Alexander Schmidt

Design und Charakterisierung einer Routing-Schnittstelle für Neuromorphe Hardware

Bachelorarbeit

Fakultät für Physik und Astronomie
Universität Heidelberg

Design und Charakterisierung einer Routing-Schnittstelle für Neuromorphe Hardware

Bachelorarbeit

in Physik

eingereicht
am 7. April 2017
von

Alexander Schmidt

geboren
am 1. Dezember 1993
in Ludwigsburg

durchgeführt
am

Kirchhoff-Institut für Physik
Ruprecht-Karls-Universität Heidelberg

unter Aufsicht von

Prof. Dr. Karlheinz Meier

Bemerkungen zur 2. Ausgabe

Im Vergleich zur eingereichten Bachelorarbeit wurden die Histogramme in Abb. 3.8 korrigiert und dementsprechend der beschreibende Text angepasst. Die Anzahl der Titelseiten wurde reduziert. Die Danksagung wurde ergänzt.

18. April 2017

Zusammenfassung

Design und Charakterisierung einer Routing-Schnittstelle für neuromorphe Hardware

Als Teil des HICANN-DLS-SR-Chips kommender Generation wurde im Zuge dieser Arbeit eine Routing-Schnittstelle zwischen der SPL1-Ebene (internes Routing) und der L2-Ebene (Ansteuerung der physischen Ports) des Chips entwickelt. Die Schnittstelle ist in der Lage, durch Lastausgleich die volle Bandbreite der L2-Links, welche das Bottleneck in der Übertragung darstellen, auszunutzen. Mittels Zeitstempeln ist es außerdem möglich, den Jitter der L2-Übertragung und des Routing-Systems selbst signifikant zu reduzieren, auf Kosten erhöhter mittleren Latenz der Übertragung. Mit einer frei wählbaren Soll-Latenz Δt kann der Anwender so zwischen verringerter Latenz und verringertem Jitter wählen.

Abstract

Desgin and characterisation of a routing interface for neuromorphic hardware

As part of the HICANN-DLS-SR chip of upcoming generation we developed a routing interface between the SPL1 layer (internal routing) and the L2 layer (control layer for physical ports) of the chip. The interface enables load balancing across the L2 links, which pose the bottleneck in transmission, in order to use their full bandwidth. Using time stamps it is also possible to significantly reduce the jitter introduced by the L2 transmission and the routing system itself at the cost of increased average latency of transmission. Using a free parameter Δt for target latency the user can shift between lower latency and lower jitter.

Inhaltsverzeichnis

1 Einführung **3**
1.1 Neuronale Netze . . . 3
1.2 Neuromorphe Hardware . . . 4
1.3 Die HICANN-Chips . . . 4
1.4 Ziel dieser Arbeit . . . 5

2 Beschreibung des Routing-Designs **7**
2.1 Anforderungen und Spezifikation . . . 7
2.2 Struktur der Nachrichten . . . 8
2.3 Die SPL1-to-L2-Schnittstelle . . . 9
2.4 Die L2-to-SPL1-Schnittstelle . . . 10
2.5 Handshake-Prinzip . . . 12

3 Analyse und Charakterisierung der Schnittstelle **13**
3.1 Verifikation des Designs . . . 13
3.2 Bandbreite und Droprate . . . 15
3.3 Lastausgleich . . . 19
3.4 Droprate an den L2-Links . . . 21
3.5 Latenz und Jitter . . . 23
3.6 vanRossum-Distanz und Korrelationsmatrix . . . 29

4 Testergebnisse und Diskussion **32**
4.1 Durchsatz und Länge der Input-Buffer . . . 32
4.2 Lastausgleich und Korrelationen . . . 33
4.3 Droprate an den L2-Links . . . 33
4.4 Die Korrekte Wahl von Δt . . . 33
4.5 Kalibration von Δt . . . 34
4.6 Länge des Zeitstempels und des Empfangsbuffers . . . 34
4.7 Zusammenfassung . . . 35

5 Ausblick **37**
5.1 Implementationsaspekte . . . 37
5.2 Varianten der designten Module . . . 38
5.3 Die weitere Entwicklung . . . 39

Kenngrößen **41**

Begriffe und Abkürzungen **41**

Kapitel 1

Einführung

Die in dieser Arbeit entwickelte Routing-Schnittstelle ist Teil des HICANN-DLS-SR-Chips, welcher sich zur Zeit in Entwicklung befindet. Als Teil der High Input Count Artificial Neural Network (HICANN)-Serie ermöglicht der HICANN-DLS-SR-Chip große, schnelle Neuronale Netze (NN) auf Hardware-Ebene. In den folgenden Abschnitten wird ein kurzer Überblick über Neuronale Netze und Neuromorphe Hardware im Allgemeinen, sowie die HICANN-Serie und den HICANN-DLS-SR im Speziellen gegeben.

1.1 Neuronale Netze

Das Gehirn ist eines der kompliziertesten Organe des menschlichen Körpers (oder allgemeiner: von Wirbeltieren). Als solches ist es deutlich besser untersucht, jedoch gleichzeitig deutlich weniger verstanden als die meisten anderen Organe. Dabei ist es im Wesentlichen aus gleichartigen Bausteinen, den Neuronen, aufgebaut. Die konzeptuelle Grundstruktur des Gehirns ist ein neuronales Netz (NN). Ein solches neuronales Netz besteht aus einzelnen Neuronen, seinen Grundbausteinen, welche über Synapsen verbunden sind. In einem natürlichen neuronalen Netz, wie es im Gehirn vorkommt, handelt es sich bei den Neuronen um Nervenzellen.

Die Übertragung und Verarbeitung von Information in einem neuronalen Netz geschieht auf eine universelle Art und Weise. An der Zellmembran des Neurons gehen elektrische Signale von Synapsen anderer Neuronen ein. Diese werden gewichtet summiert, wenn sie einen Schwellwert überschreiten entsteht ein charakteristischer Ausschlag des Membranpotentials der Zelle, das Aktionspotential („Spike“). Dieser Spike propagiert zu den Synapsen des Neurons und liefert so elektrische Signale an die Membranen anderer Neuronen.

Inspiriert von natürlichen neuronalen Netzen wurde das Konzept der künstlichen neuronalen Netze entwickelt. Diese abstrahieren die komplexe elektrochemische Struktur der Nervenzellen und Synapsen. Das Ziel künstlicher neuronaler Netze ist zum Einen, die Funktion und Evolution des Gehirns in einer kontrollierten Umgebung zu untersuchen. Zum Anderen sind Neuronale Netze die Grundlage verschiedener intelligenter Softwarealgorithmen und Kandidat zukünftiger neuartiger Computerarchitektur.

1.2 Neuromorphe Hardware

Die verbreitetste Realisierung künstlicher neuronaler Netze ist die Simulation in Software. Das hat allerdings mehrere Nachteile. Zum Einen sind Neuronale Netze in Software auf Modelle beschränkt, die sowohl in Zeit, als auch in den Potentialen diskret sind. Zum Anderen arbeiten Mikroprozessoren sequentiell und sind damit sehr ineffizient in der Simulation der inhärent parallelen neuronalen Netze.

Realisiert man Neuronale Netze direkt in Hardware, so ist die parallele Struktur intrinsisch gegeben. So ist es möglich, das Netzwerk schneller als in Software, sogar schneller als in biologischen Systemen zu betreiben. Mit digitaler Hardware können diskrete, mit analoger sogar kontinuierliche Modelle neuronaler Netze abgebildet werden.

Das menschliche Gehirn ist modernen Mikroprozessoren zwar in Bezug auf rohe Rechenleistung und Präzision unterlegen. Dafür ist es allerdings sehr gut darin, eine Vielzahl von Aufgaben gleichzeitig zu erledigen, aus einer Fülle von Sinneseindrücken relevante Information zu extrahieren. Und auch wenn in den letzten Jahrzehnten vermehrt die Entwicklung lernfähiger künstlicher Intelligenz vorangetrieben wurde ist die Lernfähigkeit des Gehirns unerreicht. Aus diesem Grund kann Neuromorphe Hardware potentiell Teil zukünftiger Computerarchitektur werden.

1.3 Die HICANN-Chips

Die High Input Count Artificial Neural Network (HICANN)-Chips sind teilanaloge Application Specific Integrated Circuit (ASIC)s, die ein Modell neuraler Netze realisieren, welches sowohl in Zeit, als ich in Amplitude kontinuierlich arbeitet.

Die ursprünglichen HICANN-Chips auf 180nm-Basis [8], sowie die aktuelle Iteration HICANN-DLS auf 65nm-Basis [7] sind auf Wafer Scale Integration (die Verwendung vollständiger Siliziumwaver als Makrochips) ausgelegt. Zusätzlich befindet sich nun der HICANN-DLS-SR-Chip als Einzelchip in Entwicklung und soll als Nachfolger des Spikey-Chips [6] dienen. Die im Zuge dieser Arbeit entwickelte Schnittstelle ist Teil des HICANN-DLS-SR-Chips.

Auch wenn der HICANN-DLS-SR als Einzelchip entwickelt wird, soll die Möglichkeit bestehen, mit anderen HICANN-Chips oder FPGA-Chips zu kommunizieren, etwa um größere Neuronale Netze zu ermöglichen.

Der HICANN-DLS-SR-Chip besteht wie seine Vorgänger aus einem Analogteil, in dem sich die Neuronen befinden und einem Digitalteil, in dem sich unter Anderem die Routing-Logik befindet. Abb. **??** stellt das Routing-System schematisch dar. Spike-Daten werden zunächst in der SPL1-Ebene in einer Merger-Matrix serialisiert. Falls nun ein Spike an eine Adresse auf einem anderen Chip adressiert ist, so wird es über die SPL1 <-> L2 Routing-Schnittstelle an die L2-Ebene weitergegeben. Diese bedient schließlich die physischen Ports des Chips. Nach aktuellem Entwicklungsstand erfolgt die Verbindung zu anderen HICANN-DLS-SR-Chips immer über einen zusätzlichen FPGA-Chip (vgl. Abb. 1.2).

Da die HICANN-Hardware im Vergleich zu biologischen Systemen um einen Faktor von $\varepsilon \approx 1000$ schneller arbeitet, sind hier große Übertragungsraten nötig. So ist bei einer typischen biologischen Spikerate von $r_{bio} = 10Hz$ pro Neuron und 512 Neuronen eine Übertragungsrate von $r_{Hardware} = \varepsilon n_n r_{bio} = 5,12MHz$ nötig, bei einer

Taktfrequenz des Chips von 250 MHz. Aus diesem Grund ist ein schnelles, robustes Routing-System notwendig.

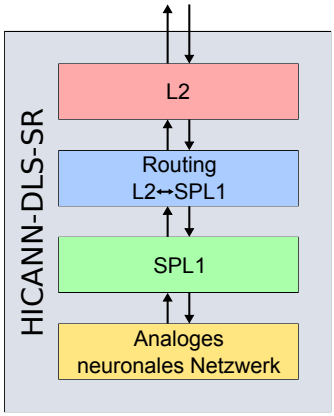

Abbildung 1.1: Konzeptionelle Struktur des Spike-Routings auf dem HICANN-DLS-SR. Die Neuronen befinden sich im analogen neuronalen Netzwerk. Die SPL1-Ebene serialisiert Spike-Daten in einer Merger-Baumstruktur. Die L2-Ebene bedient die Ports, mit denen der Chip nach außen kommuniziert. Ziel dieser Arbeit ist die Entwicklung einer Routing-Schnittstelle zwischen SPL1 und L2.

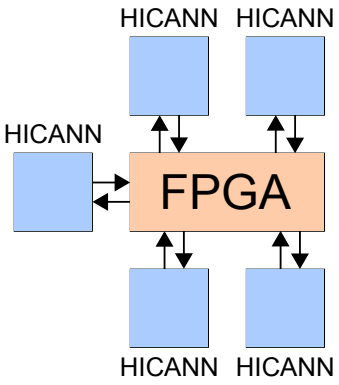

Abbildung 1.2: Prinzipielle Kommunikationsstruktur zwischen mehreren HICANN-DLS-SR-Chips. Nach aktuellem Entwicklungsstand muss zwischen mehreren HICANN-Chips immer ein FPGA geschaltet sein.

1.4 Ziel dieser Arbeit

Die externe Kommunikation des HICANN-DLS-SR-Chips wird auf L2-Ebene (beschrieben in [2]) organisiert, wo Nachrichten serialisiert und versandt bzw. empfangen und auf Fehler geprüft werden. Das interne Routing von Spikes wird in der SPL1-Ebene organisiert, wo die Spikes der Neuronen in einer Merger-Matrix serialisiert werden. Das Ziel dieser Arbeit ist, einer Schnittstelle zwischen L2 und SPL1 zu entwickeln und zu charakterisieren.

Bei allen Designentscheidungen muss auf 4 entscheidende Parameter geachtet werden: den Hardware-Aufwand, die Übertragungsrate, die Latenz und deren Schwankungen (Jitter).

Da es sich bei den HICANN-Chips um ASICs handelt, entspricht jeder Mehraufwand an Hardware einer größeren Chipfläche bzw. Einschnitten in anderen Bereichen des Chips. Die Übertragungsrate ist direkt proportional zu den maximalen Spike-Raten der Neuronen, die unterstützt werden. Eine geringere Latenz ermöglicht die Durchführung von stark timing-orientierten Experimenten (siehe z.B. [1]). Ein geringerer Jitter erhöht direkt die zeitliche Präzision von Experimenten.

Im Folgenden wird also zunächst das erarbeitete Design der Schnittstelle vorgestellt. Im Anschluss werden die vorgenommenen Tests der Module im Simulator und

ihre Ergebnisse zusammengefasst. Abschließend werden aus den Testergebnissen die geeignete Wahl der freien Parameter und Methoden für ihre Kalibration abgeleitet.

Kapitel 2

Beschreibung des Routing-Designs

Das entwickelte Design besteht aus einem Sender-Modul „SPL1-to-L2“, welches die von den SPL1-Links eingehenden Spike-Nachrichten mit einem Zeitstempel versieht und auf die ausgehenden L2-Links verteilt, sowie aus einem Empfänger-Modul „L2-to-SPL1“, welches die an den L2-Links ankommenden Spike-Nachrichten entsprechend ihrem Zeitstempel ordnet und auf die SPL1-Links verteilt.

Die folgenden Abschnitte beschreiben das Design im Detail. Um möglichen Änderungen im Design des Chips entgegen zu kommen, ist das Routing-Design so weit wie möglich parametrisiert, sodass schnelle Anpassungen einfach zu realisieren sind. Die im folgenden präsentierten Zahlen sind repräsentativ für den aktuellen Stand der Entwicklung.

2.1 Anforderungen und Spezifikation

Im folgenden werden die charakteristischen Rahmenparameter und Anforderungen an das Design vorgestellt.

Der Analogteil des Chips beinhaltet bis zu 512 Neuronen. Diese werden in sogenannten Compartments mit bis zu 256 Synapsen organisiert. Die genauere interne Struktur des Netzwerks ist für das Routing allerdings unerheblich. Von der SPL1-Ebene gehen letztlich $n_1 = 4$ SPL1-Links aus, welche die Spikes der Neuronen serialisiert übermitteln. Auf L2-Ebene existieren $n_2 = 8$ physische Links, die den Chip verlassen und die Kommunikation mit anderen Chips ermöglichen. Das hier vorgestellte Routing-Design bildet die Schnittstelle zwischen den SPL1- und L2-Links.

Das aktuelle Design stellt die Grundlage für die Kommunikation zwischen bis zu $n_1 + 1$ Einzelchips dar, prinzipiell ist sogar die Kommunikation von bis zu $n_2 + 1$ Chips möglich. Im Zuge dieser Arbeit wird der Spezialfall der Kommunikation zwischen genau 2 Chips charakterisiert.

Die Taktfrequenz des Chips beträgt $f_T = 250MHz$, was einer Taktdauer von $T_T = 4ns$ entspricht. Die physischen Links der L2-Ebene haben jeweils die Bandbreite $f_2 = 1GHz$, müssen aber die ausgehenden Spike-Nachrichten der Länge a jeweils serialisieren. Aus diesem Grund können die L2-Links Nachrichten im Abstand von

$d = 20$ Takten senden bzw. empfangen. Im Gegensatz dazu können die L1-Links Nachrichten jeden Takt senden bzw. empfangen. Es liegt also ein eindeutiges Bottleneck bei den L2-Links vor, die Nachrichten maximal mit der Rate

$$r_2^{max} = \frac{n_2}{d} = \frac{8}{20}\frac{Nachrichten}{Takt} = \frac{2}{5}\frac{Nachrichten}{Takt} \tag{2.1}$$

verarbeiten, gegenüber der Rate der L1-Links von bis zu

$$r_1^{max} = n_1\frac{Nachrichten}{Takt} = 4\frac{Nachrichten}{Takt}. \tag{2.2}$$

Tatsächlich kann also sogar ein einziger L1-Link alle L2-Links voll auslasten. Um die Übertragungsrate zu maximieren ist es also notwendig, in SPL1-to-L2-Richtung Lastausgleich (Load Balancing) zu betreiben.

Das zweite Ziel ist, eine geringe, aber einheitliche Latenz zu gewährleisten. Da die L2-Links nur in größeren Abständen Nachrichten schicken können, treten hier leichte Schwankungen (Jitter) in der Latenz auf, zusätzlich können Bufferstufen auf dem Chip die Latenz um wenige Takte verändern und so weiteren Jitter verursachen. Aus diesem Grund wird als Teil der Nachricht ein Zeitstempel verschickt, der es der Empfänger-Seite (L2-to-SPL1) ermöglicht, die Nachrichten in korrekter Reihenfolge und mit konstanter Latenz Δt an die SPL1-Links auszuliefern. Um die Latenz, aber auch den Hardware-Aufwand, insgesamt zu reduzieren werden innerhalb der hier Designten Module keine Bufferstufen tiefer als $l = 1$ eingesetzt.

2.2 Struktur der Nachrichten

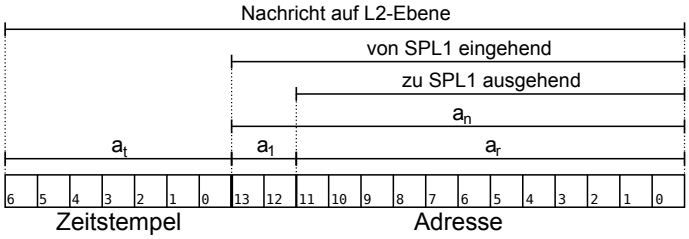

Abbildung 2.1: Struktur einer Spike-Nachricht. An den eingehenden SPL1-Links bestehen Nachrichten aus der Zieladresse. An den L2-Links bestehen Nachrichten aus der Adresse und dem Zeitstempel. An den ausgehenden SPL1-Links bestehen Nachrichten noch aus dem Restteil der Adresse ohne die Adresse des SPL1-Links

Eine Nachricht, die durch die Schnittstelle transportiert wird, entspricht einem Spike, der an ein Zielneuron gesendet wird. Die einzige relevante Information ist dabei die Zieladresse.

Die Adresse hat die Länge $a_n = 14$. Sie beinhaltet die Adresse des SPL1-Links der Länge $a_1 = log_2(n_1) = 2$. Die restlichen $a_r = 12$ Bit der Adresse bestehen aus Busadressen und der Adresse des einzelnen Zielneurons. Die genaue Struktur dieser restlichen Bits ist allerdings unerheblich für das hier erarbeitete Design.

Am Eingang des SPL1-to-L2-Moduls wird zusätzlich ein Zeitstempel der Länge a_t angeheftet. Für die im Zuge dieser Arbeit durchgeführten Tests wird $a_t = 16$ gesetzt, um Fehler durch Integer-Overflow auszuschließen. Im Laufe dieser Arbeit wird dann ein geeigneter Wert für die Implementation ermittelt. Um den Hardwareaufwand zu reduzieren, wird der Zeitstempel im realen Chip so kurz wie möglich gehalten, aus den durchgeführten Tests ergibt sich $a_t = 8$ als realistischer Wert. Insgesamt hat die Nachricht dann die Länge

$$a = a_t + a_n = 8 + 14 = 22. \tag{2.3}$$

Abb. 2.1 veranschaulicht den Aufbau einer vollständigen Nachricht, wie sie durch die L2-Ebene transportiert wird. Am L2-to-SPL1-Modul des empfangenden HICANN-Chips kommt die Nachricht nun in dieser Form an. Zunächst wird hier der Zeitstempel, im Anschluss auch der SPL1-Anteil der Adresse entfernt, sodass an den SPL1-Links schließlich eine Nachricht der Länge $a_r = 12$ ankommt.

2.3 Die SPL1-to-L2-Schnittstelle

An den SPL1-Links gehen Spike-Nachrichten ein, jeweils versehen mit der Zieladresse. Eine eingehende Nachricht wird sofort mit einen Zeitstempel versehen. Um die Bandbreite der L2-Links möglichst komplett auszunutzen folgt ein Lastausgleich-System, das die Nachrichten auf die Links verteilt.

Abb. 2.2 zeigt den schematischen Aufbau des SPL1-to-L2-Moduls. Zunächst werden die eingehenden Nachrichten der Länge a_n mit einem Zeitstempel versehen, der den letzten a_t Bit der Systemzeit entspricht. Die Systemzeit wird dabei von einem anderen Modul synchronisiert und ist nicht Teil dieser Arbeit. Damit hat die Nachricht die Länge a erreicht, mit der sie auf L2-Ebene übertragen wird.

Da selbst ein einzelner L1-Link die L2-Links vollständig auslasten kann, werden die mit dem Zeitstempel versehenen Nachrichten serialisiert. Dazu werden die parallel in $n_1 = 4$ Link eingehenden Nachrichten in einer Baumstruktur nach dem Prinzip von Mergesort (vorgestellt z.B. in [3, Kap. 5.2.5]) zusammengeführt. Priorität hat immer die ältere Nachricht. So ist garantiert, dass die Nachrichten nach der Serialisierung weiterhin zeitlich geordnet sind. Bei gleich alten Nachrichten entscheidet ein globales Prioritätsbit, welches taktweise zwischen 0 und 1 wechselt. Da das primäre Prioritätsmerkmal der Zeitstempel ist, ist hier keine komplexere Arbiter-Struktur nötig.

Im Anschluss an die Serialisierung werden die Nachrichten durch einen Arbiter auf die n_2 L2-Links verteilt. Dabei hat taktweise wechselnd einer der Links Priorität, die Nachricht geht an den (vom priorisierten ausgehend zyklisch betrachtet) ersten freien L2-Link.

Der Hardwareaufwand des SPL1-to-L2-Moduls kann durch die Anzahl ν der verwendeten Register abgeschätzt werden. Es ergibt sich ein Wert von (vgl. Abb. 2.2)

$$\nu_{SPL1-to-L2} = a(2n_1 - 1 + n_2) = 22 \cdot (7 + 8) = 330 \tag{2.4}$$

Registern für das vorgeschlagene Design.

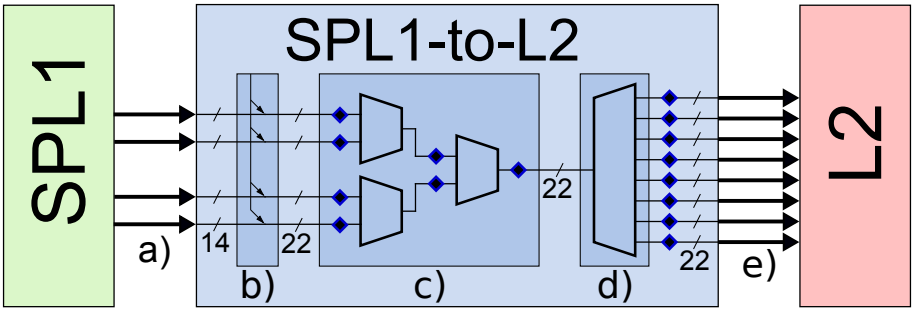

Abbildung 2.2: Schematischer Aufbau des SPL1-to-L2-Moduls. Dunkelblau markiert: Registerstufen. a) Die Nachrichten gehen an den $n_1 = 4$ SPL1-Links ein. b) An eine eingegangene Nachricht wird sofort ein Zeitstmpel angeheftet. c) Die Nachrichten werden zeitgeordnet serialisiert. d) Ein Arbiter verteilt die Nachrichten an einen der freien L2-Links. e) Die Nachrichten werden über die $n_2 = 8$ L2-Links verschickt.

2.4 Die L2-to-SPL1-Schnittstelle

Die von L2-Seite ankommende Nachrichten werden nun zurückgehalten, bis die Soll-Latenz erreicht ist und anschließend entsprechend der Zieladresse an einen der SPL1-Links ausgeliefert.

Abb. 2.3 zeigt das L2-to-SPL1-Modul. Eine an einem der L2-Links ankommende Nachricht mit Zeitstempel $t_{timestamp}$ wird also zunächst verzögert, bis die durch Δt festgelegte Latenz erreicht ist, also bis zum Zeitpunkt

$$t = t_{timestamp} + \Delta t. \tag{2.5}$$

Damit ist garantiert, dass keine Nachricht zu früh ankommt, also die festgelegte Latenz unterschreitet. Da die Nachrichten auf SPL1-to-L2-Seite zeitlich sortiert werden ist außerdem garantiert, dass die Nachrichten in der richtigen Reihenfolge ankommen. Abhängig von Δt und der Auslastung der L2-Links ist es allerdings möglich, dass vereinzelte Nachrichten verspätet ankommen.

Δt ist zunächst ein freier Parameter. Wird $\Delta t = 0$ gesetzt, so wird kein Zeitabgleich vorgenommen, die Nachrichten werden sofort ausgeliefert. Wird Δt nun erhöht, werden immer mehr Nachrichten verzögert. Das erhöht die mittlere Latenz, verringert aber den Jitter.

Es muss beachtet werden, dass Δt nur die Latenz zwischen Anheften des Zeitstempels und dem Zeitabgleich festlegt. Entsprechend wird auch nur der Anteil des Jitters reduziert, der in diesem Bereich entsteht. Das beinhaltet das Load Balancing im Sendermodul und die L2-Links, zwei große Quellen von Jitter. Der Jitter, der Im L2-to-SPL1-Modul nach dem Zeitabgleich entsteht wird nicht reduziert. Ebensowenig wird der Jitter reduziert, der außerhalb des Routing-Systems entsteht, etwa im SPL1-Merger-Tree.

Nach dem Zeitabgleich wird der Zeitstempel der Nachricht entfernt, um den Hardware-Aufwand zu reduzieren. Im Anschluss wird die Nachricht über einen Demultiplexer entsprechend den ersten $a_1 = log_2(n_1)$ Bits der Zieladresse einem der SPL1-Links zugeordnet. Zu diesem Zeitpunkt werden auch diese ersten a_1 Bits der Zieladresse entfernt. Es entsteht nun eine Verbindungsmatrix von $n_2 n_1$ Verbindungsknoten.

Die n_2 Knoten der Verbindungsmatrix, die der selben SPL1-Adresse entsprechen werden nun über einen Arbiter zusammengeführt. Die Funktionsweise dieser n_1 Arbiter entspricht dem einzelnen Arbiter auf der SPL1-to-L2-Seite. Schließlich werden die Nachrichten an die entsprechenden SPL1-Links ausgeliefert.

Die Verbindungsmatrix verursacht den größten Hardwareaufwand im gesamten Design, insgesamt $n_2 n_1 a_r = 384$ Register. Der Gesamtaufwand des vorgeschlagenen Designs für das L2-to-SPL1-Modul ist dann (vgl. Abb. 2.3)

$$\nu_{L2-to-SPL1} = n_2 a + n_2 a_n + n_1 n_2 a_r + n_1 a_r = 8 \cdot 22 + 8 \cdot 14 + 8 \cdot 4 \cdot 12 + 4 \cdot 12 = 720 \quad (2.6)$$

Register.

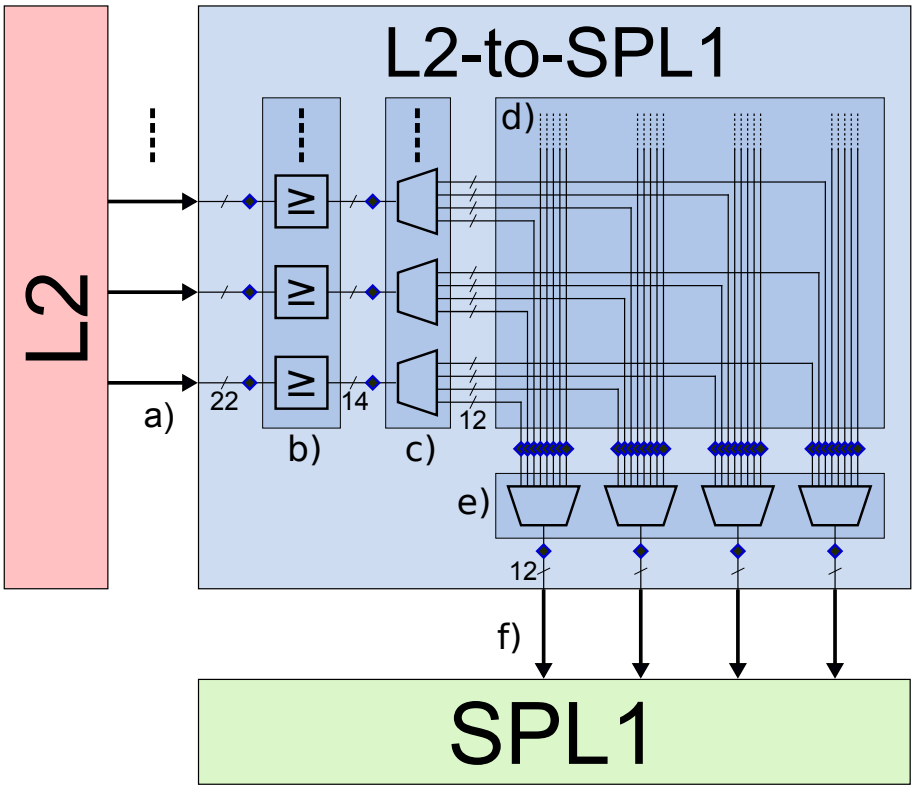

Abbildung 2.3: Schematischer Aufbau des L2-to-SPL1-Moduls. Dunkelblau markiert: Registerstufen. a) Nachrichten kommen an den $n_2 = 8$ L2-Links an. b) Die Nachrichten werden zurückgehalten, bis die vorgegebene Latenz Δt erreicht ist. Danach wird der Zeitstempel entfernt. c) Die Nachrichten werden entsprechend ihrer Zieladresse weitergeleitet. d) Die adressierten Nachrichten in der Verbindungsmatrix haben nur noch $a_r = 12$ Bit. e) Für jeden SPL1-Link existiert ein Arbiter, welcher aus den bis zu n_2 Nachrichten in der Verbindungsmatrix eine auswählt. f) Die Nachrichten werden den $n_1 = 4$ SPL1-Links übergeben

2.5 Handshake-Prinzip

Das SPL1-to-L2-Modul und das L2-to-SPL1-Modul werden an ihren Ein-und Ausgängen über ein einheitliches Handshake-Interface bedient. Abb. 2.4 veranschaulicht dieses Interface.

Der Handshake stellt sicher, dass an den Ein- und Ausgängen der Module keine Nachrichten verloren gehen können. Das führt dazu, dass sich die Nachrichten innerhalb des Moduls zum Eingang hin aufstauen. Das ist im SPL1-to-L2-Modul der Fall, wo sich die Nachrichten von den langsamen L2-Links hin zum SPL1-Eingang aufstauen und im L2-to-SPL1-Modul, wo die Nachrichten auf den Zeitabgleich warten müssen.

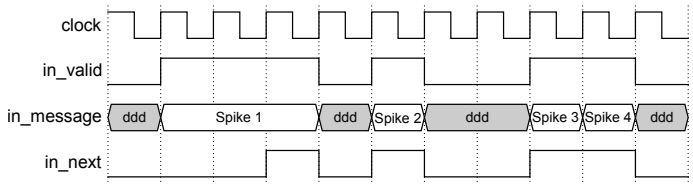

Abbildung 2.4: Beispiel für Kommunikation mit Handshake. Eingänge bestehen aus $in\_message$, in_valid und in_next, Ausgänge entsprechend aus $out\_message$, out_valid und out_next. Liegt $valid$ auf 1, so ist die angelegte Nachricht $message$ gültig. $valid$ muss so lange auf 1 gehalten werden, bis von der Gegenseite $next$ auf 1 gesetzt wird. $next$ signalisiert, dass die Nachricht erhalten wurde. Im Nächsten Takt muss dann entweder $valid$ auf 0 oder aber eine neue Nachricht anliegen. Vereinfachte Variante des Handshake-Interfaces in [4, S. 13]

Kapitel 3

Analyse und Charakterisierung der Schnittstelle

Im folgenden Kapitel wird zum Einen das erwartete Verhalten des Moduls analysiert. Zum Anderen werden Tests in Form von Simulationen durchgeführt, um das tatsächliche Verhalten mit dem erwarteten zu vergleichen.

Um das Design der Schnittstelle zu testen, werden die Module in Software simuliert. Um Latenz und Jitter, Durchsatz und Droprate des Systems zu untersuchen, wird dabei das empfangende L2-to-SPL1-Modul hinter das sendende SPL1-to-L2-Modul geschaltet und die Übertragung von Neuron zu Neuron beobachtet.

Die Testbench schaltet das SPL1-to-L2-Modul und das L2-to-SPL1-Modul hintereinander, wie in Abb. 3.1 gezeigt. Vor die SPL1-Eingänge des sendenden Moduls wird jeweils eine Input-Queue mit maximaler Tiefe l_{Ein} geschaltet, um kürzere Perioden hoher Spikerate bedienen zu können. Sobald diese Queue voll ist, gehen neu ankommende Spikes verloren.

Zwischen dem sendenden und empfangenden Modul werden die L2-Links simuliert. Die Übertragung von Senderseite zu Empfängerseite erfolgt innerhalb von einem Takt, im Anschluss muss die Verbindung für $d = 20$ Takte warten bis erneut eine Nachricht übertragen werden kann. Wird eine Nachricht gesendet, während die Empfangs-Queue der Tiefe $l_{Empfang}$ voll ist, so wird die ankommende Nachricht gedropt, geht also verloren.

Die an den Eingang des Test-Setups angelegten Spikes werden mit Zeitpunkt der Eingabe und der Zieladresse gespeichert, ebenso werden die am Ausgang ankommenden Spikes mit dem Ankunftszeitpunkt gespeichert. Zusätzlich wird jeder Verlust eines Spikes durch Overflow einer der Queues mit seinem Zeitpunkt gespeichert.

3.1 Verifikation des Designs

Zunächst wird das Routing-System verifiziert, also auf offensichtliche Fehler überprüft.

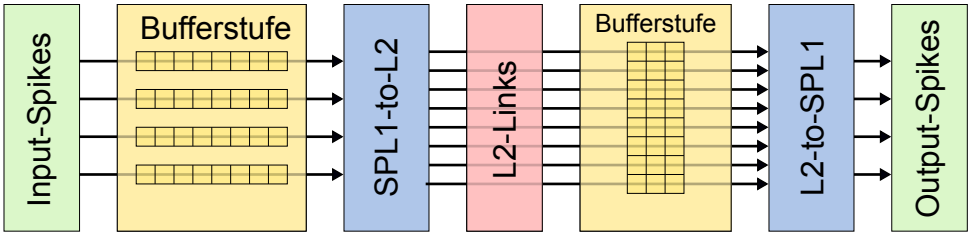

Abbildung 3.1: Schematischer Aufbau der Testbench. Vor das SPL1-to-L2-Modul wird ein Eingangsbuffer der Tiefe l_{Ein} geschaltet, in diesen werden die Eingabespikes geschrieben. Zwischen dem SPL1-to-L2-Modul und dem L2-to-SPL1-Modul werden die L2-Links simuliert, gefolgt von einem Buffer der Tiefe $l_{Empfang}$. Am SPL1-Ausgang des L2-to-SPL1-Moduls werden die Spikes schließlich ohne weitere Verzögerung ausgelesen.

An den SPL1-Links der Senderseite werden dazu Spikes eingegeben, diese propagieren durch das System und werden an den SPL1-Links der Empfängerseite wieder ausgelesen. Das System ist in der Lage, Spikes von beliebigen Sender-Neuronen an beliebige Empfänger-Neuronen zu senden. Für eine vereinfachte Analyse ist hier der Sender gleich dem Empfänger („Identity-Test“). Das Design wird verifiziert mit willkürlicher, aber realistischer Länge $l_{Ein} = 10$ des Input-Buffers, $l_{Empfang} = 1$, sowie den bereits genannten Parametern. Dabei wird das System für verschiedene Werte von Δt getestet.

Für die Verifikation wird ein vereinfachtes Modell der Neuronen auf dem Chip gewählt, mit $n_n = 256$ Neuronen, die einzeln adressiert werden können. Entsprechend der SPL1-Links werden die Neuronen gruppiert zu $n_1 = 4$ Gruppen der Größe $n_G = 64$. Die Adresse hat in diesem vereinfachten Modell die Länge $a_n = a_1 + a_G = log_2(n_1) + log_2(n_G) = 2 + 6 = 8$.

Als Testfall werden für jedes Neuron zufällige Eingangsdaten produziert. Dazu wird für jeden SPL1-Link, beginnend mit Zeitpunkt $t = 0$, der zeitliche Abstand t_d des folgenden Spikes berechnet durch

$$t_d = 1 + runden\left(\frac{z}{n_G}\right) Takte, \tag{3.1}$$

wobei z eine exponentiell verteilte Zufallszahl mit Mittelwert t_m ist. Damit wird für jeden SPL1-Link ein Poisson-Prozess der mittleren Spikerate („Intensität“) $r_{Link} = \frac{n_G}{t_m}$ genähert. Die Näherung ist dabei technisch limitiert durch den zeitlich diskreten Takt.

Die zum Spike gehörige Adresse des Neurons in der Gruppe wird gleichverteilt zufällig bestimmt. Da die Poissonverteilung additiv ist (siehe [5, S. 86]), wird so auch für jedes einzelne Neuron ein Poisson-Prozess der Intensität $r_n = \frac{1}{t_m}$ genähert. Ebenso verhält sich der Gesamt-Input aller n_1 SPL1-Links wie ein Poisson-Prozess der Intensität $r_1 = \frac{n_1 n_G}{t_m} = \frac{n_n}{t_m}$. Dabei ist zu beachten, dass diese Näherung vor Allem für große mittlere Spikeabstände t_m gut ist. Da die Charakterisierung allerdings vor Allem für größere Input-Raten r (kleinere t_m) interessant ist, wird im Folgenden die exakte mittlere Input-Rate $r_{Link} = \frac{1}{1+\frac{t_m}{n_G}}$ bzw.

$$r_1 = n_1 r_{Link} = \frac{n_1}{1 + \frac{t_m}{n_G}} \tag{3.2}$$

verwendet.

Zur Verifikation werden die offensichtlichen Sanity-Checks durchgeführt. Die Anzahl m_a der ausgehenden Spikes und die Anzahl m_d der gedropten Spikes sollten sich zur Anzahl m_e der eingehenden Spikes summieren:

$$m_e = m_a + m_d \tag{3.3}$$

Es sollten keine Nachrichten gedropt werden, solange die mittlere Input-Rate der eingehenden Spikes sich nicht im Bereich der maximalen Übertragungsrate r_2^{max} der L2-Links („kritische Rate“) befindet oder diese überschreitet (vgl. Gl. 2.1):

$$r_{krit} \coloneqq r_2^{max} = \frac{n_2}{d} = \frac{8}{20}\frac{Spikes}{Takt} = \frac{r_1^{max}}{10} \tag{3.4}$$

Genauer sollte im Bereich hinreichend niedriger Raten für jedes Neuron die Anzahl der gesendeten und empfangenen Spikes gleich sein.

Die Verifikation des Designs ist erfolgreich abgeschlossen. Alle angeführten Sanity-Checks werden erfüllt. Die Anzahl der Drops an den Input-Buffern ist erst für Raten $r \lesssim r_{krit}$ ungleich 0. Oberhalb r_{krit} steigt die Droprate stark an. An den L2-Empfangsbuffern wurden keine Drops registriert.

3.2 Bandbreite und Droprate

Im letzten Abschnitt wurde die Verifikation des Routing-Systems beschrieben. Nun sollen die genaueren Eigenschaften des Systems in Bezug auf Übertragungsrate und Droprate untersucht werden. Zunächst soll untersucht werden, unter welchen Voraussetzungen Spikes gedropt werden.

Verlorene Spikes sind bei Anwendungen im Allgemeinen unerwünscht und ein deutlich gravierenderes Problem als Latenz und Jitter. Außerdem ist die Definition von Latenz und Jitter nicht klar, solange Spikes verloren gehen. Daher wird zunächst der Durchsatz und die Droprate des Routing-Systems untersucht. Als vereinheitlichtes Konzept, das Latenz, Jitter und Spike Drops zusammenfasst wird in weiter unten die vanRossum-Distanz vorgestellt.

Um die Auswertung und Präsentation der Daten übersichtlich zu gestalten wird in Folgenden ein System mit $n_G = 4$ anstatt 64 Neuronen pro SPL1-Link untersucht. Die Spikes werden weiterhin als Poisson-Prozess generiert. Entsprechend Formel 3.2 wird das System wird daher mit 16-facher Spikerate pro Neuron im Vergleich zur Verifikation betrieben, um vergleichbare Resultate zu erzielen. Nach der Additivität der Poissonverteilung ist diese Vorgehensweise legitim.

Es wird nun die Anzahl m_a der erfolgreich übertragenen und m_d der gedropten Spikes abhängig von der Anzahl m_e der Eingabespikes bei konstanter Übertragungsdauer $T = 100.000 Takte$ untersucht. m_e wird durch den Mittelwert t_m der Spikeabstände variiert. Dieser Test wird für verschiedene Tiefen l_{Ein} des Input-Buffers durchgeführt, um den Effekt eines externen Buffers zu untersuchen. Ein solcher Buffer ist nicht Teil des Routing-Systems, allerdings verhalten sich die Merger-Trees in der SPL1-Ebene ähnlich, wenn sich Daten aufstauen. Für jede Parameter-Paarung wird der Test für 10 verschiedene Seeds des Zufallsgenerators wiederholt. Um den Durchsatz zu maximieren wird dabei $\Delta t = 0$ gesetzt, sodass Nachrichten auf der Empfängerseite sofort

ausgeliefert werden. Außerdem wird $l_{Empfang} = 1$ gesetzt.

Die mittlere Input-Rate für einen Test kann nach Gleichung 3.2 abgeschätzt werden. Aus der registrierten Anzahl m_e der Input-Spikes kann die mittlere Inputrate außerdem exakt berechnet werden:

$$r_e = \frac{m_e}{T}. \tag{3.5}$$

Analog ist die Output-Rate $r_a = \frac{m_a}{T}$ und die Droprate $r_d = \frac{m_d}{T}$.

Die Input-Rate ist limitiert durch die Bandbreite der SPL1-Links (vgl. Gl. 2.2):

$$r_e^{max} = r_1^{max} = \frac{n_1}{2} = 2\frac{Spikes}{Takt}. \tag{3.6}$$

Da zwischen Input und Output die L2-Links das Bottleneck der Übertragung darstellen ist die Output-Rate limitiert durch die Bandbreite der L2-Links (vgl. Gl. 3.4):

$$r_a^{max} = r_2^{max} = \frac{n_2}{d} = \frac{20}{8}\frac{Spikes}{Takt}. \tag{3.7}$$

Es wird erwartet, dass für kleine Raten $r_e \ll r_{krit} := r_a^{max}$ keine Drops entstehen, also $r_a = r_e$. Für Raten r_e nahe, aber unterhalb r_{krit} („subkritische Raten“) werden durch statistische Schwankungen vereinzelt Drops entstehen, mehr für kleinere Bufferlänge l_{Ein}. Für Raten in der unmittelbaren Umgebung der kritischen Rate wird erstmals eine signifikante Anzahl m_d der Drops erwartet. Hier sollte die Buffertiefe l_{Ein} eine besonders große Rolle spielen. Für Raten oberhalb r_{krit} wird r_a konstant bleiben bei r_a^{max}, gedeckelt durch die maximale Übertragungsrate der L2-Links:

$$m_a \leq r_{krit} T \tag{3.8}$$

Im Gegenzug wird in diesem Bereich m_d proportional zu m_e wachsen, weitgehend unabhängig von l_{Ein}.

Für die graphische Auswertung wird die Input-Rate r_e auf die maximale Input-Rate r_e^{max} normiert und entspricht dann der Auslastung der L1-Links:

$$\eta_e = \frac{r_e}{r_e^{max}}. \tag{3.9}$$

Analog wird die Output-Rate auf die maximale Output-Rate normiert und entspricht dann der Auslastung der L2-Links:

$$\eta_a = \frac{r_a}{r_a^{max}}. \tag{3.10}$$

Die Droprate wird relativ zur Inputrate betrachtet:

$$r_d^{rel} = \frac{r_d}{r_e} = \frac{m_d}{m_e}. \tag{3.11}$$

Abb. 3.2 zeigt die gemessene Auslastung der L2-Links abhängig von der Auslastung der L1-Links für verschiedene Buffertiefen l_{Ein}. Die gepunktete Linie stellt die maximale Output-Rate dar, also volle Auslastung der L2-Links. Die Strichpunktlinie zeigt die ideale Übertragung an, also $m_a = m_e$, $m_d = 0$. Die beiden Linien schneiden sich bei der kritischen Rate, $r_e = r_{krit} = \frac{r_e^{max}}{10}$.

Im Bereich niedriger Input-Raten liegen die Messpunkte auf der Linie der idealen Übertragung. Für subkritische Input-Raten beginnen einige Messpunkte, leicht von der Linie abzuweichen, vor Allem für die Input-Buffertiefe $l_{Ein} = 1$. Bei der kritischen Input-Rate weichen alle Messpunkte sichtbar von dem Schnittpunkt der beiden Linien ab. Die Abweichung ist größer für kleinere Buffertiefe l_{Ein}. Für $r_e \gtrsim r_{krit}$ weichen die Messpunkte leicht nach unten von der Linie der vollen L2-Auslastung ab, stärker für kleinere l_{Ein}.

Für große Raten $r_e \gg r_{krit}$ liegen die Messpunkte auf der Linie, es sind minimale Abweichungen nach oben für tiefere Buffer zu erkennen. Das ist der Messprozedur geschuldet: Während T den Zeitraum begrenzt, in dem Input-Spikes angelegt werden, wird im Anschluss abgewartet, bis alle im System verbleibenden Spikes bis zum Output propagieren. Das führt dazu, dass die kleine Anzahl der im System verbleibenden Spikes (maximal $n_2 + 2n_1 - 1 + n_1 l_{Ein} = 15 + 4l_{Ein}$ Spikes) die Output-Rate minimal nach oben verfälscht. Da zu Beginn des Tests das System komplett leer ist, wird dieser Effekt teilweise, aber eben nicht vollständig ausgeglichen.

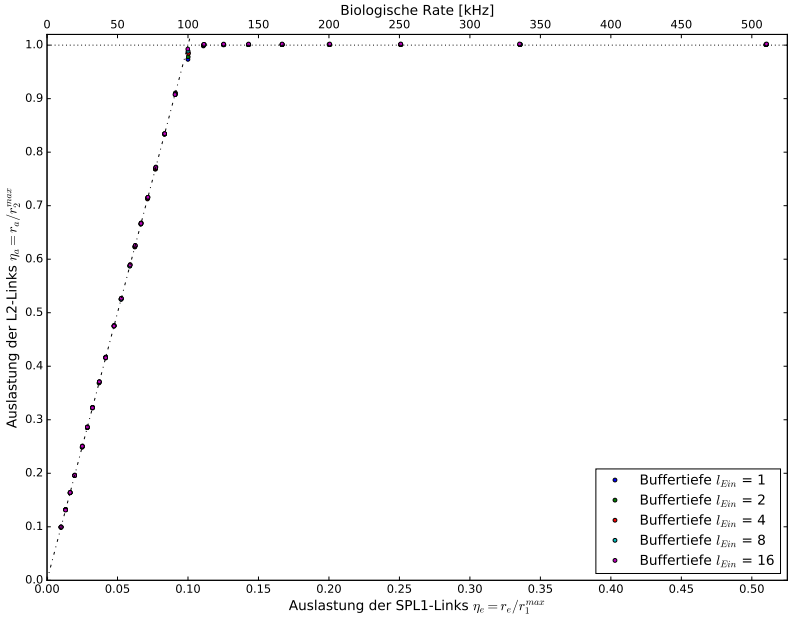

Abbildung 3.2: Gemessene Auslastung der L2-Links abhängig von der Auslastung der L1-Links für verschiedene Längen l_{Ein} des Input-Buffers. Die waagerechte gepunktete Linie stellt die komplette Auslastung der L2-Links dar, $\eta_a = 1$. Die Strichpunktlinie zeigt $r_a = r_e$, also $\eta_a = \eta_e r_a^{max}/r_e^{max}$ an, was einer verlustfreien Übertragung entspricht.

Abb. 3.3 zeigt die gemessene relative Droprate r_d^{rel}, also den Anteil der Spikes, der nicht erfolgreich übertragen wurde, abhängig von der Auslastung η_e der SPL1-Links. Die verschiedenen Buffertiefen l_{Ein} sind farblich unterschieden. Die minimal mögliche relative Droprate ist gepunktet eingezeichnet. Sie ist $r_d^{rel,min} = 0$ für $r_e < r_{krit}$ und

$$r_d^{rel,min} = \frac{r_e - r_a^{max}}{r_e} = 1 - \frac{r_a^{max}}{r_e} = 100\% - \frac{10\%}{\eta_e} \tag{3.12}$$

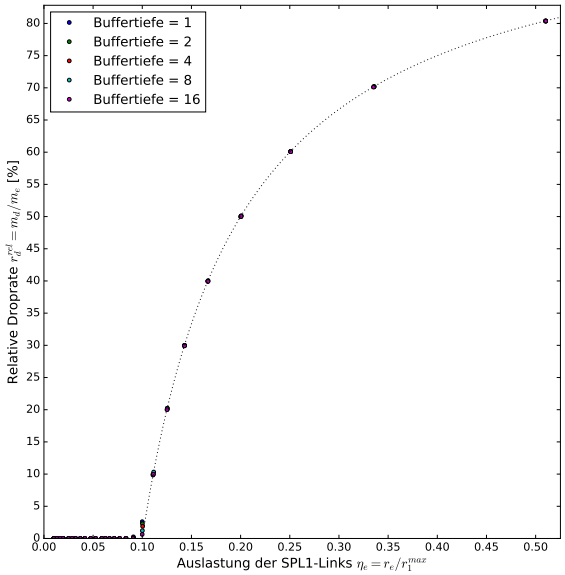

Abbildung 3.3: Gemessene Droprate abhängig von der Auslastung der L1-Links für verschiedene Längen l_{Ein} des Input-Buffers. Die gepunktete Kurve stellt die minimal mögliche Droprate dar, die bei gegebener Bandbreite der L2-Links möglich ist.

	Schwellwerte der Inputraten für folgende Dropraten:		
l_{Ein}	$r_d^{rel} > 0$	$r_d^{rel} > 0,1\%$	$r_d^{rel} > 1\%$
1	–	0,096	0,166
2	–	0,167	0,200
4	0,143	0,200	0,200
8	0,182	0,200	0,200
16	0,182	0,200	0,222

Abbildung 3.4: Charakteristische Schwellwerte der Droprate am Input und Input-Raten η_e, für die diese erstmals überschritten werden.

für $r_e > r_{krit}$.

Die beobachtete Droprate ist 0 für kleine Input-Raten r_e. für subkritische Raten r_e liegen für kleinere Buffertiefen die Messpunkte leicht über 0, am deutlichsten ist das für $l_{Ein} = 1$ erkennbar. Bei der kritischen Rate weichen alle Messpunkte deutlich von 0 ab, stärker für kleinere l_{Ein}. Für $r_e \gtrsim r_{krit}$ weichen die Messpunkte leicht nach oben von der Kurve der minimalen Droprate ab, stärker für kleinere l_{Ein}. Für $r_e \gg r_{krit}$ ist mit dem bloßen Auge keine Abweichung von der minimalenDroprate zu erkennen, das oben beschriebene Artefakt der Messprozedur tritt zwar auf, wird aber durch die größere Anzahl der gedropten Spikes stärker unterdrückt.

Tabelle 3.4 zeigt die relativen Input-Raten η_e, bei denen charakteristische Werte r_d^{rel} der Droprate erstmals überschritten werden. Die Länge $l_{Ein} = 4$ des Input-Buffers liefert robuste Ergebnisse und ist gleichzeitig hardwareschonend. Das ist ein guter Kompromiss, sodass in folgenden Tests $l_{Ein} = 4$ verwendet wird.

3.3 Lastausgleich

Eine wichtige Komponente des Routing-Systems ist die Lastausgleich-Eigenschaft. Diese soll nun gezielt analysiert werden, indem der in Abschnitt 3.2 beschriebene Test für verschiedene Kombinationen sendender SPL1-Links wiederholt wird.

Da im vorherigen Abschnitt bei statistisch verteilten Spikedaten keine starke Abweichung von den idealen Werten für die Auslastung zu erkennen ist, kann es keine groben Fehler im Lastausgleich-System geben. Es ist nicht zu erwarten, aber dennoch möglich, dass bei der detaillierten Betrachtung kleinere Schwächen zu erkennen sind, dass etwa ein einzelner SPL1-Link leicht benachteiligt wird, oder dass sich verschiedene Kombinationen zweier sendender SPL1-Links aufgrund der Asymmetrie der Merger-Struktur leicht unterschiedlich verhalten.

Wie erwähnt wird in diesem Abschnitt $l_{Ein} = 4$ konstant gesetzt. Es werden nun also in allen möglichen Kombinationen SPL1-Links deaktiviert. Die Anzahl ν der möglichen Kombinationen mit k aktiven SPL1-Links berechnet sich zu

$$\nu(k) = \binom{n_1}{k}. \tag{3.13}$$

Im Vergleich zu $k = 4$ müssen für $k \in \{1, 2, 3\}$ die Input-Raten r_e angepasst werden. Entspricht die kritische Rate etwa für $k = 4$ pro Link

$$r_{krit}^{Link}(k = 4) = \frac{r_{krit}}{k} = \frac{1}{10}\frac{Nachrichten}{Takt}, \tag{3.14}$$

so liegt sie bei einem einzelnen Link bei

$$r_{krit}^{Link}(k = 1) = r_{krit} = \frac{2}{5}\frac{Nachrichten}{Takt}. \tag{3.15}$$

Die Auswertung der Daten ergibt sich dagegen identisch zu Abschnitt 3.2.

Abb. 3.5 zeigt die gemessene Auslastung der L2-Links abhängig von der Auslastung der SPL1-Links. Für jede mögliche Anzahl k aktiver Links wird über alle $\nu(k)$ Möglichkeiten gemittelt. Es sind keine signifikanten Abweichungen von den Linien der optimal möglichen Übertragung zu erkennen, unabhängig von der Anzahl der sendenden SPL1-Links. Lediglich bei der kritischen Input-Rate weichen die Messpunkte wieder leicht von der idealen Übertragungsrate ab.

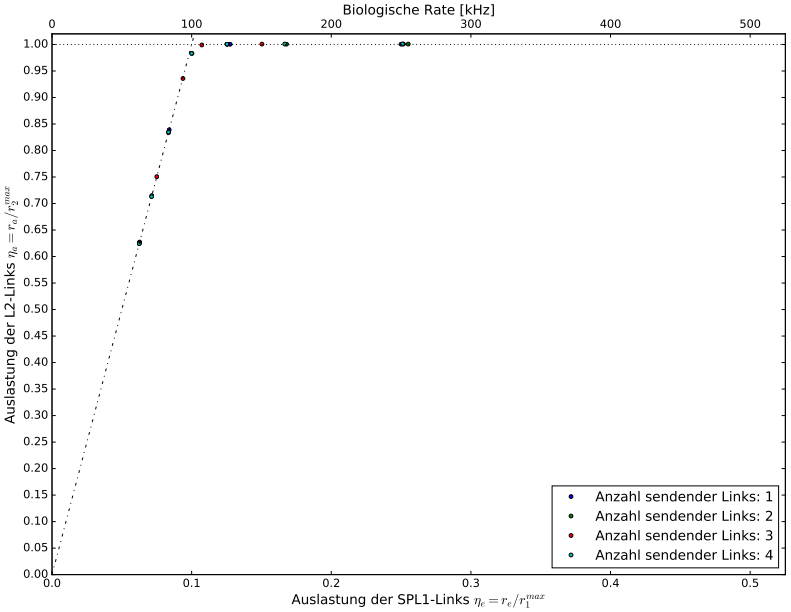

Abbildung 3.5: Gemessene Auslastung der L2-Links abhängig von der Auslastung der L1-Links für verschiedene Kombinationen der aktiv sendenden Links. Für alle Kombinationen der gleichen Anzahl aktiver Links wurde gemittelt. Die waagerechte gepunktete Linie stellt die komplette Auslastung der L2-Links dar, $\eta_a = 1$. Die Strichpunktlinie zeigt $r_a = r_e$, also $\eta_a = \eta_e r_a^{max}/r_e^{max}$ an, was einer verlustfreien Übertragung entspricht.

3.4 Droprate an den L2-Links

Bisher wurde $\Delta t = 0$ gesetzt und die Drops am SPL1-Input isoliert betrachtet. Es können aber unter Umständen auch Spikes am Ende der L2-Verbindung gedropt werden, wenn der Empfangsbuffer der Tiefe $l_{Empfang}$ voll ist. Es sollen nun zunächst theoretisch die Umstände analysiert werden, unter denen das zu erwarten ist. Anschließend sollen diese genau ausgemessen werden.

Das Bottleneck der Übertragung liegt in den L2-Links. Hinter den L2-Links, wo sich der Empfangs-Buffer befindet, fließen Spike-Nachrichten dagegen schnell ab, solange $\Delta t = 0$. Sei λ_{min} die minimal mögliche Latenz einer Spikenachricht zwischen dem Anheften des Zeitstempels am Eingang des SPL1-to-L2-Moduls und dem Zeitabgleich im L2-to-SPL1-Modul. Diese minimale Latenz tritt beispielsweise auf, wenn das System vollständig leer ist und entspricht der Anzahl der zu durchlaufenden Registerstufen in diesem Bereich. Explizit wird

$$\lambda_{min} = 7Takte \tag{3.16}$$

erwartet (vgl. Abb. 2.2, 2.3, 3.1).

Solange $\Delta t \leq \lambda_{min}$ ist der Zeitabgleich für jeden Spike erfüllt (vgl. Gl. 2.5). Wird $\Delta t > \lambda_{min}$ gesetzt, so müssen manche Spikes auf Ablauf der Soll-Latenz warten. Diese Spikes füllen zunächst das interne Input-Register des L2-to-SPL1-Moduls. Wird Δt weiter erhöht, müssen immer mehr Spikes immer länger warten, sodass sie sich sukzessive in den Empfangsbuffer der L2-Links aufstauen.

Ist der Empfangsbuffer eines L2-Links schließlich vollständig gefüllt und wird nun eine weitere Spike-Nachricht über den Link übertragen, so geht diese verloren. Da die Latenz der Nachrichten steigt, je mehr die Merger-Struktur im SPL1-to-L2-Modul gefüllt ist, müssen Nachrichten auf Empfängerseite in diesem Fall weniger warten. Es sind also für kleinere Input-Raten mehr Drops am Empfangsbuffer zu erwarten. Der Schwellwert für Δt, ab dem Drops dieser Art zu erwarten sind, wird im folgenden grob abgeschätzt.

Die Eingangsregister des L2-to-SPL1-Moduls nehmen pro L2-Link eine Spike-Nachricht auf. Die Empfangsregister der L2-Links nehmen bis zu $l_{Empfang}$ Nachrichten pro Link auf. Insgesamt könne sich also am empfangenden Ende eines Links bis zu $(l_{Empfang}+1)$ Nachrichten vor dem Zeitabgleich aufstauen. Nehme den Fall an, dass die L2-Links voll ausgelastet sind, ohne dass sich der Merging-Baum im SPL1-to-L2-Modul füllt, was im Bereich der kritischen Input-Rate passieren kann. Die Nachrichten haben dann minimale Latenz λ_{min} und müssen nach der Übertragung für $\Delta t - \lambda_{min}$ warten. In diesem Zeitraum können

$$s = abrunden\left(\frac{\Delta t - \lambda_{min}}{d}\right) = abrunden\left(\frac{\Delta t - 7}{20}\right) \tag{3.17}$$

Nachrichten über den betrachteten Einzellink übertragen werden. Spikes werden gedropt, falls

$$s > l_{Empfang} + 1. \tag{3.18}$$

Für $l_{Empfang} = 1$ ergibt sich so der Schwellwert $\Delta t = 47$, ab dem Drops auftreten, für $l_{Empfang} = 2$ entsprechend $\Delta t = 67$ und für $l_{Empfang} = 3$ $\Delta t = 87$ usw. Wie erwähnt handelt es sich hierbei um eine Überschlagsrechnung, die genauen Werte werden im Folgenden experimentell ermittelt.

Dazu wird für verschiedene Input-Raten r_e und verschiedene Längen $l_{Empfang}$ des Empfangsbuffers die Droprate r_d^{L2} an den L2-Links abhängig von Δt gemessen. Wie erwähnt bleiben die Anderen Parameter unverändert. Für die graphische Auswertung wird wieder die relative Droprate betrachtet:

$$r_d^{L2,rel} = \frac{r_d^{L2}}{r_e} = \frac{m_d^{L2}}{m_e}. \tag{3.19}$$

Die Input-Rate wird von nun an immer auf die kritische Rate normiert:

$$r_e^{rel} = \frac{r_e}{r_{krit}}. \tag{3.20}$$

Mit dieser Definition entspricht $r_e^{rel} = 1$ der vollen Auslastung der L2-Links, also dem Knick in Abb. 3.2.

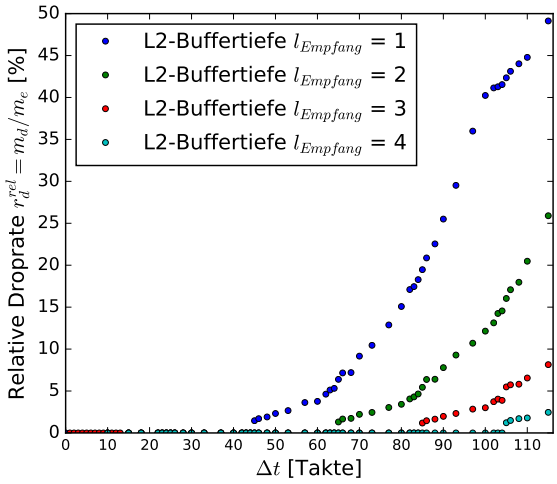

Abbildung 3.6: Gemessene relative Droprate $r_d^{L2,rel}$ am Empfangsbuffer der L2-Links abhängig von Δt für verschiedene Längen $l_{Empfang}$ des Buffers. Exemplarisch aufgenommen für die relative Input-Rate $r_e^{rel} = 1$.

Abb. 3.6 zeigt exemplarisch für die kritische Input-Rate $r_e^{rel} = 1$, wie sich die Droprate mit Δt entwickelt für verschiedene Bufferlängen $l_{Empfang}$. Für die Bufferlänge $l_{Empfang} = 1$ treten ab $\Delta t = 44$ erste Drops auf. Jeweils im Abstand von 20 und 40 Takten erhöht sich der Anstieg der Droprate deutlich. Im Abstand von 60 Takten erringert sich der Anstieg schließlich wieder.

Für die Bufferlänge $l_{Empfang} = 2$ treten ab $\Delta t = 64$ erste Drops auf, für $l_{Empfang} = 3$ ab $\Delta t = 84$ und für $l_{Empfang} = 4$ ab $\Delta t = 104$. Die Kurven für $l_{Empfang} = 2$, $l_{Empfang} = 3$ und $l_{Empfang} = 4$ gleichen in ihrer Form der Kurve für $l_{Empfang} = 1$, aber um 20, 40 und 60 Takte verschoben.

Die Kurven für kleinere Input-Raten weisen im Allgemeinen größere relative Dropraten auf, sehen ansonsten aber gleich aus. Für größere Input-Raten sind die Dropraten deutlich geringer und steigen zu beginn weniger steil an.

Die Schwellwerte für Δt, ab denen Drops an den L2-Buffern auftreten werden in Tab.3.7 zusammengefasst. Der Schwellwert hängt nicht in relevantem Maß von der

	Schwellwerte von Δt [Takte] für folgende L2-Buffertiefen:			
r_e^{rel}	1	2	3	4
0,13	45	65	85	105
0,18	44	64	84	104
0,20	44	64	84	104
0,22	44	64	84	104

Abbildung 3.7: Schwellwerte für Δt, ab denen Drops an den L2-Links auftreten für verschiedene Input-Raten r_e^{rel} und verschiedene Tiefen $l_{Empfang}$ des L2-Empfangsbuffers.

Input-Rate ab. Der Schwellwert für die minimale Buffertiefe $l_{Empfang} = 1$ ist $\Delta t = 44Takte$, für jede weitere Bufferstufe erhöht er sich um $20Takte$.

Um Drops dieser Art zu vermeiden, wird für die folgenden Tests die L2-Buffertiefe auf $l_{Empfang} = 3$ gesetzt.

3.5 Latenz und Jitter

Nun ist bekannt, für welche Parameter Drops am Input-Buffer und am L2-Empfangsbuffer auftreten. Nun kann untersucht werden, wie sich Latenz und Jitter des Systems verhalten. Zunächst soll untersucht werden, wie sich die Latenz eines Spikes von Input zu Output verhält und welchen Einfluss der Parameter Δt darauf hat.

Dazu wird das Test-Setup mit unveränderten Parametern verwendet und dabei die Input-Rate und Δt variiert. Für jedes erfolgreich übermittelte Input-Output-Datenpaar wird die Latenz $\lambda = t_{out} - t_{in}$ berechnet. Gedropte Spikes werden separat registriert. Da $l_{Empfang} = 3$ sind keine Drops an den L2-Buffern zu erwarten. Zunächst wird die statistische Verteilung der Latenz untersucht.

Für $\Delta t = 0$ werden die Nachrichten im L2-to-SPL1-Modul nicht zurückgehalten. Minimale mittlere Latenz ist garantiert, allerdings ist ein hoher Jitter zu erwarten: Durch zeitlich schwankende Eingangsrate sind die Bufferstufen zu verschiedenen Zeiten unterschiedlich voll, speziell im Merger-Baum des SPL1-to-L2-Moduls führen volle Bufferstufen direkt zu höherer Latenz. Analysiere die Verteilung der Latenz λ zunächst für $\Delta t = 0$:

Bei kleinen Input-Raten ist das SPL1-to-L2-Modul weitestgehend leer. Es ist zu erwarten, dass viele Spikes mit der minimalen Latenz $\lambda = \lambda_{min}$ übermittelt werden. Daneben sind einige Spikes mit wenigen Takten mehr als λ_{min} zu erwarten. Insgesamt ist also eine geringe Latenz und wenig Jitter zu erwarten.

Bei subkritischen Input-Raten ist das SPL1-to-L2-Modul teilweise gefüllt. Die mittlere Latenz eines Spikes ist damit erwartungsgemäß größer. Es sind weiterhin viele Spikes der Latenz $\lambda = \lambda_{min}$ und wenigen Takten mehr zu erwarten. Zusätzlich ist aber auch mit einigen Ausreißern größerer Latenz zu rechnen. Insgesamt ist hier also mit etwas größerer Latenz und etwas mehr Jitter zu rechnen. Außerdem treten erste Drops durch Overflow der Input-Buffer auf.

In der unmittelbaren Umgebung der kritischen Rate schwankt der Füllstand des SPL1-to-L2-Moduls durch die statistisch ankommenden Input-Spikes stark. So ist mit einer deutlich erhöhten mittleren Latenz λ zu rechnen, sowie starken Schwankungen in beide Richtungen. Insgesamt ist damit eine Latenz mittlerer Größe und starker Jitter zu erwarten. Außerdem treten hier Drops in relevanten Mengen auf.

Oberhalb der kritischen Rate ist das Routing-System die meiste Zeit vollständig gefüllt. Damit haben die meisten Spikes eine Latenz im Bereich knapp unterhalb der maximalen Latenz λ_{max}. Die mittlere Latenz ist also groß, mit wenig Jitter. Da hier Spike-Drops in signifikantem Maß auftreten sind die genauen Werte nur von bedingter Aussagekraft.

Die minimale Latenz berechnet sich aus der Anzahl der zu durchlaufenden Registerstufen im leeren Test-System. Das beinhaltet die Input-Bufferstufe, das SPL1-to-L2-Modul, die simulierten L2-Links mit L2-Empfangsbuffer und schließlich das L2-to-SPL1-Modul, hinter das L2-to-SPL1-Modul wurden keine weiteren Registerstufen geschaltet. (vgl. Abb. 2.2, 2.3, 3.1)

$$\lambda_{min} = \lambda_{min}^{in} + \lambda_{min}^{SPL1toL2} + \lambda_{min}^{L2} + \lambda_{min}^{L2toSPL1} \tag{3.21}$$
$$= 1 + (2 + log_2(n_1)) + 2 + 4 \tag{3.22}$$
$$= 1 + 4 + 2 + 4 = 11 \tag{3.23}$$

Die maximale Latenz kann nicht so einfach berechnet werden. Die folgende Rechnung ist lediglich eine Abschätzung. Angenommen, das Test-System ist bis auf eine freie Position im Input-Buffer vollständig gefüllt. Betrachte eine Nachricht, die in dieses System gegeben wird. Vor dem Bottleneck, auf der Senderseite des Systems befinden sich (vgl. Abb. 2.2, 3.1)

$$m_{max} = m_{max}^{in} + m_{max}^{SPL1toL2} + m_{max}^{L2-Sender} - 1 \tag{3.24}$$
$$= (l_{Ein} n_1) + (n_2 + 2n_1 - 1) + n_2 - 1 \tag{3.25}$$
$$= 2n_2 + (l_{Ein} + 2)n_1 - 2 \tag{3.26}$$
$$= 16 + 24 - 2 = 38 \tag{3.27}$$

weitere Spike-Nachrichten, die zuerst bearbeitet werden müssen. Die L2-Links arbeiten mit der maximalen Rate von

$$r_2^{max} = \frac{2}{5} \frac{Nachrichten}{Takt} \tag{3.28}$$

(vgl. Gl. 2.1). Damit wird zum abarbeiten der anderen Nachrichten die Zeit

$$t_{warten} = \frac{m_{max}}{r_2^{max}} = 95 Takte \tag{3.29}$$

benötigt. Zusätzlich muss die betrachtete Nachricht nach dem Abarbeiten der anderen Nachrichten die Empfängerseite des Testsystems durchlaufen, was weitere 5 Takte dauert (vgl. Abb. 2.3, 3.1). Insgesamt braucht die Nachricht also die maximale Latenz:

$$\lambda_{max} \approx 100 Takte. \tag{3.30}$$

Der Zeitabgleich mit Δt wirkt sich aber nur auf den Teil der Latenz aus, der zwischen dem Anheften des Zeitstempels am Input des SPL1-to-L2-Moduls und dem Zeitabgleich im L2-to-SPL1-Modul entsteht. Die Latenz λ kann also in einen beeinflussbaren

(„reduzierten“) Anteil λ^{red} und einen Restanteil λ^{rest} zerlegt werden. Die Berechnung dieser minimalen und maximalen reduzierten Latenz λ^{red}_{min} bzw. λ^{red}_{max} wird wie oben durchgeführt: Die Anzahl der Registerstufen in diesem Bereich ist (vgl. Gl. 3.16)

$$\lambda^{red}_{min} = \lambda^{SPL1toL2}_{min} + \lambda^{L2}_{min} + \lambda^{L2toLSPL1,inReg}_{min} \tag{3.31}$$
$$= (2 + aufrunden(log_2(n_1))) + 2 + 1 \tag{3.32}$$
$$= 4 + 2 + 1 = 7. \tag{3.33}$$

Ohne die Input-Buffer können sich bis zu

$$m^{red}_{max} = m^{SPL1toL2}_{max} + m^{L2-Sender}_{max} - 1 \tag{3.34}$$
$$= (n_2 + 2n_1 - 1) + n_2 - 1 \tag{3.35}$$
$$= 2n_2 + 2n_1 - 2 \tag{3.36}$$
$$= 16 + 8 - 2 = 22 \tag{3.37}$$

Nachrichten vor den L2-Links ansammeln. Das ergibt die Wartezeit

$$t^{red}_{warten} = \frac{m^{red}_{max}}{r^{max}_{2}} = 55Takte \tag{3.38}$$

Zusätzlicht brauch eine Nachricht in der Empfängerseite weitere 2 Takte. Folglich ist

$$\lambda^{red}_{max} \approx 57Takte. \tag{3.39}$$

Zusammengefasst liegt die Latenz eines Spikes ohne Zeitabgleich vom Eingang zum Ausgang zwischen $\lambda_{min} = 11$ und $\lambda_{max} \approx 100$ Takten, davon befinden sich zwischen $\lambda^{red}_{min} = 7$ und $\lambda^{red}_{max} \approx 57$ Takte in dem Bereich, der vom Zeitabgleich beeinflusst wird. Nicht beeinflusst werden die Restanteile der Latenz zwischen

$$\lambda^{rest}_{min} = \lambda_{min} - \lambda^{red}_{min} = 4Takte \tag{3.40}$$

und

$$\lambda^{rest}_{max} = \lambda_{max} - \lambda^{red}_{max} \approx 43Takte \tag{3.41}$$

Wird nun der Zeitabgleich aktiviert, indem die Soll-Latenz Δt erhöht wird, so kann die reduzierte Latenz gezielt erhöht werden, um die Schwankungen in der reduzierten Latenz zu verringern. Die Latenz einer Spike-Nachricht mit ursprünglicher Latenz

$$\lambda = \lambda^{red} + \lambda^{rest} \tag{3.42}$$

und $\lambda^{red} < \Delta t$ wird dann auf

$$\lambda' = \Delta t + \lambda^{rest} \tag{3.43}$$

erhöht.

Aus obigen Werten kann der maximal sinnvolle Wert für Δt geschätzt werden, ab dem die gesamte Latenz zwar weiter erhöht wird, der Jitter im beeinflussbaren Anteil der Latenz aber bereits auf 0 reduziert wurde und so nicht weiter verringert wird.

$$\Delta t_{max} = \lambda^{red}_{max} \approx 57Takte \tag{3.44}$$

Dieses berechnete Verhalten soll nun mit den Messwerten verglichen werden. Zuerst wird die Verteilung der Latenz histogrammatisch betrachtet. Abb. 3.8 zeigt die

Verteilung der Latenzen für verschiedene Werte von Δt und für verschiedene Input-Raten.

Ganz links in jedem Einzelbild ist der Anteil der gedropten Spikes in einem separaten Bin dargestellt. Eine rote gepunktete Linie kennzeichnet die minimale Latenz (vgl. Gl. 3.40, 3.43)

$$\lambda_{min} = \Delta t + \lambda_{min}^{rest} = \Delta t + 4, \tag{3.45}$$

die bei eingestelltem Δt zu erwarten ist. Der grüne Bereich umfasst von λ_{min} ausgehend $30Takte = 120ns = 120s(bio)$. Das entspricht 12% eines typischerweise akzeptierten Jitters von 1ms(bio). Diese Zielsetzung ist willkürlich und darin begründet, dass andere Komponenten des Gesamtsystems erwartungsgemäß einen größeren Anteil des Jitters ausmachen.

Das oberste Bild zeigt jeweils die Verteilung bei $\Delta t = 0$, also deaktiviertem Zeitabgleich. Für steigendes Δt werden immer mehr Spikes von Zeitabgleich beeinflusst und damit in ihrer Latenz auf die Soll-Latenz gehoben. Das ergibt einen sehr ausgeprägten Peak bei der Soll-Latenz und eine rasch abfallende Verteilung oberhalb der Soll-Latenz.

Wie zu erkennen ist, ist die Reduktion des Jitters sehr effektiv für kleine Input-Raten $r_e < r_{krit}$ und wenig effektiv für große Input-Raten $r_e > r_{krit}$.

Das nächste Ziel ist nun die genauere Quantifizierung von Latenz und Jitter. Als Maß für den Jitter j eines Spikes der Latenz λ wird die absolute Abweichung von der mittleren Latenz $\overline{\lambda}$ gewählt:

$$j = |\overline{\lambda} - \lambda| \tag{3.46}$$

Abb. 3.9 zeigt gemessene Latenz und Jitter abhängig von Δt für verschiedene Input-Raten. Neben dem Mittelwert (blau) sind der Median (dick schwarz), Minimum und Maximum (dünn schwarz), sowie die Quantile 0,1%, 1%, 10%, 90%, 99% und 99,9% eingezeichnet. Für kleinere Raten ist die Verteilung der Latenz in allen Quantilen stärker gebündelt. Der allgemeine Verlauf der einzelnen Quantile ist aber qualitativ ähnlich:

Zunächst sind verlaufen alle Quantile nahezu konstant. Ab $\Delta t \approx 7Takte$ beginnen die unteren Quantile, sich einer Geraden anzunähern, die linear mit Δt wächst. Sobald diese Gerade die ursprüngliche Höhe eines der Quantile erreicht, knickt der Verlauf des Quantils ab und verläuft auf der geraden weiter. Ab $\Delta t \approx 60Takte$ für hohe Raten bzw. $\Delta t \approx 45Takte$ für niedrige Raten knicken schließlich alle Quantile ab und verlaufen parallel zur Geraden weiter.

Aus den Kurven der Latenz kann bereits qualitativ abgelesen werden, wie der Jitter verläuft. Quantitativ eindeutig wird der Verlauf in den entsprechenden Kurven auf der rechten Seite. Für die niedrigen Raten fallen die Quantile des Jitters ab $\Delta t \approx 7Takte$ stark ab. Ab $\Delta t \approx 50Takte$ bleiben die Quantile dann weitgehend konstant.

Für die kritische Rate fällt der Jitter flacher und langsamer ab, ab $\Delta t \approx 60Takte$ sind keine signifikanten Änderungen mehr zu sehen. Für die Rate $r_e > r_{krit}$ Werden die oberen Quantile des Jitters deutlich reduziert, Median und Mittelwert bleiben allerdings unverändert. Ab $\Delta t \approx 60Takte$ sind keine weiteren sigmifikanten Veränderungen zu sehen.

Damit ist die Charakterisierung der technischen Daten der Schnittstelle abgeschlossen. Mit der vanRossum-Distanz folgt im nächsten Abschnitt zusätzlich eine biologisch motivierte Charakterisierung.

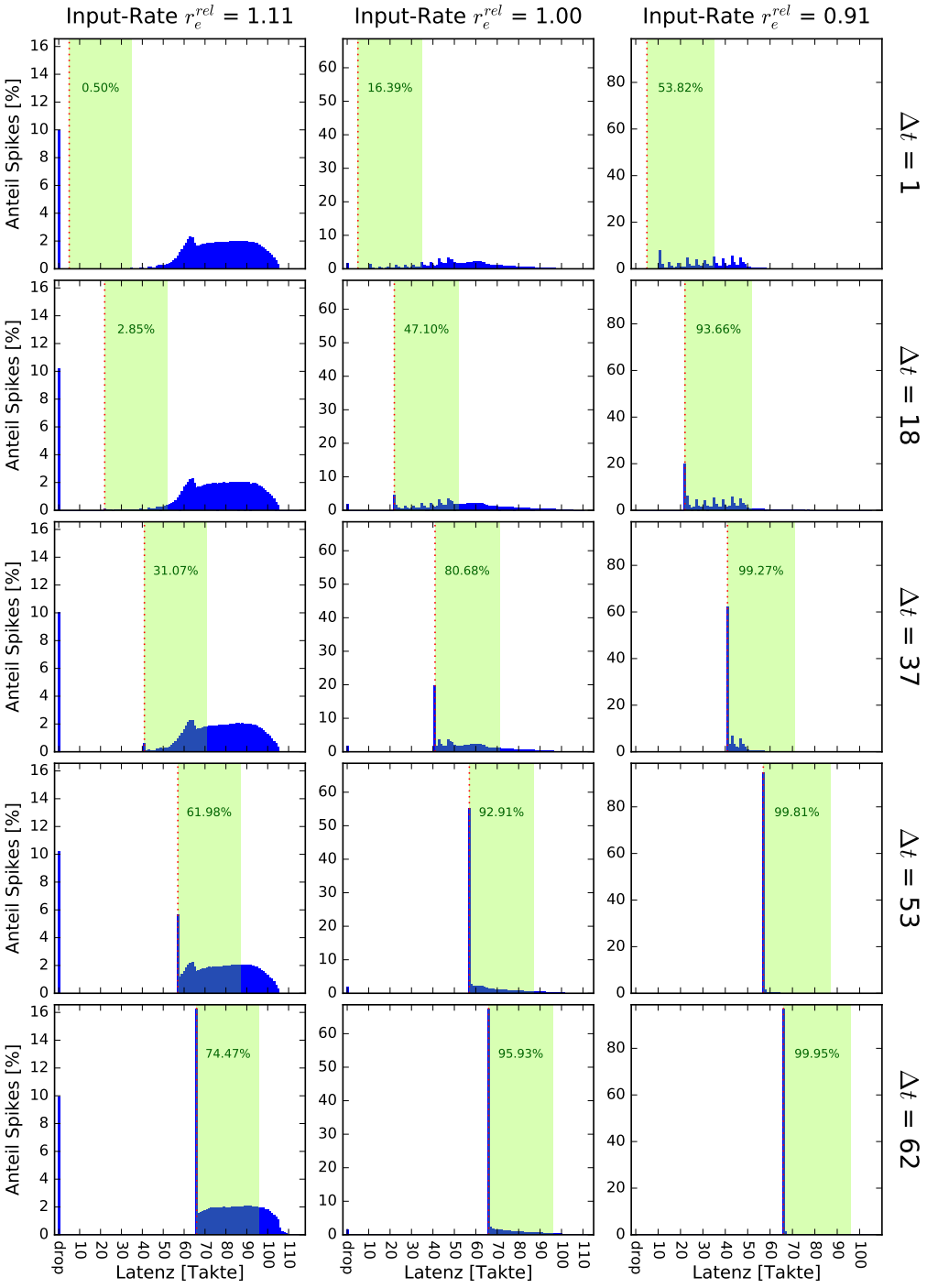

Abbildung 3.8: Histogramm: Verteilung der Latenz für verschiedene Input-Raten und verschiedene Werte von Δt. Jeweils ganz links: Anteil der Spike-Drops. Rot gepunktet: Durch eingestelltes Δt zu erwartende minimale Latenz. Grün hinterlegt: Bereich der Breite 30 Takten = 120µs(bio) von der minimalen Latenz ausgehend mit dem Anteil der Spikes in diesem Bereich.

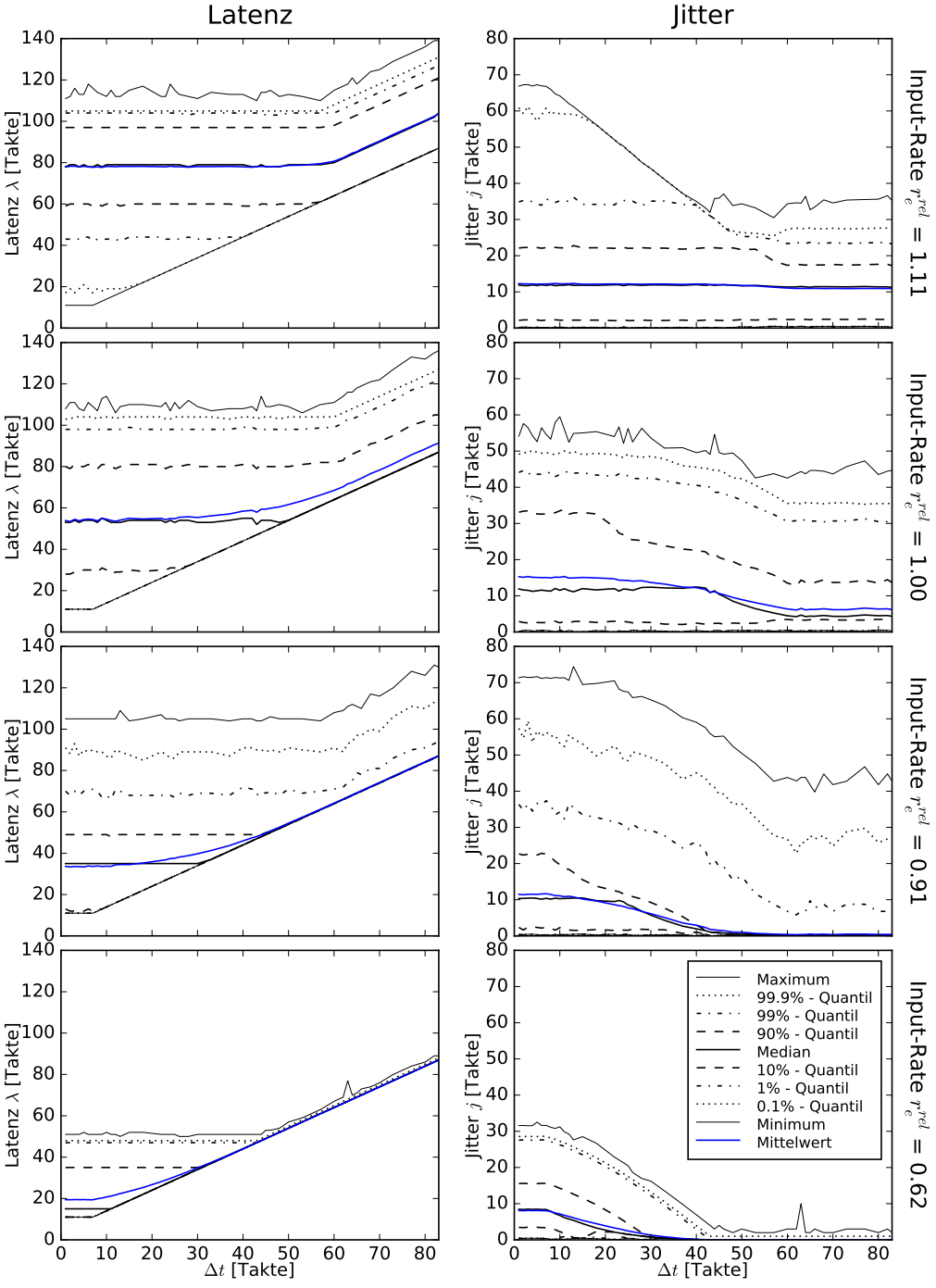

Abbildung 3.9: Verlauf der Latenz λ (links) und des Jitters j (rechts) abhängig von Δt für verschiedene Input-Raten. Eingezeichnet ist der Mittelwert(blau), sowie mehrere Quantile.

3.6 vanRossum-Distanz und Korrelationsmatrix

Mit der vanRossum-Distanz (beschrieben in [10]) existiert ein Konzept, mit dem man die Eignung des Routing-Systems für die biologisch orientierten Experimente mit neuronalen Netzen untersuchen kann. Technisch gesehen handelt es sich dabei um eine Charakterisierungsmethode, die Jitter und Droprate erfasst.

Die vanRossum-Distanz ist eine Metrik, die die Ähnlichkeit von zwei Spike-Trains S und T misst. Eine kleine vanRossum-Distanz bedeutet im Allgemeinen, dass die zwei Spike-Trains ähnlich sind. Die vanRossum-Distanz zwischen einem leeren Spike-Train S und einem Spike-Train T mit einem Spike ist normiert auf 1. Die vanRossum-Distanz wird wie folgt berechnet:

1. Für jeden Spike $i \in S$ mit Spike-Zeitpunkt t_i wird eine exponentielle Zerfallsfunktion

$$f_i(t) \coloneqq \frac{1}{\tau}\theta(t)exp(-\frac{t}{\tau}) \tag{3.47}$$

erzeugt, wobei θ die Heavisidefunktion ist. Diese Funktionen werden summiert zur Gesamtfunktion

$$f_S(t) \coloneqq \sum_{i \in S} f_i(t). \tag{3.48}$$

2. Schritt 1 wird für T wiederholt mit Ergebnis f_T.

3. Die vanRossum-Distanz ergibt sich durch

$$d_{vR}(S,T) = \int_0^\infty (f_S(t) - f_T(t))^2 dt \tag{3.49}$$

Der freie Parameter τ, die Zerfallskonstante der Exponentialfunktionen, gibt dabei die zeitliche Genauigkeit an, mit der die Spiketrains verglichen werden. Je größer τ, desto toleranter ist die Distanz gegenüber kleinen zeitlichen Verschiebungen. τ sollte daher der gewünschten zeitlichen Fehlertoleranz entsprechen. Im folgenden wird ein typischer Wert von $\tau = 0,96ms(bio)$ verwendet.

Hier wirden mittels der vanRossum-Distanz der Output-Spiketrain T mit dem Input-Spiketrain S verglichen. Dabei werden die Zeitpunkte der Output-Spikes jeweils um λ'_{min} verringert, um die konstante zeitliche Verschiebung der Output-Spikes zu kompensieren. λ'_{min} ist dabei die minimale Latenz bei aktiviertem Zeitabgleich und wird durch die Latenz des ersten gesendeten Spikes bestimmt.

Für jeden Spike, der im Routing-System gedropt wird, erhöht sich die vanRossum-Distanz um 1. Außerdem werden Abweichungen zwischen Input und Output durch Jitter festgestellt und machen sich zusätzlichen durch einen geringeren Betrag bemerkbar.

Die vanRossum-Distanz ist ein gutes Maß, um die Qualität der Übertragung für jedes der $n_n = 16$ Neuronen zu ermitteln. Da das Routing-System Lastausgleich betreibt und dabei die Spikes der $n_1 = 4$ SPL1-Links auf nichttriviale Weise zusammenführt sollen zusätzlich Korrelationen zwischen der Übertragung verschiedener Neuronen untersucht werden.

Die folgende Analysemethode für die Korrelation zwischen der Übertragung zweier verschiedener Neuronen wurde von Vitali Karasenko entwickelt. Sie verwendet

die vanRossum-Distanz, um mögliche Wechselwirkungen zu quantifizieren. Seien S_A Input- und T_A Output-Spiketrain des Neurons A und S_B, T_B die des Neurons B. Die Distanzen $d_{vR}(S_A, S_B)$ und $d_{vR}(T_A, T_B)$ haben jeweils für sich genommen keine Aussagekraft, da sie die statistische Verteilung der Input-Spikes widerspiegeln. Betrachtet man aber die (betragsmäßige) Differenz von Prä- und Post-Distanz

$$\delta(A, B) \coloneqq |d_{vR}(S_A, S_B) - d_{vR}(T_A, T_B)|, \tag{3.50}$$

so können kleine Verschiebungen sichtbar gemacht werden. Sind diese Verschiebungen statistisch verteilt, so löschen sie sich größtenteils aus. Sind diese Verschiebungen dagegen systematischer Natur, wie es etwa dann auftritt, wenn ein Link im System tendenziell bevorzugt wird, so weicht der Wert $\delta(A, B)$ deutlich von 0 ab. Somit eignet sich $\delta(A, B)$ als Maß der Korrelation zwischen der Übertragung unterschiedlicher Neuronen.

Die Schnittstelle wurde so entworfen, dass der Jitter möglichst gering ausfällt. Wie in den Testresultaten in Abschnitt 3.5 zu sehen ist, wird der Jitter deutlich reduziert, aber nicht vollständig eliminiert. Damit ist für jedes Neuron eine geringe vanRossum-Distanz $d_{vR}(S, T) \neq 0$ zu erwarten. Außerdem ist das SPL1-to-L2-Modul darauf ausgelegt, alle SPL1-Links gleichwertig zu behandeln. Damit sollten auch die Korrelationen $\delta(A, B)$ bis auf vernachlässigbare statistische Fluktuationen 0 sein.

Werden wider dieser Erwartung nicht alle SPL1-Links gleich behandelt, so sollte für zwei Neuronen A, B unterschiedlicher SPL1-Links Werte $\delta(A, B) > 0$ auftreten. Im Gegensatz dazu sollte die Korrelation zwischen zwei Neuronen A, B des selben SPL1-Links weiterhin $\delta(A, B) \approx 0$ betragen, da diese im SPL1-to-L2-Modul nicht mehr voneinander unterschieden werden können. Damit würde die Korrelation zweier Neuronen unterschiedlicher SPL1-Links die Korrelation zweier Neuronen des selben SPL1-Links deutlich übersteigen.

Die hier beschriebene Analyse wurde für verschiedene Parameter-Sets durchgeführt. Abb. 3.10 zeigt exemplarisch die Testergebnisse für die Parameter $\Delta t = 70, l_{Ein} = 4, l_{Empfang} = 3$ und der Input Rate $r_e^{rel} = 0,71$, die einem typischen Anwendungsfall entspricht. Dabei wurden für jedes Neuron die ersten 65 Spikes betrachtet. Links oben im Bild sieht man die zeitliche Verteilung der Input-Spikes für jedes Neuron als Rasterplot. Zum Vergleich sind links unten die Output-Spikes dargestellt, wobei die zeitliche Verschiebung durch die minimale Latenz kompensiert wurde.

Die Matrix rechts im Bild fasst die vanRossum-Distanzen und die Korrelationen zusammen. Die Diagonalelemente m_{ii} zeigen für die Neuronen $i \in [0, 15] \cap \mathbb{N}$ die vanRossum-Distanz $m_i i = d_{vR}(S_i, T_i)$ zwischen dem Prä-Spiketrain S_i und dem Post-Spiketrain T_i. Die anderen Elemente $m_{ij}, i \neq j$ zeigen die Korrelation $m_{ij} = m_{ji} = \delta(i, j)$ zwischen Neuron i und j an.

Die Korrelationen zwischen Neuronen, die dem selben SPL1-Link angehören, entsprechen den Elementen $m_{ij}, i \neq j$, die in 4 Blöcken der Größe 4×4 um die Diagonale der Matrix angeordnet sind. Sollten einzelne Links systematisch vernachlässigt werden, so wäre nach obiger Analyse zu erwarten, dass die Elemente außerhalb dieser Blöcke durch signifikant größere Werte (dunkler im Bild) auffallen.

Es sind allerdings keine solchen Strukturen ausgeprägt. Tatsächlich befinden sind außerhalb der Diagonalen keine deutlich erkennbaren Strukturen. Die Diagonale selbst tritt hervor durch dunklere Farbwerte, also vanRossum-Distanzen, die sich in relevantem Maß von 0 unterscheiden.

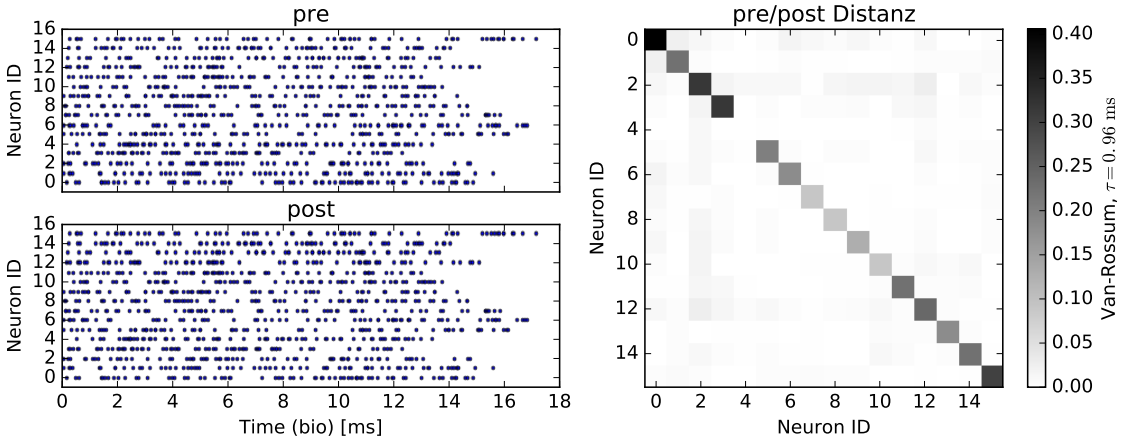

Abbildung 3.10: Biologisch orientierte Analyse der Übertragungsqualität. Links oben: Zeitpunkte der Input-Spikes für jedes Neuron als Rasterplot. Darunter: Zeitpunkte der Output-Spikes für jedes Neuron als Rasterplot. Die Output-Spikes wurden zeitlich um die Latenz λ des ersten Output-Spikes verschoben. Rechts im Bild: Die Diagonalelemente stellen für ein Neuron i die vanRossum-Distanz $d_{vR}(S_i, T_i)$ zwischen Input-Spikes S_i und Output-Spikes T_i dar. Die anderen Elemente entsprechen der Korrelationsfunktion $\delta(i, j)$ (vgl. Gl. 3.50) der zwei Neuronen. Der Skalenparameter der vanRossum-Distanz wurde auf $\tau = 0,96ms(bio)$ festgelegt.

Kapitel 4

Testergebnisse und Diskussion

Ziel dieses Abschnitts ist, die beschriebenen Testergebnisse zusammenzufassen und daraus geeignete Werte für die freien Parameter des Systems abzuleiten. Im Wesentlichen wird die korrekte Wahl der Bufferlängen l_{Ein} und $l_{Empfang}$, der Länge a_t des Zeitstempels und von Δt exemplarisch für die vorgegebenen Parameter des Testsystems beschrieben. Während die anderen Parameter zum Zeitpunkt der Hardwaresynthese festgelegt sind wird Δt frei kalibrierbar sein. Daher soll außerdem eine geeignete Kalibrationsprozedur für Δt beschrieben werden.

4.1 Durchsatz und Länge der Input-Buffer

Zuerst wurde die Übertragungsrate und Droprate der Schnittstelle untersucht. Dabei wurden verschiedene Tiefen l_{Ein} eines Input-Buffers getestet. Die in Abb. 3.2 gezeigten Messwerte entsprechen dem erwarteten Verlauf der Übertragungsrate:

Die Schnittstelle kann die volle Input-Rate bis unmittelbar unterhalb der kritischen Rate r_{krit} übertragen. Im Bereich unmittelbar um r_{krit} weicht die Durchsatzrate minimal von der idealen Übertragung ab, stärker für kurze Input-Buffer. Oberhalb der kritischen Rate kann das System die volle Auslastung der L2-Links gewährleisten.

Die Droprate (zu sehen in Abb. 3.3) ist vernachlässigbar für Input-Raten unterhalb einer Input-Rate von $r_e \approx 0.8 r_{krit}$. Bis $r_e \approx 0.9 r_{krit}$ ist eine gute Qualität der Übertragung gewährleistet. Ab $r_e \approx r_{krit}$ treten signifikante Dropraten auf. Für die Tiefe $l_{Ein} = 1$ des Input-Buffers treten Drops schon früher in relevantem Maß auf.

Insgesamt ist mit einem Input-Buffer der Länge 4 oder größer eine solide Unterdrückung der Droprate gewährleistet, wie in Tabelle 3.4 zu sehen ist. Die Input-Buffer müssen nicht als separate FIFO-Queue realisiert werden, prinzipiell ist jede Art blockender Bufferstufen, wie sie etwa im SPL1-Merger-Tree autreten, geeignet.

Ein oberes Limit für die Input-Rate, mit der eine gute Übertragung gewährleistet ist, ist $r_e \approx 0.8 r_{krit}$. Dieser Wert entspricht einer Input-Rate von $r_e = 80 MHz$ bzw. bei einem Beschleunigungsfaktor von $\varepsilon = 1000$ einer biologischen Rate von $r_e^{bio} = 80 kHz$. Bei $n_n = 500$ Neuronen entspricht das beispielsweise einer Spikerate von 160Hz pro Neuron.

4.2 Lastausgleich und Korrelationen

Als zweiter Test wurde die Lastausgleich-Funktion des SPL1-to-L2-Moduls analysiert. Wie in Abb. 3.5 zu sehen ist, wird keine bestimmte Kombination von Links bevorzugt, in allen Fällen kann die volle Rate der L2-Links genutzt werden. Das deckt sich mit den Ergebnissen der Korrelationsanalyse in Abschnitt 3.6, wo sich die Korrelation von Neuronen innerhalb einer SPL1-Gruppe nicht von der zwischen zwei SPL1-Gruppen unterscheidet. Das bedeutet, dass eventuelle Unterschiede in der Korrelation verschiedener Neuronenpaarungen rein statistischer Natur sind.

4.3 Droprate an den L2-Links

Im Anschluss wurde die Droprate an den L2-Links untersucht. Dabei wurde der Einfluss von r_e, $l_{Empfang}$ und Δt gestestet. Im Gegensatz zu den Drops an den Input-Buffern, die aufgrund des Bottlenecks an den L2-Links entstehen, können die Drops an den L2-Empfangsbuffern durch korrekte Abstimmung der freien Parameter $l_{Empfang}$ und Δt vollständig vermieden werden.

Die Dropraten an den L2-Links sind geringer für große Input-Raten, vor Allem dadurch bedingt, dass hier viele Spikes bereits an den SPL1-Input-Buffern gedropt werden. Da Drops an den L2-Links vollständig vermieden werden sollten ist allerdings interessanter, wann Drops überhaupt einsetzen. Der Grenzwert für Δt, an dem das passiert ist unabhängig von der Input-Rate. Für einen Empfangsbuffer der Länge $l_{Empfang} = 1$ liegt dieser Grenzwert bei $\Delta t = 44Takte$ (vgl. Tabelle 3.7), was im Rahmen der Genauigkeit der à priori-Schätzung von $\Delta t = 47Takte$ entspricht. Für jede Verlängerung des Empfangsbuffers um 1 wird der Grenzwert um $d = 20$ Takte erhöht.

Im folgenden Abschnitt wird die geeignete Wahl von Δt untersucht. Die Länge des L2-Empfangsbuffers muss dann den entsprechenden Wert für Δt unterstützen. Ein längerer Buffer hat keine Vorteile und führt lediglich zu vergrößertem Hardwareaufwand. Auch hier muss der Buffer nicht als separate FIFO-Queue implementiert werden, wenn entsprechende blockende Bufferstufen im L2-Empfangslayer bereits vorhanden sind.

4.4 Die Korrekte Wahl von Δt

Es wird nun also die korrekte Wahl von Δt beschrieben, woraus im Rückschluss auch ein Wert für $l_{Empfang}$ hervorgeht. Wie in den Histogrammen in Abb. 3.8 zu erkennen ist, ist die Jitter-Reduktion für große Raten nur bedingt effektiv. Dagegen kann durch die Wahl von $\Delta t = 52Takte$ bereits für eine Rate von $r_e^{rel} = 0.91$ der Anteil der Spikes mit einem Jitter $j > 30Takte = 120s(bio)$ zuverlässig auf unter 0,1% reduziert werden.

Anhand Abb. 3.9 kann die Reduktion des Jitters genauer analysiert werden. Für die Raten $r_e^{rel} = 0.6$ und $r_e^{rel} = 0.91$ wird der Jitter des Großteils der Spikes (mehr als 90%) auf unter 2 Takte reduziert. Während bei $\Delta t = 60Takte$ für $r_e^{rel} = 0.91$ noch etwa 0,1% der Spikes einen Jitter von $j > 30Takte = 120s(bio)$ besitzen, haben für den realistischeren Anwendungsfall einer Input-Rate von $r_e^{rel} = 0.6$ bereits mehr als 99,9% der Spikes einen Jitter $j < 3Takte = 12s(bio)$. Die Jitter-Reduktion funktioniert im realistischen Anwendungsfall niedriger Raten also deutlich besser als in den Extremfällen nahe der kritischen Rate.

Wie aus Abb. 3.9 hervorgeht, hat die Jitter-Reduktion bis $\Delta t \approx 45 Takte$ einen großen Effekt, bis $\Delta t \approx 60 Takte$ werden weiterhin verbleibende Ausreißer reduziert. Das entspricht der Abschätzung von $\Delta t_{max} \approx 57 Takte$ des erwarteten maximalen effektiven Werts. Wie die Maximumslinie zeigt, können einzelne Ausreißer nie vollständig ausgeschlossen werden.

Als Richtwert für das Testsystem wird im folgenden der Wert $\Delta t = 50 Takte$ gewählt, für den die Jitter-Reduktion bei realistischen Raten gute Ergebnisse liefert. Für das reale System werden die L2-Links allerdings eine andere Latenz und einen anderen Jitter besitzen. Dieser steht zu diesem Zeitpunkt noch nicht fest. Prinzipiell sind Schätzrechnungen wie in Abschnitt 3.5 immer möglich, aber die Komplexität des Systems und mehr noch die Tatsache, dass einige Bestandteile des Systems austauschbar sind, machen einen zuverlässigen Kalibrationsalgorithmus in Laufzeit notwendig.

4.5 Kalibration von $\triangle$t

Wie in Abschnitt 3.5 beschrieben besteht die Gesamtlatenz λ eines Spikes aus dem reduzierten Anteil λ^{red}, der zwischen dem Anheften und Auswerten des Zeitstempels entsteht und dem Restanteil λ^{rest}, der davor oder danach entsteht. Der reduzierte Anteil beträgt minimal λ^{red}_{min} und maximal λ^{red}_{max}. Der maximal sinnvolle Wert für Δt ist $\Delta t_{max} = \lambda^{red}_{max}$. Es ist dann zu empfehlen, $\Delta t > \lambda^{red}_{max} - 10 Takte$ zu wählen. Die korrekte Wahl für Δt hängt also nur von λ^{red}_{max} ab. Diese maximale Latenz kommt bei voller Auslastung der L2-Links vor, aber nicht unbedingt im normalen Anwendungsfall.

Es ist allerdings kaum möglich, den Anteil λ^{red}_{max} zu isolieren. Aus diesem Grund liegt nahe, die Kalibration anhand einer stochastischen Analyse ähnlich Abschnitt 3.5 durchzuführen. Es werden also im Idealfall Spikes der maximale verwendeten Input-Rate $r_e < r_{krit}$ generiert, nach Möglichkeit statistisch verteilt und ohne dabei Drops zu verursachen. Aus den Ein- und Ausgabedaten werden dann die Latenzen und der Jitter der Spikes berechnet. Für ein geeignetes Maß für den Gesamtjitter (z.B. 1%-Quantil, Mittelwert etc.) wird für steigendes Δt der Punkt ermittelt, ab dem der Gesamtjitter stationär wird, also nicht mehr sinkt. Das entspricht dem Knick in den Kurven in Abb. 3.9 bei $\Delta t \approx 60$. Dieser Punkt entspricht dann Δ_{max}, also der korrekten Wahl für Δt.

4.6 Länge des Zeitstempels und des Empfangsbuffers

Um die Größe a_t des Zeitstempels sowie die Länge $l_{Empfang}$ des Empfangsbuffers korrekt zu wählen muss $\Delta\lambda = \lambda^{red}_{max} - \lambda^{red}_{min}$ bekannt sein. In Abb. 3.9 entspricht λ^{red}_{min} dem ersten Knick bei $\Delta t \approx 7$, bei dem die Reduktion des Jitters einsetzt, und λ^{red}_{max} dem zweiten Knick bei $\Delta t \approx 60$. Da a_t und $l_{Empfang}$ allerdings bereits zum Zeitpunkt der Synthese bekannt sein müssen, kann hier eine Abschätzung nur mithilfe einer Berechnung oder Simulation anhand der finalen Daten des Chips und der Datenübertragung vorgenommen werden.

Eine gute Schätzung für die benötigte Länge $l_{Empfang}$ des Buffers geht dann aus

Formel 3.18 hervor:

$$l_{Empfang} \geq \frac{\Delta\lambda}{d} = \frac{\Delta\lambda}{20}. \tag{4.1}$$

$l_{Empfang}$ muss also linear mit $\Delta\lambda$ wachsen. Da der Wert für $\Delta\lambda$ bei der Übertragung via FPGA (vgl. Abb. 1.2) durchaus groß ausfallen kann, ist unter Umständen die umgekehrte Betrachtung sinnvoll: $l_{Empfang}$ wird so groß gewählt, wie die Hardware-Ressourcen es zulassen, damit ist ein maximal möglicher einstellbarer Wert für Δt fest vorgegeben. So ist der Reduktion des Jitters zwar ein Limit gesetzt, die Vermeidung von Datenverlust durch dieses Limit hat dabei aber eindeutig Vorrang.

Die Länge a_t des Zeitstempels muss groß genug gewählt werden, dass unabhängig von der Wahl von Δt und der Latenz λ eines Spikes der Zeitabgleich (vgl. Gl. 2.5) $t \geq t_{timestamp} + \Delta t$ eindeutig durchgeführt werden kann. Der vorzeichenlose Zeitstempel der Länge a_t kann maximal den Wert $t^{max}_{timestamp} = 2^{a_t} - 1$ annehmen. Um eine Spikes, die sehr früh ankommen, eindeutig von sehr späten Spikes unterscheiden zu können muss $t^{max}_{timestamp} \geq 2\Delta\lambda$, also

$$a_t > log_2(\Delta\lambda) + 1 \tag{4.2}$$

als Länge des Zeitstempels gewählt werden.

Abb. 4.1 zeigt schematisch, wie der Zeitabgleich mit einem Zeitstempel der Länge a_t durchgeführt wird. Blau gekennzeichnet ist der Bereich der Latenzen $\lambda^{red}_{min} \leq \lambda \leq \lambda^{red}_{max}$, die auftreten können. Beispielhaft für $a_t = 5$, $a_t = 6$ wird gezeigt, Spikes welcher Latenzen am Zeitabgleich als zu früh erkannt werden (gelb) und Spikes welcher Latenzen als verspätet erkannt werden (grün). Bei korrekter Wahl des Zeitstempels werden nur Latenzen $\lambda > \Delta t$ als verspätet erkannt. In diesem Beispiel ist also ein Zeitstempel der Länge $a_t = 5$ zu kurz.

Für die Messwerte des Testsetups ist $\Delta\lambda \approx 50$. Damit wäre ein Zeitstempel der Länge $a_t \geq 7$ nötig. Ein Zeitstempel der Länge $a_t = 8$ bietet vermutlich genügend Spielraum für weitere Jitter-Quellen im realen System. Da a_t im Gegensatz zu $l_{Empfang}$ logarithmisch mit $\Delta\lambda$ wächst, ist die Wahl von a_t weniger kritisch als die von $l_{Empfang}$.

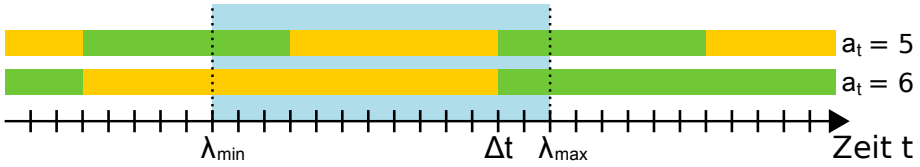

Abbildung 4.1: Veranschaulichung des Zeitabgleichs für dir Längen $a_t = 5$ und $a_t = 6$ des Zeitstempels. Blau markiert: Bereich der Latenz λ, mit der Spikes eintreffen können. Gelb markiert: Spike wird als zu früh erkannt und zurückgehalten. Grün markiert: Spike wird als verspätet erkannt und sofort ausgeliefert. In diesem Beispiel ist ein Zeitstempel der Länge $a_t = 5$ nicht ausreichend, um eine eindeutige Unterscheidung von frühen und sehr späten Spikes zu gewährleisten.

4.7 Zusammenfassung

Die Übertragungs- und Droprate des entwickelten Routing-Systems entspricht vollständig den Erwartungen. Die Schnittstelle kann die volle Bandbreite der L2-Links

ausnutzen, unabhängig davon, in welcher Kombination die SPL1-Links senden. Damit gehen erst im Bereich der kritischen Input-Rate $r_e \approx r_{krit} = r_2^{max}$ Spikes verloren. Für Input-Raten $r_e < 0.8 r_{krit}$ kann eine nahezu fehlerfreie Übertragung gewährleistet werden. Als geeignete Länge für die SPL1-Input-Buffer wurde $l_{Ein} = 4$ beobachtet.

Durch eine geeignete Wahl des freien Parameters Δt kann der Jitter der Übertragung deutlich reduziert werden. Für realistische Input-Raten sinkt der Jitter dann für über 99,9% der Spikes auf unter 3 Takte bzw. 12µs(bio). Die korrekte Wahl für Δt wird am besten über eine Kalibration im realen System ermittelt. Der optimale Wert für Δt entspricht dann dem Punkt, ab dem eine weitere Erhöhung von Δt den Jitter nicht weiter senkt.

Die korrekte Wahl der Länge a_t des Zeitstempels und der Länge $l_{Empfang}$ des L2-Empfangsbuffers hängt von der Differenz $\Delta\lambda = \lambda_{max}^{red} - \lambda_{min}^{red}$ der minimal und maximal möglichen Latenz zwischen Anheften und Auswerten des Zeitstempels. Diese Differenz ist schwer zu berechnen und muss vor Fertigstellung des Chips bekannt sein, da $l_{Empfang}$ und a_t zum Zeitpunkt der Synthese festgelegt sein müssen. Ein sinnvollerer Ansatz ist, anhand Überschlagsrechnungen und der verfügbarer Hardwareressourcen einen Wert für $l_{Empfang}$ festzulegen. Damit ist dann ein oberes Limit für die Wahl von Δt gesetzt.

Die Wahl der Länge a_t des Zeitstempels ist weniger problematisch und sollte anhand von Überschlagsrechnungen für $\Delta\lambda$ ermittelt werden. Der hier vorgeschlagene Wert von $a_t = 8$ deckt alle Fälle ab, in denen die reale Datenübertragung den hier gemessenen Jitter maximal verdoppelt.

Kapitel 5

Ausblick

In diesem Kapitel soll ein kurzer Überblick über bereits existierende und geplante Weiterentwicklungen des Routing-Systems und des HICANN-DLS-SR im Allgemeinen gegeben werden.

5.1 Implementationsaspekte

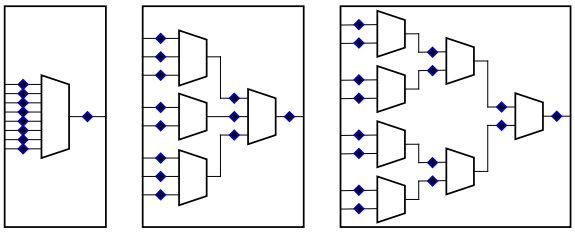

Abbildung 5.1: Möglichkeiten, einen 8-zu-1-Multiplexer durch 1, 2 oder 3 Stufen zu realisieren. Registerstufen sind blau gekennzeichnet.

Timing-Constraints

Die präsentierte Struktur der Module funktioniert zwar ohne Anpassung in Simulation. Das Ziel ist allerdings, Hardware zu produzieren. Hier bestehen Timing-Constraints, die die Setup-Zeit der kombinatorischen Schaltungen einschränken. Die genauen Constraints sind zwar noch nicht bekannt, allerdings ist absehbar, dass die Arbiter der Größe n_2-zu-1 bzw. 1-zu-n_2 nicht direkt in einem Takt realisierbar sind. Hier gibt es zwei einfache Lösungsmöglichkeiten. Zum Einen könnte eine Arbiterlogik eingesetzt werden, die 2 Takte für den entsprechenden Schritt braucht. Der Nachteil ist, dass damit die mögliche Übertragungsrate reduziert wird. Zum Anderen könnte der Arbiter durch eine Baumstruktur mit mehreren Stufen kleinerer Arbiter realisiert werden, wie in Abb. 5.1 veranschaulicht wird. Ähnliches gilt für die Adressierunglogik (1-zu-n_1 Demultiplexer).

Behandlung von Overflows

Die Behandlung von Buffer-Overflows wurde bislang nicht näher betrachtet. Bei zu hohen Spikeraten sind Overflows unvermeidbar. Aber auch bei subkritischen Spikeraten kann r_{krit} durch statistische Schwankungen temporär überschritten werden, sodass Overflows letztlich nicht vollständig vermeidbar sind.

In der aktuellen Veriante werden Nachrichten gedropt, sobald die Buffer voll sind. Das liegt am Bottleneck der L2-Links, also ist die einfachste Möglichkeit die Verwendung schnellerer L2-Links. Innerhalb der Chips kann dagegen wenig getan werden. Die Drosselung des Gesamtsystems wäre hier die einzige Alternative.

5.2 Varianten der designten Module

Für die vorgestellten Module SPL1-to-L2 und L2-to-SPL1 wurden im Zuge dieser Arbeit mehrere Varianten entwickelt. Die folgenden Varianten wurden erfolgreich verifiziert, aber nicht im selben Maße getestet und charakterisiert.

L2-to-SPL1 serialisiert

Die Verbindungsmatrix im L2-to-SPL1-Modul hat einen großen Hardwareverbrauch. Analog zum SPL1-to-L2-Modul können die ankommenden Nachrichten nach dem Zeitabgleich serialisiert werden, ohne die Bandbreite zu drosseln.

Das L2-to-SPL1-Modul verbraucht bei den verwendeten Parametern für alle Bufferstufen und den internen Zustand insgesamt 720 Register. Abb. 5.2 zeigt die serialisierte Variante, die den Hardwareaufwand auf 434 Register reduziert, was einer Einsparung von 40% entspricht. Dafür erzeugt die Serialisierung zusätzlichen Jitter, der nicht kompensiert werden kann: Kommen 8 Nachrichten mit gleichem Zeitstempel gleichzeitig an, so wird nur eine von ihnen zum vorgegebenen Zeitpunkt ausgeliefert, die anderen kommen mit 1 bis 7 Takten Verspätung an.

Letztlich hängt die Entscheidung für eine der beiden Alternativen von den verfügbaren Hardwareressourcen ab.

SPL1-to-L2 mit Gruppierungsoption

Um mit mehr als einem anderen Chip kommunizieren zu können müssen die physischen L2-Links auf die anderen Chips im Netzwerk verteilt werden. Nach dem aktuellen Konzept der hier vorgestellten Variante können die L2-Links in k Gruppen mit $k = 2^g$ aufgeteilt werden, wobei die L2-Links in einer Gruppe an den selben Chip gehen.

$g = 0, k = 1$ entspricht dem bekannten Fall, dass alle L2-Links in der selben Gruppe sind, also genau 2 Chips im Netzwerk. $g = 2, k = 4$ entspricht dem Fall, dass die L2-Links in 4 Gruppen mit je 2 L2-Links gruppiert sind.

Um die verschiedenen Chips auch einzeln adressieren zu können, werden die SPL1-Links ebenfalls in k Gruppen aufgeteilt, die den Gruppen der L2-Links entsprechen. Die SPL1-Links einer Gruppe können dann nur an die L2-Links der selben Gruppe senden. Offensichtlich ist die maximale Anzahl der Gruppen also $k_{max} = min(n_1, n_2)$.

Spaltet man den n_2-zu-1-Arbiter des SPL1-to-L2-Moduls wie in Abb. 5.1 dargestellt in mehrere Stufen von 2-zu-1-Arbitern auf, so ist die Implementation mit minimalem

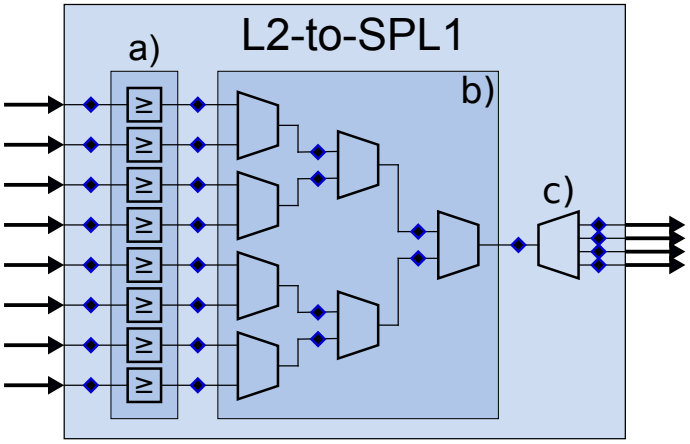

Abbildung 5.2: Serialisierte Veriante des L2-to-SPL1-Moduls (vgl. Abb. 2.3). a) Zunächst findet unverändert der Zeitabgleich statt. b) Im Anschluss werden die an den $n-2$ L2-Links parallel ankommenden Nachrichten serialisiert. c) Ein Multiplexer verteilt die serialisierten Nachrichten entsprechend ihrer Adressen an die SPL1-Links

zusätzlichem Hardwareaufwand möglich. Wie in Abb. 5.3 gezeigt wird die Stufe g des Merger-Baumes durch eine Bypass-Schaltung mit der Stufe g des Arbiter-Baumes verbunden. Der Mehraufwand in Hardware ist dann lediglich ein Multiplexer am Ende jedes Bypasses, also insgesamt 6 2-zu-1-Multiplexer.

Zusätzlich dazu ist allerdings auch zusätzliche Routing-Logik auf SPL1-Ebene erforderlich. Aus diesem Grund hängt die Implementation der Gruppierungsoption von den verfügbaren Hardwareressourcen ab.

5.3 Die weitere Entwicklung

Schnellere L2-Links

Diese Arbeit betrachtet eine Bandbreite der L2-Links von jeweils 1GHz. Das entspricht bei aktuellem Entwicklungsstand einer Verzögerung der L2-Links von $d \approx 20Takte$. Die Bandbreite ist durch den angeschlossenen FPGA-Chip limitiert. Wird der FPGA durch ein schnelleres Modell ersetzt, kann die Bandbreite der L2-Links auf 1GHz erhöht werden. Das entspricht dann einer Verzögerung der L2-Links von $d \approx 10Takte$. Auch für diesen Wert kann ein einzelner SPL1-Link alle L2-Links auslasten. Diese Rate stellt allerdings deutlich größere Ansprüche an das interne Pipelining der Routing-Module. Für diese Bandbreite sollten die Module dann nach dem selben Schema getestet werden.

Mit der erhöhten Bandbreite wäre die kritische Übertragungsrate dann $r_{krit} = \frac{r_e^{max}}{5} = 200MHz = 200kHz(bio)$. Neben der doppelten Übertragungsrate ist mit den schnelleren L2-Links auch ein geringerer Jitter möglich, da weniger Nachrichten auf einen freien L2-Link warten müssen. Gemäß Gleichung 4.1 wären dann aber auch Tiefere L2-Empfangsbuffer nötig, die dafür eingeplant werden müssen.

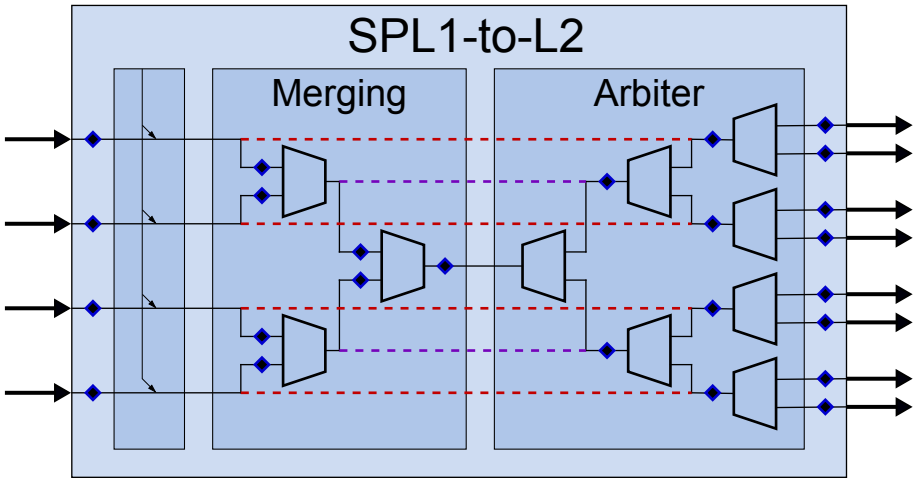

Abbildung 5.3: SPL1-to-L2-Modul mit Gruppierungsoption (vgl. Abb. 2.2). Der Arbiter am L2-Ausgang wurde als Baumstruktur realisiert. Zwischen den Bamstrukturen der Serialisierung und des Arbiters wurden Bypass-Schaltungen eingefügt. Statt vollständiger Serialisierung kann nun die Aufteilung von SPL1- und L2-Links in Gruppen gewählt werden, sodass die Kommunikation mit mehreren Partnerchips möglich ist. Wird keine der Bypass-Schaltungen aktiviert, so ist die Funktionalität unverändert.

Direktverbindung von HICANN-DLS-SR-Chips

Wie in Abb. 1.2 gezeigt, erfolgt die Übertragung von Spikes aktuell immer über einen Host-FPGA. Das ist ineffizient und kann vermieden werden, wenn die direkte Verbindung von mehreren HICANN-Chips möglich ist. Dieses Feature wird vermutlich noch nicht in den kommenden HICANN-DLS-SR integriert ist aber für zukünftige Iterationen des Chips interessant. Der Vorteil ist zum Einen, dass Latenz und Jitter deutlich reduziert werden, da kein zusätzlicher Chip in die Übertragung eingebunden werden muss. Zum Anderen können dann größere Netzwerke deutlich größere Übertragungsraten erreichen, da aktuell die Portzahl des FPGAGA-Chips ein limitierender Faktor ist. Für genau diesen Anwendungsfall wurde die Gruppierungsoption in Abschnitt 5.2 entwickelt.

Kenngrößen

Kenngröße	Formelzeichen	Wert in den Testfällen	Wert im Chip (aktueller Stand)
Gesamtzahl Neuronen	n_n	256 bzw. 16	≤ 512
Anzahl SPL1-Links	n_1	4	4
Anzahl L2-Links	n_2	8	8
Verzögerung der L2-Links	d	20	20 bzw. 10
Taktfrequenz	f_T	250MHz	250MHz
Taktdauer	$T_T = \frac{1}{f_T}$	4ns	4ns
Beschleunigungsfaktor	ε	1000	1000
Kombinierte Bandbreite der L2-Links	$r_2^{max} = \frac{n_2}{d}$	$\frac{2}{5}\frac{Nachrichten}{Takt}$	$\frac{2}{5}$ bzw. $\frac{4}{5}\ \frac{Nachrichten}{Takt}$
Kombinierte Bandbreite der SPL1-Links	$r_1^{max} = n_1$	$4\frac{Nachrichten}{Takt}$	$4\frac{Nachrichten}{Takt}$
Länge des Zeitstempels	a_t	16	8
Länge der Adresse	a_n	8 bzw. 4	14
Gesamtlänge einer Nachricht	$a = a_t + a_r$	24 bzw. 20	22
Länge des Input-Buffers	l_{Ein}	variabel	noch nicht entschieden
Länge des Empfangsbuffers der L2-Links	$l_{Empfang}$	variabel	noch nicht entschieden

Begriffe und Abkürzungen

Arbiter Als Arbiter (Schiedsrichter) bezeichnet man eine Funktionalität, die aus n konkurrierenden Eingängen einen auswählt. 9–11, 37–41

ASIC Ein Application Specific Integrated Circuit (ASIC) integriert elektronische Schaltungen in das Grundmaterial (typischerweise Silizium) durch teilweisees Entfernen des Grundmaterials, Auf und Einbringen von zusätzlichen Komponenten. Damit ist die Funktionalität eines ASICs zum Zeitpunkt seiner Fertigstellung unveränderbar vorgegeben. 4, 5, 41

Bottleneck Als Bottleneck (Engstelle) eines Systems wird der Teil eines Systems bezeichnet, der am langsamsten arbeitet und so die Gesamtgeschwindigkeit des Systems limitiert. 1, 8, 16, 21, 24, 33, 38, 41

Drop Als Drop (Verlust) von Daten bezeichnet man eine Situation, in der Daten verloren gehen. Geschieht typischerweise im Fall eines Overflows. 13, 15–17, 19, 21–24, 29, 32, 33, 41

FPGA Ein Field Programmable Gate Array (FPGA) ist ein integrierter Schaltkreis, der die Konfiguration seiner Schaltungen ermöglicht. 4, 5, 39–41

HICANN High Input Count Artificial Neural Network (HICANN) ist ein teilanaloger ASIC, der Neuronale Netze auf Hardwareebene durch Wafer Scale Integration ermöglicht. 3–5, 40, 41

HICANN-DLS HICANN with Digital Leaning (HICANN-DLS) ist der aktuellste Chip der der HICANN-Reihe. 4, 41, 42

HICANN-DLS-SR HICANN-DLS Spikey Replacement (HICANN-DLS-SR) befindet sich aktuell in der Entwicklung und ist als Einzelchip geplant. first 1, 3–5, 37, 40, 41

Jitter Jitter bezeichnet Schwankungen in der Latenz einer Datenübertragung. 1, 5, 8, 10, 13, 15, 23–26, 28, 29, 33–35, 38, 40, 41

Lastausgleich Lastausgleich (Load Balancing) bezeichnet die Technik, Aufgaben mit großem Arbeitsaufwand gleichmäßig auf parallel arbeitende Systeme zu verteilen, welche die Aufgaben dann insgesamt schneller bearbeiten können 8, 9, 19, 29, 41

Mergesort Der Mergesort-Algorithmus sortiert Daten, indem sortierte Teilmengen der Daten zusammengeführt (gemerget) werden. 9, 41

Neuromorphe Hardware Neuromorphe Hardware ist hardware spezialisiert auf die Verwirklichung großer neuronaler Netze. 3, 4, 41

Neuronale Netze Das Neuronale Netz (NN) ist das abstrahierte Konzept hinter der Signalübertragung im Gehirn. 3, 4, 41

Overflow Als Overflow (Überlauf) bezeichnet man eine Situation, in der Daten in ein bereits vollständig gefülltes System geschrieben werden sollen. 13, 23, 38, 41

Pipelining Pipelining bezeichnet die Technik, Berechnungen, die mehrere Takte benötigen, in Einzelschritte aufzuteilen, die jeweils nur einen Takt benötigen. Damit wird zwar nicht die Zeit zwischen Eingabe und Ausgabe von Daten verringert, es ist aber möglich, jeden Takt Daten in die Berechnungsschaltung einzugeben und auszulesen. 39, 41

Queue Eine Queue (Warteschlange) ist ein Datencontainer, der Daten nach dem first in first out (FIFO)-Prinzip verwaltet. 13, 32, 33, 41

Seed Als Seed bezeichnet man den Ausgangswert eines Pseudozufallsgenerators. 15, 41

Serialisierung Serialisierung bezeichnet den Prozess, parallel eingehende Datenströme in einen einzelnen linearen Datenstrom zusammenzuführen. 9, 38, 40, 41

Spike Das Aktionspotential (Spike) ist ein diskreter, spitzer Ausschlag des Membranpotentials eines Neurons. 3, 5, 7–9, 13–17, 19, 21, 23–27, 29, 33–35, 40, 41

Spikey Teilanaloger neuromorpher Hardwarechip, designt als Einzelchip. 4, 41, 42

Synthese Als Hardwaresynthese wird der Prozess bezeichnet, aus abstrakt beschriebenen Modulen und Schaltungen einen Bauplan für die konkrete Struktur und Anordnung der Hardwarebausteine zu formen. 32, 34, 36, 41

Wafer Scale Integration Mikrochips werden üblicherweise in großen, dünnen Siliziumplatten (Wafern) produziert und anschließend ausgeschnitten. Wafer Scale Integration bezeichnet die Technik, keine Einzelchips auszuschneiden und stattdessen die gesamte Siliziumplatte als großen Makrochip zu verwenden. 4, 41

Literaturverzeichnis

[1] Oliver Breitwieser. Comissioning of an fpga-based prototyping enviroment for neuromorphic hardware. Masterarbeit, Ruprecht-Karls-Universität Heidelberg, 2016.

[2] Jan Debus. Towards a neuromorphic implementation of spike-based expectation maximization. Bachelorarbeit, Ruprecht-Karls-Universität Heidelberg, 2015.

[3] Kai-Uwe Sattler Gunter Saake. Algorithmen und Datenstrukturen. dpunkt Verlag, Heidelberg, 5 edition, 2014.

[4] Vitali Karasenko. A communication infrastructure for a neuromorphic system. Masterarbeit, Ruprecht-Karls-Universität Heidelberg, 2014.

[5] Ulrich Krengel. Einführung in die Wahrscheinlichkeitstheorie und Statistik. Vieweg+Teubner Verlag, Wiesbaden, 8 edition, 2010.

[6] Thomas Pfeil, Andreas Grübl, Sebastian Jeltsch, Eric Müller, Paul Müller, Mihai A. Petrovici, Michael Schmuker, Daniel Brüderle, Johannes Schemmel, and Karlheinz Meier. Six networks on a universal neuromorphic computing substrate. Frontiers in Neuroscience, 7:11, 2013.

[7] J. Schemmel. Brainscales 2: A novel architecture for analog accelerated neuromophic computing including hybrid plasticity. in preparation.

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

[9] Johannes Schemmel, Daniel Brüderle, Andreas Grübl, Matthias Hock, Karlheinz Meier, and Sebastian Millner. A wafer-scale neuromorphic hardware system for large-scale neural modeling. In Proceedings of the 2010 IEEE International Symposium on Circuits and Systems (ISCAS), pages 1947–1950, 2010.

[10] M. C. W. van Rossum. A novel spike distance. Neural Computation 13, 751-763, 2001.

Danksagung

Vitali Karasenko,
für Beratung und Diskussion zu inhaltlichen Themen, für gute Betreuung, für viel aufgebrachte Zeit und Mühe, besonders in den letzten Wochen dieser Arbeit.

Prof Dr. Karlheinz Meier,
der diese Arbeit überhaupt erst möglich gemacht hat.

Dr. Johannes Schemmel,
für Betreuung und Beratung in technischen Fragen zum Chip.

Gerd Kiene und Arthur Heimbrecht,
fürs Korrekturlesen.

Christian Mauch,
Für Beratung und Unterstützung bei Software-Fragen und fürs Korrekturlesen.

Alexander Kugele, Arthur Heimbrecht und David Stöckel,
Für die Teilnahme am Probevortrag des Kolloquiums und konstruktive Vorschläge.

Oliver Breitwieser, Paul Müller und Dr. Mihai Petrovici,
Für Vorschläge und Beratung zur Struktur und Analyse der Tests.

Laura Kriener,
Für Hilfe mit dem Textsatzprogramm Tex.

Dr. Eric Müller,
Für Beratung und Unterstützung bei Fragen zu Software.

Die gesamte Electronic Vision(s)-Gruppe,
für eine freundliche und hilfsbereite Arbeitsumgebung und eine gute Zeit auch über das Büro hinaus.