Technische Universität München
Fakultät für Physik

Abschlussarbeit im Bachelorstudiengang Physik

Entwicklung eines CPLD-basierten Trigger-Bus-Systems für das CALIFA Kalorimeter

Florian Kurz

8. August 2011

Erstgutachter (Themensteller): Prof. R. Krücken
Zweitgutachter: Prof. L. Oberauer

Zusammenfassung

In der vorliegenden Arbeit wird das System zur Erzeugung eines globalen Energie- und Multiplizitätstriggersignals für das CALIFA Kalorimeter vorgestellt, das im R^3B Experiment an der GSI in Darmstadt zum Einsatz kommen wird. Durch die anspruchsvolle Vorgabe für die Signalverarbeitungszeit von $1\mu s$ wurde dieses System auf eine neuartige Weise durch asynchrone Logik realisiert.

Im Rahmen der Bachelorarbeit wurde eine Kleinserie von Prototypmodulen gefertigt und die Entwicklung der Software des Triggersystems abgeschlossen. Darüber hinaus wurde Software entwickelt, die es erlaubt, das System im Labor zu testen. Mit dem Prototypenaufbau wurde das Design auf seine Funktions- und Leistungsfähigkeit hin untersucht.

Das System konnte erfolgreich in Betrieb genommen und die generelle Machbarkeit des Ansatzes unter Beweis gestellt werden. Um das System zur Einsatzreife zu führen, wurde ein Konzept zur Erkennung von Übertragungsfehlern ausgearbeitet und Verbesserungsvorschläge für die Weiterentwicklung der Elektronikkomponenten gesammelt

Inhaltsverzeichnis

Kapitel 1

Einleitung

1.1 Das Kalorimeter im R3B Experiment

Streuexperimente bei relativistischen Energien sind seit vielen Jahren ein sehr erfolgreiches Mittel um die Struktur von exotischen, radioaktiven und meißt kurzlebigen Atomkernen zu untersuchen. Aus diesem Grund wird am GSI Helmholtzzentrum für Schwerionenforschung in Darmstadt zur Zeit die **F**acility for **A**ntiproton and **I**on **R**esearch (FAIR) aufgebaut. Ein Schema der Anlage zeigt Abbildung 1.1.

Den Kern der Anlage, die die bisherigen Einrichtungen an der GSI erweitert, bilden die beiden supraleitenden Synchrotronringe SIS100 und SIS300 mit magnetischen Steiffigkeiten von 100 Tm bzw. 300 Tm. Diesen nachgelagert sind der Fragmentseparator Super-FRS, mehrere Speicherringe und eine Vielzahl an unterschiedlichen Experimentiereinrichtungen. (Gut06)

Im Brennpunkt des Hochenergiezweigs des Super-FRS befindet sich das R^3B (A next generation experimental setup for studies of **R**eactions with **R**elativistic **R**adioactive **B**eams) Experiment (siehe Abbildung 1.2), dessen Schwerpunkt auf der Erforschung der Kernstruktur- und Dynamik liegt. Es ist dazu für eine Vielzahl von Reaktionstypen mit radioaktiven Strahlen und stabilen Targetkernen konzipiert. Darunter sind elastische und inelastische Streuung, Coulombanregung, Knockout Reaktionen, Ladungsaustauschreaktionen, Spaltungs- und Spallationsexperimente.

R^3B baut dabei auf dem ALADIN-LAND Setup auf, das an der GSI bereits erfolgreich betrieben wird, um Reaktionen mit instabilen Kernen zu untersuchen.

Bei allen Reaktionstypen sollen dabei kinematisch vollständige Messungen mit hoher Effizienz, Akzeptanz und Auflösung vorgenommen werden. Um dieses Ziel zu erreichen, benötigt man insbesondere Detektorsysteme, die die leichten geladenen Teilchen und Photonen, die aus dem Target austreten, nachweisen.

Einer dieser Detektoren ist das **CAL**orimeter for **I**n**F**light emitted gammas and light charged p**A**rticles for R3B (CALIFA). Er deckt fast den gesamten Raumwinkel um das Target von R^3B ab und soll die angesprochenen Teilchen mit möglichst großer Winkel- und Energieauflösung nachweisen und identifizieren. Dazu besteht der Detektor aus etwa 6000 CsI(Tl) Szintillationskristallen auf die APD (**A**valanche **P**hoto**D**iodes) aufgeklebt sind. Jeweils 16 davon sind mit einem Sampling-ADC-

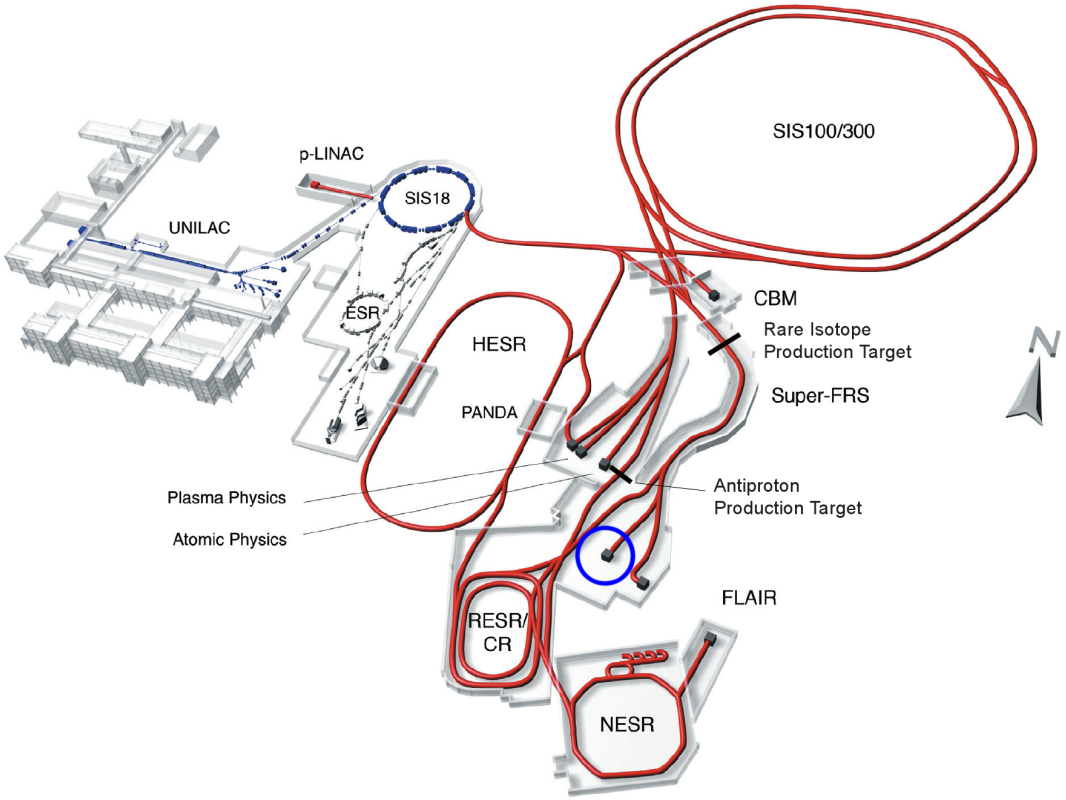


Abbildung 1.1: Schematische Zeichnung von FAIR. In rot sind diejenigen Teile der Anlage eingezeichnet, die sich im Bau bzw. in Planung befinden. Die blau eingefärbten Einrichtungen sind schon an der GSI vorhanden. R^3B befindet sich an der mit dem blauen Kreis markierten Stelle. (Gut06)

Modul verbunden, das deren Signale kontinuierlich digitalisiert und weiterverarbeitet. (TSR08) Genauere Informationen zu den Sampling-ADC-Modulen finden sich in (Hof10), (Hof11a) und (Hof11b).

1.2 Das Triggersystem für CALIFA

Da am R^3B Setup eine große Zahl unterschiedlicher Experimente durchgeführt werden soll, müssen alle Komponenten möglichst flexibel gestaltet werden. Für den CALIFA Detektor heißt das insbesondere, dass je nach Anwendung eine Vielzahl von verschiedenen Triggersignalen, basierend auf Energie- bzw. Multiplizitätsinformation, generiert werden können muss.

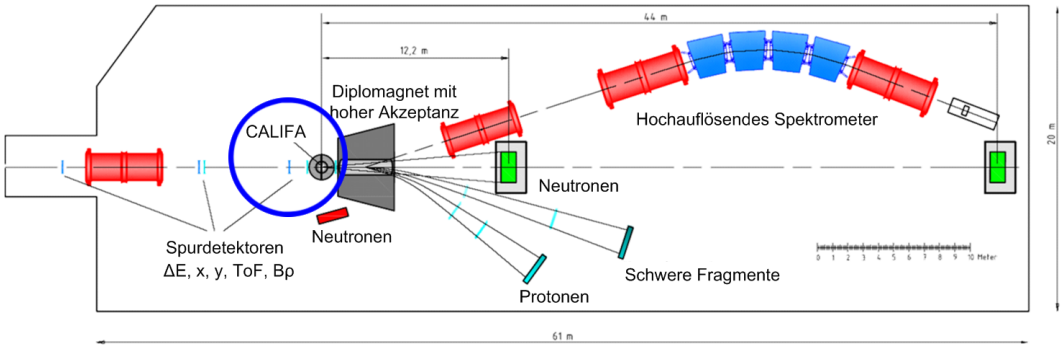


Abbildung 1.2: Geplanter Aufbau des R^3B Experiments. In der Mitte erkennt man den Dipolmagneten, vor dem CALIFA (blauer Ring) platziert ist. Je nach Ablenkung im Magneten werden die leichten geladenen Teilchen und schweren Fragmente entweder in ein Magnetspektrometer (oben) geleitet oder mit voller Raumwinkel-Akzeptanz detektiert (unten). (Ben10)

Dazu ist der Detektor in verschiedene Level, die vom einzelnen Kristall bis zum gesamten Detektor reichen, aufgeteilt. Jedes Level enthält dabei eine gewisse Anzahl von Einheiten des Levels darunter. Auf jedem Level soll ein Trigger ausgelöst werden können, wenn ein Einzelsignal des Levels darunter einen Schwellenwert überschreitet. Darüber hinaus soll auch eine Triggerentscheidung auf Grundlage der Summeninformation des darunterliegenden Level getroffen werden können.

Diese Arbeit konzentriert sich auf den globalen Summenenergie und -multipliziätstrigger. Die Aufgabe dieses Systems ist es, die Summenenergie und -multiplizitätsinformation des gesamten Detektors zu sammeln und auf dieser Basis eine Schwellwertentscheidung zu treffen. Diese Information wird von den Samplig-ADC-Modulen (im Folgenden IP für Information Provider genannt) geliefert, die ebenfalls für CALIFA entwickelt werden.

1.3 Physikalische und technische Anforderungen

Nachdem FAIR komplett neu konzipiert und aufgebaut wird, sollen alle elektronischen Komponenten vollständig digital ausgelegt werden. Jedoch ist geplant, dass in der Aufbau- und Entwicklungsphase auch existierende, analoge Datenanalysesysteme zum Einsatz kommen werden.

Für das Bussystem, das den globalen Summenenergie und -Multipliziätstrigger generieren und transportieren soll, entstehen daraus anspruchsvolle technische Anforderungen.

Viele der vorhandenen Systeme sind so konzipiert, dass sie das Triggersignal

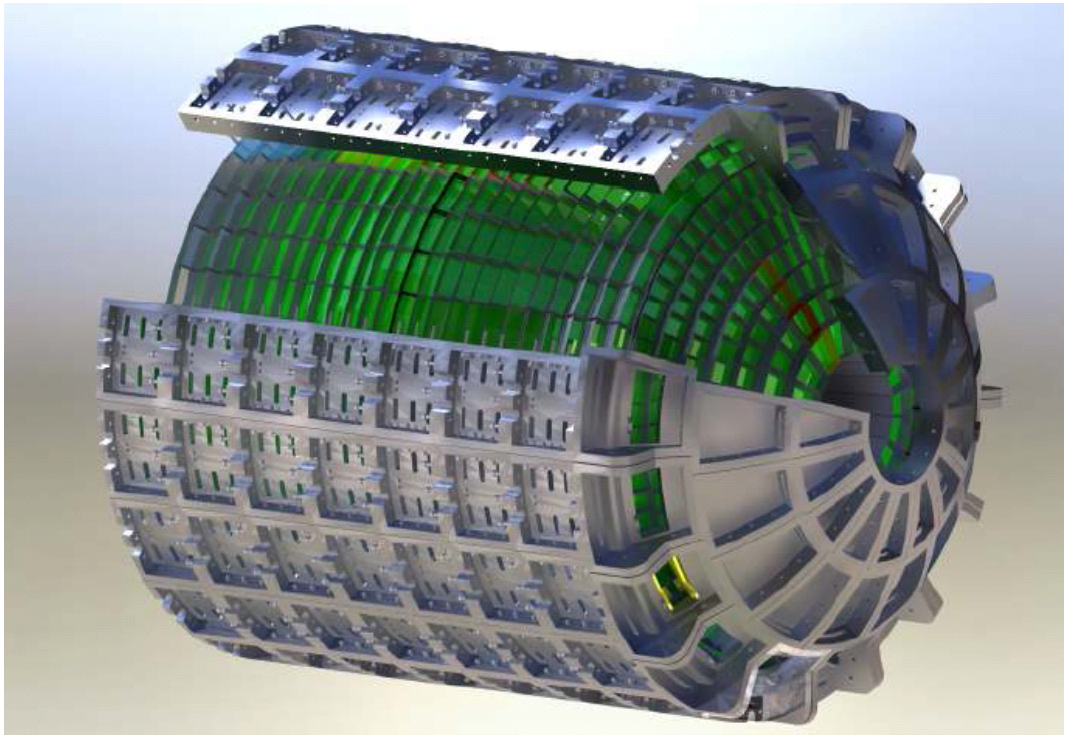


Abbildung 1.3: Mechanischer Aufbau des CALIFA Detektors. Man erkennt die zwei Hauptbestandteile des Detektors, die Front Cap und Barrel genannt werden, sie bestehen aus je etwa 3000 Szintillationskristallen. (TSR08)

höchstens 1μs nach dem Event erwarten. Dies kann bei den meißten Modulen auch nicht auf einfache Weise geändert werden. Digitale Bussysteme weisen jedoch im Allgemeinen wesentlich größere Latenzen auf, da diese die Signale vollständig getaktet, also synchron, verarbeiten.

Es muss also eine alternative Lösung gefunden werden, die es erlaubt ein möglichst genaues, digitales Triggersignal für Summenenergie und -Multiplizität innerhalb einer Mikrosekunde nach dem Event zu erzeugen und zum Auslesesystem zu transportieren.

Kapitel 2

Der globale Summen-Energie und -Multiplizitäts Trigger

Um die oben beschriebenen Anforderungen mit einem vollständig digitalen System erfüllen zu können, entwarf Max Winkel eine neuartige Lösung. Die Neuartigkeit des Ansatzes liegt im wesentlichen in der Verwendung von asynchroner Logik in einem Bus. Von solchen Designs wurde bisher abgesehen, da sie extrem empfindlich auf Störungen der zeitlichen Korrelation der Signale reagieren. In diesem Fall verspricht jedoch nur eine solche Lösung die Einhaltung des vorgegebenen Zeitfensters.

2.1 Funktionsweise

Das Triggersystem besteht aus einer bzw. mehreren Ketten von Triggermodulen. Die Module sind dabei über einen, vier Leitungen umfassenden, Bus verbunden. Die Leitungen transportieren ein globales Taktsignal (CLK), ein Synchronisierungssignal (BIT0) sowie jeweils eine serielle Busleitung für die Energie und die Multiplizität. Abbildung 2.1 zeigt eine Übersicht über die Busstruktur.

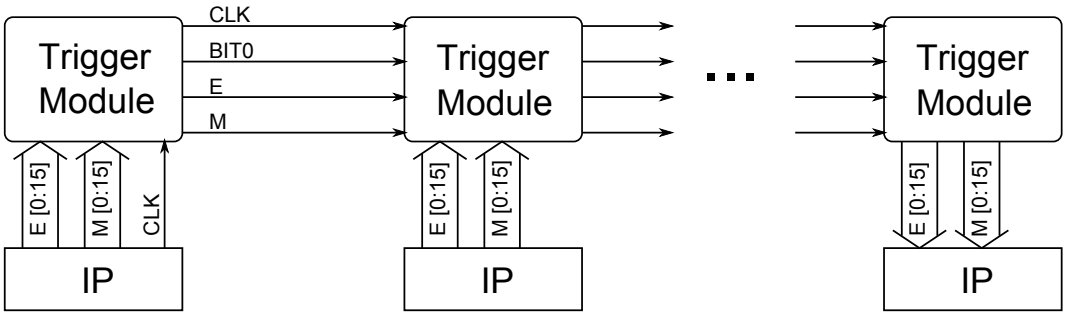
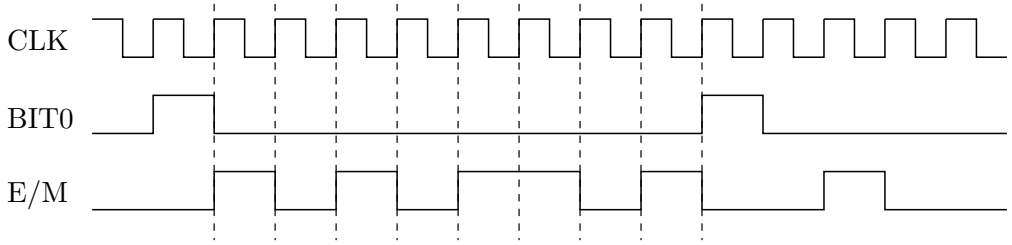


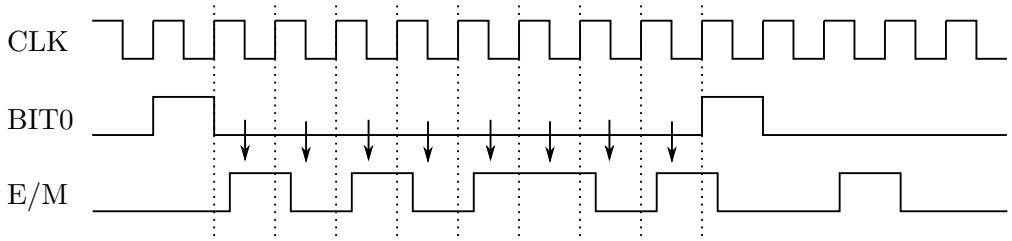
Abbildung 2.1: Der Triggerbus besteht aus einer Kette von Modulen, die die Werte ihres Information Provider (IP) on the fly auf den Wert des Bus aufaddieren.

Jedes Triggermodul, bis auf das letzte Modul in der Kette, arbeitet prinzipiell nach demselben Prinzip für die Energie- und Multiplizitätsbusleitung. Nachdem der Beginn eines neuen Datenworts detektiert wurde, wird in jeder Taktperiode das vom Bus empfangene Bit und das Bit mit dem entsprechenden Stellenwert vom IP addiert. Ein eventueller Übertrag wird dabei in einem getakteten Flip-Flop zwischengespeichert und im nächsten Taktzyklus aufaddiert. Das erste in der Kette befindliche Modul verarbeitet nur das Datenwort, das es aufaddieren soll, da kein zu empfangendes Bit vorhanden ist.

Dabei wird die Logik, die für die Addition zuständig ist, rein kombinatorisch ausgeführt. Auf diese Weise lässt sich in jedem Takt ein Bit über den ganzen Bus senden, ohne durch die Synchronisation der Signale, wie sie bei einer synchronen Herangehensweise nötig wäre, Zeit zu verlieren. Abbildung 2.2(a) zeigt ein Zeitablaufdiagramm eines ideal funktionierenden Bus.



(a) Vernachlässigt man Unterschiede in den Signallaufzeiten, sind synchrones und asynchrones Design gleichwertig.



(b) Im realen Fall jedoch führen diese Unterschiede zu zunehmendem Versatz zwischen Steuer- und Datensignalen.

Abbildung 2.2: Zeitablaufdiagramme eines ideal bzw. real funktionierenden Bus. Die Zeitpunkte, zu denen das Signal beim realen Bus vom letzten Modul abgefragt wird sind durch Pfeile markiert.

Außerdem gibt das Modul das BIT0 Signal an den Information Provider weiter, damit dieser den maximalen Energie- und Multiplizitätswert, der während des letzten

Datenworts auftrat, zum Beginn des neuen Datenworts anlegen kann. Auf diese Weise wird sichergestellt, dass kein Peak zwischen zwei Datenworten verloren geht.

Da es kleine Unterschiede in der Software des ersten Moduls in der Kette und der folgenden Module gibt, ist eine Unterscheidung dieser so genannten Busrollen in TOP und MID nötig.

Das letzte Modul in der Kette, genannt BOT, ist für die Deserialisierung der Daten zuständig. Dafür setzt es mit dem Beginn eines neuen Datenworts das erste Bit seines Datenausgangs auf den vom Bus empfangenen Wert, im nächsten Takt wird das zweite Bit gesetzt und so weiter bis zum Beginn des neuen Datenworts. Neben den Daten gibt es auch das BIT0-Signal an seinen IP weiter, so dass dieser auf BIT0 die Schwellwertentscheidung treffen kann.

Da die Steuersignale CLK und BIT0 nur durch das CPLD geschleift werden, die Datensignale jedoch verarbeitet, ergeben sich unterschiedliche Laufzeiten für die zwei Signalarten durch das CPLD. Dies führt zu einer Situation, wie sie in Abbildung 2.2(b) dargestellt ist. Dabei verstärkt sich die Verschiebung der Signale mit zunehmender Position in der Kette, da sich die einzelnen Verzögerungen aufsummieren. Bei einem vollständig synchronen Design würde man diesen Effekt dadurch ausgleichen, dass man die Signale jeweils so verzögert, dass die Signal- und Taktflanke wieder zusammefallen. Auf diese Weise verlängert sich die Signallaufzeit pro Modul um mindestens einen Takt, was die Einhaltung der Zeitvorgabe erschwert.

Um dem Problem der Signalverschiebung, das durch die asynchrone Konzeption des Bus entsteht, in diesem Design zu begegnen, werden die Signale auf steigender Taktflanke erzeugt und verarbeitet, jedoch vom letzten Modul bei fallender Taktflanke (in der Abbildung durch Pfeile markiert) deserialisiert. Dadurch kann der maximale Versatz zwischen Takt und Daten eine halbe Taktperiode betragen, bevor die zeitliche Korrelation der Signale und damit die Funktionalität des Bus verloren geht. Es gilt also

$$f_{max} = \frac{1}{2\left(n_M |d_{DAT} - d_{CLK}| + t_{SU}\right)} \tag{2.1}$$

Dabei meint n_M die Kettenlänge und d_{DAT} bzw. d_{CLK} die mittlere Gesamtlaufzeit der entsprechenden Signale durch ein Modul. Dies ist gerechtfertigt, da die Signale durch weitere Bausteine auf der Platine verzögert werden (siehe 2.2). Die so gennte setup zeit des Übertrags-Flip-Flop t_{SU} wird im folgenden Absatz erklärt.

Neben diesem Problem entstehen aus der asynchronen Auslegung des Systems jedoch noch weitere Schwierigkeiten. Die unterschiedliche Laufzeit von Takt- und Datensignalen führt nämlich nicht nur zu einer zeitlichen Verschiebung der Signale. Vielmehr entstehen dadurch, dass jedes Modul die Taktflanke relativ zur Signalflanke ein wenig später erkennt als das vorhergehende, Bereiche um die Signalflanke in denen das Signal keinen gültigen Wert besitzt. Hier fluktuiert das Signal, da es von jedem Modul zu einer leicht unterschiedlichen Zeit geändert wird. Diese so genannten Glitches können in nachfolgenden Filp-Flops oder anderen Logikgliedern

zu Problemen führen, da alle getakteten Logikelemente nur dann korrekt arbeiten können, wenn die Eingangssignale bereits einige Zeit vor der Taktflanke stabil sind. Ist dies nicht der Fall, so schlägt die Signalerkennung bei der Taktflanke fehl und das Logikgatter geht einen undefinierten Zustand über, der für unbekannte Zeit anhält. Dies setzt das betroffene CPLD und damit die ganze Kette für die Zeit des undefinierten Zustands außer Funktion. Ob diese so genannte setup time von etwa 1 ns mit diesem Design zuverlässig genug eingehalten werden kann, bestimmt ganz wesentlich ob ein solches System stabil implementiert werden kann.

Aufgrund dieser Schwierigkeiten wurden solche asynchrone Designs bisher nicht realisiert.

Einen weiteren limitierenden Faktor stellt die Anforderung dar, das Triggersignal innerhalb von höchstens 1.0 μs zu Verfügung zu stellen. Dies begrenzt über die maximale Signallaufzeit die maximal mögliche Kettenlänge.

Die Signallaufzeit durch die gesamte Kette setzt sich zusammen aus der physikalischen Laufzeit durch die Module und der Zeit, um ein Datenwort zu übertragen. Für die physikalische Laufzeit ist das Maximum der mittleren Laufzeit von Takt- bzw. Datensignal $\max(d_{DAT}, d_{CLK})$ maßgeblich, da dieses Signal am längsten benötigt um von einem Ende der Kette zum anderen zu propagieren. Die Zeit, die ein Datenwort zur Übertragung benötigt, hängt im wesentlichen von der Anzahl der Bits pro Datenwort, also der Busbreite, ab.

Es gilt also

$$d_{ges} = n_M \max(d_{DAT}, d_{CLK}) + \frac{n_B}{f_B} \tag{2.2}$$

wobei n_M die Kettenlänge, n_B die Busbreite und f_B die Taktfrequenz bezeichnet.

Im Idealfall wird die Triggerschwelle gerade zum Beginn eines neuen Datenworts überschritten. Dann ist durch 2.2 die komplette Verzögerung gegeben. Im schlechtesten Fall jedoch tritt die Überschreitung kurz nach dem Anfang des Datenworts auf. Dann kann der entspechnede Wert für Energie oder Multiplizität erst mit dem nächsten Datenwort und damit um bis zu $\frac{n_B}{f_B}$ verzögert übertragen werden. Insgesammt liegt die Zeit zwischen dem Überschreiten der Triggerschwelle und dem ziehen des Triggers durch das letzte Modul d also zwischen

$$n_M \max(d_{DAT}, d_{CLK}) + \frac{n_B}{f_B} \leq d \leq n_M \max(d_{DAT}, d_{CLK}) + 2\frac{n_B}{f_B} \tag{2.3}$$

und die mittlere Verzögerung beträgt

$$\bar{d} = n_M \max(d_{DAT}, d_{CLK}) + \frac{3}{2}\frac{n_B}{f_B} \tag{2.4}$$

Bei einer Betriebsfrequenz von 40 MHz und 24 Modulen pro Kette ergibt sich damit eine Signallaufzeit von 473 ns bis 748 ns wenn 11 Bit pro Datenwort übertragen werden. Dabei wurde $d_{DAT} = 8.26$ ns und $d_{CLK} = 7.91$ ns angenommen. Bei vollständig

synchroner Übertragung würde das Signal, wie oben erläutert, um mindestens 24 Takte zusätzlich verzögert und die Signallafzeit läge unter gleichen Bedingungen zwischen 1073 ns und 1948 ns. Erst mit einer deutlich höheren Busfrequenz von etwa 150 MHz könnten ähnliche Werte wie im asynchronen Fall ereicht werden. Diese hohen Frequenzen sind jedoch schwer zu handhaben (siehe Abschnitt 4.4).

2.2 Hardware

Um das System zu testen, wurde ein Prototyp entwickelt. Das dabei entstandene Platinenlayout von Max Winkel ist in Abbildung 2.3 gezeigt.

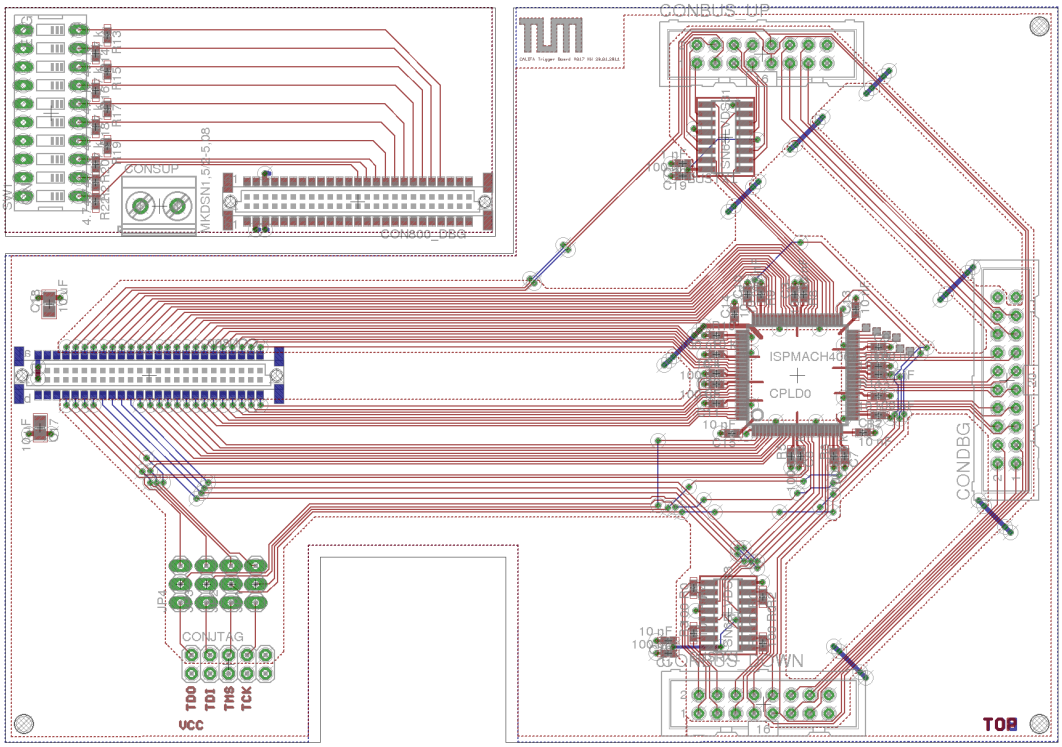


Abbildung 2.3: Abbildung des Platinenlayouts des Prototypen. Man erkennt das CPLD in der Mitte, umgeben von den differentiellen Treiber- und Empfangsbausteinen. Der Informationsaustausch zwischen FPGA und CPLD funktioniert über den fünfzigpoligen Verbinder an der linken Seite.

Das Herzstück des Entwurfes bildet ein so genanntes CPLD ispMACH 4000V von Lattice Semiconductor Corp., das in der Mitte platziert ist. Das Acronym steht für

Complex **P**rogrammable **L**ogic **D**evice. Es handelt sich dabei um einen komplexen logischen Baustein, der im Wesentlichen aus vier so genannten Makrozellen, d. h. Programmierbaren Arrays von Logikgattern, besteht. Diese sind über einen Bus mit einem Routingblock verbunden. Das Programm auf dem CPLD legt dabei fest, in welcher Abfolge die Signale vom einem Eingangspin des CPLD durch die Makrozellen und dann zum Ausgang läuft. Auf diese Weise lassen sich mit einem solchen Baustein komplexe Logikelemente mit variabler Pinkonfiguration realisieren. Eine für die Verwendung in diesem System sehr wichtige Eigenschaft der CPLD ist dabei, dass jedes Signal nach dem Durchlaufen einer Makrozelle immer wieder zum Routingblock zurück läuft, selbst wenn es danach wieder in dieselbe Makrozelle geleitet wird. Dadurch sind die Laufzeiten der einzelnen Signale durch das CPLD sehr genau vorhersagbar. Das CPLD soll in diesem Design die gesamte Signalverarbeitung übernehmen.

Da die Signale über lange Busleitungen geleitet werden sollen, ist es nicht zweckmäßig sie ohne Schutzmechanismen gegen Störungen von außen zu übertragen. Daher befinden sich auf jeder Platine jeweils ein Receiver und Sender für differentielle Signale nach dem LVDS Standard. Diese liegen in der Nähe der Wannenstecker für die Busleitungen. Es handelt sich dabei im ersten Entwurf um die Bausteine LVDS33 und LVDS31 von Texas Instruments. Weitere Informationen zu diesem Bausteinen finden sich in (LVD07) und (LVD04).

Die restlichen, auf der Platine befindlichen, Bauteile sind ein Steckverbinder für den Informationsaustausch mit dem Information Provider, der Steckverbinder für die Programmierung und ein Steckverbinder zu testzwecken.

Die Rolle des Information Providers übernehmen bei dem Prototypen so genannte ADC-Module. Diese wurden für das HADES Experiment in der GSI entwickelt und beinhalten mehrere analog zu digital Wandler sowie ein FPGA (**F**ield **P**rogrammable **G**ate **A**rray). Dieser Baustein funktioniert ähnlich wie ein CPLD ist jedoch wesentlich komplexer aufgebaut und erlaubt damit mehr Logikfunktionen auszuführen. Das FPGA ist mit einer Software ausgestattet, die es erlaubt Daten über ein Glasfasernetzwerk auszutauschen und das ADC-Board verfügt weiterhin über einen fünfzigpoligen Steckverbinder über den FPGA und CPLD miteinander kommunizieren können. Weitere Informationen zu den ADC-Modulen finden sich in (Bö99).

2.3 Software

Die Software auf dem CPLD besteht, je nach Position in der Kette, aus verschiedenen Elementen, die jeweils identisch für die Übertragung der Energie und der Multiplizitätssumme ausgeführt sind.

Alle bis auf das letzte Modul enthalten einen $n_B : 1$ Multiplexer an dem die zu Übertragenden Daten anliegen. Dieser ist mit einem getakteten Zähler verbunden, der

das zu übertragende Bit auswählt und auf BIT0 zurückgesetzt wird. Der Ausgang des Multiplexers und die entsprechende Busleitung führen dann in einen kombinatorisch ausgeführten Volladdierer, dessen Carry-Ausgang mit einem getakteten Flip-Flop verbunden ist. Der Ausgang dieses Flip-Flop führt ebenfalls in den Addierer, so dass der Übertrag aus dem vorherigen Taktzyklus im nächsten aufaddiert werden kann.

Im letzten Modul ist nur ein getaktetes Schieberegister implementiert, das auf BIT0 zurückgesetzt wird, um die ankommenden Daten zu deserialisieren. Diese Logik arbeitet, im Gegensatz zu der der anderen Module in der Kette, wie oben ausgeführt auf fallender Taktflanke.

Kapitel 3

Aufbau und Weiterentwicklung des Testsystems

3.1 Entwicklung von Software für CPLD und FPGA

Die Software für das CPLD und das FPGA wurde im Industriestandard VHDL (**V**ery High Speed Integrated Circuit **H**ardware **D**escription **L**anguage) geschrieben. Es handelt sich dabei um eine abstrakte Programmiersprache, die es erlaubt, die benötigten Logikfunktionen und deren Ablauf festzulegen. Der Compiler erzeugt dann ein Programm indem er versucht, die verfügbaren Logikgatter möglichst günstig miteinander zu verbinden.

Ein erster Entwurf der Software für beide Module wurde noch von Max Winkel angefertigt. Mit diesem war man in der Lage, den Bus zu betreiben, d. h. Takt- und BIT0-Signal zu erzeugen, und statische Daten zu senden. Das Auslesen der gesendeten Daten am Ende der Kette bzw. eine Triggerentscheidung war jedoch noch nicht möglich.

Schon in dieser Version wurde die Rolle des BIT0-Signals effektiv in die eines BIT($n_B - 1$)-Signals umdefiniert. Dies war nötig, da das Signal im FPGA aus dessen Taktsignal erzeugt wurde und damit mit einer kleinen Verzögerung gegenüber dem CLK-Signal vorlag. Dadurch konnten die in 2.1 angesprochenen setup times der CPLD unmöglich eingehalten werden und das Synchronisierungssignal wurde erst im Takt nach dem es erzeugt wurde von den CPLD erkannt.

In einem ersten Schritt wurde die bis dahin fest voreingestellte Taktfrequenz zur Laufzeit einstellbar gemacht und die fehlende Logik für die BOT Module implementiert. Es stellte sich jedoch schnell heraus, dass die verwendete Methode zur Takterzeugung optimiert werden musste und der existierende Programmcode schwer zu Warten war. Daher wurde nahezu die gesamte Software neu entwickelt um größtmögliche Funktionssicherheit zu gewährleisten.

Im Zuge dieser Neuentwicklung wurde der Progammcode für die einzelnen Busrollen, der bis dahin zusammengefasst war, in einzelene Teile aufgespalten und verschiedene Test- und Überwachungsmechanismen implementiert.

Insbesondere wurde ein so genannter Event Player (EP) in die FPGA-Software für

TOP und MID Module integriert. Dieser besteht im Wesentlichen aus einem RAM-Baustein mit 1024 Adressen, der über das Glasfasernetzwerk gefüllt werden kann. Wird das entsprechende Bit im Statusregister gesetzt, so wird bei jeder steigenden Signalflanke von BIT0 taktsynchron der nächste Wert aus dem RAM an den Energie-Eingang des CPLD angelegt. Ist das Ende des Speicher erreicht, so beginnt der Zyklus von neuem. Damit lässt sich eine realistische Anwendungssituation simulieren, da die Module (IP), die im Realbetrieb mit den Triggermodulen verbunden sein werden, ihre Daten ebenfalls taktsynchron anlegen werden (siehe Abschnitt 2.1).

Die BOT Module wurden mit dem Gegenstück zum Event Player, genannt Event Recorder (ER), ausgestattet. Dieser zeichnet je 1023 empfangene Samples vor und nach dem Ziehen des Summenenergie-Triggers durch das FPGA auf. Dies ist möglich durch die Verwendung zweier RAM Bausteine. Dabei wird der erste während des Wartens auf Überschreitung des Schwellenwerts kontinuierlich gefüllt und bei einem Überlauf seines Adressregisters wieder von Vorne begonnen. Wird nun der Trigger vom FPGA ausgelöst, so wird der zweite RAM Baustein bis zum Ende gefüllt und der Event Recorder geht in den ready Zustand über. Da die letzte Adresse des ersten RAM festgehalten wird, ist es dann möglich je 1023 Samples vor und nach dem Triggerzeitpunkt bereitzustellen.

Die einzelnen Funktionen der FPGA-Software wurden durch die Implementierung eines Statusregisters und mehrerer funktionsspezifischer Register einstellbar und überwachbar gestaltet. In Anhang B findet sich eine Beschreibung der entsprechenden Registerbelegung und sonstigen technischen Details der Implementierung.

Um die Kapazität des CPLD nicht zu überschreiten und größtmögliche Funktionssicherheit zu erreichen, wurden in der Software für das CPLD nur die unter 2.3 beschriebenen Funktionen implementiert.

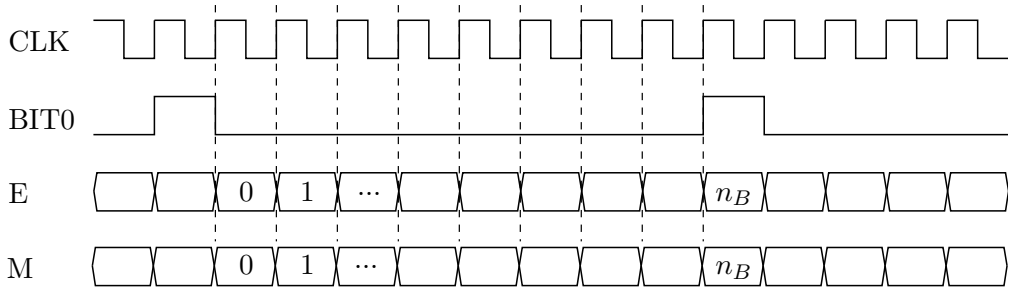
3.2 Mögliche Betriebsmodi

Da sich im Laufe der Tests abzeichnete, dass die mit den Prototypen erreichbare maximale Taktfrequenz begrenzt ist, wurde neben dem oben beschriebenen Betriebsmodus mit paralleler Übertragung von Energie und Multiplizität ein Hochgeschwindigkeitsmodus erprobt.

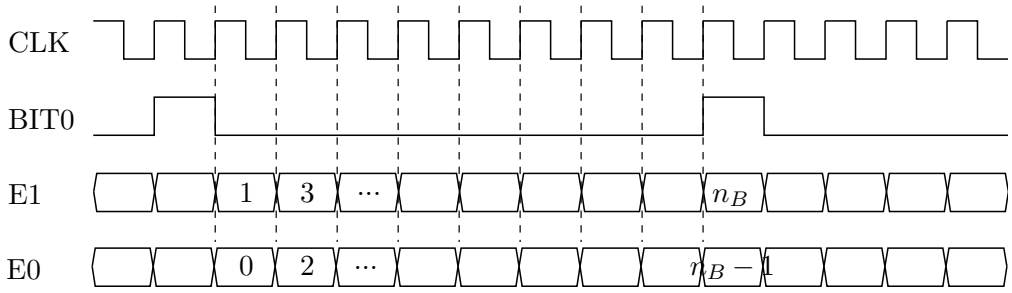
In diesem Modus wird nur ein Wert über den Bus übertragen, wobei die beiden Busleitungen verschränkt werden. Das heißt, das erste, dritte usw. Bit wird über die eine Busleitung und das zweite, vierte usw. Bit parallel über die andere Leitung übertragen. Auf diese Weise halbiert sich die Zeit, die benötigt wird, um ein Datenwort über den Bus zu senden.

Um diesen Modus testen zu können, wurde eine alternative Software für das CPLD entwickelt, die sich von der normalen darin unterscheidet, dass der Multiplexer in den TOP und MID Modulen nun je zwei Bit liefert und der Addierer simultan zwei Bit

vom Multiplexer und zwei Bit vom Bus addiert. Dabei ist weiterhin nur ein Flip-Flop zur Speicherung des Übertrags nötig. In den BOT Modulen wurde entsprechend ein $2 : n_B$ Demultiplexer implementiert, um diese Datenstruktur zu deserialisieren.



(a) Im Normalen Betrieb werden das Endergie- und Multiplizitätssignal parallel gesendet.



(b) Im Hochgeschwindigkeitsmodus wird nur die Energieinformation übertragen, wobei jeweils zwei Bit gleichzeitig gesendet werden.

Abbildung 3.1: Zeitablaufdiagramme für den Normalen Betriebsmodus und den Hochgeschwindigkeitsmodus. Die Zahlen geben den Stellenwert des jeweiligen Bit an.

Abbildung 3.1 illustriert das unterschiedliche Verhalten des CPLD in den zwei Betriebsmodi.

Kapitel 4

Test des Systems

4.1 Beschreibung des Testsystems

Alle Software- und Hardwaretests wurden an einem Test- und Kalibrierungsaufbau für das HADES Experiment durchgeführt. Abbildung 4.1 zeigt ein Foto dieses Aufbaus mit einer zum Test aufgebauten Modulkette.

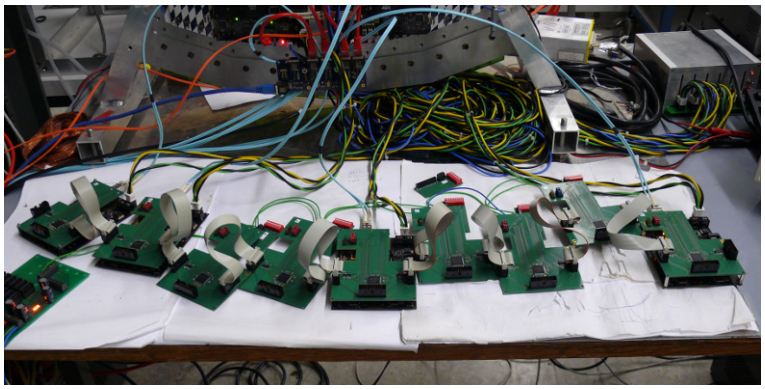


Abbildung 4.1: Foto des Aufbaus für Testläufe. Die Signale laufen von links nach rechts wobei an das zweite und fünfte Modul ein ADC-Board mit aktivem EP angeschlossen ist.

Ein wesentlicher Teil der Infrastruktur des Teststandes besteht aus dem Glasfasernetzwerk, an das alle FPAG-Module angebunden werden können. Über dieses Netzwerk, dass von einem sog. ETRAX-Modul verwaltet wird, ist es möglich, jedes Modul mit seiner eindeutigen Adresse anzusprechen und seine Register zu lesen bzw. schreiben. Da auf dem ETRAX-Modul eine Version des Embedded Linux busybox läuft ist es darüber hinaus möglich shell scripts und crosscompilierte C-Programme auszuführen und die Testabläufe so zu automatisieren. Eine Beschreibung der entwickelten Test- und Managementprogramme findet sich in Anhnag C.

Darüber hinaus steht ein volldigitales Oszilloskop Tektronix 7104 mit Windows

Betriebssystem zur Verfügung, das unter anderem dazu benutzt wurde um die CPLD auf den Triggermodulen mit der entsprechenden Software zu programmieren.

4.2 Maximale Kettenlänge

Wie oben erwähnt, ist die maximale Kettenlänge, mit der der Bus weniger als 1 μs Gesamtverzögerung aufweist, wesentlich durch die Laufzeit der Steuersignale und der Nutzdaten begrenzt. Diese wird durch die Verzögerung der Signale durch das CPLD und durch die LVDS Bustreiber und -receiver dominiert.

Im Falle des CPLD sind diese Verzögerungen berechenbar und werden vom Compiler in Form eines Timing Report ausgegeben. Die Werte für die aktuelle Software sind in Tabelle 4.1 angegeben.

	Verzögerung von			
Busrole	CLK	BIT0	DAT0	DAT1
TOP	3.30 ns	3.39 ns	—	—
MID	3.95 ns	3.36 ns	4.00 ns	4.00 ns
BOT	—	4.06 ns	1.83 ns	1.83 ns

Tabelle 4.1: Vom Compiler berechnete Werte für die Laufzeitverzögerung der Bussignale durch das CPLD.

Für die in den Prototypen zuerst verwendeten differentiellen Treiber und Receiver LVDS31 und LVDS33 von Texas Instruments sind die spezifizierten Werte in der oberen Hälfte von Tabelle 4.2 angegeben.

		Propagation Delay		Output skew
Hesteller	Baustein	Low to High	High to Low	
Texas Instruments	LVDS31	$0.5 - 2$ns	$2 - 2.5$ns	$0 - 0.3$ns
	LVDS33	$2.5 - 6$ns	$2.5 - 6$ns	≈ 0.150ns
Analog Devices	ADN4665	$0.8 - 2.0$ns		$0 - 0.5$ns
	ADN4666	$1.8 - 3.3$ns		$0 - 0.5$ns

Tabelle 4.2: Spezifizierte Kenngrößen der differentiellen Bustreiber LVDS31 und -receiver LVDS33 sowie der als Ersatz verwendeten Module. Output skew meint die zeitliche Verschiebung des Signals zwischen unterschiedlichen Kanälen desselben Bausteins. Die Daten stammen aus (LVD07), (LVD04) sowie (ADN09a) und (ADN09b).

Da diese Werte, insbesondere für den Receiver LVDS33, in einem recht großen

Intervall liegen und für den reibungslosen Betrieb des Buses sehr wichtig sind, wurden alle gefertigten Prototypen daraufhin charakterisiert.

Das heißt, es wurde die Zeit gemessen, die das CLK- bzw. DAT0-Signal benötigt um durch das Modul zu propagieren und um vom Ausgang des LVDS-Receiver zum Eingang des LVDS-Treiber zu gelangen. Auf diese Weise lässt sich berechnen, wie lange die Signale jeweils durch die Kombination aus differentiellem Receiver und Treiber verzögert werden. Alle Messungen wurden mit einem Oszilloskop Tektronix DPO4054 durchgeführt wobei die Verzögerung bei steigender Signalflanke vermessen wurde.

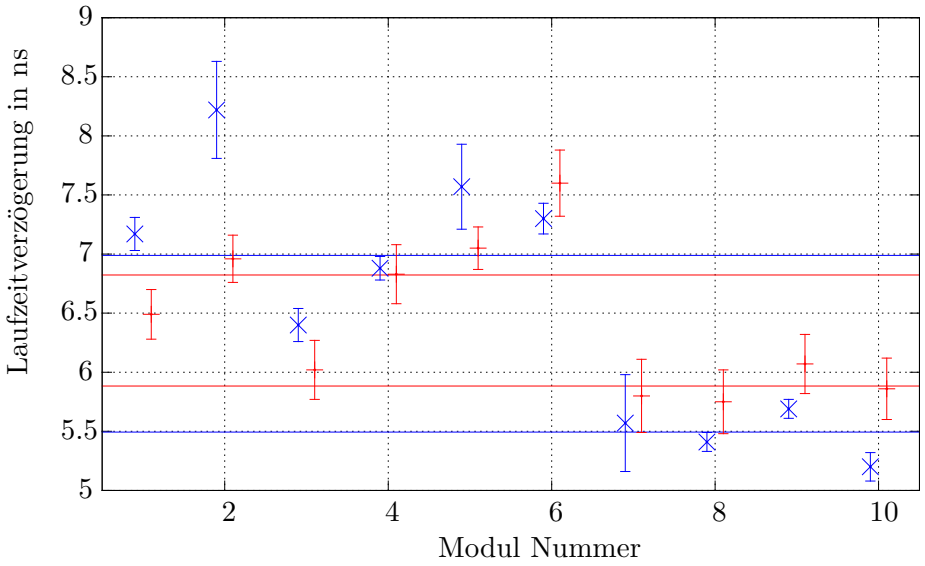


Abbildung 4.2: Laufzeitverzögerung von CLK (blau) und DAT0 (rot) für alle charakteriserten Triggermodule. Die vertikalen Linie markieren die jeweiligen Durchschnittswerte für die differentiellen Bausteine LVDS3X (Modul 1 - 6) bzw ADN466X (Modul 7 - 10).

An den in der linken Hälfte von Abbildung 4.2 gezeigten Daten erkennt man, dass durch LVDS31 und LVDS33 eine Laufzeitverzögerung von durchschnittlich 7 ns pro Modul entsteht. Da sich die Module am oberen Rand der spezifizierten Werte bewegen, kommt mit der Verzögerung durch das CPLD eine Signallaufzeit von etwa 10 ns zu Stande. An Gleichung (2.2) lässt sich ablesen, dass diese Signalverzögerung bei höheren Taktfrequenzen der wesentliche Faktor ist, der die maximale Länge der Modulkette limitiert.

Aus diesem Grund wurde nach differentiellen Bustreibern und -Receivern gesucht, die LVDS31 und LVDS33 ersetzen können und eine signifikant geringere Signalverzö-

gerung bewirken. Nach gründlicher Durchsicht des relevanten Angebots wurden die Bausteine ADN4666 und ADN4665 von Analog Devices Inc. als Ersatz ausgewählt und vier Module damit bestückt. Informationen zu diesen Basuteinen finden sich in (ADN09a) und (ADN09b). Die spezifizierten Verzögerungen des ADN4665 und ADN4666, die ebenfalls in Tabelle 4.2 zusammengestellt sind, ließen eine um $1-2$ ns geringere Singalverzögerung pro Modul erwarten.

Die Ergebnisse der Charakterisierung der neuen Module zeigt die rechte Hälfte von Abbildung 4.2. Es ist deutlich erkennbar, dass die Signallaufzeit wie erwartet um $1-2$ ns geringer ist als mit den differentiellen Bausteinen von Texas Instruments. Besonders deutlich wird dies beim Vergleich von Modul Nummer 1 und 9. Da Modul 1 die schlechteste Charakteristik aufwies, wurden die zuerst verwendeten Bausteine von diesem Modul entfernt und durch den Ersatz von Analog Devices ausgetauscht. Mit dem Ergebnis, dass sich die Laufzeit des Taktsignals um $(1.2 \pm 0.2)\ ns$ und die des DAT0-Signals um $(0.4 \pm 0.2)\ ns$ verringert hat.

Im Durchschnitt ergibt sich eine Verringerung der Signallaufzeiten von Takt und Daten von 0.94 ± 0.16 ns bzw. 1.50 ± 0.08 ns. Dies ermöglicht nach Gleichung (2.4) bei $n_B = 12$ ca. 5 Module mehr pro Kette zu verwenden.

Eine weitere Steigerung der maximalen Kettelänge ist mit den im Moment am Markt befindlichen LVDS-Komponenten aller Voraussicht nach nicht möglich. Jedoch ist die LVDS-Technologie gängiger Industriestandard und wird in der modernen Elektronik intensiv verwendet, so dass sich die Leistungsfähigkeit der Bausteine mit großer Wahrscheinlichkeit weiter verbessert hat bis eine Großserie der Triggermodule entworfen wird.

4.3 Maximale Betriebsfrequenz

Der zweite Faktor auf den die Signalverzögerung Einfluss hat, ist wie oben ausgeführt, die maximal mögliche Betriebsfrequenz. Diese hängt nach Gleichung (2.1) vom Unterschied der mittleren Verzögerung von Takt- und Datensignal ab.

An den in der linken Hälfte von Abbildung 4.2 zusammengefassten Ergebnissen erkennt man, dass sich alle vermessenen Module mit den Bausteinen LVDS31 und LVDS33 innerhalb der Spezifikation bewegen.

Jedoch zeigen z. B. die Module 1 und 6 ein für diese asynchrone Anwendung ungünstiges Zeitverhalten, da sie eine Verzögerung von etwa 0.4ns zwischen Takt und Daten erzeugen. Werden mehrere Module mit derart großem Versatz hintereinander platziert, so mindert dies die maximale Taktfrequenz erheblich, denn die sich aufsummierende Verzögerung führt bei einem zu schnellen Taktsignal dazu, dass Takt und Daten nach diesen Modulen zeitlich um mehr als eine halbe Taktperiode auseinander liegen und so deren Zuordnung verloren geht.

Die, ebenfalls eingezeichneten, gewichteten Mittelwerte der Takt- bzw. Datenver-

zögerung zeigen jedoch, dass mit dieser Konfiguration bei geschickter Platzierung der Module durchaus eine annehmbare Busfrequenz möglich ist. Da die Differenz der Mittelwerte 0.17 ns beträgt, kann wenn die Module so platziert werden, dass sich die unterschiedlichen Verzögerungen kompensieren nach (2.1) eine Maximalfrequenz von 76.1 MHz bei 24 Modulen pro Kette erreicht werden.

Für die neuen Module ist der mittlere Unterschied zwischen den Laufzeiten von Takt und Daten mit 0.39 ns deutlich größer und die Maximalfrequenz mit 52.5 MHz bei 24 Modulen pro Kette damit wesentlich niedriger. Außerdem kann man erkennen, dass alle vier vermessenen Module das Datensignal stärker verzögern als den Takt. Dies bedeutet auf der einen Seite, dass die Module ihre unterschiedlichen Verzögerungen nicht gegenseitig kompensieren. Andererseits kann dieser Effekt dadurch kompensiert werden, dass eine Delayleitung auf der Platine vorgesehen wird.

4.4 Vergleich verschiedener Takterzeugungsmechanismen

Während der ersten Versuche, die Taktfrequenz zu erhöhen entstand die Vermutung, dass die Konzipierung der Hardware und insbesondere das Design der Platine ein weiterer limitierender Faktor für die maximal mögliche Taktfrequenz sein könnte. Dieser Verdacht begründete sich im Wesentlichen auf der schlechten Signalform, die die CLK-Busleitung zeigte.

Abbildung 4.7 auf Seite 27 zeigt die Ergebnisse der weiteren Untersuchung dieses Umstandes. In Abbildung 4.7(a) ist dabei die anfängliche Signalform, am Ausgang des FPGA zu sehen.

Die erste Version der FPGA-Software, die die Taktfrequenz erzeugt, verwendete einen einfachen kombinatorischen Ansatz. Das CLK-Signal wurde dabei aus dem internen Takt des FPGA erzeugt. Dieser zählte eine gewisse Anzahl von Taktzyklen ab, bevor er die Taktleitung invertierte und den Zähler zurücksetzte. Dieser Ansatz ist vor allem deshalb ungünstig, da das Taktsignal dann vom Compiler als Datensignal betrachtet wird und nicht über den internen Taktbus des FPGA geleitet wird. Dies hat zur Folge, dass das so generierte Taktsignal wesentlich stärker über- und unterschwingt, als eines das über den Taktbus geleitet würde (vergleiche Abbildung 4.7(a) und 4.7(b)). Insbesondere bei höheren Frequenzen führt dies zu Problemen, da wie in der Abbildung zu erkennen, schon bei 50 MHz Taktfrequenz das Signal praktisch nur mehr aus Über- und Unterschwingern mit einer dem Signalabstand vergleichbaren Amplitude besteht und diese fälschlicherweise als Taktflanken erkannt werden könnten. Die Folge wäre ein nichtdeterministisches Verhalten der getakteten Logik.

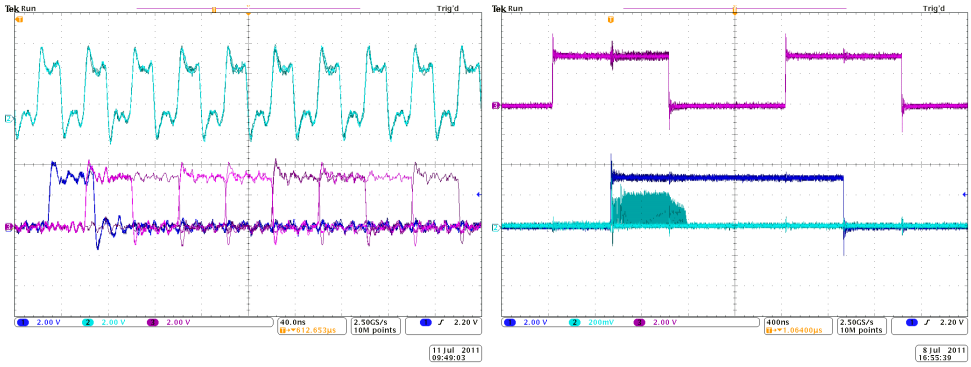
Um dieses Problem zu beseitigen, wurde die Software so umgeschrieben, dass das Taktsignal nicht mehr kombinatorisch sondern durch eine so genannte Phase Locked Loop (PLL) Schaltung erzeugt wurde. Dabei handelt es sich um einen komplexen

Schaltungstyp der durch einen einstellbaren Oszillator ein Ausgangssignal generiert, das in einer festen Phasenbeziehung zum Eingangssignal steht. Die PLL Schaltung existiert als von Hersteller vorprogrammierter Softwarebaustein mit zur Compilierzeit einstellbarer Frequenz. Den Protokolldateien des Compilers war zu entnehmen, dass dieses Signal als Taktsignal erkannt und über den Taktbus geleitet wurde. Abbildung 4.7(b) zeigt das Taktsignal, das mit einer PLL Schaltung erzeugt wurde, am Ausgang des FPGA. Wie man sieht, ist dieses Signal im Vergleich zum kombinatorisch erzeugten qualitativ deutlich besser, da keine Über- oder Unterschwinger auftreten.

Diese Änderung machte jedoch keine merkliche Erhöhung der maximalen Taktfrequenz möglich da das Taktsignal am Ende der Kette qualitativ nicht wesentlich besser war als zuvor. Da das Signal jedes CPLD auf seinem Weg passiert und von diesem neu getrieben wird, muss die Ursache für die ungenügende Signalqualität am Anfang der Kette liegen. Um dies zu überprüfen, wurde an der Leiterplatte eines der Prototypen ein Filament angelötet, mit dem das Taktsignal zwischen dem FPAG und dem CPLD abgegriffen werden konnte. Wie in Abbildung 4.7(c) zu sehen ist, ist bereits an dieser Stelle kaum ein Unterschied zwischen dem kombinatorisch und durch die PLL Schaltung erzeugten Signal zu erkennen.

Dies legt die Vermutung nahe, dass die Leitungsdämpfung auf dieser Übertragungsstrecke zu hoch ist, als dass ein hochfrequentes Taktsignal darüber übertragen werden kann. Leider hätte eine genauere Überprüfung dieser Vermutung Änderungen am Platinenlayout nötig gemacht, die im Rahmen dieser Arbeit nicht durchführbar waren. Des Weiteren ist fraglich, ob eine Änderung des Layouts des Triggermoduls dieses Problem gelöst hätte, da sich etwa die Hälfte des Übertragungswegs auf dem FPGA-Board befindet, dessen Layout nicht änderbar ist. Auch eine differentielle Übertragung des Taktsignals vom FPGA zum CPLD würde die Signalqualität deutlich verbessern. Auch dies wäre jedoch nur mit einem Redesign der Platine überprüfbar gewesen.

Stattdessen wurde das System für die weiteren Tests auf den Betrieb mit einem extern generierten Taktsignal umgestellt. Dafür wurde die TOP Version der CPLD Software so umgeschrieben, dass auch diese ihr Taktsignal vom Bus-Steckverbinder bezieht. Dabei muss das Synchronisierungssignal (BIT0) nun von der CPLD-Software generiert werden, da das FPGA dann unabhängig vom Bustakt arbeitet. In diesem Zug wurde die Logik zur Erzeugung des Synchronisierungssignals gegenüber der ursprünglichen Version verbessert. Die Änderung besteht darin, dass das CPLD ein Taktsignal mit der doppelten Busfrequenz in der Frequenz halbiert und das BIT0-Signal so generiert, dass es um einen viertel Takt gegenüber dem CLK-Signal nach hinten verschoben ist. Auf diese Weise ist sichergestellt, dass das Synchronisierungssignal bestmöglich gegen Verschiebungen relativ zum Taktsignal geschützt ist. Abbildung 4.3 zeigt das Ergebnis dieser Änderung.



(a) Deutlich erkennt man die Verschiebung des Synchronisierungssignals gegen den Takt. Das Datenwort beginnt dadurch bei drei Vierteln des Synchronisierungsbits.

(b) Die Detailaufnahme bei bei einer niedrigeren Frequenz zeigt die Verschiebung deutlich. Hier erkennt man auch das Übersprechen der Signale auf dem Weg zwischen den diff. Bauteilen und dem CPLD.

Abbildung 4.3: Veranschaulichung des verbesserten Taktgenerierungsmechanismus.

4.5 Test unter Realbedingungen

Um die Funktion des Systems unter realistischen Bedingungen zu testen, wurden längere Ketten von bis zu sieben Modulen Länge aufgebaut und die Event Player Funktion der FPGA Software genutzt um Daten über den Bus zu senden. Um dem Anwendungsfall nahe zu kommen, wurden die Event Player für alle Tests mit gaußförmigen Traces vorgeladen (siehe Abbildung 4.5). Dadurch werden Signale über den Bus gesendet, die denen, die vom FPGA generiert werden, recht nahe kommen.

Sehr früh zeigte sich, dass die probeweise weggelassene Synchronisation der Signale vom Bus mit dem Takt des FPGA zu erheblichen Problemen beim Betrieb führt. Wird das BIT0-Signal, das dem Event Player den Beginn eines neuen Datenworts anzeigt nicht auf den Takt des FPGA synchronisiert, so kann es passieren, dass dieses genau zum Zeitpunkt einer Taktflanke des FPGA-Takts, also dann wenn es stabil sein muss um zuverlässig erkannt zu werden, von einem Pegel auf den anderen wechselt. Wie in Abschnitt 2.1 beschrieben, führt dies im Allgemeinen zu einem undefinierten Zustand im FPGA, der für unbekannte Zeit anhalten kann. Bei Testreihen zeigte sich dies darin, dass das die Event Player einige Zeit fehlerfrei arbeiteten, jedoch plötzlich aussetzten. Meist funktionierten die Event Player nach kurzer Zeit wieder korrekt, da sich der undefinierte Zustand im FPGA normalerweise ohne äußeres Zutun wieder löst.

Der Grund für diese Versuche war, dass die Synchronisation üblicherweise dadurch

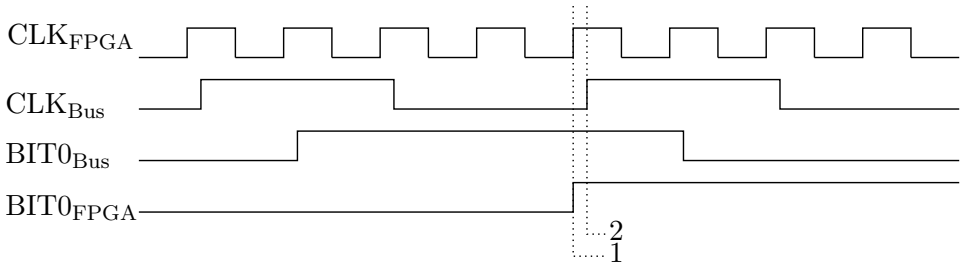


Abbildung 4.4: Zeitablaufdiagramm zu Illustration des Synchronisierungsvorgangs. Dadurch, dass das BIT0-Signal vom Bus im FPGA durch zwei getaktete Flip-Flops geleitet wird, ist es nach schlechtenstenfalls drei FPGA-Takten (bei Punkt 1) synchron zum Takt des FPGA. Erst dann kann das nächste Datenwort aus dem Speicher des Event Player geholt und an den Extension Header angelegt werden. Bei Punkt 2 wird dann das erste Bit dieses Datenworts vom CPLD erkannt und verarbeitet. Ist die Bedingung $f_{BUS} \leq \frac{1}{4} f_{FPGA}$ nicht erfüllt, so liegt das neue Datenwort erst am Extension Header an, wenn es schon verarbeitet werden müsste.

erreicht wird, dass man das zu synchonisierende Signal im FPGA durch zwei getaktete Flip-Flops leitet bevor es verarbeitet wird. Auf diese Weise ist das synchrone Signal um zwei bis drei Taktzyklen verzögert (siehe Abbildung 4.4). Implementiert man diesen Mechanismus nun für das BIT0-Signal, so beschränkt dies die maximal mögliche Busfrequenz auf ein viertel der Taktfrequenz des FPGA. Dies kommt daher, dass das Energie- bzw. Multiplizitätssignal vom FPGA umgeschaltet werden muss, bevor das neue Datenwort beginnt. Dieser Punkt liegt drei viertel der Bustaktperiode hinter der steigenden Flanke von BIT0, da das BIT0-Signal wie oben ausgeführt um 90° gegenüber dem CLK-Signal verschoben ist (Punkt 2 in Abbildung 4.4). Es folgt also

$$\frac{3}{4} T_{Bus} > 3 T_{FPGA} \Leftrightarrow f_{BUS} < \frac{1}{4} f_{FPGA}$$

Dabei ist zu beachten, dass die Busfrequenz echt kleiner als ein Viertel des FPGA-Takts sein muss, weil die benötigten setup times (siehe Abschnitt 2.1) in der obigen Überlegung vernachlässigt wurden.

Diese Überlegung ist auch im Produktivbetrieb relevant, da, wie in Abschnitt 2.1 beschrieben, auch hier das Energie- bzw. Multiplizitätssignal synchron mit BIT0 aktualisiert werden soll um nicht Gefahr zu laufen, einen Peak zwischen zwei Datenwörtern zu verlieren.

Nach der Implementierung der Synchronisation des BIT0 auf den Takt des FPGA und Erhöhung dessen Taktfrequenz auf 200 MHz konnte ein System aufgebaut werden, dass bei $f_B = 20$ MHz während einem, einige Tage dauernden, Testlauf fehlerfrei

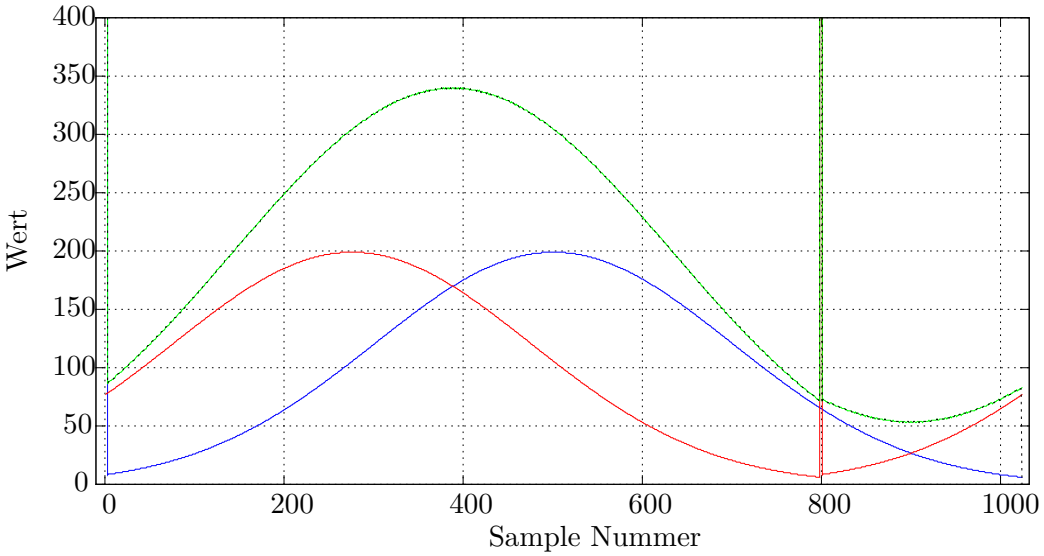


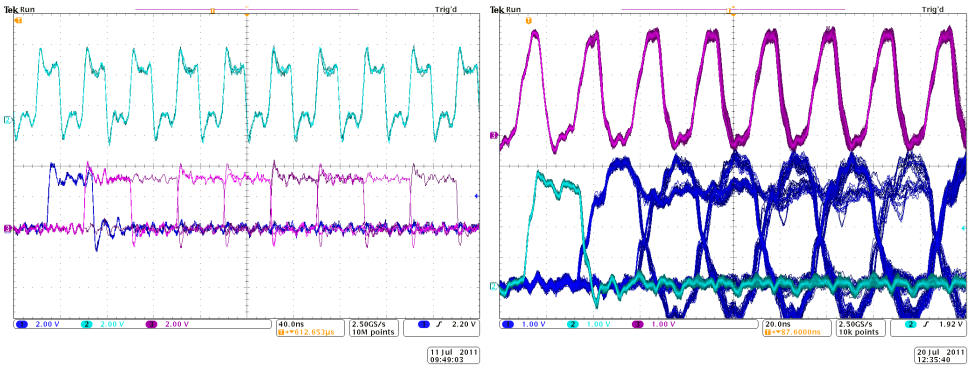
Abbildung 4.5: Beispielhafte Trace (schwarz gepunktet), wie sie vom Event Recorder nach sieben Modulen bei 20MHz Betriebsfrequenz aufgenommen wurde. Dabei waren an zwei Modulen Event Player mit demselben gaußförmigen Signalverlauf angeschlossen (rot und blau). Die spitzen Peaks sind keine Übertragungsfehler, sondern sind im Event Player zum kontrollierten Auslösen des Triggers vorgesehen. Die Erwartung ist in grün eingezeichnet und deckt sich genau mit der empfangenen Trace.

funktionierte. Dies wurde durch das kontinuierliche Auslesen und wieder Scharfstellen des Triggers überprüft. Eine typische Trace, die dabei entstanden ist, zeigt Abbildung 4.5. Diese wurde am Ende einer Kette von sieben Modulen aufgenommen, wobei zwei der Module mit einem FPGA-Modul mit Event Player verbunden waren. Man erkennt deutlich, dass die zwei identischen Traces, deren Anfang durch die spitzen Peaks markiert ist, korrekt aufaddiert und übertragen werden. Auch nach mehreren Tagen konnte keine Abweichung zwischen den empfangenen Traces und der Erwartung festgestellt werden.

Weiterführende Tests nach einer Erhöhung der Betriebsfrequenz auf 40 MHz zeigten, dass hier statistische Fehler auftreten, die fast ausschließlich das erste Bit nach dem Synchronisierungssignal betreffen. Eine Analyse von 30 Traces bei einer Kettenlänge von sieben Modulen und zwei angeschlossenen EP ist in Tabelle 4.3 zu finden.

Abweichung	-2	-1	0	1	2
Häufigkeit	—	$0.43 \pm 0.08\%$	$86.7 \pm 0.3\%$	$12.5 \pm 0.3\%$	$0.35 \pm 0.03\%$

Tabelle 4.3: Übersicht über die Häufigkeit von Abweichungen zwischen den empfangenen und erwarteten Samples bei 40 MHz Betriebsfrequenz, sieben Modulen in der Kette und zwei aktiven EP. Es wurden 30 Traces á 1023 Samples analysiert.

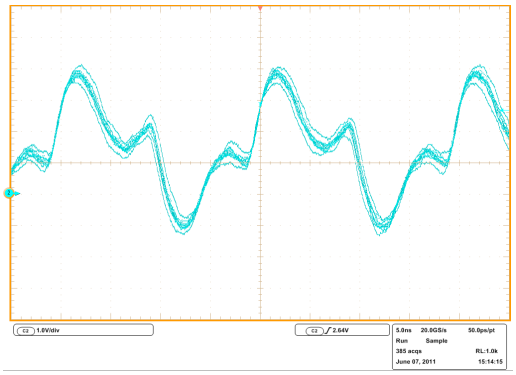


(a) Bei 25 MHz Busfrequenz ist die Signalform akzeptabel.

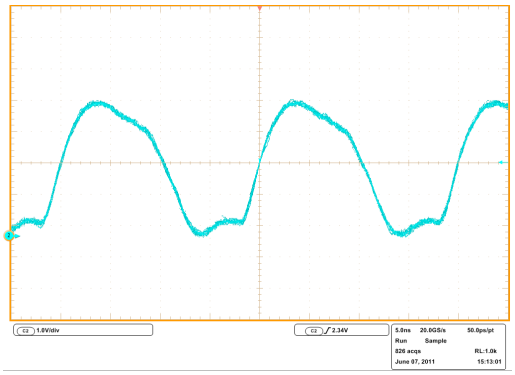
(b) Erhöht man die Busfrequenz auf 40 MHz , so verschlechtert sich die Signalform deutlich.

Abbildung 4.6: Veranschaulichung der Verschlechterung der Signalform bei Erhöhung auf 40 MHz Busfrequenz. Beide Aufnahmen sind nach einer Kette von sieben Modulen und zwei aktiven EP entstanden. Dabei ist jeweils oben das CLK-Signal und unten das BIT0-Signal und das Energie-Signal zu sehen.

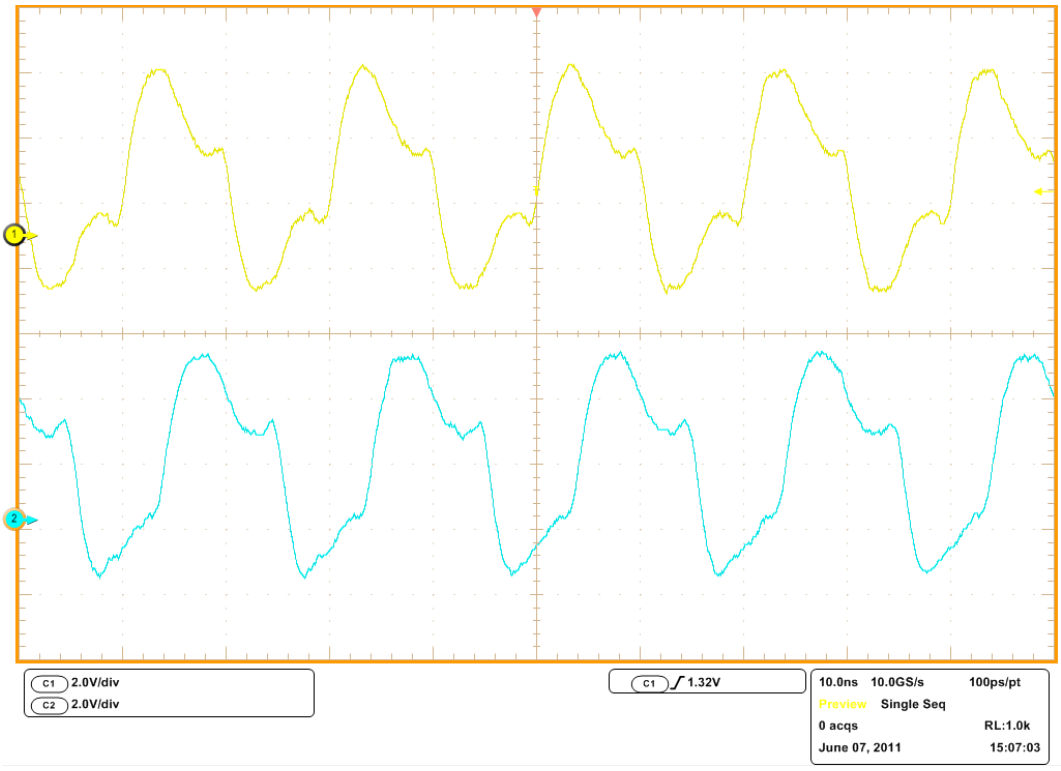
Die Oszilloskopaufnahmen in Abbilldung 4.6 legen nahe, dass bei einer Busfrequenz von 40 MHz die Signalqualität nicht mehr ausreichend hoch ist um den Bus fehlerfrei zu betreiben. In 4.6 (b) erkennt man deutlich, dass die Zeit, die die Signale benötigen um das logische Niveau zu wechseln mit der Taktzeit vergleichbar ist. Unter diesen Umständen können die Logikbausteine im CPLD nicht zuverlässig arbeiten, da die setup times nicht eingehalten werden können.



(a) Wird das Taktsignal durch kombinatorische Logik erzeugt, ist es kaum für die weitere Verwendung brauchbar.



(b) Bei der Erzeugung durch eine PLL-Schaltung ergibt sich ein deutlich besseres Signal.



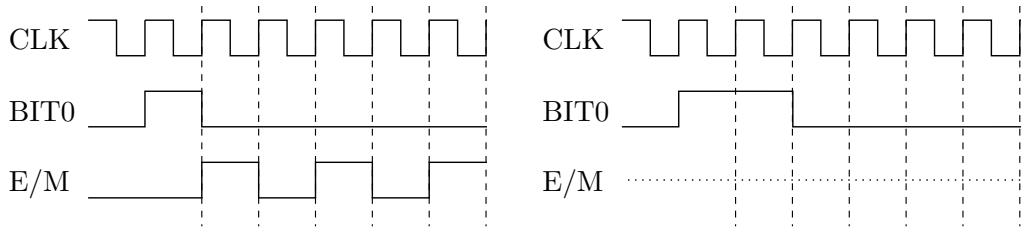
(c) Nach dem Durchgang durch das CPLD sind beide Signale kaum mehr zu unterscheiden und qualitativ zu schlecht, um den Bus damit zu betreiben.

Abbildung 4.7: Aufnahmen eines 50 MHz Taktsignals direkt am Verbinder von FPAG- und CPLD-Board (oben) bzw. nach dem CPLD (unten).

Kapitel 5

Fehlerbehandlung

Übertragungsfehler gleich welcher Art wirken sich bei zeitkritischen Anwendungen wie diesem asynchronen Triggerbus besonders schwerwiegend aus, da hier keine Zeit bleibt um sie zu lokalisieren und gegebenenfalls zu beheben. Stattdessen muss das Hauptaugenmerk auf der sicheren Erkennung und Markierung der fehlerhaft übertragenen Datenwörtern liegen damit es nicht zu unnötigen Triggerauslösungen kommt. Da dieses System mit einer Kettenstruktur arbeitet ist es zudem wünschenswert, dass permanente Fehlerquellen wie zum Beispiel defekte Triggermodule zuverlässig und mit möglichst geringem Aufwand gefunden werden können. Dazu sind alle FPGA-Module eindeutig adressiert.



(a) Im normalen Betrieb ist das BIT0 Signal einen Taktzyklus lang und dient nur dazu das Ende eines Datenworts zu signalisieren.

(b) Tritt ein Übertragungsfehler auf, so wird das BIT0 Signal länger auf logisch eins gehalten um einen Fehler anzuzeigen.

Abbildung 5.1: Zeitverlaufsdiagramme zur Veranschaulichung der Fehlersignalisierung.

Um einen Fehler im Bus zu signalisieren, eignet sich die Busleitung mit dem Synchronisierungssignal, da diese im normalen Betrieb wenig Information transportiert. Darüber hinaus zeichnet sich dieses Signal durch eine geringe Variationsbreite aus, so dass es leicht auf Plausibilität geprüft werden kann. Da die komplette Logik, die für die eigentliche Funktionalität des Buses sorgt, nur auf die steigende Flanke des BIT0-Signals reagiert, kann ein Fehler dadurch signalisiert werden, dass das Trig-

germodul das den Fehler feststellt das nächste BIT0-Signal um einen oder mehrere Taktzyklen verlängert. Siehe dazu Abbildung 5.1.

Dadurch ergeben sich bei einer Datenwortlänge von n Bit $n-2$ verschiedene Zustände, die möglichen Fehlern zugeordnet werden können. Da das Signal auf diese Weise keine zusätzlichen Flanken erhält, bedarf es bei diesem Verfahren keinerlei Änderungen an der produktiven Logik. Darüber hinaus bleibt die Variationsbreite des Signals gering, so dass ein ein Fehler auf dieser Leitung durch das letzte Triggermodul leicht festgestellt werden kann.

Grundsätzlich können zwei Arten von Fehlern unterschieden werden, die die Fuktionalität des Triggerbus gefährden. Dies sind zum einen fehlerhafte Signalverarbeitung und zum anderen Fehler bei der Übertragung des Signals.

5.1 Fehler bei der Datenverarbeitung

Fehler bei der Datenverarbeitung entstehen zum Beispiel durch undefinierte Signalpegel oder durch statistische Fehler bei der Übertragung der Daten an das CPLD welche zum Beispiel durch Störsignale entstehen können.

Allen diesen Defekten ist gemein, dass sie nur durch eine Plausibilitätsprüfung der übertragenen Daten gefunden werden können, da die Übertragung an sich funktioniert. Dabei muss ein Kompromiss zwischen Übertragungssicherheit und -geschwindigkeit des Bus gefunden werden, da der Aufwand mit der Leistungsfähigkeit eines solchen Verfahrens deutlich ansteigt. Darüber hinaus ist die Zahl der pro Taktzyklus vom CPLD ausführbaren Logikfunktionen eng begrenzt, was die Komplexität der realisierbaren Lösungen zudem einschränkt.

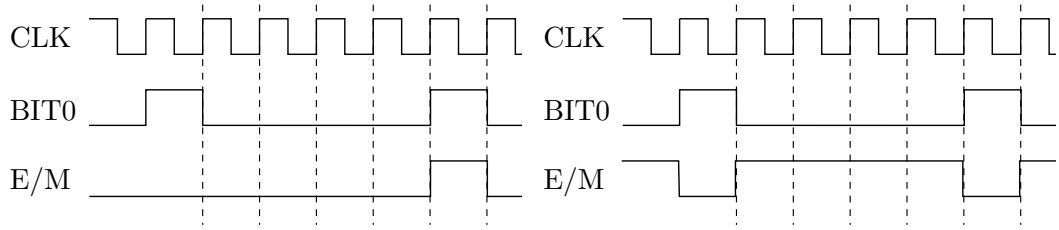
Mit recht geringem Logik- sowie Zeitaufwand kann eine Paritätsprüfung implementiert werden. Dabei wird an jedes Datenwort ein zusätzliches Bit angehängt, dass logisch eins oder null ist, abhängig davon ob die Summe der übertragenen Einsen im Datenwort gerade oder ungerade ist.

Da der Logikaufwand im CPLD so gering wie möglich sein sollte, um die Signallaufzeiten nicht unnötig zu verlängern und die Anzahl an möglichen Logikfunktionen pro Takt nicht zu überschreiten, ist es sinnvoll die Überprüfung des Paritätsbits im FPGA vorzunehmen. Dazu sind alle Busleitungen nach dem differentiellen Receiver auch zum FPGA geführt. Das Bussignal muss dabei im FPGA nicht parallelisiert werden. Es ist lediglich ein getaktetes Flip-Flop nötig, dass immer dann umschaltet, wenn eine logische eins empfangen wird. Am Ende jedes Datenwortes kann dann durch den Vergleich des ermittelten und des empfangenen Paritätsbits ein Fehler festgestellt werden. Dieser wird dann durch ein Fehlerregister angezeigt.

Wird ein Fehler detektiert, könnte dieser, wie oben beschrieben, durch ein überlanges Synchronisationssignal an die nachfolgenden Module gemeldet werden. Dabei ist zu beachten, dass diese Meldung für das vorhergehende Datenwort gilt, da der Fehler

erst am Ende eines Datenworts erkannt werden kann. Die nachfolgenden Module können also nicht auf den Defekt reagieren sondern das Signal nur bis zum letzten Modul weiterleiten, das das entsprechende Datenwort dann ignoriert. Dadurch ergäbe sich eine weitere Verzögerung des Signals durch die Paritätsprüfung, die die Triggerentscheidung um einen Übertragungszyklus verspätet. Aus diesem Grund wurde eine zusätzliche Busleitung vorgesehen um Paritätsfehler zu signalisieren. Diese muss nicht differentiell ausgeführt werden, da sie keine hochfrequenten Signale überträgt. Sie wird nur bei einem Paritätsfehler vom entsprechenden Modul von logisch null auf logisch eins angehoben. Damit kann dass letzte Modul das fehlerhafte Datenwort ohne Verzögerung erkennen und ignorieren.

Um Fehler bei der Datenverarbeitung zu erkennen, die systematischer Natur sind, bietet es sich an, das Paritätsbit invertiert zu übertragen also so, dass es bei gerader Paritätssumme logisch eins ist. Dadurch kann der häufigste dieser Fehler erkannt werden, nämlich dass ein Triggermodul die Busleitungen dauerhaft auf logisch null hält. Da eine korrekte Null dann mit einem gesetzten Paritästsbit übertragen wird, lässt sich der Übertragungsfehler durch das Ausbleiben des Paritätsbits leicht erkennen. Wird die Busbreite zudem gerade gewählt, so kann auch erkannt werden wenn die Busleitung fälschlich auf logisch eins gehalten wird, da das Paritatsbit im Fall eines Datenworts in dem alle Bits gesetzt sind logisch null sein muss. Siehe dazu Abbildung 5.2.



(a) Wird das Paritätsbit invertiert übertragen, so ist eine korrekt übertragene Null durch ein gesetztes Paritätsbit gekennzeichnet.

(b) Wird die maximal mögliche Zahl übertragen, so ist bei gerader Busbreite das invertierte Paritätsbit logisch null.

Abbildung 5.2: Zeitverlaufsdiagramme zur Veranschaulichung der Erkennung von systematischen Fehlern bei der Datenverarbeitung und im Übertagungsmedium.

5.2 Fehler im Übertagungsmedium

Da im Produktivbetrieb lange Ketten gebildet werden sollen, ist die Wahrscheinlichkeit für Fehler im Übertragungsmedium gegeben. Diese können zum Beispiel

durch defekte Kabel (Kabelbruch), schlechte Steckverbindungen oder kalte Lötstellen entstehen.

Solange die Steuerleitugen, d. h. CLK und BIT0 intakt sind, können solche Fehler leicht von der Software auf dem FPGA erkannt und signalisiert werden.

Ein Fehler im Übertagungsmedium führt in den meißten Fällen zu einem Signal wie es bei dauerhaft hoch oder heruntergezogener Datenleitungen entsteht. Da das CPLD und das FPGA an ihren Eingängen nur zwischen logisch eins und null unterscheiden, versuchen sie den undefinierten Signalpegel einem dieser Werte zuzuordnen. Dabei kann nicht vorhergesagt werden, welcher Wert erkannt wird, jedoch fluktuiert dieser Wert meist nicht. Im Ergebnis wird dann dauerhaft logisch null oder eins erkannt was der oben beschriebenen Situation entspricht.

In beiden Fällen ist es sinvoll, die oben beschriebene Methode des verlängerten BIT0 anzuwenden, da der Fehler permanenter Natur ist. Der Fehlercode kann nach einer festen Anzahl an Fehlern bei einem Modul ausgelöst werden um anzuzeigen, dass ein Modul einen dauerhaften Fehler vermutet. Daraufhin können die Fehlerregister aller Module ausgelesen werden, um das schadhafte Modul oder Kabel zu finden.

Kapitel 6

Fazit und Ausblick

Wie sich gezeigt hat, ist es durchaus möglich, mit der heute verfügbaren Hardware ein asynchrones Bussystem aufzubauen. Sofern man eine begrenzte Anzahl an asynchronen Elementen verwendet, kann die Stabilität des Systems noch gewährleistet werden.

Mit den Prototypen ist es, wie oben beschrieben, möglich Signale mit 20 MHz Busfrequenz durch bis zu sieben Module fehlerfrei zu übertragen und aufzuaddieren.

Jedoch zwingen die dabei auftretenden Probleme zu Abstrichen, insbesondere bei der Übertragungsgeschwindigkeit. Wegen der in Abschnitt 4.5 begründeten Notwendigkeit, die Bussignale in den FPGA zu synchronisieren und den daraus folgenden Einschränkungen für die Busfrequenz ist es nicht sinnvoll eine weitere Erhöhung über eine Taktfrequenz von etwa 50 MHz zu forcieren. Wie in Abschnitt 4.5 ausgeführt, erfordert dies, dass die Komponenten des FPGA, die für die Übergabe der Datenwerte an das CPLD verantwortlich sind, mit mehr als 200 MHz betrieben werden. Zwar sind Frequenzen bis etwa 400 MHz mit modernen CPLD und FPGA theoretisch möglich. Praktisch jedoch ergeben sich aus derart hohen Taktraten unnötige Einschränkungen bezüglich der möglichen Komplexität der Logik in den Bausteinen. Darüber hinaus steigt die Wahrscheinlichkeit, dass der Compiler keine Konfiguration bestimmen kann, die die daraus entstehenden zeitlichen Vorgaben erfüllen kann.

Stattdessen sollte das Hauptaugenmerk auf der weiteren Verringerung der physikalischen Laufzeit liegen um die Übertragungszeit zu verkürzen. Dies kann durch die Auswahl neuerer differentieller Bausteine geschehen. Eine Durchsicht der Datenblätter vergangener Bausteingenerationen macht zuversichtlich, dass diese zum Zeitpunkt der Entwicklung des Serienmodells zur Verfügung stehen.

Nach den in Abschnitt 4.4 dokumentierten Ergebnissen bezüglich der Qualität der übertragenen Taktfrequenz ist es sich sicherlich ratsam eine differentielle Übertragungsstrecke vom ersten FPGA zum entsprechenden CPLD vorzusehen. Dies würde es erlauben das System auch mit kürzeren Taktzeiten aus sich heraus zu betreiben.

Eine Möglichkeit die beiden oben genannten Punkte zu verwirklichen, ist ein CPLD zu verwenden, das mit differentiellen Signalen umgehen kann.

Neben der vollständig differentiellen Übertragung des Taktsignals sollte bei einem

Redesign der Platine auch darauf geachtet werden, dass die Signalleitungen, die zwischen den eventuellen differentiellen Bausteinen und dem CPLD verlaufen, weiter auseinander platziert werden. Wie sich herausstellte, kommt es durch die Nähe der Leitungen zu Übersprechen von einer Leitung auf die andere. Insbesondere die Taktleitung prägt durch die regelmäßigen steilen Flanken ein auf den meisten Oszilloskopaufnahmen gut erkennbares Störsignal auf die anderen Leitungen auf. Dies beeinträchtigt die Funktion der getesteten Ketten von bis zu sieben Modulen noch nicht, jedoch kann dies bei längeren Ketten und insbesondere bei längeren Kabeln zwischen den Modulen zu vermeidbaren Fehlerquellen führen. Da schon die Platine des Prototypen zweilagig ausgeführt ist, kann das Layout idealerweise so gestaltet werden, dass je zwei Leitungen in großer Entfernung auf einem Layer geführt werden. Ist dies nicht möglich, so könnten Masseleitungen zwischen den einzelnen Signalleitungen vorgesehen werden.

Auch bei den Kabeln, die zwischen den Modulen verlegt werden, besteht die Möglichkeit zu einfachen Verbesserungen. Um Ketten aus den Prototypen zu bilden wurden ungeschirmte Flachbandkabel verwendet. Dies wirkte sich wegen der relativ geringen Länge dieser Kabel noch nicht negativ auf die Funktionsstabilität aus. Für den produktiven Einsatz ist es jedoch ratsam Kabel zu verwenden, die Störsignale besser abschirmen. Es bietet sich dabei an, so genannte Twisted-Pair-Kabel der Kategorie 5 zu verwenden. Diese finden als Netzwerkkabel breite Verwendung und sind daher günstig und leicht zu beschaffen. Darüber hinaus sind diese Kabel für Betriebfrequenzen bis zu 100 MHz spezifiziert und gut gegen Störsignale abgeschirmt. Auch die dazugehörigen Steckverbinder für Platinen sind einfach zu beziehen und verarbeiten.

Das Hauptaugenmerk der weiteren Softwareentwicklung sollte nach den Ergebnissen der Testphase darin liegen, den Logikaufwand im CPLD so gering wie möglich zu halten. Es hat sich gezeigt, dass die Probleme, die durch die asynchrone Auslegung entstehen, umso einfacher zu lösen sind, je weniger getaktete Komponenten verwendet werden. Das heißt, nur wenn die Logik, die zum Betrieb des Bus und zur Fehlererkennung implementiert wird, ein gewisses niedriges Niveau an Komplexität nicht übersteigt kann ein funktionssicheres System aufgebaut werden.

Mit diesen Modifikationen wird sich die Betriebsfrequenz und insbesondere die physikalische Laufzeitverzögerung weiter minimieren lassen und ein system ermöglichen, dass den gestellten Anforderungen noch besser gerecht wird.

Literaturverzeichnis

[ADN09a] Analog Devices Incorporated (Hrsg.): *ADN4665 - 3 V, LVDS, Quad, CMOS Differential Line Driver*. One Technology Way, P.O. Box 9106, Norwood, MA 02062-9106, U.S.A.: Analog Devices Incorporated, 5 2009

[ADN09b] Analog Devices Incorporated (Hrsg.): *ADN4666 - 3 V, LVDS, Quad, CMOS Differential Line Receiver*. One Technology Way, P.O. Box 9106, Norwood, MA 02062-9106, U.S.A.: Analog Devices Incorporated, 6 2009

[Bö99] Böhmer, Michael: *Das Auslesesystem für den Ringabbildenden Čerenkovdetektor im HADES Spektrometer*, Technische Universität München, Diplomarbeit, 1999

[Ben10] Bendel, Michael: *Entwicklung und Test einer digitalen Auslese für das CALIFA-Kalorimeter*, Technische Universität München, Diplomarbeit, 2010

[Gut06] Gutbrod, H. H. (Hrsg.): *FAIR Baseline Technical Report*. FAIR GmbH, 2006 (FAIR Baseline Technical Report)

[Hof10] Hoffmann, J. ; GSI Helmholtzzentrum für Schwerionenforschung GmbH (Hrsg.): *FEBEX1/8, preliminary specification*. 0. Planckstraße 1, 64291 Darmstadt: GSI Helmholtzzentrum für Schwerionenforschung GmbH, 2 2010

[Hof11a] Hoffmann, J. ; GSI Helmholtzzentrum für Schwerionenforschung GmbH (Hrsg.): *FEBEX2, preliminary specification*. 0. Planckstraße 1, 64291 Darmstadt: GSI Helmholtzzentrum für Schwerionenforschung GmbH, 1 2011

[Hof11b] Hoffmann, J. ; GSI Helmholtzzentrum für Schwerionenforschung GmbH (Hrsg.): *FEBEX3/16, preliminary specification*. 0. Planckstraße 1, 64291 Darmstadt: GSI Helmholtzzentrum für Schwerionenforschung GmbH, 1 2011

[LVD04] Texas Instruments Incorporated (Hrsg.): *HIGH-SPEED DIFFERENTIAL RECEIVERS - SN65LVDS33, SN65LVDT33 SN65LVDS34, SN65LVDT34*. Texas Instruments, Post Office Box 655303, Dallas, Texas 75265: Texas Instruments Incorporated, 2004

[LVD07] Texas Instruments Incorporated (Hrsg.): *HIGH-SPEED DIFFERENTIAL LINE DRIVERS - SN55LVDS31, SN65LVDS31 SN65LVDS3487, SN65LVDS9638*. Texas Instruments, Post Office Box 655303, Dallas, Texas 75265: Texas Instruments Incorporated, 2007

[TSR08] R^3B Collaboration: TECHNICAL STATUS REPORT FOR CALIFA: THE R^3B Calorimeter for Photons and High Energy Charged Paticles. 2008. – Forschungsbericht

Abbildungsverzeichnis

Anhang A

Glossar

BOT Bezeichnung für das letzte Modul einer Kette. Es deserialisiert den Datenstrom.

CALIFA **CAL**orimeter for **I**n**F**light emitted gammas and light charged p**A**rticles for R^3B

CPLD Complex Programmable Logic Device. Programierbarer Logikbaustein auf dem Triggermodul.

EP Event Player. Funktion der FPGA-Software die veränderliche Datenwerte simuliert.

ER Event Recorder. Funktion der FPGA-Software die eine gewissen Anzahl von Samples vor und nach einer Triggerauslösung aufzeichnet.

FPGA Field Programmable Gate Array. Programierbarer Logikbaustein auf dem ADC-Modul.

IP Information Provider. An das Triggermodul angeschlossener Datenlieferant.

LVDS **L**ow **V**oltage **D**igital **S**ignaling

MID Bezeichnung für alle mittleren Module in einer Kette. Sie addieren ihre jeweiligen Werte zu denen auf den Bus.

R^3B A next generation experimental setup for studies of **R**eactions with **R**elativistic **R**adioactive **B**eams

Sample Ein Datenwort, das über den Bus gesendet wurde.

TOP Bezeichnung für das erste Modul in einer Modulkette. Es erzeugt das Takt- und BIT0-Signal.

Trace Gesamtheit mehrerer aufeinander folgender Samples.

VHDL **V**ery High Speed Integrated Circuit **H**ardware **D**escription **L**anguage

Anhang B

Handbuch zur FPGA-Software

Wie unter 3.1 erwähnt, kann die FPGA-Software mittels eines Statusregisters, das über das Glasfasernetzwerk gelesen und geschrieben werden kann, gesteuert werden. Darüber hinaus, sind abhängig von der Busrolle verschiedene andere Register zugänglich, mit denen die Funktion der Software überprüft und Event Player bzw. Recorder gesteuert werden können. Tabelle B.1 zeigt eine Übersicht über die verschiedenen Register und deren Belegung. Das Stausregister ist in mehrere Bereiche aufgeteilt, deren Funktion Tabelle B.2 entnommen werden kann. Es ist dabei zu beachten, dass einige Bits nicht geschrieben werden können. Die Tabelle verwendet die üblichen Abkürzungen und die Zustandscodes, die in Tabelle B.3 zusammengefasst sind. Die Statuscodes wurden soweit wie möglich ähnlich gewählt, um die Handhabung zu vereinfachen. Dabei ist insbesondere anzumerken, dass der Statuscode 0 konsequent den Zustand idle markiert.

Die Handhabung der Simulationstools ist vollkommen analog für Event Player und Recorder, mit dem einen Unterschied, dass die Register des EP nur geschrieben und

	Busrolle		
Register	BOTtom	MIDdle	TOP
0x9200	Energieschwelle	Debug Energie	
0x9201	Multiplizitätsschwelle	Debug Multiplizität	
0x9202	—	—	$\frac{100\text{MHz}}{f_{CLK}} + 1$*
0x9203	Status Register		
0x9204	ER Adresse	*EP Adresse*	
0x9205	ER Daten	*EP Daten*	
0x9206	Energie	—	—
0x9207	Multiplizität	—	—

Tabelle B.1: Zusammenfassung aller erreichbaren Register der FPGA-Software. Register in normaler Schrift sind schreib- und lesbar. *Kursive Schrift* kennzeichnet Register, die nur gelesen werden können und Register die nur geschrieben werden können sind unterstrichen. * Das Register 0x9202 ist nur bei Softwareversionen vor der Umstellung der Takterzeugung auf PLL nutzbar.

...	7	6	5	4	3	2	1	0
...			*PLL*	EP	ER		*BR*	

Tabelle B.2: Übersicht über die Belegung des Statusregisters. Auch hier kennzeichnet *kursive Schrift* Register, die nur gelesen werden können. Die Statuscodes der einzelnen Beriche sind in B.3 zu finden.

Statuscode	0	1	2	3
Busrolle	TOP	MID	BOT	—
Event Recorder	idle	playing	—	—
Event Elayer		armed	recording	ready

Tabelle B.3: Zusammenfassung aller Statuscodes.

die des ER nur gelesen werde können. In Beiden Fällen kann immer das Register des Speichers gelesen oder geschrieben werden, dessen Adresse im entsprechenden Adressregister gesetzt ist. Dabei ist zu beachten, dass der Speicher des EP maximal 1023 Einträge fasst. Der Speicher des ER fasst 2046 Einträge, wobei der Zeitpunkt der Triggerauslösung zwischen den Samples 1023 und 1024 liegt.

Anhang C

Beschreibung der Managementprogramme

Um die Arbeitsabläufe beim Test und Betrieb des Triggersystems zu straffen, wurde ein Satz von Hilfsprogrammen entwickelt, die als Shell Scripts auf dem ETRAX Modul laufen. In der folgenden Auflistung sind sind jeweils der Aufruf, die Funktion und die Parameter der Programme zusammengefasst.

cti.sh `./cti.sh <address>`

Dieses Script detektiert die Busrolle des Moduls mit der entsprechenden Adresse anhand dessen Statusregister und zeigt, abhängig von der Busrolle, die relevanten Statusinformationen an. Dies umfasst in allen Fällen die aktuellen Werte für Energie und Multiplizität sowie den Status von PLL und Event Player für TOP Module. Bei BOT Modulen wird der Status des Event Recorders und die Schwellenwerte angezeigt.

cts.sh `./cts.sh <address>`

Mit diesem Script können, abhängig von der Busrolle des Moduls, alle relevanten Parameter geändert werden. Die Änderungen können dabei interaktiv vorgenommen werden.

fill_eventplayer_file.sh `./fill_eventplayer_file.sh <address> <datfile>`

Mit diesem Programm kann der Event Player des entsprechenden Moduls mit den Werten aus einer Textdatei gefüllt werden. Diese muss je Zeile einen Datensatz mit Adresse und Wert, durch Tabulator getrennt, enthalten.

fill_eventplayer_logic.sh `./fill_eventplayer_loop.sh <address>`

Dieses Script erlaubt das Befüllen des Event Players bei `adress` durch im Script zu definierende Logik.

dump_er.sh `./dump_er.sh <address> [from] [to]`

Mit diesem Programm ist es möglich, einen Auszug aus dem Event Recorder eines BOT Moduls zu erhalten. Die optionalen Parameter `from` und `to` begrenzen dabei die Menge an Samples, die ausgegeben wird. Konkret werden `from` Samples vor und `to` Samples nach dem Auslösen Triggers ausgegeben. Wird einer bzw. beide Parameter nicht angegeben, so wird standardmäßig 1024 angenommen.

prog.sh `./prog.sh <address> <bitfile>`

Dieses Script automatisiert das Programmieren und neu laden des Moduls mit der Adresse `address`.

Alle diese Programme verfügen nur über rudimentäre Eingabeprüfung, so dass falsche bzw. unpassende Eingaben zu unvorhersehbarem Verhalten führen können. Die beiden ersten Programme verwenden die Textdatei `codes.txt`, die im selben Verzeichnis angenommen wird. Sie enthält eine Auflistung der möglichen Statusregistereinträge um die Busrolle und den Status der Module zu bestimmen.

Anhang D

Ergebnisse der Charakterisierung der Prototypen

Mod.	CLK		DAT0	
Nr.	d_{ges}	d_{diff}	d_{ges}	d_{diff}
1	9.00 ± 0.19	6.49 ± 0.21	9.27 ± 0.10	7.17 ± 0.14
3	8.45 ± 0.25	6.02 ± 0.25	9.30 ± 0.10	6.40 ± 0.14
4	9.34 ± 0.24	6.83 ± 0.25	9.28 ± 0.09	6.88 ± 0.10
6	9.40 ± 0.19	7.60 ± 0.28	9.30 ± 0.08	7.30 ± 0.13
2	9.46 ± 0.17	6.96 ± 0.20	9.98 ± 0.43	8.22 ± 0.41
5	9.57 ± 0.16	7.05 ± 0.18	9.43 ± 0.20	7.57 ± 0.36
7	8.20 ± 0.24	5.80 ± 0.31	8.40 ± 0.40	5.57 ± 0.41
8	8.37 ± 0.26	5.75 ± 0.27	8.50 ± 0.07	5.41 ± 0.08
9	8.42 ± 0.24	6.07 ± 0.25	8.54 ± 0.07	5.69 ± 0.08
10	7.86 ± 0.17	5.86 ± 0.26	7.88 ± 0.11	5.20 ± 0.12

Tabelle D.1: Übersicht über die Ergebnisse der Charakterisierung der Prototypen auf ihre zeitlichen Eigenschaften. Die horizontale Linie markiert den Übergang zu den neuen differentiellen Bausteinen. Alle Angaben sind in ns zu verstehen.