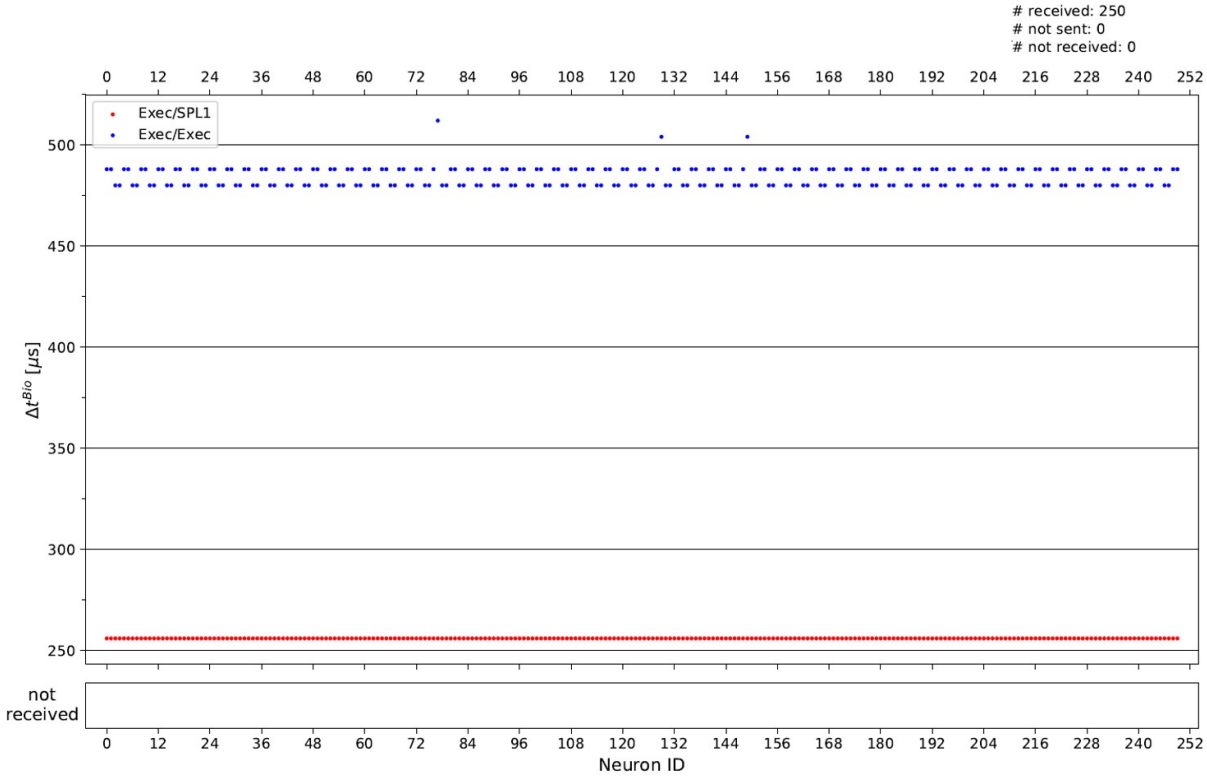

Figure 15: Latencies per event for a 5 kHz Poisson spike train (Δt_{spec} = 55).

At 120 kHz (Figure 16), variations in both latencies can already be clearly observed. However, the variations of the $\Delta t^{Bio}_{Exec/SPL1}$ latency are lower and more homogeneous than those of the $\Delta t^{Bio}_{Exec/Exec}$ latency. With the latter, outlier groups of values around 700 µs have formed across the spike train. A single event even has a $\Delta t^{Bio}_{Exec/Exec}$ latency of 808 µs. So far no events have been lost.

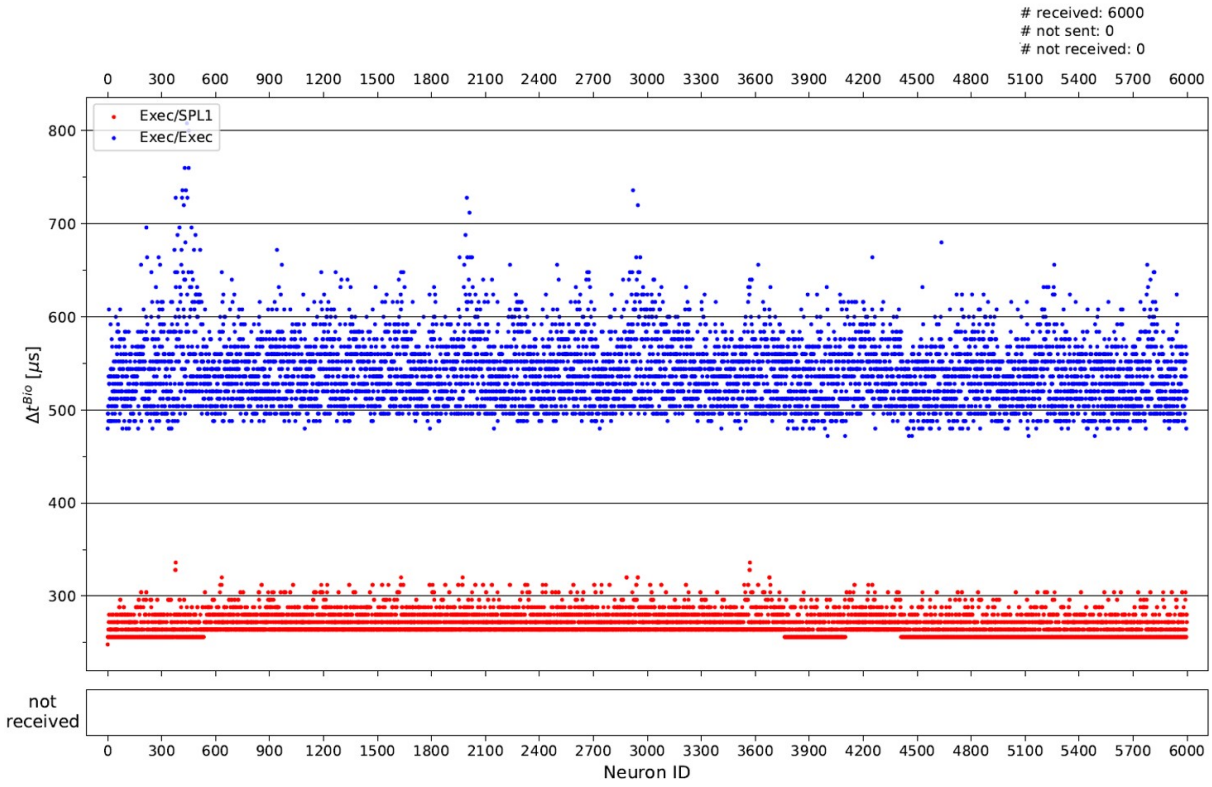

Figure 16: Latencies per event for a 120 kHz Poisson spike train (Δt_{spec} = 55).

At 125 kHz (Figure 17) the spread of the $\Delta t^{Bio}_{Exec/SPL1}$ values has hardly changed compared to 120 kHz. With the $\Delta t^{Bio}_{Exec/Exec}$ values, on the other hand, even more outlier groups have formed, which partly contain events with latencies of over 800 µs. One group already contains events with latencies around 1,100 µs. Two events from this group were dropped during transport. This already indicates a threshold value for $\Delta t^{Bio}_{Exec/Exec}$, from which event losses occur.

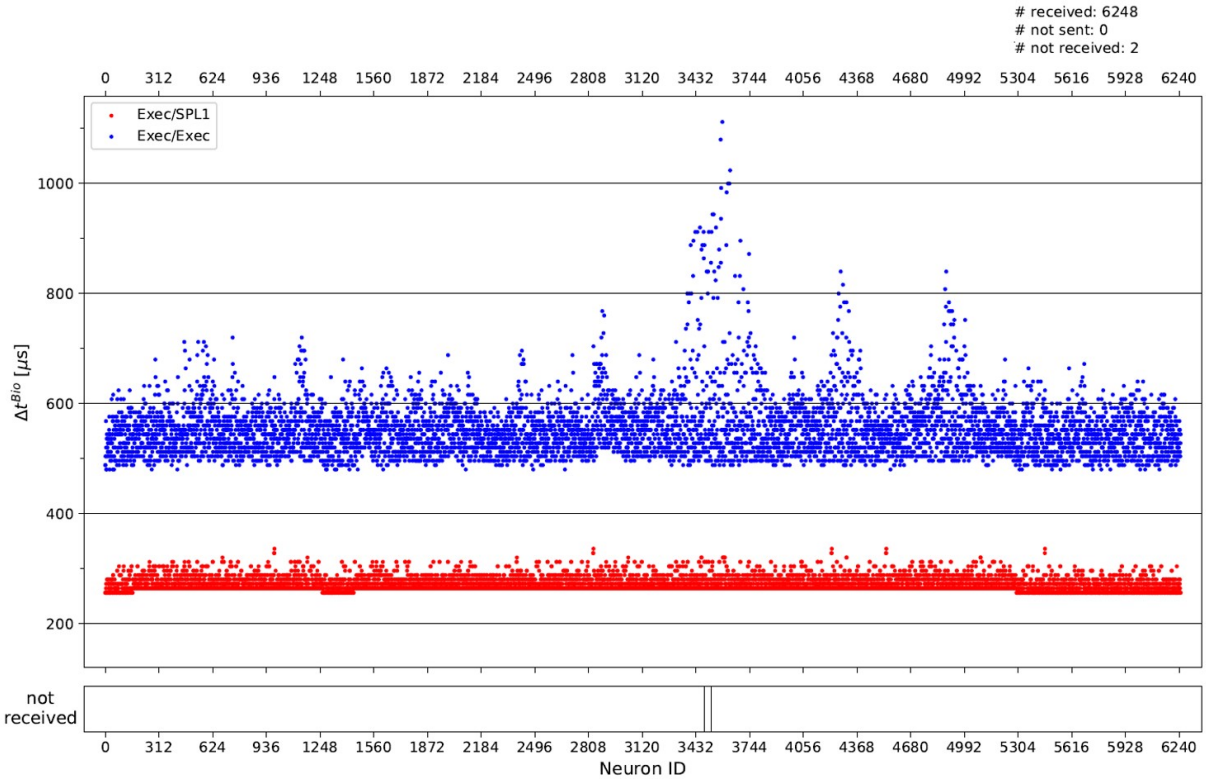

Figure 17: Latencies per event for a 125 kHz Poisson spike train (Δt_{spec} = 55).

At 130 kHz (Figure 18) the spread of the $\Delta t^{Bio}_{Exec/SPL1}$ values remains unchanged at a low level. The number of events with $\Delta t^{Bio}_{Exec/Exec}$ latencies over 800 µs has increased significantly, along with more events exceeding the 1,100 µs mark. The event losses during transport have clearly increased, with drops occurring only in clusters of events with $\Delta t^{Bio}_{Exec/Exec}$ values over 1,100 µs. This value thus emerges as the threshold value for event losses. Furthermore, a first event was removed during modification of the spike train.

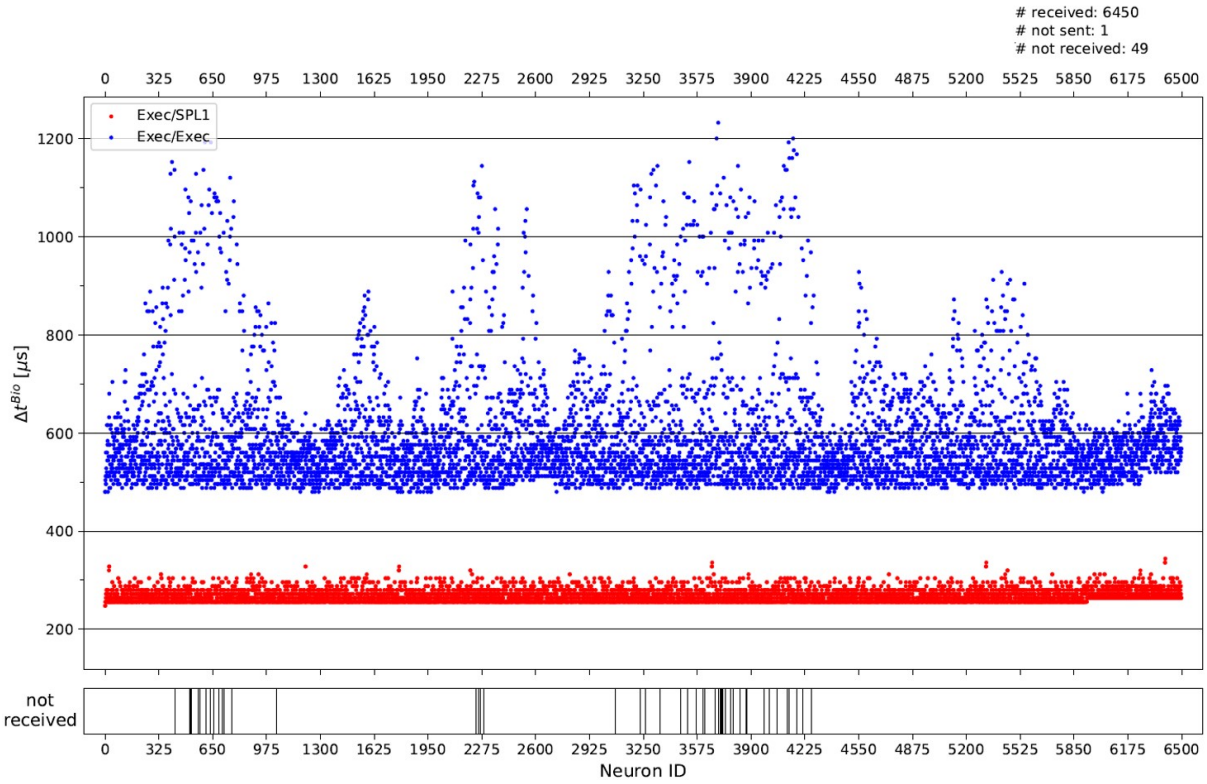

Figure 18: Latencies per event for a 130 kHz Poisson spike train (Δt_{spec} = 55).

At 135 kHz (Figure 19) the number of drops has again increased significantly. Individual events already exhibit a $\Delta t^{Bio}_{Exec/Exec}$ latency of about 1,300 µs. At these high latencies, four $\Delta t^{Bio}_{Exec/SPL1}$ values (marked by a circle) appear, which cause the massive increase in the spread of the upper 15 % of the $\Delta t^{Bio}_{Exec/SPL1}$ values shown in Figure 14.

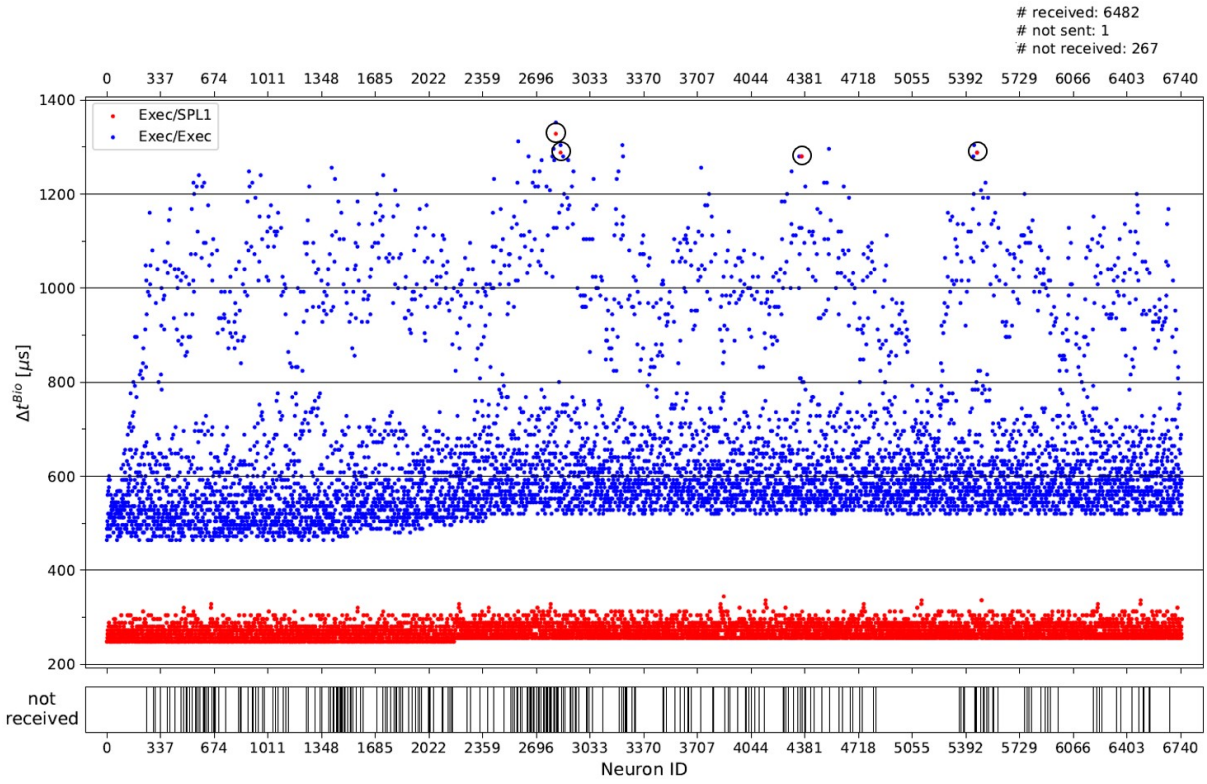

Figure 19: Latencies per event for a 135 kHz Poisson spike train (Δt_{spec} = 55).

As the minimum time span $\Delta t^{Bio}_{SPL1/Exec,\,min}$ between the registration of an event at one of the SPL1 links in the HICANN and its forwarding from the Executor to the Decoder/Encoder is 224 µs, there should be $\Delta t^{Bio}_{Exec/Exec}$ values above 1,500 µs belonging to the four high $\Delta t^{Bio}_{Exec/SPL1}$ values. But since the highest $\Delta t^{Bio}_{Exec/Exec}$ latencies lie below 1,400 µs, either the high $\Delta t^{Bio}_{Exec/SPL1}$ values or their associated $\Delta t^{Bio}_{Exec/Exec}$ values must have been determined incorrectly. While the times used to calculate the $\Delta t^{Bio}_{Exec/Exec}$ values are completely known, the event times t_{event} used to calculate the $\Delta t^{Bio}_{Exec/SPL1}$ values must be reconstructed because only their lower seven bits are known. As already explained in Chapter 2.4.2, the reconstruction may result in incorrectly reconstructed event times if the time span between the registration of an event at one of the SPL1 links in the HICANN and its forwarding from the Executor to the Decoder/Encoder reaches the critical value $\Delta t^{Bio}_{SPL1/Exec,\,crit}$ = 128 clock cycles. The minimum latency $\Delta t^{Bio}_{Exec/SPL1,\,min}$ and $\Delta t^{Bio}_{SPL1/Exec,\,crit}$ yield a critical time span $\Delta t^{Bio}_{Exec/Exec,\,crit}$, from which erroneously reconstructed

event times can occur:

$$\Delta t^{Bio}_{Exec/Exec,\,crit} = \Delta t^{Bio}_{Exec/SPL1,\,min} + \Delta t^{Bio}_{SPL1/Exec,\,crit} \tag{22}$$

$$= 256\ \mu s + 128 * 8\ \mu s \tag{23}$$

$$= 1,280\ \mu s \tag{24}$$

This value is only valid for Δt_{spec} = 55. For other values of Δt_{spec}, $\Delta t^{Bio}_{Exec/SPL1,\,min}$ and thus also $\Delta t^{Bio}_{Exec/Exec,\,crit}$ will change. The calculated value of 1,280 µs is reached for the first time at 135 kHz. The explanations in Chapter 2.4.2 also imply that an incorrectly reconstructed event time is at least 128 clock cycles, i.e. 1,024 µs (biological time) greater than the true event time. Consequently, the $\Delta t^{Bio}_{Exec/SPL1}$ latency calculated from an incorrectly reconstructed event time will also be at least 1,024 µs greater than the true latency and thus lie in ranges above 1,280 µs. The massive spread of the upper 15 % of the $\Delta t^{Bio}_{Exec/SPL1}$ values from biological input event rates of 135 kHz onwards is thus caused by erroneous values which result from a false reconstruction of the event times due to $\Delta t^{Bio}_{SPL1/Exec}$ latencies over 127 clock cycles.

At 250 kHz (Figure 20) numerous $\Delta t^{Bio}_{Exec/Exec}$ latencies are above 1,280 µs. Accordingly, significantly more incorrect $\Delta t^{Bio}_{Exec/SPL1}$ latencies occur. Meanwhile, the rate of events dropped during transport has increased to about 42 %. In addition, a region with very few values has formed between the $\Delta t^{Bio}_{Exec/Exec}$ latencies up to about 1,000 µs and those from about 1,200 µs upwards. The spread of the correct $\Delta t^{Bio}_{Exec/SPL1}$ values has increased as well: They vary predominantly between 256 and 344 µs, with outliers of up to 448 µs. 646 events were removed during modification of the spike train. Due to a rearrangement of the event indices after the modification, the impression arises as if the last 646 events of the spike train had been removed. However, they were actually removed at various positions of the original spike train, as shown in Figure 10.

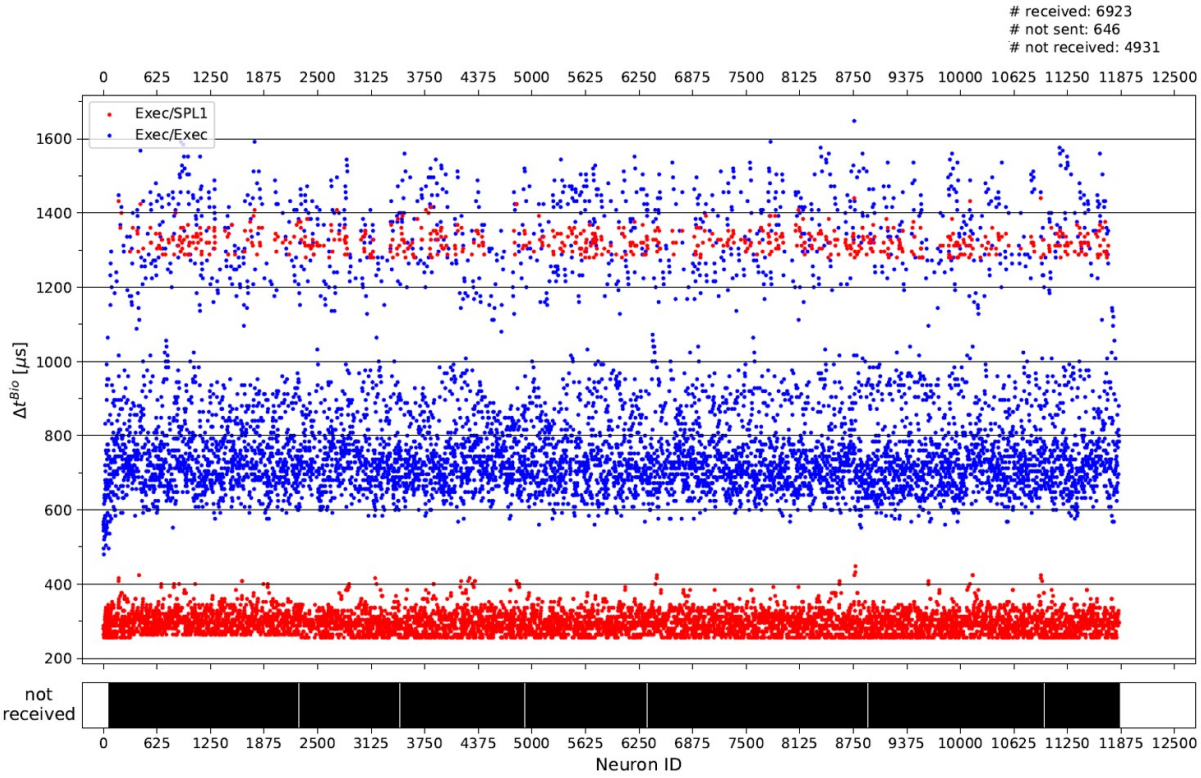

Figure 20: Latencies per event for a 250 kHz Poisson spike train (Δt_{spec} = 55).

Figure 21 illustrates the latency and jitter development at different values of Δt_{spec}. As expected, the medians of the $\Delta t^{Bio}_{Exec/SPL1}$ latencies and thus also those of the $\Delta t^{Bio}_{Exec/Exec}$ latencies shift to larger values with increasing Δt_{spec}. The increase of the medians at low event rates is thereby greater than at high event rates, so that the median curve as a whole becomes flatter. Consequently, the dependence of the latencies on the input event rate decreases with larger Δt_{spec}, but with the disadvantage of higher minimum latencies. The spread of the correctly determined $\Delta t^{Bio}_{Exec/SPL1}$ values (i.e. all values $\Delta t^{Bio}_{Exec/SPL1} < Q_{0.85}$) decreases with larger Δt_{spec}. Since $\Delta t^{Bio}_{Exec/SPL1,\,min}$ increases with larger Δt_{spec}, the critical value $\Delta t^{Bio}_{Exec/Exec,\,crit}$ also increases according to equation (22). Consequently, this critical value is reached at slightly higher event rates, so that incorrectly calculated $\Delta t^{Bio}_{Exec/SPL1}$ values only occur at slightly higher event rates as well.

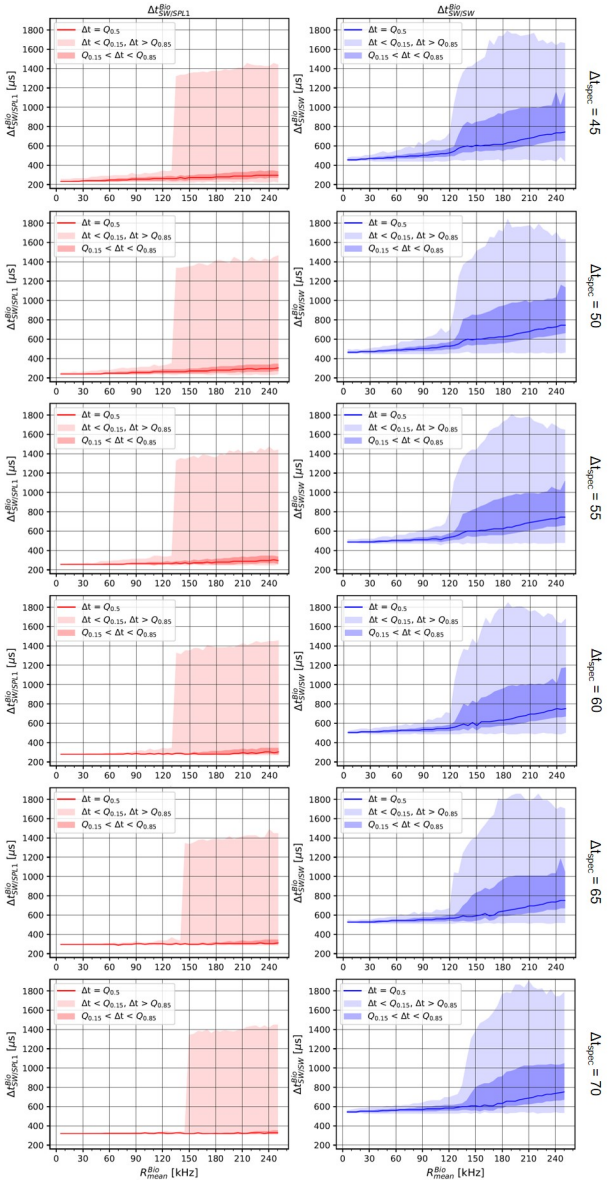

Figure 21: Latency and jitter development for different values of Δt_{spec} (Poisson spike trains).

4 Discussion & Outlook

4.1 Summary

The major goal of this thesis was to characterize the transport of event data through the FPGA communication infrastructure between the Host ARQ and the HICANN-X chip. For this purpose, the dependence of event data losses and the variation of the transmission time on the input event rate were to be investigated.

This investigation required the prior implementation of a Playback Executor HDL module, which controls the data traffic to and from the HICANN and generates a global system time enabling a time-coordinated sending of input event data to the HICANN as well as the forwarding of event data returned by the HICANN to the software level and its classification into a chronological sequence.

For the investigation, the modules of the FPGA communication infrastructure and the relevant digital part of the HICANN were tested using HDL simulations, as the real HICANN-X chip was still in the manufacturing process. In the course of the simulations, events were handed over to the Executor module in the form of *fire* instructions at specified times t_{send}. From there, they propagated through the FPGA communication infrastructure and via the highspeed connection to the HICANN. On the HICANN, they were looped back after traversing the routing interface between the L2 module and the SPL1 level and provided with a timestamp when passing the SPL1 links again at the time t_{event}. Then they propagated back to the Executor module and were forwarded to the software level at the time $t_{receive}$. Based on the number of sent and received events, the event drop rate was determined. Using the timestamps contained in the received event data, the times t_{event} could be reconstructed, which enabled the calculation of different latencies. The determined drop rates and latencies for different input event rates were then compared. The influence of the parameter Δt_{spec}, which is intended to reduce the jitter during transport of event data to the HICANN, was investigated as well.

4.2 Discussion of the Results

For input spike trains with uniformly distributed ISIs as well as for spike trains with Poisson-distributed ISIs, events were dropped during transport through the system from a (mean) biological input event rate of 130 kHz upwards for Δt_{spec} values up to and including 60 HICANN clock cycles. At larger Δt_{spec} values, the event drop rate for Poisson input spike trains already increased from 65 kHz upwards. In all cases, a relative drop rate of about 46 % was measured at a biological input event rate of 250 kHz. This means that the highspeed connection between the FPGA and the HICANN is not the bottleneck of the communication infrastructure, as expected, but another HDL module.

In the course of the jitter analysis a very low dependency of the $\Delta t^{Bio}_{Exec/SPL1}$ latency (i.e. the time span between the transmission of an event at the Executor and its registration at one of the SPL1 links in the HICANN) on the input event rate was determined. In contrast, both the mean $\Delta t^{Bio}_{Exec/Exec}$ latency and the spread of individual $\Delta t^{Bio}_{Exec/Exec}$ values increase significantly from biological input event rates around 120 kHz upwards. Thereby a threshold value of $\Delta t^{Bio}_{Exec/Exec}$ $\approx$ 1,100 µs was identified, from which events are lost. At input event rates where this threshold value is reached, the $\Delta t^{Bio}_{Exec/SPL1}$ latencies are only slightly larger than at low event rates. Consequently, event losses must inevitably be caused by high $\Delta t^{Bio}_{SPL1/Exec}$ latencies, i.e. long transport times from the SPL1 links in the HICANN to the Executor. If the delays on this way are too large so that the critical time span $\Delta t^{Bio}_{SPL1/Exec,\,crit}$ = 1,024 µs is reached, faulty $\Delta t^{Bio}_{Exec/SPL1}$ values will occur due to incorrectly reconstructed event times t_{event}. As the event times are of central importance for the execution and analysis of spike experiments, false reconstructions must necessarily be avoided. An event transport without event losses and incorrectly reconstructed event times is guaranteed for the tested design of the communication infrastructure up to a biological input event rate of 125 kHz, provided that the selected Δt_{spec} does not exceed a value of 60.

The investigation of the latency and jitter development at different values of Δt_{spec} showed that the greater Δt_{spec} is chosen, the lower will be the dependence of the average latencies on the input event rate as well as the spread of individual $\Delta t^{Bio}_{Exec/SPL1}$ values. This enables a more precise stimulation of the neurons on the HICANN. Although a larger Δt_{spec} value causes an increase of the minimum latencies, the resulting longer duration of a spike experiment is negligibly small. For this reason, Δt_{spec} should be chosen as large as possible, whereby a value of 60 should not be exceeded, since otherwise the drop rate can already increase at biological input event rates of less than 125 kHz.

In order to estimate the influence of the event transport on the achievable accuracy of spike experiments, it is sufficient to consider the $\Delta t^{Bio}_{Exec/SPL1}$ latency, as only this latency directly affects the temporal precision of the neuron stimulation. The largest spread of correct $\Delta t^{Bio}_{Exec/SPL1}$ values was measured at a mean biological input event rate of 250 kHz and is 192 µs. The spread of the mean 70 % of the measured values at 250 kHz is 80 µs. In contrast, the achievable accuracy of the membrane time constant τ_m, which significantly influences the spike behaviour of the neurons, is in the order of 1 ms [3]. The error caused by variations in the $\Delta t^{Bio}_{Exec/SPL1}$ latency is therefore negligibly small.

4.3 Outlook

A further analysis identified the switch of the Transport Layer on the FPGA as potential bottleneck of the communication infrastructure. In upstream direction, the switch receives event

data from the eight UTs of the Link Layer and forwards it through a single channel to the event interface (Figure 22). Via this channel, double and triple packed events (i.e. event messages containing data from two or three different events) can also be forwarded. The event packing (i.e. the merging of several event data into a single event message) currently happens exclusively in the HICANN between the SPL1 level and the L2 switch, which distributes the event messages to the UTs on the HICANN. During the analysis, however, an event packing could only be observed in very few cases. At mean biological input event rates of more than 125 kHz, where more than one event per clock cycle is transmitted to and from the HICANN via the highspeed connection, the FPGA switch also receives more than one event message per clock cycle due to the inefficient event packing. However, since the switch can only forward one event message per clock cycle to the event interface, congestions in the individual switch nodes will inevitably occur, causing the nodes to block subsequent event messages from the UTs. The UTs are connected to the PHYs of the Physical Layer via non-blocking ut-interfaces and therefore cannot pause accepting further event messages. This causes the event messages contained in the UTs to be overwritten by subsequent messages and thus leads to event losses.

The event losses inside the Link Layer of the FPGA due to the inefficient event packing in the HICANN can probably be significantly reduced by implementing so-called event compressors between the switch nodes and the UTs (Figure 22). The event compressors improve the event packing by merging the single and double packed events received from the UTs into double and triple packed events if they cannot directly be forwarded to the corresponding switch node. An additional event compressor between the switch and the event interface prevents event losses due to delays between the Executor and the Decoder/Encoder. The event compressors should be designed in such a way that they only modify event data while forwarding all other data unchanged to the switch nodes. In the ideal case, event losses up to biological input event rates of 250 kHz can be avoided. This upper limit results from the maximum data transfer rate of the highspeed connection.

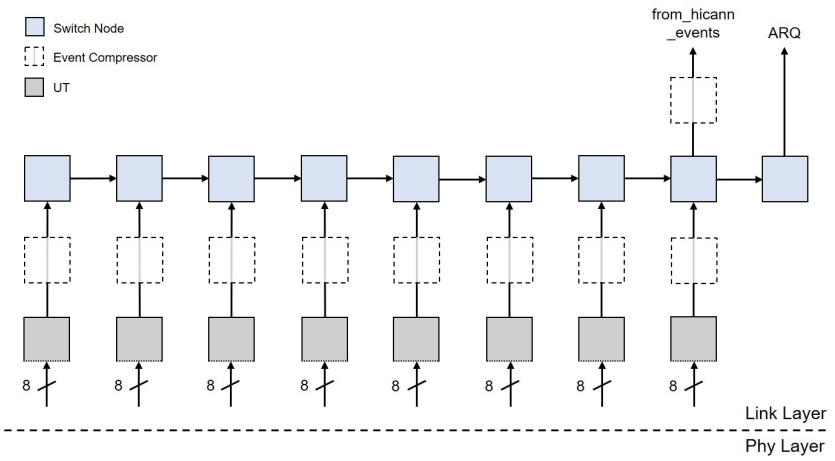

Figure 22: Event compressor integration.

References

[1] https://www.humanbrainproject.eu/en/silicon-brains/.

[2] https://www.kip.uni-heidelberg.de/vision/previous-projects/facets/neuromorphic-hardware/.

[3] S. A. Aamir et al. An accelerated lif neuronal network array for a large scale mixed-signal neuromorphic architecture, 2018.

[4] A. Gruebl and A. Baumbach. *F09/F10 Neuromorphic Computing*. University of Heidelberg.

[5] L. S. Kanzleiter. A parametrizable switch for neuromorphic hardware. Bachelor thesis, University of Heidelberg, 2018.

[6] M. Rettig. Verification of a parameterizable jtag driver module. Project internship report, University of Heidelberg, 2019.

[7] J. Schemmel, J. Fieres, and K. H. Meier. Wafer-scale integration of analog neural networks, 2008.

[8] A. Schmidt. Design und charakterisierung einer routing-schnittstelle fuer neuromorphe hardware. Bachelor thesis, University of Heidelberg, 2017.

Acknowledgements

I would like to express my gratitude to

- Prof. Dr. Karlheinz Meier and Dr. Johannes Schemmel for giving me the opportunity to be part of the Electronic Vision(s) group and to carry out my bachelor's thesis on this topic.
- Vitali Karasenko for the comprehensive and patient supervision throughout my internship and bachelor's thesis.
- Mitja Kleider for his support while struggling with the simulation tools.
- Oliver Breitwieser for the inspiring suggestions regarding the visualization of my results.
- Timo Wunderlich for proofreading my thesis.
- The whole group for the welcoming atmosphere.

Statement of Originality

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

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

Heidelberg, May 16, 2019

.....

(signature)