**Department of Physics and Astronomy**

**University of Heidelberg**

**Master Thesis**

in Physics

submitted by

**Simon Rosenkranz**

born in Jena, Germany

**2024**

# High-Speed Link Stability Improvements for Neuromorphic Multi-Chip Systems

This Master Thesis has been carried out by

**Simon Rosenkranz**

at the

Kirchhoff-Institute for Physics

Ruprecht-Karls-Universität Heidelberg

under the supervision of

**Dr. rer. nat., Priv.-Doz. Johannes Schemmel**

# High-Speed Link Stability Improvements for Neuromorphic Multi-Chip Systems

The *BrainScaleS-2 neuromorphic hardware* (BSS-2) enables the accelerated emulation of spiking neural networks on a mixed-signal substrate. Being the principal building block of this system, the *High Input Count Analog Neural Network with HAGEN Extensions* (HICANN-X) *application-specific integrated circuit* (ASIC) employs a total of 512 neuron circuits. In the framework of this thesis, a multi-chip system for the BSS-2 neuromorphic system has been designed to allow the realization of more complex networks. The presented platform offers the possibility to interconnect up to four HICANN-X ASICs in a star-topology by employing a *field programmable gate array* (FPGA) as the central connecting node. Successful communication tests have been performed with the commissioned setup in single-chip operation. Networks with sizes of up to 2048 neurons at an expected event transmission latency of 1 ms in biological time scale can be realized with further development efforts. This work further focuses on the stabilization of the HICANN-X high-speed communication interface. When scaling the network size in a multi-chip approach, an increasing amount of event data must be exchanged between the network constituents and the experiment control nodes. A testing routine has been developed to quantify the link stability in terms of a probability for link lost events. In-hardware probing enabled the optimization of the clock-to-data alignment algorithm, reducing the probability for follow-up link failures. Investigations of relevant ASIC system parameters show a dependency of the observed link losses on the core supply voltage. A simple test for calibration and commissioning was developed that allows hardware users to find a good operating point for their system.

# Verbesserung der Stabilität von Hochgeschwindigkeitslinks in Neuromorphen Multi-Chip Systemen

Das neuromorphe Hardware-System *BrainScaleS-2* (BSS-2) ermöglicht das Emulieren von spikenden neuronalen Netzen auf beschleunigten analogen Schaltungen. Der mit 512 analogen Neuronschaltkreisen ausgestattete *High Input Count Analog Neural Network with HAGEN Extensions* (HICANN-X) *application-specific integrated circuit* (ASIC) ist der Grundbaustein des Systems. Für komplexere Netze wurde im Rahmen der vorliegenden Arbeit ein Multi-Chip System für BSS-2 entwickelt. Das System ermöglicht die Verbindung von bis zu vier HICANN-X ASICs in einer Sterntopologie durch ein zentrales *field programmable gate array* (FPGA). Kommunikationstests mit einem einzelnen Chip konnten erfolgreich durchgeführt werden. Erweiterungen des Systems für Netzwerke von bis zu 2048 Neuronen bei einer erwarteten Event-Latenz von 1 ms sind möglich. Weiterhin wurde die Stabilisierung der Hochgeschwindigkeitskommunikationsschnittstelle des HICANN-X behandelt. Wenn die Größe der neuronalen Netze durch die Einbindung mehrerer Chips skaliert wird, muss eine größer werdende Menge an Daten zwischen den Netzwerkpartnern und den Kontrollinstanzen des Experiments ausgetauscht werden. Für die Quantifizierung der Linkstabilität wurde eine Testroutine entwickelt. Die Optimierung der Angleichung von Takt und Daten mittels In-Hardware Probing führt zu einer Verringerung von Folgefehlern der Links. Untersuchungen der Systemparameter der ASICs zeigen eine Abhängigkeit zwischen der Core-Versorgungsspannung und den beobachteten Linkausfällen. Ein einfacher Test wurde entwickelt, um Benutzer*innen der Hardware die Stabilisierung der Eventübertragung durch Kalibration zu ermöglichen.

# Contents

# 1 Introduction

In the field of computational neuroscience, emulation of biologically inspired neural network models can be carried out with *neuromorphic* hardware. In contrast to simulation-based approaches, the modeling of the network dynamics is not done through numerically solving the underlying differential equations. Neuromorphic systems rather aim to realize a physical representation of the system to be studied. In doing so, these highly specialized machines offer an improvement in efficiency, speed and scalability for neuroscientific modeling over traditional Von-Neumann architectures (Petrovici 2016).

The *BrainScaleS-2 neuromorphic hardware* (BSS-2) architecture implements a mixed-signal, accelerated, physical emulation of bio-inspired *spiking neural network*s (SNNs) (Pehle et al. 2022). The principal building block of the BSS-2 system is the *High Input Count Analog Neural Network with HAGEN Extensions* (HICANN-X) *application-specific integrated circuit* (ASIC) (cf. Figure 1.1), a custom analog accelerator core with a tightly coupled digital plasticity processing unit and a digital event-routing network (Schemmel et al. 2022). The analog core of a single HICANN-X ASIC consists of 512 neuron circuits that mimic the action potential dynamics of real biological neurons. As the intrinsic time constants of the analog circuitry are several orders of magnitude smaller than the processes that happen on a biological time scale, the emulation of the network dynamics is accelerated by a factor of approximately 1000 compared to biology.

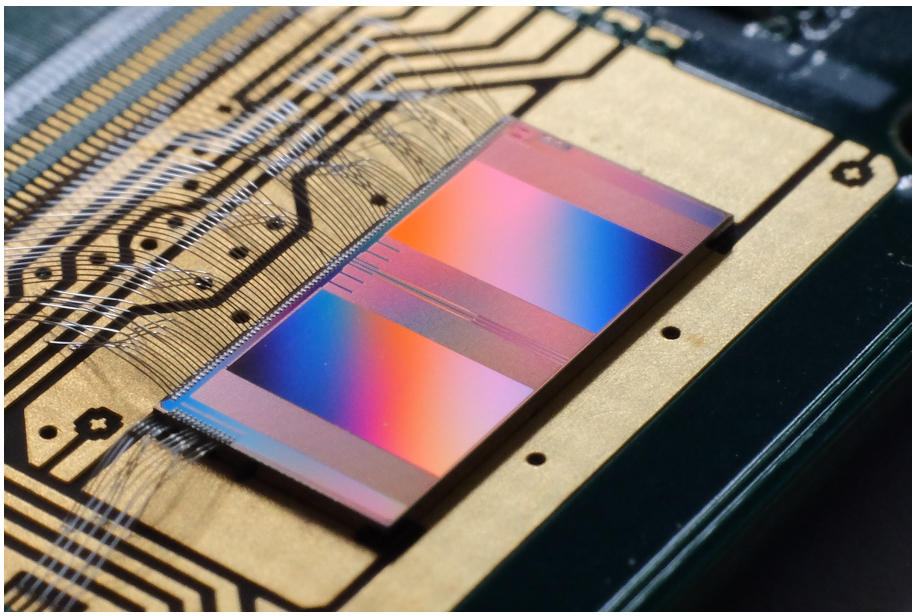


**Figure 1.1:** Photograph of a HICANN-X ASIC, the principal building block within the BSS-2 architecture, bonded to a chip carrier. Photo by Eric Müller, taken from `https://wiki.ebrains.eu/bin/view/Collabs/neuromorphic/BrainScaleS/` (accessed 29 Nov, 2023).

At the present time, experiments with the BSS-2 architecture are most commonly executed on so-

called *Cube*[1] systems (cf. Figure 1.2), a hardware setup consisting of several *printed circuit board*s (PCBs). The Cube implements full host-to-chip communication for a single HICANN-X ASIC by employing a *field programmable gate array* (FPGA) for real-time control during experiment execution. The FPGA buffers stimulus and output data, manages memory access for the plasticity processing unit and provides connection to and from the host system (Pehle et al. 2022). Injection of timed spike stimuli is done through a high-speed communication interface between the ASIC and the FPGA.

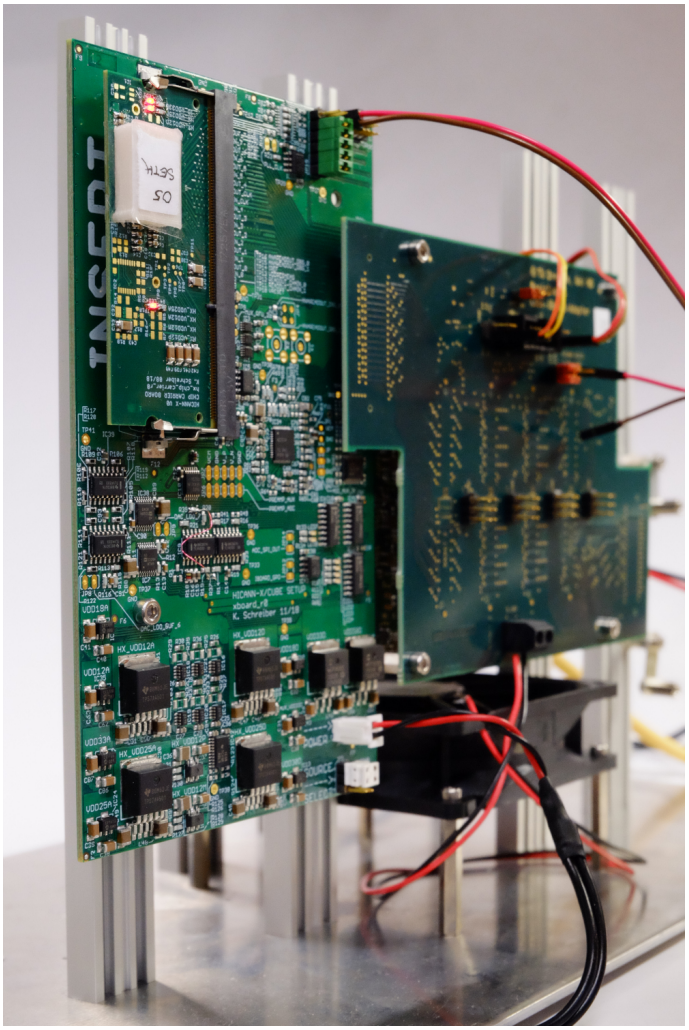


**Figure 1.2:** Photograph of a Cube system, taken from Pehle et al. 2022.

Although heavily utilized during prototyping and commissioning, the limitation of the Cube system to a single neuromorphic chip with 512 neurons limits the execution of complex experiments. Current design efforts for BSS-2 target the development of multi-chip systems which integrate multiple standalone HICANN-X ASICs into a larger system. The realizability of such systems depends on a robust inter-chip connection interface that manages to fulfill the high bandwidth demands implied by the event transmission rates of the accelerated network dynamics. While this communication framework exists for the BSS-2 architecture, it has in the past been observed to be afflicted with faulty link behavior. This limits the event throughput and can seriously impact the feasibility of the targeted multi-chip systems.

[1] The denomination stems from the form factor of the enclosing rack, not fully depicted in Figure 1.2.

In the scope of this thesis, two aspects of developing a multi-chip platform have been covered: Firstly, a multi-chip setup has been designed that targets the interconnection of four HICANN-X ASICs through a centralized node FPGA. The setup has been commissioned up to single-chip operation with successful first communication tests having been carried out. However, the further development of the platform was hindered due to a hardware defect. Secondly, the high-speed communication interface between ASIC[2] and FPGA has been investigated with respect to the observed link lost failures. The stability of the high-speed links has been characterized by means of a time to failure and a link lost probability. By probing the FPGA implementation with in-hardware logic analyzing tools, faulty behavior of the delay training algorithm has been detected and corrected. An examination of ASIC system parameters yielded an additional improvement of the link stability.

The thesis at hand is structured as follows: Chapter 2 describes the planning and design of the multi-chip platform and outlines the commissioning process from a hardware design point of view. Chapter 3 describes the observed link stability issues and defines a stability criterion. A testing routine is outlined to gather statistics that are comparable across a variety of systems. Subsequently, the clock-to-data alignment procedure is described along an example recorded from in-hardware probing. Potential errors in the algorithm are pointed out along with giving corrections that address the faulty link training examples. The chapter concludes with an examination of additional sources of error that may be linked to ASIC system parameters. Lastly, Chapter 4 provides a summary of the work and an outlook on future developments.

[2]Note that throughout this document, the HICANN-X will frequently just be referred to as *the ASIC*.

# 2 Multi-Chip Platform

The computational power of neural networks stems to a large extent from their complexity in terms of constituent numbers. Naturally, one of the main challenges in brain-inspired computing is the realization of networks that are comprised of an increasing number of neurons to enable the execution of ever more complex experiments. In a physical modeling approach to building such systems – as is targeted with the BSS-2 architecture – the network size is especially tightly coupled to the size of the physical substrate. That is, in order to realize more complex network structures it is required that the neuromorphic architecture is *scaled*.

Within the BSS-2 architecture, the HICANN-X is designed as a structural building block with a clear purpose: It serves as a testing unit that allows to first be well understood and controllable by thorough verification and execution of standalone experiments before being integrated into larger systems. While the total number of 512 neurons in a single HICANN-X is sufficient for testing and smaller experiments, the network size can quickly become an issue. For instance, when the total neuron count is traded against equipping the network constituents with a sub-structure – as is for instance done in compartment neurons (Kaiser et al. 2022) – the neuron count of a single HICANN-X can be considered as a limiting factor.

Although the ultimate goal behind the BSS-2 architecture is to accomplish wafer-scale integration as achieved with the *BrainScaleS neuromorphic hardware* (BSS) architecture, the realization of such large-scale systems is a time and resource consuming process. The upgrade path of the system hence foresees the realization of medium size systems on the order of ten HICANN-X ASICs. This would allow researchers to run experiments that are more involved than those running on a single chip, but do not require the availability of full wafer-scale resources.

The HICANN-X is constructed in a way that in principle allows for the exchange of event data at high throughput with other individual chips. Therefore, the enlargement of the available analog substrate is also possible through the interconnection of single chips. The buildup of such *multi-chip* networks is one of the most important topics within the current development efforts of the BSS-2 platform.

This chapter introduces a design for a multi-chip setup that has been developed in the framework of this thesis. The presented platform offers the possibility to interconnect up to four HICANN-X ASICs. Section 2.1 describes the considerations made during the planning phase of the system – highlighting the design choices made and their potential advantages and disadvantages – and the constraints that are imposed on the system by interfacing hardware elements. Section 2.2 outlines the concrete hardware development steps that have been taken in commissioning the system up to first communication tests in single-chip operation. The chapter concludes with a summary and discussion of the current development state of the platform in Section 2.3.

## 2.1 Design Planning

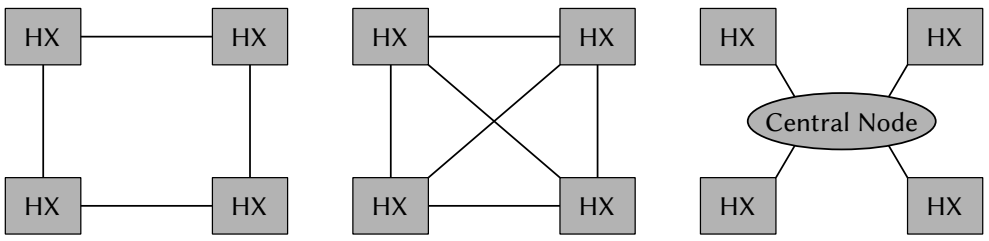
The upscaling procedure presented in this thesis is performed in a modular fashion. The idea is to use as many of the already existing hardware components as possible that the BSS and BSS-2 architectures offer to minimize the design overhead. This section provides a brief overview over the planning efforts to develop such a multi-chip platform for the BSS-2 architecture. First, some theoretical considerations will be presented that outline the further design goals for the platform. Subsequently, the design constraints posed by the existing hardware stack that is to be incorporated in the system will be presented. Lastly, an estimation of the expected network size and latency for the targeted system will be given.

### 2.1.1 Theoretical Considerations

In order to build up a network of individual components, one needs to first address the underlying network topology that is targeted for the system. This choice greatly impacts the constraints that the design needs to adhere to. Hence, an overview over the considered upscaling approaches will be presented in the following.

#### Topology

There are plenty of topologies available to assemble a set of single components in a fully connected network, and all of them come with specific advantages and disadvantages. Figure 2.1 shows a couple of examples for possible interconnection variants imaginable to set up a large-scale network of HICANN-X ASICs.




**(a)** Mesh connection. **(b)** All-to-all connection. **(c)** Star connection.

**Figure 2.1:** Set of exemplary interconnection topologies for multi-chip systems.

In the following, a network of four ASICs will be considered in which each chip may send and receive events to and from all three other partners.

A straightforward way to set up a connection scheme for the network constituents could be to arrange them on a two-dimensional grid with horizontal and vertical nearest-neighbor connections like in Figure 2.1a. This mesh-type connection scheme would require only four edges in total to allow for each node to talk to every other network constituent. However, this approach inherits a certain complexity in how nodes that are not directly connected via an edge are able to send data to each other. Some kind of event routing logic is required at every node. This ensures that data, which is received from a neighboring node, but intended for the second directly connected

partner, is properly transmitted through the mesh. The issue becomes more complex with an increasing number of network constituents. Moreover, the routing logic must be attachable to every single node, which requires a significant amount of hardware resources to be added to the existing architecture. An advantage of this approach is that the network can in principle scale well in two dimensions, as the transmission lines between network partners can be kept short[1].

An alternative option would be to equip every node with direct connections to every other network partner, such as shown in the all-to-all arrangement in Figure 2.1b. This topology bypasses the need for a more involved routing scheme as every node has direct access to all partners, but it quickly becomes infeasible to add more components to the network. The number of required edges for a total of $N$ nodes grows as $\frac{N}{2}(N-1)$ and hence quadratic in network size. So for smaller networks, this could be an option, but as the goal is to interconnect several hundreds of chips eventually, this connection topology does not serve as a suitable option.

Lastly, the star configuration displayed in Figure 2.1c has been considered. In this topology, all nodes share access to a central connecting node. The central component handles the routing of the events between all network constituents. Thereby, the topology is somewhat similar to the mesh-type configuration in that it also requires the existence of a dedicated routing scheme. On the other hand, the overhead for additional hardware resources is greatly reduced, given that the central node is powerful enough to handle the large amount of data that is piped through it. There very likely are limitations to the number of possible connections to the central node as it cannot provide an infinite amount of *input and output* (IO) resources. Yet, medium-size networks of tenths of chips could be imaginable. The main disadvantage of this approach is that the network is greatly dependent on the performance of the central node, but the simplicity of the hardware requirements somewhat alleviates this issue.

Overall, the star configuration promises to strike a good balance between the realizable network sizes and the required engineering efforts. It is therefore the topology of choice for a medium-size multi-chip platform on the order of ten interconnected ASICs.

### Central Node

With the star topology targeted as the desired configuration for the multi-chip system, the question arises what could serve as the central node in the setup. In the approach presented in this thesis, the interconnection is realized through an FPGA.

#### Field-Programmable Gate Array

A *field programmable gate array* (FPGA) is an *integrated circuit* (IC) that provides a set of configurable logic elements and routing resources which together can be programmed to realize a user-defined circuit. That is, in contrast to an ASIC, the circuit that is realized with an FPGA does not need to be specified upfront, before the IC is manufactured. It can be defined at a later point in time while, in many cases, using the same fabric or device is possible. Typically, the logic elements within the FPGA fabric are arranged in arrays and can vary in their functional complexity. The simplest elements provide basic combinatorial logic functions while more complex ones realize specialized functions. Among others, these can be dedicated blocks for handling IO to the fabric,

[1] For instance, the wafer-scale integration of ASICs in the BSS architecture used this kind of topology.

memory elements, advanced functions such as multipliers or even complete processors. In many cases, the dedicated IO cells also support various signaling standards. FPGAs therefore provide lots of flexibility during the development cycle of digital designs and can be employed in a vast range of applications.

Readily available for the setup targeted in this thesis has been a Xilinx Virtex UltraScale+ VU9P FPGA. It provides access to roughly $2.6 \times 10^6$ system logic cells[2] (AMD 2024), which is roughly equivalent to 16 times the resources of a Kintex-7 FPGA that is used as an experiment control node in the BSS-2 architecture. The FPGA is embedded in an *AMD Virtex UltraScale+ FPGA VCU118 Evaluation Kit* (VCU118) (AMD 2023), which is displayed in Figure 2.2. The VCU118 is a $24 \times 18\,\mathrm{cm}^2$ PCB that provides many IO options to the user for prototyping a design with the FPGA in early development phases. In the case of the multi-chip platform, most interesting are the 192 user-configurable IO pins and 24 multi-gigabit transceivers that are connected to the FPGA and accessible via two high-pin-count *FPGA Mezzanine Card* (FMC) connectors at the upper edge of the board.

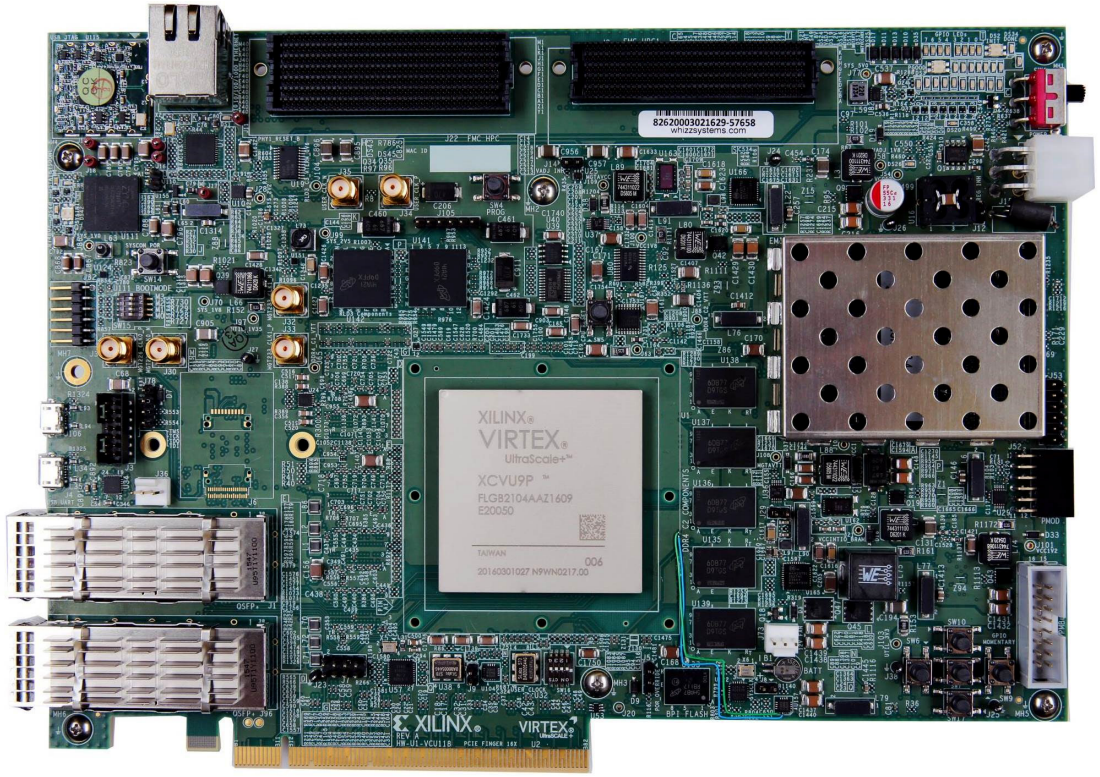



**Figure 2.2:** Photograph of the VCU118, taken from AMD 2023. The FPGA is assembled at the center of the PCB (*grey*). At the upper edge of the board, two high pin count FMC and *FPGA Mezzanine Card Plus* (FMC+) connectors (*black*) are located that provide a large portion of the IO to the FPGA. The FMC+ connector corresponds to the left of the two connector arrays and is slightly larger than the FMC connector to its right due to a higher pin count.

With the vast amount of logic resources available, the FPGA on the VCU118 should in theory be able to handle the experiment control logic needed for multiple HICANN-X ASICs while leaving enough headroom for the extension with a dedicated routing protocol to link between the single control entities.

[2] Marketing term employed by Xilinx to give a metric for the complexity of a design that can be fit onto a device.

### 2.1.2 Design Constraints

As briefly mentioned before, the VCU118 provides access to most of the FPGA fabric through two *high pin count* (HPC) FMC and FMC+ connectors, respectively. These connectors adhere to the *ANSI/VITA 57.1-2019 FPGA Mezzanine Card (FMC) Standard* (VITA 2019) and the *ANSI/VITA 57.4-2018 FPGA Mezzanine Card Plus (FMC+) Standard* (VITA 2018). They are large arrays of 400 and 560 pins, respectively, with standardized locations of, for instance, user configurable IO, power distribution, clocking or high-speed transceiver resources. The idea behind this standardization is to bundle the IO of the FPGA in a single component so that when utilizing a board such as the VCU118 the user only has to adhere to a single standard. The specific IO requirements of the application are outsourced to a *mezzanine* PCB that can be attached to the FMC connectors on the *carrier* board hosting the FPGA. The mezzanine board needs to be specified and designed by the user. When the IO requirements change at a later point in development, only the mezzanine board has to be re-designed and the carrier board can be re-used.

So, in order to interface with the VCU118, a custom mezzanine PCB is needed. A prototype for such a mezzanine card has been developed in the scope of this thesis. An overview of the board is provided in Section 2.2.2.

With the interface to the central node being defined through the mezzanine card, the hardware interface towards the HICANN-X needs to be defined as well. In BSS-2 systems, the HICANN-X ASICs are bonded on dedicated *chip carrier* boards – an example is displayed in Figure 2.3 – that provide a low-cost solution to change and upgrade the neuromorphic substrate without needing to build up entirely new setups.

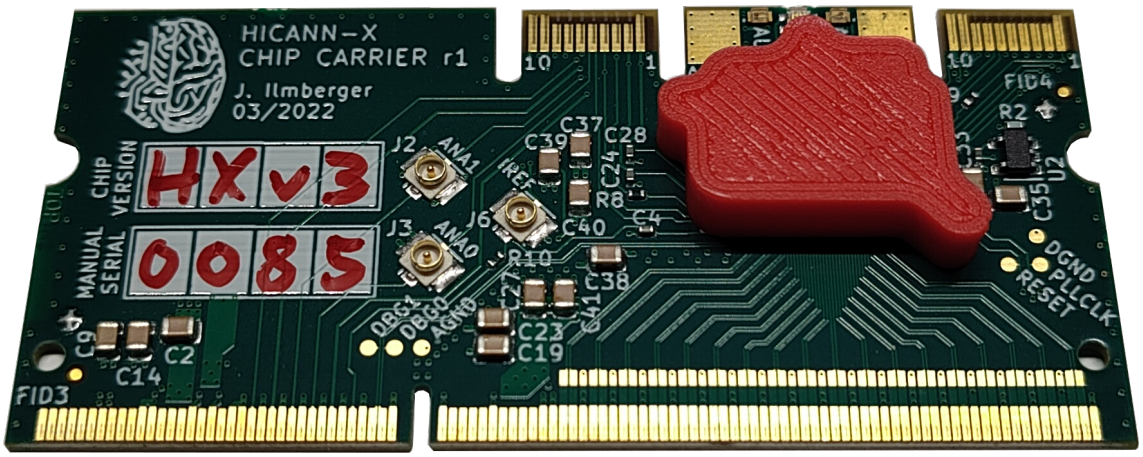



**Figure 2.3:** Photograph of the HICANN-X chip carrier board.

The chip carrier interfaces to other components through a board-edge connector, which provides access to power, high-speed links for communication with the experiment control node and configuration via JTAG. In the currently employed single-chip Cube systems, the chip carriers interface to the *xBoard*, which generates the supply voltages of the HICANN-X and provides the reference voltage and current for the analog circuitry, among other things. However, for the multi-chip setup, the xBoard is planned to be replaced with a new *ASIC Adapter Board* (ASIC-AB) developed for the *just a Bunch Of ASICs* (jBOA) system that aims to replace the Cube systems in the long run[3].

[3] Personal communication with Joscha Ilmberger.

An example of the latest version of the ASIC-AB is displayed in Figure 2.4. The feature set of the ASIC-AB includes

- *low-dropout regulator*s (LDOs) for regulation of the supply voltages of the ASIC,
- generation of the reference current for the ASIC,
- buffering of a differential reference clock that needs to be provided single-ended to the *phase-locked loop* (PLL) of the ASIC,
- level shifters for the reset, JTAG and *general purpose input/output* (GPIO) pins of the ASIC and
- forwarding of the high-speed communication interface between FPGA and ASIC.

The connectivity from the experiment control node side to the ASIC-AB is again realized through a board-edge solution. The full interface towards the ASIC-AB from this side includes

1. eight high-speed *low voltage differential signaling* (LVDS) data signals from FPGA to ASIC with one LVDS clock signal,
2. eight high-speed LVDS data signals from ASIC to FPGA with one LVDS clock signal,
3. single-ended slow control via *Inter-Integrated Circuit bus* (I2C) and JTAG,
4. one asynchronous reset signal,
5. one LVDS reference clock signal for the PLL,
6. 1.2 V ASIC core supply voltage,
7. 2.5 V ASIC IO supply voltage,
8. 3.3 V ASIC auxiliary supply and standby voltage,
9. 12 V ASIC auxiliary supply voltage,
10. 1.8 V FPGA IO voltage used in the level shifters of the slow control signals and
11. one power enable pin for the ASIC supply voltages.

When considering multiple pairs of chip carriers and ASIC-ABs, the signals listed under bullet point 1 to 5 in the previous list need to be supplied to each ASIC-AB individually. In total, this sums up to 45 individual signals to be routed towards the ASIC-AB per HICANN-X to be integrated in the network. The two FMC+ and FMC connectors on the VCU118 offer a total of 192 user configurable pins, so in theory a maximum of four ASICs with 180 signal connections could be connected to the central node with full high-speed interface bandwidth.


**Figure 2.4:** Photograph of the ASIC-AB.

Additional connections could be realized by exploiting the 24 transceiver connections and interposing a Kintex-7 FPGA node board for interfacing to each HICANN-X before establishing the connection towards the central node FPGA on the VCU118. This option has been considered in the early planning phase of the multi-chip platform, but ultimately rendered not possible to be realized within the time limit of this thesis.

### 2.1.3 Network Size and Latency

In Section 2.1.2 it was discussed that a total of four HICANN-X ASICs at full 8 Gbit/s event data throughput towards the experiment control node can in theory be aggregated on the VCU118 FPGA when utilizing the user configurable IO pins. With 512 neurons per chip, a maximum of 2048 neurons could be aggregated in the resulting multi-chip network, offering a rather moderate upgrade in terms of the total neuron count compared to the currently employed single-chip Cube setups. However, since the footprint of the setup is mainly dominated by the dimensions of the VCU118, the multi-chip platform can offer a significant upgrade in neuron density while exhibiting a comparable or even smaller form factor than the Cube system.

The intrinsic acceleration of the neuron activity within the analog SNNs in the BSS-2 architecture additionally poses critical demands on the latency that is introduced during event transmission. In order to route an event from a neuron located on one HICANN-X to a network partner located on a different ASIC, the transmission delays from the ASIC to the FPGA and vice versa need to be taken into account. An estimation of the from-chip and to-chip delays is provided in Karasenko 2020 and sums up to a worst-case latency of approximately 1 µs for a full ASIC-to-FPGA and FPGA-to-ASIC loop. Additionally, the routing of the event within the central node FPGA needs to be considered. The delay introduced by the routing logic is expected to be on the order of a couple of system clock cycles of the FPGA. This would, for instance, correspond to some clock cycles at a frequency of 125 MHz[4]. Therefore, the delay of the routing logic should be negligible compared to the total from- and to-chip delay of 1 µs, which is then expected to dominate the total latency. The latency corresponds to roughly 1 ms in biological time per event transmission, when taking the implicit 1000-fold acceleration factor of the BSS-2 hardware into account. Compared

[4]Personal communication with Joscha Ilmberger.

to typical neuron parameters, this can be considered as significant. The typical neuron response latency is of the order of 0 ms to 8 ms (Müller 2017), so the latency in the multi-chip system is expected to be of the same order of magnitude. However, it should be located on the lower end of the latency range and may potentially be compensated for during training, if the latency is deterministic and not heavily affected by jitter[5].

## 2.2 Hardware Commissioning

With the theoretical design constraints for the multi-chip platform outlined in Section 2.1.1, the following section describes the steps that have been taken to realize a working single-chip system ready to be extended for the incorporation of additional ASICs. However, a defect of the VCU118 prohibited the further development of the platform for full multi-chip support. This section provides a schematic overview over the targeted multi-chip system before detailing the design choices for the mezzanine card that completes the required hardware stack. Further, the necessary adjustments of the existing FPGA design for single-chip operation to comply with the VCU118 will be described. Lastly, the results of first single-chip hardware tests before the defect of the VCU118 will be described and discussed.

### 2.2.1 Setup Overview

A schematic overview of the multi-chip platform developed in this thesis is provided in Figure 2.5. The mezzanine card prototype developed for this configuration – which will be described in more detail in Section 2.2.2 – is referred to as *Mezzanine card for Mixed multi-chip setup evaluation* (MezzaMix), as it provides the IO necessary to evaluate two distinct upscaling paths: On the one hand, it features the connectors to directly attach four ASIC-ABs, which interface with the user configurable IO pins of the FMC and FMC+ connectors, and on the other hand, it features four USB-C receptacles[6], each of which connect to one multi-gigabit transceiver on the VCU118 FPGA. As briefly touched upon in Section 2.1.1, the upscaling path utilizing the transceivers of the VCU118 has not been possible to be evaluated in the limited time of this work, but nevertheless the integration of the required interface on the MezzaMix is mentioned here for completeness.

For the direct attachment of ASIC-AB and chip carrier pairs, the MezzaMix PCB must be able to perform the following tasks:

- Distribute all required supply voltages to the ASIC-AB and the HICANN-X.
- Forward the high-speed communication interface between FPGA and ASIC.
- Connect slow control and auxiliary signals from the FPGA to the HICANN-X.

The schematic shown in Figure 2.5 summarizes how all required signals are distributed between the central node FPGA and the ASICs through the MezzaMix PCB. The major portion is routed through the FMC+ connector, as it provides a larger amount of accessible pins, while the FMC

[5] Personal communication with Joscha Ilmberger.
[6] Not depicted in Figure 2.5

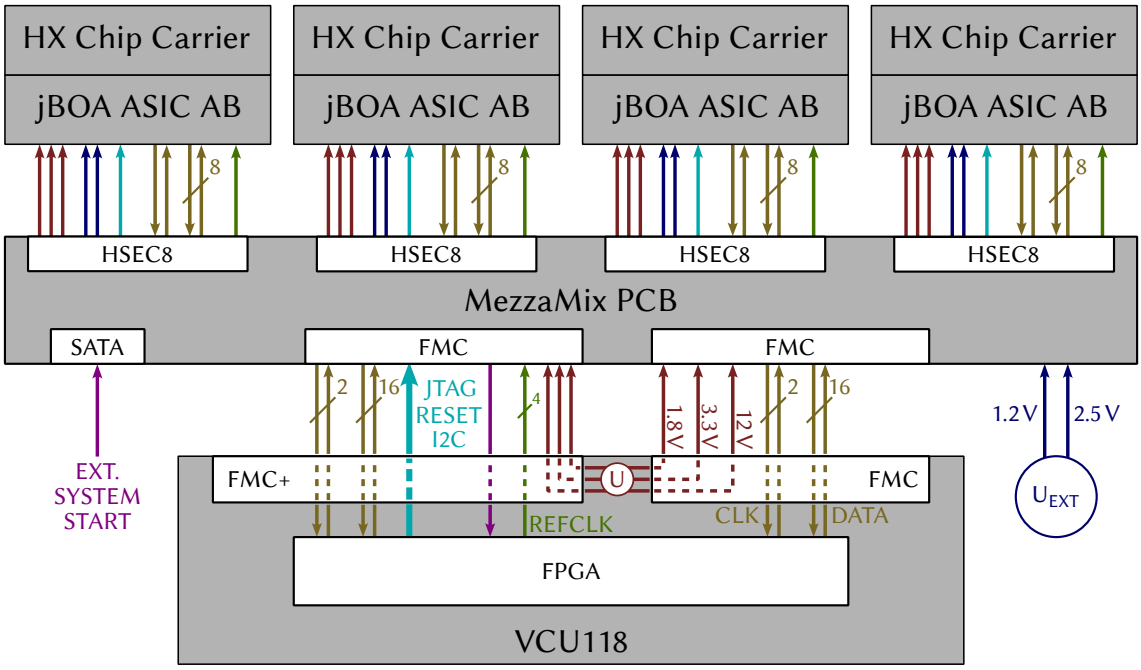



**Figure 2.5:** Schematic overview of the multi-chip platform with the MezzaMix PCB interfacing between the VCU118 serving as central node and four HICANN-X chip carrier and jBOA ASIC-AB pairs. Component sizes are not to scale. *Color coding*: *Yellow* arrows correspond to the full-duplex high-speed links with one differential clock and eight differential data signals per ASIC and direction. Sixteen links and two clock pairs are routed per FMC and FMC+ connector, respectively, aggregating all links of single chips at separate IO banks of the FPGA. *Turquoise*: Bundled slow control signals providing JTAG and I2C connectivity as well as an asynchronous reset for each ASIC. *Green*: One differential 50 MHz reference clock signal per ASIC is forwarded to the PLL. *Red*: 3.3 V, 12 V and 1.8 V power is shared across all ASIC-ABs and supplied by the VCU118. *Blue*: 1.2 V and 2.5 V power are supplied externally. *Purple*: External reference clock and system start differential signals are forwarded to the VCU118 to optionally synchronize the experiment start.

connector is mainly used to distribute the high-speed connectivity for two ASICs (*yellow*) and facilitate power distribution (*red*). The high-speed links are connected in such a way that all signals corresponding to single ASICs are bundled at distinct IO banks within the FPGA. This is done to facilitate the clock distribution and ultimately the timing closure for the system. Slow control (*turquoise*), differential reference clock (*green*) and external synchronization signals (*magenta*) are routed through the FMC+ connector. The ASIC 1.2 V core and 2.5 V IO supply voltages are supplied externally (*blue*).

### 2.2.2 Mezzanine Card

The full schematic of the MezzaMix PCB is a twenty-three page document in hierarchical structure, consisting of pages with size up to A2. It is therefore not included in this document, but it can be accessed through the *Electronic Vision(s)* group internal *git* server in a dedicated `pcb-mezzamix` repository[7]. In the context of this thesis, the layout of the PCB will be discussed by referencing

[7] URL: `https://gerrit.bioai.eu:9443/c/pcb-mezzamix/+/22076`;
Commit hash at production state: `78c8b668a900c046c9444b8437ac8c4e98f8e48c`.

the manufactured and assembled board shown in Figure 2.6 and 2.7. Besides the components shown explicitly on the top and bottom surface of the PCB, the routing of signals on the inner layers is highlighted.

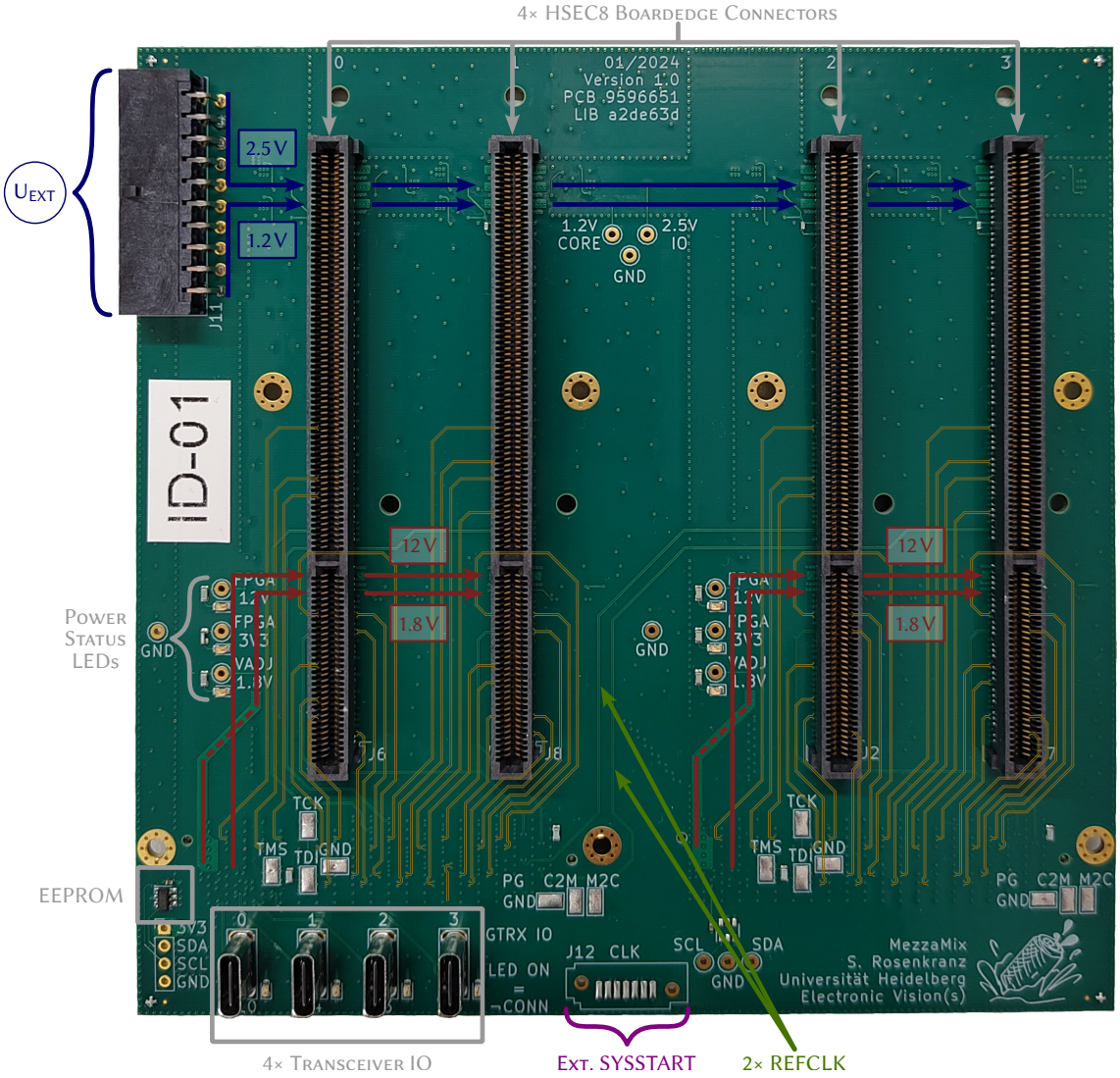



**Figure 2.6:** Photograph of the MezzaMix PCB top side. Connectors and other components are highlighted explicitly. *Color coding*: *Yellow*: High-speed differential clock and data signals routed on a dedicated inner layer. Additional traces are routed on the top and the bottom layer of the PCB and are not highlighted explicitly. *Red*: Routing of power signals supplied by the VCU118 on the top and a dedicated inner layer. *Blue*: Routing of externally supplied ASIC IO and core voltages on a dedicated inner layer. Each voltage rail is fed into the board through an equally split number of pins on a $2 \times 10$ pin through hole connector. *Green*: Highlighted routing of two traces for 50 MHz reference clock signals to the two rightmost ASIC-AB slots. The signals to the connectors on the left side are not highlighted for visual clarity. *Purple*: Landing pad of the connector[8] for optional system synchronization signals.

Figure 2.6 shows the top side of the MezzaMix card. It is an eight layer $13.9 \times 13.75\,\mathrm{cm}^2$ PCB. The central feature are four *Samtec HSEC8-1100-01* connectors (Samtec 2024b) that allow for boardedge

[8]Not assembled.

attachment of the ASIC-ABs. Their placement has been chosen such that the fanout of the critical high-speed LVDS signals (*yellow*) from the FMC and FMC+ connectors results in short trace lengths, while guaranteeing the landing of all signals from a single ASIC on a dedicated IO bank of the FPGA. The 1.2 V core and the 2.5 V IO supply voltages for the ASICs are provided through a *Wuerth WR-MPC3 66202021022* $2 \times 10$ pin through hole connector (Wuerth 2016) (*blue*). The 12 V auxiliary and 1.8 V FPGA IO voltages are routed on dedicated inner layers from the locations of the FMC and FMC+ connectors on the bottom, similar to the 3.3 V auxiliary supply voltage shown in Figure 2.7 (*red*). For these power rails, status *light-emitting diode*s (LEDs) are provided.

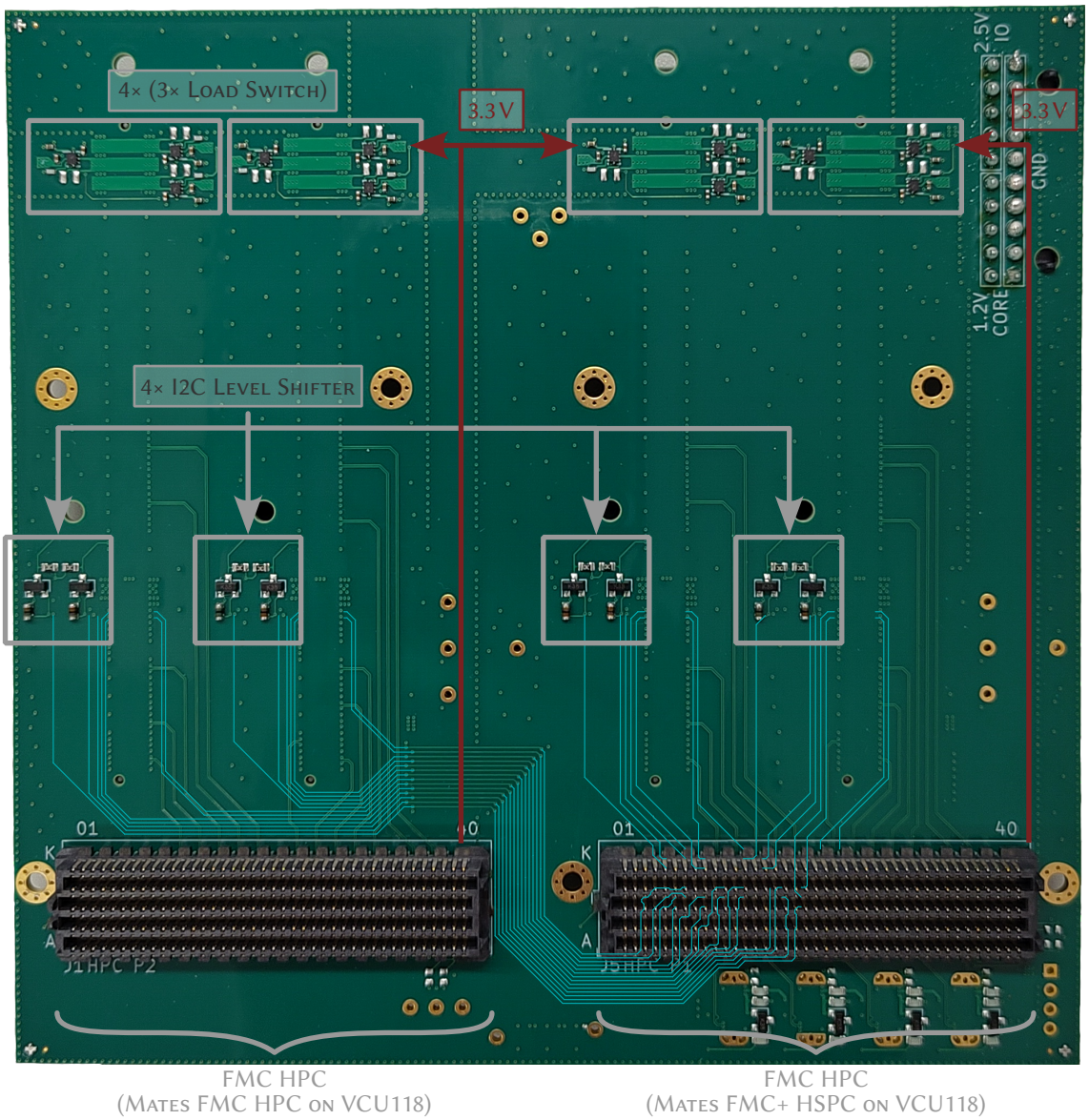



**Figure 2.7:** Photograph of the MezzaMix PCB bottom side. Connectors and other components are highlighted explicitly. *Color coding*: *Turquoise*: Single-ended slow control signals and asynchronous reset routed on a dedicated inner layer. *Red*: Routing of power signals supplied by the VCU118 on a dedicated inner layer.

The 50 MHz differential reference clock signals for the PLL are routed on the top side of the PCB (*green*). Since these signals require to be connected to some of the limited clock capable outputs

of the FPGA, their routing has been prioritized and the placement of other signals adapted to the resulting spatial constraints. A *Molex SATA 0678005005*[9] connector can optionally be assembled to provide system synchronization signals to the FPGA (*purple*). An I2C addressable 2 kbit *Microchip 24AA025UID Electrically Erasable Programmable Read-Only Memory* (EEPROM) (Microchip Technology Inc. 2013) is located at the bottom left and can be used to specify configuration data for the setting of the adjustable 1.8 V FPGA IO voltage. Also visible on the bottom left are four vertical *Hirose CX80B1-24P USB3.2 Type-C* receptacles (Hirose Electric Co., LTD. 2024) interfacing to the multi-gigabit transceivers.

The bottom side of the PCB shown in Figure 2.7 features two identical *Samtec ASP-134488-01* connectors (Samtec 2024a) that are mechanically and electrically compatible with both the FMC and the FMC+ connector on the VCU118. Slow control signals to each ASIC-AB are provided by the FMC+ mating connector and routed on a dedicated inner layer (*turquoise*). The I2C signals are passed through a level shifting circuit to convert from 1.8 V on the FPGA side to 3.3 V on the ASIC-AB side. The 3.3 V auxiliary supply voltage is provided by the VCU118 and routed from the FMC connectors to the upper edge of the board. Together with the 1.2 V and 2.5 V supply voltages, it is fed to distinct *Texas Instruments TPS22992RXP* load switches (Texas Instruments Incorporated 2021) that limit the inrush current on these rails towards the ASIC-AB such that power sequencing can be controlled.

### Layer Stack-Up

Due to the high pin count and dense arrangement of signals in the vicinity of the FMC connectors, a stack-up of eight layers for the MezzaMix PCB has been chosen. Table 2.1 summarizes the layers and their primary functionality.

### High-Speed Signal Integrity Measures

When the frequencies of transmitted signals on PCBs are high, the wavelengths of the frequency components that contribute to the signal eventually shrink to the order of the wire length that they are transmitted along. In this case, there may be multiple bits sent along the transmission line at the same point in time, which may lead to undesired mutual degradation of the digital pulses, if no special care is taken in the design phase to mitigate these effects. There is plenty of good literature on high-speed signal integrity such as for instance Hall 2009.

In many cases, approximative formulas are sufficient to describe the transmission properties of PCBs at high frequencies – at least relative to the accuracy that can be achieved by the manufacturer during the production process. To get an approximate idea whether special signal integrity measures have to be considered for the MezzaMix PCB, the following estimations have been made: In the case of the targeted multi-chip platform, the high-speed LVDS links between FPGA and ASIC represent the fastest changing signals, which run at a clock frequency of $f = 500\,\mathrm{MHz}$. The components with the highest frequency in the signal are defined by the rise and fall time of the square wave edges, which is not exactly known. So, for the following estimations $f = 500\,\mathrm{MHz}$ will be used to get a *best case* estimate for the critical harmonics.

[9] No data sheet was available by the manufacturer at the time of thesis submission; More information can be found, for instance, via third party distributors such as DigiKey.

| Layer name | Function |
|---|---|
| TOP | Connector landing pads<br>Routing of PLL reference clock<br>Routing of high-speed LVDS signals<br>Partial routing of slow control signals<br>Partial distribution of 1.8 V power rail |
| IN1 | Ground |
| IN2 | Routing of high-speed LVDS signals |
| IN3 | Distribution of 1.2 V, 2.5 V and 3.3 V power rails<br>Ground |
| IN4 | Distribution of 1.8 V, 3.3 V and 12 V power rails<br>Ground |
| IN5 | Routing of slow control signals |
| IN6 | Ground |
| BOT | Connector landing pads<br>Routing of high-speed LVDS signals<br>Partial distribution of 1.2 V, 2.5 V and 3.3 V power rails |

**Table 2.1:** Summary of layer functions for the MezzaMix PCB.

The wavelength $\lambda$ of a signal propagating along a transmission line can be given in terms of the relative dielectric permittivity $\epsilon_r$ of the dielectric used in the PCB:

$$\lambda = \frac{c}{\sqrt{\mu_r \epsilon_r} \cdot f}. \tag{2.1}$$

In this equation, $c$ corresponds to the vacuum speed of light $c = 3 \times 10^8$ m/s and $\mu_r$ to the relative magnetic permeability of the dielectric, which can be considered as $\mu_r = 1$ for most materials in this context. For transmission lines on a PCB, the exact value of $\epsilon_r$ often corresponds to an effective value of the various media that enclose the copper trace. A common dielectric used in PCBs is FR4 with a relative dielectric permeability of approximately $\epsilon_r \approx 4$. Hence, for the high-speed links between FPGA and ASIC

$$\lambda \approx \frac{3 \times 10^8 \,\text{m/s}}{\sqrt{4} \cdot 500 \,\text{MHz}} = 30 \,\text{cm}. \tag{2.2}$$

Equation 2.2 provides somewhat of an *upper boundary* for the *smallest wavelengths* of the contained harmonics, which very likely are significantly smaller. For instance, it is known that the LVDS drivers of the ASIC operate at a high, fixed slew rate, meaning that the signal contains harmonics with a comparably high frequency[10]. The etch length of the copper traces of the links along the four different PCBs in the setup shown in Figure 2.5 spans across several cm in length. This means that the smallest wavelengths contributing to the high-speed signals are expected to very

[10]Personal communication with Johannes Schemmel and Joscha Ilmberger.

likely be of the same order. Hence, considerations have been made to mitigate the most critical effects for high-speed signal transmission in the design phase, which will be outlined briefly in the following. As a guideline for this process, two application notes (Weiler, Pakosta, and Verma 2006 and Texas Instruments Incorporated 2018) provided by *Texas Instruments Incorporated* have been used.

### Propagation Delay

In differential signaling standards, the difference of two complementary signals of opposite polarity across a pair of wires is used to define the logical level of a digital signal. If the wire pair is routed properly, then this approach can help to cancel out noise coupling in from external sources, as the noise couples into both traces with the same polarity. Taking the difference of the individual signals then largely eliminates the disturbance. If, however, the lengths of the traces are mismatched, then the noise does not couple into both traces symmetrically and is not accounted for at the receiver. Additionally, the quality of the square signal is further degraded at the receiver by part of the signal difference coupling into the common mode, which is ultimately rejected. In most cases, the mismatch is due to board layout constraints not allowing for a symmetric routing of the traces, which can be accounted for by purposefully stretching the shorter trace close to the mismatched end. Care must be taken though, as the stretching – which is typically done by drawing a wavy trace shape – can lead to an increased susceptibility to crosstalk.

Due to the tight layout constraints posed onto the routing of the high-speed signals at the ball grid array of the FMC connectors, a maximum intra-pair skew $s$ of roughly $s \approx 860\,\mu\text{m}$ has been measured. Similar to before, the phase velocity $v_p$ of a harmonic of the signal on the transmission line can be approximated via

$$v_p = \frac{c}{\sqrt{\epsilon_r}} \approx \frac{3 \times 10^8\,\text{m/s}}{2}. \tag{2.3}$$

The resulting temporal delay $\Delta t$ of the signal between the mismatched wires hence corresponds to

$$\Delta t = \frac{s}{v_p} \approx 5.8\,\text{ps}. \tag{2.4}$$

Relative to the period $T = \frac{1}{f}$ of the 500 MHz high-speed clock, this corresponds to a mismatch of

$$\frac{\Delta t}{T} = \frac{5.8\,\text{ps}}{1\,/\,500\,\text{MHz}} \approx 0.3\,\%. \tag{2.5}$$

For the fastest signal components, this ratio is expected to be larger. However, regarding the FPGA side of the links, the *ANSI/VITA Standard* specifies the skew between user defined pins to be kept smaller than 10 % of the unit interval for the targeted data rates to maximize performance (VITA 2019). As the maximum skew on the MezzaMix is expected to be well below this, the propagation delay has not been optimized further.

### Impedance Matching

Another important guideline to consider for high-speed signal routing is to account for impedance discontinuities along the full path of the signal between transmitter and receiver. When an electromagnetic wave propagates across the interface of two transport media with different characteristic impedances $Z_0$ – for instance through a connector between two different PCBs – then part of the wave is transmitted through the interface to the next transmission line while part of it is reflected back along the initial transmission line to the source. This phenomenon can lead to over- and undershoots of the transmitted wave and ultimately a degraded signal quality at the receiver. For two transmission lines 1 and 2 with characteristic impedances $Z_{01}$ and $Z_{02}$, respectively, the reflection coefficient $\rho$ provides a measure for how much of the wave's amplitude is reflected back to the source:

$$\rho = \frac{Z_{02} - Z_{01}}{Z_{02} + Z_{01}}. \tag{2.6}$$

In the extreme case, where $Z_{02} \to \infty$ – corresponding to an open circuit – then $\rho \to 1$. That is, all of the wave's amplitude is reflected. In order to minimize the reflections, $\rho = 0$ demands that $Z_{01} = Z_{02}$. So the characteristic impedances of the different transmission lines need to be matched. For single-ended signaling, typically $Z_0 = 50\,\Omega$ is targeted, while for differential signaling $Z_0 = 100\,\Omega$ is used – given that the transmission line is terminated accordingly at the receiver.

The characteristic impedance $Z_0$ of the transmission line is constrained by the properties of the employed materials on the PCB and the specific geometry used. Therefore, most manufacturers provide impedance calculators for their respectively employed production process. This allows the designer to adapt the trace widths and spacings – in the case of differential signaling – to match the targeted impedances. During the design process of the MezzaMix PCB, the constraints provided by a *defined generalized layer stack-up* from *Multi Leiterplatten GmbH* (Multi Leiterplatten GmbH 2024) have been used to fix the differential trace geometries on the different layers of the PCB. To match the targeted impedances on the inner layers, the trace widths $w$ and differential pair spacings $h$ have been cross-checked with an impedance calculator tool in *Cadence Allegro*. Note that in general, to fully account for critical effects, extensive simulations to solve Maxwell's equations for the given geometry and boundary conditions need to be performed to compute the characteristic impedance $Z_0$, which has been considered to not be within the scope of this thesis[11]. Additionally, the impedances of the single-ended slow control signals have not been optimized as they are not expected to greatly diminish the signal quality at low frequencies[12]. The resulting design parameters are summarized in Table 2.2.

| PCB layer | Targeted impedance $Z_0$ [Ω] | Trace width $w$ [µm] | Pair spacing $h$ [µm] | Approximated impedance $Z_0$ [Ω] |
|---|---|---|---|---|
| TOP and BOT | 100 | 120 | 115 | 98.6 |
| IN2 | 100 | 100 | 160 | 97.5 |

**Table 2.2:** Impedance matched differential signal geometries for the MezzaMix PCB.

[11] Personal communication with Joscha Ilmberger.
[12] Personal communication with Joscha Ilmberger, Andreas Grübl and Lars Sterzenbach.

#### Crosstalk

Lastly, measures have been taken to minimize the susceptibility of individual traces to crosstalk. The term *crosstalk* describes the mutual influence of closely routed traces due to inductive and capacitive coupling. The trace that carries the signal – referred to as the *aggressor* – influences another nearby trace – called the *victim* – by inducing forward and backward currents in the victim trace. For differential signal pairs, the tight coupling between the aggressor and victim is actually desired, but this can only be achieved, if other differential signal pairs are kept sufficiently far away. For high-speed data transmission, a minimum distance of five times the width $w$ of the traces in the differential pairs is recommended, which has been adhered to during the routing process for the MezzaMix PCB.

### 2.2.3 FPGA Implementation

With the hardware stack-up completed by adding the MezzaMix PCB, the next step in the setup commissioning procedure is to realize the communication between the VCU118 and a single HICANN-X. To this end, the basic FPGA implementation used in, for instance, the Cube systems must be adapted to the new platform. In general, this requires only minor changes regarding the IO constraints, clocking and memory utilization. No major re-design of the communication protocol is needed as the Kintex-7 and the Virtex UltraScale+ architectures are similar enough and the FPGA on the VCU118 offers considerably more logic resources such that small-scale optimizations are not considered to be required for the moment[13]. In the following, a brief overview over the implemented FPGA design will be provided and differences to the current state of the Cube systems highlighted.

#### Constraints

The assignment of the user definable pins of the FMC and FMC+ connectors relative to the MezzaMix PCB requires to specify IO constraints in order to set the IO cells to the desired signaling standard. As a basis for this step, a publicly available, open source, predefined constraints list[14] has been used and adapted further. The constraints are specified by means of *Xilinx Design Constraints* (XDC) commands of the following form:

```
set_property -dict {LOC P37 IOSTANDARD LVDS DIFF_TERM TRUE} \
   [get_ports {I_CHIP_DAT_RX_P[0][1]}]
set_property -dict {LOC N37 IOSTANDARD LVDS DIFF_TERM TRUE} \
   [get_ports {I_CHIP_DAT_RX_N[0][1]}]
```

This example defines the two signals routed to pin `P37` and `N37` of the FPGA fabric as an input LVDS pair with enabled 100 Ω differential termination. The signal names such as `I_CHIP_DAT_RX_P[0][1]` need to match the declaration in the top-level port list of the design. This denomination corresponds to the differential signal with positive polarity coming from link 1 of the ASIC located at slot 0 on the MezzaMix PCB. Similar constraints have been added for the other high-speed links and the slow control signals towards the HICANN-X.

[13] Personal communication with Joscha Ilmberger.
[14] Provided by Alex Forencich via *GitHub*: `https://github.com/alexforencich/verilog-ethernet`.

### Top-Level

A schematic of the MezzaMix FPGA top-level for single-chip communication is provided in Figure 2.8. It corresponds to a mixture of existing FPGA implementations for the Cube systems and for host communication to the VCU118 via Ethernet[15]. In the latter, an interface has been introduced to connect the *Host Automatic Repeat-Request* (Host ARQ) protocol – used as a transport layer for host communication in the BSS-2 architecture – and an *User Datagram Protocol* (UDP) core that ultimately establishes the connection to an Ethernet port on the VCU118. The MezzaMix FPGA top-level displayed in Figure 2.8 implements features of both top-level specifications.

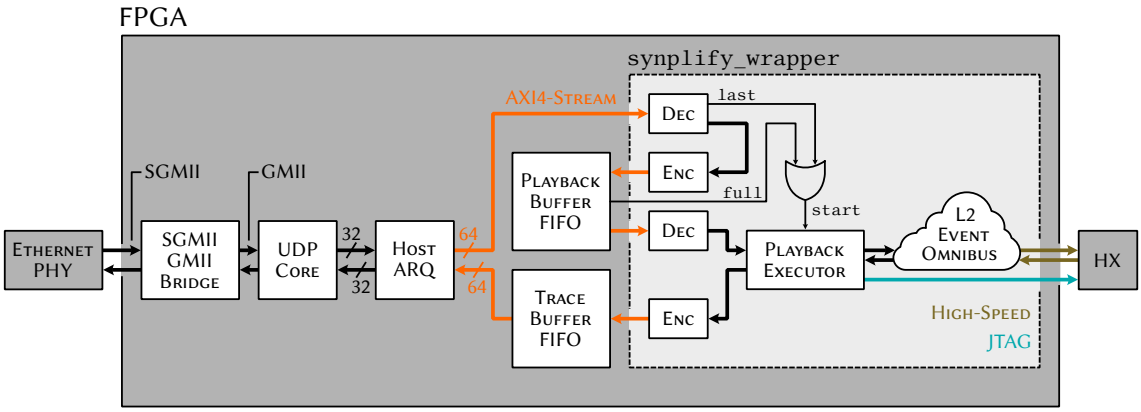



**Figure 2.8:** Schematic representation of the MezzaMix FPGA top-level for single-chip communication. Two FIFOs buffer the playback program and the response data to and from the playback executor. The transmission of application data to the Host ARQ and the `synplify_wrapper` is performed via an AXI4-Stream bus protocol.

Within the FPGA implementation of the BSS-2 architecture, the major part of the experiment control logic is wrapped in a `synplify_wrapper`[16] module. This module can be thought of as the single highest level entity within the FPGA implementation that can communicate with the HICANN-X ASIC. For the targeted multi-chip platform, the `synplify_wrapper` can be considered as a fixed design component and hence is implemented as-is for the MezzaMix top-level[17]. It contains a *Playback Executor* module that allows for the timed release of events to the HICANN-X. It does so by executing pre-buffered *Playback Programs* that contain a sequence of timed read and write instructions that, for instance, may be used to modify the weight of a specific synapse during runtime. The response data from the HICANN-X needs to be buffered by the FPGA to be accessible to the user.

In order to be able to send and receive data to the HICANN-X from a host computer, application data must be transferable from the Host ARQ to the `synplify_wrapper`. Similar to the Cube FPGA implementation, this in done through a 64 bit *Advanced eXtensible Interface* (AXI) for the MezzaMix FPGA top-level (*orange*). Data transferred from the Host ARQ to the `synplify_wrapper` is first piped to a *decoder* stage that translates the application data to instructions for the Playback Executor[18].

[15] URL: `https://gerrit.bioai.eu:9443/c/hxfpga/+/19261/6`;
Commit hash `8b663f304877d7514c807e9703458a630b549d25`.

[16] For legacy reasons, the synthesis of the `synplify_wrapper` is done in *Synopsys Synplify*, hence the denomination of the module.

[17] Personal communication with Joscha Ilmberger.

[18] The decoder and encoder stages are explained in more detail in Section 3.1.1.

In an additional *encoder* stage, the instructions are encoded back to 64 bit words and streamed to a *playback buffer*. Within the MezzaMix FPGA implementation, the buffering is realized by employing a Xilinx *intellectual property* (IP) *first-in-first-out buffer* (FIFO) with a depth of 8192. Due to the large depth desired for the buffering of playback data, *UltraRAM* (URAM) blocks[19] are utilized for the FIFO. The buffered data is then held back until the Playback Executor releases the newly decoded instructions via the high-speed communication interface – denoted as *Layer 2* (L2) – for the transmission of timed event data and *Omnibus* configuration data[20] to the HICANN-X. The execution of the playback program is either started when the decoder recognizes that the last instruction has been sent or when the playback buffer is almost full.

The response data from the HICANN-X is then encoded again to 64 bit words and stored in a *trace buffer* FIFO. In this implementation, the depth is again set to 8192, which is a rather arbitrary choice as the VCU118 FPGA offers a vast amount of logic resources. Optimization of the memory utilization could be done when extending the implemented design for communication with more than one ASIC. When the user requests the readout of the response data, the buffered data is streamed back to the Host ARQ again.

Table 2.3 summarizes the resource utilization for the full communication logic from a host computer to the HICANN-X in the VCU118 FPGA.

| Resource | Available | Full Design Utilization | Full Design Rel. Utilization | `synwrap` Utilization | `synwrap` Rel. Utilization |
|---|---|---|---|---|---|
| LUT | 1182240 | 39535 | 3.34 % | 19458 | 1.65 % |
| LUTRAM | 591840 | 3102 | 0.52 % | 1794 | 0.30 % |
| FF | 2364480 | 40757 | 1.72 % | 14134 | 0.60 % |
| BRAM | 2160 | 167 | 7.73 % | 50.5 | 2.34 % |
| URAM | 960 | 12 | 1.25 % | 0 | 0 % |
| DSP | 6840 | 42 | 0.61 % | 39 | 0.57 % |
| IO | 832 | 274 | 32.93 % | 0 | 0 % |
| BUFG | 1800 | 16 | 0.89 % | 2 | 0.11 % |
| MMCM | 30 | 2 | 6.67 % | 0 | 0 % |
| PLL | 60 | 5 | 8.33 % | 0 | 0 % |

**Table 2.3:** Resource utilization of the VCU118 FPGA for single chip communication. Summarized are the resources spent on the entire design and on a single instance of the `synplify_wrapper` (`synwrap`). Note that the IO resource utilization of the full design implies the constraints specified for the full available pin out on the FMC and FMC+ connectors, that is, for four directly attached HICANN-X ASICs.

The relative utilization of the IO resources of the full design displayed in Table 2.3 includes the entire pin out of the FMC and FMC+ connectors, that is, for four directly attached HICANN-X ASICs. While the design utilizes about 7.7 % of the available *block random-access memory* (BRAM), the `synplify_wrapper` only accounts for roughly 2.3 % of the available elements. Also, the utilization of the remaining logic elements is quite small such that it is expected that the experiment control logic can be instantiated multiple times for additional chips without reaching capacity limits soon.

[19] Essentially, this is high storage capacity BRAM.
[20] Initial configuration of the HICANN-X is performed via JTAG until the high-speed interface is accessible.

The full FPGA implementation for the MezzaMix is available via *git* in the `hxfpga` repository[21].

### 2.2.4 Single-Chip Setup

After completing the hardware stack-up with the MezzaMix PCB and adapting the Cube and VCU118 Ethernet FPGA implementations to allow for full Host-to-ASIC communication, the multi-chip platform has been assembled. Figure 2.9 and 2.10 display the setup during operation with a single HICANN-X. The programming of the FPGA is done using a JTAG adapter, while the host communication to the ASIC is established via Ethernet.

**Figure 2.9:** Top-view photograph of the assembled multi-chip setup during operation with a single HICANN-X.

First connectivity tests for the connection between host and FPGA, as well as between FPGA and ASIC have been performed, which will be summarized briefly on a qualitative level: Loop-back data tests between host and FPGA were successful, including the transmission of application data to and from the Playback Executor. Additional loop-back tests have been performed to test the data throughput on the high-speed links between the FPGA and the HICANN-X. Compared to the Cube systems, a lower throughput at the high-speed interface has been determined. The degraded performance could potentially be due to electrical problems on the MezzaMix PCB or uncompensated clock-to-data skew at the FPGA high-speed LVDS de-serializers.

[21] URL: `https://gerrit.bioai.eu:9443/c/hxfpga/+/22197`;
Commit hash at state of writing this thesis: `62b414f012cef9179e5d4dc571306844d4a0d792`.

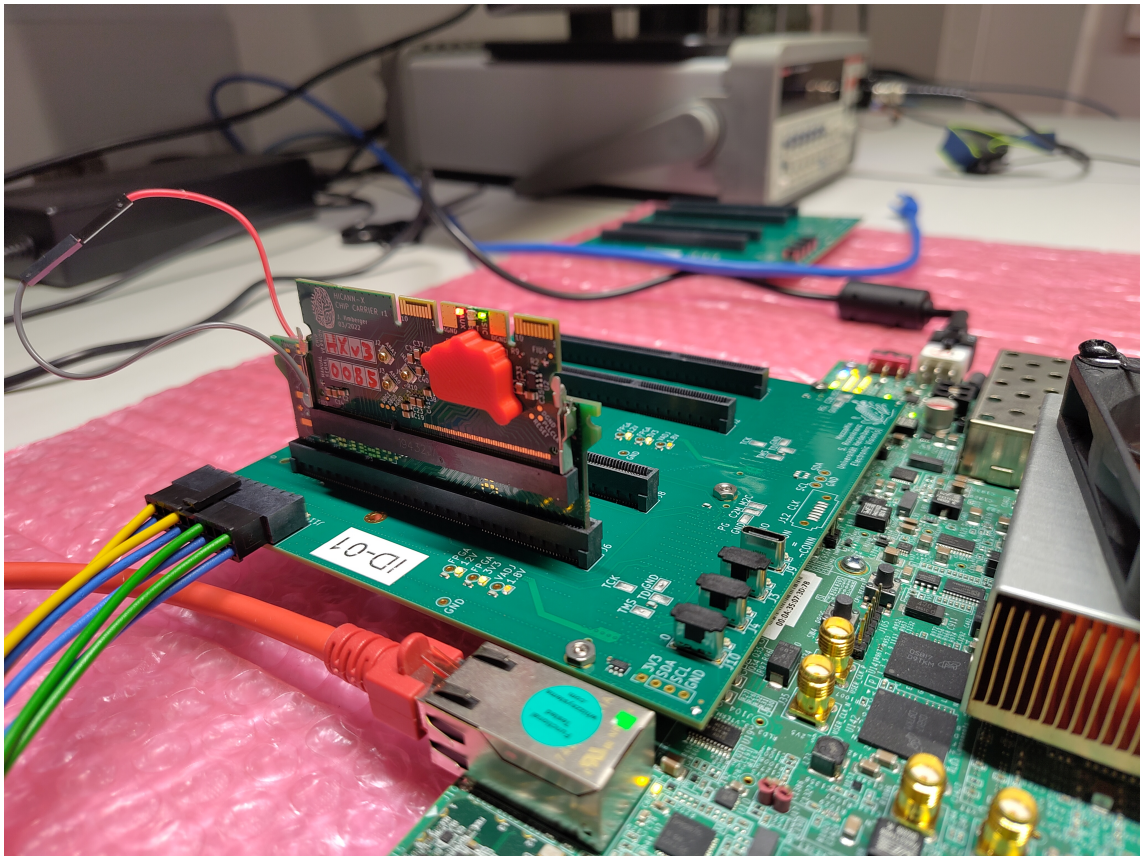


**Figure 2.10:** Close-up photograph of a chip carrier board during operation in the multi-chip setup.

However, it was not possible to further investigate the performance of the high-speed links on the multi-chip platform due to a defect of the VCU118 rendering the board insusceptible for additional tests.

## 2.3 Summary and Discussion

Up to this point, an operational single-chip setup with the VCU118 has been commissioned. The development included the design of the MezzaMix PCB – a mezzanine board to the VCU118 – that closes the gap in the hardware stack to connect up to four HICANN-X ASICs to a single FPGA. Additionally, the FPGA implementation for single-chip communication with the Virtex UltraScale+ FPGA on the VCU118 has been fully established for each of the four individual slots. First connectivity tests have been performed, showing promising results regarding the operation with a single HICANN-X.

The single-chip system can be extended into a *multi-single-chip* setup that would allow for communication between host and multiple HICANN-X ASICs through a single FPGA. In order to complete the system in this fashion, additional experiment control blocks – that is, an additional `synplify_wrapper` for each ASIC – would need to be instantiated, accompanied by an attached Host ARQ block, respectively. To address the individual ASICs, a multiplexing stage would need to be introduced between the UDP core and the Host ARQ instances.

To develop the platform into a *full multi-chip* system, the FPGA implementation needs to be extended further by means of a router core that interconnects the experiment control logic on the L2 level. This core is currently under active development.

With the defect of the VCU118, however, it was not possible to continue these efforts in the scope of this work, as no second board for replacement was available within the thesis time constraints. This also highlights a crucial drawback of the design choices outlined in Section 2.1.1: In the targeted star topology for the interconnection of chips, the central node is the most critical network constituent. If it fails, the connection of the entire network breaks down until the failure is corrected or the node replaced. Without adapting the topology, there is little protection of the network against such failures. This issue needs to be considered in further attempts to extend the BSS-2 architecture for multi-chip operation.

# 3 High-Speed Link Stabilization

During experiment execution with the BSS-2 family of devices, experiment data needs to be transferred between the experiment control host and the neuromorphic substrate. This includes, but is not limited to, experiment configuration data such as neuron parameters, real time injection of spike events, or readout of experiment data. This information is used for network re-configuration or further processing in the evaluation of the experiment. As for example described in Karasenko 2020, the injection of spike events requires precise timing due to their analog and asynchronous nature and the 1000-fold acceleration factor compared to biological real time of the analog substrate.

In addition to posing critical demands on latency and jitter, the small time scales in play call for a high-speed event interface between the control host and the substrate. The HICANN-X therefore employs LVDS (de-)serializers (Scholze et al. 2012). A variant of the (de-)serializer macro also exists at the L2 of the FPGA design within the experiment control unit (cf. Figure 2.8) by employing, for example, the *input serial-to-parallel converter* (ISERDESE2) and *output parallel-to-serial converter* (OSERDESE2) primitives provided for Kintex-7 FPGAs (Xilinx, Inc. 2018).

Within the BSS-2 architecture, eight (de-)serializer pairs form the high-speed communication interface between the host FPGA and the ASIC (Karasenko 2020). Each of the (de-)serializer pairs corresponds to a point-to-point, byte aligned, full-duplex link with one differential data pin and a shared differential clock pin for all eight data pins per direction. The data transfer in each direction is realized source synchronously in *double data rate* (DDR) fashion at a clock frequency of 500 MHz, allowing for a data transfer rate of 1 Gbit/s[1] per each of the eight links[2].

The (de-)serializer macros described in Scholze et al. 2012 implement an automatic link-training mechanism that is run during link initialization. This routine aims to align the data patterns sent by the serializer at the transmitting end and seen by the de-serializer at the receiving end. The mechanism works by using a combination of programmable delay elements and flip-flop registers in the de-serializer. This allows for compensation of clock-to-data skew – for instance accumulated through the routing on the PCB – and bitwise alignment of the delayed data pattern to correctly recognize the first and the last bit in the transmitted word. For a more detailed description of the link training mechanism the reader may be referred to Section 3.2.

Section 2.2.4 describes performance limitations in the high-speed link communication between FPGA and ASIC when performing a high-speed data loop-back test in a single-chip configuration on the MezzaMix setup. Similar, yet less frequent issues have been observed on other systems as well, for instance on the Cube or the BSS-2 mobile systems. Due to a defect on the VCU118 – rendering the board insusceptible for further analysis of this problem on the multi-chip platform

[1] Technically, the ASIC PHY can operate at data rates up to 2 Gbit/s. The data rate is limited by the FPGA.

[2] As a side note, a single link is commonly referred to as a PHY in the BSS-2 architecture, a convention that will be adopted and used interchangeably throughout the chapter.

– the decision was made to continue with the investigation on the existing Cube systems[3].

The Cube and the MezzaMix setups are not exactly comparable as they come with different PCB layouts, transmission line lengths, impedance discontinuities across connectors and so on. Yet, first qualitative investigations on the MezzaMix setup have at least shown some similarities in the behavior of the links to the Cube systems. There are single links that tend to fail more often than others, failures seem to occur on different timescales – that is, almost immediately after link initialization as well as some seconds later – and some links tend to fail a second time, immediately after they have been re-trained due to a first detected failure.

In this chapter, a deeper analysis of the stability of the high-speed links on BSS-2 systems will be given. Section 3.1 provides an overview over the relevant statistics that will be used to quantify the stability of the links and introduces a routine to sample them systematically. As a representative example for the current state of the high-speed links, stability results from a Cube system are presented. Section 3.2 explains the training algorithm run during link initialization in detail and shows trace data from an in-hardware debugging method for both working and failing link trainings. Corrections to the observed flaws in the training procedure are presented and the performance of the updated algorithm is evaluated. The chapter concludes with an investigation of the influence that the configuration of the ASIC receiver exerts on the link stability in Section 3.4 and a discussion of the gathered results in Section 3.5.

## 3.1 Prerequisites and Current State

This section aims to define a criterion suitable for evaluating the stability of the high-speed links. After introducing the relevant statistics to be investigated and outlining a routine to systematically test the link stability on different setups an example of the current state of the high-speed links is provided.

### 3.1.1 Stability Criterion

The usual way to characterize the *qualitity* of a link is to define a *bit error rate* that counts the number of bits which have been altered on their way between the transmitting and receiving end of a communication channel. In many cases this can be caused by electrical problems such as noise on the transmission line or crosstalk coupling in from neighboring data channels. Another cause of failure can be synchronization issues that may lead to false detection of the bit logic value by sampling the voltage level on the transmission line at suboptimal points in time. However, the raw bit error rate is not always easy to quantify. It requires access to and knowledge about the raw data transmitted via the communication channel by digging deep into the *PHYsical layer* (PHY) of the system under test, which is not always possible. To circumvent this issue, it is useful to define the quality of the link on a different level of abstraction, for instance by assessing the data handled on the link layer.

In case of the BSS-2 systems, the link layer is comprised of so called *Universal Translator* (UT) modules that interface to the (de-)serializer and serve as encoders, respectively decoders, between

[3] Personal communication with Johannes Schemmel

data words accessible to the user on the application layer and bytes transmitted on the PHY (Karasenko 2020). The encoding scheme in the UT offers the possibility to append *cyclic redundancy check* (CRC)[4] information to a data word which is being handed over to the serializer, allowing for automated error detection at the far end of the communication channel. Consequently, the UT module also provides a link checking mechanism that is able to detect link errors by evaluating CRC data. When CRC errors are detected, the UT module issues a re-training command to the (de-)serializer, rendering the corresponding link unaccessible for transmission of user data until it is re-initialized anew.

It is important to note that a single CRC error is not the sole condition that can lead to a re-training of a link. In particular, the UT requires two consecutive CRC errors to be reported, while it also implements additional sanity checks to monitor the health state of the link. More specifically, one of the following conditions must be met:

1. Two consecutive CRC validations must fail,

2. the validation of header information specific to the employed encoding scheme must fail or

3. a user configurable timeout since the last successful CRC validation is met.

For a more detailed description of this behavior the reader may again be pointed to Karasenko 2020.

Important to note is that the re-training of a link after failure cannot be conducted independently for the two endpoints – FPGA and ASIC – of the communication channel. Triggering the re-training on either of the two endpoints leads to transmission of a predetermined training signal to the other link partner, who now essentially tries to decode garbage data (Karasenko 2020). By failing to validate the non-existing header information in the training pattern, the UT within the second endpoint issues a re-training command as well, ultimately resulting in a complete re-training of the link.

Besides the link checking mechanism, the UT also provides information about the link status to the user by accessing the event queue of the Playback Executor. A 5 bit status value – that is, one bit for each of the failure conditions listed above and two additional bits for a successful training after system initialization and re-training after failure – together with the 3 bit ID of the corresponding link is annotated with a 43 bit time stamp and buffered for readout to the experiment host. Thus, whenever a link is (re-)initialized or a link check condition is met that leads to a re-training of a link, a status notification is generated. In the current implementation, this information is however limited to the FPGA side of the communication channel. The peculiarity of the link being re-trained in both directions – regardless of the location at which the first critical failure has been detected – still guarantees that all link failures are reported, although one needs to be careful when interpreting the status information as it only gives insight about the FPGA side of the link.

Due to its well-defined implementation and ease of access, the time-annotated status information of the UT located in the FPGA can be considered as a suitable data structure to gain insight about the link stability. Since not only the failures but also the (re-)initializations of a link are reported

[4] Roughly speaking, a CRC involves polynomial division of the data word to be transmitted to calculate a check value which can be checked against by performing the same calculation on the received data word.

and time-stamped, it is possible to construct well-defined time intervals in which the link has been *stable*, that is, in a state after a successful (re-)initialization with no error having been detected yet.

### 3.1.2 Time to First and Second Failure

With the considerations about a suitable stability criterion outlined in Section 3.1.1, it is possible to define statistics that encapsulate the link stability more rigorously.

Let $t_1^{\text{init}}$ denote the point in time at which the UT reports a link to be fully trained and operational after a system init in units of system clock cycles of the FPGA. Further, $t_1^{\text{fail}}$ denotes the point in time at which the link reports the first failure after initialization according to the conditions listed above. One can then define the *time to first failure* $\tau_1$ as follows:

$$\tau_1 = t_1^{\text{fail}} - t_1^{\text{init}} \in \left[1, 2^{43} - 1\right], \tag{3.1}$$

assuming that $t_1^{\text{init}}$ can in principle be synchronized to the initialization of the FPGA system time counter.

It is possible to define the *time to second failure* $\tau_2$ in a similar manner. Let $t_2^{\text{init}}$ denote the point in time at which a link has been *re*-initialized after a first error and $t_2^{\text{fail}}$ the point in time at which the first error after re-initialization of the link occurred. It holds $0 < t_1^{\text{init}} < t_1^{\text{fail}} < t_2^{\text{init}} < t_2^{\text{fail}}$. Then

$$\tau_2 = t_2^{\text{fail}} - t_2^{\text{init}} \in \left[1, 2^{43} - 1 - \tau_1\right]. \tag{3.2}$$

Additional time to failure intervals may be defined for the third, fourth and other consecutive errors, yet for simplicity only the first two errors will be examined throughout this chapter.

Distinguishing between $\tau_1$ and $\tau_2$ in contrast to using flat time intervals is important as both statistics may not necessarily stem from the same statistical distribution. For instance, the temporal evolution of the system under test up to $t_1^{\text{fail}}$ always involves a complete reset of the FPGA and the ASIC followed by simultaneous initialization processes of up to eight links. This is in stark contrast to the time evolution of the system between $t_1^{\text{fail}}$ and $t_2^{\text{fail}}$, where re-trainings of multiple links may happen, yet are considered to be less deterministic.

A useful property of the definitions in equations (3.1) and (3.2) is that the subtraction of the offsets $t^{\text{init}}$, which may be specific to a given link or even entire setup, enables a comparison of $\tau_{\{1,2\}}$ across multiple links and systems.

Lastly, by using the known system clock frequency of the FPGA $f_{\text{clk}}^{\text{sys}} = 125\,\text{MHz}$[5], the time intervals in equation (3.1) and (3.2) can be converted to more concise time scales, for instance to

[5]Frequency stability $\Delta f/f = 50\,\text{ppm}$, the error is considered as negligible.

seconds:

$$\tau' = \frac{\tau}{1\,\mathrm{s} \cdot f_{\mathrm{clk}}^{\mathrm{sys}}}. \tag{3.3}$$

Throughout this chapter, the convention is adopted to drop the prime of the rescaled time to failure intervals and simply write $\tau$ instead of $\tau'$.

### 3.1.3 Testing Routine

To investigate the time to first and second failure $\tau_1$ and $\tau_2$, a systematic testing routine has been developed. It builds upon a performance test known as `perftest` that has formerly been used to measure the throughput of looped back data between the host FPGA and the ASIC (Karasenko 2020). Broadly speaking, the `perftest` implements a *finite state machine* (FSM) that increments a counter value which is then looped back between the FPGA and the ASIC, mimicking data transfers for a user specified testing duration. During the test, the UT performs all of the link checks listed in Section 3.1.1. This allows to read out the status notifications of all eight links after completion of the `perftest`, providing the FPGA time stamps at which a link has been initialized, failing due to one of the three link checking conditions or re-initialized.

However, the timestamps returned by the `perftest` cannot be used in the link stability test setting without making some changes to the procedure that establishes the digital communication to the ASIC. The current software stack encapsulates this in a `DigitalInit` class on the runtime control layer `stadls`. In a typical experiment, this initialization procedure starts with the configuration of the supply voltages for the ASIC and a chip-side reset before initializing the PLL on the ASIC via JTAG. Afterwards, the high-speed links are enabled and the FPGA and the ASIC enter a synchronization step to align their respective system time bases. Finally, the recording of events is enabled and the initialization procedure advances to the configuration of analog parameters of the ASIC.

Since the synchronization of the system time bases is done via the high-speed links, it is required that the links have already been fully initialized. This means that the notification for the first successful link initialization from the UT does not contain a valid time stamp, as the system time is reset by the synchronization procedure. Hence, in order to get an unbiased time to first failure $\tau_1$ according to equation (3.1) it is necessary to *disable* the system time synchronization step in the `DigitalInit`. Additionally, the recording of events needs to be activated *before* the enabling of the high-speed links to receive *unbuffered* link status notifications.

With these changes, the `perftest` can be applied repeatedly for a fixed duration to sample the distribution of $\tau_1$ and $\tau_2$ for the various links. The modifications to the `DigitalInit` are encapsulated in a *git* commit in the `haldls` repository[6].

[6] URL: `https://gerrit.bioai.eu:9443/c/haldls/+/23378`;
Commit hash at state of writing this thesis: `a53b0e897fa53fa7807113ba56af9bbd49e4f2e2`.

### 3.1.4 Example: Development Cube System

With the testing procedure outlined in Section 3.1.3, first tests of the link stability have been carried out on Cube systems X0/W60F0, X0/W60F3 and X5/W65F3 which serve as test setups for FPGA development. For simplicity, the following example is confined to setup X0/W60F0. The hardware components used in the Cube system under test are summarized in Table 3.1.

| Hardware component | ID |
|---|---|
| iBoard | 00 |
| xBoard | 23-13 |
| Chip Carrier | 0x640CF1 |

**Table 3.1:** Hardware components of Cube system X0/W60F0 corresponding to data shown in Figure 3.1, 3.2, 3.3 and 3.4.

The `perftest` has been performed repeatedly for $N = 1372$ times at a testing duration of 60 s with all PHYs enabled. The testing duration has been chosen to strike a balance between the included time scales at which errors can be observed and a sufficient sampling of the distributions of $\tau_1$ and $\tau_2$. As the analog substrate of the ASIC operates with a 1000-fold acceleration factor compared to biological time scales, this roughly corresponds to 16.7 h in biological time. This should be sufficient for most current experiments, but some cases may not be addressed in which a longer link uptime is required[7]. Other parameters set by the `DigitalInit` have been left at standard settings. Table 3.2 summarizes the `perftest` parameter settings.

| Parameter | Setting |
|---|---|
| Duration | 60 s |
| Repetitions | 1372 |
| PHYs enabled | $\{0, 1, 2, 3, 4, 5, 6, 7\}$ |

**Table 3.2:** Parameters of the `perftest` generating data shown in Figure 3.1, 3.2, 3.3 and 3.4.

Figure 3.1 displays the distribution of the times to first failure $\tau_1$ for PHY 5. The number of observations with a link failure per bin $N_{\text{bin}}$ is normalized to the total number of observations $N_{\text{obs}}$. Hence, the bin height $N_{\text{bin}} \,/\, N_{\text{obs}}$ corresponds to a *frequentist* probability to find a first error within the time interval $[\tau_1, \tau_1 + \Delta\tau_1]$, where the bin width $\Delta\tau_1$ corresponds to $\Delta\tau_1 = 0.3\overline{3}$ s. The sum of failure counts over all bins normalized to the total number of observations,

$$p_{\text{fail}} = N_{\text{obs}}^{-1} \cdot \sum_{\text{bin}} N_{\text{bin}}, \tag{3.4}$$

yields the frequentist probability for the PHY to fail within the considered time window of 60 s.

Since the probability in equation (3.4) is linked to events that can be encoded as binary data – either the link exhibited an error during the considered time interval or not – the estimated failure probabilities $p_{\text{fail}}$ should correspond to estimating the characteristic parameter $p$ of a Bernoulli

[7] Personal communication with Philipp Spilger, Yannik Stradmann and Jakob Kaiser.

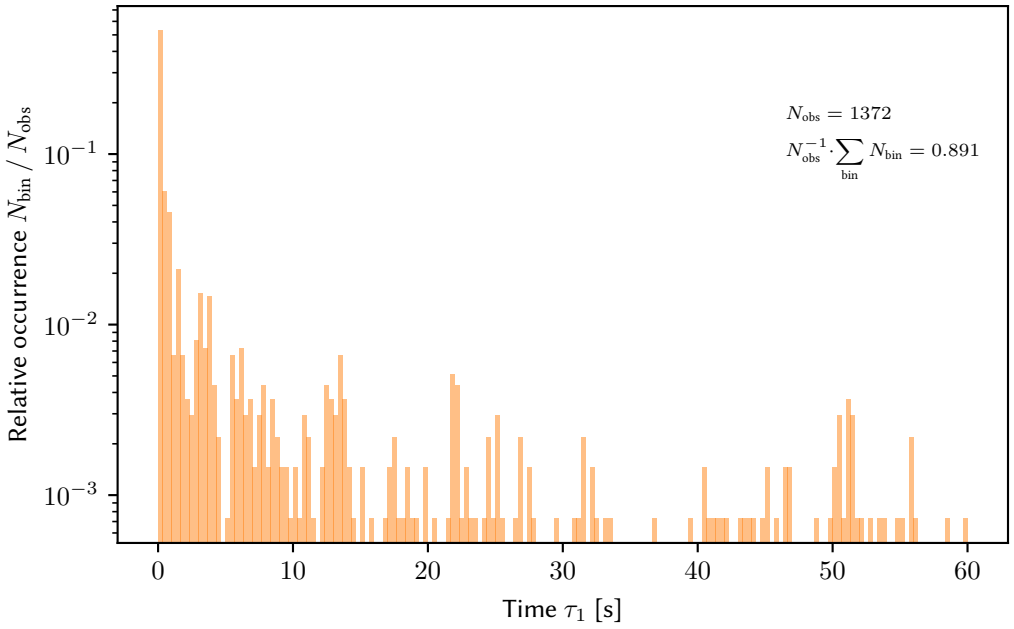



**Figure 3.1:** Relative number of link failures after first initialization for PHY 5. The histogram bin width corresponds to $\Delta\tau = 0.3\overline{3}\,\text{s}$.

distribution

$$f(x) = P(X = x) = \begin{cases} 1-p & \text{if } x = 0 \\ p & \text{if } x = 1 \\ 0 & \text{else} \end{cases}, \tag{3.5}$$

with the outcomes of the Bernoulli experiment defined as $x = 0$ corresponding to *no error having occurred* on the link and $x = 1$ corresponding to *an error having occurred*.

The standard deviation of the Bernoulli distribution given in equation (3.5) is defined as

$$\sigma = \sqrt{p \cdot (1-p)}. \tag{3.6}$$

Hence, the value of $p_{\text{fail}}$ can be considered as sampling the mean of the parameter $p$, equipped with a *standard error of the mean* (SEM) of

$$\text{SEM}(p_{\text{fail}}) = \frac{\sigma}{\sqrt{N_{\text{obs}}}} = \sqrt{\frac{p_{\text{fail}} \cdot (1 - p_{\text{fail}})}{N_{\text{obs}}}}. \tag{3.7}$$

The expression for the SEM given in equation (3.7) will be used throughout this chapter to quantify the statistical uncertainty of $p_{\text{fail}}$ for a repeated `perftest` with a given time window.

Although the distribution is somewhat under sampled at values $\tau_1 > 10\,\mathrm{s}$, Figure 3.1 shows that errors on this link seem to occur on all time scales that are viable given the limited testing duration. Furthermore, the distribution of the times to first fail is not uniformly shaped. It rather exhibits a peak at small $\tau_1$, with approximately 50 % of the total number of failures being contained within the first 333 ms interval. Towards larger $\tau_1$, the failure probability decreases to the order of $1 \times 10^{-3}$ to $1 \times 10^{-2}$ per time interval. This shows that a major fraction of the first failures occurs shortly after the initialization of the link, while the failure probability decreases towards larger time scales, yet does not vanish completely. The cumulated probability for the link to fail within the first 60 s is approximately $p_{\mathrm{fail}} = (89.1 \pm 0.8)\,\%$.

This behavior is not exclusive to PHY 5, but is also exhibited by other links. Figure 3.2 shows the distribution of $\tau_1$ for PHY 3. The total probability for a fail to occur is slightly lower at approximately $p_{\mathrm{fail}} = (70.7 \pm 1.2)\,\%$, but the overall shape of the distribution is the same. Again, most of the errors occur shortly after the link initialization with approximately 40 % of the observed failures occurring within the first 333 ms.

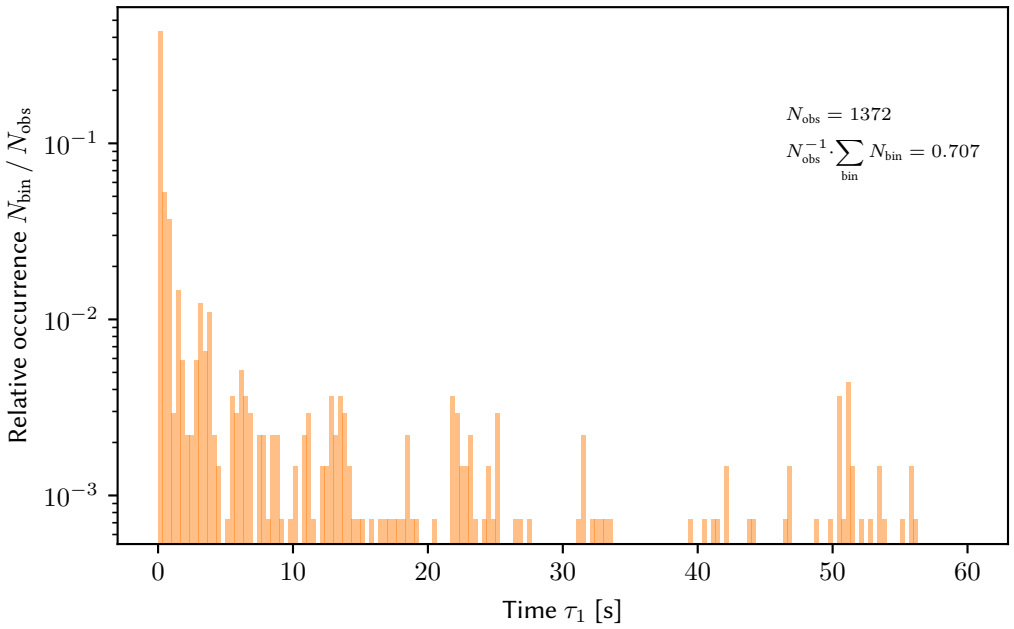



**Figure 3.2:** Relative number of link failures after first initialization for PHY 3. The histogram bin width corresponds to $\Delta\tau = 0.3\overline{3}\,\mathrm{s}$.

Curiously, while certain links like PHY 5 and 3 show a significant probability to fail in this specific setup, there are other links that hardly ever exhibit errors. An example for one of these links is PHY 6, whose distribution for $\tau_1$ is displayed in Figure 3.3. Over the course of 1372 repetitions of the `perftest`, PHY 6 only failed a total of three times, yielding a cumulated failure probability $p_{\mathrm{fail}} = (0.2 \pm 0.1)\,\%$ that is smaller by two orders of magnitude compared to PHY 5 or 3. Other PHYs exhibit even lower probabilities to fail at least once in the considered time window, with some links not even failing a single time. Table 3.3 summarizes the cumulated failure probabilities $p_{\mathrm{fail}}$ for all eight PHYs.

For this specific setup, PHY 3 and 5 seem to be the most problematic ones in terms of their first failure probability.

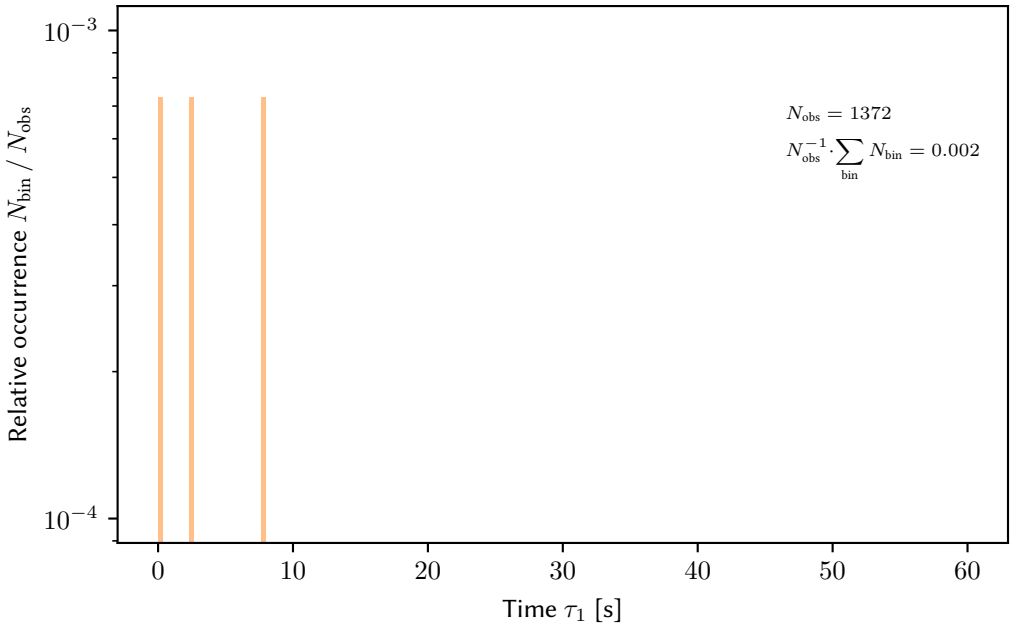



**Figure 3.3:** Relative number of link failures after first initialization for PHY 6. The histogram bin width corresponds to $\Delta\tau = 0.3\overline{3}$ s.

| PHY | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| $p_{\text{fail}}$ [%] | 0.3 | 0.0 | 0.0 | 70.7 | 0.0 | 89.1 | 0.2 | 0.1 |
| SEM [%] | ±0.1 | ±0.0 | ±0.0 | ±1.2 | ±0.0 | ±0.8 | ±0.1 | ±0.1 |
| $N_{\text{obs}}$ | 1372 | 1372 | 1372 | 1372 | 1372 | 1372 | 1372 | 1372 |

**Table 3.3:** Probability $p_{\text{fail}}$ for a first fail to occur for PHYs 0 to 7 during each of 1372 repetitions of a 60 s `perftest`.

In addition to the distributions of $\tau_1$, the times to the second link failure $\tau_2$ have been evaluated. For PHY 5, the distribution of $\tau_2$ is shown in Figure 3.4. The distribution exhibits a shape similar to the one of $\tau_1$ shown in Figure 3.1. Note that the distribution yet might be slightly distorted towards larger values of $\tau_2$ due to the total testing time of 60 s. The consecutive second errors for lately occurring first errors are not observed in the testing time window, so the tail of the distribution in Figure 3.4 may very likely be underestimated. Keeping this underestimation in mind, within the given time frame PHY 5 nonetheless had a chance of $p_{\text{fail}} = (67.9 \pm 1.3)$ % to fail a second time after being re-initialized due to a first error. Note that $p_{\text{fail}}$ is now normalized to the number of test runs in which a first error has been observed at all $N_{\text{obs}} = 1223$, as the normalization to the total number of 1372 test repetitions would underestimate the probability for a second failure. Again, with approximately 40 %, the major portion of second errors happens in the first 333 ms after re-initialization.

Table 3.4 summarizes the cumulated probabilities for a second link failure in the considered time window across the eight different links. The values for PHY 0 and 7 in Table 3.4 need to be taken with a grain of salt, as their probability to fail for a first time is small and thus the number of consecutive second failures is sparsely sampled. First failures on PHY 3 have been sampled comparably often however. Similar to PHY 5, its cumulated probability of $p_{\text{fail}} = (58.9 \pm 1.6)$ % to fail a second time is rather high as well. In other words, a link that has been re-trained due to a

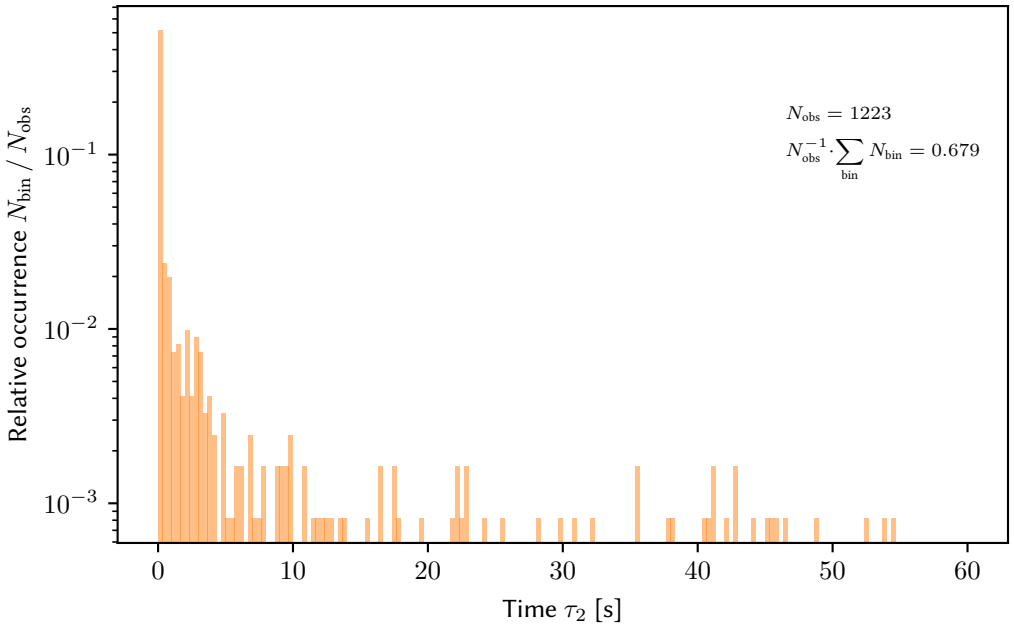



**Figure 3.4:** Relative number of second link failures after first re-training for PHY 5. The histogram bin width corresponds to $\Delta\tau = 0.3\overline{3}$ s.

| PHY | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| $p_{\text{fail}}$ [%] | 25.0 | 0.0 | 0.0 | 58.9 | 0.0 | 67.9 | 0.0 | 100.0 |
| SEM [%] | ±21.6 | ±0.0 | ±0.0 | ±1.6 | ±0.0 | ±1.3 | ±0.0 | ±0.0 |
| $N_{\text{obs}}$ | 4 | 0 | 0 | 970 | 0 | 1223 | 0 | 1 |

**Table 3.4:** Probability $p_{\text{fail}}$ for a second fail to occur for PHYs 0 to 7 during each of 1372 repetitions of a 60 s `perftest`.

first error being detected by the UT has a significantly high probability of failing for a second time consecutively.

The high-speed link issues that are visible in the presented test case can be summarized as follows:

1. Different PHYs on the same ASIC and Cube setup exhibit significant differences in their probability to fail during operation,

2. the failure probabilities across the different PHYs range from “failure highly unlikely” to “failure almost certain”,

3. most of the observed failures happen directly after link initialization and

4. PHYs that fail for a first time showed a $> 58\,\%$ chance to fail a second time consecutively, where enough data was obtained.

It is important to re-emphasize that the significance of the presented link error probabilities is limited to the first 60 s after link initialization. In an ideal setting, the probability for a link to

fail should be close to zero for arbitrary time spans. Long term effects are not investigated in the course of this study, yet as the distributions for $\tau_1$ and $\tau_2$ seem to be dominated by errors on small time scales, a focus on the link stability within the first milliseconds to seconds after the first link initialization seems reasonable.

## 3.2 Link Training

The early points in time of the link failures described in Section 3.1.4 suggest an issue with the initialization process of the PHYs. In combination with the high probability of a second failure to occur right after the re-training of the failing links, an issue with the link training algorithm responsible for setting the programmable delays of the de-serializers has been suspected as a possible source of error.

An additional compromise to the link stability could be posed by a known mismatch between the LVDS drivers on the ASIC and the receivers on the FPGA side: The full custom LVDS drivers on the ASIC side are capable of providing a current of up to 7 mA (Scholze et al. 2012). For the implementation within the HICANN-X, the drivers are fixed at an output current of $I = 6\,\mathrm{mA}$, which is considerably higher than the 3.5 mA that are typically used in applications with $R_{\mathrm{term}} = 100\,\Omega$ differential termination[8]. The high output current is suspected to lead to a large signal amplitude at the receiver, which may present an issue to the differential input buffers of the FPGA. For instance, for the Kintex-7 FPGAs that are employed as experiment control hosts in the Cube setups, a maximum differential input voltage of $V_{\mathrm{diff}}^{\mathrm{max}} = 600\,\mathrm{mV}$ for the LVDS capable I/O banks within the fabric (Xilinx, Inc. 2021a) is recommended. This is close to being exceeded by the theoretical value that would be delivered by the custom LVDS drivers:

$$V_{\mathrm{diff}} = R_{\mathrm{term}} \cdot I = 100\,\Omega \cdot 6\,\mathrm{mA} = 600\,\mathrm{mV}. \tag{3.8}$$

The assumption is that the potentially distorted signal that is received at the FPGA side of the link leads to an increased failure probability for the links in the ASIC-to-FPGA direction. In combination with a sub-optimal training procedure for the links, this would provide a reasonable explanation for the observed first and immediately following second errors on the links, as the first error may be triggered by the distorted signal at the receiver and the subsequent ones by the training algorithm failing to find a good delay setting again.

In this section, the currently employed link training algorithm will be analyzed. First, the logic behind the algorithm will be explained, before introducing the methods used to investigate its behavior in a running system. The search for an optimal delay value will be described in a working training example alongside real captured data. Subsequently, potential culprits in the training algorithm will be presented that lead to bad settings of the programmable delays, accompanied by adjustments to the training algorithm that improve the chance of finding a good delay state.

[8] Personal communication with Johannes Schemmel and Joscha Ilmberger.

### 3.2.1 Current Training Algorithm

In a source-synchronous data transmission scheme such as the one used for the high-speed links in the BSS-2 systems, simultaneously generated pulses on the clock and data line may pick up a finite skew relative to another before arriving at the receiver. This can be caused by – but is not limited to – mismatch of the copper trace lengths on the PCBs, mismatch of the internal clock routing within the ASIC or mismatch of the amplifier stages at the receiver. Thus, configurable delay elements in the de-serializer are needed to ensure a correct sampling of the data with respect to the clock edges. The amount of delay to be added is configured automatically by a link training algorithm.

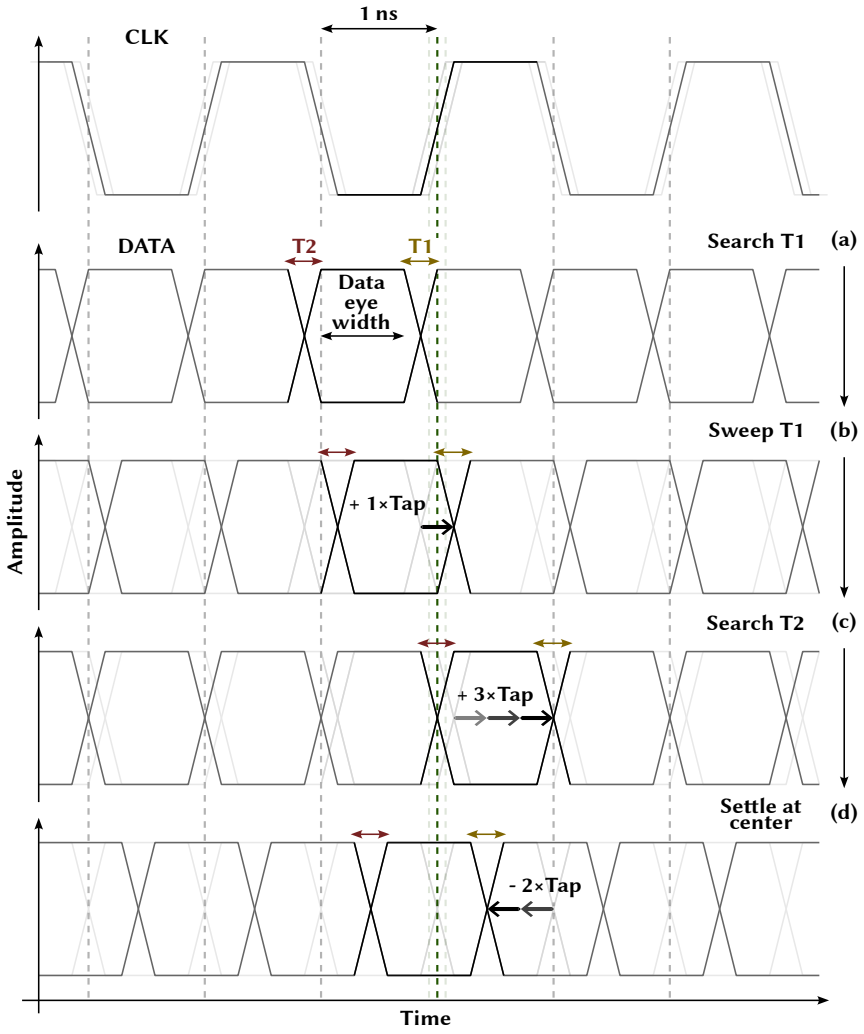



**Figure 3.5:** Schematic representation of the bit alignment process for the serialized data signal relative to the clock signal, adapted from Scholze et al. 2012. The sub-traces (a) to (d) represent the chronological progression of the alignment steps for the data signal at the de-serializer of the receiver. **(a)** The data signal is unaligned relative to the clock. The rising edge of the clock *(green)* arrives during the first transition region T1 of the data signal. **(b)** Delaying the data signal by one tap shifts the data signal such that the rising clock edge arrives at the end of T1. **(c)** Repeated increment of the delay shifts the rising edge of the clock into the second transition region T2. **(d)** The delay is decremented such that the clock edge meets the middle of the data eye approximately.

The algorithm that is used within the (de-)serializer macros of the BSS-2 systems has been introduced in Scholze et al. 2012. It is an adaption of an application note of Xilinx for Virtex-V FPGAs (Burton 2006). The basic functionality of the algorithm is split into two parts: bit alignment and word alignment.

### Bit Alignment

The first step of the algorithm can best be understood with the help of Figure 3.5. It displays a schematic adapted from Scholze et al. 2012 that outlines the bit alignment process for a serialized data stream relative to the clock signal.

In this schematic, the signals for clock and data are shown as observed at the de-serializer of the receiver in a source-synchronous DDR transmission scheme. The data signal is shown for multiple sub-steps (a) to (d) that are carried out sequentially in the bit alignment process. At the beginning, the data signal arriving slightly too early at the receiver. Therefore, the rising edge of the clock samples the data signal too close to the instable region in which the signal level transitions between zero and one, respectively. In a real system, the edges of a clock are subject to *jitter*, meaning that they do not arrive deterministically in fixed intervals but rather a bit too early in some periods or too late in others. The jitter results in the receiver observing oscillations even in a fixed bit pattern as sample-and-hold violations may impact the sampling process. As a countermeasure to compensate for the combination of signal skew and jitter, clock and data are re-aligned at the receiver. This is realized by shifting the signals relative to another such that the clock edges arrive primarily in time windows where the data can be considered as stable.

For the current training algorithm, this process can roughly be split into four parts that are depicted in Figure 3.5:

(a) The receiver sees a serialized bit stream at a frequency of 500 MHz, which is sampled at 1 GHz DDR and de-serialized into words of eight bit width[9]. The de-serialized byte word is checked for stability. Typically, this is done by comparing it to a fixed training pattern that is known to both, the transmitter and the receiver. The comparison is done at an eighth of the sampling speed, that is 125 MHz. In this schematic, the sampling happens close to the temporal transition region in which the data signal can change its level. As a result, the pattern seen by the receiver will be unstable and differ from the common reference training pattern. The positioning of clock edge relative to the data signal is said to lie within the *first transition*, denoted as T1.

(b) The receiver activates a delay element in the de-serializer fabric that delays the data signal relative to the clock edge by a fixed amount of time, a *tap*. The data signal is now sampled at the late end of the instable time interval – the algorithm has swept through T1 and stores the tap count value temporarily.

(c) Having exited T1, the receiver now sees a stable data pattern and continues to incrementally change the tap setting until it observes the pattern to become unstable again. It has now entered the next instable region of the temporal relationship between clock and data and essentially measured the width of the data eye. The rising clock edge is said to sample the data signal within the *second transition* T2.

[9] De-serialization not depicted in schematic.

(d) Having kept the tap setting of T1 stored and comparing it with the tap setting of T2, the receiver decrements the tap count to the mean of the two values. The data signal is now delayed such that the clock edge is as close as possible to the center of the data eye. The bit alignment process is complete.

#### Word Alignment

The bit alignment step ensures a stable sampling of the serialized data stream. However, it does not guarantee that the sampling starts at the first transmitted bit of each byte word. This is where an additional word alignment step needs to be performed.

To ensure that the ordering of the bit pattern is unambiguous, a training pattern is chosen that only allows for one valid ordering for cyclic permutation of all bits. In the present algorithm, this pattern is chosen to be $\texttt{0x2C} = \texttt{00101011}$. If the bit aligned byte word does not match the training pattern, a permutation of the bit pattern is performed, called a *bit slip*. The de-serializer macro presented in Scholze et al. 2012 implements the bit slip logic through fifteen flip flop registers, eight of which capture the data at the positive clock edge and seven at the negative. Given an eight bit pattern to be de-serialized, this register stage allows for eight different capture positions, effectively realizing all possible cyclic permutations of the pattern. When the receiver recognizes the training pattern `0x2C`, the training algorithm is terminated and the link considered as operational.

### 3.2.2 Bit Align Machine

The complete currently employed alignment logic for the HICANN-X is described in *hardware description language* (HDL) code in a *bit align machine* (BAM) module. The BAM implements an FSM to drive the incremental and decremental change of the delay and the word alignment in the de-serializer. In order to explain the functionality of the BAM, a block diagram of the FSM has been adapted from Burton 2006, which is displayed in Figure 3.6. It has slightly been changed to reflect the implementation of the BAM in the HICANN-X.

Again, the bit alignment process FSM depicted in Figure 3.6 can roughly be split into the four sub-steps (a) to (d) that are displayed in Figure 3.5. Sub-step (b) is additionally extended with a subroutine that attempts to prevent a false positive detection of T2 at sub-step (c). The idea is that the training algorithm shall cope with different shapes of jitter: *Random* jitter, which affects the data pattern seen within T1 to change rapidly – for example with each system clock cycle – and *deterministic* jitter, which renders the data pattern in T1 seen as stable, but incorrect due to constant bit flips in the pattern. Sub-step (e) corresponds to the final word alignment process.

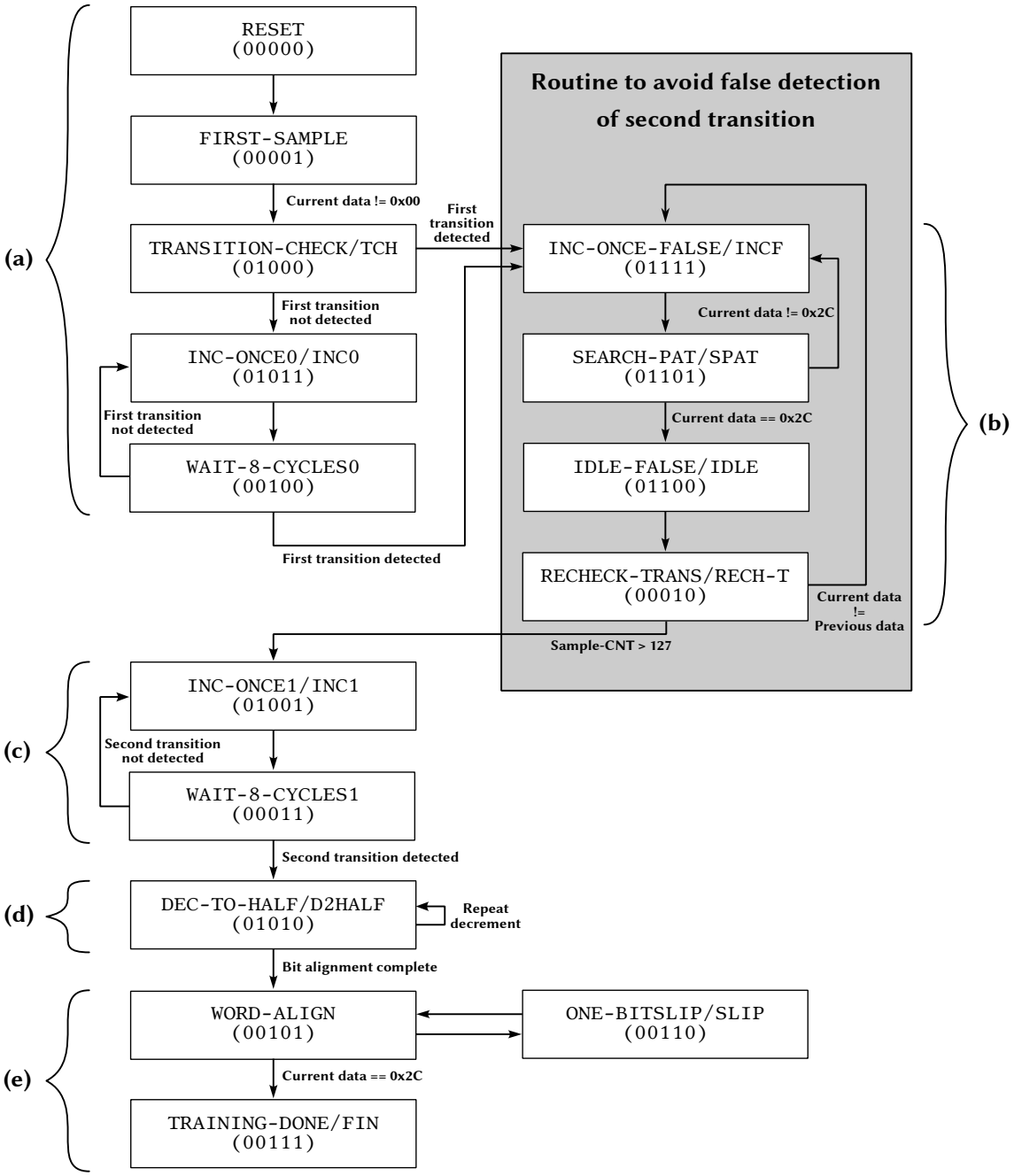



**Figure 3.6:** Block diagram of the FSM used in the BAM, adapted from Burton 2006. Changes have been made to the layout to fit the implementation in the custom (de-)serializer macros. The states of the FSM are denoted in long and abbreviated fashion.

The states within the FSM perform the following tasks:

(a) Search for start of T1:

- RESET: Reset internal signals of BAM to known state.
- FIRST-SAMPLE: Observe de-serialized byte words and check for deviation from 0x00

for eight consecutive clock cycles, which assumes that the transmission of the training pattern 0x2C by the transmitter has begun.

- TRANSITION-CHECK: After first non-zero sample has arrived, check the sample for stability. If it is unstable, move to sub-step (b), else increment the delay by a tap to move towards T1.

- INC-ONCE0: Pulse a wire connected to the delay elements for one system clock cycle to increment the tap count by one.

- WAIT-8-CYCLES0: Propagating the increment command to the delay elements takes a finite amount of time, so wait for eight system clock cycles if a change in the deserialized byte word takes effect. If it changes, move to sub-step (b), else continue with another increment of the tap count.

(b) Sweep through T1, while avoiding a false positive exit of T1 through auxiliary pattern checks:

- INC-ONCE-FALSE: With the start of T1 considered as detected, store the current tap count value and immediately increment the tap count by one to move further through T1. The stored tap count is considered to mark the first edge of the data eye.

- SEARCH-PAT: To avoid deterministic jitter to corrupt the data pattern in a stable manner, check if the byte word seen after the last delay increment still corresponds to the training signal 0x2C by performing repeated bit slips to cycle through all possible permutations of the observed data pattern. If it does correspond to 0x2C, move on to a stability check. If not, the sampling still takes place in T1, so the delay is incremented by another tap.

- IDLE-FALSE: An auxiliary state that is entered for one system clock cycle for stability reasons.

- RECHECK-TRANS: A timeout of 16 clock cycles first allows the bit slip operations of the SEARCH-PAT state to settle. The stability of the data pattern seen by the receiver is then checked for a total of 128 system clock cycles by comparing the sampled data with the pattern seen during the previous clock cycle. If they do not match, random jitter has been encountered and the sampling point is considered to still lie within T1. Else, T1 is considered to be exited, so move to sub-step (c).

(c) Search for start of T2:

- INC-ONCE1: With T1 considered as swept through, immediately increment the tap count by one to move through the data eye towards the start for T2.

- WAIT-8-CYCLES1: Again wait for the increment pulse to propagate to the delay elements and observe the data pattern for potential changes. If no change is detected within eight system clock cycles, repeat the incrementing of the delay. Else, the start of T2 is considered to be found, so move to sub-step (d). The tap count at this point marks the second edge of the data eye.

(d) Settle at the center of the data eye:

- DEC-TO-HALF: With the tap count settings known at both edges of the data eye, the mean of the two values is calculated and rounded down. The currently set tap count is decremented by its difference to the mean value. When the setting is finished, move to the word alignment process.

(e) Perform word alignment:

- WORD-ALIGN: Compare the observed data pattern with the training pattern `0x2C` for sixteen system clock cycles to allow the changes in the tap count and bit slip logic to take action on the received data pattern. If the patterns do not match, permute the data pattern. If they do match, consider the alignment process as complete.

- ONE-BITSLIP: Pulse a wire connected to the flip flop register stage for one system clock cycle to perform one bit slip on the received data pattern.

- TRAINING-DONE: Auxiliary state to inform connected modules that the alignment process on the current link is completed.

### 3.2.3 Method: In-Hardware Probing

In order to be able to check the performance of the BAM, an in-hardware probing approach has been chosen. This is mainly due to the difficulty of simulating the clock-to-data alignment as there is for instance no suitable model for the data eye available. Furthermore, this also allows to observe the behavior of the BAM under real operating conditions. To this end, it is required to observe the raw data seen by the receiver and the state of the FSM. Additionally, gathering information about the state of the delay tap count and the moments in time at which the BAM issues a bit slip command is desired.

Most hardware designs in modern FPGAs can be analyzed at runtime with some form of logic analyzer module. In the case of Xilinx FPGAs, devices can be programmed with a customizable *Integrated Logic Analyzer* (ILA), an IP core that can be used to monitor the internal signals and interfaces of a hardware design (Xilinx, Inc. 2021b). This allows to observe the internal state of certain modules in the design under real operation conditions at runtime that are otherwise only accessible in simulation. The signals that are connected to an ILA core are sampled at design speeds and stored using BRAM available in the FPGA fabric. The ILA cores can be programmed with boolean trigger equations such that certain events that are interesting to the user can be filtered. By buffering a user-specified amount of data, the captured signal traces provide information about the state of the system before the trigger condition was met, while the remaining memory used for capture is filled with trace data following the trigger.

The probing of signals with cores such as the ILA IP requires to specify the relevant signals before synthesis and connect them to the ILA core in the synthesized netlist. In the present mixed-vendor design flow[10], this is done by inserting debug `wire` signals to the design within the `synplify_wrapper` as needed and adding them to the port list of the module. Within the

[10] Compare Section 2.2.3.

top-level of the hardware design, the debug signals are then flagged with a special debugging attribute of the following form:

```
(* MARK_DEBUG = "true" *) wire my_debug_probe;
```

This implies a SYN_KEEP, which prevents the signal from being deleted through optimization steps during synthesis.

The instantiation of the ILA core and the connection to the relevant signals is then performed through XDC commands in the project generating *Tool command language* (Tcl) script:

```
...

# Synthesizing
launch_runs synth_1
wait_on_run synth_1
open_run synth_1

# Creating Debug Core
create_debug_core u_ila_0 ila
set_property C_DATA_DEPTH 8192 [get_debug_cores u_ila_0]
set_property C_TRIGIN_EN false [get_debug_cores u_ila_0]
set_property C_TRIGOUT_EN false [get_debug_cores u_ila_0]
set_property C_ADV_TRIGGER true [get_debug_cores u_ila_0]
set_property C_INPUT_PIPE_STAGES 2 [get_debug_cores u_ila_0]
set_property C_EN_STRG_QUAL true [get_debug_cores u_ila_0]
set_property ALL_PROBE_SAME_MU true [get_debug_cores u_ila_0]
set_property ALL_PROBE_SAME_MU_CNT 4 [get_debug_cores u_ila_0]

# Connecting Clock
set_property port_width 1 [get_debug_ports u_ila_0/clk]
connect_debug_port u_ila_0/clk [get_nets clk_sys]

# Connecting Probe
set_property port_width 1 [get_debug_ports u_ila_0/probe0]
set_property PROBE_TYPE DATA_AND_TRIGGER \
   [get_debug_ports u_ila_0/probe0]
connect_debug_port u_ila_0/probe0 [get_nets {my_debug_probe}]

# Connecting Additional Probes
create_debug_port u_ila_0 probe1
...

# Implementation
launch_runs impl_1
...
```

This example instantiates a debug core with a sampling depth of 8192 bit, two input pipe stages to meet timing constraints, and sets the number of comparators for each debug probe to four, allowing for complex trigger conditions. The system clock is connected to the clock port of the core, which defines the sampling rate of the signal traces. The signal to be debugged is connected to the default probe port of the core and configured to be available in defining the

trigger condition.

After programming the FPGA with the design containing the debug probe logic and capturing data subject to a given trigger condition, the signal traces have been read out via JTAG and analyzed in a *graphical user interface* (GUI). For availability of the signal traces in this thesis, the signal data has been exported to *value change dump* (VCD) format, converted to *JavaScript Object Notation* (JSON) files with the tool `vcd2wavedrom`[11] and post processed with `wavedrom-rs`[12], an interactive waveform editor.

### 3.2.4 Working Link Training Example

In the following, data from a working example captured with the ILA core will be used to describe how the training logic of the BAM within the FPGA is reflected under real operating conditions. The correct re-training procedure will be referred to as a baseline when checking the training algorithm for potential flaws. Unfortunately, it is not possible to probe the link training mechanism at the ASIC side of the communication channel as the fabric cannot be re-configured in the same way as a FPGA. Nevertheless, it is still possible to draw some conclusions about the state of the re-training at the ASIC side due to the mutual triggering of the re-training described in Section 3.1.1.

For a more complete overview over the interaction of the BAM with the link checking mechanism of the UT and the raw data seen by the receiver, ILA probes are inserted at multiple points within the FPGA design. Table 3.5 provides an overview over the captured signals that will be referred to throughout this section. The function of the probed signals in Table 3.5 can be summarized as follows:

TX-DATA Data transmitted to the ASIC.

RX-DATA Data received from the ASIC.

RX-VALID Indicates that the PHY receives valid data.

FSM-STATE Current state of the link training FSM.

TAP-COUNT Current number of delay taps activated.

BIT-SLIP Pulses bit pattern permutation at de-serializer.

ALIGNED Indicates that training algorithm has completed.

CRC-CNT Counts the number of failed CRCs performed by the UT.

START-LINK Active-low signal pulsed by the UT to request a re-training of a link.

[11] `https://github.com/Toroid-io/vcd2wavedrom`
[12] `https://github.com/coastalwhite/wavedrom-rs`
[13] Index `i` in range `0` to `7`, corresponds to PHY index.

| Signal | HDL module | HDL signal name | Width [bit] |
|---|---|---|---|
| TX-DATA | fpgaif_txrx_ddr_lvds_if | tx_data_link | 8 |
| RX-DATA | fpgaif_txrx_ddr_lvds_if | rx_data_link | 8 |
| RX-VALID | synplify_wrapper | phy_conn[i].rx_valid | 1 |
| FSM-STATE | bit_align_machine | current_state | 5 |
| TAP-COUNT | bit_align_machine | tap_count | 7 |
| BIT-SLIP | ddr_rx_channel | bit_slip_from_machine | 1 |
| ALIGNED | ddr_rx_channel | data_aligned | 1 |
| CRC-CNT | ut_sync | config_regs_in[4+i][4:0][13] | 5 |
| START-LINK | ut_sync | start_link_ut | 1 |

**Table 3.5:** Summary of the signals probed within the FPGA side of the high-speed links for investigation of the link training algorithm. The collection of signals corresponds to the data of a *single* link.

All data throughout this section corresponds to PHY 4, which has been captured on Cube system X0/W60F0 with the hardware components installed as summarized in Table 3.6. The PHY was identified as one of the links subject to frequent re-training in the given setup[14].

| Hardware component | ID |
|---|---|
| iBoard | 00 |
| xBoard | no ID available, silkscreen annotation: xboard_r0, 11/18 |
| Chip Carrier | 0x476FDB |

**Table 3.6:** Hardware components of Cube system X0/W60F0 used for probing of link training signals.

A low pulse of START-LINK issued by the UT for PHY 4 has been used as trigger condition for capturing the link training process. That is, the presented trace data corresponds to a *re*-training of the link after the UT observes a violation of one of its link stability criteria. Repeated `perftest` executions with a short runtime of 1 ms have been carried out to explicitly trigger on link failures shortly after link initialization. PHY 1 and PHY 4 have been enabled during the `perftest` in the training example shown below.

Figure 3.7 shows an excerpt of the first sub-step (a) of the bit alignment process of PHY 4. The timing diagram shows the current state for all signals listed in Table 3.5 at each sampling clock edge, grouped by their affiliation to the (macroscopic) modules PHY, BAM and UT within the FPGA design.

All signals have been captured at 125 MHz system clock frequency, that is, the temporal resolution of the diagram corresponds to 8 ns. Where applicable, long time intervals with no relevant events in the trace data have been cropped for visual clearance. These crops are indicated with a rolling strike through and annotated with the length of the time span that has been left out. Also important to note is that the data displayed on RX-DATA lags behind by approximately up to eight clock cycles relative to the signals grouped for the BAM. This is due to RX-DATA being probed behind an asynchronous FIFO that manages the clock domain crossing to the system clock frequency. The FIFO has a depth of eight, meaning that – depending on the internal state of the FIFO – the lag introduced can be up to eight clock cycles.

[14] For the data presented in Section 3.1.4, a slightly different configuration has been used that may have caused differences in the re-training probability for this PHY, cf. Table 3.1.

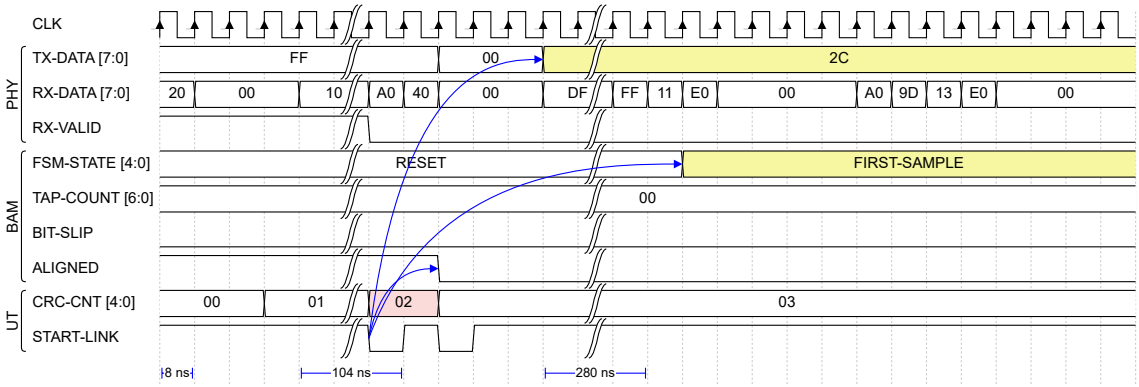



**Figure 3.7:** Timing diagram showing an excerpt of sub-step (a) of the bit alignment procedure for PHY 4. The re-training is triggered by a CRC error. The waveforms have been cropped at two points in time for visual clarity. All signals are grouped according to their module affiliation (PHY, UT or BAM). The signal values are given in hexadecimal representation, with the exception of FSM-STATE, whose data has been converted to state machine states for clarity. Note that the data shown on RX-DATA lags behind by approximately up to eight clock cycles compared to the signals of the BAM due to buffering inside an asynchronous FIFO. *Signal colors*: *Red*: CRC counter in the UT being increased after two consecutively failed checks. RX-VALID transitions from high to low, indicating that the data on PHY 4 is not valid. *Yellow*: The UT pulses START-LINK low, indicating that PHY 4 needs to be retrained. PHY 4 begins to transmit the training signal 0x2C to the ASIC. *Blue arrows* indicate causal relations between events. The UT pulsing START-LINK leads to the training signal to be transmitted, the data on PHY 4 being flagged as unaligned by the BAM and the training FSM collecting first samples after a timeout of 32 clock cycles.

The re-training shown in Figure 3.7 and in the following has been triggered by two consecutively failed CRC checks on the FPGA side of the link (*red*). The UT considers the link to be unstable and issues a re-training command for PHY 4 to the BAM by pulsing START-LINK low (*yellow*). This leads to the BAM flagging clock and data on the given link as unaligned and the FSM entering the initial sampling phase, in which it scans the byte words on RX-DATA for a divergence from 0x00 for eight consecutive clock cycles. In Figure 3.7 and the following timing diagrams, causal relations between events – such as the one between START-LINK and FSM-STATE for instance – are indicated by colorized arrows.

In parallel to awaiting the reception of the training signal, the PHY starts to transmit 0x2C to the ASIC as well. This will eventually force a re-training of the ASIC side of the link either, since the UT in the ASIC will fail to decode non-existing header information within the training signal. Even though both link partners will be re-trained eventually, the exact determination of the point of failure within the communication channel is indeed possible. In the example presented in Figure 3.7, it is the FPGA who starts to transmit the training pattern first, while the ASIC continues to transmit regular data. This means that the UT on the FPGA side spotted an error on the link before the UT within the ASIC.

After some time, the ASIC also begins to transmit the training signal, and the training FSM starts to search for the first transition. Figure 3.8 displays an excerpt of the corresponding sub-step (a) in the bit alignment process. As long as the pattern on RX-DATA remains unchanged (*yellow*), the BAM increases the delay setting in the de-serializer by a tap in the INC0 state (*magenta*). In this example, the first increase of the tap count already leads to a change of the received data pattern (*orange*). This indicates that the start of the first transition region has been detected and

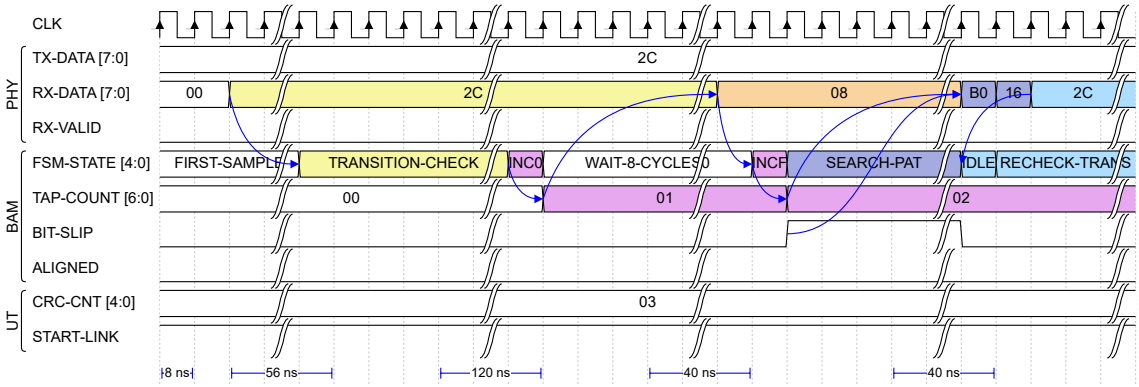



**Figure 3.8:** Timing diagram showing an excerpt of sub-steps (a) and (b) of the bit alignment procedure for PHY 4. The start of the first transition is detected and the training algorithm starts to sweep through it. The final rolling strike through denotes that the displayed trace data continues uninterrupted in Figure 3.9. *Signal colors*: *Yellow*: The ASIC has started to transmit the training signal 0x2C and the training FSM detects a pattern different from 0x00 on RX-DATA. A first stability check is performed for 16 clock cycles by entering the TRANSITION-CHECK state. *Magenta*: The training FSM sees no instability on RX-DATA and sends a command to increment the delay by a tap. *Orange*: After incrementing the delay, the bit pattern on RX-DATA changes. This is interpreted as the start of the first transition and marks the end of sub-step (a). *Dark blue*: The training FSM enters sub-step (b) of the bit alignment process. An auxiliary pattern check attempts to prevent a false positive detection of the second transition by searching for the training pattern 0x2C on RX-DATA. Keeping BIT-SLIP pulled high, the BAM induces a rapid permutation of the bit pattern on RX-DATA at every clock cycle. *Light blue*: The training pattern 0x2C is seen by the training FSM. The BAM continues to observe the stability of RX-DATA.

the training FSM moves to sub-step (b) of the bit alignment process.

In this sub-step, the training algorithm starts to sweep through the first transition. Consequently, the training FSM immediately increments the delay by a second tap in the INCF state. Note that INCF always stores the current tap count setting before incrementing the delay again. This is needed for the BAM to calculate the center of the data eye in a later step. The stored value corresponds to a tap count of $0x01 = 1$ in the example shown in Figure 3.8.

In an attempt to cope with the effects of deterministic jitter, the training FSM continues with an auxiliary pattern check to prevent a false positive detection of the second transition (*dark blue*). The BAM continuously asserts a bit slip command for PHY 4, resulting in a rapidly changing pattern on RX-DATA with every clock cycle. As soon as the training pattern 0x2C is observed, the training algorithm assumes that deterministic jitter does not impact the sampling. It moves to an additional stability check in RECHECK-TRANS to scan for random jitter (*light blue*). Note how the buffering of RX-DATA with respect to FSM-STATE is visible in this context: The IDLE state is entered slightly before 0x2C appears in the trace of RX-DATA.

Figure 3.9 picks off exactly at the point in time where the last rolling strike through in Figure 3.8 is located. The training FSM stays within the RECHECK-TRANS state for a total of 16 clock cycles before incrementing the delay value in the INCF state again (*magenta*). The additional increment is triggered by the oscillations of the pattern on RX-DATA between 0xC2 and 0xC0, which the training logic interprets as the impact of random jitter on the sampling process (*yellow*).

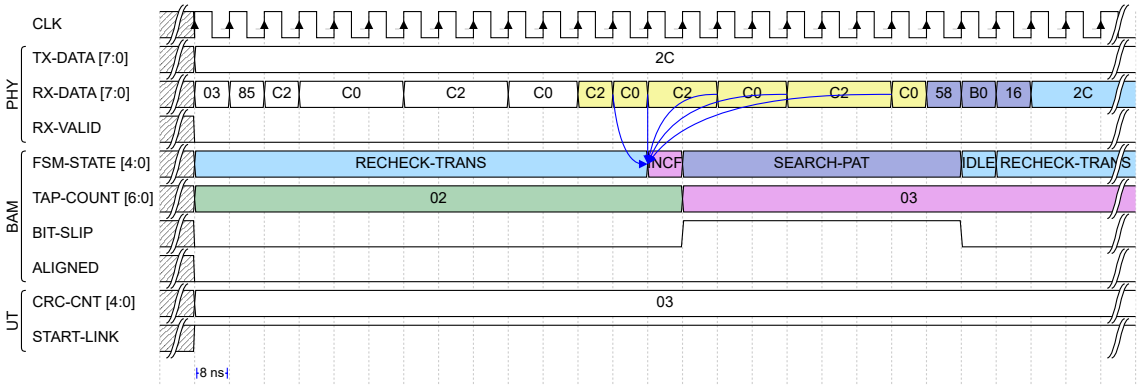



**Figure 3.9:** Timing diagram showing an excerpt of sub-step (b) in the bit alignment procedure for PHY 4. A delay increment sweeps through the first transition. The tap count value for the first edge of the data eye is updated and the observed pattern on RX-DATA is again checked against deterministic jitter. *Signal colors*: *Yellow*: The bit pattern on RX-DATA became unstable with the last delay increment in Figure 3.8, indicated by the oscillation between 0xC2 and 0xC0. Due to the buffered probing of RX-DATA, the exact point in time at which the stability check fails remains unclear. This is emphasized by multiple arrows pointing to the next FSM state. *Magenta*: The training FSM sends a pulse to increment the delay by one tap. *Green*: During the INCF state, the current TAP-COUNT value of 0x02 is stored, marking the new first edge of the data eye. *Dark blue*: The training FSM starts a pattern check to prevent a false positive detection of the second transition. The training pattern 0x2C is searched on RX-DATA by keeping BIT-SLIP pulled high, which leads to a rapid permutation of the bit pattern on every clock cycle. *Light blue*: The training pattern 0x2C is seen by the training FSM. It continues to check for random jitter by scanning the stability of RX-DATA for 128 clock cycles. *Grey dashed with rolling strike through*: Indication that first signal values are corresponding to their final states Figure 3.8.

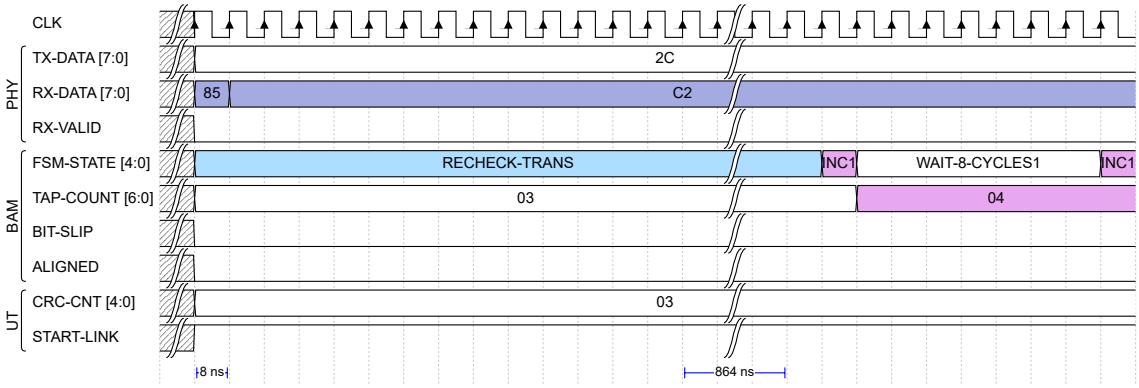



**Figure 3.10:** Timing diagram showing an excerpt of sub-step (b) in the bit alignment procedure for PHY 4. The passed stability check against random jitter indicates end of first transition. The bit pattern on RX-DATA has not changed during 128 clock cycles, so the transition is considered to be exited. The previously stored TAP-COUNT value of $0x02 = 2$ represents the first edge of the data eye. *Signal colors*: *Light blue*: The training FSM checks for random jitter by scanning the stability of RX-DATA for 128 clock cycles. *Magenta*: The training FSM sends a pulse to increment the delay by one tap after the passed stability test. *Dark blue*: The bit slip operation slightly overshoots with respect to the data displayed in Figure 3.9. *Grey dashed with rolling strike through*: Indication that first signal values are corresponding to their final states Figure 3.9.

Note that there are already pattern changes visible on RX-DATA before this instability, which do not oscillate between 0xC2 and 0xC0. This is due to a slight overshoot of the bit slip operation performed during the SEARCH-PAT step. The bit slip command only settles with a finite delay relative to the transition of FSM-STATE into the RECHECK-TRANS state. This is also the reason why the comparison of old samples against current ones on RX-DATA requires a timeout, which is set to 16 clock cycles in the current training algorithm.

The further delayed data pattern is again checked for deterministic jitter by permuting through all possible bit orders in search for the training pattern. It is observed in the final clock cycles of Figure 3.9. The trace data continues uninterrupted in Figure 3.10. Again, the constantly asserted bit slip command leads to a slight overshoot of the permutations on RX-DATA, which settles in a bit arrangement corresponding to 0xC2 (*dark blue*).

With the additional tap added to the delay chain, the BAM has moved the sampling point further through the transition, but this time the stability check is passed. The previously stored tap count setting of $0x02 = 2$ is considered as marking the end of the first transition. Hence, the training FSM moves to sub-step (c) of the bit alignment process. In Figure 3.10, this is reflected by an increment of the delay value in the INC1 state. The training FSM now searches for the start of the second transition by repeatedly incrementing the delay value and waiting for eight clock cycles to observe a change in the pattern on RX-DATA.

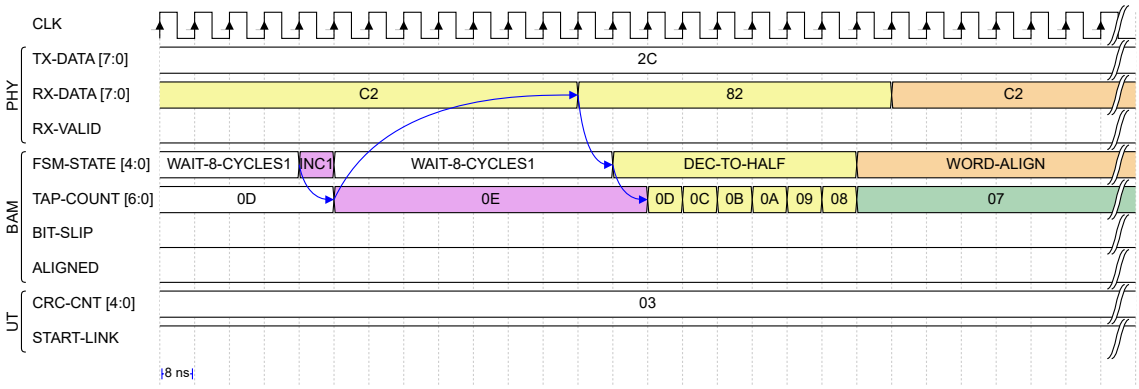



**Figure 3.11:** Timing diagram showing sub-steps (c) and (d) of the bit alignment process for PHY 4, in which the second transition is found and the clock edge aligned with the middle of the data eye. *Signal colors*: *Magenta*: BAM increments the delay value by one tap. *Yellow*: Bit pattern on RX-DATA changes in response to an increment of the delay value. The BAM interprets this as the start of the second transition – which marks the second edge of the data eye – and decrements the delay value. *Green*: The final tap count of $0x07 = 7$ corresponds to the middle of the data eye. *Orange*: BAM compares the bit pattern on RX-DATA with the training signal 0x2C.

The point in time at which the second transition is found is displayed in Figure 3.11, which shows the delay increment (*magenta*) that finally renders the data pattern unstable again (*yellow*). The current tap count setting of $0x0E = 14$ now corresponds to the second edge of the data eye. This marks the end of sub-step (c) and the training FSM moves to sub-step (d) of the bit alignment process. During this step, the FSM enters the DEC-TO-HALF state and decrements the delay value to the mean of the stored tap count settings. In this case, the final value should correspond to $14 - \lfloor (14 - 2) \div 2 \rfloor = 8$. However, the DEC-TO-HALF state has a slight quirk to overshoot the decrement by one tap and settles at a final value of $0x07 = 7$ (*green*). When the optimized delay

value is set, the training FSM moves to the word alignment sub-step (e) (*orange*). The BAM first waits for the tap count decrement to settle. Then, it checks RX-DATA again for the training pattern 0x2C.

Figure 3.12 shows how repeated single clock cycle bit slips are applied to perform precise permutations of the sampled data without overshooting (*dark blue*). Eventually, the training pattern is found on RX-DATA. With the delay value optimized and the byte word aligned, the training FSM enters its final state FIN (*green*). Clock and data on the trained link are flagged as aligned by asserting the ALIGNED signal, and the PHY follows shortly by asserting RX-VALID as well. The training of the link is complete.

During the `perftest` run that corresponds to the presented trace data, PHY 4 did not fail for a second time. Furthermore, the final delay tap count setting of $0x07 = 7$ was verified to be set in multiple `perftest` repetitions in which PHY 4 did not fail at all. This emphasizes that the shown re-training example is indeed representative for a good training of PHY 4.

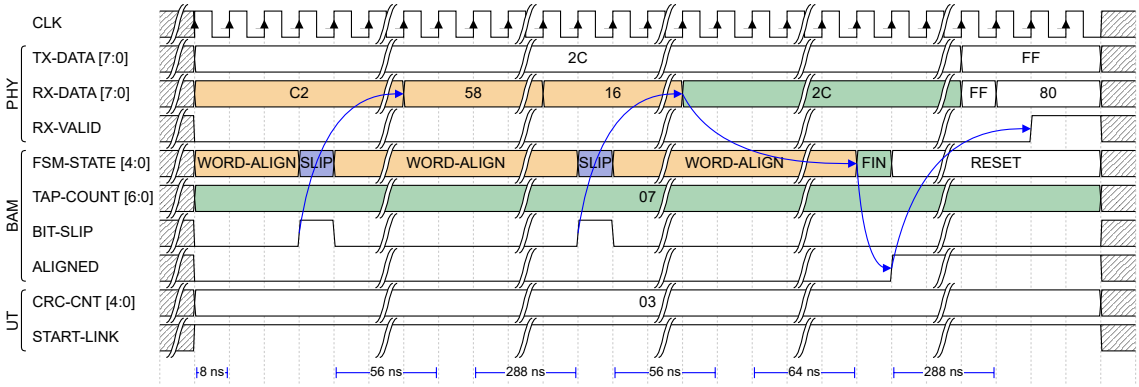



**Figure 3.12:** Timing diagram for sub-step (e) of the training FSM in a retraining of PHY 4. The word alignment step of the algorithm permutes the bit pattern on RX-DATA until it matches the training pattern 0x2C. *Signal colors*: *Orange*: BAM compares the bit pattern on RX-DATA with the training signal 0x2C. *Dark blue*: BAM pulses BIT-SLIP for one clock cycle to permute the bit pattern on RX-DATA. *Green*: The observation of the training pattern 0x2C on RX-DATA serves as the terminating condition of the training algorithm. PHY 4 is flagged as aligned with a final tap count of $0x07 = 7$ and RX-DATA is marked as valid. *Grey dashed with rolling strike through*: Indication that first signal values are corresponding to their final states Figure 3.11.

## 3.3 Training Errors and Corrections

The performance of the link training algorithm is dictated by whether or not it is capable of finding a good delay setting such that it hits the center of the data eye reliably. A prerequisite for this is that the temporal resolution of the delay added between clock and data is fine enough to detect the edges of the data eye. For the ISERDESE2 primitives employed as de-serializers on the FPGA side of the link, the delay is set in a *programmable delay primitive* (IDELAYE2) that can be connected to the ISERDESE2 (Xilinx, Inc. 2018). According to Xilinx, Inc. 2021a, the delay resolution $T_{\text{IDELAYRESOLUTION}}$ of the IDELAYE2 is calculated via

$$T_{\text{IDELAYRESOLUTION}} = (32 \cdot 2 \cdot F_{\text{REF}})^{-1} . \tag{3.9}$$

In this equation, $F_{\text{REF}}$ is the frequency of a reference clock that needs to be provided to a *tap delay value control block* (IDELAYCTRL) to derive a *process, voltage, and temperature* (PVT) independent voltage bias for precise tap values (Xilinx, Inc. 2021c). In the case of the Kintex-7 FPGAs used within the Cube systems, this reference frequency corresponds to $F_{\text{REF}} = 200\,\text{MHz}$. Hence, the delay resolution for the FPGA side of the Cube systems is

$$T_{\text{IDELAYRESOLUTION}} \approx 0.078\,\text{ns}, \tag{3.10}$$

which corresponds to 7.8 % of the unit interval at 1 Gbit/s data rate in 500 MHz DDR data transmission. In a worst case scenario, the training initially needs to sweep almost the entire data eye to find the first transition region before sweeping through the data eye again in search of the second transition. That is, a maximum of 2 ns / 0.078 ns $\approx$ 26 taps are needed to detect the center of the data eye. This corresponds to 26 / 32 $\approx$ 81 % of the available tap resources for the IDELAYE2 (Xilinx, Inc. 2018). Hence, the tap settings match the time span to be swept rather well. Therefore, a slight mismatch of the final tap value by one tap – such as the one observed in Figure 3.11 for example – is not expected to degrade the link stability greatly. This is, of course, given that the actual width of the data eye leaves enough headroom for such inaccuracies, which may not always be the case.

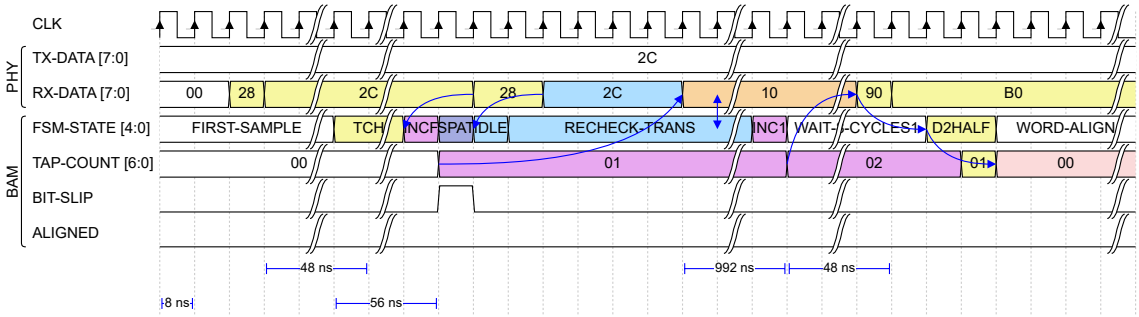
Keeping these considerations in mind, a stark impact on the link stability is expected when the final delay value deviates by approximately half a unit interval from the optimal value. This would correspond to a 90° phase shift of the clock edge into the instable region of the data signal. Considering the tap resolution from equation 3.10, such effects should be visible if the delay differs by approximately 0.5 ns / 0.078 ns $\approx$ 6 taps from the optimal value.

Indeed several examples for large deviations of the final tap count value have been recorded for the PHY shown in Section 3.2.4. Having established a baseline for a good re-training procedure of PHY 4, it is possible to trigger on delay values that do not match the final state of the algorithm.

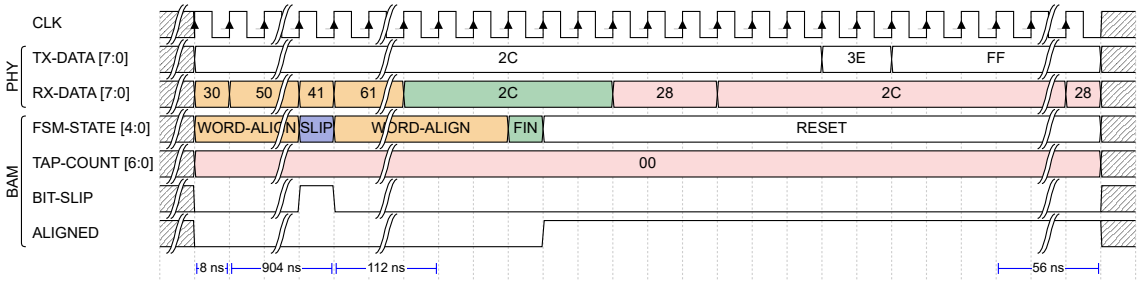
This section presents potential culprits of the link training procedure that lead to a large mismatch between the optimal value of 7 taps and the final state after a re-training of the link. For each observed error in the training procedure, a corresponding fix will be presented that should lead to an overall improved robustness of the link initialization.

### 3.3.1 Failing Deterministic Jitter Check

The most important training error that could be observed concerns sub-step (b) of the bit alignment process in the training FSM in Figure 3.6. Figure 3.13 displays trace data for PHY 4, in which the training terminates with a final tap count value of 0.




**(a)** Excerpt of the training algorithm between start of the training and the detection of the second transition. The BAM misinterprets the end of the first transition as the start of the second transition and sets the delay tap value too low. *Signal colors*: *Orange*: Deterministic jitter leads to a corrupted bit pattern $0x10 = 00010000$ on RX-DATA, which is no valid permutation of the training signal $0x2C = 00101100$. However, the BAM considers it as stable for 128 clock cycles during the check for random jitter.




**(b)** Excerpt of the pattern alignment and termination phase of the training algorithm. The BAM observes the training pattern on RX-DATA and stops the alignment procedure, but the pattern actually remains unstable. *Signal colors*: *Orange*: BAM cycles through permutations of the bit pattern on RX-DATA.

**Figure 3.13:** Timing diagram for a suboptimal training of PHY 4 stuck in first transition. The training FSM settles in a broken tap count setting due to a failed check for deterministic jitter. *Signal colors*: *Orange*: Meaning is contextual to sub figures (a) and (b). *Yellow*: Bit pattern on RX-DATA oscillates, which the BAM interprets as the start of the first and the second transition of the data eye, respectively. *Magenta*: BAM increments the delay by one tap. *Dark blue*: BAM applies a bit slip to permute the bit pattern on RX-DATA. *Light blue*: BAM enters the pattern check to check for deterministic jitter. *Green*: Training pattern on RX-DATA serves as the terminating condition of the algorithm and the PHY is flagged as trained. *Red*: Tap count of delay locked to a bad state, leading to oscillations of the bit pattern on RX-DATA even after termination of the algorithm. *Grey dashed with rolling strike through*: Indication that first signal values in sub-figure (b) are corresponding to their final states in sub-figure (a).

The training procedure displayed in Figure 3.13a starts slightly different than in the working example, as the pattern is already observed to be unstable before applying a first increment of the delay (*yellow*). The training FSM immediately jumps to sub-step (b), which adds a tap to the delay chain in the INCF state and stores the setting $0x00 = 0$ as marking the edge of the data eye temporarily (*magenta*). It takes a few clock cycles for the additional delay to take effect. The tap count increment is immediately followed by the pattern check, in which RX-DATA is compared to the training signal $0x2C$ to check for deterministic jitter (*dark blue*). By coincidence, the training pattern can already be observed at the start of the re-training without any delay adjustments (*light blue*). This is only possible as the delay increment from INCF has not yet taken effect on the sampling process. As a consequence, the pattern check is instantly passed and the state RECHECK-TRANS performing the stability check for random jitter is entered. Due to a timeout of 16 clock cycles, the stability check only reacts to changes on RX-DATA after a finite amount of

time. This allows the delay issued within the INCF state to finally take effect. The data pattern changes to 0x10, which is no valid permutation of the training signal 0x2C (*orange*). Nevertheless, as the stability check only tests for random jitter by comparing old data samples against new ones without explicitly checking, if it is actually the training pattern that is being observed, the corrupted data pattern passes the check. The training FSM falsely assumes to have exited the first transition region, while it is actually still sampling within it.

The training FSM now enters sub-step (c) and begins to scan for the second transition. In reality, it is the following delay increment in the INC1 state that exits the first transition (*magenta*). However, the BAM interprets the resulting pattern change as the second edge of the data eye, decrementing the delay back to the mean of the current tap count and the stored value for the first edge $2 - \lfloor (2 - 0) \div 2 \rfloor = 1$. Together with the overshoot of the decrement in the D2HALF step by one tap, the final tap count settles back at zero (*red*). Since this setting occasionally allows to observe the correct training pattern, also the word alignment procedure displayed in Figure 3.13b succeeds (*orange*). The data on PHY 4 is flagged as aligned and the training is terminated (*green*), although the pattern on RX-DATA shows oscillations between 0x28 and 0x2C shortly after.

The final tap count value roughly corresponds to a 90° shift of the clock edge with respect to the optimal setting, and indeed many of the link failures due to CRC errors during the signal probing process correlate with a tap count of zero.

### Training Correction

While initially the main difference between the working training example discussed in Section 3.2.4 and the present failure example is the tap count value at which the training FSM enters sub-step (b), it should not be the ultimate reason of failure. As outlined before, the training routine should ideally be robust enough to compensate for a variation in the delay setting by one or two taps. However, the detection of the first data eye edge at a delay setting that is smaller by one tap relative to the working example – compare Figure 3.8 and 3.13a – ultimately leads to a mismatch in the final setting by seven taps, which should not be the case.

Essentially, the current training algorithm is incomplete in the sense that it cannot fully account for the influence of deterministic jitter and that it does not make any prior assumptions about the width of the data eye. This incompleteness can be summarized in two rules that can be derived from the observations made in Figure 3.13a:

1. The pattern check against deterministic jitter should never be exited with a data pattern that is no valid permutation of the training signal.

2. The beginning of the second transition should never be considered as found, if the stability check against random jitter in the first transition has *just recently* passed. That is, the algorithm should be agnostic to the resolution of the delay elements in some sense. For any reasonable proportion of the data eye width compared to the delay resolution, the first and the second edge of the data eye should never be considered to be located just one tap count apart from another. Otherwise, the algorithm would never be able to decrement to the mean value of the two settings.

In order to implement these rules, the logic of sub-steps (b) and (c) in the training FSM has been

changed to increase the stability of the check for deterministic jitter. Figure 3.14 provides a schematic of the updated excerpt in the BAM FSM.

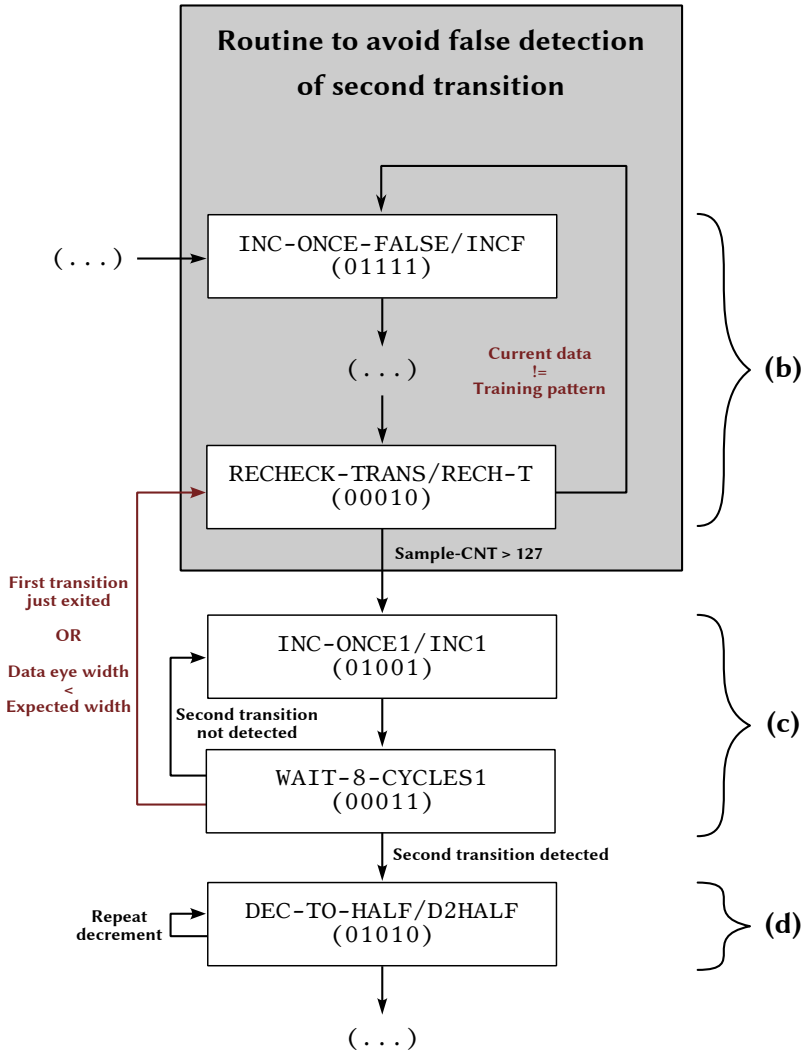



**Figure 3.14:** Schematic of the updated deterministic jitter check in the BAM FSM. Changes made to the training routine are highlighted in *red*.

In particular, the following adjustments have been made:

1. During the stability check state against random jitter, RECHECK-TRANS, the failure condition does not correspond to new samples matching old samples anymore. Instead, the observed data pattern is explicitly checked to match the training pattern 0x2C. This acts as a fail-safe condition in case the pattern check does not recognize the existence of deterministic jitter in time. At the same time, the previous stability condition is implied as any divergence in the received data word from the expected pattern leads back to another increment of the delay.

2. A *sticky bit* is introduced that reflects whether or not the algorithm considers the first transition to have been exited *just recently*. That is, whenever the RECHECK-TRANS state is exited towards the INC-ONCE1 state, the sticky bit is set. During the WAIT-8-CYCLES1 state in which the start of the second transition is checked, the value of the sticky bit is read out.

If it is set, the FSM knows that the first transition has been exited *just one tap ago*.

3. Finally, a parameter has been introduced that specifies a heuristic minimum for the unit interval. If the width measured by the algorithm is smaller than the specified value, any pattern instability during the WAIT-8-CYCLES1 state is considered to not correspond to the second edge of the data eye but rather as an ongoing sampling in the first transition region.

Additionally, within the SEARCH-PAT state, the bit slip operations are not rapidly applied by keeping BIT-SLIP pulled high anymore. Instead, bit slips are applied in a controlled fashion by pulsing the signal once every sixteen clock cycles. This allows for the operation to settle and be represented in the received data pattern during the SEARCH-PAT check. This is necessary as only in this way the SEARCH-PAT state is guaranteed to be exited with a bit slip setting that permutes the bit ordering in the received data to in principle correspond to the training pattern. Otherwise, the stability check against random jitter would not be able to check for the existence of the training pattern and fail instantly. This adaption of the algorithm is not depicted in Figure 3.14. Note that this change towards an increased stability of the pattern check is at the expense of the speed of the training algorithm. In the worst case word alignment, the bit slip would have to be applied seven consecutive times, introducing an overhead of approximately $7 \cdot 16 = 112$ clock cycles compared to the rapid sweep of the pattern. However, as the bit slip setting remains in a state that guarantees the correct word alignment – otherwise, the consecutive stability check against random jitter would fail now – the final word alignment process in sub-step (e) of the algorithm is expected to be exited more quickly.

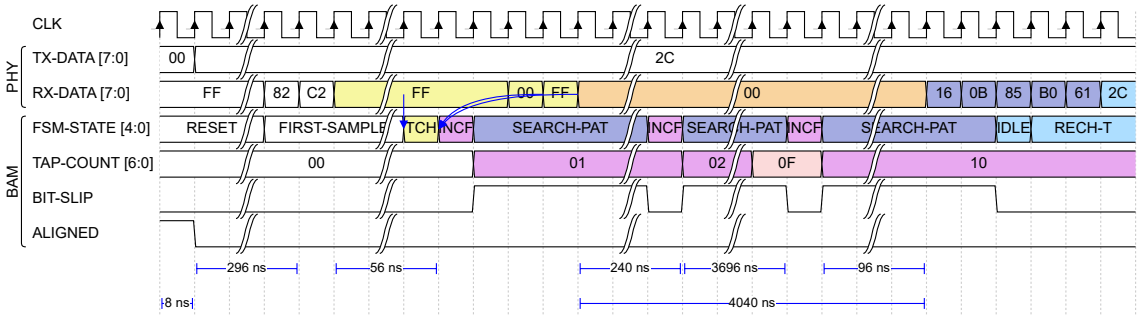
The changes introduced to the training algorithm are encapsulated in a *git* commit in the `hicann-dls-private` repository[15].

### 3.3.2 Premature Re-training

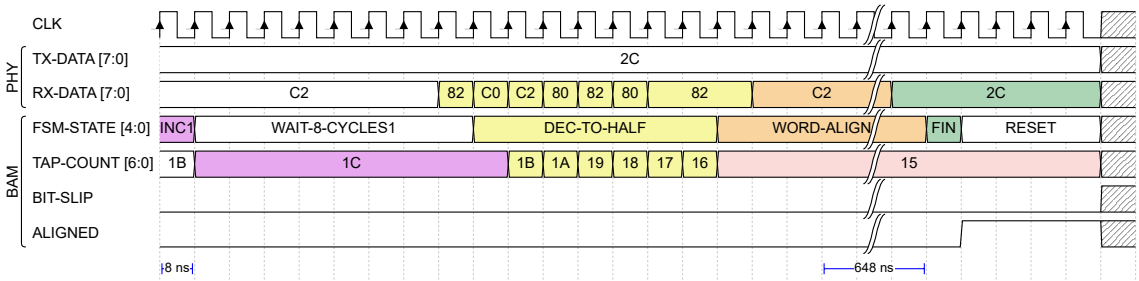
The second issue within the link training routine that has been detected concerns sub-steps (a) and (b) of the bit alignment procedure, in which the start of the first transition is searched and the transition region swept. Consider the excerpt of trace data recorded for PHY 4 displayed in Figure 3.15. In this example, the training terminates with a final tap count value of `0x15` $= 21$.

Figure 3.15a shows an excerpt of the re-training during sub-step (a) of the training FSM. The trace data shows how FSM-STATE transitions from the RESET state to FIRST-SAMPLE. Simultaneously, the PHY starts to transmit the training pattern `0x2C` to the ASIC. Meanwhile, the data received by the ASIC on RX-DATA corresponds to regular event data as the UT link checking mechanism in the ASIC only fails with a finite delay in response to the reception of the training pattern (*yellow*). After a timeout for the RESET state is reached, the training FSM moves on to check the received data stream for a divergence from `0x00`. By observing the *old* data that is received from before the re-training has been triggered on the ASIC side of the link, the training FSM concludes that the transmission of the training pattern by the ASIC has begun. Consequently, it enters the TRANSITION-CHECK/TCH state.

[15] URL: `https://gerrit.bioai.eu:9443/c/hicann-dls-private/+/22982`;
Commit hash at state of writing this thesis: `51fa2ea9564aebd3dd8b443cbddbd99adaa55190`.




**(a)** Excerpt of the initialization phase and detection of the start of the first transition during sub-step (a) of the training FSM. *Signal colors*: *Yellow*: Old data from the ASIC not being flushed in time triggers the TRANSITION-CHECK state of the BAM. Additionally, pattern changes of the received data are confused with proper instabilities of the data signal, ultimately leading to an overshoot of the tap value at the start of the first transition. *Orange*: While the BAM has already started to search for the first transition, the ASIC repeatedly sends zeros for more than 4000 ns until the training pattern is transmitted for the first time.




**(b)** Excerpt of sub-steps (c), (d) and (e) of the training FSM with the detection of the second transition and the termination of the algorithm. The final delay tap count after marking the data eye sweep as finished is set to $0x15 = 21$, which is 14 taps larger than in a successful training of PHY 4 (cf. Figure 3.12). *Signal colors*: *Yellow*: BAM senses a pattern instability and decrements the tap count to the mean of the stored values $0x0F = 15$ and $0x1C = 28$. *Orange*: BAM compares bit pattern on RX-DATA with the training pattern 0x2C.

**Figure 3.15:** Timing diagram for a suboptimal training of PHY 4 sampling old data. The training FSM settles at a tap count corresponding to a 180° phase shift with respect to the optimal value due to an early triggering of the TRANSITION-CHECK state by old data from the ASIC. *Signal colors*: *Yellow* and *orange*: Meaning is contextual to sub figures (a) and (b). *Magenta*: BAM increments the delay value by one tap. *Dark blue*: BAM applies bit slips to permute the bit pattern on RX-DATA in search of the training pattern 0x2C to account for deterministic jitter. *Light blue*: BAM enters the pattern stability check to check for random jitter. *Red*: Tap count values for the first edge of the data eye and the center of the data eye have overshot due to confusion of old data with the start of the first transition. *Green*: BAM observes the training pattern and terminates the training algorithm.

Approximately 400 ns after the FPGA started the transmission of the training pattern, the ASIC terminates the transmission of regular data words and sends a continuous stream of zeros for roughly 4000 ns (*orange*). The training FSM interprets the resulting pattern change as the start of the first transition and enters sub-step (b) of the bit alignment procedure. The training logic performs the pattern check against deterministic jitter (*dark blue*) and repeatedly increments the delay value (*magenta*), as it is not able to detect the training pattern 0x2C. By the time the ASIC actually transmits the training signal, the tap count that marks the first edge of the data eye has already increased to $0x0F = 15$ (*red*), which is 13 taps larger than the corresponding value in a working retraining (cf. Figure 3.9).

The further re-training procedure in sub-steps (c), (d) and (e) of the training FSM is shown in Figure 3.15b. With the training pattern now being transmitted by the ASIC as expected, the algorithm continues just like in a working example. The only difference is that the final tap count for the data eye center after detecting the second transition is now set to $\texttt{0x15} = 21$ (*red*). Compared to the working link training example shown in Figure 3.12, this corresponds to an increase of 14 taps or, by translating the tap count to a unit of time using the delay resolution from equation (3.10), to an additional delay of

$$14 \cdot T_{\text{IDELAYRESOLUTION}} \approx 1.09\,\text{ns}. \tag{3.11}$$

With respect to the unit interval of 1 ns, the delay difference essentially boils down to a 180° phase shift from sampling the data at the rising clock edge to sampling it at the falling one. Although this shift can in principle also represent a working state – since the clock edges should still be mostly aligned with the data eye centers – the additionally enabled delay elements may lead to an increase in jitter, which should be avoided. During the investigation of the BAM probe traces, frequent re-trainings of PHY 4 have been observed after the UT on the FPGA side reported consecutive CRC errors with the delay value set to 21 taps.

#### Training Correction

The bit alignment procedure in the BAM is triggered by an external control logic block, which in turn is accessed whenever the UT triggers the retraining due to a failed link health check. The control logic specifies a timeout value before a command to start the training procedure is issued to the BAM. By changing this value, the time until the training FSM starts to expect the first training pattern samples from the ASIC can be increased. Indeed, when doubling the timeout setting from 32 to 64 system clock cycles, no further tap count settings of 21 could be observed in the re-training of PHY 4. More generally, the timeout has been made adjustable by implementing it as a design parameter to be able to tune it for various systems.

The change introduced to the BAM control logic is accessible in a *git* commit in the `hicann-dls-private` repository[16].

### 3.3.3 Missing Bad State Timeout

A more critical issue with the current implementation of the training FSM is displayed in Figure 3.6. It concerns the absence of a global timeout that terminates the algorithm in case that no good combination of delay value and bit slip permutation can be found. Figure 3.16 shows an example of the resulting erroneous re-training for PHY 4.

Due to a previous error – likely stemming from an erroneous passing of the check against deterministic jitter described in Section 3.3.1 – the final delay setting in the re-training example settles at a tap count of $\texttt{0x01} = 1$ (*red*). Similar to before, this corresponds to a phase shift of roughly 90° in the timing relationship between clock and data into the transition regions.

[16] URL: `https://gerrit.bioai.eu:9443/c/hicann-dls-private/+/22982`;
Commit hash at state of writing this thesis: `51fa2ea9564aebd3dd8b443cbddbd99adaa55190`.

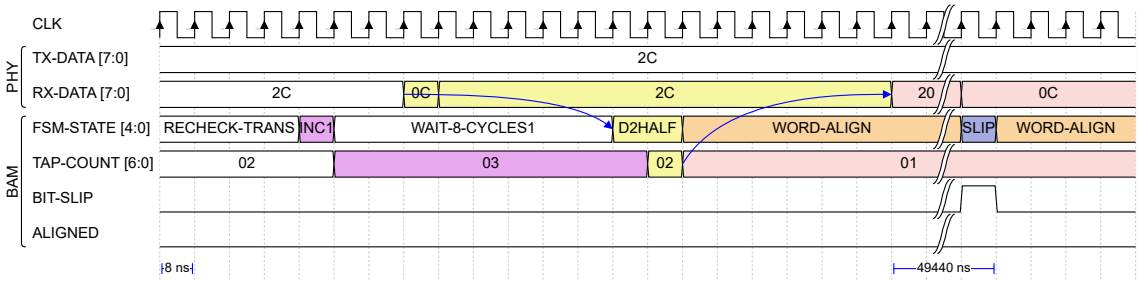



**Figure 3.16:** Timing diagram of sub-steps (c), (d) and (e) in the training algorithm with PHY 4 being stuck in the final word alignment step. The absence of a global timeout in the training FSM leads to the BAM not observing the training pattern during sub-step (e) of the re-training for more than 49 µs. *Signal colors*: *Magenta*: An increment of the delay value renders the pattern on RX-DATA unstable. *Yellow*: The instability of the received data pattern in response to the delay increment is interpreted as the detection of the second edge of the data eye. The training FSM enters the D2HALF step, in which it decrements the delay to the center of the data eye. *Red*: Due to earlier errors in the alignment procedure, the final delay value is set to $0x01 = 1$ taps. The observed data pattern on RX-DATA is corrupted by a combination of random and deterministic jitter alike. *Orange*: The training FSM searches for the training pattern 0x2C in the final word alignment sub-step (e). *Dark blue*: BAM issues a bit slip command to the de-serializer.

Consequently, the observed pattern on RX-DATA is distorted as jitter dominates the sampling process. When the training FSM permutes the data pattern in the WORD-ALIGN state (*orange*) by applying repeated bit slips (*dark blue*), the bad alignment between clock edge and data prevents the training pattern 0x2C to be observed for more than 49 µs. For comparison, the typical working training procedure for a single link takes about 8 µs. Essentially, the training FSM is stuck in sub-step (e) (cf. Figure 3.6) for an indefinite amount of time, in which it now cycles back and forth between observing the received data pattern for eight clock cycles and pulsing a bit slip command to the de-serializer. As a result, the PHY is never fully re-initialized and operational.

### Training Correction

The missing timeout for such broken delay settings can simply be implemented by adding a SOFT-RESET state to the training FSM. The additional state is entered after seven repeated bit slips without successful observation of the training pattern. Under this condition, the BAM has cycled through all possible permutations of the eight bit wide pattern. If the training pattern cannot be observed in the corresponding bit slip cycle, then the conclusion must be that the current delay setting is broken. Figure 3.17 shows an excerpt of the FSM that sketches the integration of the SOFT-RESET.

By entering the SOFT-RESET state, repeated decrements of the delay are issued to the de-serializer until the current tap count is set back to zero. Additionally, the registers storing the tap count settings for the data eye edges are reset before entering the initial sampling phase in FIRST-SAMPLE again.

The change introduced to the BAM FSM is accessible in the same *git* commit that alters the deterministic jitter check discussed in Section 3.3.1[17].

[17] URL: `https://gerrit.bioai.eu:9443/c/hicann-dls-private/+/22981`;
Commit hash at state of writing this thesis: `d0bb29a8b78efc981149408edd2d662af84db720`.

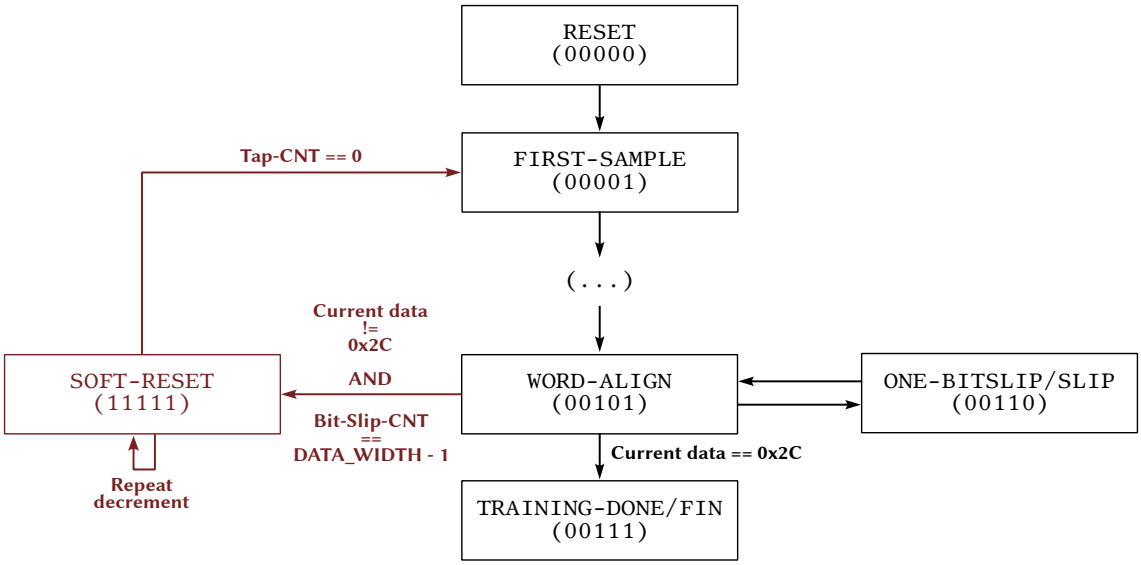



**Figure 3.17:** Schematic of the integration of a soft reset in the BAM FSM. Changes made to the training routine are highlighted in *red*.

### 3.3.4 Zero Word Timeout

A last insight that has been gained into the training procedure through the trace data analysis concerns an asymmetry in the mutual re-training process between FPGA and ASIC. In Section 3.3.2, the observation has been described that in case of a link error, the ASIC takes about 4 µs after it terminates the sending of regular data before it starts to transmit the training pattern `0x2C` (cf. Figure 3.15a). During this period, the ASIC sends a continuous stream of zeros. This behavior appears to generally apply to *any* training event, both if the re-training is triggered by the UT on the FPGA side as well as if it is triggered by the UT on the ASIC side. The FPGA on the other hand almost instantly begins with the transmission of the training signal after terminating the transmission of regular data, as is shown on TX-DATA in Figure 3.7.

Interestingly, the FPGA side of the link does not instantly report a failed header check when the static stream of zeros is observed – which is different from its behavior when it sees the training signal `0x2C` for instance, in which case it immediately issues a re-training command to the BAM. This is possible as, to the UT, a stream of zeros corresponds to a valid word that is being used during system time synchronization[18]. Although this behavior does not necessarily lead to a bad clock-to-data alignment after the training algorithm has terminated, it results in the FPGA UT to belatedly recognize the already failed ASIC side of the link. In the current implementation of the training algorithm and the UT, the ultimate reason why also the FPGA side of the link is re-trained in such events is that the UT runs into a timeout. Recall Section 3.1.1 in which the conditions have been listed that the UT uses for checking the health state of the high-speed links: One condition that classifies the link as operational is that at least one successful CRC check has to be performed within a user-specified time window by means of periodically sending a link checking word. This check is currently performed every 256 system clock cycles, which corresponds to a time span of approximately 2 µs – an interval that is shorter than the time delta during which the ASIC continuously sends zeros in advance of the training signal. Hence, the initiation of a re-training on the ASIC side of the link is expected to strongly correlate with the

[18] Personal communication with Joscha Ilmberger.

FPGA side starting a re-training in response to UT timeout notifications.

#### Training Correction

In order to shorten the time between a link failure and a successfully completed re-training, the UT has been adapted to support a new link checking condition: Whenever the receiver observes a *continuous stream of zeros* that is considered as *too long*, a re-training command is issued, assuming that the other link partner has already done the same. This is done by implementing a counter that observes the number of consecutively received eight bit wide *zero words*. When this value reaches a user-specified limit, then a re-training command is issued by the UT. In order to make sure that this condition is not applied during the re-training itself – in which the ASIC is guaranteed to send a long stream of zeros – the condition is only checked if the received data on the PHY is flagged as valid, that is, when RX-VALID is pulled high. Additionally, one needs to make sure that the chosen limit of consecutive zero words is not set too low since the link may then trigger the re-training in response to valid event data as well. For all results presented in the following, the value has been set to 128 consecutive zero words, that is half of the original link checking timeout of the UT.

A nice property that comes with the additional zero word check is that the event itself is expected to provide valuable information about *which side* of the communication channel has *failed initially*, FPGA or ASIC. Recall that the link status notifications are only reported by the FPGA side, which in principle gives the user only information about the state of the link on this end of the communication channel. If a zero word error is now reported by the UT on the FPGA, then this means that it was the ASIC that initiated the re-training first, per definition of the zero word notification. By reporting link status notifications for this link checking condition similar to other events such as link initializations or CRC errors, statistics can be gathered about the amount of re-trainings that are issued initially by the FPGA or the ASIC.

The change that implements the additional UT link checking condition and renders the corresponding status notifications accessible in software is available in a *git* commit in the `hxcomm` repository[19].

### 3.3.5 Performance Comparison

With the changes to the link training procedure that have been described in Section 3.3 at hand, it is insightful to re-evaluate the stability test of the links that has been presented in Section 3.1. Similar to before, a `perftest` is performed with the parameter settings summarized in Table 3.7 and the hardware components of the utilized Cube system summarized in Table 3.8.

Figure 3.18 displays the distribution of the time to first error $\tau_1$ with the updated training routine for PHY 5, which has previously been identified as the most frequently re-trained link for the hardware configuration shown in Table 3.8. In addition to showing the raw distribution of errors (*pink*) also the distribution of *zero word errors* (*silver*) is overlaid. This is done by filtering the link status notifications for the zero word error that has been introduced with the changes described in Section 3.3.4.

[19] URL: `https://gerrit.bioai.eu:9443/c/hxcomm/+/22989`;
Commit hash at state of writing this thesis: `fa7fb2cdc8faa30a66309c891ff1752a16a9167f`.

| Parameter | Setting |
|---|---|
| Duration | 60 s |
| Repetitions | 1075 |
| PHYs enabled | $\{0, 1, 2, 3, 4, 5, 6, 7\}$ |

**Table 3.7:** Parameters of the `perftest` generating data shown in Figure 3.18 and 3.19.

| Hardware component | ID |
|---|---|
| iBoard | `00` |
| xBoard | `23-13` |
| Chip Carrier | `0x640CF1` |

**Table 3.8:** Hardware components of Cube system `X0/W60F0` corresponding to data shown in Figure 3.18 and 3.19.

The distribution for $\tau_1$ largely exhibits the same shape as the one shown for the original re-training state (cf. Figure 3.1). Again, the majority of errors is reported shortly after link initialization with the error probability decreasing towards larger values of $\tau_1$. Most notably, the majority of the initiated link re-trainings seems to have occurred on the ASIC side of the communication channel, as the distribution of the zero word errors overlaps almost entirely with the distribution for all error status notifications.

The cumulated probability for a first failure for PHY 5 $p'_{\text{fail}} = (85.8 \pm 1.1)\,\%$ is of the same order as in the test with the original training procedure. Note that in the following, the failure probabilities observed with the *post* change training procedure will be denoted with a prime to discriminate them from the values observed in the original *pre* change training state. Table 3.9 summarizes the observed probabilities for a first failure across all PHYs, split between both training routines.

| | PHY | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|---|
| Post | $p'_{\text{fail}}$ [%] | 0.0 | 0.0 | 0.0 | 71.8 | 0.0 | 85.8 | 0.1 | 0.0 |
| | SEM [%] | ±0.0 | ±0.0 | ±0.0 | ±1.4 | ±0.0 | ±1.1 | ±0.1 | ±0.0 |
| | $N_{\text{obs}}$ | 1075 | 1075 | 1075 | 1075 | 1075 | 1075 | 1075 | 1075 |
| Pre | $p_{\text{fail}}$ [%] | 0.3 | 0.0 | 0.0 | 70.7 | 0.0 | 89.1 | 0.2 | 0.1 |
| | SEM [%] | ±0.1 | ±0.0 | ±0.0 | ±1.2 | ±0.0 | ±0.8 | ±0.1 | ±0.1 |
| | $N_{\text{obs}}$ | 1372 | 1372 | 1372 | 1372 | 1372 | 1372 | 1372 | 1372 |

**Table 3.9:** Probability $p'_{\text{fail}}$ for a first fail to occur for PHYs 0 to 7 during each of 1075 repetitions of a 60 s `perftest` with updated training procedure. For comparison, also the failure probabilities $p_{\text{fail}}$ with the original link training procedure are summarized.

Within the corresponding three sigma intervals specified by the respective SEM, the probabilities for a first fail do not change significantly across all PHYs. That is, the updated training procedure does not seem to have a considerable impact on whether or not a link fails for a first time in the given time window of 60 s.

Intuitively, this seems reasonable as the distribution of zero word errors displayed in Figure 3.18 almost accounts for the entirety of the observed errors. With the definition of the zero word error, this is to be interpreted such that the UT on the ASIC side almost exclusively observes

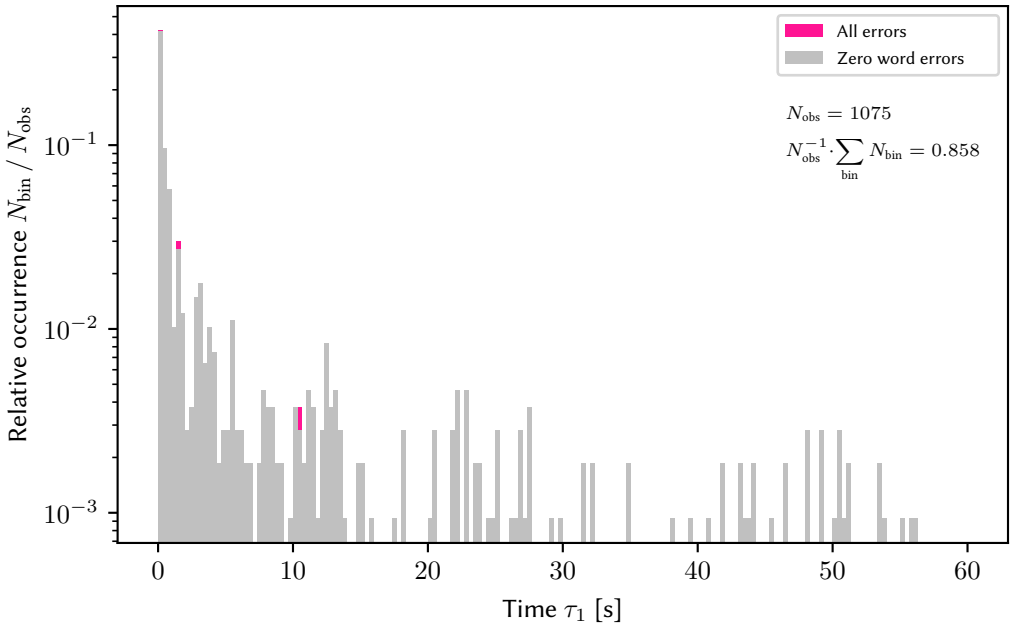



**Figure 3.18:** Relative number of link failures after first initialization for PHY 5 with updated link training procedure and zero word notifications. The distribution of zero word errors (*silver*) is plotted over the raw distribution (*pink*) and accounts for almost the entire amount of observed errors. The histogram bin width corresponds to $\Delta\tau = 0.3\overline{3}\,\text{s}$.

the first errors that occur on the link in total. Since the changes to the training procedure only concern the FPGA side of the link, their impact on the probability for a first error is expected to be minimal. Additionally, the training errors that have been presented in Section 3.3 have always been observed in *response* to a first error. That is, only the *re*-training of the links has been adjusted, not the initial link training performed at system start up.

A more pronounced impact of the changes to the training routine can be observed regarding the time to second failure $\tau_2$. Similar to before, Figure 3.19 displays the distribution of $\tau_2$ together with the distribution of errors that can be linked to a zero word error. In contrast to the state of the original training routine presented in Figure 3.4, there have essentially no errors been observed for $\tau_2 > 2\,\text{s}$. At a total of $N_{\text{obs}} = 922$ first error observations, second errors are occasionally reported right after the link has been re-initialized anew. However, the relative occurrence has decreased compared to the original state. Moreover, the observed errors can exclusively be linked to a failure of the ASIC side of the link, as the distribution for the zero word errors matches the raw distribution of all errors exactly.

The cumulated failure probability of $p'_{\text{fail}} = (1.2 \pm 0.4)\,\%$ for a second failure for PHY 5 is significantly smaller than in the original state. Table 3.10 summarizes the total probabilities for a second failure across all PHYs and provides the values corresponding to the original training procedure for comparison.

With respect to the three sigma intervals defined by the SEM, the probabilities for PHY 3 and 5 to fail for a second time within the given time window of 60 s have decreased significantly between the original and the updated training routine. That is to say, for the considered Cube setup `X0/W60F0` with the components listed in Table 3.8 the link stability seems to have increased

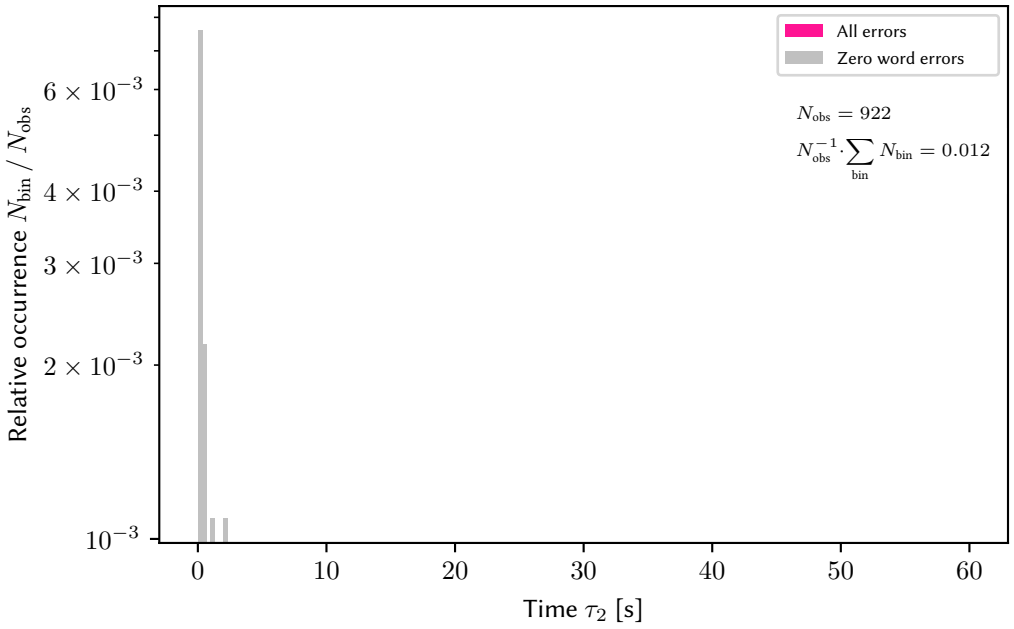



**Figure 3.19:** Relative number of second link failures after first re-training for PHY 5 with updated link training procedure and zero word notifications. The distribution of zero word errors (*silver*) is plotted over the raw distribution (*pink*) and accounts for the entirety of observed errors. The histogram bin width corresponds to $\Delta\tau = 0.3\overline{3}$ s.

in response to an initial first error. The observed first errors and remaining second errors seem to be dominated by the link failing at the ASIC side. The re-training procedure on this side of the links is however implemented statically and can only be changed in an updated version of the HICANN-X.

| | PHY | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|---|
| Post | $p'_{\text{fail}}$ [%] | 0.0 | 0.0 | 0.0 | 1.4 | 0.0 | 1.2 | 0.0 | 0.0 |
| | SEM [%] | ±0.0 | ±0.0 | ±0.0 | ±0.4 | ±0.0 | ±0.4 | ±0.0 | ±0.0 |
| | $N_{\text{obs}}$ | 0 | 0 | 0 | 772 | 0 | 922 | 0 | 0 |
| Pre | $p_{\text{fail}}$ [%] | 25.0 | 0.0 | 0.0 | 58.9 | 0.0 | 67.9 | 0.0 | 100.0 |
| | SEM [%] | ±21.6 | ±0.0 | ±0.0 | ±1.6 | ±0.0 | ±1.3 | ±0.0 | ±0.0 |
| | $N_{\text{obs}}$ | 4 | 0 | 0 | 970 | 0 | 1223 | 0 | 1 |

**Table 3.10:** Probability $p'_{\text{fail}}$ for a second fail to occur for PHYs 0 to 7 during each of 1075 repetitions of a 60 s `perftest` with updated training procedure. For comparison, also the failure probabilities $p_{\text{fail}}$ with the original link training procedure are summarized.

## 3.4 Investigation of Relevant ASIC System Parameters

Although the ASIC side of the high-speed links is inaccessible for changes of the training procedure, from a user's point of view there still are settings available for configuration that may further alter the performance of the high-speed links. A different way to potentially improve the link stability is to try to tweak the parameters of the LVDS input stage of the de-serializer in the HICANN-X.

The goal is to find an operating point of the circuitry for which the probability of first failures after initialization is minimized given the fixed setup configuration.

Figure 3.20 displays a simplified schematic of the LVDS input cell implemented in the ASIC receiver macros. The input current over the differential pair is converted to a voltage difference $V_{\mathrm{IN}}^{+} - V_{\mathrm{IN}}^{-}$ by an internal 100 Ω differential termination. Both the positive and the negative input are guarded by *electrostatic discharge* (ESD) protection structures. The input cell then converts the voltage difference from the 2.5 V IO voltage domain to a single-ended output $V_{\mathrm{OUT}}$ in the 1.2 V core voltage domain. $V_{\mathrm{OUT}}$ is subsequently passed to the de-serializer circuitry. The amplifier stages used in the input cell are operating on digital supply voltages, $V_{\mathrm{DD25D}}$ and $V_{\mathrm{DD12D}}$, respectively, and a bias current $I_{\mathrm{BIAS}}$ that is drawn from a common current source. The supply voltages as well as the bias current are adjustable and can be configured by the user.

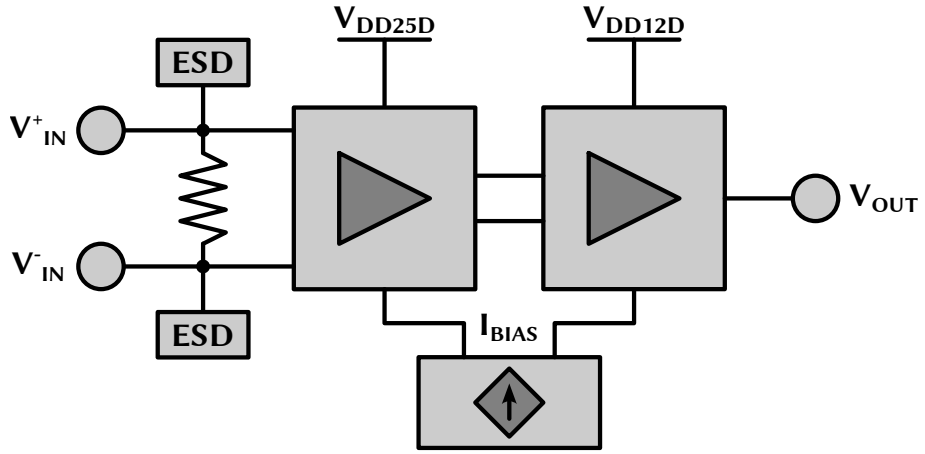



**Figure 3.20:** Simplified schematic of the LVDS receiver cell of the HICANN-X. The input cell consists of two input pads that are connected via a 100 Ω differential termination resistor across which the differential input signal $V_{\mathrm{IN}}^{+} - V_{\mathrm{IN}}^{-}$ is measured. ESD protection structures are connected to the input pads as well. A differential amplifier stage converts the differential signal from the $V_{\mathrm{DD25D}} = 2.5\,\mathrm{V}$ IO to the $V_{\mathrm{DD12D}} = 1.2\,\mathrm{V}$ core voltage domain. The differential signal is fed to a comparator stage to convert it to a single-ended signal $V_{\mathrm{OUT}}$ that is passed forward to the de-serializer circuit. Both the differential amplifier and the comparator stage operate on a bias current $I_{\mathrm{BIAS}}$ that can be adjusted through a voltage controlled current source.

By varying the digital supply voltages $V_{\mathrm{DD12D}}$ and $V_{\mathrm{DD25D}}$, respectively, as well as the bias current $I_{\mathrm{BIAS}}$, the quiescent points of the transistors in the amplifier stages are shifted. This may alter the performance of the employed amplifier stages and thus the quality of the signal that is passed to the de-serializer. Additionally, the line of configurable delay elements used within the clock-to-data alignment (cf. Section 3.2) operates within the core voltage domain, which may be impacted by changing the 1.2 V supply voltage. Ultimately, one expects this to lead to an overall change in the failure rates of the high-speed links. Note that with the 1.8 V FPGA IO voltage there is an additional parameter for configuration available at the transmitter side on the FPGA which is expected to have an impact on the performance. However, the variation of this parameter has not been prioritized as the configuration of the ASIC input stage is expected to have a larger impact on the failure probabilities[20].

This section describes the testing procedure for finding a suitable operating point with respect to $V_{\mathrm{DD12D}}$, $V_{\mathrm{DD12D}}$ and $I_{\mathrm{BIAS}}$. Multiple systems have been investigated and exhibit similarities in the

[20] Personal communication with Joscha Ilmberger.

link failure probability in response to changes of the system parameters. As this investigation has been carried out shortly before the submission of this thesis, only preliminary results will be presented. Besides a discussion of the gathered results, a recommendation on how a user can identify a good operating point for any setup will be given.

### 3.4.1 Testing Routine

For the most part, the system parameter sweeps discussed in the following have been performed with the same testing systematic as outlined in Section 3.1.3. Slight adaptations have been done to account for the additionally required measurements of the parameters to be varied and the resulting increase in the total testing time.

#### Voltage Measurement Procedure

When testing for a specific configuration of $V_{\mathrm{DD12D}}$, $V_{\mathrm{DD25D}}$ and $I_{\mathrm{BIAS}}$, the setting of these parameters is followed by a `perftest` with a fixed duration. Similar to before, the execution of the `perftest` corresponds to a Bernoulli experiment, whether or not a given link will fail within the given time frame. The configuration of the system parameters is done before executing the `perftest` by modifying the corresponding member values in the `DigitalInit` of the experiment. Additionally, a single-shot read of the $V_{\mathrm{DD12D}}$ and the $V_{\mathrm{DD25D}}$ supply voltages is performed in advance of the `perftest`. This is done by reading out the bus supply voltage registers of a *Texas Instruments INA219* bidirectional current and power monitoring device (Texas Instruments Incorporated 2008) located on the xBoard of the Cube systems.

Concerning the accuracy of these measurements, the application note for the INA219 reports a bus voltage resolution of 4 mV per *least significant bit* (LSB) and a direct current accuracy of the measured bus voltage of 0.2 % with respect to the absolute reading. Given a supply voltage setting of $V_{\mathrm{DD12D}} = 1.2\,\mathrm{V}$ and $V_{\mathrm{DD25D}} = 2.5\,\mathrm{V}$, respectively, the absolute measurement errors $\sigma_{\mathrm{DD12D}}$ and $\sigma_{\mathrm{DD25D}}$ are expected to be on the order of

$$\sigma_{\mathrm{DD12D}} \approx 0.002 \cdot 1.2\,\mathrm{V} = 2.4\,\mathrm{mV}, \tag{3.12}$$

and

$$\sigma_{\mathrm{DD25D}} \approx 0.002 \cdot 2.5\,\mathrm{V} = 5\,\mathrm{mV}. \tag{3.13}$$

The read out register values $V_{\mathrm{DD}}^{\mathrm{bit}} \in [0, 4095]$ are converted to an uncalibrated voltage via

$$V_{\mathrm{DD}} = \frac{4 \cdot V_{\mathrm{DD}}^{\mathrm{bit}}}{1000}\ [\mathrm{V}]\,, \tag{3.14}$$

as no calibration data has been readily available for the various setups. In this equation, the subscript DD serves as a placeholder for both DD12D and DD25D, respectively. Due to the

missing calibration data, an unknown mismatch is to be expected when comparing the voltage measurements of different Cube setups.

The precision of the measurements is determined by averaging the single-shot voltage read outs that correspond to a specific setting of core and IO voltage as well as the bias current. That is, given a single measurement $V_{\mathrm{DD},i}^{\boldsymbol{\theta}}$ for a distinct combination of parameters $\boldsymbol{\theta} = \left(V_{\mathrm{DD12D}}^{\mathrm{bit}}, V_{\mathrm{DD25D}}^{\mathrm{bit}}, I_{\mathrm{BIAS}}^{\mathrm{bit}}\right)$ during `perftest` repetition $i$, the mean of the set voltages is estimated by averaging all samples that have been recorded at the identical parameter settings over the total amount of test repetitions $N_{\mathrm{obs}}$,

$$\overline{V}_{\mathrm{DD}}^{\boldsymbol{\theta}} = \frac{1}{N_{\mathrm{obs}}} \cdot \sum_{i=1}^{N_{\mathrm{obs}}} V_{\mathrm{DD},i}^{\boldsymbol{\theta}}. \tag{3.15}$$

The corresponding unbiased estimate of the standard deviation,

$$\hat{\sigma} = \sqrt{\frac{1}{N_{\mathrm{obs}} - 1} \cdot \sum_{i=1}^{N_{\mathrm{obs}}} \left(V_{\mathrm{DD},i}^{\boldsymbol{\theta}} - \overline{V}_{\mathrm{DD}}^{\boldsymbol{\theta}}\right)^2}, \tag{3.16}$$

gives rise to the SEM of the voltage setting:

$$\mathrm{SEM}\left(V_{\mathrm{DD}}^{\boldsymbol{\theta}}\right) = \frac{\hat{\sigma}}{\sqrt{N_{\mathrm{obs}}}}. \tag{3.17}$$

This is the *statistical* error that every distinct voltage setting for either $V_{\mathrm{DD12D}}$ or $V_{\mathrm{DD25D}}$ is equipped with, given a *fixed* setting of the other parameters. The *systematic* errors of the voltage readings are given by the measurement errors $\sigma_{\mathrm{DD12D}}$ and $\sigma_{\mathrm{DD25D}}$ as specified for the INA219 and an unknown offset stemming from the missing calibration data.

Note that concerning the bias current, there is no similar direct monitoring option available. Therefore, only the high-level configuration setting of the bias current $I_{\mathrm{BIAS}}^{\mathrm{bit}}$ will be referred to in the following. Also note that, as the measurements for a given hardware setup are carried out over several minutes and hours, the setting of the parameters is fully randomized by shuffling the settings for each repetition of the sweep. This procedure aims to account for systematic environmental changes, for instance in room temperature, during the test execution.

### Testing Time

The variation of the supply voltages in a suitable range and at a proper resolution while performing changes to the bias current has punishing demands on the overall testing time. Essentially, the search for a good operating point corresponds to performing a sweep of the link failure probabilities along a three-dimensional grid in the parameter space spanned by $V_{\mathrm{DD12D}}$, $V_{\mathrm{DD25D}}$ and $I_{\mathrm{BIAS}}$. In the case of the Cube systems, the digital supply voltages for the ASIC are controlled by 12 bit *digital-to-analog converter*s (DACs), respectively, allowing for 4096 different settings

each. The bias current in the ASIC is set up by a 3 bit JTAG control word and allows for eight different configurations. Consider performing, for instance, a sparse sweep of five by five values of the supply voltages at a fixed bias current: The accumulated testing time of the loop-back tests for moving through all possible combinations of $V_{\mathrm{DD12D}}$ and $V_{\mathrm{DD25D}}$ at the same testing time of 60 s per iteration that used in Section 3.1.4 would amount to 25 min. In order to gather 100 samples per distinct setting, the total testing time would correspond to

$$100 \cdot 25\,\mathrm{min} \approx 42\,\mathrm{h}. \tag{3.18}$$

However, the stability tests performed in Section 3.3.5 have shown that the majority of first errors occurs directly after link initialization. For instance, for the most frequently failing links PHY 3 and 5, about 40 % to 50 % of the first failures have been observed within the first 333 ms after initialization. Hence, in order to increase the sampling speed of the statistics during the system parameter sweep it has been considered as sufficient to focus on the failure probabilities within a time window of 350 ms[21].

#### Hyper-Parameter Settings

Unless otherwise noted, the `perftest` settings summarized in Table 3.11 have been used for sampling the failure probabilities during the system parameter sweeps.

| Parameter | Setting |
|---|---|
| Duration | 350 ms |
| Repetitions per system parameter setting $\boldsymbol{\theta}$ | 100 |
| PHYs enabled | $\{0, 1, 2, 3, 4, 5, 6, 7\}$ |

**Table 3.11:** Parameters of the `perftest` used during ASIC system parameter sweeps.

### 3.4.2 Results

During an initial $5 \times 5$ grid search with a symmetric variation of $\pm 10$ % relative to the standard settings $V_{\mathrm{DD12D}} = 1.2\,\mathrm{V}$ and $V_{\mathrm{DD25D}} = 2.5\,\mathrm{V}$ it has been observed that varying $V_{\mathrm{DD12D}}$ seems to have a more pronounced impact on the failure probabilities compared to $V_{\mathrm{DD25D}}$. Additionally, no major change in the failure probabilities could be detected when setting the bias current to the maximum and the minimum configurable value. Therefore, a fine sweep of the 1.2 V core supply voltage $V_{\mathrm{DD12D}}$ has been carried out.

#### Core Supply Voltage

Figure 3.21 displays the probability for a first failure depending on the core supply voltage $V_{\mathrm{DD12D}}$ for Cube setup `X0/W60F3` with the hardware components summarized in Table 3.12.

[21] Personal communication with Johannes Schemmel and Joscha Ilmberger.

| Hardware component | ID |
|---|---|
| iBoard | `00` |
| xBoard | no ID available, silkscreen annotation: `xboard_r0, 11/18` |
| Chip Carrier | `0x640CF1` |

**Table 3.12:** Hardware components of Cube system `X0/W60F3` used for ASIC system parameter sweeps corresponding to Figure 3.21.

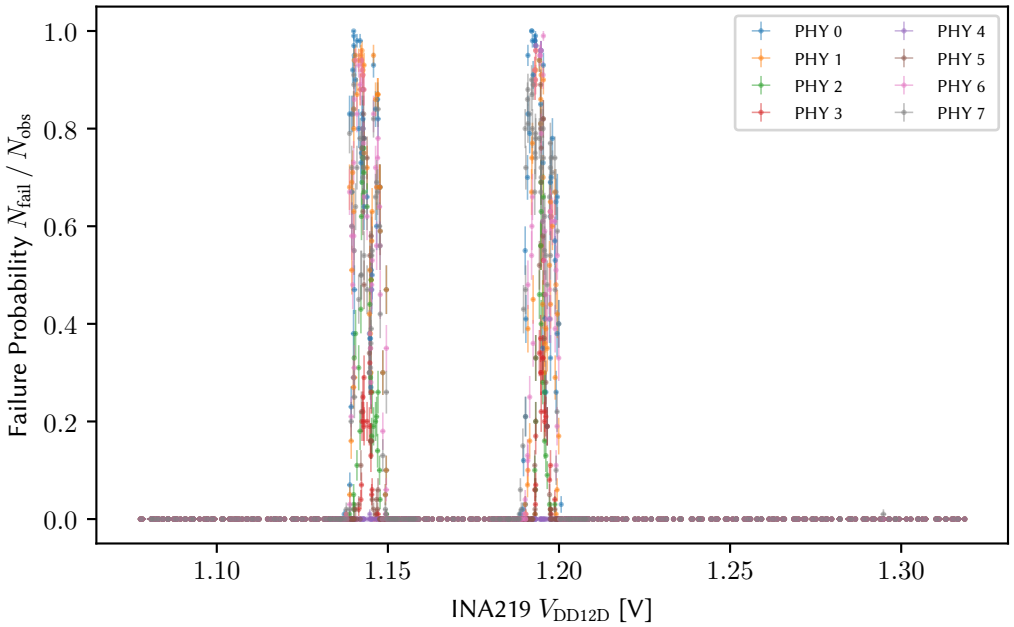



**Figure 3.21:** Probability for a first link failure during a 350 ms `perftest` for all eight PHYs of chip carrier `0x640CF1` on Cube system `X0/W60F3` dependent on the digital ASIC core supply voltage $V_{\text{DD12D}}$. The error bars in the x- and y-direction correspond to the SEM estimated from the INA219 bus voltage samples and the statistics of the Bernoulli distribution. An offset error of the voltage readout of $\sigma_{\text{DD12D}} \approx 2.4\,\text{mV}$ over the entire voltage range and an unknown error from missing calibration data are to be expected.

The 1.2 V core supply voltage has been varied by approximately $\pm 10\,\%$ relative to the standard setting of $V_{\text{DD12D}} = 1.2\,\text{V}$. The probabilities for a first link failure during the 350 ms `perftest` are displayed in Figure 3.21 for all eight links. Two distinct and sharply pronounced peaks can be observed in the voltage ranges of 1.14 V to 1.15 V and 1.19 V to 1.20 V. The failure probabilities in the corresponding ranges peak up to 1.0 for certain links such as PHY 0 (*blue*), 1 (*yellow*) and 6 (*pink*). Other links such as PHY 3 (*red*), for instance, also show an increased failure probability but are not failing as frequently. Only PHY 4 (*purple*) seems to be stable within the specified voltage range. Outside and in between the two distinct peaks, the failure probabilities for all PHYs are close to zero.

The location and sharp outline of the peaks is unexpected. Physically, it would be more natural to assume that close to the default setting of $V_{\text{DD12D}} = 1.2\,\text{V}$ the failure probability is minimal and that deviations towards higher or lower voltages lead to a higher probability for link losses. The results for the sweep shown in Figure 3.21 are in stark contrast to this assumption. The small distance of the right peak to the default setting of 1.2 V also explains the high observed failure

probability for first link failures outlined in the 60 s `perftest` in Section 3.3.5. The measurements have been carried out very close to the range of hazardous $V_{\mathrm{DD12D}}$ settings. Moreover, the existence of *two* distinct peaks, separated by a voltage range in which the failure probability is close to zero, is unexpected. This raises the question about the cause of the observed dependency of the link failure probability on the core supply voltage.

In order to examine the observed sharp increase in the failure probability more closely, the right peak shown in Figure 3.21 has been swept repeatedly for different setup and chip carrier configurations. Figure 3.22 displays the accumulated data for multiple chip carriers installed in Cube system `X0/W60F3`. Apart from the chip carrier, the hardware configuration is the same as summarized in Table 3.12.

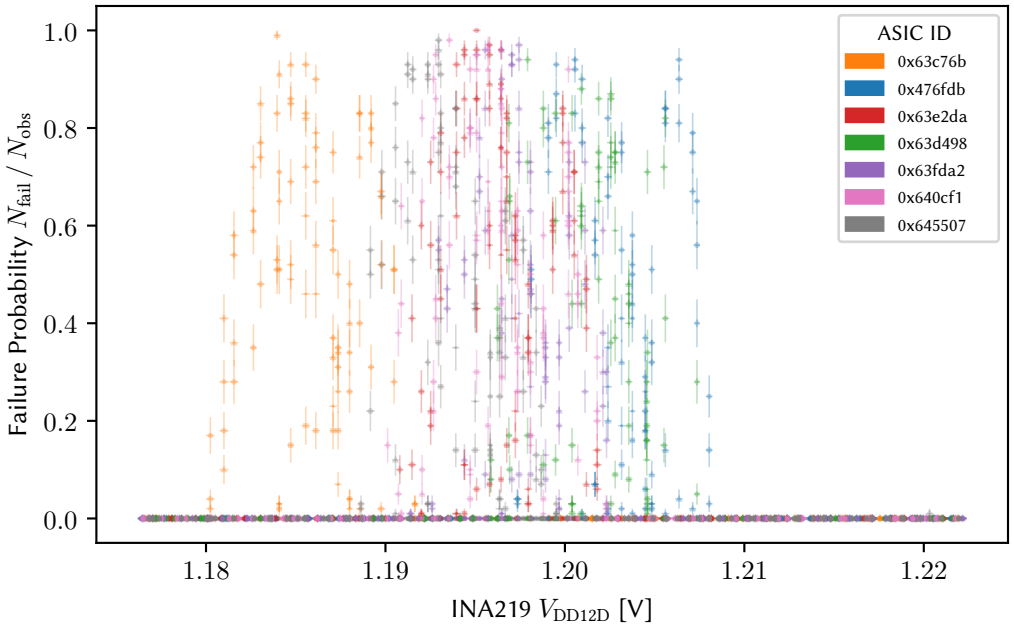



**Figure 3.22:** Probability for a first link failure during a 350 ms `perftest` dependent on the digital ASIC core supply voltage $V_{\mathrm{DD12D}}$, aggregated for various chip carriers on Cube system `X0/W60F3`. The eight distinct PHYs per chip carrier are encoded through individual marker symbols. The error bars in the x- and y-direction correspond to the SEM estimated from the INA219 bus voltage samples and the statistics of the Bernoulli distribution. An offset error of the voltage readout of $\sigma_{\mathrm{DD12D}} \approx 2.4\,\mathrm{mV}$ over the entire voltage range and an unknown error from missing calibration data are to be expected.

Figure 3.22 shows that the failure probability is increased in a $V_{\mathrm{DD12D}}$ range of 1.18 V to 1.21 V for multiple PHYs across different chip carriers. The position of the peak is slightly shifted between different chip carriers, that is, the hazardous voltage range is slightly offset on an order of 10 mV when utilizing different ASICs on the same setup.

The observed peak in the probability for a first failure has been confirmed to be present on other systems as well. Figure 3.23 displays the aggregated sweep data for the same chip carriers tested in Cube system `X5/W65F3` with the hardware configuration summarized in Table 3.13.

In contrast to Cube setup `X0/W60F3`, Cube setup `X5/W65F3` is powered by a lab power supply. However, the variation in the hardware configuration does not seem to impact the overall high failure probability across the different chip carriers when comparing the sweep results shown

| Hardware component | ID |
|---|---|
| iBoard | X5 |
| xBoard | 23-12 |

**Table 3.13:** Hardware components of Cube system X5/W65F3 used for ASIC system parameter sweeps corresponding to Figure 3.23.

in Figure 3.22 and 3.23. The relative offset in between the peaks for different chip carriers seems to vary when the boards are being swapped across the Cube systems. Also, the total hazardous voltage range for Cube setup X5/W65F3 is slightly broader, spanning across a range of approximately 1.17 V to 1.21 V. It is to be re-emphasized though that the absolute voltage scale between the two figures is not exactly comparable, as the measurement error specified for the INA219 and the unknown calibration data for the setups need to be taken into account. Nevertheless, the varying location of the peak for different combinations of chip carrier and Cube setup could be an explanation for the observation that the high-speed links fail on certain setups more often than on others.

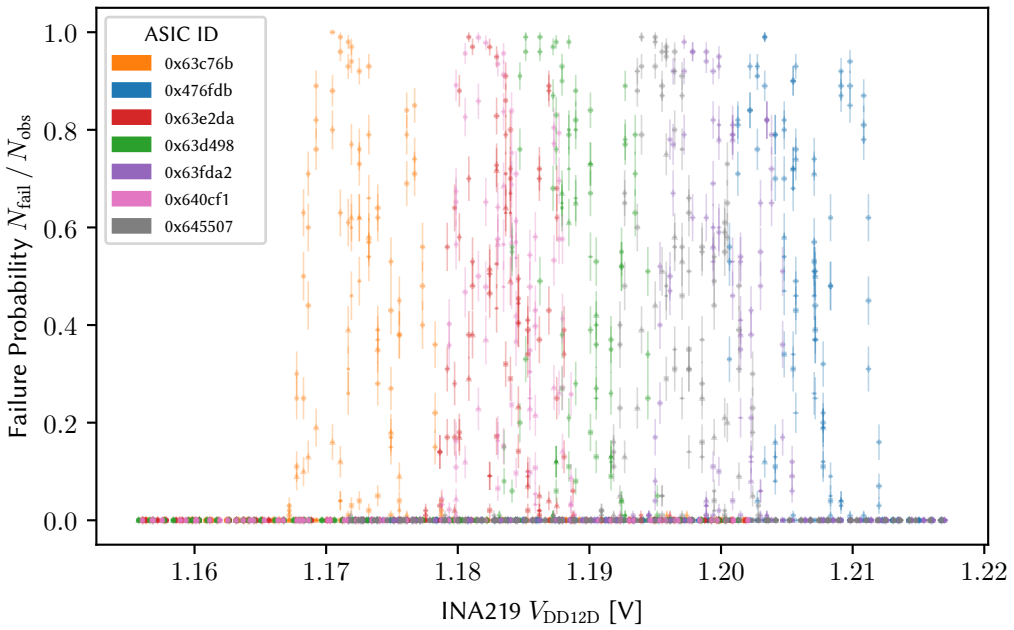



**Figure 3.23:** Probability for a first link failure during a 350 ms `perftest` dependent on the digital ASIC core supply voltage $V_{\text{DD12D}}$, aggregated for various chip carriers on Cube system X5/W65F3. The eight distinct PHYs per chip carrier are encoded through individual marker symbols. The error bars in the x- and y-direction correspond to the SEM estimated from the INA219 bus voltage samples and the statistics of the Bernoulli distribution. An offset error of the voltage readout of $\sigma_{\text{DD12D}} \approx 2.4\,\text{mV}$ over the entire voltage range and an unknown error from missing calibration data are to be expected.

Lastly, the failure probability also seems to be increased on a setup with a vastly different hardware configuration. Figure 3.24 displays the results of a $V_{\text{DD12D}}$ sweep on a jBOA prototype setup. This setup uses a different set of PCBs equipped with different LDOs for supplying the digital IO and core voltages to the ASICs. The range of possible settings of the DAC for setting the supply voltages is only eight bit wide, hence the resulting distributions are sampled more sparsely. Again, the failure probabilities across all PHYs of all chip carriers are strongly increased. The

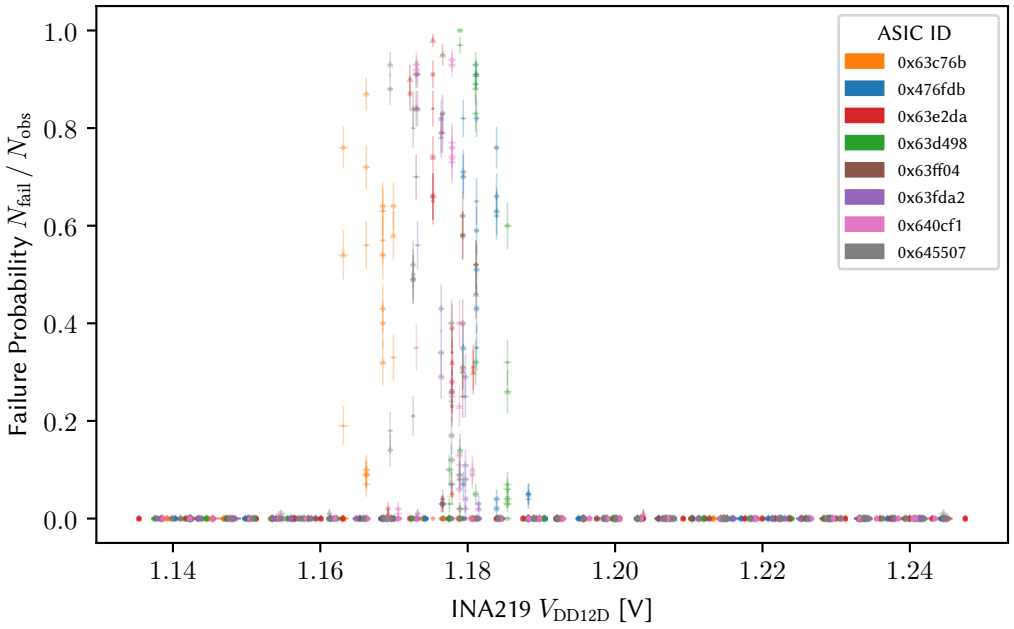



**Figure 3.24:** Probability for a first link failure during a 350 ms `perftest` dependent on the digital ASIC core supply voltage $V_{\text{DD12D}}$, aggregated for various chip carriers on jBOA system `W80F0`. The eight distinct PHYs per chip carrier are encoded through individual marker symbols. The error bars in the x- and y-direction correspond to the SEM estimated from the INA219 bus voltage samples and the statistics of the Bernoulli distribution. An offset error of the voltage readout of $\sigma_{\text{DD12D}} \approx 2.4\,\text{mV}$ over the entire voltage range and an unknown error from missing calibration data are to be expected.

corresponding hazardous voltage range is approximately 1.16 V to 1.19 V.

### IO Supply Voltage

With the ASIC core supply voltage range of roughly 1.16 V to 1.21 V fixed, in which the sharp increase of the failure probability can be observed reliably, the IO supply voltage $V_{\text{DD25D}}$ has been swept. Figure 3.25 displays sweeps of the core supply voltage, stacked for different settings of the IO supply voltage. In this case, the step size in $V_{\text{DD25D}}$ has been chosen coarsely and corresponds to approximately 50 mV. This is done in order to first identify a potential shift in the distributions along the $V_{\text{DD12D}}$ voltage axis with respect to changes in $V_{\text{DD25D}}$, in which case a full grid sweep of the core against the IO supply voltage would be necessary.

The set of peaks shown in Figure 3.25 has been sampled on Cube system `X5/W65F3` with the hardware configuration summarized in Table 3.13 and chip carrier `0x645507` being tested. Across all of the five different IO supply voltage settings that have been tested, no noticeable shift of the peaks in the failure probability for the different PHYs has been observed. The location and shape of the peak seems to be stable against any change of $V_{\text{DD25D}}$ in the considered range of approximately 2.45 V to 2.66 V.

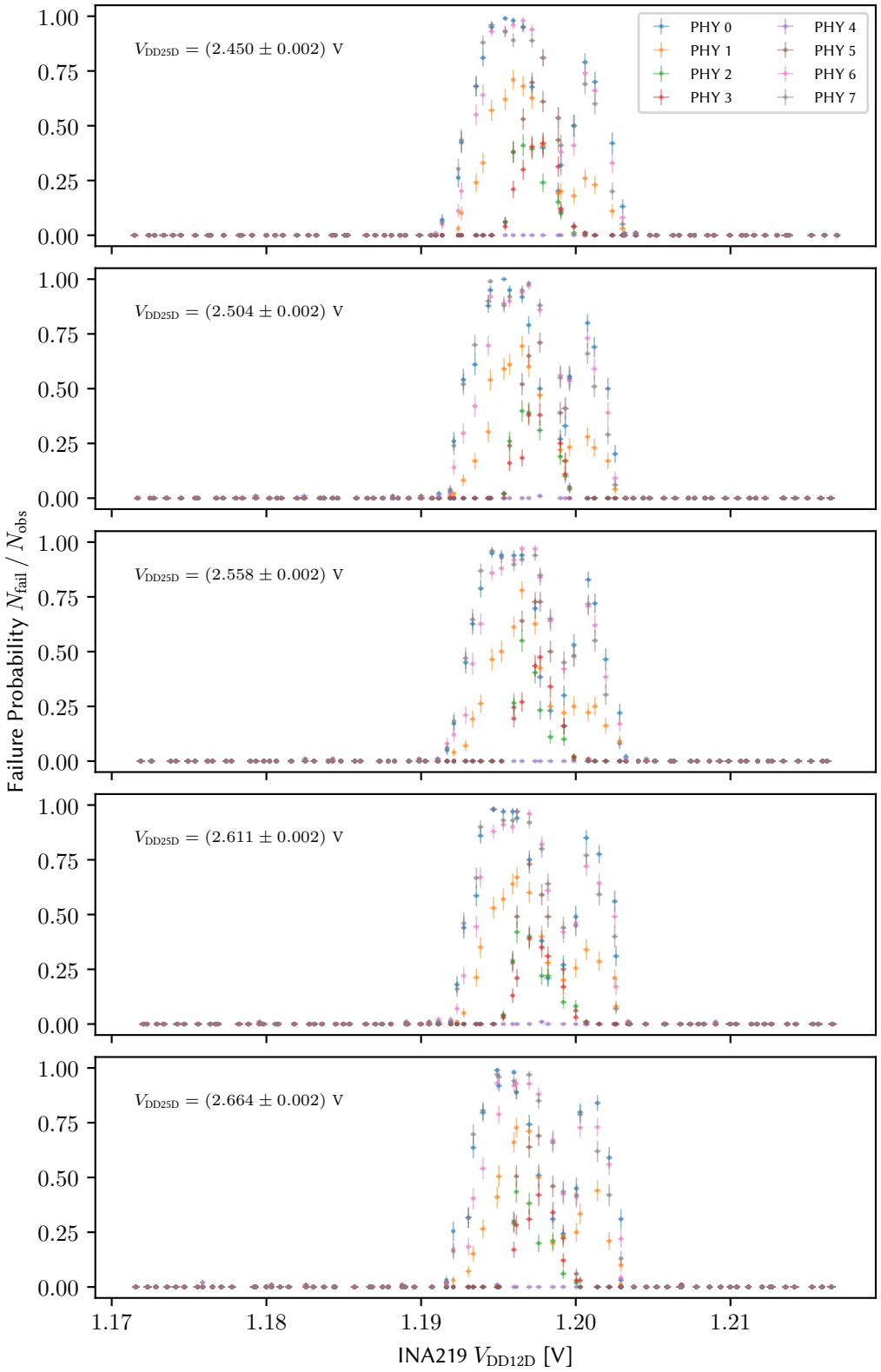



**Figure 3.25:** Probability for a first link failure with chip carrier `0x645507` on Cube system `X5/W65F3` dependent on the digital ASIC core supply voltage $V_{DD12D}$ for different configurations of the IO supply voltage $V_{DD25D}$. The error bars in the x- and y-direction correspond to the SEM estimated from the INA219 bus voltage samples and the statistics of the Bernoulli distribution. An offset error of the voltage readout of $\sigma_{DD12D} \approx 2.4\,\text{mV}$ over the entire voltage range and an unknown error from missing calibration data are to be expected. For the IO supply voltage, an offset error of approximately $\sigma_{DD25D} \approx 5\,\text{mV}$ is to be expected.

In order to verify if a change in the ASIC IO supply voltage does indeed not have an impact on the observed failure probabilities, a fine sweep of $V_{\mathrm{DD25D}}$ has been carried out with the same hardware configuration. By choosing a value of $V_{\mathrm{DD12D}} \approx 1.21\,\mathrm{V}$, the sweep has been performed at a core voltage value outside the characteristic peak range, in which the influence of $V_{\mathrm{DD12D}}$ should be minimal with respect to changes in the IO supply voltage (cf. Figure 3.25).

Figure 3.26 displays the results for the fine sweep of $V_{\mathrm{DD25D}}$ at the fixed core voltage setting. With the chosen value of $V_{\mathrm{DD12D}} \approx 1.21\,\mathrm{V}$, there are essentially no failures observable across all links for any setting of the IO voltage. The links are stable for almost any configuration – with the exception of a few failures occurring at voltages $V_{\mathrm{DD25D}} > 2.6\,\mathrm{V}$. For the given setup and chip carrier pair, the chosen core supply voltage may be considered as a valid operating point that minimizes the link failure probability.

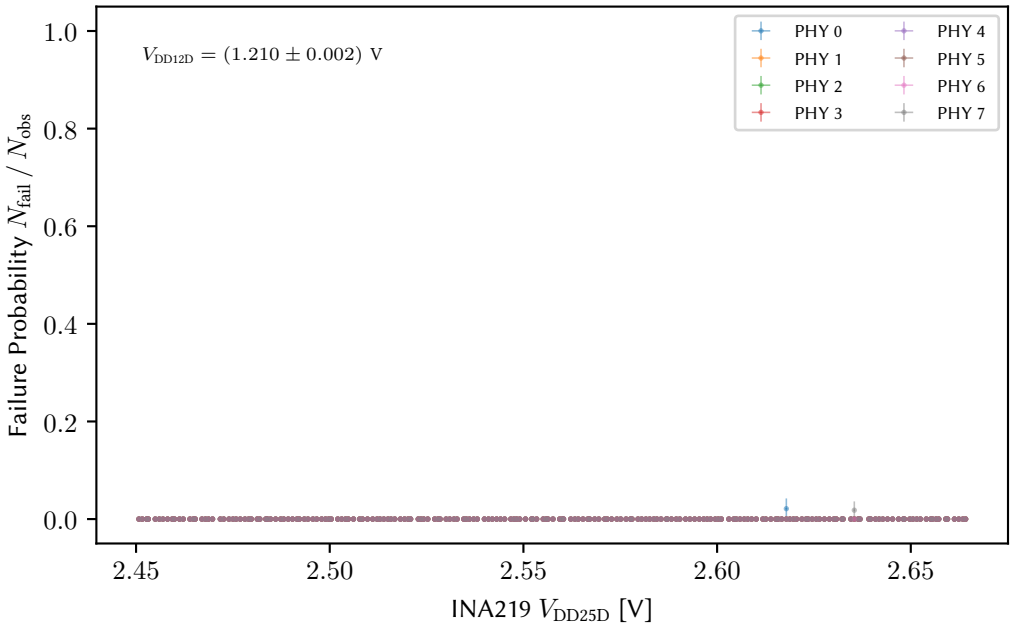



**Figure 3.26:** Probability for a first link failure with chip carrier `0x645507` on Cube system `X5/W65F3` dependent on the digital ASIC IO supply voltage $V_{\mathrm{DD25D}}$ at fixed $V_{\mathrm{DD12D}} \approx 1.21\,\mathrm{V}$. The error bars in the x- and y-direction correspond to the SEM estimated from the INA219 bus voltage samples and the statistics of the Bernoulli distribution. An offset error of the voltage readout of $\sigma_{\mathrm{DD25D}} \approx 5.0\,\mathrm{mV}$ over the entire voltage range and an unknown error from missing calibration data are to be expected. For the core supply voltage, an offset error of approximately $\sigma_{\mathrm{DD12D}} \approx 2.4\,\mathrm{mV}$ is to be expected.

By again choosing $V_{\mathrm{DD12D}} \approx 1.21\,\mathrm{V}$ and $V_{\mathrm{DD25D}} \approx 2.50\,\mathrm{V}$ for the same hardware configuration as used in the sweep shown in Figure 3.26, the stability of the links has also been tested with an extended testing time of 60 s without variation of the ASIC system parameters. For $N_{\mathrm{obs}} \approx 1600$ repetitions of the `perftest` there has *no* first error been observed on any of the eight links. This suggests that users may indeed find a good operating point by picking a suitable setting of $V_{\mathrm{DD12D}}$.

#### Bias Current

During all performed sweeps of the digital supply voltages, no measurable effect of changing the configuration of the bias current $I_{\mathrm{BIAS}}$ in the LVDS input cell on the failure probabilities could be detected. This has been tested by setting the eight bit configuration register to `000` and `111`, respectively. At the point of writing this thesis, the reason for the bias current not having any measurable impact on the failure probabilities remains unclear.

#### Other Hyper-Parameters

Additional checks have been performed in order to identify hyper-parameters that may have an impact on the location and shape of the peaks in the failure probability with respect to the core supply voltage $V_{\mathrm{DD12D}}$. This includes

- variations of the PLL supply voltage by approximately $\pm 20\,\mathrm{mV}$ relative to the default settings and
- a decrease of the number of enabled links from eight to enabling only two PHYs during the `perftest` execution.

The PLL in the ASIC operates on a separate 1.2 V power rail and provides clocking for the PHYs and core logic, so perhaps an overlap with the digital core supply voltage leads to an increased instability of the links. However, within the specified variation range, no impact has been observed on the location and shape of the failure probability peaks. The range of variation has been chosen narrowly to match the small voltage ranges that define the observed peaks. It is unclear whether the failure probabilities are indeed independent of the PLL supply voltage or the variation around the default value has been chosen too small to exert a measurable effect.

Concerning the influence of the number of high-speed links that are enabled simultaneously during system initialization, there is a possibility for an impact on the failure probabilities. Specific links exhibit a slightly decreased probability to fail given that only one other link is enabled. This is however not the case for all setups `W60F3`, `W60F0` and `W80F0`: Repeating the test with the same chip carrier across the different systems does lead to an improvement in some cases, but to no measurable improvement in others.

In either case, further testing is needed to fully investigate the effect of the parameter variations on the failure probabilities.

## 3.5 Summary and Discussion

In this chapter, the stability of the high-speed serial links between FPGA and ASIC has been analyzed. The stability of the links has been characterized by two quantities: A time to failure and a frequentist probability for a link to fail within a given time frame. A systematic testing routine has been developed to investigate and compare the link stability for different PHYs on different

setups. By analyzing the in-hardware trace data of the FPGA receiver, potential culprits in the link training routine have been identified that lead to suboptimal delay settings. Additionally, sweeps of the system parameters of the ASIC have been performed to optimize the quality of the received signal.

Up to this point, two major factors that influence the stability of the high-speed serial links between FPGA and ASIC have been identified:

1. The probability for a first error strongly depends on specific regions within the digital ASIC core supply voltage.

2. At a fixed operating point, the probability for a subsequent error in response to a link re-training triggered by a first error can be reduced by adapting the clock-to-data alignment routine at the FPGA receiver.

Although it is unclear at the time of thesis submission where the strong dependency of the first failure probability on $V_{\mathrm{DD12D}}$ stems from, the effect seems to be avoidable by choosing a suitable setting for the core supply voltage. The sweeps that have been carried out for the set of chip carriers presented in Section 3.4.2 suggest that a slightly increased setting of $V_{\mathrm{DD12D}} > 1.21\,\mathrm{V}$ can lead to a stark improvement in the first failure probabilities for the links. Further tests are needed to confirm whether a common default setting can be applied for all individual setups or if distinct target values for each hardware configuration are required. In any case, by applying the test routine that has been used for the sweeps of $V_{\mathrm{DD12D}}$, the user should be able to identify a good operating point for the specific system to be tested. Moreover, by implementing the adapted link training routine in the FPGA design, the user should be able to also decrease the probability for second failures, as the algorithm should be more robust in finding a suitable delay setting.

Since the sharp peak of the first failure probabilities can be observed on multiple different setups, which notably are being operated on varying power distribution schemes, the ASICs may be considered as a common denominator to the problematic. This sheds light on a parameter that has not been varied during the ASIC system parameter sweeps: All used HICANN-X ASICs correspond to the same tape-out version *v3*. Therefore, there is potentially little information about the process variation coupling into the observed failure probabilities. A reasonable cross-check would be to repeat the tests with a *v2* ASIC, which is guaranteed to stem from a different wafer than the currently tested chips. Additionally, a potential influence of temperature variations has not been analyzed systematically yet. All tests have been performed over the course of several days at a comparable time of day, but the exact temperature variations at the tested setups are unknown. Repeating the tests under controlled conditions in a climate chamber may be insightful.

In order to fully account for all remaining second errors, the link training procedure would be required to be updated on the ASIC side as well. The exact impact on performance is however difficult to estimate as the delay line implementations between FPGA and ASIC vary. The changes in the routine are kept minimal though, so the delay training implementation in a future chip version is expected to be more robust as well.

# 4 Conclusion and Outlook

So far...

... a concept for a multi-chip platform with up to four HICANN-X ASICs interconnected in a star-topology has been introduced. The presented system may feature 2048 neurons at an expected event transmission latency of 1 ms biological time. In commissioning the setup, a mezzanine board for the shared experiment control node has been developed. The mezzanine PCB aggregates the high-speed communication and slow control interfaces of the ASICs and passes them to the central connecting FPGA. By adapting the current experiment control design to the new FPGA platform, host-to-chip communication has been established. The correct operation of the platform has been tested successfully in single-chip communication tests.

In performing loop-back tests between FPGA and ASIC, instabilities of the high-speed links have been observed on the setup under construction. As a hardware defect of the central node permitted the investigation of these issues on the multi-chip platform, similar instabilities on the currently employed single-chip Cube systems have been examined. In the course of these studies, a stability criterion in terms of a time to fail and a failure probability given a fixed testing time has been defined. A systematized testing routine has been developed that enables the automated aggregation of statistics concerning the link failure probability.

The data gathered from the Cube systems showed that link lost events predominantly happen shortly after link initialization. Consequently, the automated clock-to-data alignment algorithm has been inspected using in-hardware logic analyzing tools. The gathered trace data yielded multiple issues with the training logic that are responsible for a poor clock-to-data alignment. Corrections to the observed errors have been proposed and implemented. The adapted training routine leads to an improvement in the probability for follow-up link losses in the event that a re-training had to be performed previously.

As the probability for a first link failure was left unchanged by the adaptation of the link training algorithm, system parameters of the ASIC receiver cells have additionally been examined. The performed sweeps of the supply voltages and bias current for the digital IO yielded a dependency of the observed link lost events on the core supply voltage. The link failure probability exhibits drastic peaks in narrow ranges of the core voltage setting that is reproducible over numerous chip carrier and setup configurations. Although the exact source of the observed peaks remains unclear at the time of thesis submission, the effect seems preventable by the user choosing a suitable operating point for the system under test. In doing so, link lost probabilities near zero can be achieved for the Cube systems.

In the future...

... the efforts in building a full multi-chip system for the BSS-2 architecture will continue. The occurred hardware defect described in this thesis highlights the vulnerability of the star network topology in case the central node hardware is sparsely available. Further upscaling work will take this into account by moving towards more disposable nodes. The integration of a dedicated routing core logic is underway as well. Moreover, other network topologies will be explored, as there are efforts pushing towards building up a mesh-structured interconnection of chips by employing dedicated routing ASICs.

A robust high-speed event communication interface is one of the central requirements for any of the mentioned upscaling pathways. As a next step towards stabilizing the high-speed links for the currently employed systems, a minimal test routine for commissioning and calibration should be deployed that allows hardware users to automatically find a good operating point for robust event transmission. However, it should be noted that a precise understanding of the observed dependency of the link lost events on certain settings of the ASIC core supply voltage is important as well. Most notably, there is currently no information about the influence of process and temperature variations on the probability for link failures. Also, there are additional configurations available for the ASIC de-serializer – for instance, the option to activate an additional chain of delay elements to recalibrate the currently set clock-to-data alignment – that could be investigated. Ultimately, the updated training algorithm is recommended to be implemented in a future revision of the HICANN-X as well. A more involved and resource consuming, but desirable upgrade to the training procedure would be to render the training logic programmable by employing an embedded microprocessor. The configuration of the delay could then be handled in software instead of being fixed by the logic embedded in the fabric.

# Acknowledgements

The work carried out in this Master 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. 720270, 785907 and 945539 (Human Brain Project, HBP) and Horizon Europe grant agreement No. 101147319 (EBRAINS 2.0).

I would like to thank Dr. Johannes Schemmel for the opportunity to contribute to the visionary work of the Electronic Vision(s) Group, as well as guiding and teaching me throughout this thesis. His encouraging and supportive attitude, especially during times when the focus of this thesis had to be shifted, was greatly appreciated. I also would like to thank Prof. Dr. Dirk Koch for conducting the second review and his advice on composing this thesis.

I further wish to express my deepest gratitude to Joscha Ilmberger for his exceptional and greatly valued efforts in supervising this thesis. In the past months, I have picked up skills and knowledge in the broadest hardware design related topics from him – ranging from PCB and FPGA design to drilling proper holes in aluminum plates.

The combined efforts of many people advising me in various contexts of this thesis are greatly appreciated: Jakob Kaiser and Philipp Spilger provided software support and helped to cast light on this black box for me. Dr. Andreas Grübel and Lars Sterzenbach offered advice and support during the PCB design process. Alexander Dobler of the electronics workshop assembled the PCBs and Sascha Braun of the mechanical workshop manufactured required mechanical components. Dan Husmann instructed me in handling the machinery of our local mechanical workshop. Dr. Sebastian Billaudelle and Yannik Stradmann offered assistance across all things hardware related. Dr. Hartmut Schmidt and Florian Fischer provided support in proofreading this thesis. For your help, I would like to express my gratefulness to you.

Special thanks to my dear colleagues of room 00.234, Jakob Huhle, Arik Küster and Carl von Randow, for the welcoming atmosphere and the well-regulated room temperature. And for the co-working power sessions late at night.

Additionally, I would like to thank all people involved in the numerous darts sessions that I was allowed to participate in. And I would like to apologize for my Grantelbartigkeit at times when the flights have been manipulated once again.

Actually, thanks to all (former) members of the Electronic Vision(s) Group for your openness and kindliness. It is a pleasure to share time and work with you.

Lastly, I would like to thank my friends and family for supporting me throughout my studies. There are many people I am grateful for in my life and this page is too short, so I will just mention a few. Mum, Dad, Johannes, Tetiana, Anita, Mariia and Amelie. I love you people.

# Experiment Environment

All data of the experiments described in this thesis have been obtained with the software state documented in Table 4.1. The commit hashes of each repository are accompanied by Gerrit changeset numbers, if the software state was ahead of the master branch.

| Repository | *git* hash | Changeset |
|---|---|---|
| bss-hw-params | be3c6c33fc3a85bf296be779c9b4cd5b96f4ae29 | |
| code-format | e718d56928d006669376fde991bb9b4605a6818a | |
| extoll-driver | fcfad13745f2434ee63834478d67b1141b62da2f | |
| fisch | 803ece36156a3792b697716fa6e48b0fc970d3b4 | 22990 |
| flange | 28e729d59df3b4ff380f84351c40d4da3086bed8 | |
| halco | 34cbdc02f6ae82c31309a1116815f6f006ab149a | |
| haldls | a53b0e897fa53fa7807113ba56af9bbd49e4f2e2 | 23378 |
| hate | 1b3eafcca8eee9bca00bafb14570e0072f731d26 | |
| hwdb | ab92e3558732a89ca87b75a08413a13d0ed212bf | |
| hxcomm | fa7fb2cdc8faa30a66309c891ff1752a16a9167f | 22989 |
| lib-boost-patches | 136c5b41cb046afe2c726aa4646928bf5190622e | |
| lib-rcf | af6fc768c1735b67c3bf189de7c1fca58e67c0e3 | |
| libnux | fc3b137384596ea5adbd5d4ee1ddfc9761a2aabc | |
| librma | 3fae4e359b8cbb2760b32046aff93a3da24a3943 | |
| logger | 73dadb3ce413c521845ef7d36f818073eee4fefa | |
| nhtl-extoll | a140dc1ae3114bda0ae08f1cd67833887fc0bd17 | |
| pywrap | 5e2af30e9593882b471d3cd02df00b93f13ff479 | |
| rant | 722edd57c9e42462a660db8a1febb0211ffad07c | |
| sctrltp | 1d854f953f7e8c8ead44406a22bb80421ca3857c | |
| visions-slurm | 8f41ea4f5bd1573d8f4623e9ed698a29f30036a3 | |
| ztl | b6745261d8bfdce44516d58d632c3c73834839d2 | |

**Table 4.1:** Software state used for all experiments described in this thesis.

Utilized third-party software is tracked with the image of the Singularity container listed in Table 4.2.

| Key | Value |
|---|---|
| path | /containers/stable/2024-04-17_1.img |
| app | dls |

**Table 4.2:** Singularity container used for all experiments described in this thesis.

# Glossary

**ASIC** *application-specific integrated circuit*

**ASIC-AB** *ASIC Adapter Board*

**AXI** *Advanced eXtensible Interface*

**BAM** *bit align machine*

**BRAM** *block random-access memory*

**BSS** *BrainScaleS neuromorphic hardware*

**BSS-2** *BrainScaleS-2 neuromorphic hardware*

**CRC** *cyclic redundancy check*

**DAC** *digital-to-analog converter*

**DDR** *double data rate*

**EEPROM** *Electrically Erasable Programmable Read-Only Memory*

**ESD** *electrostatic discharge*

**FIFO** *first-in-first-out buffer*

**FMC** *FPGA Mezzanine Card*

**FMC+** *FPGA Mezzanine Card Plus*

**FPGA** *field programmable gate array*

**FSM** *finite state machine*

**GPIO** *general purpose input/output*

**GUI** *graphical user interface*

**HDL** *hardware description language*

**HICANN-X** *High Input Count Analog Neural Network with HAGEN Extensions*

**Host ARQ** *Host Automatic Repeat-Request*

**HPC** *high pin count*

**I2C** *Inter-Integrated Circuit bus*

**IC** *integrated circuit*

**IDELAYCTRL** *tap delay value control block*

**IDELAYE2** *programmable delay primitive*

**ILA** *Integrated Logic Analyzer*

**IO** *input and output*

**IP** *intellectual property*

**ISERDESE2** *input serial-to-parallel converter*

**jBOA** *just a Bunch Of ASICs*

**JSON** *JavaScript Object Notation*

**JTAG** *Joint Test Action Group*

**L2** *Layer 2*

**LDO** *low-dropout regulator*

**LED** *light-emitting diode*

**LSB** *least significant bit*

**LVDS** *low voltage differential signaling*

**MezzaMix** *Mezzanine card for Mixed multi-chip setup evaluation*

**OSERDESE2** *output parallel-to-serial converter*

**PCB** *printed circuit board*

**PHY** *PHYsical layer*

**Playback Executor** *Playback Program Executor module for FPGA*

**PLL** *phase-locked loop*

**PVT** *process, voltage, and temperature*

**SEM** *standard error of the mean*

**SNN** *spiking neural network*

**Tcl** *Tool command language*

**UDP** *User Datagram Protocol*

**URAM** *UltraRAM*

**UT** *Universal Translator*

**VCD** *value change dump*

**VCU118** *AMD Virtex UltraScale+ FPGA VCU118 Evaluation Kit*

**XDC** *Xilinx Design Constraints*

# List of Figures

# List of Tables

# Bibliography


[1] AMD. *AMD UltraScale+ FPGAs Product Selection Guide*. XMP103. Rev. 07/15/2024 Version 2.4.1. July 2024. URL: https://docs.amd.com/v/u/en-US/ultrascale-plus-fpga-product-selection-guide.

[2] AMD. *VCU118 Evaluation Board, User Guide*. UG1224. Rev. 03/15/2023 Version 1.5. Mar. 2023. URL: https://docs.amd.com/v/u/en-US/ug1224-vcu118-eval-bd.

[3] Greg Burton. *16-Channel, DDR LVDS Interface with Per-Channel Alignment*. XAPP855. Rev. 10/13/06 Version 1.0. Xilinx, Inc. Oct. 2006. URL: https://docs.amd.com/v/u/en-US/xapp855.

[4] Stephen H Hall. *Advanced signal integrity for high-speed digital designs*. eng. Ed. by Howard L [MitwirkendeR] Heck. Includes bibliographical references and index. - Description based on print version record. Hoboken, N.J.: J. Wiley & Sons, 2009, 1 online resource (1 v.) URL: https://learning.oreilly.com/library/view/-/9780470192351/?ar.

[5] Hirose Electric Co., LTD. *CX Series USB Type-C Connector Product Catalog*. Rev. 08/01/2024. Aug. 2024. URL: https://www.hirose.com/de/product/document?clcode=CL0480-0625-0-00&productname=CX80B1-24P&series=CX&documenttype=Catalog&lang=de&documentid=D52488_en.

[6] Jakob Kaiser et al. "Emulating Dendritic Computing Paradigms on Analog Neuromorphic Hardware". In: *Neuroscience* 489 (2022), pp. 290–300. DOI: 10.1016/j.neuroscience.2021.08.013.

[7] Vitali Karasenko. "Von Neumann bottlenecks in non-von Neumann computing architectures". Dissertation. Universität Heidelberg, 2020. DOI: 10.11588/heidok.00028691. URL: http://www.ub.uni-heidelberg.de/archiv/28691.

[8] Microchip Technology Inc. *2K I2C™ Serial EEPROMs with Unique 32-bit Serial Number*. DS20005202A. Revision A (05/13). May 2013. URL: https://ww1.microchip.com/downloads/en/DeviceDoc/20005202A.pdf.

[9] Paul Müller. *Modeling and verification for a scalable neuromorphic substrate*. eng. Heidelberg, 2017, 1 Online–Ressource (204 Seiten). DOI: 10.11588/heidok.00023716. URL: https://nbn-resolving.org/urn:nbn:de:bsz:16-heidok-237168.

[10] Multi Leiterplatten GmbH. *Definierter Lagenaufbau und Impedanzen-Beispiele für Multilayer Leiterplatten*. Document accessed as of 08/22/2024. 2024. URL: https://www.multi-circuit-boards.eu/fileadmin/pdf/leiterplatten_lagenaufbau/Multi-CB_Definierter_Lagenaufbau_Impedanzen_de.pdf.

[11] Christian Pehle et al. "The BrainScaleS-2 accelerated neuromorphic system with hybrid plasticity". eng. In: *Frontiers in neuroscience* 16.Artikel-ID 795876 (2022). Gesehen am 30.06.2022, pp. 1–21. ISSN: 1662-453X. DOI: 10.3389/fnins.2022.795876. URL: https://www.frontiersin.org/article/10.3389/fnins.2022.795876.

[12] Mihai A. Petrovici. *Form vs. function. theory and models for neuronal substrates*. eng. Überarbeitete Fassung. Erstellungsdatum der Online-Ressource: 27. Juni 2016. Heidelberg, 2016, 1 Online–Ressource (405 Seiten). DOI: 10.11588/heidok.00021402. URL: https://nbn-resolving.org/urn:nbn:de:bsz:16-heidok-214026.

[13] Samtec. *HIGH-DENSITY OPEN-PIN-FIELD ARRAYS Product Guide*. F-224. Rev. 08/13/2024. Aug. 2024. URL: https://suddendocs.samtec.com/catalog_english/seam.pdf.

[14] Samtec. *HIGH-SPEED EDGE CARD SYSTEMS Product Guide*. F-224. Rev. 04/04/2024. Apr. 2024. URL: https://suddendocs.samtec.com/catalog_english/hsec8_dv.pdf.

[15] Johannes Schemmel et al. "Accelerated Analog Neuromorphic Computing". In: *Analog Circuits for Machine Learning, Current/Voltage/Temperature Sensors, and High-speed Communication: Advances in Analog Circuit Design 2021*. Springer International Publishing, 2022, pp. 83–102. ISBN: 978-3-030-91741-8. DOI: 10.1007/978-3-030-91741-8_6.

[16] Stefan Scholze et al. "A 32GBit/s communication SoC for a waferscale neuromorphic system". In: *Integration* 45.1 (2012), pp. 61–75. ISSN: 0167-9260. DOI: https://doi.org/10.1016/j.vlsi.2011.05.003. URL: https://www.sciencedirect.com/science/article/pii/S0167926011000538.

[17] Texas Instruments Incorporated. *High-Speed Interface Layout Guidelines*. SPRAAR7J. Rev. J, revised February 2023. Nov. 2018. URL: https://www.ti.com/lit/an/spraar7j/spraar7j.pdf.

[18] Texas Instruments Incorporated. *INA219 Zerø-Drift, Bidirectional Current/Power Monitor With I2C Interface*. SBOS448G. Revised December 2015. Aug. 2008. URL: https://www.ti.com/lit/ds/symlink/ina219.pdf.

[19] Texas Instruments Incorporated. *TPS22992x 5-V, 8.7-mΩ, 6-A Load Switch With Adjustable Rise Time*. SLVSFT0A. Revised December 2021. June 2021. URL: https://www.ti.com/lit/ds/symlink/tps22992.pdf.

[20] VITA. *ANSI/VITA 57.1-2019 FPGA Mezzanine Card (FMC) Standard*. Rev. 02/01/2019. American National Standards Institute, Inc. Feb. 2019. ISBN: 978-1-948739-06-1.

[21] VITA. *ANSI/VITA 57.4-2018 FPGA Mezzanine Card Plus (FMC+) Standard*. Rev. 08/24/2018. American National Standards Institute, Inc. Aug. 2018. ISBN: 978-1-948739-04-7.

[22] Alexander Weiler, Alexander Pakosta, and Ankur Verma. *High-Speed Layout Guidelines*. SCAA082A. Rev. A, revised August 2017. Texas Instruments Incorporated. Nov. 2006. URL: https://www.ti.com/lit/an/scaa082a/scaa082a.pdf.

[23] Wuerth. *WR-MPC3 - 3.00MM MALE DUAL ROW RIGHT ANGLE HEADER WITH PLASTIC PEGS Datasheet*. Rev. 03/22/2016. Mar. 2016. URL: https://www.we-online.com/components/products/datasheet/66202021022.pdf.

[24] Xilinx, Inc. *7 Series FPGAs SelectIO Resources User Guide*. UG471. Rev. 05/08/18 Version 1.10. May 2018. URL: https://docs.amd.com/v/u/en-US/ug471_7Series_SelectIO.

[25] Xilinx, Inc. *Kintex-7 FPGAs Data Sheet: DC and AC Switching Characteristics (DS182)*. DS182. Rev. 03/26/2021 Version 2.19. Mar. 2021. URL: https://docs.amd.com/v/u/en-US/ds182_Kintex_7_Data_Sheet.

[26] Xilinx, Inc. *System Integrated Logic Analyzer v1.1 Product Guide*. PG261. Rev. 02/04/2021 Version 1.0. Feb. 2021. URL: https://docs.amd.com/v/u/en-US/pg261-system-ila.

[27] Xilinx, Inc. *Vivado Design Suite 7 Series FPGA and Zynq-7000 SoC Libraries Guide (UG953)*. UG953. Rev. 10/22/2021 v2021.2. Oct. 2021. URL: https://docs.amd.com/r/2021.2-English/ug953-vivado-7series-libraries/IDELAYCTRL.

## Statement of Originality (Erklärung):

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

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

Heidelberg, August 30, 2024 ……..

(signature)