Monitoringsystem für das Mu3e Frontendboard

von

Bastian Keßler

Bachelorarbeit in Physik
vorgelegt dem Fachbereich Physik, Mathematik und Informatik (FB 08)
der Johannes Gutenberg-Universität Mainz
am 17. November 2020

1. Gutachter: Prof. Dr. Niklaus Berger
2. Gutachter: Prof. Dr. Volker Büscher

Ich versichere, dass ich die Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie Zitate kenntlich gemacht habe.

Mainz, den 17.11.2020

Bastian Keßler
AG Berger
Institut für Kernphysik
Staudingerweg 9
Johannes Gutenberg-Universität D-55099 Mainz
bkessler@uni-mainz.de

Zusammenfassung

Das Mu3e Experiment soll Aufschlüsse über Physik jenseits des Standardmodells liefern, indem es im Myonzerfall nach dem, im Standardmodell stak unterdrückten, Zerfall in drei Elektronen sucht. Die Datenaufnahme wird im Detektor nicht getriggert, und so müssen bis zu 100 $\frac{GBit}{s}$ Daten aus dem Detektor geleitet werden. Um diese Datenmenge handhaben zu können, werden in der ersten Stufe der Datenauswertung Field-Programmable-Gate-Arrays (FPGAs) eingesetzt, welche mit ihrem Trägerboard im Detektor verbaut werden. Das Trägerboard verfügt dabei über zwei FPGAs: Der Arria5 ist für die Auswertung der Detektordaten verantwortlich und den Max10, welcher für die Überwachung und Kontrolle des Boards im Detektor verantwortlich sein wird. Für die Überwachung wird im Zuge dieser Arbeit die Aufnahme und Auswertung der Sensordaten auf dem Max10 erläutert. Für die Übertragung und Bereitstellung dieser Daten am Arria5 FPGA wurde ein Serial Peripheral Interface verwendet, auf dessen Funktionsweise und Steuerung tiefer eingegangen wird. Mit den Ergebnissen dieser Arbeit ist es möglich in Zukunft eine technische Überwachung außerhalb des Detektors einzurichten.

Inhaltsverzeichnis

1. Suche nach Physik jenseits des Standardmodells

1.1. Das Standardmodell der Teilchenphysik

Das Standardmodell der Teilchenphysik beschreibt die Elementarteilchen aus Abbildung 1.1 und die drei fundamentalen Wechselwirkungen, welche von Bosonen, den Austauschteilchen übertragen werden. Die starke Wechselwirkung wird vom Gluon übertragen und ist für den Zusammenhalt der Quarks in Protonen, Neutronen und damit auch derer im Atomkern verantwortlich. Das Austauschteilchen für die elektromagnetische Wechselwirkung ist das Photon und wirkt auf alle geladen Fermionen, damit bildet es mit Protonen, Neutronen und Elektronen die sichtbare Materie. Die schwache Wechselwirkung wird von $W^{+/-}$ und Z^0 Bosonen übertragen und ist mitunter für den β-Zerfall verantwortlich.

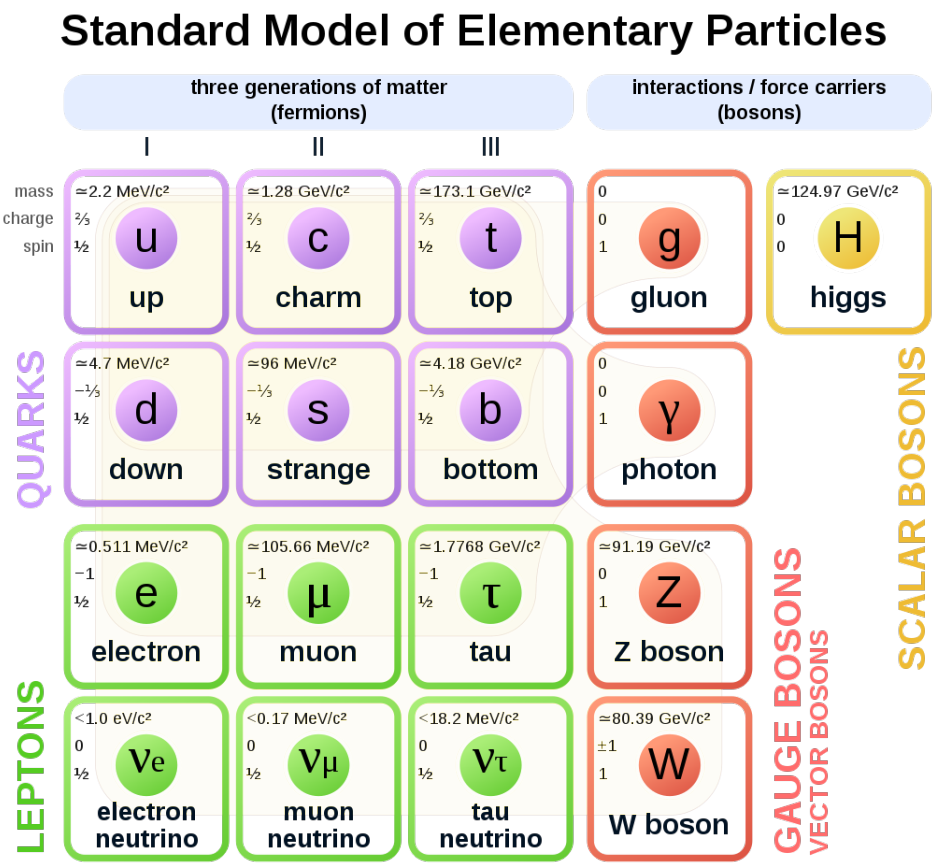

Abbildung 1.1.: Elementarteilchen nach dem Standardmodell der Teilchenphysik [1] .

Mit dem experimentellen Nachweis des Higgsbosons [2] sind alle vorhergesagten Teilchen des Standartmodells auch bestätigt worden. Doch obwohl das Standartmodell viele Mechanismen in der Teilchenphysik beschreibt, hat es wie jedes Modell seine Grenzen.

1.2. Leptonenfamilienzahlverletzung (lepton flavor violation)

Nach dem Standartmodell ist sowohl die Leptonenzahl als auch die Leptonfamilienzahl eine Erhaltungsgröße. Wobei die Erhaltung der Familienzahl aus der Annahme herrührt, dass Neutrinos keine Masse besitzen. Die Leptonenzahl errechnet sich aus der Differenz der Leptonen und der Antileptonen:

$$L = n_l - n_{\bar{l}} \tag{1.1}$$

Die Leptonfamilienzahl dementsprechend bildet sich aus der Differenz in der Leptonfamilie:

$$L_e = n_e - n_{\bar{e}} \ , \ L_\mu = n_\mu - n_{\bar{\mu}} \ , \ L_\tau = n_\tau - n_{\bar{\tau}} \tag{1.2}$$

Die Familien stehen jeweils für eine der 3 Generationen e, μ, τ. So bildet das Elektron zusammen mit dem Elektron-Neutrino und deren Antiteilchen eine Leptonfamilie. Ein Wechsel zwischen den Familien ohne die Erzeugung von Anti/-Neutrinos ist also nicht möglich. Experimente zur Messung von solaren Neutrinos wie z. B. Super-Kamiokande [3] haben aber gezeigt, dass Neutrinos zwischen den Familien oszillieren, ihre Familie also wechseln können und somit auch nicht masselos sein dürfen. Das Ende des Standardmodells bedeutet dies aber nicht, da die Masse der Neutrinos sehr gering ist [1] und somit weiterhin ausreichend gute Vorhersagen möglich sind.
Zerfälle, welche nach dem Standardmodell aber stark unterdrückt sein müssten, da sie die Leptonfamilienzahl verletzten, könnten dadurch wahrscheinlicher sein, als durch das Standardmodell vorhergesagt.

1.3. Myon Zerfall

Nach dem Standartmodell ist der Zerfall eines Myons gegeben durch $\mu^+ \to e^+ + \overline{\nu}_\mu \nu_e$, in seltenen Fällen entsteht auch noch ein e^-/e^+ Paar : $\mu^+ \to e^+ e^- e^+ + \overline{\nu}_\mu \nu_e$
Zerfälle wie:

$$\mu^+ \to e^+ e^+ e^- \tag{1.3}$$

$$\mu^+ \to e^+ + \gamma \tag{1.4}$$

Verletzen die Leptonenfamielienzahlerhaltung, sind demnach bei masselosen Neutrinos theoretisch mit einem Verzweigungsverhältnis von O^{-50} stark unterdrückt und somit experimentell nicht nachweisbar. Sollte also einer dieser Zerfälle gemessen werden, wäre dies ein starker Hinweis auf Physik jenseits des Standardmodells.

[1] Zum Vergleich: $m_\nu < 10^{-6} m_e$ [4].

2. Das Mu3e Experiment

Das Mu3e Experiment wird am Paul-Scherrer Institut (PSI) durchgeführt werden, um den nach dem Standardmodell stark unterdrückten $\mu^+ \to e^+e^+e^-$ Zerfall zu bemessen. Das Experiment soll dafür insgesamt über 10^{16} Zerfälle dokumentieren. Um diese Anzahl zu erreichen, benötigt es, bei einer Rate von $> 10^9$ Zerfällen/s, 100 Tage kontinuierliche Strahlzeit[8].

2.1. Aufbau

Als Myonenquelle dienen, durch die Kollision von 590 MeV Protonen mit dem Kohlenstoff Target-E, erzeugte Pionen, welche nach $\pi^+ \to \mu^+\nu_\mu$ in Myonen zerfallen. Zurzeit kann eine Rate von $10^8 \frac{\mu}{s}$ erzeugt werden, dies soll aber im Zuge des High Intensity Muon Beam Projekts (HiMB) auf bis zu $10^{10} \frac{\mu}{s}$ erhöht werden[9]. Dies bedeutet eine deutliche Reduzierung der Strahlzeit, aber auch eine Erhöhung der Anforderung an die Datenentnahme.

2.2. Detektor

Zur Unterscheidung der Zerfälle bestimmt man die Gesamtenergie der entstandenen e^-/e^{+1}. Ist diese geringer als die Masse[2] des zerfallenen Myons, müssen noch Neutrinos erzeugt worden sein, welche die fehlende Energie enthalten. Da Neutrinos sehr leicht sind, muss die Impulsauflösung[3] des Detektor sehr hoch sein. [8]. Um dies zu erreichen, durchlaufen die $e^{+/-}$, in einem 1 Tesla starken Magnetfeld, eine Doppellage von Pixelsensoren[4], gefolgt von szintillierenden Fasern und dann eine weitere Doppellage von Pixelsensoren. Wenn sie dann wieder, bei der Rückkehr im Magnetfeld, auf den Detektor treffen, durchlaufen sie wieder eine Doppellage von Pixelsensoren und abschließend eine Lage Szintillatoren(Siehe Abbildung 2.1). Die Pixelsensoren dienen hierbei der Spur- und damit Impuls-/Energiebestimmung. Die Szintillatoren liefern die zeitliche Einordnung.

[1] Elektron/Positron.
[2] Myonen werden am Target zu stillstand gebracht und Zerfallen dort in Ruhe.
[3] Aus dem Impuls lässt sich die Energie berechnen.
[4] Bei dem verwendeten MuPix handelt es sich um einen High Voltage Monolithic Active Pixel Sensor (HVMAPS)[5].

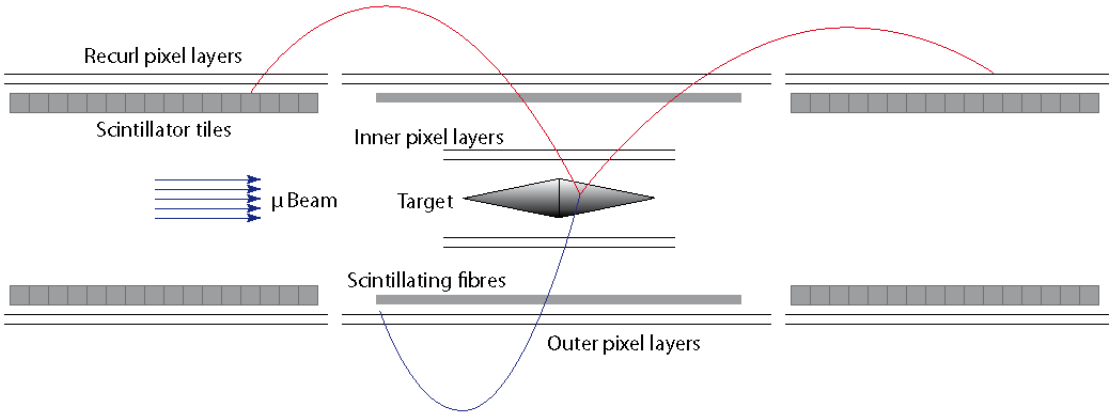

Abbildung 2.1.: Mu3e Detector Schema, entnommen aus: [6]

2.3. Datenverarbeitung

In der ersten Phase der Datenentnahme zu einer existierenden Strahlführung sollen noch bis zu $80\frac{GBit}{s}$ Daten an den Sensoren erzeugt und verarbeitet werden, später wird die Aufnahme auf bis zu $1\frac{TBit}{s}$ erhöht. Die dafür entwickelte Datenauslese erfolgt über 3 Stufen (Abbildung 2.2). Die erste Stufe bilden speziell entworfene Frontendboards (FEBs)[5], welche die Rohdaten eines Sensortyps ordnen und zu Paketen zusammenfügen. Die Pakete werden dann über optische Leitungen zu den Switching-Boards geleitet und dort mit den Daten aus den anderen Sensoren verknüpft. Abschließend werden in einer Rechenfarm die Zerfälle rekonstruiert.

[5] Auf den Aufbau wird in der Arbeit noch vertiefend eingegangen.

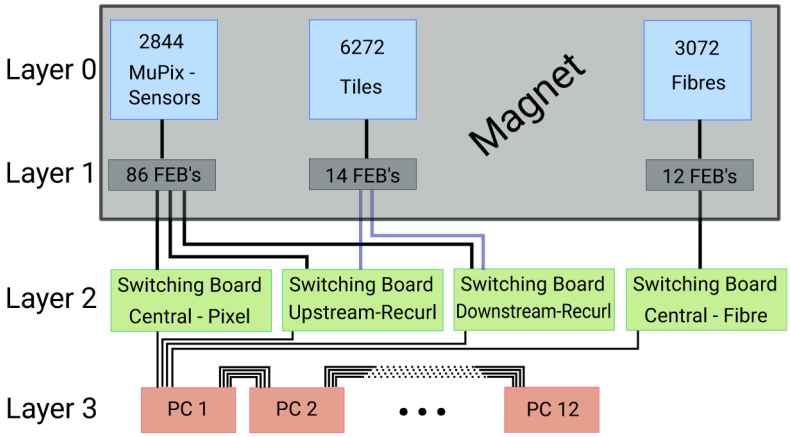

Abbildung 2.2.: Datenlesesystem des Mu3e Experiments, entnommen aus [7].

3. FPGAs

In den ersten beiden Stufen der Datenauswertung kommen programmierbare Logikgatter, (englisch Field Programmable Gate Arrays) kurz FPGAs zum Einsatz. FPGAs können dabei auf Hardwareebene für spezielle Operationen konfiguriert werden und erreichen dadurch einen deutlich höheren Datendurchsatz als die vielseitig einsetzbaren seriellen Prozessoren[1].

3.1. Technische Einführung

Ein FGPA besteht aus einer Aneinanderreihung von Adaptive-Logic-Modules (ALMs, Siehe Abbildung 3.1), welche anhand von Lookup-Tabellen(LUTs), jede individuelle $N \rightarrow 1$ Logik zwischen den Registern abbilden können. Das in Abbildung 3.1 gezeigte ALM verfügt über 8 Inputs. Falls dies nicht ausreicht, können ALMs hintereinander geschaltet werden. Hierfür wird das Ergebnis des ALMs nicht in das Register gespeichert, sondern direkt über combout(0) weitergegeben.
Die Programmierung des FPGAs erfolgt also über Festsetzung der LUT, welche in einem Static Random-Access-Memory (SRAM) gespeichert werden (Abbildung 3.2) und der Verdrahtung der ALMs. Da SRAMs ihre Information nach dem Wegfallen der Betriebsspannung verlieren, muss dieser bei jedem Neustart des FPGAs neu beschrieben werden. Die Konfiguration erfolgt vor allem in der Entwicklung über einen angeschlossen PC[2]. Wenn die Konfiguration aber nicht mehr geändert werden soll, kann auch die Programmierung auf einem angeschlossenen Flash Speicher[3] gespeichert werden und von dort durch einen Mikrocontroller beim Anschalten des FPGAs auf diesen geladen werden. Ein FPGA kann dabei selbst auch als Mikrocontroller dienen um einen anderen zu programmieren.

[1] z.B. CPU.
[2] Zur Kommunikation zwischen PC und FPGA wurde JTAG verwendet [11].
[3] Flash Speicher sind "nicht flüchtige" Speicher.

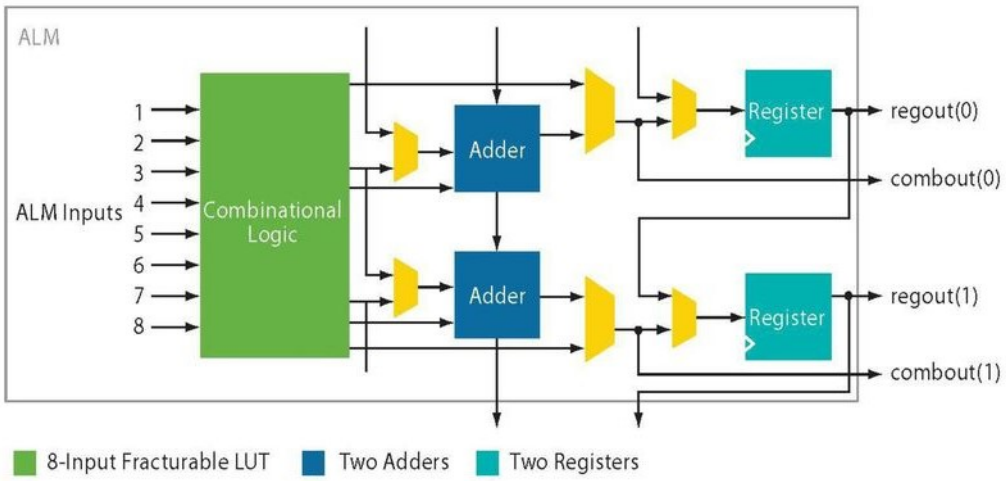

Abbildung 3.1.: ALM Block Diagramm von Altera [10].

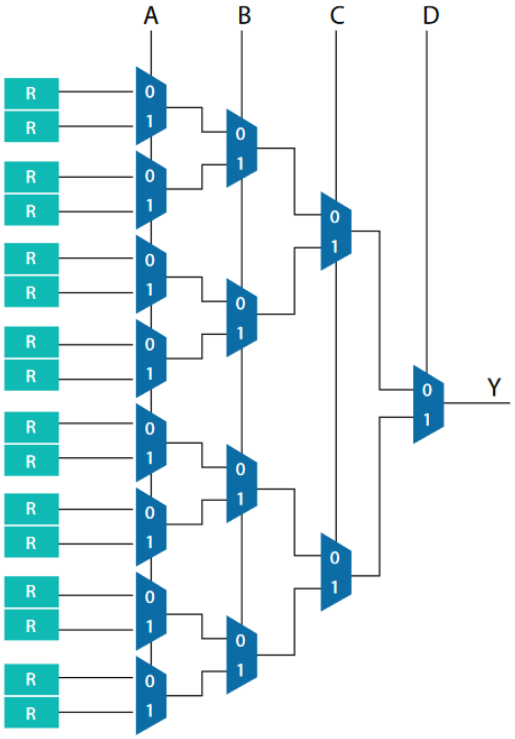

Abbildung 3.2.: $4 \rightarrow 1$ LUT bestehend aus 16 Registern [10].

3.2. Clocking und Synchronisation

Alle Register in einem FPGA haben definierte Zeitfenster, bei denen das zu speichernde Signal stabil sein muss. Dieses Zeitfenster setzt sich aus der Setup-Time t_{SU} und der Hold-Time t_H zusammen. Wechselt das Signal in diesem Zeitfenster, gibt es drei Möglichkeiten: Das Register übernimmt den alten Wert, es übernimmt den neuen Wert oder das Register nimmt einen metastabilen Zustand an, siehe Abbildung 3.3.

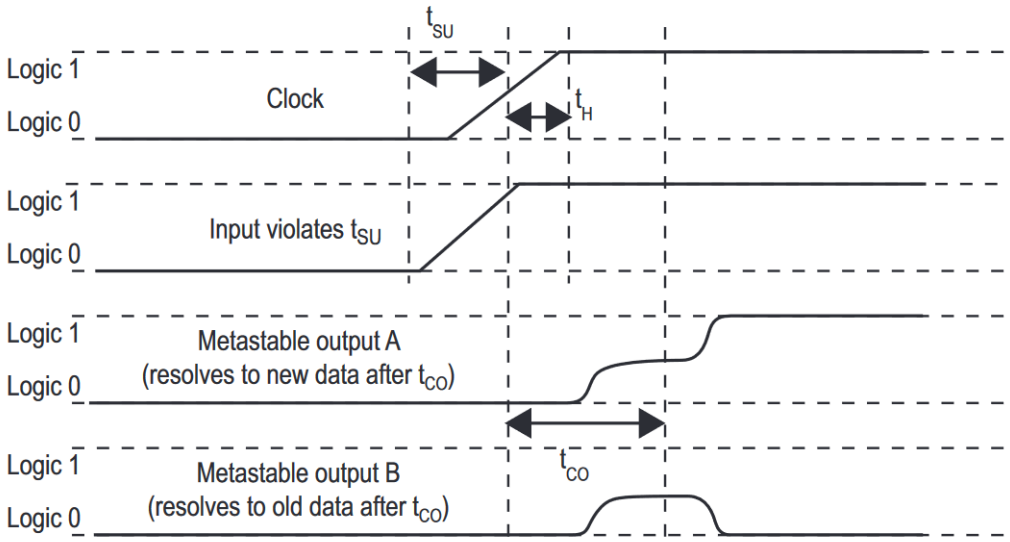

Abbildung 3.3.: Signaldiagramm eines metastabilen Zustands. t_{CO} ist die Zeit zwischen zwei steigenden Clock Flanken und Registerausgabe. Entnommen aus [12].

Solche Timing Probleme treten auf, wenn aufgrund zu hoher Propagationszeit t_{prop} das Signal in dem Zeitraum von t_{SU} und t_H das Register erreicht. Dies tritt ein, wenn die Zeit, die benötigt wird, um das Signal aus dem Register auszugeben und es durch die Logik zu treiben, größer ist, als Differenz von Clockperiode (t_{clk}) und Setup-Time t_{SU}. Für t_{prop} muss also gelten:

$$t_H - t_{CO} < t_{Prop} < t_{clk} - t_{SU} - t_{CO} \tag{3.1}$$

Gründe für eine zu hohe Propagationszeit kann ein ineffizienter Algorithmus sein, welcher zu viel Logik zwischen den Registern benötigt. Dies kann dann nur mit einer Überarbeitung gelöst werden. Häufiger liegt der Grund aber in der Platzierung der Logikgatter durch den Compiler, wodurch bei schlechter Platzierung lange Wege zwischen Bauteilen auf dem Board entstehen können. Da die Platzierung durch den Compiler immer einen Zufallsfaktor beinhaltet, kann zwar durch wiederholtes kompilieren ein funktionierendes Design erzeugt werden. Bei größeren Projekten kann die Kompilierung aber auch schon mehrere Stunden dauern, wird dadurch sehr zeitintensiv und das Endprodukt sollte nicht auf Zufall basieren.

Um Timing Probleme zu beheben, bietet Quartus Prime[4] einen Timing-Analysator, um mögliche Timing Verletzungen erkennen und beheben zu können.

3.3. CLk Übergänge und FIFOs

Eine weitere Quelle für Metastabilität ist Asynchronität, sei es ein asynchroner externer Input oder der Übergang zwischen asynchronen Clock-Bereichen. Um Daten zwischen diesen Bereichen zu übermitteln, müssen diese erst synchronisiert werden. Für einzelne Signale verwendet man eine Synchronization-Chain(Abbildung 3.4), bei der die Daten erst durch mehrere Register laufen und so ein metastabiler Zustand mehr Zeit und Gelegenheit hat, sich aufzulösen.

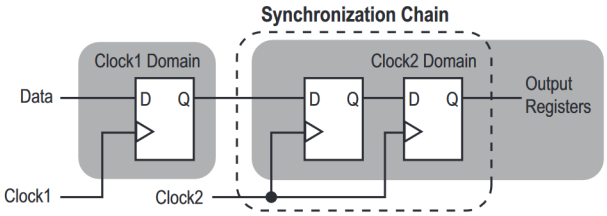

Abbildung 3.4.: Synchronization-Chain bestehend aus zwei Registern, entnommen aus [12]

.

Für die Übertragung vieler Daten eignet sich eine Synchronization-Chain nicht, da sie nur metastabile Zustände reduziert, der übertragene Bit kann dennoch falsch sein. Wenn also eine Timingverletzung auftritt und n Bits versendet werden, kann jeder dieser n Bits entweder den alten oder neuen Wert annehmen. Man möchte also für eine Übertragung zunächst die Signale stabilisieren und im Idealfall erst dann eine Abfrage zulassen. Hier kommen asynchrone First-In-First-Out(A-FIFO) Speicher zum Einsatz. Ein FIFO besteht aus einem Array von Registern[5] und einem Lese- bzw. Schreibzeiger (read/write Pointer), wobei der write Pointer immer auf das nächste zu beschreibende Register zeigt und der read Pointer auf das nächste zu lesende. Wenn der FIFO gelesen bzw. beschrieben wird, erhöht sich der jeweilige Pointer. Wenn er dabei über die letzte Adresse des FIFOs kommt, zeigt er stattdessen auf die erste Adresse(Siehe: Abbildung 3.5). Sobald der read und write Pointer auf das selbe Register zeigen, also den selben Wert besitzen, gilt der FIFO als leer, wenn der read Pointer den write Pointer eingeholt hat, oder als voll, wenn der write den read pointer überrunden würde.

[4]Quartus Prime ist die Entwicklungssoftware für Altera und Intel FPGAs, welche in dieser Arbeit verwendet wurde.
[5]I.d.R. SRAM.

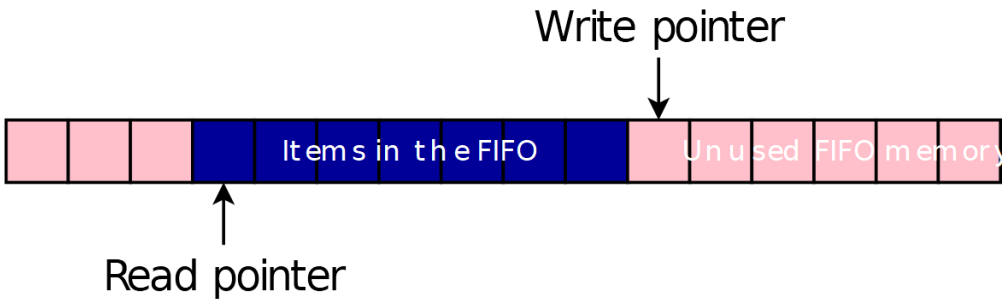

Abbildung 3.5.: Beispiel eines halbvollen FIFOs. Die Pointer wandern dabei nach rechts, entnommen aus [20]

Bei einem asynchronen FIFO werden wie in Abbildung 3.6 die Daten von der einen Bereichs-Clock eingespeichert und von der anderen ausgelesen. Die Schwierigkeit liegt hier bei der Synchronisation der Pointer, da diese angeben, ob ein FIFO voll oder leer ist. Da bei binärer Zählweise n Bits geändert werden, können im Falle eines metastabilen Zustandes 2^n verschiedene Zustände entstehen. Beim Hochzählen von 0111_b auf 1000_b werden zum Beispiel alle Bits geändert. Wird der Zähler nun zu früh abgefragt, kann jede Zahl zurückgegeben werden und der FIFO fälschlicherweise ein leer oder voll zurückgeben. Um dies zu verhindern, sind die Zähler für die Pointer in Gray Code codiert [19], sie verändern also beim hoch- bzw. runterzählen wie in Tabelle 3.1 zu sehen immer nur ein Bit. Im schlimmsten Fall wird statt dem neuen nur der alte Wert abgefragt.

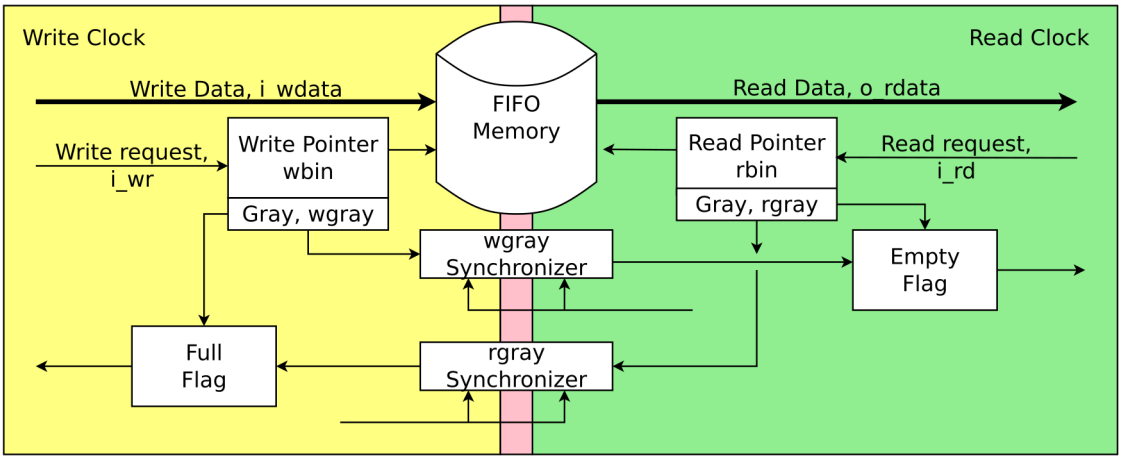

Abbildung 3.6.: Schema eines asynchronen FIFOs, im roten Bereich können metastabile Zustände auftreten. Entnommen aus [20].

Counter		
Decimal	Binary	Gray Code
0	000	000
1	001	001
2	010	011
3	011	010
4	100	110
5	101	111
6	110	101
7	111	100

Tabelle 3.1.: 3 Bit Gray Codierung eines Zählers

.

4. Frontendboard

Die erste Stufe der Datenauslese findet auf Frontendboards (FEBs) statt. Hierbei werden die Daten der Detektoren von einem FPGA, dem Arria5 [18], gesammelt, sortiert und über Glasfaser aus dem Detektor geleitet. Da die FEBs später im Detektor verbaut und somit schwer erreichbar sein werden, muss die Programmierung des Arrias und die Überwachung vom FEB von außerhalb des Detektors steuerbar sein. Für diesen Zweck befindet sich ein kleinerer FPGA auf dem FEB, der MAX10 [14] [13].

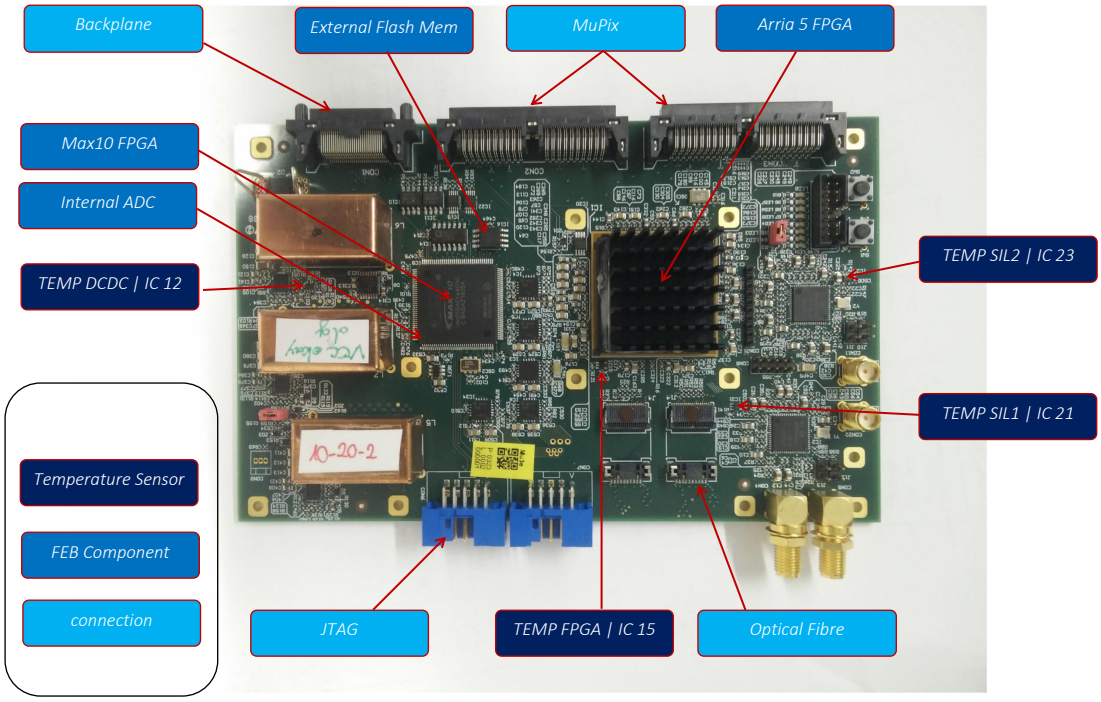

Abbildung 4.1.: FEB Schema

Das Herzstück des Boards bildet der Arria5 FPGA. Dieser verarbeitet die von den MuPix Sensoren kommenden Daten, welche dann von optischen Transceivern[1] zur weiteren Verarbeitung aus dem Detektor heraus gesendet werden. Neben dem Anschluss an eine Backplane wird der optische Anschluss die einzige Möglichkeit

[1] Auf dem Board werden FireFlys von samtec verwendet.

für eine Kommunikation mit den FEBs im Detektor sein. Die Stromversorgung von 20V findet im Detektor über die Blackplane statt. Die zur Spannungsumwandlung benötigten Spulen[2] verfügen über keinen Eisenkern, da sie im Magnetfeld des Detektors operieren und fallen daher größer aus.
Im Labor werden die beiden FPGAs noch über einen JTAG-Anschluss programmiert und überwacht. Später im Detektor besteht keine direkte Verbindung mehr zu einem Computer, die Programmierung muss also auf dem Board stattfinden. Hierfür besitzt der Max10 einen internen Flash-Speicher, mit dem dieser sich selbst programmieren kann, danach programmiert er den Arria5, die Daten dafür befinden sich auf einem externen Flash Speicher. Während die Programmierung des Max10 nur über JTAG geändert werden kann, wofür das FEB aus dem Detektor entfernt werden müsste, kann der externe Flash und damit die Programmierung des Arria auch in dem Detektor über die optischen Anschlüsse oder die Backplane via MSCB[3] angepasst werden.

4.1. Sensoren

Zur Überwachung des FEBs besitzt der Max10 einen fest eingebauten Analog-zu-Digital-Konverter (ADC), mit dem es möglich ist, Spannungen bis zur Referenzspannung von 2.5 V in 12 Bit Daten umzuwandeln. An den ADC sind folgende Spannungen angeschlossen: Zur Überwachung der Netzspannung ist diese über einen 1 zu 10 Spannungsteiler an VCC20 angeschlossen. Zudem sind noch mit VCC 3V3, VCC 2V5, VCC 1V8 und VCC 1V1 feste Sollspannungen von 3.3V , 2.5V ,1.8V und 1.1V mit dem ADC verbunden, wobei VCC 3V3 über der Referenzspannung liegt und der ADC somit seinen Maximalwert von 2.5V zurückgibt. Bei VCC 2V5 liegt die Spannung an der Referenzspannung. Der Anschluss für 1.8 V: VCC 1V8 ist durch einen Fehler nicht wie nach dem Schaltplan angeschlossen.
Zur Temperaturüberwachung verfügt der ADC über einen eigenen Onchip-Sensor, die Rückgabewerte sind aber mit einer Auflösung von 1°C recht ungenau und müssen über eine Tabelle des Datenblatts zugeordnet werden [15]. Für die Überwachung befinden sich 4 Temperatursensoren vom Typ TMP235 [16] auf dem restlichen Board verteilt (Siehe Abbildung 4.1). Diese Sensoren bilden einen Temperaturbereich von -40 bis 150°C auf 0.1 bis 2V ab[16]. Die analogen Ausgabespannungen werden dann von dem Max10 ADC digitalisiert.

Um die Komponenten auf dem Max10 leichter zu steuern, ist auf dem Max10 ein 32-Bit RISC Prozessor implementiert: Der NIOSII [17]. Der NIOSII wurde von Altera speziell für die Verwendung in ihren FPGAs entwickelt und kann mithilfe von Quartus Prime leicht mit anderen Altera-Komponenten, wie auch dem ADC, verbunden werden. Der Nios lässt sich programmieren und seine Terminalausgabe

[2] von den Kupfergehäusen abgedeckt, siehe Abbildung 4.1.
[3] MSCB steht für Midas Slow Control Bus [7].

kann über JTAG an einem PC ausgelesen werden. Dies erleichtert das Testen von angeschlossenen Komponenten.

Channel	Sensor
0	VCC 20
1	VCC 3.3 V
2	VCC 2.5 V
3	VCC 1.8 V
4	VCC 1.1 V
5	TEMP DCDC
6	TEMP FPGA
7	TEMP SIL2
8	TEMP SIL1
9	TSD Onchip-Sensor

Tabelle 4.1.: ADC Channel Konfiguration

4.2. Kommunikation mit dem FEB

Die Software, welche die Daten anfordert und verarbeitet, heißt Midas [4] und wird im PSI und bei TRIUMF zur Auslese, Steuerung und Überwachung von Experimenten entwickelt. Die vom Max10 gemessenen Spannungen und Temperaturen sollen vom Max10 via dem Arria5 über den selben optischen Link wie die Daten aus dem Detektor heraus übertragen werden. Eine detaillierte Beschreibung der Datenübertragung findet sich in [7].
Der für diese Arbeit relevante Anschluss an Midas ist das Slowcontol-Register, ein Speicher, an den Daten gesendet oder abgerufen werden können. Das Slowcontrol-Register befindet sich auf dem Arria5, eine direkte Verbindung von MIDAS zum Max10 besteht nicht. Kontrolldaten wie die ADC-Daten, welche auf dem Max10 aufgenommen werden, müssen also erst auf den Arria5 gebracht werden, um von dort an Midas geschickt werden zu können. Umgekehrt müssen Daten, die an den Max10 gesendet werden sollen auch über den Arria5 geleitet werden. Für diese Kommunikation wird ein Quad-SPI verwendet, dessen Funktionsweise und Einrichtung im späteren Teil der Arbeit erarbeitet wird.

[4] Maximum Integrated Data Acquisition System

5. Übertragung der Daten vom Max10 zum Mu3e Slowcontrol-System

Zwischen dem Arria5 und dem Max10 sollen hauptsächlich die ADC Daten versendet werden, aber auch Kontrolldaten, Instruktionen und gegebenenfalls eine neue Programmierung für den Arria5. Da diese Daten meist mehrere Bits groß sind und die FPGAs nur über eine begrenzte Anzahl an Anschlüssen verfügen, ist eine parallele Übertragung der Daten[1] von dem Slowcontrol-Register zu den Komponenten auf dem Max10 nicht möglich. Alternativ zur parallelen Übertragung von Daten ist eine serielle Übertragung, bei der die Daten aufgeteilt, stückweise versendet und wieder zusammengesetzt werden.

5.1. SPI

Ein Serial-Peripheral-Interface kurz SPI, besteht aus einem SPI-Master und einem oder mehreren SPI-Slaves(Abbildung 5.1), wobei der SPI-Master mit setzten des Chip-Select[2](CS)-Signals einen bestimmten Slave ansprechen kann. Der Master sendet über die Clock-Verbindung die Clock, dies hat den Vorteil, dass Slave und Master sich im selben Clock Bereich befinden, wodurch das Auftreten metastabiler Zustände stark reduziert wird. Die Daten werden dann über die Daten-Verbindung gesendet. Hierfür kann man für das Senden und Empfangen dedizierte Verbindungen verwenden, so für das Senden vom Master, den Master-Out-Slave-In (MOSI), bzw. Master-In-Slave-Out (MISO) für das Empfangen von Daten vom Slave. Um die Anzahl an Verbindungen zu reduzieren, kann man die Verbindungen sowohl für das Senden als auch das Empfangen verwenden. Da es dabei aber möglich ist, dass ein Kurzschluss entsteht[3], benötigt dies mehr Logik im Master bzw. Slave.
Die Datenübertragung selbst kann in zwei Phasen aufgeteilt werden: Zunächst sendet der Master die gewünschte Adresse der anzusprechenden Komponente oder der Register, zusammen mit einem Lese- oder Schreibbefehl(read/write request). Danach sendet der Master im Falle eine write request die Daten. Liegt ein read request vor, sendet der Slave die Daten.

[1] Das Übertragen eines Datenworts von n Bit Größe über n Verbindungen in einem Clock Zyklus.
[2] häufig auch als Slave-Select (SS) bezeichnet.
[3] z.B. der Slave treibt '1' und der Master treibt '0'.

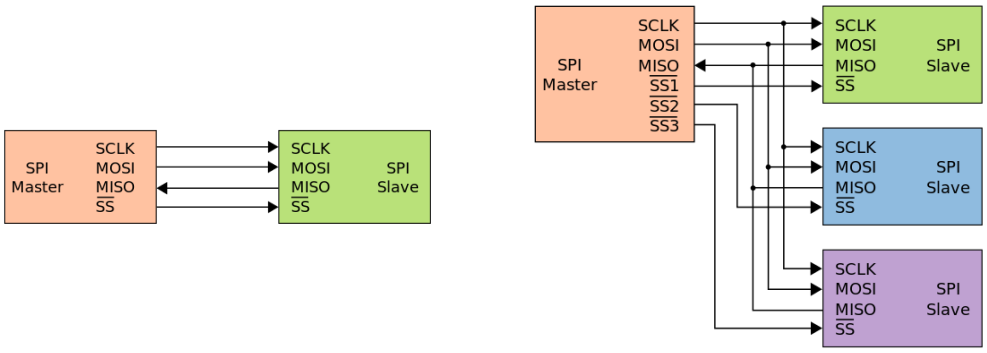

Abbildung 5.1.: (links) SPI mit einem Master und Slave [21]. (rechts) SPI mit einem Master und drei Slaves [22].

5.2. Quad-SPI

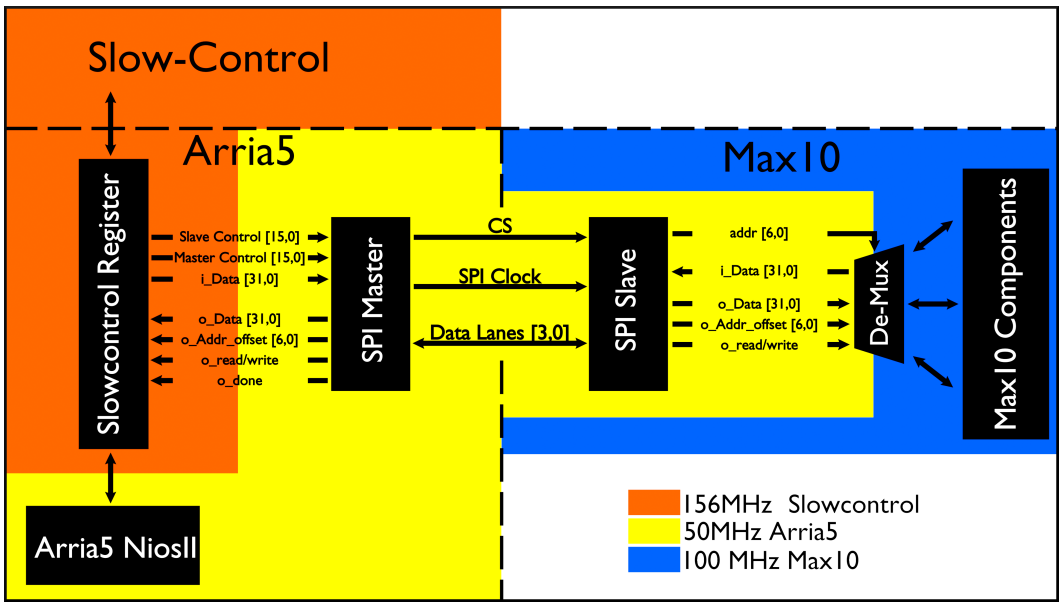

Abbildung 5.2.: SPI Clock Bereiche auf dem Arria5 und Max10. Die Verbindungen beschreiben die Anschlüsse am SPI-Master bzw. SPI-Slave.

Um die Datenübertragungsrate eines SPI zu erhöhen, kann man die Taktfrequenz der treibenden Clock erhöhen, oder man parallelisiert die Datenübertragung zu einem gewissen Grad. Statt die Daten nur über ein Signal zu senden, teilt man sie auf mehrere auf, dabei erhöht sich die Datenrate proportional zur Anzahl an Signalen[4]. Bei dem entwickelten SPI handelt es sich um ein Quad-SPI, da vier

[4]gilt streng genommen nur, wenn die Signalanzahl ein Teiler der Datenwort Länge ist. So braucht

Daten-Signale genutzt werden. Das SPI verfügt über eine Chip-Select-, eine Clock-, die 4 Daten-Verbindung und eine zum Zeitpunkt der Arbeit defekte Verbindung[5].

Wie auf Abbildung 5.2 zu sehen ist, wird das SPI von einer 50 MHz Clock betrieben, welche vom Arria5 bereitgestellt wird. Diese Clock wird als SPI-Clk an SPI-Slave auf dem Max10 weitergeleitet. Wenn die Daten dann zu Komponenten auf dem Max10 gesendet oder empfangen werden, müssen sie die Clock-Grenze zur 100 MHz Clock des Max10 überschreiten. Dadurch, dass die Max10-Clock schneller als die des SPIs ist, steigt die Wahrscheinlichkeit, dass bei der Übernahme von Daten zum Max10 diese gerade aufgenommen werden, wenn sie von SPI Seite aus gesetzt werden und ein falscher bzw. metastabiler Zustand entsteht. Auf der anderen Seite muss bei der Übertragung vom Max10 das Signal lange genug gehalten werden, damit das SPI es aufnehmen kann.
Das Slowcontrol-Register auf dem Arria5 wird mit einer 156 MHz Clock betrieben, was einen weiteren Synchronisationsschritt erfordert. Mit der verwendeten 50 MHz Clock und 4 Daten-Verbindungen kann somit eine maximale Übertragungsrate von bis zu 25 $\frac{Mbyte}{s}$ erreicht werden. Zur Übertragung der 16 Bit SPI Slave-control (Adresse) benötigt es 4 Clock-Zyklen und für die darauf folgenden 32Bit Datenpakete jeweils 8.

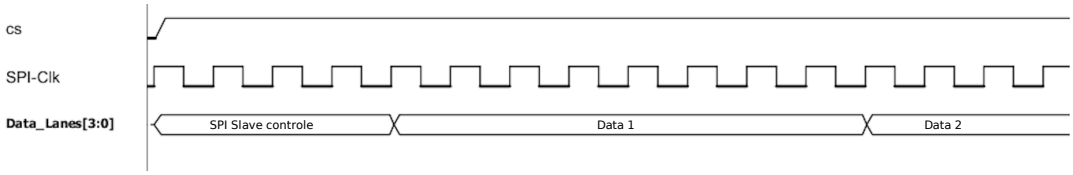

Abbildung 5.3.: Quad-SPI Signal Diagramm

5.3. SPI Steuerung

Der SPI-Master wird über ein externes 32 Bit Command-Register gesteuert, welches folgendermaßen aufgebaut ist.

SPI Slave control			SPI Master control			
31-24	23-17	16	15-8	7-2	1	0
free	addr	read/write	free	word count	read/write	aktiv

Tabelle 5.1.: SPI Command-Register Pinout

Für die Steuerung des Masters werden nur die unteren 16 Bit verwendet. Sobald Aktiv auf '1' gehoben wird, wird das im Register liegende Command ausgeführt. Wenn

ein 4 Bit Wort bei 3 Signalen immer noch 2 Zyklen.

[5] In der FEB-Version an der während für die Arbeit gearbeitet wurde, ist die ursprünglich zur Übertragung der Clock gedachte Verbindung durch einen Fehler in der Bezeichnung nicht angeschlossen. Glücklicherweise wurde eine zusätzliche Daten-Verbindung verbaut, welche eigentlich nicht nötig gewesen wäre, so erwies sich die 'überflüssige' Verbindung als Glücksfall.

Aktiv zurück auf '0' gezogen wird, fällt der Master in seinen Grundzustand zurück. Das Read/Write Bit gibt an, ob Daten übertragen oder angefordert werden. Hierbei werden im Falle einer '1' Daten vom Max10 angefordert und bei '0' entsprechend gesendet. Der Wordcount[6] gibt an, wie viele 32 Bit Wörter gesendet oder empfangen werden sollen, wobei der Count bei 0 startet. Die Bits 8-15 sind frei, wenn mehr als 256 Byte pro Befehl übertragen werden sollen, können diese verwenden werden, um den Wordcount zu erhöhen. Die restlichen 16 Bit werden zu Beginn jeder Übertragung an den SPI Slave gesendet und dort wie folgt verarbeitet. Die Besetzung des Read Write ist identisch zum Master. Die 7 Bit Adresse ist die gewünschte Adresse, die Adressbelegung wird später noch aufgelöst. Die Bits 31-24 sind auch hier frei und könnten mit zusätzlichen Befehlen belegt werden.
Die Befehle werden auf beiden Seiten durch eine State-Machine ausgeführt(Siehe Abbildung 5.4), welche konträr zueinander laufen, sodass Slave und Master die Daten-Lanes nicht gegeneinander treiben.

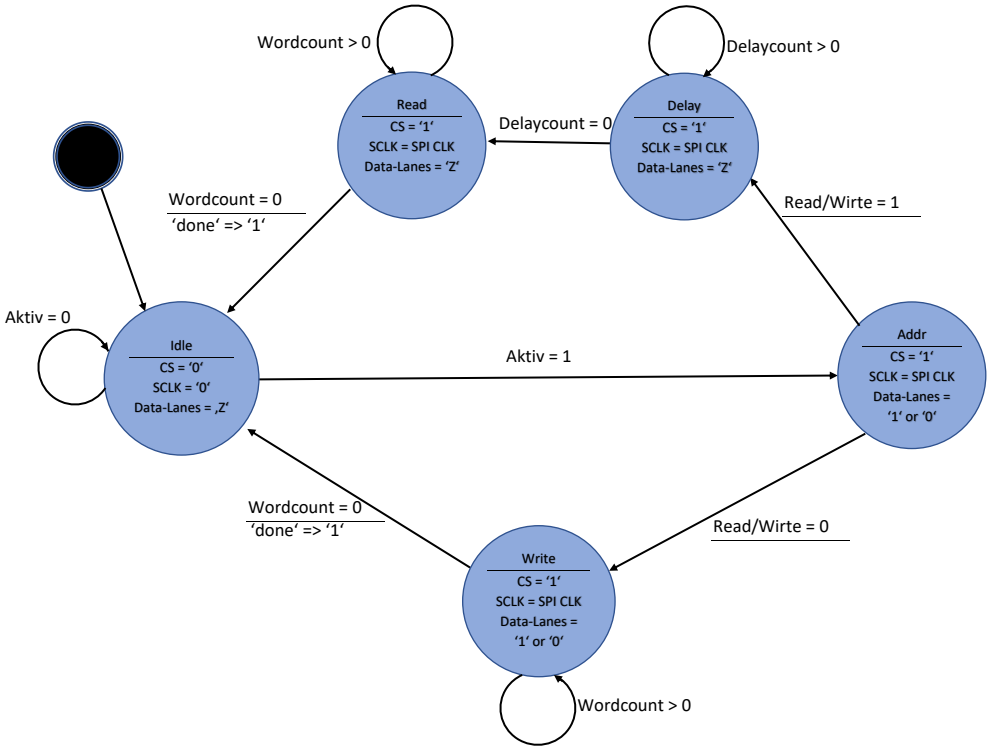

Abbildung 5.4.: State Machine vom SPI-Master (SPI-Slave: siehe Abbildung A.4).

Die Übertragung startet mit dem Setzen des Chip Select und dem Senden der Clock. Als erstes werden die Adressdaten und der Read/Write Bit übertragen. Wenn ein Write Befehl vorliegt, werden nun die Datenpakete gesendet. Zur Datenübertragung

[6] Wortzähler.

vom Arria5 hin zum Max10 speichert der SPI-Master alle 8 Zyklen die anliegenden Daten in Schieberegister, welche die Daten auf die Datenverbindungen übertragen. Auf der Seite des Slaves werden diese wieder durch Schieberegister zusammengefügt und mit einem Write Bit ausgegeben. Ist der Word-Counter bei 0 angekommen, sendet der SPI Master ein 'done' Signal und geht zurück in den Idle-State. Da der SPI Master durch ein externes Kontrollregister seine Instruktion erhält, muss dieses nach dem Erhalten von 'done' 'aktiv' auf '0' ziehen, da sonst der SPI Master nach einem Zyklus in Idle den letzten Befehl wiederholt. Der SPI-Slave führt seine Tätigkeit solange aus, bis der CS wieder auf low fällt. Um fortlaufend Register zu lesen oder zu schreiben, geben beide SPI-Komponenten einen Address_offset aus, welcher nach jedem Schreiben oder Lesen erhöht wird. Der Offset startet immer bei Null.

5.4. Anschluss an Max10

Zunächst wird die Adresse in einem De-Multiplexer aufgelöst und die Verbindung zu dem gewünschten Register eingestellt. Unabhängig von einem R/W Befehl sendet das Register seine gespeicherten Daten an den SPI-Slave. Um Synchronisationsprobleme zu vermeiden wird nach jedem Aufnehmen der Daten vom SPI-Slave der Addr_offset erhöht, wodurch schon die nächsten Daten angefordert werden. Beim Schreiben auf dem Max10 werden die Daten zusammen mit dem Setzen des Write Bits ausgegeben. Um hier Fehler zu vermeiden, wird das Write Bit nach einem Clock Zyklus wieder zurückgenommen. Im Zyklus darauf wird die Adresse erhöht, sodass, ähnlich zum Graycode, nur maximal eine Komponente gleichzeitig verändert wird.

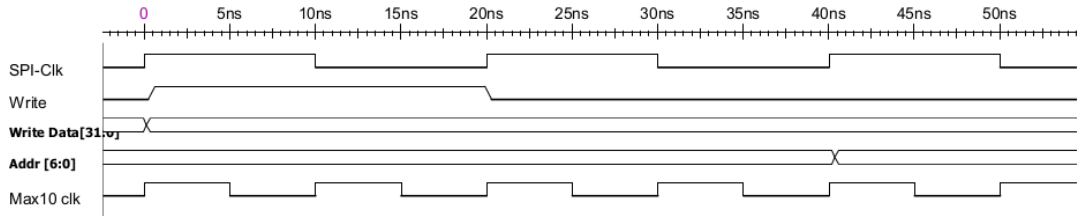

Abbildung 5.5.: Signaldiagramm zur Ausgabe von Daten aus dem SPI-Slaves.

Zu beachten ist, dass dies nur durch die doppelt so schnelle Max10 clock ermöglicht wird. So kann zwar immer noch während eines Bitwechsels ein clock Event[7] stattfinden, aber es ist garantiert, dass mindestens ein Event während eines stabilen Zustandes die Daten übernimmt. Wenn die an den SPI-Slave angeschlossene Clock Domain diese Voraussetzung nicht mehr erfüllt oder gar langsamer ist, kann das Problem mit einem asynchronen FIFO behoben werden.

[7] Mit Clock Event ist eine steigende Clock Flanke gemeint, bei der die Daten von einem Register übernommen werden.

5.5. Anschluss an den Arria5

Die Datenverarbeitung auf der SPI-Master-Seite ist zum größten Teil identisch. Der entstandene Phasenunterschied zwischen der ursprünglichen gesendeten Clock und der sendenden Clock des SPI-Slaves kann zu metastabilen Zuständen führen, da die Daten verzögert zum Clock Event beim Master ankommen. Um dies zu verhindern, könnten die ankommenden Signale mit einer 100MHz Clock abgetastet werden und somit die Daten zwischen den SPI_clk Events übernommen werden[8]. Die weitere Verarbeitung und Zusammensetzung ist durch die 50 MHz Clock gegeben.
Für den Anschluss an das Slowcontrol Register und dem damit verbundenen Clock Übergang zur 156 MHz Clock ist die Verwendung von FIFOs geplant.

5.6. Möglichkeiten zur Verwendung

Das SPI soll hauptsächlich die ADC-Daten für die Slowcontrol bereitstellen, aber auch zum Beschreiben des Flash, oder auch indirekt zur Programmierung des Max10 verwendet werden. Die einfachste Möglichkeit, Daten zu senden oder abzufragen, ist es, die Kontrolle über das Slowcontrol-Register zu leiten. Hierbei werden die zu sendenden Daten auf einem FIFO am Slowcontrol Register geschrieben. Das Slowcontrol-Register übernimmt dabei die Aufgabe als externes Command-Register des SPIs. Mit dem Setzen des Schreib-Commands arbeitet der SPI den FIFO dann ab. Bei Lesen wird der Befehl gesetzt, welcher wieder vom SPI durchgeführt wird und die gewünschten Daten auf einen FIFO für die Slowcontrol bereitstellt.
Alternativ können häufige Befehle wie das Abfragen der ADC-Daten von einem Mikrocontroller periodisch erledigt werden. Hierbei kopiert der SPI die ADC-Daten alle X-Zyklen vom Max10 auf den Arria5, von dort können diese bei einer Anfrage direkt übertragen werden. Für das Lesen oder schreiben über die Slowcontrol wird weiterhin ein FIFO verwendet, wobei Befehle von der Slowcontrol an den Controller geschickt und ggf. eingereiht werden müssen. Befehle vom Midas sollten dabei gegenüber dem vom ADC lesen bevorzugt werden.
In der folgenden Tabelle sind die geplanten Befehle für das SPI Interface aufgelistet, wobei die Adresse für den SPI-Slave gilt und die Datenmenge(Payload) als Anzahl an 32Bit Datenpaketen an den Word-Count des SPI-Masters übergeben werden muss. So braucht man für das Schreiben des FIFOs 256Byte also 64 32Bit Datenpakete.

Address	Type	Payload Length	Comment
`0x0`	R	32 Bit	Version git hash
`0x1`	W	8 Bit	Write enable register (write `0xA3 to enable WWE registers`)
`0x2`	R	32 Bit	Status register

[8] Zwar wurden die Signale im späteren Test mit derselben Clock abgetastet. Probleme traten dadurch aber nicht auf.

Address	Type	Payload Length	Comment
0x3	WWE	32 Bit	Reset register
		Quad SPI flash interface	
0x10	R	32 Bit	Programming status register
0x11	R	32 Bit	Programming counter register
0x12	WWE	8 Bit	Programming control register
0x13	W	32 Bit	Flash command address register
0x14	W	256 Byte	Flash write FIFO
0x15	R	256 Byte	Flash read FIFO
		ADC Interface	
0x20	R	40 Byte	Read ADC Data
0x21	WR	32 Bit	Read/Write ADC control register

An sich ist es auch möglich, zusätzliche Register an das SPI anzuschließen. Ein direkter Anschluss an den RAM[9] vom Nios Prozessor ist auch möglich.

[9] Random-Access Memory

6. FEB Überwachung in Helium Umgebung

6.1. Test in einer Helium Umgebung

Das FEB wird später in einer Heliumumgebung operieren. Auch wenn Helium als Edelgas unproblematisch erscheint, hat es bei erhöhter Konzentration Auswirkung auf elektrische Bauteile. So stört es die Clock Erzeugung von MEMS-Oszillatoren [23]. Zwar wurden diese Oszillatoren nicht explizit auf dem Board angebracht, aber bei fremden Bauteilen, wie z. B. den Fireflys sind die genauen Spezifikationen häufig nicht verfügbar[1]. Um spätere Probleme im Detektor vorzubeugen, werden diese Komponenten unter realistischen Bedingungen getestet. Dafür wird das FEB in einer Box eingeschlossen, welche langsam mit Helium durchlaufen wird. Bei den ersten beiden Messungen werden die Temperaturentwicklungen bei Verwendung eines der Fireflys, aufgenommen. Hierfür werden noch Sensoren verwendet, welche extern ausgelesen werden. In der dritten Messung wird das Überwachungssystem getestet, hierfür werden die auf dem Board befindlichen Sensoren verwendet,siehe Abbildung 4.1. Ihre digitalisierten Messdaten werden über das SPI-Interface an den Arria5 gesendet. Zum Vergleich wird Temperatur des Max10 mit einem externen Sensor aufgenommen. Später sollen die Daten auf dem Slowcontrol Register von Midas abgefragt werden, da hier nur das SPI getestet werden soll, reicht es aus, die Daten vom Nios abzufragen und über JTAG auszugeben.
Die ersten beiden Messungen sind kein Bestandteil dieser Arbeit und werden daher nicht näher erläutert, das Protokoll A.1 ist angehängt. Zusammengefasst lässt sich aber sagen, dass auch beim längeren Testen kein Problem im Helium aufgetreten ist.

6.2. Überwachungssystem in Helium

Da bei der dritten Messung nur die ADC Daten abfragen werden sollen, ist der auszuführende Befehl für das SPI-Interface stets der Selbe. Daher setzt ein Prozess beim SPI-Master alle 2048 50Mhz Clock Zyklen das 'Aktiv' Bit des Control-Registers auf '1'. Der auf dem Arria5 befindlichen Nios fragt die Daten dann rund[2] alle 10 sec über einen Read Request vom Slowcontrol-Register ab und gibt sie über das NIOS-Terminal aus. Die zeitliche Einordnung durch Hinzufügen eines Zeitstempels erfolgt erst, wenn

[1] meistens aus Urheberrechtlichen Gründen.
[2] Der Nios2 Prozessor ist bei Zeitzählung nur bedingt genau, daher werden die Zeitstempel auch außerhalb hinzugefügt.

die Daten auf den verarbeitenden Computern eintreffen.
Fehler bei der Übertragung wären durch eine Bitverschiebung sichtbar. Diese tritt

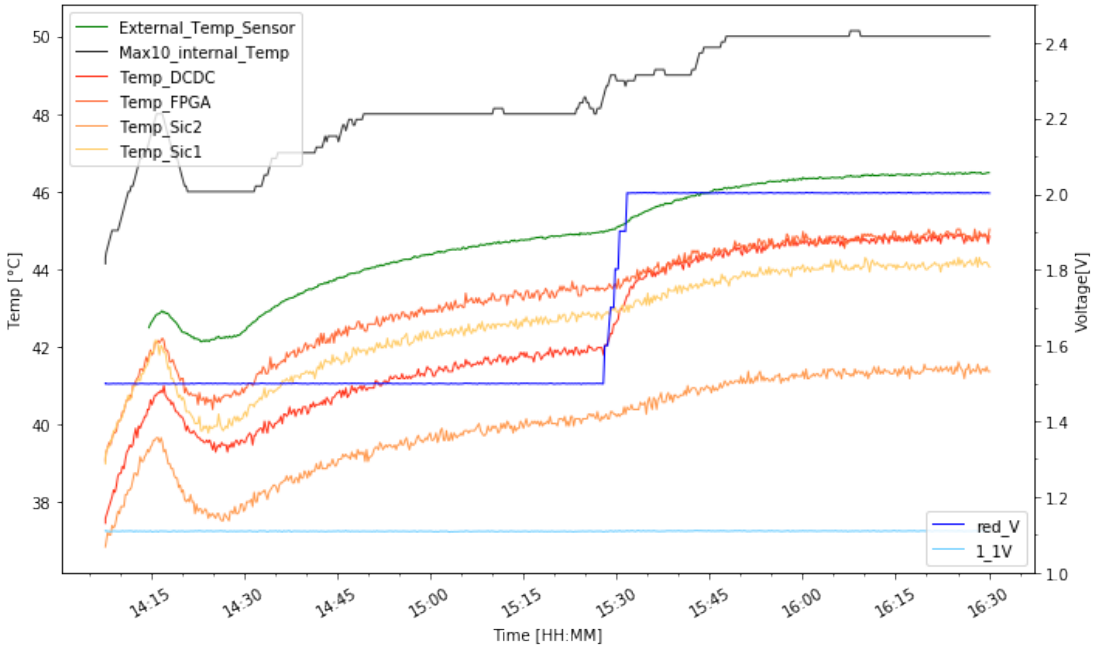

Abbildung 6.1.: Test des SPIs in Helium mit gemittelten Max10_internal_Temp Daten ohne VCC 3V3 , 2V5 , 1V8. Rohdaten Abbildung A.1

auf, wenn das SPI die Daten einen Zyklus zu früh oder zu spät aufnimmt, was zu einer Halbierung oder Verdopplung der Sensorwerte führen würde[3]. Das Auftreten eines metastabilen Zustandes oder ein daraus folgender fehlerhafter Bit hätte vereinzelt zu falschen Messergebnissen geführt. Da keiner dieser Fehler aufgetreten ist und die Daten alle in einem realistischen Bereich liegen, hat das SPI die Sensordaten korrekt erhalten und übertragen.
Zu Beginn wir noch ohne Helium gemessen. Bei Minute 14:15 wurde die Box mit Helium befüllt, zu erkennen an dem kurzzeitigen Sinken der Temperaturen, hervorgerufen durch die nun herrschende Heliumzirkulation. Nach Einstellen des Temperaturgleichgewichtes, wird die Betriebsspannung stufenweise von 15 auf 20 V erhöht. Der Strombedarf steigt von 9.4 W auf 10 Watt. Da dies zusätzliche Heizleitung bedeutet, steigen die Temperaturen wieder. Der sprunghafte Anstieg am DC-DC-Temperatursensor lässt darauf schließen, dass ein Großteil der zusätzliche Leistung im Spannungsumwandler benötigt wird.
Über die Kalibrierung der Sensoren bzw. des ADC bietet diese Messung nur einen groben Aufschluss, da einerseits die Auflösung des internen Max10 Sensors nur 1K beträgt, was eine Feinkalibrierung erschwert. Und andererseits der externe Tempera-

[3] rechts bzw. linksshift von Bits.

tursensor, welcher nicht direkt auf dem Max10 aufgebracht wurde, sondern auf einer ein mm dicken Schicht eines wärmeleitenden Materials und somit ein unbekannten Temperaturabfall im Vergleich zum internen Sensor aufweißt. Zwar hat man keine Auskunft über die Temperatur auf dem Arria5, es passt aber ins Bild, dass der externe Sensor auf dem Max10 eine höhere Temperatur aufweist als der TEMP-FPGA, welcher für die Temperaturüberwachung des Arria5 gedacht ist, aber keine direkte Verbindung zu ihm besitzt.
Um einen möglichen Offset genau zu messen, sollten externe Sensoren möglichst nah an den Onboard Sensoren angebracht werden. Bis dahin sollte ein Fehler von $\Delta T \pm .5K$ in Betracht gezogen werden. Über einen möglichen Offset beim ADC lässt sich auch nur eine grobe Einschätzung mit den Daten machen. Zwar stimmen die Daten für die reduzierte Spannung, welche durch einen 1/10 Spannungsteiler erzeugt wird, mit der Eingabe am Spannungsgenerator überein, die Anzeige des Generators weist aber selbst eine Ungenauigkeit auf. Diese Messung ist nicht dafür geeignet um einen genauen Aufschluss über die Genauigkeit des Überwachungssystems zu liefern. Sie reicht für ein Überwachungssystem aber aus, obwohl genauere Messungen angebracht wären. Zudem hat sie gezeigt, dass die Sensordaten aufgenommen, übertragen und ausgegeben werden können, was ein funktionierendes SPI voraussetzt.

7. Zusammenfassung und Ausblick

Das Mu3e Experiment versucht den, im Standardmodell stark unterdrückten, Myonzerfall in drei Elektronen zu finden und somit Aufschlüsse über Physik jenseits des Standardmodells zu liefern. Um den Zerfall zu beobachten, wird die Auslese des Experiments nicht getriggert, wodurch bis zu 100 $\frac{GBit}{s}$ Daten aus dem Experiment heraus transportiert werden müssen. Um diese Datenmengen zu handhaben zu können, wird die erste Stufe der Datenauswertung, bestehend aus Field-Programmable-Gate-Arrays (FPGAs) und ihrem Trägerboards, im Detektor verbaut. Dies erschwert aber die Überwachung und mögliche Anpassungen am Code. Hierfür befindet sich neben dem Detektordaten verarbeitenden Arria5 FPGA ein weiter FPGA, der Max10, auf dem Bord.
Zur Temperaturüberwachung des Boards befinden sich 5 Temperatursensoren über das Board verteilt. Zur Überwachung der Spannungen werden 5 Sollspannungen aufgenommen. Die Verarbeitung und Digitalisierung der Sensordaten findet auf dem Max10 statt. Zur Übertragung der Daten aus dem Detektor müssen diese, zusammen mit anderen Daten, an das Slowcontrol-Register auf den Arria5 gebracht werden. Um die Daten vom Max10 hin zum Arria5 zu transportieren wurde im Zuge dieser Arbeit ein quad-SPI implementiert, welches zur Übertragung von 32Bit Datenblöcken optimiert wurde. In einem Helium Test wurde das SPI für die Übertragung und Bereitstellung der Daten an dem Slowcontrol-Register erfolgreich getestet. Zwar wurde der Befehl an das SPI noch periodisch von einem Mikrocontroller gegeben und die Daten nicht wie später geplant über die Glasfaser abgefragt, sondern über einem auf dem Arria5 befindlichen Prozessor. Der erste Test hat aber gezeigt, dass das SPI in der Lage ist, den gewünschten Befehl auszuführen und die Daten vollständig zu liefern. Auf Seite des Max10 können mit dem SPI bei voller Adressbelegung 128 (7Bit) Adressen angesprochen werden, der Adressraum könnte hier mit einer Hardwareanpassung auf 15Bit erhöht werden. Zurzeit ist nur das Abfragen der ADC Daten implementiert, der Zugriff auf Control und Command Register ist schon geplant, sowie die Möglichkeit, die Programmdaten für den Arria5 auf dem Externen Flash-Speicher zu bearbeiten. Dies würde das Programmieren des Arria5 auch innerhalb des Detektors ermöglichen. In der jetzigen Implementierung können pro Befehl 64(6Bit) mal 32Bit Datenblöcke übertragen werden, bei Bedarf könnte auch hier die Anzahl mit einer Hardwareanpassung auf 14Bit mal 32Bit erhöht werden und somit bis zu 65.5 KByte übertragen werden. Zu beachten ist aber, dass die Daten dem SPI rechtzeitig bereitgestellt werden müssen und bei größeren Daten auch größere Zwischenspeicher benötigt werden.

A. Anhang

A.1. Tabellen und Abbildungen

Date	Time	Link 1-3	T Arria (C)	T Firefly 1 (C)	T Firefly 2 (C)	T LM35 (C)	Event
23.10.2020	14:29	ok	39	42	45		
	14:38		41	43	46	32	
	14:42	ok					start helium flow @ 1l/min
	16:28	ok	42	44	44	35	
	17:27	ok	42	33	33	35	still 0.634 A, turn off flow, power down FEB
26.10.2020	14:44	ok	27	30	30	24.3	start new measurement, set He flow 0.5 l/min, 0.624A
	17:47	ok	42	44	44	35	set flow <0.2 l/min for long he exposure
28.10.2020	10:45	ok	42	44	44	35	
					T Max 10 (C)	**T LM35 (C)**	
	14:48				48	42	
	15:31				49	45	go up to 20VDC, from 9.4 ->10W

Tabelle A.1.: Tabellarisches Protokoll des Helium Test. Durchgeführt und Angelegt von Dr. Frederik Wauters, AG Berger.

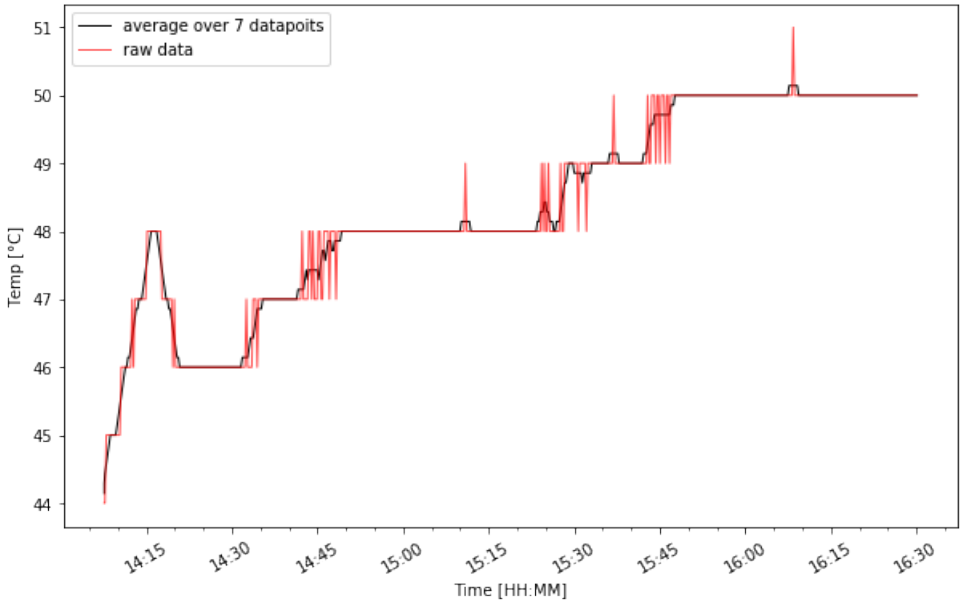

Abbildung A.1.: Roh-Daten vom internen Max10 Sensor.

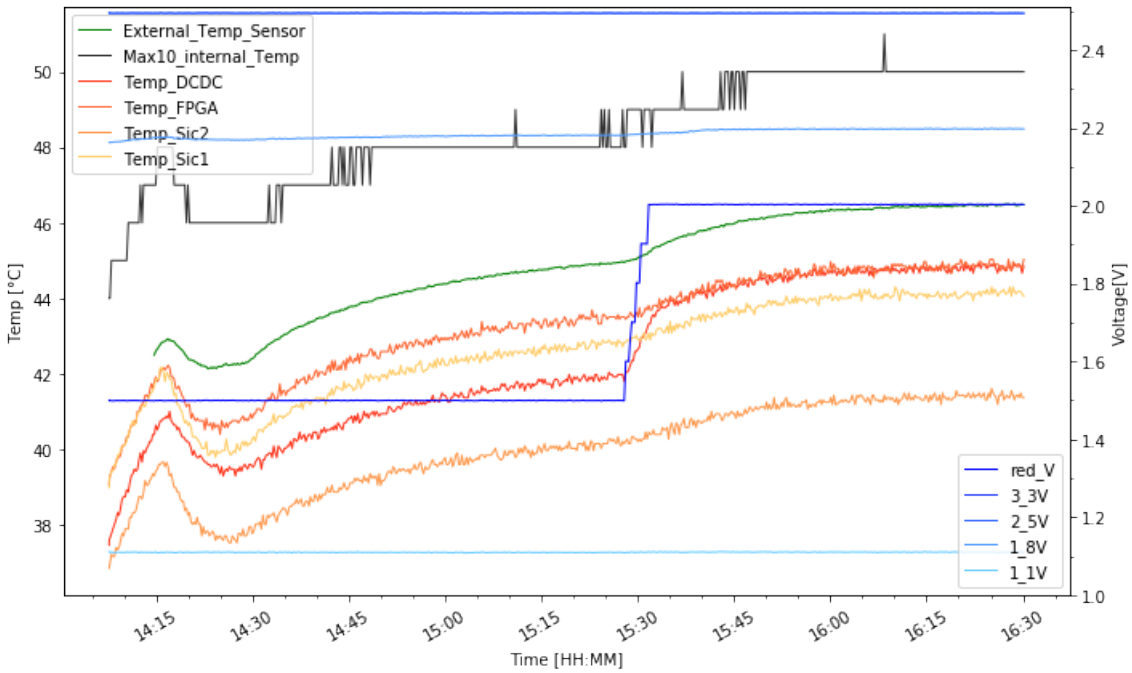

Abbildung A.2.: Gesamte Helium Messung, wie zu sehen sind die Ergebnisse für 3.3 und 2.5 V beim maximalen zu messenden Wert. Durch einen Anschlussfehler im FEB gibt 1_8V eine fehlerhafte Spannung wieder.

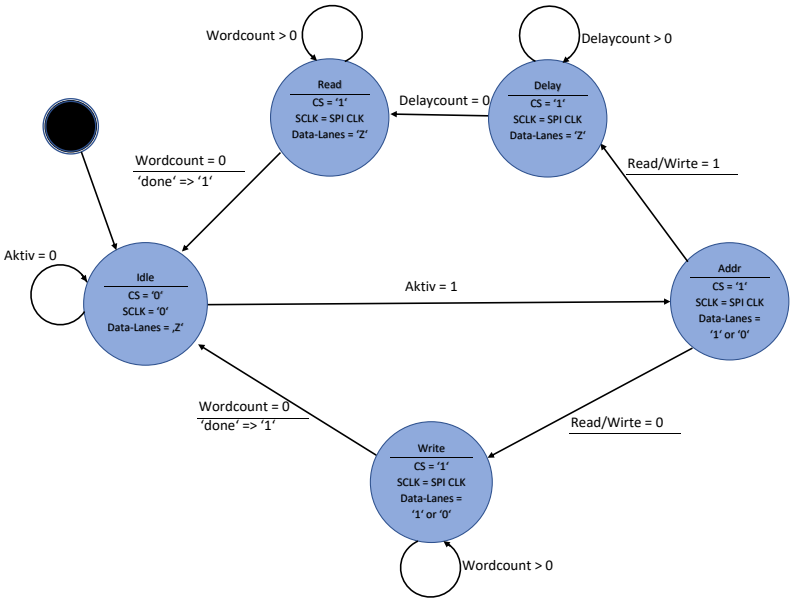

Abbildung A.3.: SPI-Master State Machine

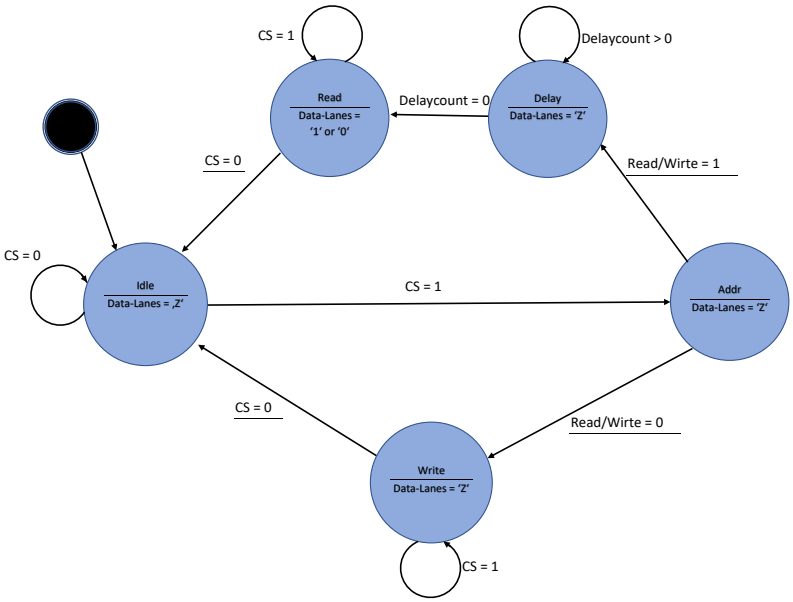

Abbildung A.4.: SPI-Slave State Machine

B. Literaturverzeichnis

Literaturverzeichnis

[1] Cush
https://commons.wikimedia.org/wiki/File:Standard_Model_of_Elementary_Particles.svg,
„Standard Model of Elementary Particles“, https://creativecommons.org/licenses/by/3.0/legalcode

[2] Georges Aad et al.
Observation of a new particle in the search for the Standard Model Higgs boson with the ATLAS detector at the LHC.
Phys. Lett., B716:1–29, 2012.

[3] Y. Fukuda et al. (Super-Kamiokande Collaboration)
Evidence for Oscillation of Atmospheric Neutrinos
Phys. Rev. Lett. 81, 1562 – Published 24 August 1998

[4] M. Aker et al. (KATRIN Collaboration)
Improved Upper Limit on the Neutrino Mass from a Direct Kinematic Method by KATRIN
Phys. Rev. Lett. 123, 221802 – Published 25 November 2019

[5] I. Peric,
“A novel monolithic pixelated particle detector implemented in high-voltage CMOS technology,”
Nucl. Instrum. Meth. A **582** (2007), 876-885
doi:10.1016/j.nima.2007.07.115

[6] Internal PSI WIKI for Mu3e

[7] Martin Müller.
A Control System for the Mu3eData Acquisition
Masterarbeit. Johannes Gutenberg-Universität Mainz

[8] Homepage PSI Mu3e
https://www.psi.ch/en/mu3e/introduction

[9] Angela Papa, University of Pisa/INFN and Paul Scherrer Institut Ghent, Belgium EPS 2019
Towards an High intensity Muon Beam (HiMB) at PSI
https://indico.cern.ch/event/577856/contributions/3420391/attachments/1879682/3097488/Papa_HiMB_EPS2019.pdf

Literaturverzeichnis

[10] Fpga architecture.
Altera Corp., San Jose, CA, USA, 2006.
White Paper WP-01003-1.0.

[11] Randy Johnson et al.
Jtag 101.
Intel Corp. White Paper, 2009.
www.intel.com/content/dam/www/public/us/en/documents/white-papers/jtag-101-ieee-1149x-paper.pdf

[12] Jennifer Stephenson,
Understanding Metastability in FPGAs
Altera Corp., San Jose, CA, USA, 2009.
White Paper WP-01082-1.2.

[13] Intel® Max® 10 FPGA Device Overview,
https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/hb/max-10/m10_overview.pdf

[14] Intel® MAX® 10 FPGA Device Datasheet,
https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/hb/max-10/m10_datasheet.pdf

[15] Intel® MAX® 10 Analog to DigitalConverter User Guide,
https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/hb/max-10/ug_m10_adc.pdf

[16] TMP235A2 datasheet: TMP23x Low-Power,High-Accuracy Analog Output Temperature Sensors,
$https://www.ti.com/lit/ds/symlink/tmp235.pdf?ts=1603738021857&ref_url=https%253A%252F%252Fwww.google.co.uk%252F$

[17] Nios® II Processor Reference Guide
https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/hb/nios2/n2cpu-nii5v1gen2.pdf

[18] Arria V Device Datasheet,
https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/hb/arria-v/av_51002.pdff

[19] Clifford E. Cummings, Sunburst Design, Inc.
Simulation and Synthesis Techniques for AsynchronousFIFO Design (2005)
http://www.sunburst-design.com/papers/CummingsSNUG2002SJ_FIFO1.pdf

[20] The ZipCPU by Gisselquist Technology Crossing clock domains with an Asynchronous FIFO Jul 6, 2018
https://zipcpu.com/blog/2018/07/06/afifo.html

[21] en:User:Cburnett
https://commons.wikimedia.org/wiki/File:SPI_single_slave.svg,

Literaturverzeichnis

„SPI single slave“, https://creativecommons.org/licenses/by-sa/3.0/legalcode

[22] en:User:Cburnett
https://commons.wikimedia.org/wiki/File:SPI_three_slaves.svg,
„SPI three slaves“, https://creativecommons.org/licenses/by-sa/3.0/legalcode

[23] Kyle Wiens
iPhones Are Allergic to Helium
https://de.ifixit.com/News/11986/iphones-are-allergic-to-helium

C. Danksagung

Zunächst möchte ich mich bei Prof. Niklaus Berger für die Möglichkeit einer Arbeit an diesem Experiment und seiner fortwährenden Unterstützung während meiner Arbeit bedanken.
Auch ohne die Unterstützung und Hilfe von Martin Müller, Marius Köppel, Frederik Wauters und Alexandr Kozlinskiy wäre meine Arbeit nicht möglich gewesen. Vielen Dank an euch und den anderen Mitgliedern der AG Berger.
Danke auch an alle, welche meine Arbeit Korrektur gelesen haben: Alexander Kerth, Fabian Keßler, und Prof. Niklaus Berger.
Hier möchte ich auch Prof. Büscher dafür danken, dass er sich bereit erklärt hat, mein Zweitgutachter zu sein.
Abschließend möchte ich auch noch meiner Familie für ihre Unterstützung danken.