HAW HAMBURG

Bachelorarbeit

Sebastian Paulsen

Entwicklung und Evaluierung kompakter Hardware zur Realisierung künstlicher neuronaler Netze im Bereich autonomer Fahrzeuge

Fakultät Technik und Informatik
Department Informatik

Faculty of Computer Science and Engineering
Department Computer Science

Sebastian Paulsen

Entwicklung und Evaluierung kompakter Hardware zur Realisierung künstlicher neuronaler Netze im Bereich autonomer Fahrzeuge

Bachelorarbeit eingereicht im Rahmen der Bachelorprüfung
im Studiengang Bachelor of Science Technische Informatik
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer: Prof. Dr. Tim Tiedemann
Zweitgutachter: Prof. Dr. Michael Schäfers

Eingereicht am: 28. September 2019

Sebastian Paulsen

Thema der Arbeit

Entwicklung und Evaluierung kompakter Hardware zur Realisierung künstlicher neuronaler Netze im Bereich autonomer Fahrzeuge

Stichworte

Hardware, Neuronale Netze, autonome Fahrzeuge, miniatur, FPGA, SoC, Kamera, MIPI, Leiterplatten, JTAG, 3D-Modell

Kurzzusammenfassung

Die vorliegende Arbeit beschäftigt sich mit der Entwicklung von kompakter Hardware für autonome Miniaturfahrzeuge. Hierfür wird ein Testfahrzeug inklusive Elektronik im Maßstab H0 (1:87) entwickelt und in Betrieb genommen. Abschließend werden die Ergebnisse bezüglich der Anforderungen an die Hardware diskutiert, ein Fazit gezogen und ein Ausblick gegeben.

Sebastian Paulsen

Title of Thesis

Development and evaluation of compact hardware for realizing neuronal networks in autonomous cars

Keywords

Hardware, Neuronal Network, autonomous Cars, miniature, FPGA, SoC, Camera, MIPI, PCB, JTAG, 3D model

Abstract

This thesis focuses on the development of compact hardware for autonomous miniature vehicles. For this purpose, a test vehicle including electronics in scale H0 (1:87) is developed and put into operation. Finally, the results regarding the hardware requirements are discussed, a conclusion is drawn and an outlook is given.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung

Alle zwei Jahre verdoppelt sich die Menge an Transistoren, die auf eine bestimmte Fläche passen [13]. Folglich wird der benötigte Platz für die gleiche Rechenleistung immer kompakter, womit Entwicklungen in der Industrie auch immer kompakter und leistungsstärker werden. Was vor einigen Jahren einen ganzen Raum ausgefüllt hat, lässt sich heutzutage in der Hosentasche unterbringen, was zum Beispiel an der Entwicklung des Smartphone ersichtlich wird. Diese Entwicklung lässt sich auch auf die Rechenhardware in der Automobilindustrie übertragen.

Heutige Fahrzeuge fahren mit vielen Assistenzsystemen, die dem Fahrer helfen. Ein Beispiel hierfür ist der Distanzsensor, welcher durch die Messung der Distanz zum vorausfahrenden Fahrzeug eventuelle Aufpralle ermittelt und den Fahrer durch visuelle und akustische Signale warnen kann oder eine Bremsung einleitet. Die Assistenzsysteme bilden Bestandteile autonomes Fahrens.

1.1 Motivation

Katastrophen wie die Kernschmelze in einem Kernkraftwerk in Fukushima [Vgl. 4] machen exemplarisch deutlich, dass es Probleme gibt, für dessen Lösung nur besonders kleine und spezialisierte Roboter benutzt werden können. Dies gilt für die Orte, an denen wegen Unzugänglichkeit und Abschirmung weder Menschen noch ferngesteuerte Roboter Zugriff haben. Für dieses Szenario benötigt man autonome Fahrzeuge, die sich ohne menschliche Interaktion durch Geröll bewegen können und verletzte Personen finden oder notwendige Arbeiten ausführen.MDiese Arbeit soll einen Beitrag dazu leisten, sich mit autonomen Miniaturfahrzeugen außeinanderzusetzen und deren Entwicklungswege auszuloten.

1.2 Zielsetzung

Das Ziel dieser Arbeit ist es, zu evaluieren, welche elektronischen Bauteile für ein autonomes Fahrzeug benötigt werden und wie sich die Auswahl der Bauteile auf eine minimale Größe des Fahrzeugs auswirkt. Deshalb geht diese Arbeit der folgenden Fragestellung nach, inwiefern die Entwicklung einer kompakten Hardware für autonome Fahrzeuge realisiert werden kann.

Dabei werden die Bauteile auf eine Platine gebracht, welche in das Fahrzeug einbaut werden soll. Die Platine soll die Daten der Sensoren in Steuerungssignale für die Aktoren des Fahrzeuges umwandeln. Auf der Platine soll es neben den Sensoren und Aktoren noch weitere Interfaces geben, die für die Initialisierung einer Platine benutzt werden können.

1.3 Vorgehen

Um das oben genannte Ziel zu verfolgen, ergibt sich der erste Schritt, nach Informationen über bereits vorhandene Arbeiten und Ansätze bezüglich des autonomen Fahrens in Miniaturfahrzeugen zu recherchieren. Dafür wird der aktuelle Forschungsstand analysiert und beschrieben.

In einem nächsten Schritt werden die für die darauf folgenden Kapitel benötigten theoretischen Grundlagen dargestellt und erläutert. Die theoretischen Grundlagen sollen einen Überblick über die beschriebene Thematik geben und für diese Arbeit relevanten Unterpunkte eine Basis bieten.

Darauf folgend wird die Konzeption und die Umsetzung der kompakten Hardware und seiner elektronischen Bauteile beschrieben. Der Fokus dabei wird auf die Auswahl der elektrischen Bauteile für diverse Interfaces gelegt. Diese definieren die Optionen, welche Sensoren, Aktoren und weitere Peripherie angeschlossen werden können.

Nachdem die Planung abgeschlossen ist, wird die daraus resultierende Hardware in Betrieb genommen.

Abschließend werden die Ergebnisse diskutiert, sowie ein Fazit gezogen. In Hinsicht auf die Ergebnisse der Umsetzung werden eine Diskussion und ein Fazit die Arbeit vollenden.

2 Theoretische Grundlagen

Die folgenden theoretischen Grundlagen sollen den theoretischen Rahmen für die Entwicklung kompakter Hardware bieten, die im Laufe dieser Arbeit verwendet werden.

Folgend werden datenverarbeitende elektrische Schaltungen beschrieben.

2.1 Prozessor

Prozessoren sind elektronische Schaltungen, die Daten verarbeiten. Es gibt unterschiedliche Prozessortypen, die je nach Einsatzgebiet anders spezialisiert sind. Dabei besitzen sie zum Beispiel unterschiedliche Registerbreiten, Befehlssätze oder Architekturen.

IC Zum besseren Verständnis der folgenden datenverarbeitenden Schaltung wird der Begriff Integrated Circuit (IC) an dieser Stelle erläutert. Unter einem IC versteht man eine elektronische Funktionseinheit, die sich durch eine auf einem gemeinsamen Halbleitersubstrat(Chip) realisierte Vielzahl elektrisch und mechanisch untrennbar miteinander verbundener elektronischer Funktionselemente(Transistor- und Diodenfunktionen, Widerstände, Kondensatoren etc.) mit Abmessungen im Mikrometer- und Submikrometer-Bereich kennzeichnet[Vgl. S.394, 2].

CPU Der Kern eines Computers ist die zentrale Verarbeitungseinheit (Central Processing Unit, CPU). Diese besteht im wesentlichen aus den Komponenten eines Steuerwerks, Rechenwerks, mehreren Registern und eines Verbindungssystems zur Ankopplung von Speicher- und Peripheriekomponenten. [S.18, 2] Der Begriff Mikroprozessor beschreibt die auf einem Chip realisierte CPU eines Computersystems.[S.19, 2] Damit sind die meisten Prozessoren, die in Computern eingesetzt werden, Mikroprozessoren.

GPU Graphical Processing Unit (GPU) sind spezialisierte Prozessoren die hauptsächlich für die rechenintensiven Berechnung von 2D- und 3D-Grafiken eingesetzt werden.[15]

Mikrocontroller Ein Mikrocontroller ist ein vollständiges Mikrocomputersystem auf einem Chip. Mikrocontrollerkern (Core), Speicher, Peripheriekomponenten und Interrupt-System sind gemeinsam auf einem Chip integriert und über einen bzw. mehrere Busse miteinander verbunden. Ein Mikrocontrollerkern ist im Sinne eines modularen MC-Konzepts die On-Chip-integrierte CPU. Dieser beinhaltete im Wesentlichen ein komplexes Steuerwerk, mehrere Register, eine Arithmetic Logic Unit (ALU) und eine Bussteuereinheit. [S.242-243, 2]

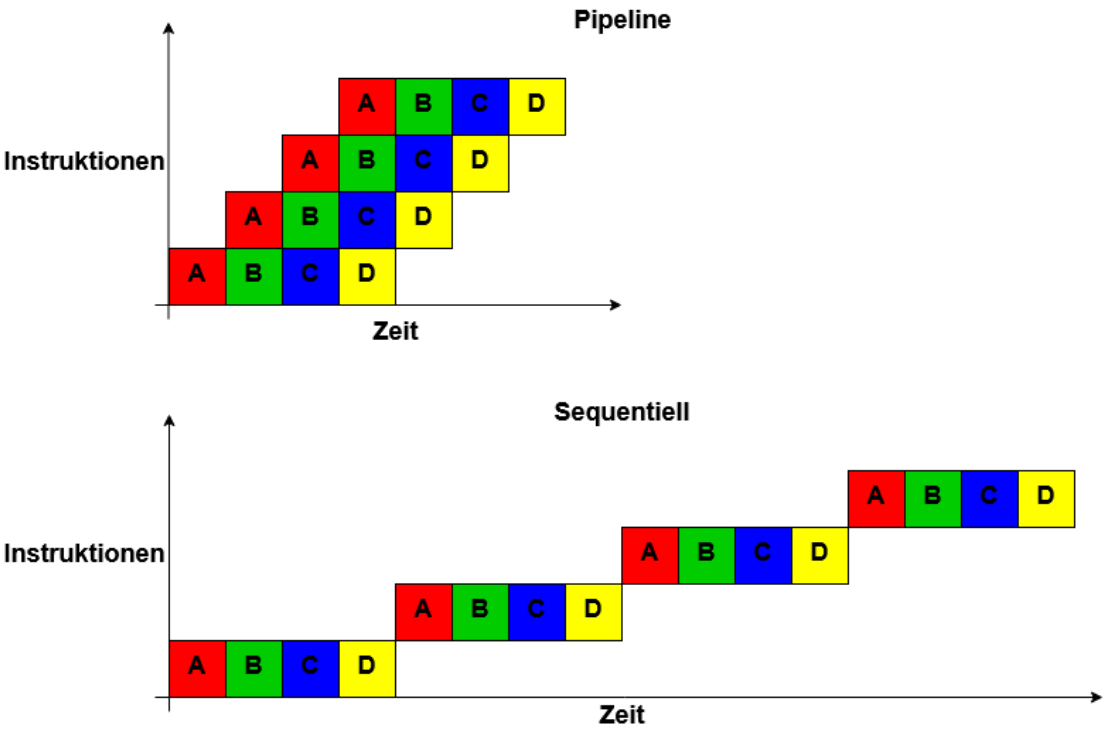
Prozessorarchitekturen CPUs, wie sie in Mikrocontrollern vorkommen, haben unterschiedliche Prozessarchitekturen implementiert. Prozessorarchitekturen zeichnen sich durch ihre unterschiedlichen Befehlssätze, Datenbreite und Registeranzahl aus. Diese machen sie je nach Konfiguration besonders schnell oder platzsparend. Beispiele für Prozessorarchitekturen sind RISC, x86 oder CISC.

FPGA FPGA (Field Programmable Gate Array) bestehen aus Transistoren, die in Gatter angeordnet sind. Diese Transistoren lassen sich bei FPGAs, außer bei Antifuse-based FPGAs nur einmalig, beliebig häufig programmieren und ermöglichen durch Verkettungen die Realisierung unterschiedlicher Aufgaben. Durch die Anordnung der Transistoren in Gattern in einem FPGA ist es möglich, parallel viele Daten zu verarbeiten. Diese Eigenschaft ist besonders in der Datenverarbeitung von Datenströmen nötig. Die FPGA-Programme werden am Computer erstellt und dann insofern optimiert, als sie in die Gatterform passen und falls nötig Bereiche der Schaltung teilen. In FPGA gespeicherte Programme sind insofern flüchtig, dass sobald an dem Chip keine Spannung mehr anliegt, sie das gespeicherte Programm verlieren. Da FPGAs frei programmierbar sind und keiner Prozessorarchitektur unterliegen, werden sie mit einer Hardware Description Language (HDL) programmiert. Diese wird dann von der menschenlesbaren Hochsprache in die Gattersynthese übersetzt. FPGAs sind durch ihren Aufbau sehr flexibel, aber sie zeichnen sich auch darin aus, dass sie anspruchsvoll zu programmieren sind, da Signallaufzeiten eingehalten werden müssen. [Vgl. S.388-391, 2]

SoC SoC (System on Chip) gibt es in vielen Varianten. Als die verbreiteste Variante kann die Verbindung von einem FPGA und einem Mikrocontroller auf einem Silizium Chip gelten. Der Vorteil dieses Chips ist es, dass die rechenleistungsintensiven Aufgaben im FPGA berechnet werden können, wohingegen im Mikrocontroller die Entscheidungslogik ausgeführt wird. Mit dieser Übersicht über die Hardware können vergleichende Bewertungen im Konzept getätigt werden.

2.2 Pipelining

Pipelining stammt ursprünglich aus der Zeit der Industrialisierung, in welcher in Fabriken Arbeiten am Fließband ausgeführt worden sind. Es beschreibt in der Informatik die Abfolge der Aufgabenverarbeitung eines Prozessors. Dabei werden im Prozessor Aufgaben nicht sequenziell abgearbeitet, sondern die Aufgaben laufen parallel ab. Dies hat zur Folge, dass die Menge an bearbeiteten Aufgaben ab einer bestimmten Zeit erhöht wird und dies ermöglicht neue Ergebnisse nach jedem neuen Takt. Die folgende Grafik2.1 stellt dar, wie mehr Aufgaben im Pipeline-Verfahren im Vergleich zum sequentiellen Verfahren ab einem gewissen Zeitpunkt verarbeitet werden. [Vgl. 19]



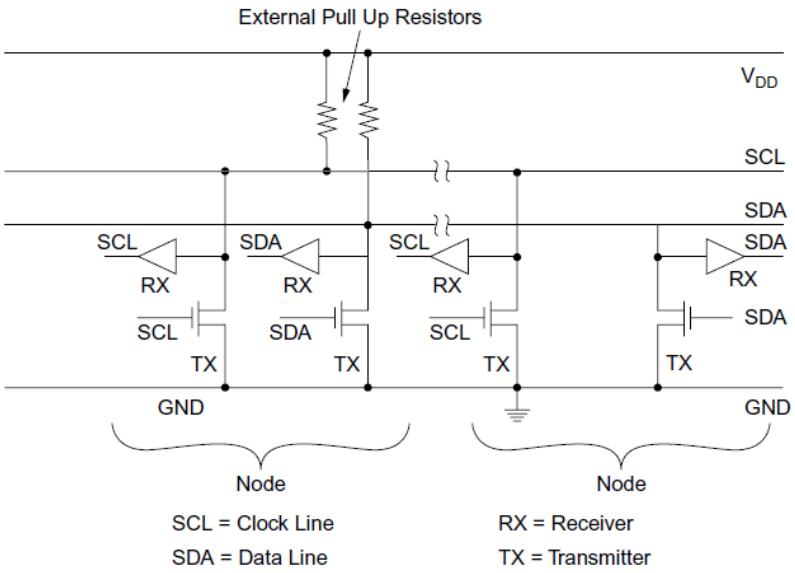
Quelle: Sebastian Paulsen

Abbildung 2.1: Vergleich Pipeline-Bearbeitung mit sequentieller Bearbeitung

2.3 Serielle Datenübertragung

Serielle Datenübertragung ist die verbreitetste Datenübertragung zwischen ICs. Die Datenübertragung findet meist zwischen zwei Platinen oder zwischen einer Platine und einem Sensor statt. Jede serielle Übertragungsart hat unterschiedliche Protokolle, Leitungsanzahlen, elektrische Charakteristiken und Nachrichtenformate. Das macht die Datenübertragungen besonders flexibel, daher können meist die gleichen Sensoren mit unterschiedlichener Transmitterhardware an unterschiedlichen Schnittstellen angeschlossen werden. [Vgl. S.109-112, 5]
Die folgende Aufzählung über I2C, SPI, und USB zeigt die in der Industrie häufig verwendeten Standards und stellt deren Hauptmerkmale dar:

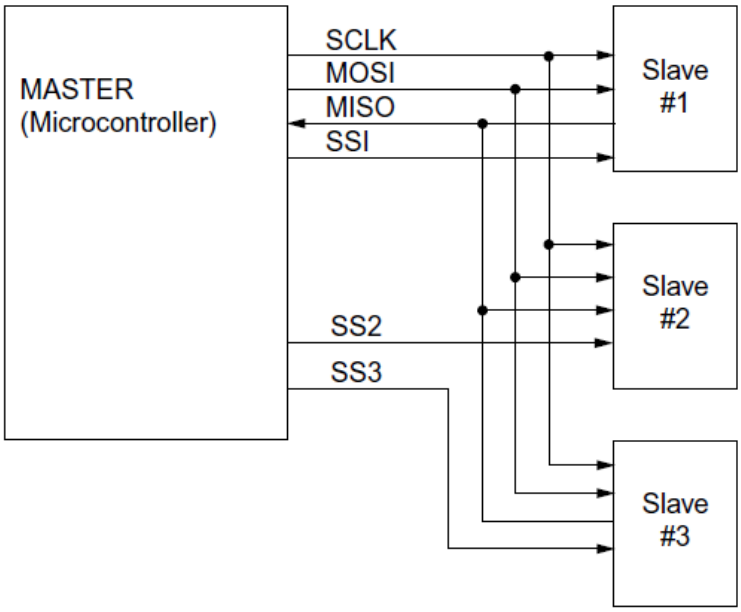
I2C Leitungsanzahl: SCL SDA VDD GND
Übertragungsgeschwindigkeit: 100 kb/s, 400 kb/s, 1 Mb/s, 3,4 Mb/s
Maximallänge: Hängt von der Datenraten ab. Maximallänge 10 m bei niedrigster Datenrate.[Vgl. S.65-70, 5]



Quelle: [Vgl. S.66, 5]

Abbildung 2.2: I^2C mit zwei Knoten

SPI Leitungsanzahl: MISO MOSI SCLK SSI
Übertragungsgeschwindigkeit: 20 Mb/s bis 100 Mb/s
Maximallänge: Bis zu 1 m aber typischerweise nur wenige Fuß[Vgl. S.143ff, 5]



Quelle: [Vgl. S.144, 5]

Abbildung 2.3: SPI

USB Leitungen: Vier Leitungen Ground, DC 4.75 bis 5,25 V, Data Positive(DP) und Data Minus(DM)

Version	Übertragungsgeschwindigkeit
1.1 1.5	12 Mbps
2.0	480 Mbps
3.1	5 oder 10 Gbps

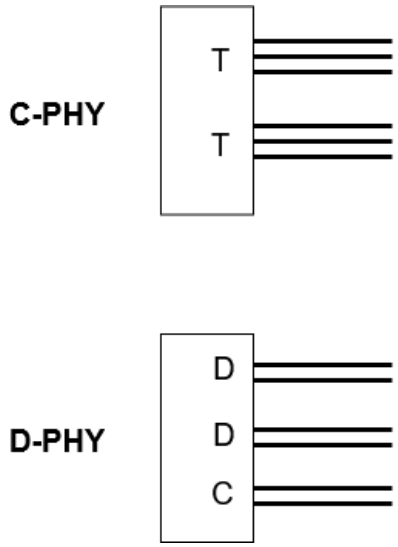
Tabelle 2.1: USB Versionsvergleich

Maximallänge: 3 m bei Version 3.1 sonst 5 m bei den anderen Versionen bis hin zu 100 m mit Repeatern.
[Vgl. S.147-151, 5]

MIPI Standard Der MIPI (Mobile Industry Processor Interface) Standard beschränkt sich nicht wie der Name sagt nur auf den mobilen Sektor sondern ist vielmehr in den letzten Jahren zu einem allgemein genutzten Standard geworden. Der Standard definiert zwei

Hardware Schnittstellen (D-PHY und C-PHY) 2.4, zu denen zwei Softwareschnittstellen(CSI-2 und DSI-2) kompatibel sind. CSI-2 steht für Camera Serial Interface und beschreibt die kommunikation zwischen einer Camera und einer Platine. DSI-2 steht für Display Serial Interface und beschreibt die Kommunikation zwischen einem Display und einer Platine.
[20] Leitungsanzahl: Variable Anzahl von differentiellen Leitungen
Übertragungsgeschwindigkeit: Maximal 5,7 Gbps
Maximallänge: Wenige cm auf der Platine oder kurze Leitungen.
[Vgl. S.193f, 5] Ein Vergleich des Aufbaus der physikalischen Schnittstellen C-PHY und D-PHY:

Vergleich der beiden elektrischen Standards C-PHY D-PHY
D-PHY besitzt ein differentielles Clock-Signal und ab einem Datensignal. C-PHY dagegen besitzt dreier Leitungen.



Quelle: Sebastian Paulsen

Abbildung 2.4: Vergleich MIPI C-PHY und D-PHY

Ethernet Anzahl der Leitungen: Je nach Standard
Übertragungsgeschwindigkeit: 10 Mb/s bis 100 Gb/s
Maximallänge: Bis zu 100 m ohne Repeater oder Ähnliches.
[Vgl. S.129ff, 5] Das Verständnis für serielle Kommunikation ist wichtig Voraussetzung für die im vierten Kapitel folgende Konzeption

2.4 Neuronale Netze

Um die Hardware Anforderungen für neuronale Netze bewerten zu können werden diese im Folgenden erläutert. Neuronale Netze sind eine Ansammlung von künstlichen Neuronen, die in drei unterschiedlichen Schichten unterteilt sind - die Eingangsschicht, die versteckte Schicht und die Ausgangsschicht. Künstliche Neuronen siehe Grafik 2.5 besitzen wie biologische Neuronen auch Eingänge (Dendriten) und Ausgänge (Axonen).

Diese Ein- und Ausgänge leiten elektrische Signale vom vorherigen bzw. zum nächsten Neuron. Dabei werden die Eingangssignale(x_i), welche mit unterschiedlichen variablen Gewichten(W_i) multipliziert werden, im Neuron aufaddiert und mit einem Schwellwert(θ) substrahiert. Daraus resultiert der sogenannte Aktivierungswert(α):

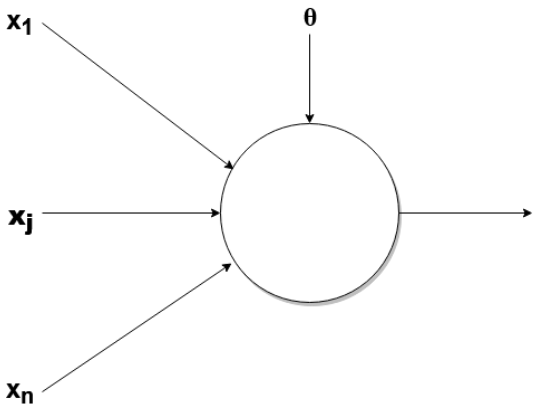
$$\alpha = W_1 * x_1 + W_2 * x_2 - \theta$$

Der Aktivierungswert wird in die Aktivierungsfunktion $f(\alpha)$ eingesetzt. Die Aktivierungsfunktion der Neuronen ist eine mathematische Funktion, wobei anfänglich eine binäre Funktionen verwendet wurde:

$$y = \quad sign\ \alpha$$

$$y = +1, \quad wenn\ \alpha \geq 0$$

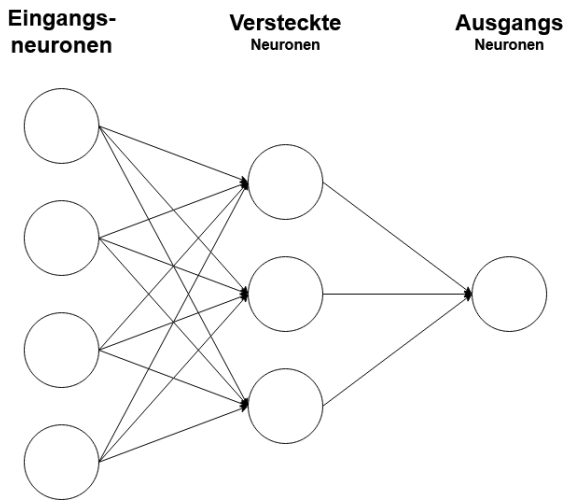
$$y = -1, \quad wenn\ \alpha < 0$$



Quelle: Sebastian Paulsen

Abbildung 2.5: Neuron

Die Eingangsschicht fungiert als eine der beiden Schnittstellen zum neuronalen Netz 2.6. Sie reagiert nicht auf synaptische Verbindungen, sondern auf Daten von außerhalb. Die zweite Schichtkategorie ist die optionale versteckte Schicht, die aus mehreren Unterschichten bestehen kann. Die dritte Schichtkategorie ist die Ausgangsschicht, welche als zweite Schnittstelle fungiert und Daten an ihre Knoten liefert. [Vgl. 25]



Quelle: Sebastian Paulsen

Abbildung 2.6: Übersicht Neuronales Netz

2.4.1 Training von Neuronalen Netzen

Durch die variablen Gewichte, können Neuronale Netze lernen. Es gibt unterschiedliche Ansätze wie das Netzt trainiert wird bzw. lernt.

Backpropagation Bei der Backpropagation werden dem Netz die Eingangswerte und die erwarteten Ausgangswerte als Tupel gegeben. Mit diesen Werten werden die Gewichte angepasst, damit der Eingangswert den erwarteten Ausgangswert ausgibt. Dies wird mit möglichst vielen Eingangs- und Ausgangswerten (Tupeln) gemacht. Durch Iteration lernt das Neuronale Netz. [Vgl. 25]

Genetisches Lernen Beim genetischen Lernen werden zu Beginn zufällige neuronale Netze generiert. Nach den Durchläufen der Testdaten in den Netzen werden die 50% der neuronalen Netze mit der höchsten Güte selektiert. Dann werden die restlichen 50% wieder aufgefüllt und dabei gibt es zwei Möglichkeiten, wie aufgefüllt werden kann. Entweder werden vorhandene kopiert oder mutiert, also leicht verändert, oder miteinander gekreuzt. Dieser Vorgang wird für mehrere Generationen wiederholt, bis sich höhere Güten der neuronalen Netze erzielen lassen. [Vgl. S.62-63, 16] Durch die Grundlagen der neuronalen Netze wird deutlich wie rechenintensiv die Berechnung von neuronalen Netzen ist.

2.5 Autonome Fahrzeuge

Weil die Evaluation einer Hardware für autonome Fahrzeug im Zentrum dieser Arbeit steht, wird im Folgenden eine Kategorisierung für autonome Fahrzeuge vorgestellt. Autonome Fahrzeuge, insbesondere Automobile, werden laut der Society of Automotive Engineers (SAE) in sechs unterschiedliche Level der Autonomie unterteilt. Kriterium für autonomes Fahren ist die Bewertung der Rolle des Fahrers und die Art des autonomen Systems. [Vgl. S.19-23, 17]

2.5.1 Die 6 Level des Autonomen Fahrens

Level 0 Das Auto hat keine Assistenzsysteme verbaut. Der Fahrer steuert das Fahrzeug.

Level 1 Das erste Level beschreibt das Vorhandensein von Assistenzsystemen, die dem Autofahrer in seltenen Situationen helfen und entweder bremsen bzw. beschleunigen oder steuern.

Level 2 Das zweite Level beschreibt das Vorhandensein von Assistenzsystemen, die dem Autofahrer während des Fahrens gleichzeitig bei dem Bremsen bzw. Beschleunigen und bei dem Steuern helfen.

Level 3 Das dritte Level beschreibt ein Assistenzsystem, dass das Auto vollständig steuern kann, aber in bestimmten Situationen die Kontrolle dem Menschen übergibt.

Level 4 Beim vierten Level steuert das Assistenzsystem das gesamte Fahrzeug und gibt nur in Notfällen die Kontrolle an den Fahrer zurück.

Level 5 Menschen sind nur noch Passagiere in Fahrzeugen und steuern das Fahrzeug nicht mehr.

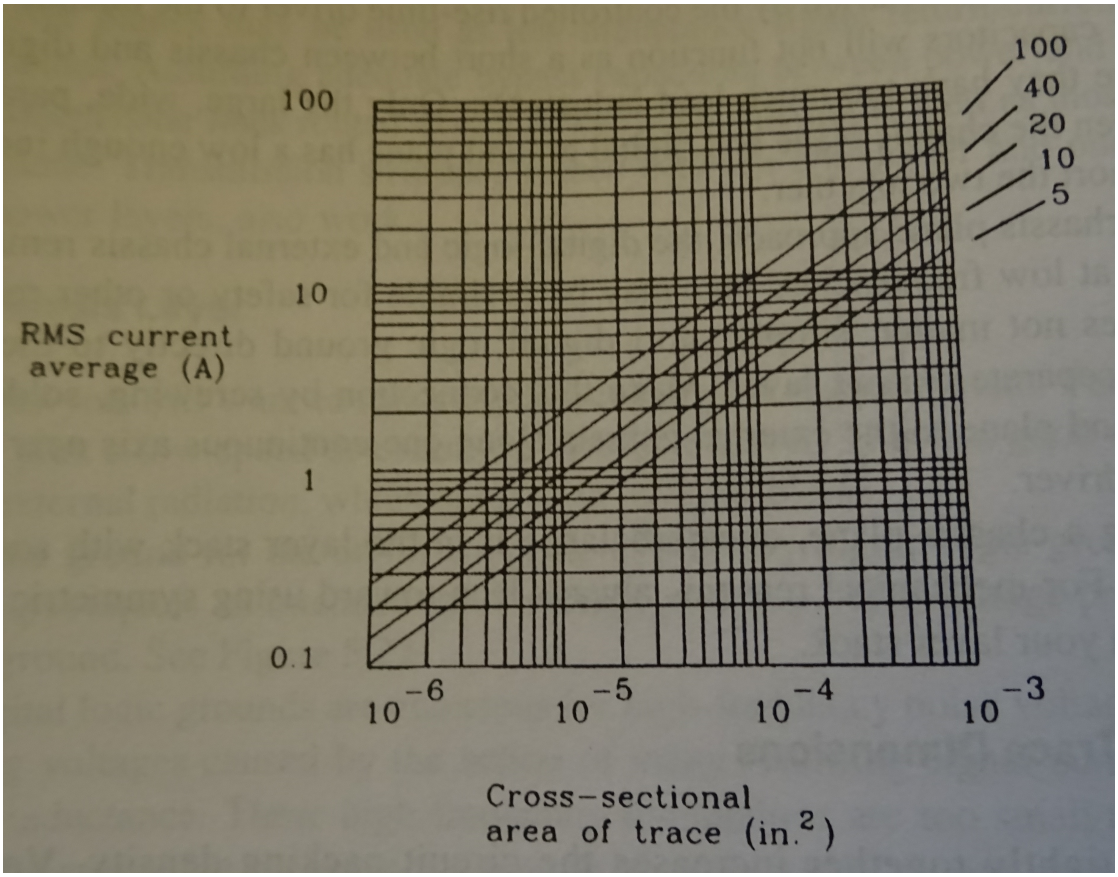
Wenn im Folgenden autonome Fahrzeuge beschrieben werden, ist darunter die Teilmenge der gesamten Fahrzeuge zu verstehen, die durch Sensoren und Aktoren gesteuert werden.

2.6 Elektronik – Schaltungstechnik

Um die unterschiedlichen Hardware Ansätze richtig bewerten zu können, werden im Folgenden einige Grundlagen erläutert.

2.6.1 Spannungsversorgung

Leiterbahnen, durch die Strom fließt, erhitzen sich abhängig von dem Durchmesser der Leitung und der elektrischen Stromstärke. Deswegen ist es wichtig, die Leitungsbreite so zu dimensionieren, dass die Temperatur der Platine nicht über die maximale Temperatur der Bauteile steigt. Im Folgenden ist der Graph der Leitungsdurchmesser und die elektrische Stromstärke abgebildet. [Vgl. S.224ff., 10]



Quelle: [Vgl. S.226, 10]

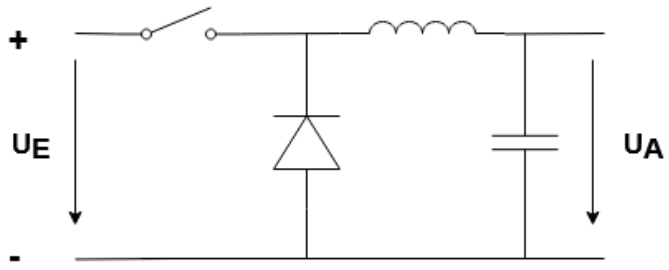
Abbildung 2.7: Temperaturzunahme bei unterschiedlichem Leiterdurchmessern

Für digitale Schaltungen werden Kondensatoren an den Eingängen verwendet. Diese verhindern, dass die Spannung bei schnellen Schaltungen am Eingang der Bauteile fällt. Diese sollen Störungen auf der Versorgungsleitung dämpfen, sodass ein fehlerfreier Betrieb möglich ist. Eine solche Störung kann zum Beispiel das kurzzeitige Einsinken der Versorgungsspannung sein.

2.6.2 Schaltregler

Ein Schaltregler bzw. der spezielle Buck Converter ist ein elektrisches Bauteil, dass aus höheren Spannungen durch das gezielte Regeln eines binären Schalters mit Hilfe eines Tiefpassfilters, eines Oszillators und eines Spannungsteilers für die Rückinformation niedrigere Spannungen erzeugt, um die unterschiedlichen Spannungen für Bauteile auf einer

Platine zu ermöglichen. Mit der Frequenz des Oszillators überprüft der Schaltregler die Spannung am Spannungsteiler und schaltet den Schalter an oder aus. Der Tiefpassfilter schafft mit Hilfe der Induktivität die als Zwischenspeicher der Energie dient, aus den kurzen Spannungsimpulsen eine geglättete Ausgangsspannung.



Quelle: Sebastian Paulsen

Abbildung 2.8: Buck Converter

2.6.3 Differentielle Signale

Bei der Übertragung von Signalen über elektrische Leitungen können äußere Einflüsse, wie elektromagnetische Impulse, die Signale beeinflussen und verzerren. Um dem entgegenzuwirken, werden Techniken angewandt, die die empfangenen Übertragungsinhalte auf ihre Korrektheit überprüfen. Ein Beispiel dafür ist das Paritätsbit, welches indiziert, ob eine ungerade oder gerade Anzahl von High-Signalen in einer Nachricht enthalten sind. Solche Übertragungen sind aber weiter anfällig für Störungen von außen, da sie sich nicht auf jedes Bit einzeln beziehen, sondern die Gesamtheit der Bits betrachten [Vgl. S.66, 2].

Differentielle Signale vermeiden Übertragungsfehler indem sie über zwei elektrische Leitung jeweils ein unnegiertes und ein negiertes Signal übertragen, somit werden die beiden Signale miteinander referenziert. Dabei können die Signale auch negative Spannungen aufweisen. Durch die differentiellen Signale können Fehler in der Übertragungen leichter detektiert werden und vermeiden die Berechnung von Paritätsbits [Vgl. S.1116-1123, 9].

2.7 Leiterplattentechnik

Für die Analyse von Hardware im Konzept Kapitel werden im Folgenden einige Grundlagen vorgestellt.

2.7.1 Lagenaufbau

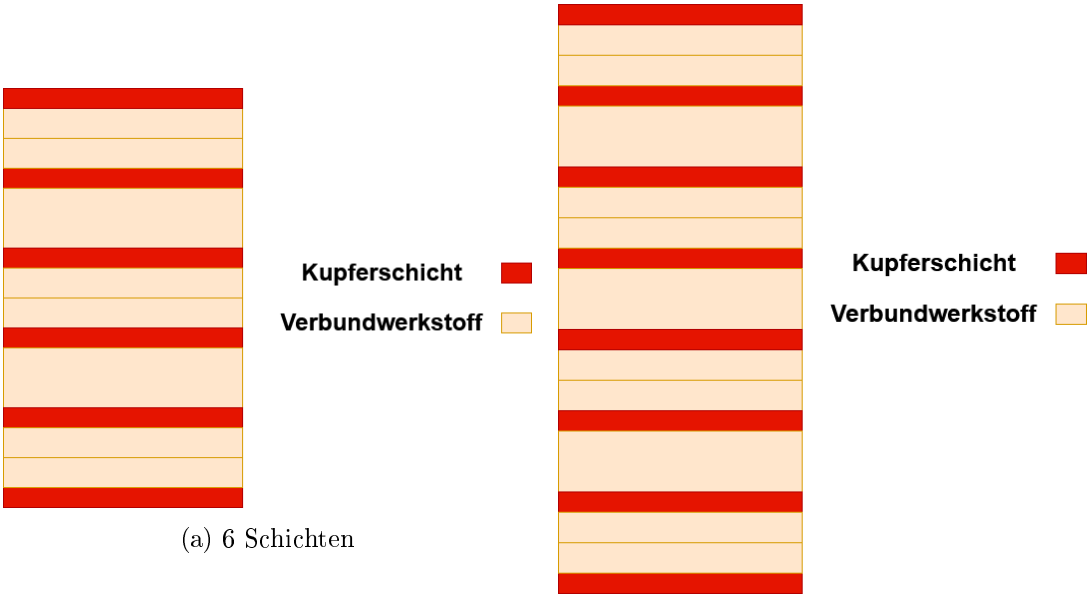
Leiterplatten sind in unterschiedliche Schichten unterteilt. Die Anzahl der Schichten variiert je nach Anwendungsfall und ergibt meist eine Potenz von zwei.

Folgend wird der beispielhafte Aufbau von Platinen mit dargestellt.

(a) 2 Schichten

(b) 4 Schichten

Quelle: Sebastian Paulsen



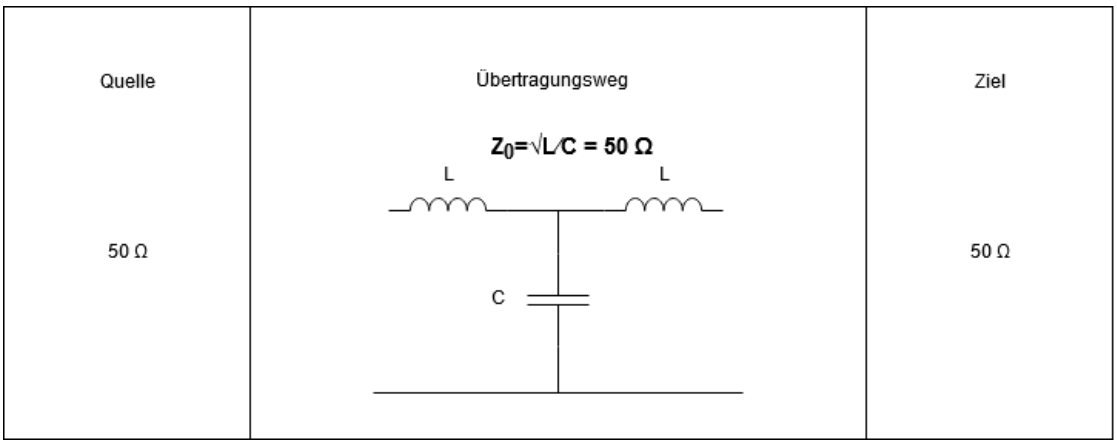
(a) 6 Schichten

(b) 8 Schichten

Quelle: Sebastian Paulsen

2.7.2 Impedanzen

Differentielle Signalleiterbahnen müssen in einer vom Übertragungsstandard vorgegebenen Weise verlegt werden, damit sie resistent gegen äußere Einflüsse sind. Dabei sind die Leiterbahnbreite, die Dicke und die umliegenden Schichten für die Impedanz der Leitung entscheidend. Die folgende Grafik2.11 zeigt die Aufteilung der Impedanzen vom Sender bis zum Empfänger.



Quelle: Sebastian Paulsen

Abbildung 2.11: Grafik Impedanzen Aufteilung Empfänger Sender Leitung

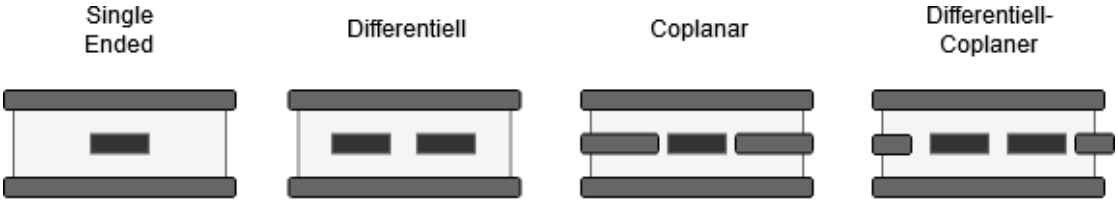
Differentielle Signale nutzen je nach Art der Übertragung unterschiedliche einheitliche Impedanzen. Beispielsweise verwendet Ethernet 50 Ω[9]. Für die Impedanz einer Leitung betrachtet man die folgenden unterschiedlichen Impedanzklassen, die sich nach dem Ort des Bezugspotenzial unterscheiden. [Vgl. S.108-115, 26]

Single Ended Impedance Die Impedanz ergibt sich durch die Anordnung eines einzelnen Leiters über einem oder zwischen zwei Bezugspotenzialen. [S.110, 26]

Differential Impedance Die Impedanz ergibt sich durch zusammengehörige (differenzielle) Signale in zwei parallel zueinander verlaufenden Leitern, die in Referenz zu Bezugspotentialen stehen. [S.110, 26]

Coplanar Impedance Die Impedanz ergibt sich durch einen Leiter, der in eine Potenzialfläche eingebettet ist, bezüglich einer Referenz zu einem oder zu zwei Bezugspotenzialen. [S.110, 26]

Differential-Coplanar Impedance Die Impedanz ergibt sich durch zusammengehörige (differenzielle) Signale in zwei parallel verlaufenden Leitern, die in Potenzialflächen eingebettet sind, mit Referenz zu einer oder zu zwei Bezugspotenzialen. [S.110, 26] Folgende Formel resultiert daraus für die Impedanz.



Quelle: Sebastian Paulsen

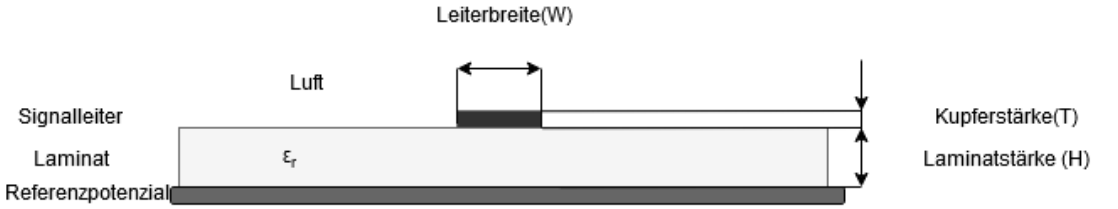
Abbildung 2.12: Grafik unterschiedlicher Impedanzklassen

Die Impedanzklassen werden weiter in Impedanztypen unterteilt, die sich durch die Lage zu den Potenzialen auszeichen:

Microstrip Die Signalleitung ist über einem Potenzial angeordnet. [Vgl. S.110, 26]

Stripline Die Signalleitung liegt zwischen zwei Potenzialen. [Vgl. S.110, 26]

Surface Microstrip



Quelle: Sebastian Paulsen

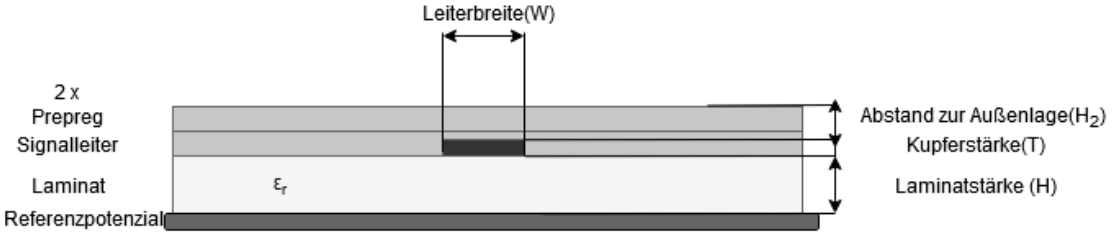
Abbildung 2.13: Surface Microstrip

$$Z_0 = \frac{60}{\sqrt{\varepsilon_{eff}}} \cdot ln\left(\frac{5,98H}{0{,}8\,\mathrm{W} + T}\right) \cdot \Omega$$

mit

$$\varepsilon_{eff} = 0,475\varepsilon_r + 0,67$$

Embedded Microstrip



Quelle: Sebastian Paulsen

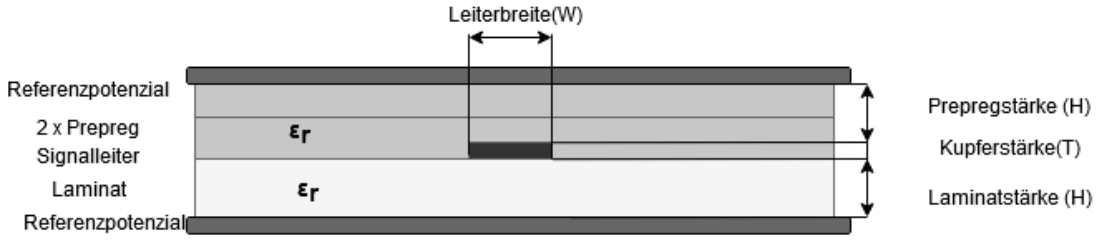
Abbildung 2.14: Embedded Microstrip

$$Z_0 = \frac{60}{\sqrt{\varepsilon_{eff}}} \cdot ln\left(\frac{5,98H_1}{0{,}8\,\mathrm{W} + T}\right) \cdot \Omega$$

mit

$$\varepsilon_{eff} = \varepsilon_r \left(1 - e^{\frac{-1,55(H_1+H_2)}{H_1}}\right)$$

Single Stripline symmetrisch

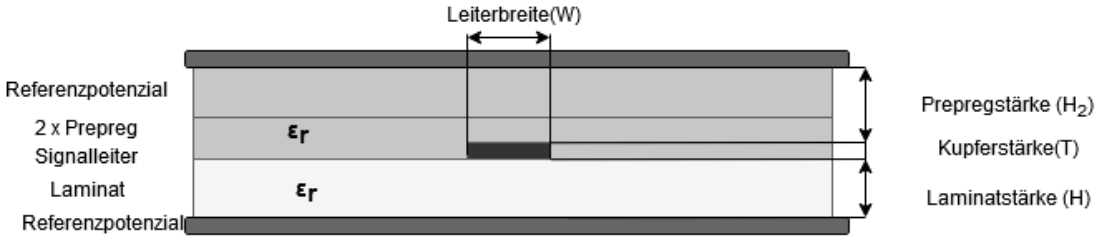


Quelle: Sebastian Paulsen

Abbildung 2.15: Single Stripline symmetrisch

$$Z_0 = \frac{60}{\sqrt{\varepsilon_r}} \cdot ln\left(\frac{1,9(2H+T)}{0{,}8\,\mathrm{W}+T}\right) \cdot \Omega$$

Single Stripline asymmetrisch



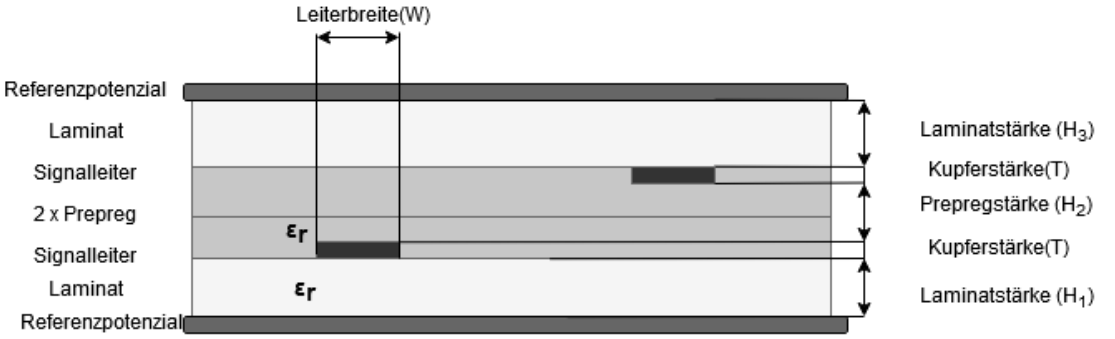
Quelle: Sebastian Paulsen

Abbildung 2.16: Single Stripline asymmetrisch

$$Z_0 = \frac{80}{\sqrt{\varepsilon_r}} \cdot ln\left(\frac{1,9(2H_1+T)}{0{,}8\,\mathrm{W}+T}\right) \cdot \left(1-\frac{H_1}{4H_2}\right) \cdot \Omega$$

bei $H_1 \leq H_2$

Dual Stripline



Quelle: Sebastian Paulsen

Abbildung 2.17: Dual Stripline

$$Z_0 = \frac{80}{\sqrt{\varepsilon_r}} \cdot ln\left(\frac{1,9(2H_1 + T)}{0{,}8\,\mathrm{W} + T}\right) \cdot \left(1 - \frac{H_1}{4(H_2 + H_3 + T)}\right) \cdot \Omega$$

Der Hauptgrund die Leitungen an die Impedanzen anzupassen sind Reflexionen auf der Leitung bei sehr schnellen Signalen[Vgl. S.1117f, 9]. Nach den theoretische Grundlagen wird nun mit dem Forschungsstand fortgeführt.

3 Forschungsstand

Der Forschungsstand soll einen Überblick über momentane Entwicklungen im Bereich der autonomen Miniaturfahrzeuge geben.

3.1 Faller & DC-Car-System

Es werden nun zwei Hersteller von selbstfahrenden Fahrzeugen im H0-Standard vorgestellt. Das Fallersystem sowie das DC-Car System verfolgen das Ziel, Modellfahrzeuge im H0 Standard über eine Modelllandschaft fahren zu lassen. Die Steuerung wird mit einem Fahrzeug realisiert, dass eine durch einen Elektromotor angetriebene Hinterachse besitzt und dessen Vorderachse mit einem kleinen Magneten ausgestattet ist. Mithilfe von Elektromagneten unter der Fahrbahn wird die Vorderachse in eine vorgegebene Richtung abgelenkt. So kann der Magnet und damit auch das Fahrzeug der Spur folgen oder je nach Schaltung der Elektromagnete auch Abbiegen und Kreuzungen befahren. Die Steuerung des Elektromotors für die Hinterachse wird von einem Funkmodul in dem Fahrzeug getätigt, das wiederum mit einem weiteren Funkmodul in einem Controller verbunden ist. Dieser Controller ist an einen Computer angeschlossen. Dieses Setup ermöglicht es mehrere Fahrzeuge gleichzeitig zu steuern. Nun im Folgenden Abbildungen 3.1 einer der Fahrzeuge, der jeweiligen Hersteller.

Quelle: https://www.faller.de/xs_db/BILD_-DB/1/161/www/750/161484_fg_01.jpg

(a) Fahrzeug Faller

Quelle: http://www.h0-car-action.de/WIM/911-1.png

(b) Fahrzeug DC-Car

Abbildung 3.1

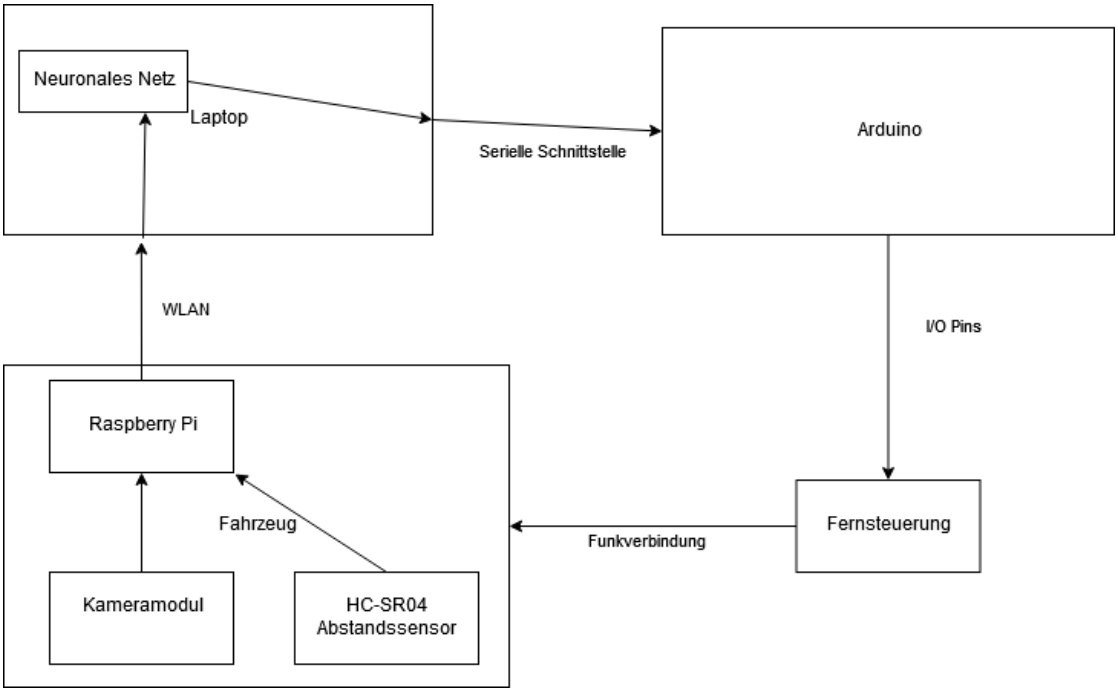
Als Erweiterung zu dem Faller-System besitzen die Fahrzeuge im DC-Car-System noch einen Abstandssensor, der nach Vorne ausgerichtet ist. Mithilfe des Abstandsensors kann das Fahrzeug Hindernisse erkennen und davor stoppen.

3.2 Selbstfahrendes Fahrzeug mit externer Prozesseinheit

Zunächst wird ein Ansatz betrachtet der ein Fahrzeug mit externer Prozesseinheit vorstellt.

Der folgende Ansatz 3.2 von Zheng Wang ist ein kleines selbstfahrendes Auto, das die Sensordaten von Abstandsensor und Kamera via WLAN vom Fahrzeug an einen Laptop sendet. Auf dem ferngesteuerten Fahrzeug ist ein HC-SR04 Abstandssensor, ein Raspberry Pi model B+, die Raspberry Pi Kamera und eine Powerbank verbaut. Der Raspberry PI ist mit einer Kamera über ein Flachbandkabel über die MIPI Schnittstelle verbunden. Der Abstandsensor ist mit dem Raspberry PI über vier Kabel verbunden. Zudem werden die Ausgänge VCC für die Spannungsversorgung, GND für die Masse, Trigger als Input und Echo als Output an den Raspberry PI angeschlossen. Der Raspberry PI initiiert eine Messung, indem er den Triggerpin umschaltet und wartet dann auf die Spannungsflanke der Echo-Leitung. Aus der Laufzeit lässt sich der Abstand ermitteln. Die Komponenten für die Steuerung des ferngesteuerten Modellfahrzeugs sind in ihrem ursprünglichen Zustand gelassen worden. Auf dem Laptop läuft ein TCP/IP Server, der die Daten an

das OpenCV und ein neuronales Netz weiterleitet, die Daten interpretiert und die Steuerungsbefehle an das Fahrzeug zurückleitet. Dies wird dadurch realisiert, dass die Daten vom Laptop über eine serielle Schnittstelle an einen Arduino gesendet werden, der mit seinen Pins direkt an die Fernsteuerung angeschlossen ist, die serielle Befehle interpretiert und das ferngesteuerte Fahrzeug damit steuert.[Vgl. 22]



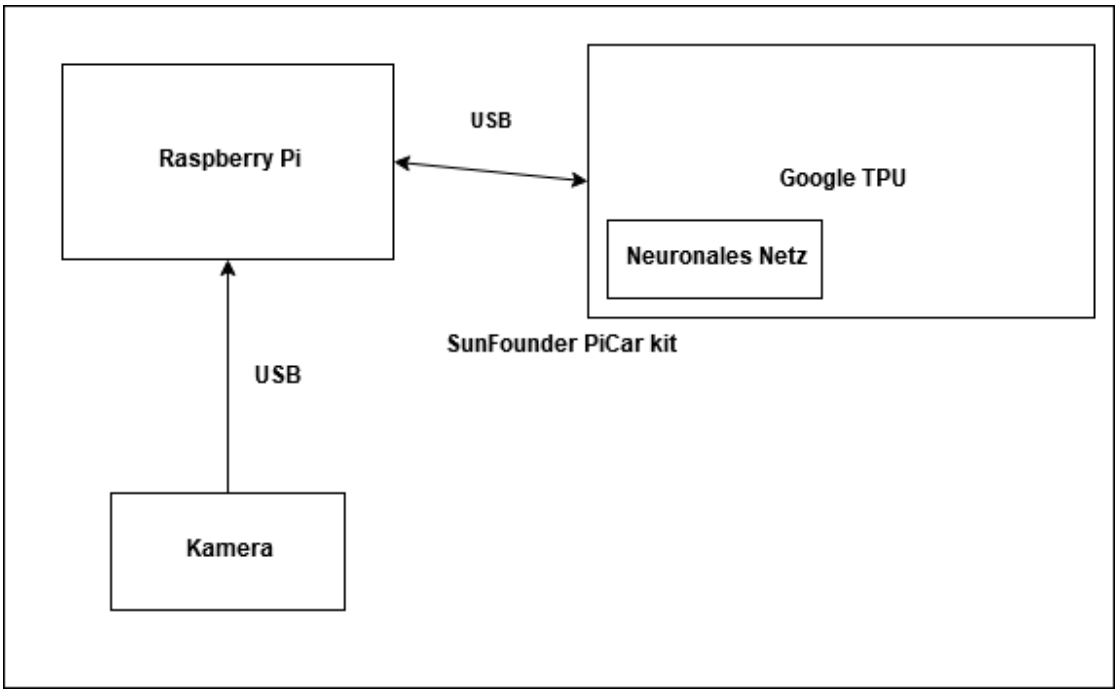
Quelle: Sebastian Paulsen

Abbildung 3.2: Architektur Projekt Zheng Wang

3.3 Selbstfahrendes Fahrzeug mit integrierter Prozessoreinheit

Gegenüber dem vorangegangenen Ansatz verfolgt dieser Ansatz 3.3 von David Tian den Weg die Prozessoreinheit in Form einer Google-Tensor Processing Unit (TPU) und eines Raspberry PIs auf dem Fahrzeug (SunFounder PiCar kit) zu realisieren. Die Google-TPU ist via USB an den Raspberry PI angeschlossen. Die Kamera wird an den Raspberry PI ebenfalls via USB angeschlossen. Hier wird das OpenCV-Framework in Kombination mit dem Tensor Flow-Tool von Google verwendet, um die Kameradaten zu interpretieren

und die Steuerungsimpulse zu generieren. Durch die TPU werden bis zu 12 Bildern pro Sekunde verarbeitet und interpretiert. Würde nur der Raspberry PI die Verarbeitung erledigen, läge die Verarbeitungsgeschwindigkeit nur bei einem Bild pro Sekunde. [Vgl.



Quelle: Sebastian Paulsen

Abbildung 3.3: Architektur PiCar

21] Mit diesem Überblick über einen Teil des Forschungsstandes wird mit dem Konzept fortgeführt.

4 Konzept

Dieses Konzept teilt das Ziel, eine Hardware für ein weitgehend autonomes Miniaturfahrzeug zu entwickeln, in kleinere Teile auf. Aus den Teilzielen werden Anforderungen generiert und diese werden im Folgenden analysiert, evaluiert und konkretisiert. Im ersten Schritt wird das generelle Fahrzeugkonzept eingegrenzt. Im zweiten Schritt wird die Größe des Fahrzeugs festgelegt gefolgt vom dritten Schritt bei dem der Ort der Datenverarbeitung bestimmt wird. Im vierten Schritt wird die Auswahl der berechnenden Hardware getätigt und zuletzt im fünften Schritt wird die Peripherie der berechnenden Hardware gewählt.

4.1 Autonomes Fahrzeug

Diese Arbeit beschränkt sich nur auf das Steuern von zweiachsigen Fahrzeugen mit Elektromotoren als Aktoren und mit Digitalkameramodulen als Sensoren. Für das Fahrzeug bedeutet dies, es erfasst die Umgebung mit Hilfe von Sensoren, bildet diese ab und bewegt sich darin mit Aktoren fort. Als Verbindung und Verarbeitung zwischen der Sensorik und der Aktorik benötigt ein autonomes Fahrzeug eine Intelligenz, welcher es möglich sein sollte, die Sensordaten zu interpretieren und die Aktoren zu steuern. Die zweidimensionale Steuerung des Fahrzeugs wird auf zwei Elektromotoren aufgeteilt, einer der Elektromotoren ist für die Lenkung und der andere ist für den Antrieb zuständig.

Quelle: Sebastian Paulsen

Abbildung 4.1: Konzept Autonomes Fahren mit Sensorik-Prozessor-Aktorik

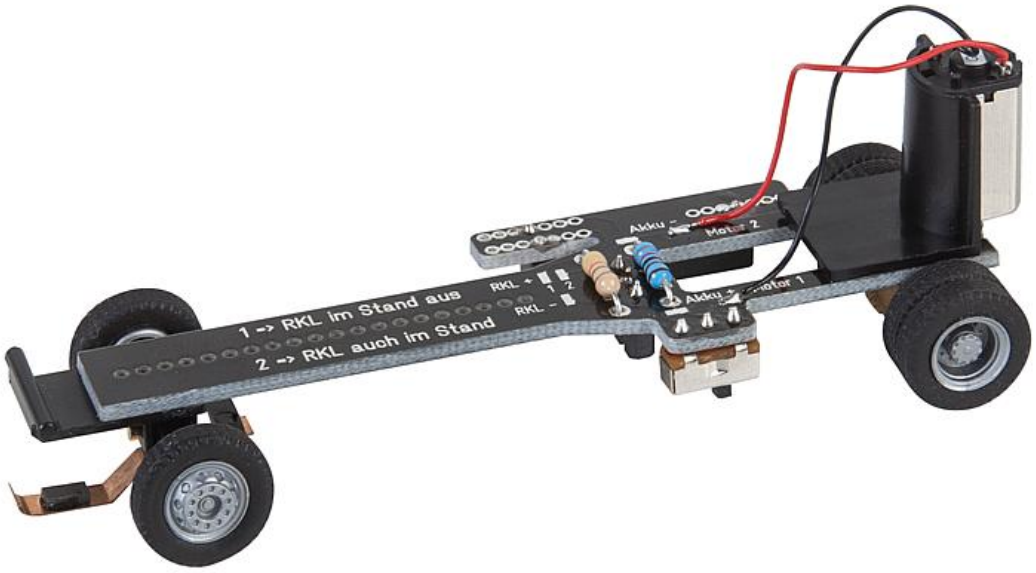
4.2 Fahrzeugabmaße

Im zweiten Schritt wird eine Größe für das Fahrzeug festgelegt und die daraus resultierenden Anforderungen evaluiert. Es bietet sich an, eine Größe zu wählen, in der es bereits Entwicklungen gibt, mit denen man das Fahrzeug vergleichen kann und die eigene Arbeit somit daran anknüpfen kann. Es wird der 1:87 H0 Standard gewählt, denn dieser ermöglicht durch die weite Verbreitung im Modellbau und die Auswahl an Fahrzeugen eine Referenzmöglichkeit. Außerdem ist ein Bausatz für ein Chassis im H0-Standard bereits im Rahmen der Hochschule vorhanden und kann für diese Arbeit benutzt werden.

Aufgabe ist es nun, zu analysieren, wie viel Platz in einem Modellfahrzeug, mit einem Verhältnis von 1:87(H0), für die Sensoren, Aktoren und Intelligenz vorhanden ist.

4.2.1 Erweiterung des Chassis - Design und Herstellung

Der Faller-Bausatz "Car System Chassis-Kit"hat keine Möglichkeit die Bauteile zu befestigen, daher wird ein 3D-Modell erstellt, dass auf das vorhandene Chassis aufgesetzt werden kann. Doch um das 3D-Modell zu erstellen, müssen die Abmaße bestimmt werden.



Quelle: https://www.faller.de/xs_db/BILD_DB/1/163/www/750/163703_fg_01.jpg

Abbildung 4.2: Faller Car System Chassis-Kit

	Bus	Gigaliner Zugmaschine	Gigaliner Auflieger
Breite	31 mm	28 mm	28 mm
Höhe	42 mm	30 mm	30 mm
Länge	137 mm	85 mm	160 mm

Tabelle 4.1: Messung der Größen von H0 Bus und Gigaliner

Referenz Messung im Miniaturwunderland Für die Ermittlung der maximal zulässigen Abmaße für das Fahrzeug im Modellbau mit dem Standard H0 werden Messungen im Miniatur Wunderland durchgeführt, welche diesen Standard verwenden. Im Miniatur Wunderland besteht die größte zusammenhängende Modellanlage der Welt. Im Rahmen der Zielsetzung dieser Arbeit wurde der Fokus auf die größten Modelle im Miniatur Wunderland gelegt, um im Rahmen des H0-Standards den größtmöglichen Platz im 3D-Modell zu erhalten. Die folgenden Abbildungen 4.3 4.4 zeigen die gemessenen Modelle.

Quelle: http://www.modellbahn-alstertal.de/WebRoot/Store21/Shops/62437818/4B88/1531/9777/D586/45FE/C0A8/2936/00FE/156646-1.jpg

Abbildung 4.3: H0 Gigaliner

Quelle: https://i.ebayimg.com/images/g/5SIAAOSwTnBczCHc/s-l640.jpg

Abbildung 4.4: H0 Bus

Die Abmessungen aus der Messung werden nun mit einem Bus in Originalgröße verglichen, um die gemessenen Werte zu verifizieren Als Vergleich wird ein Bus der Firma Mercedes Benz herangezogen. Dieser hat eine Breite von 2550 mm eine Länge von 12 135 mm und Höhe von 3400 mm. Dies entspricht bezogen auf den H0 Standard:

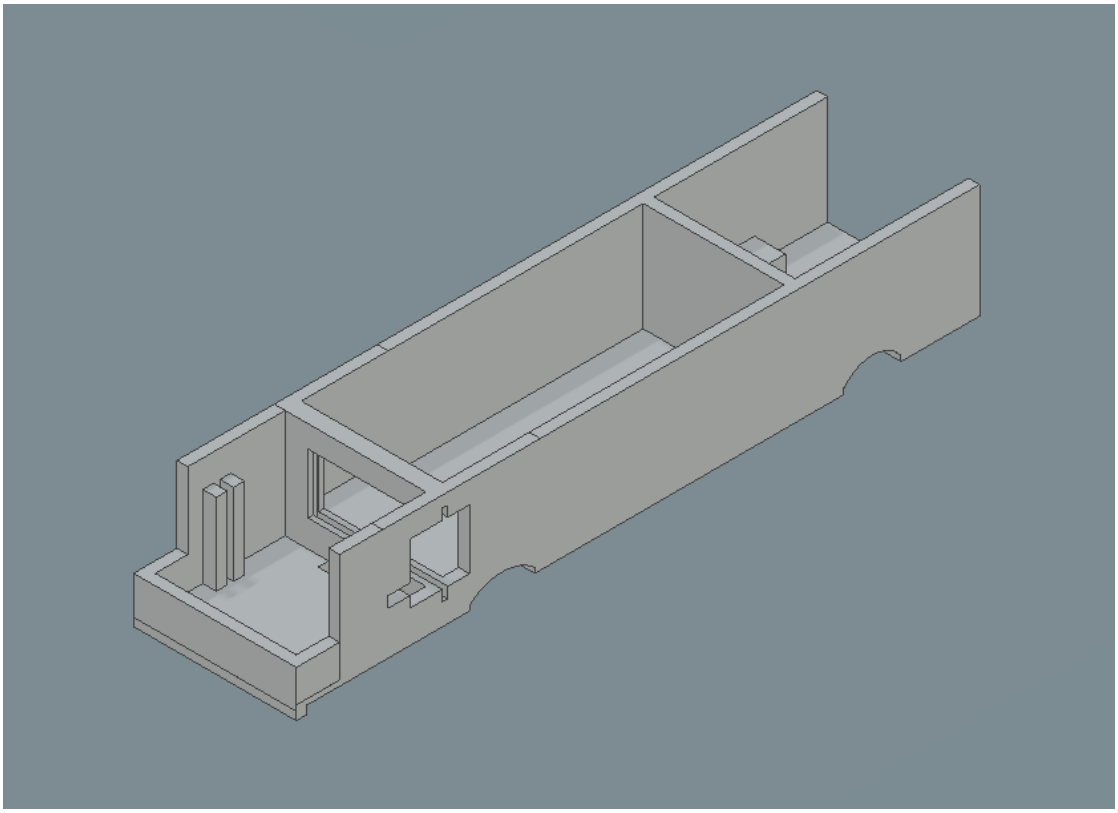
$$\frac{2{,}550\,\text{mm}}{87} \approx 30\,\text{mm}\ \textit{Breite}$$

$$\frac{12{,}135\,\text{mm}}{87} \approx 140\,\text{mm}\ \textit{Länge}$$

$$\frac{3{,}400\,\text{mm}}{87} \approx 39\,\text{mm}\ \textit{Höhe}$$

[S.4, 8]
Die Messungen werden von der Berechnung bestätigt. Mit diesen Abmessungen des Busses kann das 3D-Modell erstellt werden.

CAD Modell Für das Zeichnen des Chassis wird mit der Software Inventor von Autodesk gearbeitet. Diese erlaubt es Zeichnungen zu erstellen und zu exportieren um die Zeichnungen dann mit einem 3D-Drucker zu realisieren. In das Design des Modells wird das Faller-Chassis mit aufgenommen und Platz für die Chassisgrundplatte gelassen. Im Chassis-Set enthalten ist ein DC-Motor für den Antrieb der Hinterachse. Für die Steuerung der Vorderachse wird ein vorhandener Servomotor verwendet. Dieser muss so platziert werden, sodass er mit der Lenkerstange der Vorderachse verbunden werden kann. Außerdem wird im vorderen Bereich des 3D-Modells eine Halterung für eine Kamera eingebaut. Als Vorlage wird hierfür die Raspberry PI Kamera verwendet.



Quelle: Sebastian Paulsen

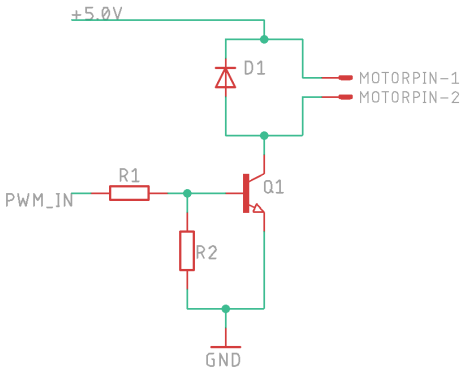
Abbildung 4.5: CAD-Zeichnung des Chassis

Zur Überprüfung der Ergebnisse wird im Miniatur Wunderland noch eine Gegenüberstellung von dem H0 Bus und dem erstellen Modell getätigt.

Quelle: Sebastian Paulsen

Abbildung 4.6: Gegenüberstellung 3D-Modell mit H0 im Miniatur Wunderland

Elektronische Steuerung Für die realistische Ermittlung des vorhandenen Platzes für die Platine im Modell wird auch die elektrische Schaltung zum Betrieb des Fahrzeuges realisiert. Dafür wird eine Schaltung entwickelt, die die Ansteuerung des DC-Motors mit einem Pulsweitenmodulationssignal ermöglicht. Dies ist notwendig, da die I/O-Ports von Platinen zwar die benötigte Spannungen liefern aber nicht den benötigten Strom.



Quelle: Sebastian Paulsen

Abbildung 4.7: Schaltung PWM-Steuerung

Nach dem Einsetzen der Schaltung, der Kamera und der Motoren in das Fahrzeug wird der Platz für die Hardware in der Umsetzung des 3D-Modells vermessen.
Vermessung des inneren Kerns beträgt:

- 64 mm Länge
- 18 mm Tiefe

Ergebnis der Evaluation des Chassis Dadurch, dass die Wände noch entfernt bzw. versetzt oder der DC-Motor gedreht werden kann, wird auf die Länge des inneren Kerns, 12 mm addiert.
Die maximal verfügbare Gesamtlänge im Inneren des Fahrzeugs für die Platine beträgt 76 mm. Die maximal verfügbare Gesamttiefe beträgt in Folge dieser Messungen 18 mm. Die maximal verfügbare Gesamtehöhe beträgt 31 mm

4.3 Verarbeitung der Kameradaten

Im dritten und nun folgenden Schritt wird die Verarbeitung der Kameradaten betrachtet. Um die Kameradaten zu analysieren und zu interpretieren, bietet es sich an, ein neuronales Netz zu verwenden. Wie in den theoretischen Grundlagen erläutert, ermöglichen es

trainierte neuronale Netze, Bilder zu interpretieren. So sollen zum Beispiel die Straßenführung oder auch Straßenschilder erkannt werden. Aus diesen Interpretationen lassen sich Steuerimpulse für die Aktorik generieren.

4.3.1 Datenverarbeitung im Fahrzeug oder extern

Wie bereits im Forschungsstand beschrieben, gibt es zwei unterschiedliche Ansätze, wie in einem autonomen Fahrzeug die Kameradaten interpretiert werden.

Beim ersten Ansatz wird das Fahrzeug nicht direkt von der Hardware auf dem Fahrzeug gesteuert, sondern sendet die Bilder der Kamera über W-LAN an einen Computer. Der Computer leitet die Bilder nach der Übertragung an die Eingangsschicht eines trainierten neuronalen Netzes weiter. Dieses neuronale Netz verarbeitet die Bilder auf dem Computer und ermittelt die Steuerimpulse daraus, die an das Fahrzeug zurück übermittelt werden.

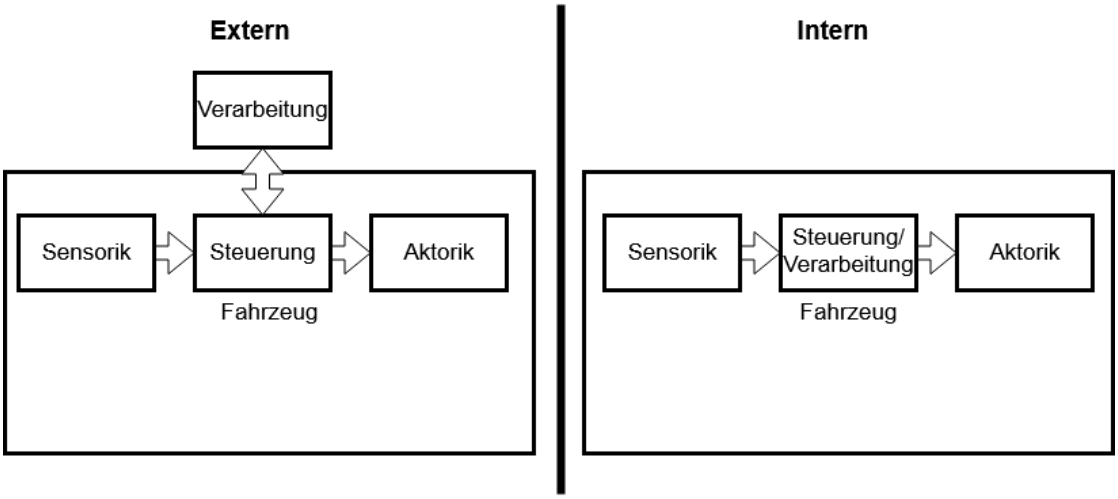
Beim zweiten Ansatz wird die Verarbeitung der Bilder mithilfe eines neuronalen Netzes auf dem Fahrzeug selber ausgeführt. Dies vermeidet die Übertragung der Daten vom Fahrzeug zum Computer und erlaubt es, die Steuerimpulse direkt an die Aktorik zu leiten.

Evaluierung der Ansätze Die Vorteile des ersten Ansatzes zeichnen sich dadurch aus, dass die Hardware auf dem Fahrzeug nur die Übertragung der Kameradaten und die Steuerung der Aktorik ausführen muss. Dies ermöglicht eine kompaktere Bauform und ermöglicht es, durch die Rechenleistung des Computers sehr viele Daten zu verarbeiten, welche nur durch die Übertragungsgeschwindigkeit gedrosselt werden.

Die Vorteile des ersten Ansatzes bestehen darin, dass die Übertragungszeiten zwischen den einzelnen Komponenten des Aufbaus relativ gering sind, sodass im Gegensatz zum Ansatz mit externer Hardware keine Übertragungsprotokolle implementiert werden müssen und, dass keine externe Hardware benötigt wird.

Der Nachteil des ersten Ansatzes ergibt sich aus der Übertragung der Daten vom Fahrzeug zum Computer. Diese können durch Interferenzen gestört werden oder abbrechen, weil die Übertragungsdistanz zwischen den beiden Komponenten zu groß wird.

Der Nachteil des zweiten Ansatzes besteht darin, dass die Umsetzung im Vergleich zum ersten Ansatz komplexer ist und weniger Rechenleistung als beim ersten Ansatz vorhanden ist. Die Visualisierung der beiden Ansätze kann in der folgenden Grafik betrachtet werden 4.8.



Quelle: Sebastian Paulsen

Abbildung 4.8: Ansatzübersicht

Ergebnis der Evaluation Der zweite Ansatz wird gewählt, denn dieser verspricht durch die fehlende Datenübertragung eine geringere Fehleranfälligkeit und macht externe Hardware gegenstandslos.

4.3.2 Auswahl der verarbeitenden Hardware

Neuronale Netze benötigen durch die Menge an Multiplikationen und Additionen eine enorme Rechenleistung und werden deswegen, neben CPU und GPU, häufig auch auf spezialisierter Hardware wie beispielsweise Application-Specific Integrated Circuits (ASIC) implementiert. Im Folgenden werden die Komponenten FPGA, CPU und GPU nach ihrer Fähigkeit ein neuronales Netz zu Implementieren evaluiert.

tinyTPU Ein vielversprechender Ansatz für ein neuronales Netz implementiert dieses auf einem SoC. Der SoC besteht hier aus einem FPGA und einen dual-core Mikrocontroller. Dieser Ansatz macht sich die Rechenleistung der Digital Signal Processor (DSP)-

Blöcke auf dem FPGA des SoCs zu nutze. Genauer verwendet dieser Ansatz das Z-turn Board mit dem Xilinx XC7Z020 SoC, welches einen FPGA und einen Mikrocontroller integriert hat[6].

Evaluierung der Hardwareoptionen
In [18] werden CPU, GPU und FPGA mithilfe des Min-Warping Algorithmus miteinander verglichen. Der Min-Warping-Algorithmus ist mit seiner parallelen Datenverarbeitung für einen Vergleich zwischen paralleler Hardware geeignet. Dabei wird besonders auf die Energieeffizienz und die Rechenleistung geachtet. Die Implementation auf dem FPGA wird hier wie in [6] mit Hilfe der DSP-Blöcke realisiert. Im niedrigen Leistungsbereich ist der FPGA schneller und effizienter als CPU und GPU. Eine weiter Publikation die diese These stützt ist [1]

Ergebnis der Evaluation Die Wahl fällt auf den FPGA integriert mit einem Mikrocontroller auf einem SoC, da der FPGA im Vergleich energieeffizienter und leistungsstärker ist.

4.3.3 Training von neuronalen Netzen

Neuronale Netze benötigen Training mit aufwändigen Lernalgorithmen, die auf kompakten Systemen zu Speicherplatzproblemen führen können. Es bietet sich daher an, das Training des neuronalen Netzes auf einem externen Rechner zu tätigen und das trainierte Netz zu übertragen. Dies ist, unter dem Gesichtspunkts des Speicherbedarfs der Trainingsdaten betrachtet, eine elegante Lösung, die es vermeidet, Lernalgorithmen wie genetische Algorithmen implementieren zu müssen und Speicherplatz auf dem SoC zu verbrauchen.

4.3.4 Boardauswahl oder eigene Platine

Nachdem der FPGA bzw. der SoC als verarbeitende Hardware festgelegt wird, wird im vierten Schritt nun die Frage nach der umliegenden Peripherie geklärt.

Vorstellung der Ansätze Ein Ansatz wäre es, bereits vorhandene Boards zu verwenden, auf denen der Xilinx XC7Z020 SoC aus [6] verwendet wird. Es gibt mehrere Evaluationsboards, die diesen SoC verwenden, wie z.B. das PYNQ-Board von Digilent, das TE0720 von Trenz Elektronik oder das ZED-Board von AVNET. [14]

Ein zweiter Ansatz wäre es, eine eigene Platine zu designen und fertigen zu lassen, die den Platzanforderungen entspricht.

Als dritter Ansatz wäre es möglich, die Funktionalität unterschiedlicher Boards mit einer Trägerplatine zu verbinden, beispielsweise werden dabei die Steuerung des Fahrzeugs und das Interpretieren der Kameradaten auf zwei Boards aufgeteilt.

Evaluierung der Ansätze Der letzte vorgestellte Ansatz fällt aus der Auswahl, da die Wahl der Hardware auf einen SoC fiel und somit das gesamte Projekt auf diesem SoC umgesetzt werden muss, ist keine weitere Berechnungshardware notwendig, weil der SoC bereits die Möglichkeiten bietet, die die Trägerplatine gehabt hätte.

Der Vorteil des ersten Ansatzes besteht darin, dass von den Herstellern bereits getestete und funktionierende Hardware zur Verfügung gestellt wird. Die Peripherie der Boards wird nach der Möglichkeit, eine Kamera anzuschließen, untersucht:

Board	Peripherie	Breite	Länge
PYNQ	**HDMI in, Ethernet, I/O Ports, USB OTG, PMOD**	8,7 cm	12,2 cm
Z-Turn	**USB OTG, Ethernet, 2x** 1,27 mm **pitch 80-pin SMT female connectors**	6,3 cm	10,2 cm
Trenz	**2x B2B connector Samtec Razor Beam LSHM-150, B2B connector Samtec Razor Beam LSHM-130**	4 cm	5 cm

Tabelle 4.2: Peripherie Vergleich

[14] [3] Das Trenz-Board hat keine Peripherie und es ist nötig, ein Trägerboard zu entwickeln, deswegen fällt dieses Board auch aus der Auswahl wie der zu letzt genannte Ansatz. Sowohl das PYNQ-Board sowie das ZED-Board ermöglichen das Anschließen einer Kamera, jedoch ist das ZED-Board wesentlich kleiner als das PYNQ-Board und daher zu präferieren.

Der Nachteil des ersten Ansatzes ist es jedoch, dass die Boards nicht die Platzanforderungen erfüllen. Die recherchierten Boards sind zu groß für das Modell.

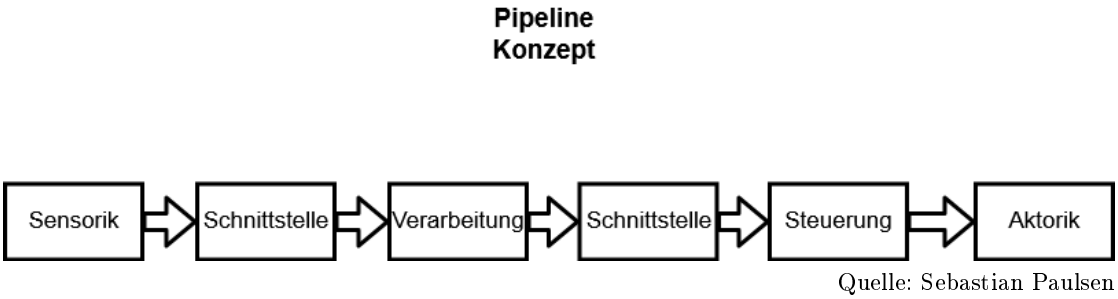
Der zweite Ansatz hat den Vorteil, dass die Auswahl der Peripherie gänzlich den Anforderungen angepasst werden kann und sie deswegen sehr flexibel ist.

Der zweite Ansatz hat hingegen den Nachteil, dass die Entwicklung von eigener Hardware aufwändiger ist.

Ergebnis der Evaluation Die Argumente sprechen für den zweiten Ansatz, da dieser Flexibilität bietet, dies ist beim zweiten Ansatz durch vorhandene Bauteile bereits eingeschränkt und überschreitet nicht die Platzanforderung.

4.4 Pipelinekonzeption

Im fünften Schritt nun ein Ansatz der in die Zielsetzung integriert wird. Der Vorteil einer Pipeline ist es, dass die Aufgabe in kleinere Unteraufgaben aufgeteilt werden können, welche parallel von dedizierten Komponenten bearbeitet werden können. Dies wird nun auf die Problemstellung der Verarbeitung großer Datenmengen angewandt. Daher werden nun aus der Problemstellung Teilaufgaben und Schnittstellen definiert. Die Prozesseinheit wird aufteilt in das Verarbeiten der Kameradaten und die Ansteuerung der Motoren. Das Verarbeiten der Kameradaten wird weiter aufgeteilt in Kameradaten empfangen, Kameradatenvorverarbeiten (Filter), Kameradaten analysieren, Kameradaten interpretieren und die Ergebnisse weiterleiten. Die Steuerung teilt sich in das Empfangen von Ergebnissen, das Umwandeln von Ergebnissen in Steuerimpulse und das Ausführen von Steuerimpulsen auf. Die Konzeption wird in folgender Grafik zusammengefügt 4.9.



Quelle: Sebastian Paulsen

Abbildung 4.9: Pipelineüberblick Sensorik zu Aktorik

4.5 Weitere Anforderung an die Platine

Weitere Funktionen, die nicht zwingend für die bisherigen Anforderungen nötig sind, stellen die Folgenden Punkte dar. Weitere Fähigkeit des SoCs ist die Möglichkeit auf dem Mikrocontrollern ein Betriebssystem zu verwenden, doch das Betriebssystem benötigt allerdings Arbeitsspeicher und persistenten Speicher. Mit unterschiedlichen Bootmedien (persistenter Speicher und interner Speicher) wird die Auswahl des Bootmediums vor dem Start des Bootloaders zwingend. Die Ausführung des gewählten Bootmediums übernimmt ein Bootloader. Dieser muss während des Bootvorgangs bereits das gewählte Medium kennen. [Vgl. S.170, 24] Daraus folgend werden in die Architektur ein Speichermedium, ein Arbeitsspeicher und eine Möglichkeit, dem Bootloader die Auswahl mitzuteilen, integriert.

4.6 Zusammenfassung des Konzepts

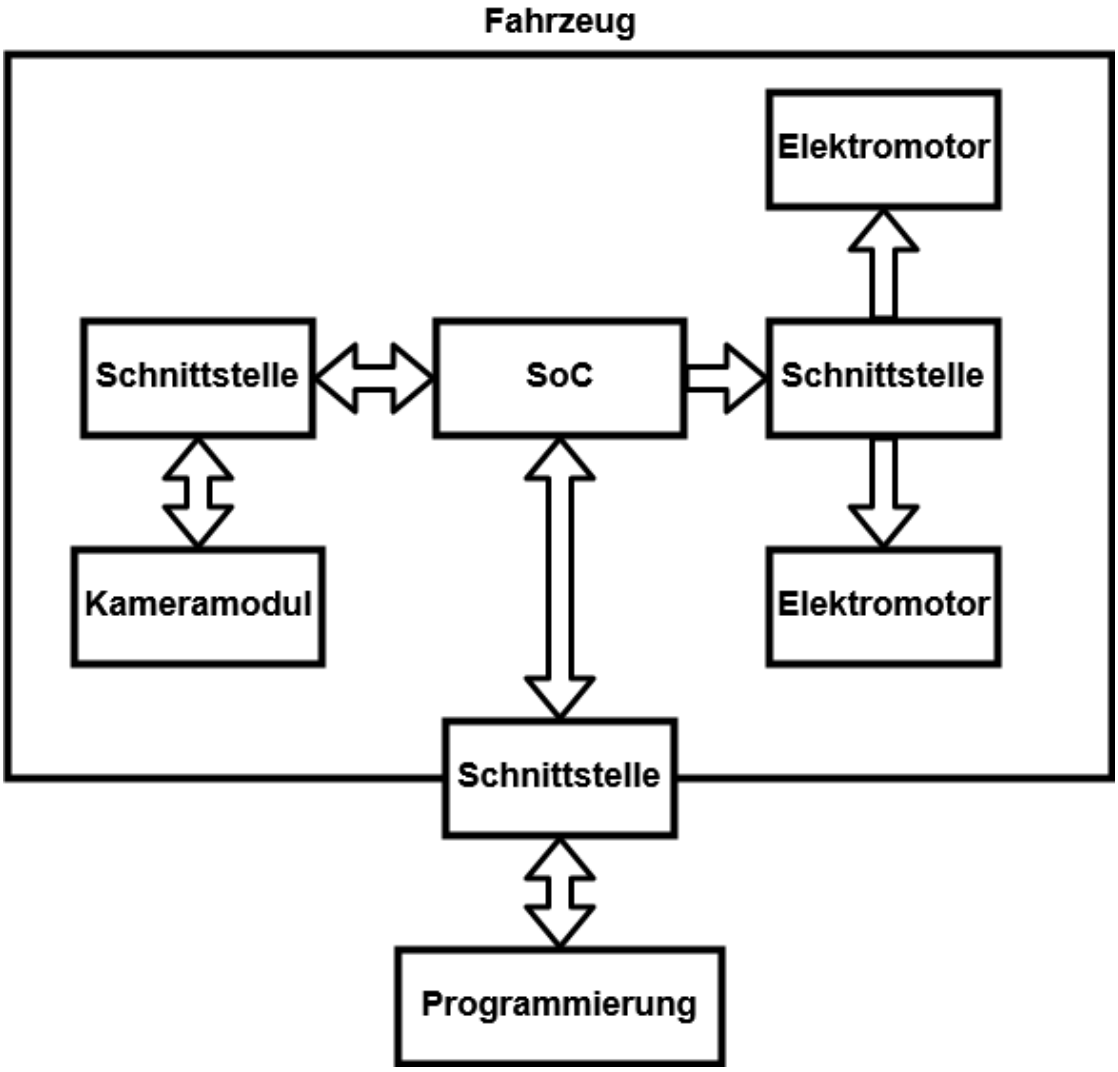
Es wird eine Platine entwickelt, die an zwei Elektromotoren angeschlossen ist und die Elektromotoren zu steuern vermag - Dabei ist die Leistungselektronik extern realisiert und nicht Teil des Konzepts der Platine. Desweiteren ist auf der Platine ein SoC verbaut, der über eine Übertragungsschnittstelle an eine Kamera angeschlossen ist. Die Daten werden von der Übertragungsschnittstelle zu einem im FPGA des SoC implementierten neuronalen Netz weitergeleitet und interpretiert. Aus den interpretierten Daten sollen dann Steuerimpulse generiert werden, die an die Elektromotoren weitergeleitet werden. Ferner soll der SoC außerhalb der Platine programmiert werden können und die gesamte Platine soll die für die Funktionen benötigten Spannungen aus einer Gleichspannung generieren können.

Darüber hinaus soll die Platine folgende Abmaße haben und somit in ein Modellautomobil im H0 Standard passen:

- Länge 76 mm
- Breite 18 mm
- Höhe 31 mm

4.7 Architektur des Konzepts

Aus dem Konzept und den Anforderungen lässt sich folgende Grafik der Architektur ableiten 4.10.



Quelle: Sebastian Paulsen

Abbildung 4.10: Architektur des Konzepts

5 Umsetzung

Maßgeblich für die Umsetzung ist die Wahl des SoCs als Prozessoreinheit sowie der verwendete SoC in der Realisierung des neuronalen Netztes aus [6]. Die Umsetzung gliedert sich in zwei Teile, zum einen in die Entwicklung des Schaltplans und zum anderen in den Platinenentwurf. Im Gegensatz zum Platinenentwurf, der nicht aufgeteilt ist, wird der Schaltplan beim Hinzufügen eines Bauteils Stück für Stück erweitert.

Damit nicht die gesamte Platine neuentwickelt werden muss, werden für die Umsetzung diverse im Handel erhältliche Platinen als Referenz genommen.

Eagle Im Weiteren wird die für Studenten frei erhältliche Version 9.4.2 von Eagle für das Zeichnen der Schaltpläne und die Erstellung des Platinenentwurfs verwendet.

5.1 Auswahl des SoC-Modells

Xilinx vertreibt neben dem Z-7020 auch leistungsstärkere SoCs die denselben Aufbau haben wie der Z-7020. Sodass der nächst größere SoC der Z-7030 verwendet werden könnte, ohne Anpassungen am neuronalen Netz oder der Struktur der Umgebung des neuronalen Netzes vornehmen zu müssen.

5.1.1 Vergleich Xilinx Z-7020 mit Xilinx Z-7030

Die signifikanten Daten der SoCs in einer Tabelle zusammengefasst

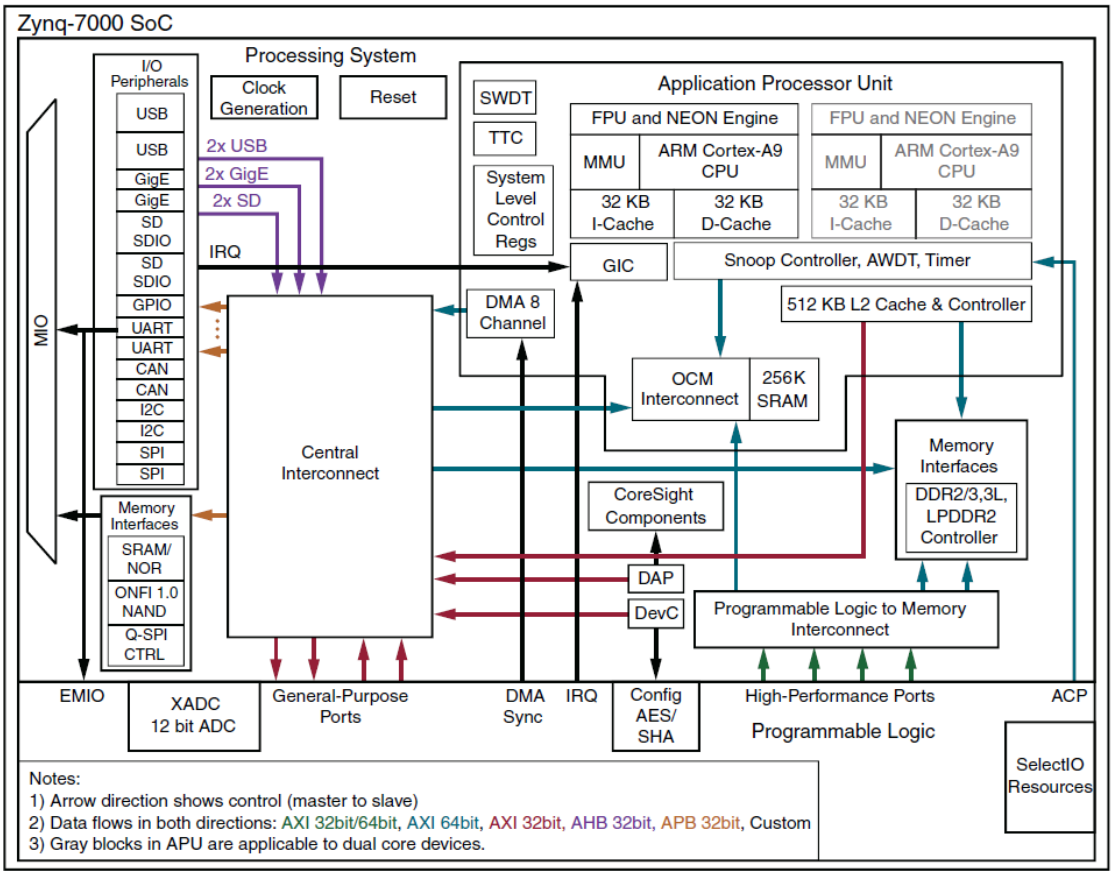
	Z-7020	Z-7030
LUTS	53,200	78,600
Flip Flops	106,400	157,200
DSP Blöcke	220	400
BlockRAM Memory	4,9 Mb	9,3 Mb
Abmaße	17 mm	23 mm

Tabelle 5.1: Vergleich Z-7020 mit Z-7030[Vgl. S.2 23]

Die Anzahl der für das neuronale Netz wichtigen Bestandteile, wie der DSP Blöcke oder des BlockRAMs, sind auf dem Z-7030 fast doppelt vorhanden. Mit diesen signifikanten Vorteilen, im Hinblick auf das neuronale Netz, ist die größere Bauform des Z-7030 zu vernachlässigen und der Z-7030 zu verwenden.

5.2 Xilinx Z-7030

Der Z-7030 wird in zwei Bereiche unterteilt, die Programmable Logic (PL) und das Programming System (PS). Im Grunde genommen entspricht die PL dem FPGA und das PS dem Mikrocontroller.



Quelle: [23, S.27]

Abbildung 5.1: Block Diagram

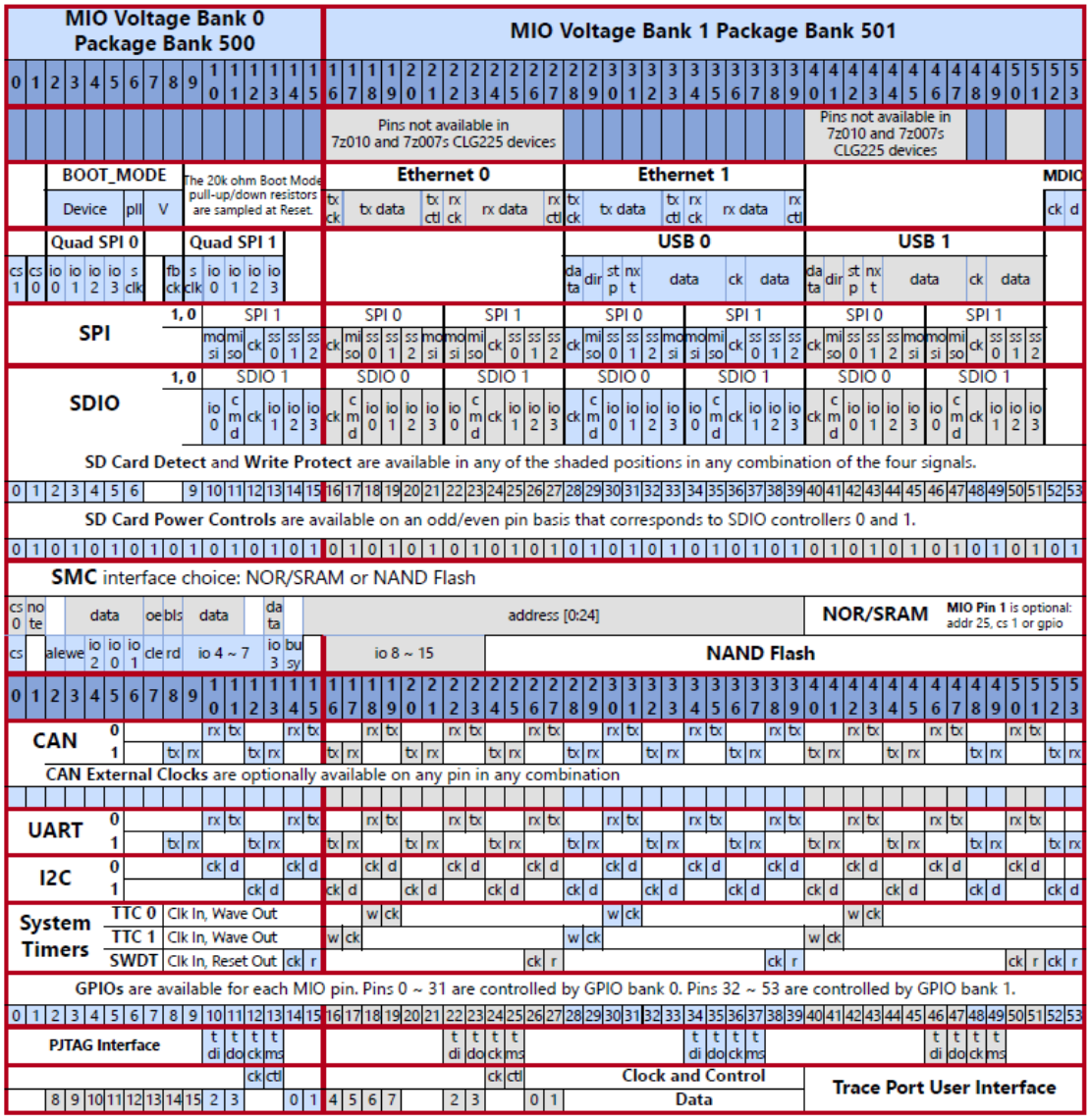
5.2.1 IO und Controller

Im Folgenden werden einige Aspekte des Z-7030 vorgestellt.

AXI AXI steht für Advanced eXtensible Interface, es verbindet die einzelnen Komponenten des SoCs miteinander und ermöglicht die Ansteuerung von den gleichen I/O Pins aus der PL sowie aus dem PS. Dies vollzieht sich in einem Master-Slave-System, welches mit SPI vergleichbar ist. [24]

MIO Multiplexed I/Os (MIO) ermöglichen es die selben I/O-Pins durch multiplexing an das PS oder die PL zu binden. Diese sind besonders vielseitig, da sie intern an un-

terschiedliche im SoC vorhandene Controller wie CAN, SPI, I2C angeschlossen werden können. Eine Übersich über die verbauten Controller und die MIO Pins folgen in dieser Grafik 5.2.



Quelle: [24, S.52]

Abbildung 5.2: MIO Übersicht

5.2.2 LVDS

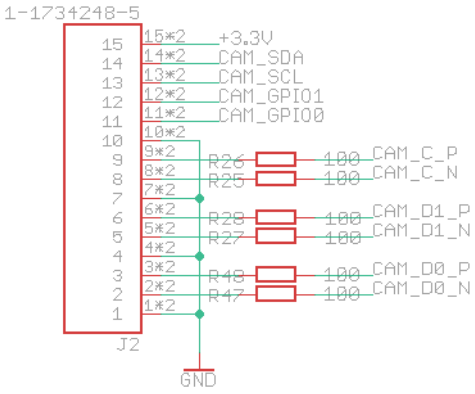
Low Voltage Differential Signaling(LVDS) ist eine Form der seriellen Datenübertragung. Der SoC besitzt zwei verschiedene Varianten, die mit den Spannungen 2,5 V oder 1,8 V betrieben werden.

5.3 Kamerainterface

Der Xilinx Z-7030 SoC hat diverse Schnittstellen für den Anschluss von Kameras wie z.B. USB OTG oder Ethernet. Desweiteren besitzt er LVDS. Diese erlauben es, Interfaces zu implementieren, die nicht direkt mit einem der internen Controller funktionieren. So wie auch das MIPI D-PHY Interface, dass im Mobiltelefon- und Industriebereich, wie z.B. im Raspberry PI eingesetzt wird und weit verbreitet ist. Für die D-PHY Schnittstelle gibt es im Xilinx-IP-Store einen IP-Core den man käuflich erwerben kann. Der Vorteil dieses Interfaces gegenüber den anderen besteht in der großen Anzahl von passenden Geräten und die hohe Übertragungsgeschwindigkeit von mehreren giga bits per second (gbps).

5.3.1 Pinbelegung Kamera Header

Aus den MIPI Bestimmungen wird also folgender Header genommen.



Quelle: Sebastian Paulsen

Abbildung 5.3: Pin Belegung

5.4 Programmierung des SoC

Nach der Auswahl des SoC besteht der nächste Schritt darin, diesen zu programmieren. Der SoC besitzt eine JTAG-Schnittstelle, die mit vier Pins an Peripherie nach außen gelegt werden kann.

5.4.1 USB-Chip

Der Chip 2232HQ der Firma Future Technology Devices International (FTDI) verbindet mehrere Funktionen in einem Gerät. Dies spart Platz auf der Platine, erschwert allerdings aufgrund der damit verbundenen Komplexität die Konfiguration des Bauteils. Der USB-Chip verfügt über diverse Schnittstellen:

- USB zu dual UART
- USB zu dual FIFO
- USB zu dual JTAG
- USB zu dual SPI
- USB zu dual I^2C
- USB zu Kombination aus zwei Genannten[Vgl. S.1 12]

Das bedeutet, dass der Benutzer die Möglichkeit hat über ein Micro-USB Kabel das Gerät via JTAG zu programmieren, eine serielle Verbindung damit aufzubauen, zu Debuggen und die Platine mit Strom zu versorgen.

5.4.2 Konfiguration des USB-Chips

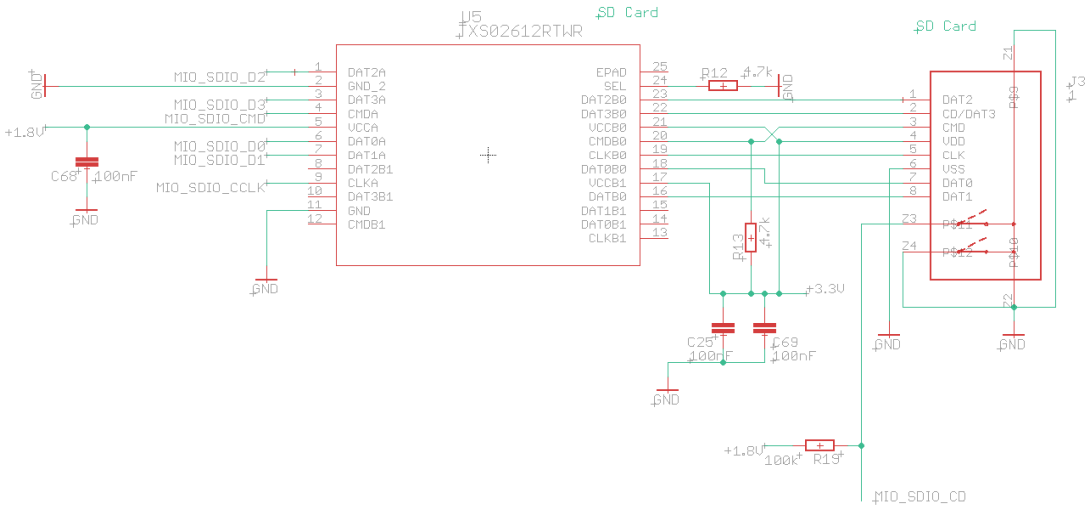
Der FTDI-Chip benötigt zum Speichern der unterschiedlichen Konfigurationen einen EEPROM-Speicher, der an seine EEPROM-Pins angeschlossen wird. Als EEPROM wird ein 93LC66 verwendet. Dieser wird im Manual des USB-Chips aufgeführt.

5.5 Persistenter Speicher des SoC

Für das neuronale Netz wird Speicherplatz für die Bias und Gewichte benötigt. Desweiteren benötigt der Linuxkernel einen Mindestspeicherplatz von 4 Mb. Die Lösung wäre eine SD Karte als persistentes Speichermedium zu verwenden.

5.5.1 SD-Chip

Um die SD Karte benutzen zu können muss sie angesteuert werden, dies geschieht durch einen SD Karten Controller der als Bindeglied zwischen SD-Kartenslot und SoC agiert. Der SD Karten Chip ist mit den MIO Pins 41-45, 46 und der internen SD Schnittstelle des SoCs dem SDIO Controller verbunden.[Vgl. S.366, 24] Dies wird in der folgenden Grafik dargestellt 5.4.



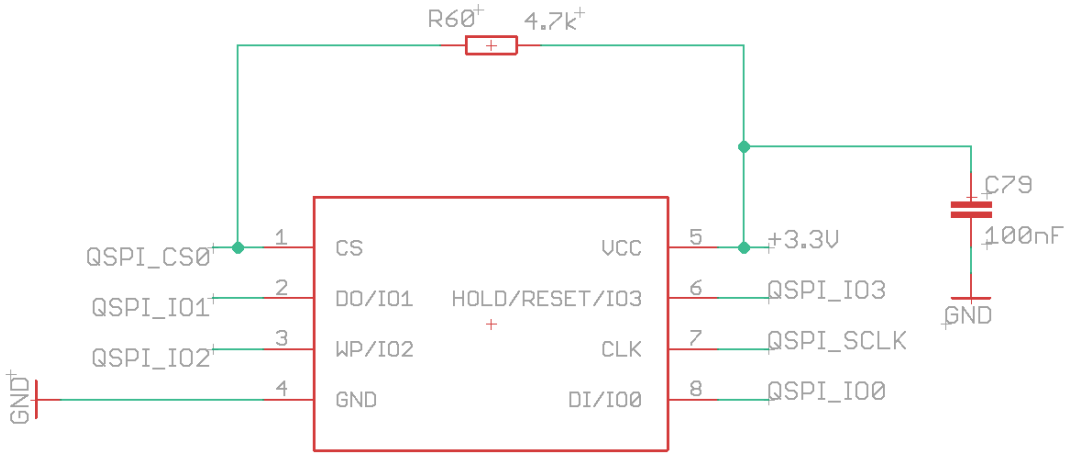
Quelle: Sebastian Paulsen

Abbildung 5.4: SD Chip Symbol

5.5.2 Flashspeicher

Der FPGA des SoCs verliert nach Abschalten der Versorgungsspannung seine Konfiguration. Sobald man den FPGA außerhalb des Setups, mit direkter Verbindung zum PC via USB Kabel betreiben möchte, wird daher persistenter Speicher notwendig. Ein Ansatz

ist es, den FPGA seine Konfiguration von einem Flashspeicher laden zu lassen. Dieser behält auch nach dem Abschalten der Versorgungsspannung seine Daten bei und kann wiederholt programmiert werden. Die Auswahl des Speichers ist von dessen Platzbedarf, dessen Speichergröße und dessen Möglichkeit der schnellen Übertragung via QSPI abhängig. Deswegen wird der W25Q128JVSIQ verwendet. Hier der Schaltplan 5.5.



Quelle: Sebastian Paulsen

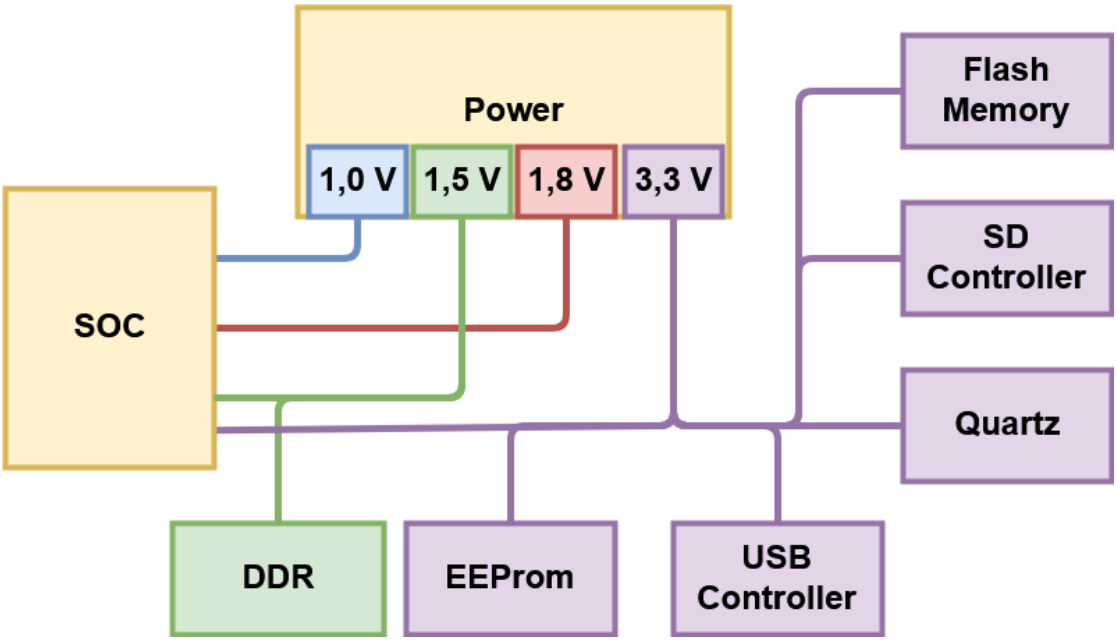
Abbildung 5.5: W25Q128JVSIQ

5.6 Auslagerungsspeicher - DRAM

Die zusätzliche Anforderung ein Linux Betriebssytem auf dem Board ausführen zu können, macht es notwendig, Daten des Prozessors auslagern zu können und 4 Mb festen Speicherplatz für das Betriebssystem zur Verfügung zu stellen. Der gewählte SoC ist mit einem 256 KB großen SRAM-On-Chip ausgerüstet [Vgl. S.31, 24] Dieser Speicher ist für ein Linux Betriebsystem zu gering, deswegen wird ein externer Speicher verwendet. Der SoC hat für diesen Fall einen DDR Controller integriert, der DDR3, DDR3L, DDR2, LPDDR-2 unterstützt. [Vgl. S.32, 24] Es bietet sich an, einen DRAM als Speicher und mit dem DDR Controller zusammen zu verwenden. Eine weitere Möglichkeit wäre es, die SD-Karte als Auslagerungsspeicher zu verwenden. Diese wäre aber im Vergleich deutlich langsamer.

5.7 Spannungsversorgung

Der Zusammenhang der Betriebsspannungen auf der Platine stellt sich wie folgt dar 5.6.



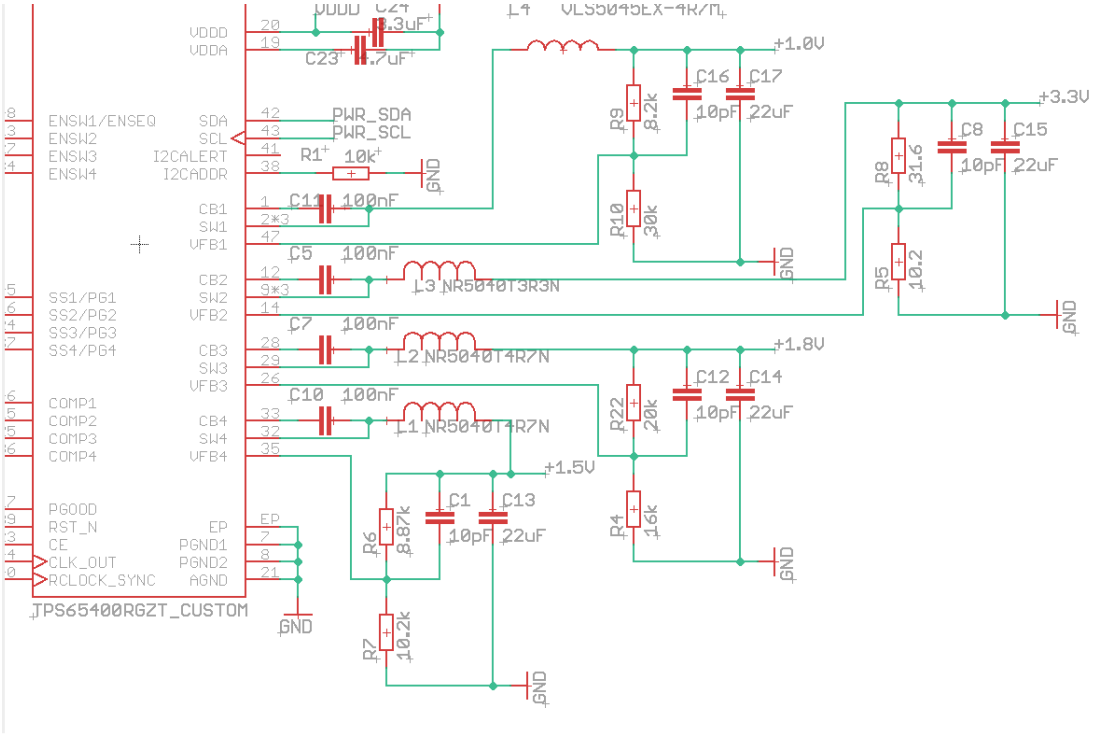
Quelle: Sebastian Paulsen

Abbildung 5.6: Spannungsübersicht

Nach intensiver Recherche und Vergleichen mit anderen Boards wird der TPS-65400 Chip von Texas Instrument ausgewählt. Dieser bietet die Möglichkeit vier unterschiedliche Spannung aus einem Eingangssignal zu generieren[11].

5.7.1 Vierfach Spannungsregler - TI TPS65400

Der TPS-65400 ist eine integrierte Power Management Unit (PMU) mit einer Effizienz von bis zu 95%. Die Eingangsspannung kann von 4,5 V bis 18 V und die Ausgangsspannung kann von 0,6 V bis 90% der Eingangsspannung betragen. Ausgang 1 und 2 unterstützen Stromstärken von bis zu 4 A, Ausgang 3 und 4 bis zu 2 A. [11]



Quelle: Sebastian Paulsen

Abbildung 5.7: Schaltplan Widerstände

Dimensionierung der Widerstände für die Ausgangsspannungen Die Berechnung der Ausgangsspannung mit der Formel

$$V_{out} = V_{ref} \cdot \left(1 + \frac{RFB1}{RFB2}\right)$$

mit der Voreinstellung $V_{ref} = 0{,}8\,\mathrm{V}$. [11]

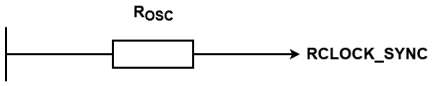
$$1{,}0\,\mathrm{V} \approx 0{,}8\,\mathrm{V} \cdot \left(1 + \frac{8{,}2\,\mathrm{k\Omega}}{30\,\mathrm{k\Omega}}\right)$$

$$1{,}5\,\mathrm{V} \approx 0{,}8\,\mathrm{V} \cdot \left(1 + \frac{8{,}87\,\mathrm{k\Omega}}{10{,}2\,\mathrm{k\Omega}}\right)$$

$$1{,}8\,\mathrm{V} \approx 0{,}8\,\mathrm{V} \cdot \left(1 + \frac{20\,\mathrm{k\Omega}}{16\,\mathrm{k\Omega}}\right)$$

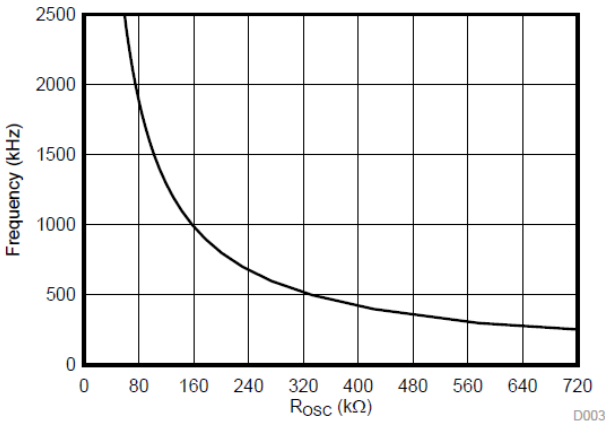
$$3{,}3\,\mathrm{V} \approx 0{,}8\,\mathrm{V} \cdot \left(1 + \frac{31{,}6\,\Omega}{10{,}2\,\mathrm{k}\Omega}\right)$$

Beschaltung Die Frequenz des Schaltreglers lässt sich über einen Widerstand wählen, der in der Grafik 5.8a beschrieben wird. Mit einem Widerstand von 150 kΩ stellt man eine Frequenz von einen MHz ein 5.8b.



Quelle: Sebastian Paulsen

(a) Pin Belegung



Quelle: [S.19, 11]

(b) PWM Wechsel Frequenz abhängig vom Widerstand an Pin RCLOCK_SYNC

Quelle: Sebastian Paulsen

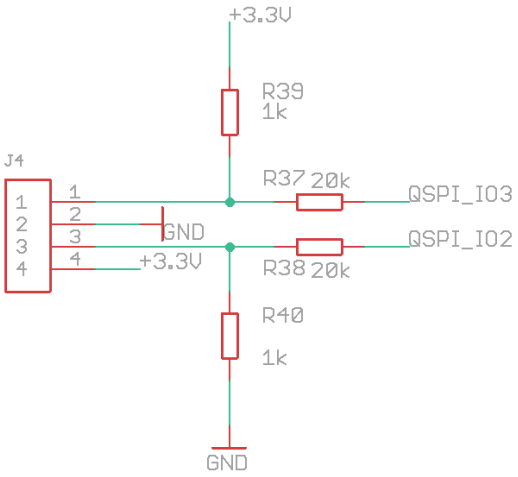
5.8 Peripherie

Weitere Anschlüsse, die an den SoC angeschlossen werden, um Funktionen des SoCs von außen zugreifbar zu machen, werden in dem folgenden Kapitel weiter erläutert.

5.8.1 Steckbrücken und Stiftleiste

Um das Bootmedium des SoCs auszuwählen muss diesem vor dem Start des Bootloaders bereits mitgeteilt werden, welches Bootmedium verwendet werden soll. Dafür bietet es

sich an Eingänge des Socs so zu schalten, dass der Bootloader, durch den Zustand der Eingänge, das Bootmedium auswählen kann. Die drei unterschiedlichen Bootmedien belaufen sich auf Flashspeicher, SD-Karte und die Programmierung via JTAG. Für diese drei unterschiedlichen Bootmedien werden folglich zwei Eingänge des SoCs benötigt, da sich mit diesen binär bis vier zählen lässt. Eine einfache Möglichkeit ist die Benutzung einer Stiftleiste mit vier Pins, die drei unterschiedliche Steckbrückenstellungen ermöglicht.



Quelle: Sebastian Paulsen

Abbildung 5.9: Pin Belegung Stiftleiste

Die Positionen der zweipoligen Steckbrücke befinden sich also Oben, Mitte und Unten, darauß ergibt sich folgende Tabelle 5.2.

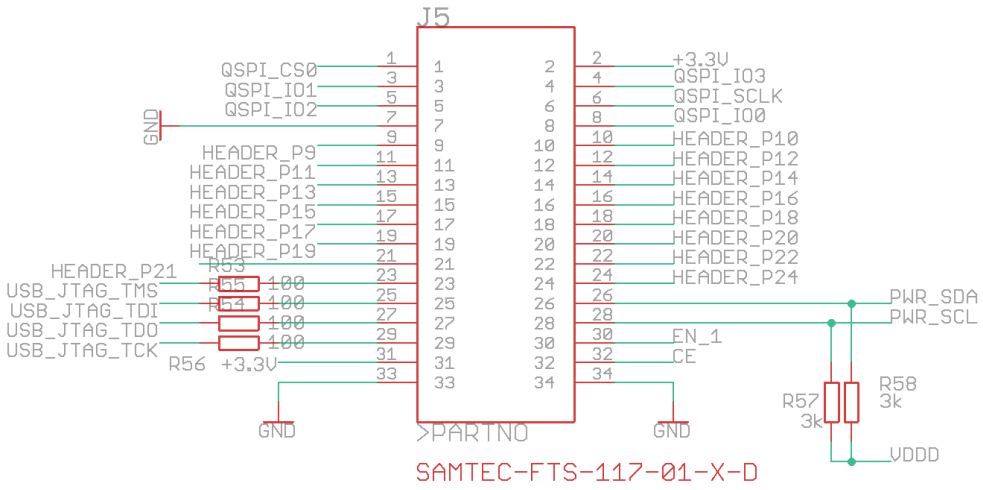
Stellung	QSPI_IO3	QSPI_IO2
Oben	0 V	3,3 V
Mitte	3,3 V	0 V
Unten	3,3 V	3,3 V

Tabelle 5.2: Stellungen der Steckbrücke

Die Abfrage der Pins 5.2 ist im First Stage Bootloader(FSBL) implementiert. [Vgl. 24]

5.8.2 I/O Header

Ein I/O Header 5.10 bietet die Möglichkeit Peripherieanschlüsse leicht erreichbar nach außen zu legen. Die Pins des Flashspeichers, der JTAG-Schnittstelle und des I2C Zugangs des TPS 65400 wird, neben einiger MIO Pins des SoCs, für den externen Zugriff auf den Header geroutet, da es nötig sein könnte auf diese zugreifen zu können.



Quelle: Sebastian Paulsen

Abbildung 5.10: Pin Belegung I/O Header

Damit ist der Schaltplan fertig gestellt und es wird der Platinenentwurf im Folgenden umgesetzt.

5.9 Layout

In diesem Kapitel wird die zweite Komponente von Eagle benutzt. Eagle ermöglicht die Umsetzung von Schaltplänen mithilfe eines integrierten Tools für das Platinen-Layout. Dabei werden im ersten Schritt die Anzahl der Schichten der Platine grob geschätzt und festgelegt. Die verwendeten Bauteile werden daneben platziert und mit gelben Linien verbunden. Diese gelben Linien sind sogenannte Airwires und geben an, welche Komponenten miteinander verbunden werden sollen. Ziel ist es dabei, die Komponenten auf der Platine zu platzieren und die Airwires durch das Verlegen von Leiterbahnen zu entfernen.

Zum Einhalten gewisser Layout-Regeln wie beispielsweise dem Mindestabstand zwischen den Leiterbahnen, haben die meisten Platinenhersteller sogenannte Design Rule Check (DRC)-Dateien. Diese können in Eagle eingebunden werden und ermöglichen es, die bisherige Platine nach den Voraussetungen des Platinenherstellers zu analysieren. Anhand dieser Bewertung kann man mögliche Fehler und Begrenzungen erkennen und beheben.

5.9.1 Umsetzung des Layouts

Nachdem die Schaltpläne fertiggestellt sind, werden die Abmaße der Platine mit den Maximalwerten von 75 mm Länge, 35 mm Breite und 20 mm Tiefe in Eagle eingetragen.

Lagenanzahl Die Lagenanzahl wird nach einer anfänglichen Platz-Abschätzung auf Sechs Lagen festgelegt. Eine weitere Erhöhung auf zum Beispiel Acht Lagen ist bei Bedarf weiterhin möglich. Es wird wie folgt weiter vorgegangen:

1. Platzierungsraster einstellen
2. Bauelemente mit vorbestimmten Positionen platzieren (Stecker, Bedienelemente)
3. Bauelemente sortieren
   - nach den Baugruppen
   - nach Oberseite/Unterseite Leiterplatte
4. Bauelemente mit hoher Priorität platzieren(Störung/Verlustleistung)
5. Stützkondensatoren neben Schaltkreise legen
6. restliche Bauelemente nach Platzierhinweisen[Vgl. S.92, 26]

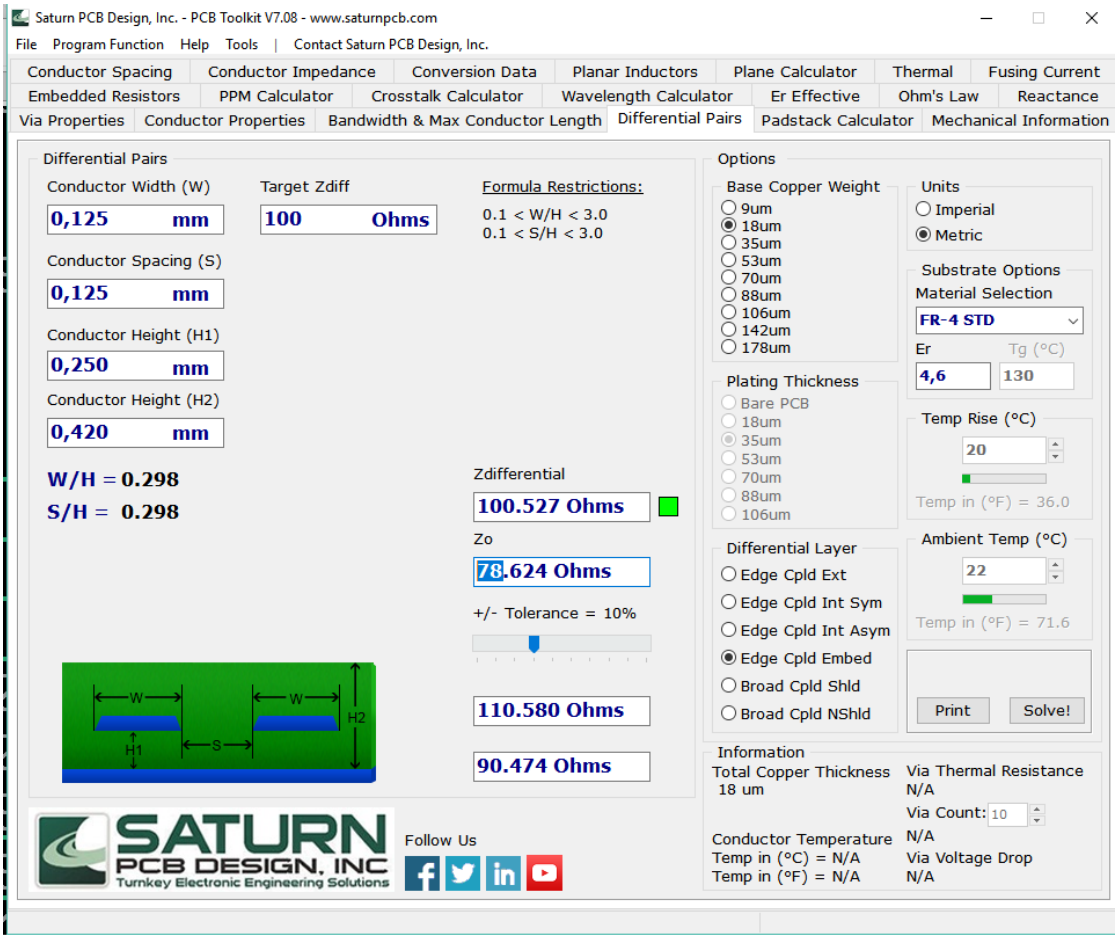
Als nächstes werden die Bauteile so rotiert, dass die Airwires sich möglichst nicht kreuzen und sich möglichst kurze Strecken ergeben. Dadurch wird versucht, die Anzahl der sich kreuzenden Signalleitungen gering zu halten.

5.9.2 DDR

Der DDR-Baustein ist durch den geringen Platz auf der Platine und der großen Anzahl von Signalen am kompliziertesten an den SoC anzuschließen. Er benötigt möglichst kurze Wege zum FPGA und besitzt ein differentielles Signalpaar. Da der Platz für die Leitung nicht ausreicht, wird die Platine auf acht Lagen erweitert.

5.9.3 Layout von differentiellen Signalen

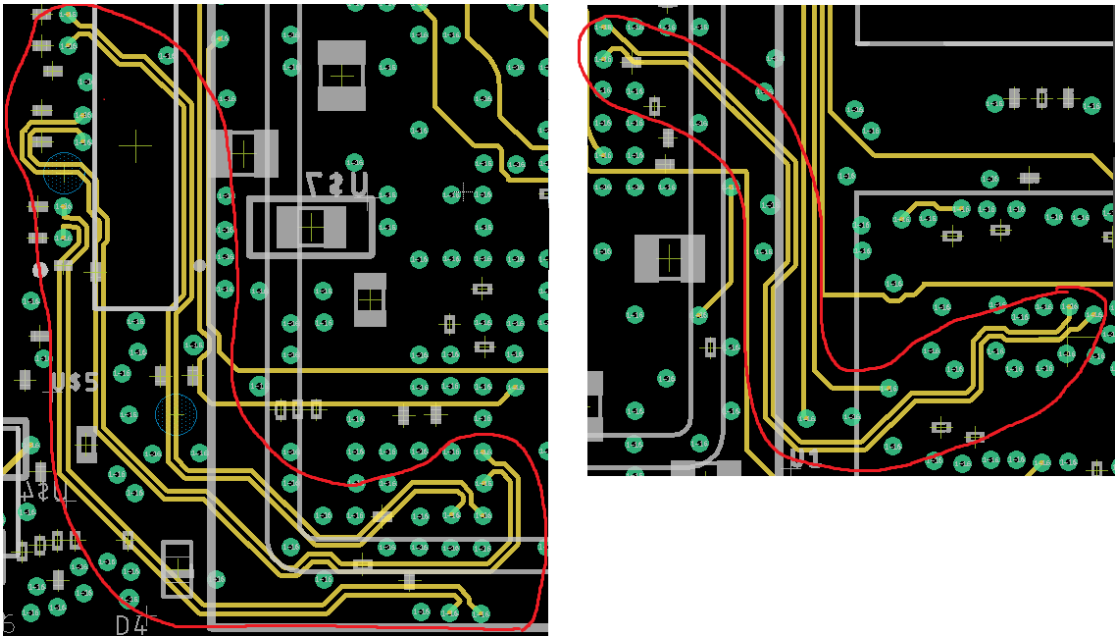
Die Anzahl der differentiellen Signale beläuft sich auf drei auf dieser Platine, eine für die DQS Leitungen im DDR und zwei im D-PHY-Interface. Um diese Signale zu routen müssen Berechnungen der Impedanzen durchgeführt werden. Dafür ist die Software PCB Toolkit in der Version 7.08 von der Firma Saturn PCB Design, Inc. hilfreich. Es verwendet die Formeln, welche im Kapitel Grundlagen für die Impedanzberechnung beschrieben werden. Diese ermöglicht es durch Variation der Werte vergleichbar zeiteffizient die richtigen Impedanzen zu ermitteln, siehe Grafik 5.11.



Quelle: Sebastian Paulsen

Abbildung 5.11: Aufnahme von PCB Toolkit

Aus den Berechnungen wird ersichtlich, dass die differentiellen Signale für eine Impedanz von 100 Ω in der zweiten Schicht bei einem Leiterbahnabstand von 125 µm, einer Höhe von 250 µm und einem Abstand von 420 µm zur Leiterplatten darüber verlegt werden müssen. Da diese Leitungen durch ihre besondere Geometrie Priorität haben, werden andere Leitungen entfernt und anders platziert oder verschoben.

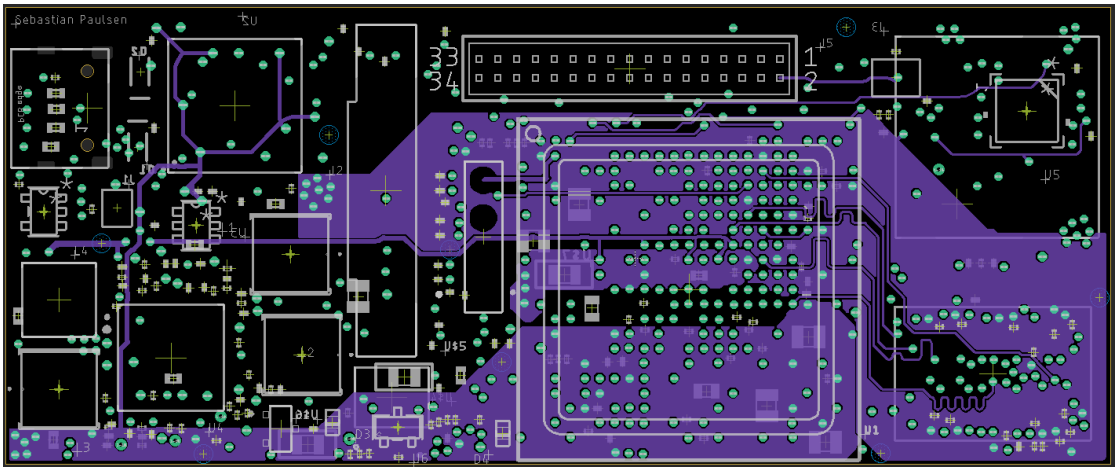


Quelle: Sebastian Paulsen

Abbildung 5.12: Verlegung differentielle Signale

5.9.4 Powerschichten

Dadurch, dass diverse Bauteile auf der Platinen die gleiche Spannung benötigen, lassen sich diese zusammenfassen und mit großflächigen Leiterbahnen verbinden. Der Vorteil an großflächigen Kupferflächen ist der geringe Widerstand im Vergleich zu schmalen. Dadurch ergibt sich eine geringere Erwärmung bei höheren Strömen. Es wird für jede der vier Spannungen (1,0 V;1,5 V;1,8 V;3,3 V) ein großflächiges Polygon verwendet.



Quelle: Sebastian Paulsen

Abbildung 5.13: Struktur der Versorgungsleitungen

6 Inbetriebname und Evaluation der Hardware

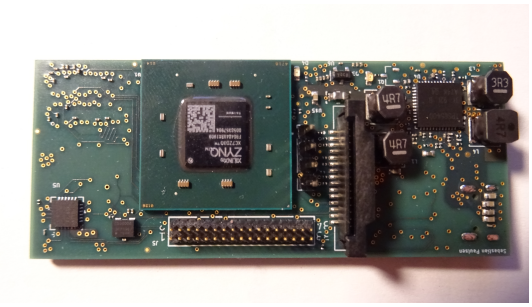
Nachdem zwei Boards nach dem entwickelten Schaltplan von einem Bestücker und einem Fertiger gefertigt worden sind, werden diese nun auf ihre Funktion untersucht. In den folgenden Schritten wird beim Umgang mit den elektrischen Bauteilen auf die elektrostatische Entladung geachtet, damit keine Bauteile beschädigt werden.

6.1 Elektrische Überprüfung

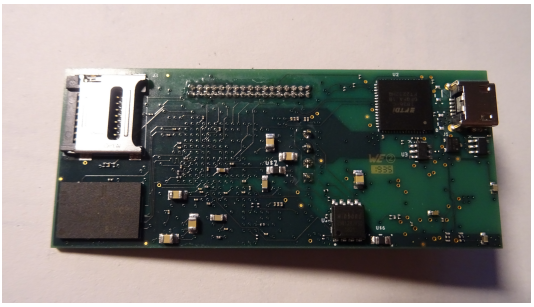
Bevor das System in einer höheren Abstraktionsebene getestet werden kann, müssen erst die einzelnen elektrischen Komponenten untersucht werden.

6.1.1 Optische Inspektion

Im ersten Schritt wird eine optische Inspektion durchgeführt. Diese soll etwaige Probleme die beim Lötvorgang entstanden sein können frühzeitig erkennen. Dadurch soll verhindert werden, dass bei der Inbetriebnahme eine Beschädigung einzelner Komponenten oder der gesamten Platine entstehen.

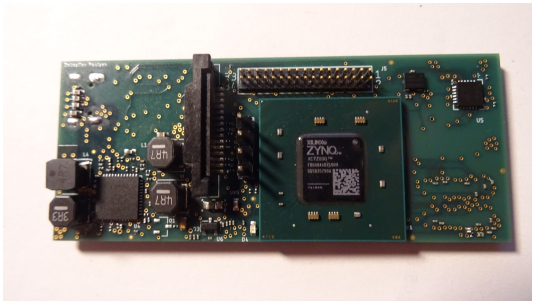


(a) Platine 1 oben

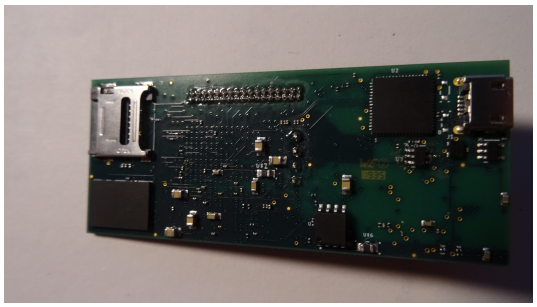


(b) Platine 1 unten

Quelle: Sebastian Paulsen

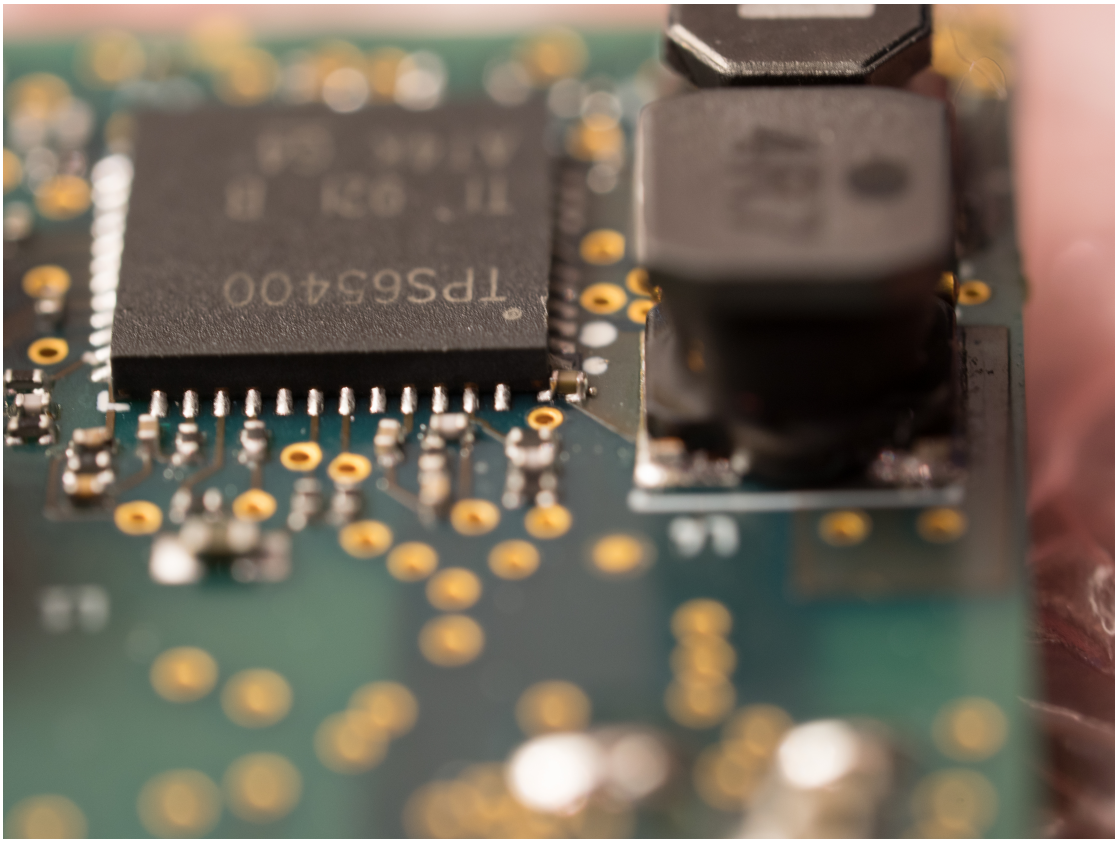


(a) Platine 2 oben



(b) Platine 2 unten

Quelle: Sebastian Paulsen



Quelle: Sebastian Paulsen

Abbildung 6.3: Spannungsregler Kondensator erstes Board

Ergebnis der optischen Inspektion Beim ersten Board ist ein Kondensator unter dem Spannungsregler auffällig, dies ist im Bild 6.3 zu erkennen. Das zweite Board hat keine weiteren Auffälligkeiten.

6.1.2 Durchgangsprüfung von Kondensatoren

Im zweiten Schritt werden die Kondensatoren hinsichtlich des Durchgangs überprüft. Dafür werden die Kondensatoren mit einem Ohm-Meter auf Durchgang geprüft. Dann aber auch bitte noch den Hinweis: Da die Bauteile bereits auf die Platine in eine Schaltung gelötet sind, kann diese einen gewissen Einfluss auf das Messergebnis haben. Wenn dieser Widerstand sehr gering ist, deutet dies auf eine mögliche mechanische Beschädigung der Kondensatoren hin.

Ergebnis der Durchgangsprüfung Es konnte kein Kondensator gefunden werden, dessen Widerstand sehr gering ist.

6.1.3 Stromzufuhr

Im dritten Schritt der Inbetriebnahme wird das Board mit Strom versorgt, dabei ist zu beachten, dass das Board nicht direkt die volle Eingangsspannung erhält, damit eventuelle Fehler auf der Platine nicht direkt einen Kurzschluss verursachen. Die definierte Eingangsspannung ist 5 V. Die Spannung wird langsam von 0 V auf 5 V in 0,5 V-Schritten erhöht, dabei wird der Strom auf 300 mA, wegen der Gefahr des Kurzschlusses begrenzt. Besonderes Augenmerk wird dabei auf die Stromaufnahme gelegt. Sollte die Stromaufnahme die 300 mA-Grenze übersteigen, ist davon auszugehen, dass ein Fehler in Form eines Kurschlusses vorliegt. Der Spannungsregler ist durch ein Enable Signal geschaltet, sobald dieses 1,2 V übersteigt, wird der Spannungsregler aktiv.

Während der schrittweisen Erhöhung des Stromes wird die Temperatur der Platine, die Spannung hinter der Spule L3 (siehe Schaltplan im Anhang) und die Spannung des Enable Signals gemessen. Für die Messungen des Stromes wird die Anzeige des digitalen Netzteils (Keysight E36313A) abgelesen. Für die Spannungen des Enable Signals und der Spule wird ein Oszilloskop (Tektronix TDS 2024C) verwendet. Die Temperatur wird mit Hilfe eines Infrarot-Thermometers gemessen.

Für das erste Board sehen die Werte wie in der Tabelle 6.1 dargestellt aus.

Spannungs-begrenzung	**Strom-begrenzung in A**	**Strom Eingang in mA**	**Spannung Enable Signal**	**Spannung 3,3 V Ausgang**	Temperatur
0,1	0,3	14	0,05	0	22
0,5	0,3	28	0,178	0	23
1	0,3	55	0,4	0	24
1,5	0,3	75	0,6	0	28
2	0,3	100	0,7	0	30
2,5	0,3	148	750	0	31
3	0,3	183	867	0	34
3,5	0,3	204	933	0	38
4	0,3	228	1	0	44
4,5	0,3	450	1,41	3,35	62

Tabelle 6.1: Messung am Board 1

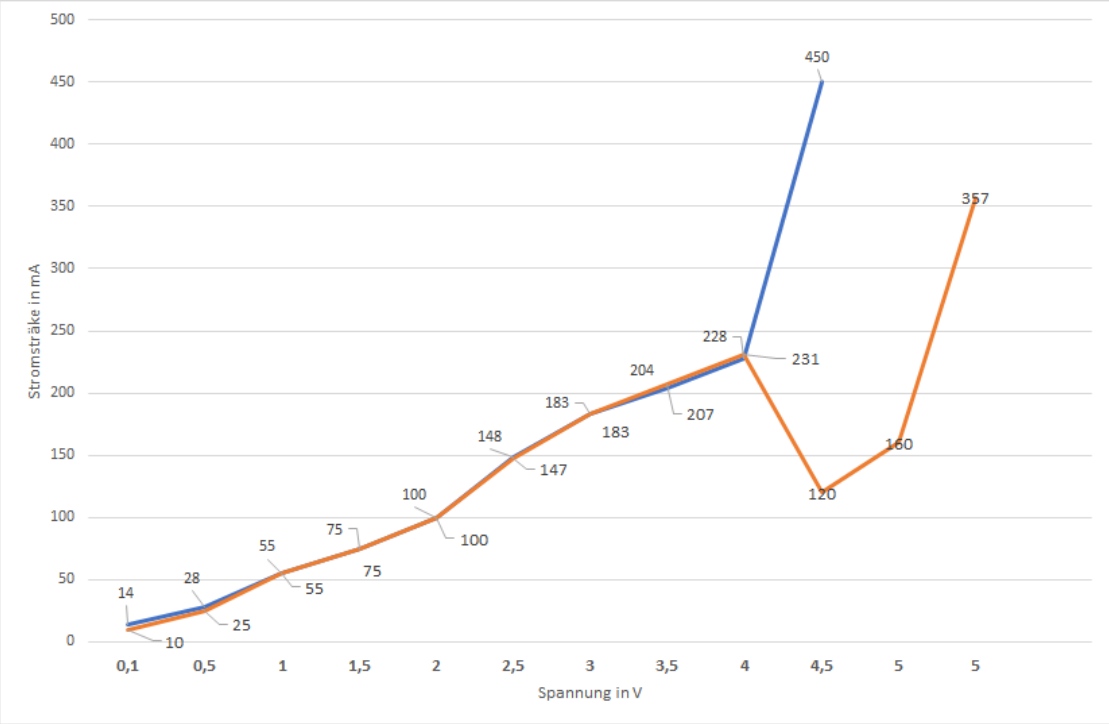
Die Messungen wird ab 4,5 V abgebrochen, da der Strom beim Übertritt des Schwellwertes für das Enable Signal von 1,12-1,28 V einen erheblichen Sprung macht und zugleich die Temperatur stark stieg.

Für das zweite Board wurde die gleichen Messungen getätigt und in der Tabelle 6.2 zusammengefasst.

Spannungs-begrenzung	**Strom-begrenzung in A**	**Strom Eingang in mA**	**Spannung Enable Signal**	**Spannung 3,3 V Ausgang**	Temperatur
0,1	0,3	10	40	0	22
0,5	0,3	25	181	0	23
1	0,3	55	340	0	24
1,5	0,3	75	520	0	28
2	0,3	100	640	0	30
2,5	0,3	147	753	0	31
3	0,3	183	850	0	31
3,5	0,3	207	929	0	34
4	0,3	231	981	0	35
4,5	0,3	120	1510	1,76	40
5	0,3	160	1,93	1,62	45
5	0,5	357	1,67	3,3	54

Tabelle 6.2: Messung am Board 2

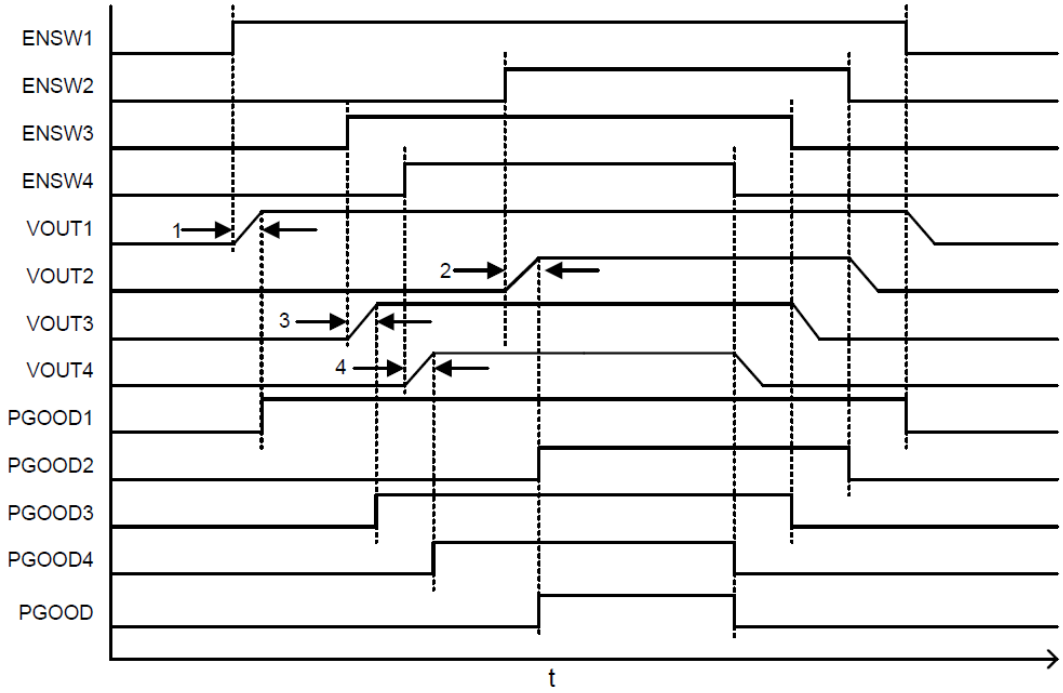
Beim zweiten Board sehen die gemessenen Spannungen und Stromstärken so aus wie es geplant wurde. In der Folgenden Abbildung 6.4 ist eine grafische Gegenüberstellung der Spannungen und Ströme beider Platinen dargestellt.



Quelle: Sebastian Paulsen

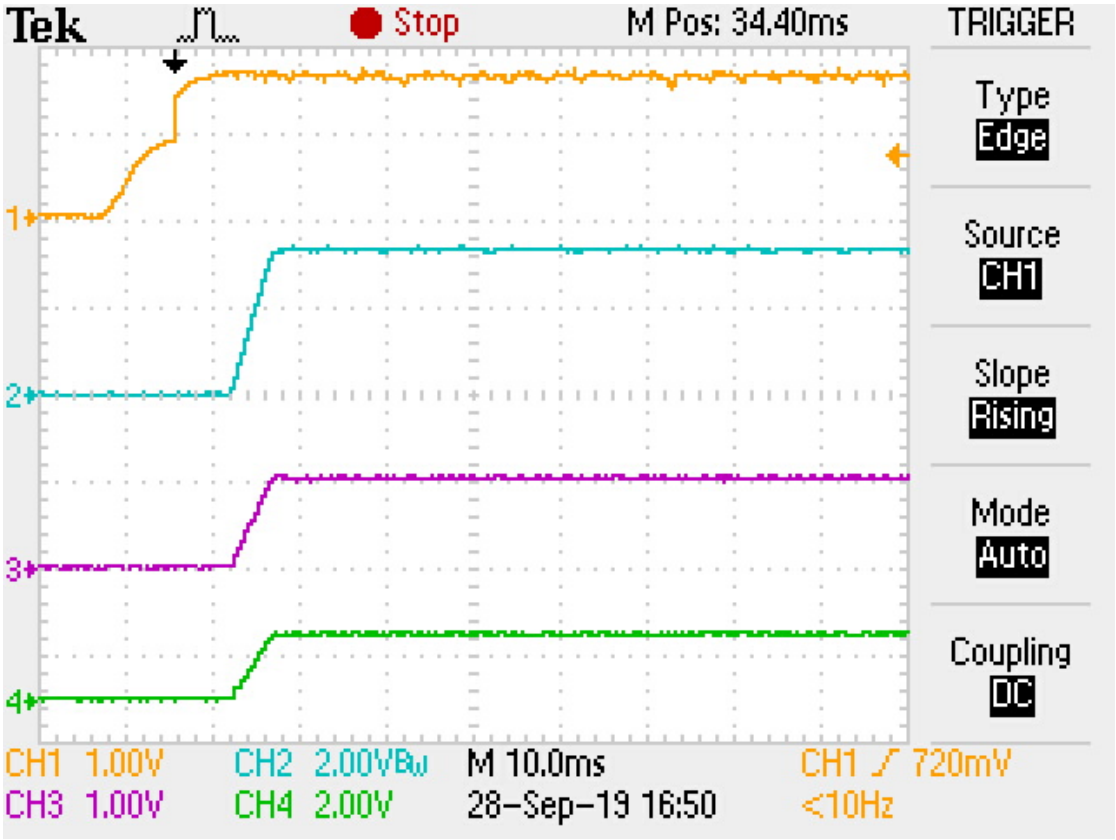
Abbildung 6.4: Vergleich Stromverlauf zu Spannung

Es wird eine weitere Messung mit dem zweiten Board durchgeführt, um die Ausgangsspannungen der Spulen L1, L2, L3 und L4 (siehe Schaltplan im Anhang) zu verifizieren. In der Grafik 6.5 im Datenblatt werden die EN Signale zu unterschiedlichen Zeichen getriggert wohin gegen die Grafik6.6 zeigt, dass wie erwartet alle Ausgangsspannungen gleichzeitig anlaufen da EN1=EN2=EN3=EN4.



Quelle: Sebastian Paulsen

Abbildung 6.5: Vorgabe Datenblatt TPS65400



Quelle: Sebastian Paulsen

Abbildung 6.6: Messung von oben nach unten EN1, VOUT1, VOUT2, VOUT3

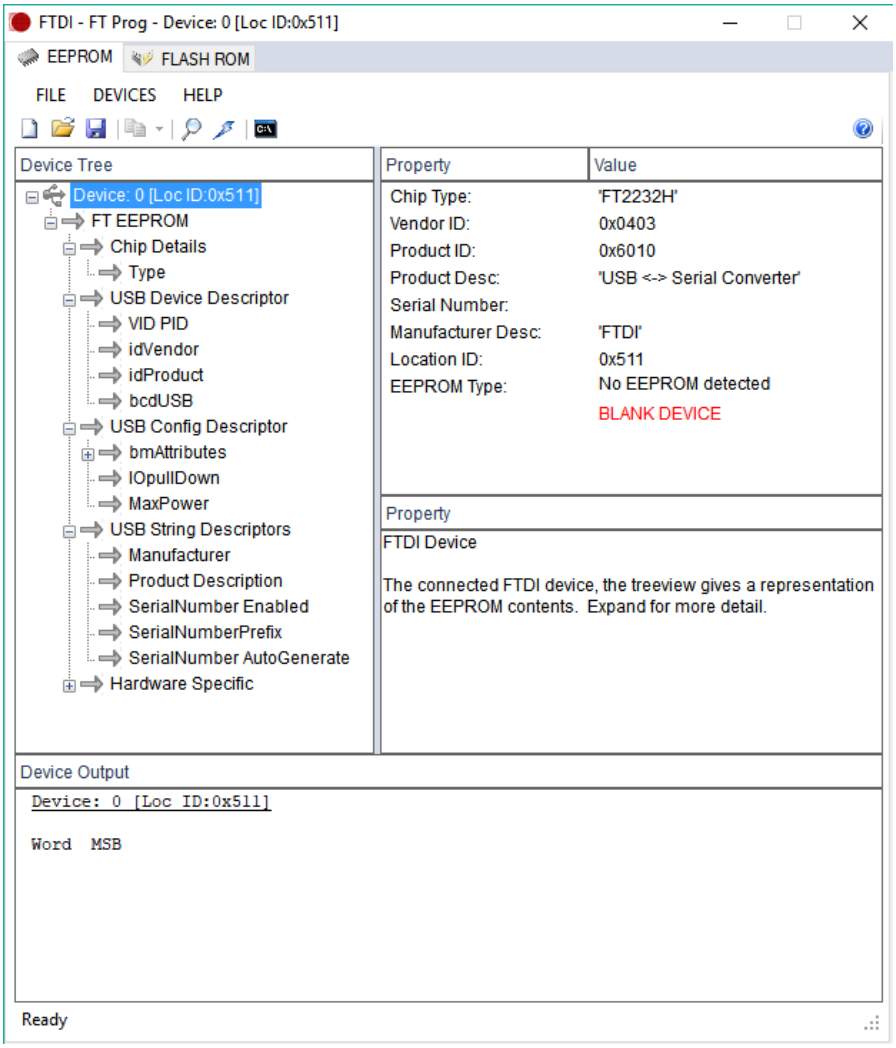
Die Ausgangsspannung VOUT4 beträgt nach Messungen 3,3 V.

Ergebnis der Stromzufuhr Die Ausgangsspannungen des Spannungsreglers entsprechen der berechneten und erwünschten Spannung.

6.2 Funktionsprüfung der Peripherie

Nach dem die internen Spannungen überprüft worden sind, wird die Programmierung des SoCs evaluiert. Der erste Schritt besteht darin, die Platine an einen Computer anzuschließen. Es werden die FTDI-Treiber auf dem Computer installiert, daraufhin taucht der Chip als Anzeige von zwei serielle Ports im Gerätemanager des Windows PCs auf. Damit die JTAG Schnittstelle benutzt werden kann, muss der FTDI-Chip konfiguriert

werden. Dies geschieht durch das Programm FT-Prog, welches auf der Website des Chipherstellers verfügbar ist. FT-Prog ermöglicht den EEPROM, welcher an den FTDI-Chip angeschlossen ist, zu beschreiben. Der FTDI Chip läd seine Konfiguration bei jedem Neustart aus diesem EEPROM und ist sonst nur für die RS232 Schnittstelle konfiguriert. Nach dem Start des Programms und nachdem 'Scan and Parse' ausgeführt wurde, wird der Chip angezeigt. Es fällt auf, dass kein EEPROM erkannt wird. Um folglich Lötprobleme auszuschließen, werden die Signale einzeln überprüft.

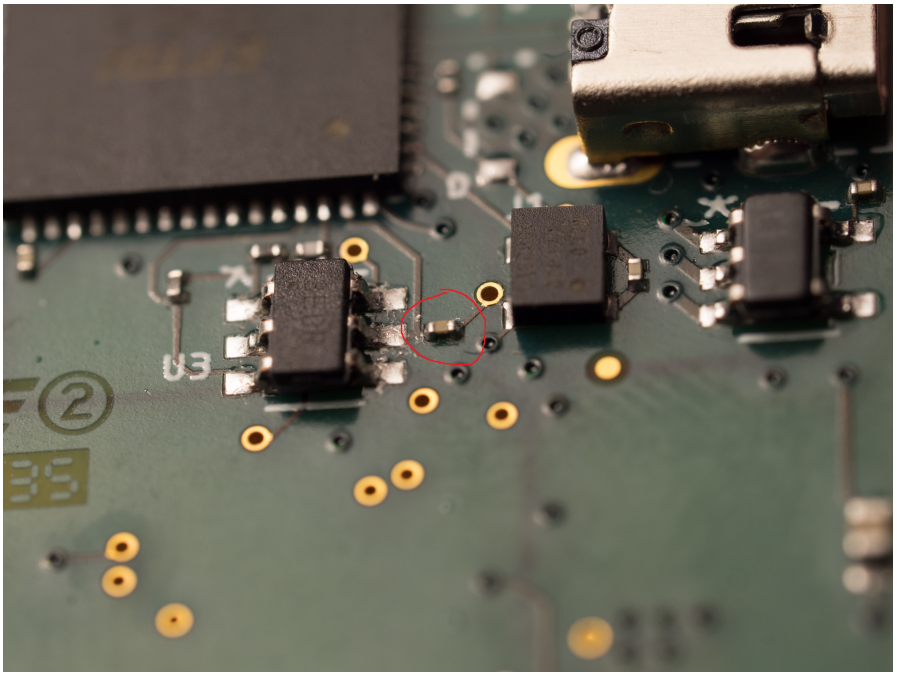


Quelle: Sebastian Paulsen

Abbildung 6.7: EEPROM nicht erkannt in FT-Prog

Hierbei fällt auf, dass ein Kondensator vom Bestücker an einer Stelle verlötet worden ist,

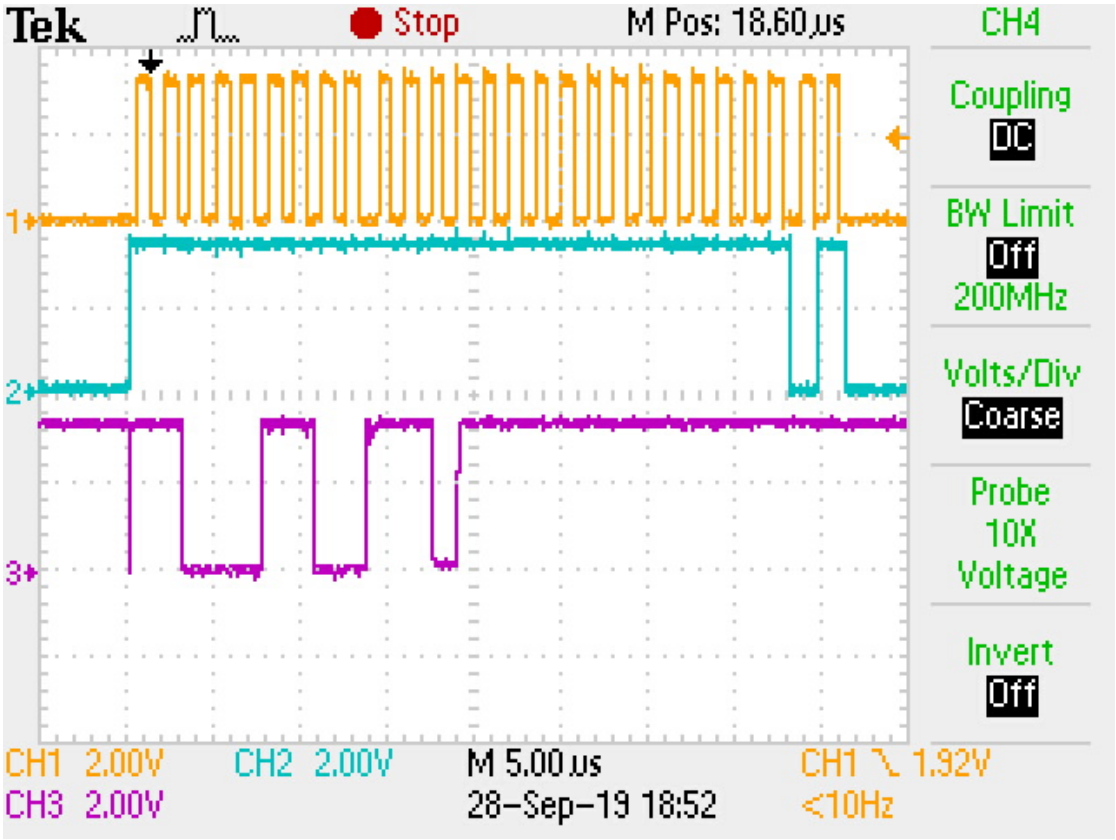
wo im Schaltplan ein Widerstand verzeichnet ist siehe Bild 6.8.



Quelle: Sebastian Paulsen

Abbildung 6.8: Kondensator an Stelle eine Widerstandes eingelötet

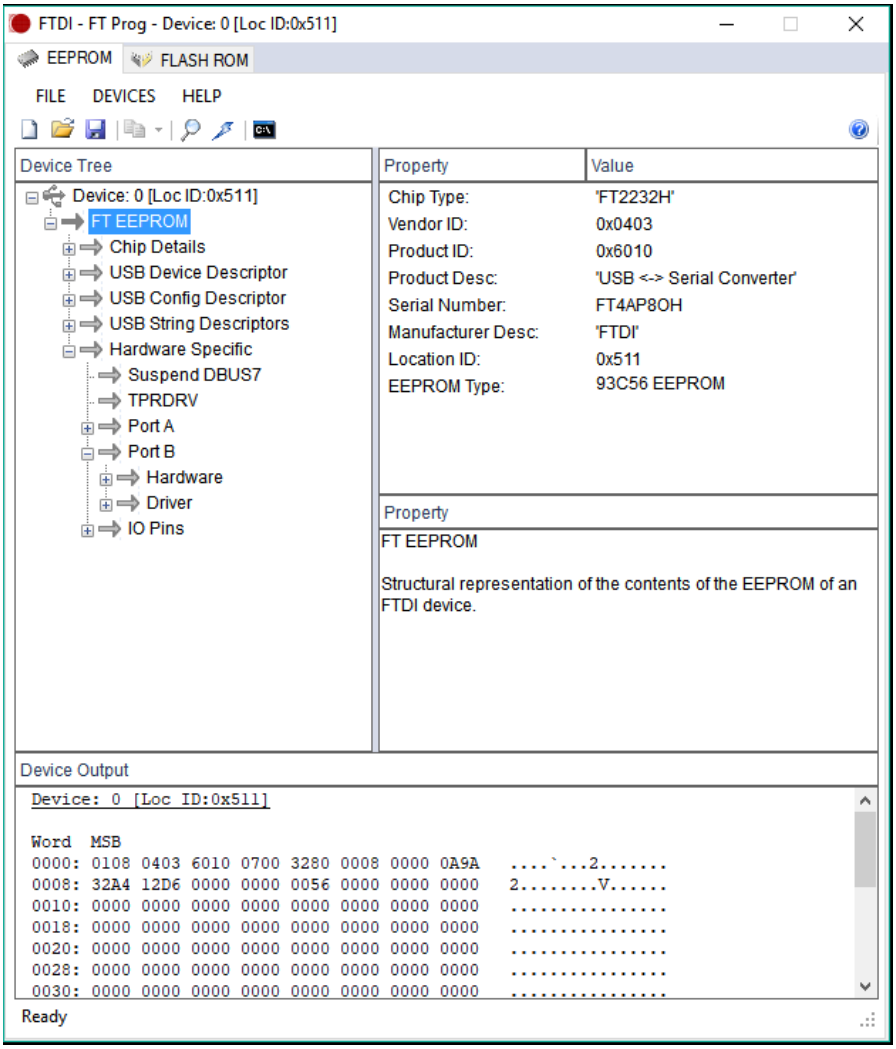
Der Kondensator wird ausgelötet und mit dem richtigen Widerstand ersetzt. Nachdem der Kondensator ausgetauscht ist, wird erneut kein EEPROM im FT-Prog erkannt. Deshalb werden die EEPROM-Signale Clock, Data, und Chip Select mit dem Oszilloskop während des 'Scan and Parse'-Vorgangs gemessen.



Quelle: Sebastian Paulsen

Abbildung 6.9: EEPROM Kommunikation

Es zeigt sich daher, dass eine Verbindung vorhanden ist, doch der EEPROM wird trotzdem nicht erkannt. An dieser Stelle werden die Datenblätter und die darin enthaltenen Schaltungen überprüft. Die Schaltungen entsprechen den Beispielschaltungen im Datenblatt des FTDI-Chips. Bei weiterer Recherche fällt auf, dass die Breite des verwendeten EEPROMs in den Datenblättern mit 8 Bit angegeben ist. Im FTDI-Chip werden aber nur EEPROMs mit einer Breite von 16 Bits unterstützt. Anstatt eines 93LC66A hätte ein 93LC66B verwendet werden müssen. Es wird im Folgenden ein 93LC56B bestellt, da die Lieferzeit für diesen geringer ist als für den 93LC66B. Nach dem Austausch des EEPROMs, wird der neu verbaute EEPROM vom Chip erkannt und der EEPROM kann mit der Konfiguration für die JTAG Schnittstelle beschrieben werden. Für weitere Informationen über die Konfiguration siehe [7].



Quelle: Sebastian Paulsen

Abbildung 6.10: EEPROM erkannt in FT-Prog

Die Konfiguration des EEPROMs ist erfolgreich. Nach erneutem Anschließen der Platine an den Computer, werden zwei USB serielle Schnittstellen erkannt. Als nächster Schritt wird versucht den SoC im Hardwaremanager des Vivado 2018.2 zu finden. Vivado ist die offizielle Software von Xilinx, dem Hersteller des verwendeten Chips. Doch der Chip wird nicht erkannt.

Ein anderer Ansatz wird gewählt, ein vorhandener tinyFPGA-JTAG-Programmer wird an den Header der Platine angeschlossen. Doch auch bei diesem JTAG-Programmer

erscheint der SoC nicht in Vivado. Vivado gibt weder Fehlermeldungen aus, noch den Grund warum etwas nicht erkannt wird.

7 Diskussion der Ergebnisse

Die Inbetriebnahme der Platine stellte sich schwieriger als erwartet dar, da eine der gefertigten Platinen einen durch den Bestücker verursachten Lötfehler aufweiste und dieser in einem Kurzschluss resultierte, konnte nur eine Platine verwendet werden. Somit konnten keine Vergleiche zwischen Spannungen an bestimmten Bauteilen zwischen den Platinen gezogen werden. Besonders Fehler in der Bestückung bei der ein Kondensator anstelle eines Widerstandes verlötet wurde, machten die Inbetriebnahme schwieriger, da sie schwer zu finden waren. Dadurch mussten umfangreichere Tests durchgeführt werden. Des Weiteren stellte sich heraus, dass der USB-Chip, welcher dem Benutzer der Platine eine einfache Möglichkeit geben sollte, die Platine zu programmieren, die Anforderung die Platine, ohne externe Hardware in Form von einem JTAG-Programmer programmieren zu können, nicht erfüllen konnte. Dies hat zur Folge, dass ein JTAG-Programmer zum Programmieren der Platine benötigt wird. Auch die Software des SoC-Herstellers Vivado gab keine genaueren Informationen über die Verbindung von dem Computer zu dem SoC an, womit die Fehlersuche erheblich erschwert wurde. Folglich kann im Hinblick auf die Funktion des USB-Chips vermutet werden, dass das Softwaretool Vivado intern eine Prüfung der angeschlossenen Hardware vornimmt, um so nur die eigene JTAG-Hardware zu unterstützen. Doch auch ist es möglich, dass die von Vivado vorausgesetzte JTAG-Hardware bestimmte Kriterien, wie zum Beispiel die Übertragungsfrequenz erfüllen muss, welche die andere hier verwendete JTAG-Hardware nicht erfüllt und deswegen nicht vom Softwaretool erkannt wird.

Es zeigte sich, dass bei der Umsetzung und der Auswahl der Bauteile für die Platine, die Auswahl des SoCs entscheidend war. Die Funktionen des SoCs, wie beispielsweise diverse Controller, haben die Auswahl der Bauteile stark beschränkt. Besonders die Wahl des Kamerainterfaces war durch die enorme Menge an Möglichkeiten diffizil. Darüber hinaus wurde durch die Verwendung des nicht öffentlichen MIPI-Standard als Kamerainterface die Umsetzung des Kamerainterfaces erschwert, weil die Voraussetzungen aus mehreren Dokumenten komplex zusammengesetzt werden musste. Das Layout der Platine war neben den komplexen Anforderungen der differentiellen Signale durch den geringen Platz

und die acht Lagen eine besonders aufwendige Tätigkeit, darüber hinaus war die Eagle Funktion des Autoroutens nicht für diese Platine geeignet.

Dennoch lässt sich sagen, dass die Abmaße der Platine im Rahmen der Platzanforderung geblieben sind, der USB-Chip sich mit dem Computer verbunden hat, der EEPROM sich konfigurieren ließ und die Spannungsversorgung der einzelnen Bauteile funktioniert hat.

Das Konzept der eigenen Platinenentwicklung stellte sich im Vergleich mit den anderen Ansätzen als aufwendigste Variante dar, da kaum Hardwareentwicklungen im Bereich der selbstfahrenden Miniaturfahrzeuge als Referenz öffentlich zugänglich sind und die öffentlichen Dokumente zum Teil bestimmte Schaltungen in Schaltplänen vermutlich absichtlich nicht veröffentlichen, was die unvollständigen Schaltpläne nahe legen. Die Erfüllung der Platzanforderung ist im Hinblick auf die Zielstellung dieser Arbeit besonders hervorzuheben, da dies das stets handlungs- und entscheidungsleitende Ziel war und eine Einschränkung in der Wahl der Ansätze, wie beispielsweise die Verwendung fertiger Hardware, nach sich zog.

In Hinblick auf den Forschungsstand lässt sich sagen, dass die gewonnenen Ergebnisse die Entwicklung der Hardware für ein autonomes Fahrzeug im H0-Standard als möglich einstufen lassen. Dies stellt eine Ergänzung zu der Forschung in autonomen Fahrzeugen mit interner Berechnungshardware dar. Im Vergleich zu bisherigen Entwicklungen sind die Abmessungen der entwickelten Platine sehr kompakt, dadurch kommt es aber auch zu Temperaturen im Normalbetrieb von 55°C, was wiederum eventuell Kühlkörper nötig macht. Nichtsdestotrotz ist die Menge an Funktionen, die auf dieser Platine mit beschränkten Abmessungen untergekommen sind, bezogen auf den verfügbaren Platz als erfolgsversprechend einzustufen. Dadurch sind, im Vergleich zu anderen Ansätzen, mehr Funktionen pro Fläche umsetzbar.

8 Fazit und Ausblick

8.1 Fazit

Ziel der vorliegenden Arbeit war es, eine kompakte Hardware zur Realisierung künstlicher neuronaler Netze im Bereich autonomer Fahrzeuge zu entwickeln. Deshalb wurde der Fragestellung nachgegangen, inwieweit sich eine kompakte Hardware für autonome Fahrzeuge mit künstlichen neuronalen Netzen realisieren lässt. Zunächst lässt sich dazu zusammenfassend sagen, dass die Aufgabenstellung eine Hardware zu entwickeln, die ein autonomes Miniaturfahrzeug realisieren soll, sich in Bezug auf die Grundfunktionen, der Spannungsversorgung und der Verbindung von dem Computer zu dem USB-Chip, erfüllen lies. Dabei zeigte sich jedoch, dass sich die Funktionen des SoCs aufgrund von der nicht zustande gekommenen Verbindung zum Xilinx Softwaretool noch nicht überprüfen ließen. Insbesondere wurde deutlich, dass die Platzanforderungen maßgeblich handlungs- und entscheidungsleitend waren, was die Wahl nach sich zog, eine eigene Platine zu designen, anstatt auf bestehende Hardware zurückzugreifen. Dabei ist zu reflektieren, dass bei den verwendeten Abmaßen im Vergleich zu größeren Abmaßen die Komplexität der Umsetzung zunimmt. Insgesamt lässt sich resümieren, dass die Konzeption der wichtigste Bestandteil dieser Arbeit war, weil in diesem Arbeitsschritt die maßgeblichen Entscheidungen für das Designen der Hardware getroffen worden sind.

8.2 Ausblick

Im Folgenden werden abschließend Möglichkeiten der Weiterentwicklung dieser kompakten Hardware vorgeschlagen. Zunächst lässt sich sagen, dass die bereits funktionierenden Teile der Platine für eine Überarbeitung dieser Platine übernommen werden könnten, wie die Spannungsversorgung und die Kommunikation des USB-Chips mit seriellen Verbindungen. Es wäre in dem Zusammenhang mit der Überarbeitung lohnenswert, vorher

zu untersuchen, inwieweit sich die Funktionen, die noch nicht überprüft werden konnten, mit einem Xilinx JTAG-Programmer überprüfen lassen würden. Somit könnte die Behebung von Fehlern in einer weiteren Version des Boards resultieren.

Eine weitere Möglichkeit, die grundlegende Fähigkeit für die Funktion des SoCs zu überprüfen, wäre es, die Lötstellen des SoCs mit Hilfe einer Röntgeninspektion zu betrachten, um zu gewährleisten, dass beim Lötvorgang keine Fehler erfolgt sind.

Auch könnte in einer weiteren Version des Boards in Betracht gezogen werden, die USB-Verbindung zum Board zu entfernen und direkt die Verbindung vom Computer zum SoC über einen JTAG-Header anzustreben.

Mit erheblichen Aufwand könnte auch die Verbindung zwischen dem USB-Chip und dem Vivado Softwaretool analysiert werden, um herauszufinden, wie die Kommunikation zwischen diesen beiden Komponenten berichtigt werden könnte.

Literaturverzeichnis

[1] Asano, S. ; Maruyama, T. ; Yamaguchi, Y.: Performance Comparison of FPGA, GPU and CPU in Image Processing. In: *2009 International Conference on Field Programmable Logic and Applications*, 2009 (#), S. 126–131

[2] Beierlein, Thomas (Hrsg.) ; Hagenbruch, Olaf (Hrsg.): *Taschenbuch Mikroprozessortechnik: mit 111 Tabellen.* 2., verb. Aufl. München : Fachbuchverl. Leipzig im Hanser-Verl, 2001. – OCLC: 76217006. – ISBN 978-3-446-21686-0

[3] Digilent: *PYNQ Reference Manual.* https://reference.digilentinc.com/_media/reference/programmable-logic/pynq-z1/pynq-rm.pdf

[4] Doulah, Selina: Fukushima wird zum Roboterfriedhof. In: *#https://www.industr.com* (2017), März

[5] Frenzel, Louis E.: *Handbook of Serial Communications Interfaces: A Comprehensive Compendium of Serial Digital Input/Output (I/o) Standards.* Oxford ; Waltham, MA : Newnes, an imprint of Elsevier, 2016. – OCLC: ocn866615409. – ISBN 978-0-12-800629-0

[6] Fuhrmann, Jonas: Implementierung einer Tensor Processing Unit mit dem Fokus auf Embedded Systems und das Internet of Things. In: Unger, Herwig (Hrsg.): *Echtzeit 2019.* Wiesbaden : Springer Fachmedien Wiesbaden, 2019, S. 61–70. – ISBN 978-3-658-27807-6 978-3-658-27808-3

[7] Future Technology Devices International, Austin: User Guide for FTDI FT_PROG Utility. (2016), S. 1–47

[8] GmbH, EvoBus: *Der Neue eCitaro. Technische Information.* o.J.

[9] Horowitz, Paul: *The Art of Electronics.* Third edition. New York, NY : Cambridge University Press, 2015. – ISBN 978-0-521-80926-9

[10] Howard, Johnson ; Graham, Martin: *High Speed Digital Design: A Handbook of Black Magic*. # : Pearson India, Januar 1993. – ISBN 978-81-317-1412-6

[11] Instruments, Texas: *Tps65400 4.5- to 18-V Input Flexible Power Management Uni With PMBus/I2C Interface- Datasheet*. 2018

[12] International, Future Technology D.: FT2232H Dual High Speed USB to Multipurpose UART/FIFO IC. (o.J.), S. 1–69

[13] Moore, G.E.: Cramming More Components Onto Integrated Circuits. In: *Proceedings of the IEEE* 86 (1998), Januar, Nr. 1, S. 82–85. – ISSN 0018-9219, 1558-2256

[14] MYIR tech: *Z-Turn Board*. o.J.. – S. 1-9

[15] NVIDIA: *Press Release: NVIDIA Launches the Worldś First Graphics Processing Unit: GeForce 256*. https://www.nvidia.com/object/IO_20020111_5424.html. August 1999

[16] Rey, Günter D. ; Wender, Karl F.: *Neuronale Netze: eine Einführung in die Grundlagen, Anwendungen und Datenauswertung*. 3., überarbeitete Auflage. Bern : Hogrefe, 2018. – OCLC: 1025317232. – ISBN 978-3-456-85796-1 978-3-456-95796-8

[17] SAE International: *Surface Vehicle Recommended Practice J3016*. Juni 2018

[18] Schenck, Wolfram ; Horst, Michael ; Tiedemann, Tim ; Gaulik, Sergius ; Möller, Ralf: Comparing Parallel Hardware Architectures for Visually Guided Robot Navigation: Comparing Parallel Hardware Architectures for Visually Guided Robot Navigation. In: *Concurrency and Computation: Practice and Experience* 29 (2017), Februar, Nr. 4, S. e3833. – ISSN 15320626

[19] Tanenbaum, Andrew S.: *Computerarchitektur: Strukturen, Konzepte, Grundlagen*. 5. Aufl. München : Pearson Studium, 2006 (Informatik Rechnerarchitektur). – OCLC: 166106382. – ISBN 978-3-8273-7151-5

[20] Thanigasalam, Haran: MIPI Camera Serial Interface Overview. (2015), S. 20

[21] Tian, David: *DeepPiCar — Part 1: How to Build a Deep Learning, Self Driving Robotic Car on a Shoestring Budget*. https://towardsdatascience.com/deeppicar-part-1-102e03c83f2c. Mai 2019

[22] Wang, Zheng: *Self Driving RC Car*. Juli 2015

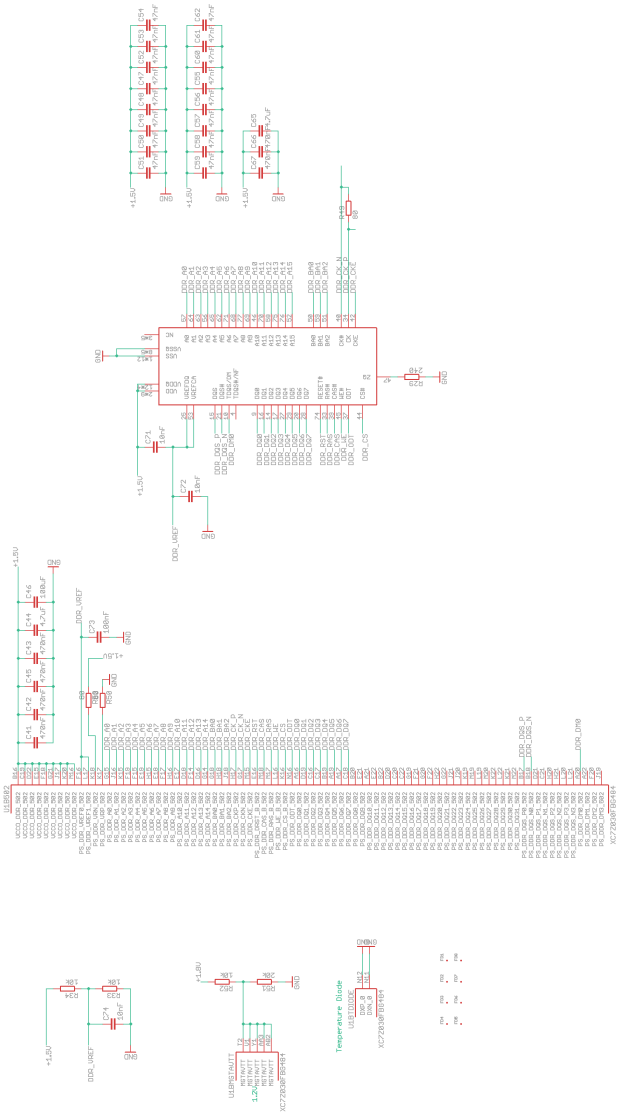
[23] Xilinx: *ZYNQ 7000 Product Selection Guide*. 2014

[24] Xilinx: *Zynq-7000 SoC Technical Reference Manual (UG585)*. 2018

[25] Zakharian, Serge ; Ladewig-Riebler, Patricia ; Thoer, Stefan: *Neuronale Netze für Ingenieure: Arbeits- und Übungsbuch für regelungstechnische Anwendungen*. Braunschweig : Vieweg, 1998 (Vieweg Ausbildung und Studium). – OCLC: 75898355. – ISBN 978-3-528-05578-3

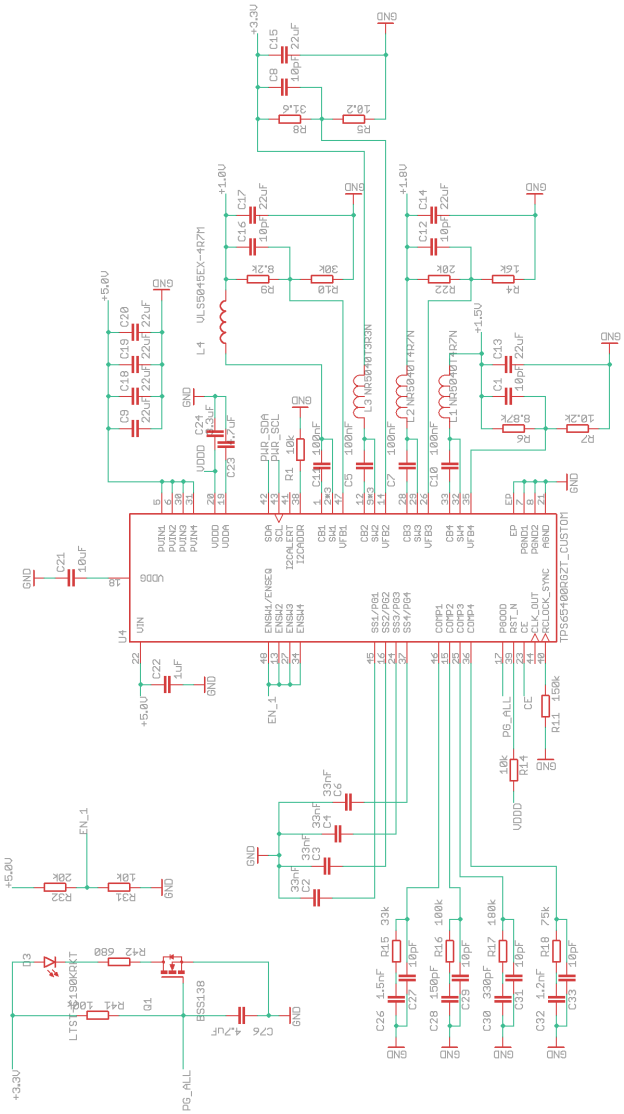
[26] Zickert, Gerald: *Leiterplatten: Stromlaufplan, Layout und Fertigung ; ein Lehrbuch für Einsteiger ; mit 10 Tabellen, 31 Aufgaben und Lösungen*. München : Fachbuchverlag Leipzig im Carl Hanser Verlag, 2015. – OCLC: 898294571. – ISBN 978-3-446-44289-4 978-3-446-44416-4

A Anhang



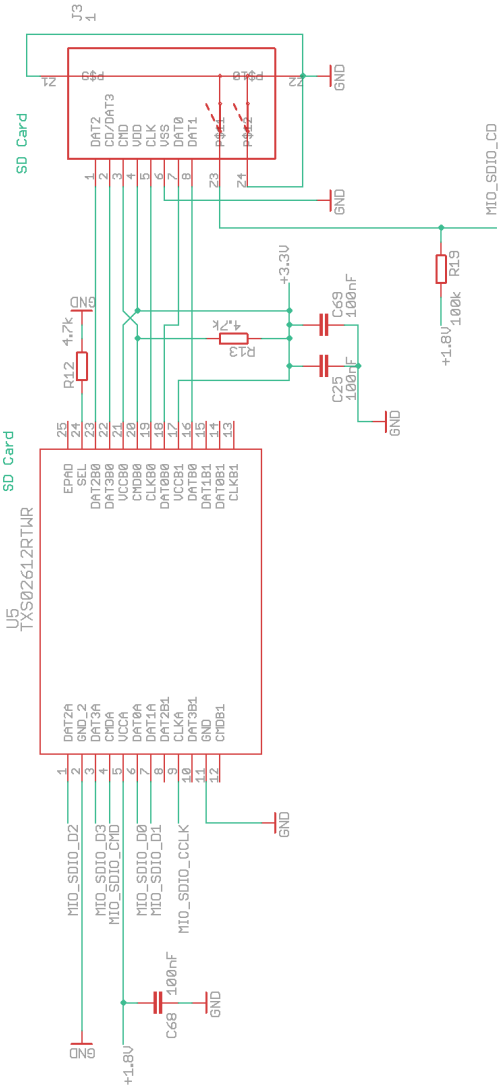
Quelle: Sebastian Paulsen

Abbildung A.1: Schaltplan Seite 1



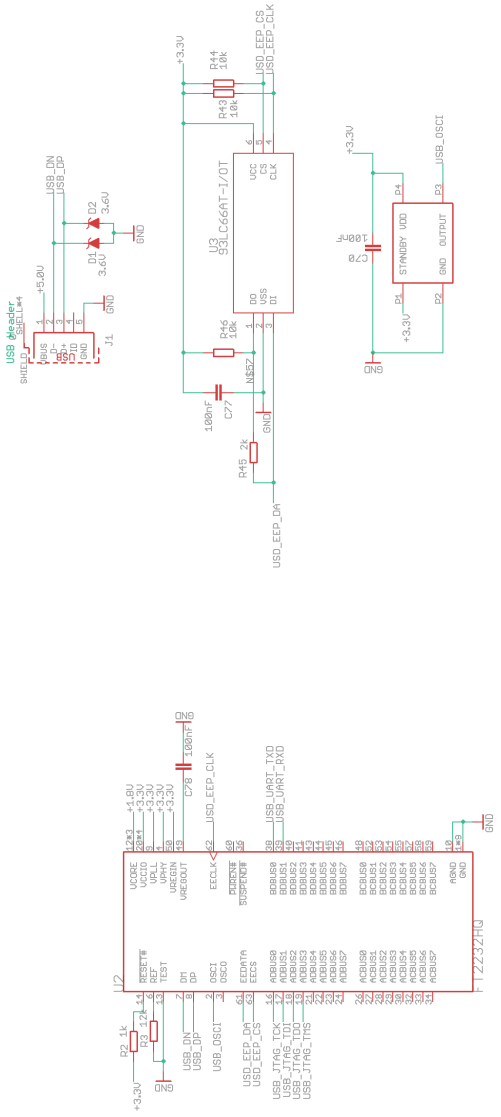
Quelle: Sebastian Paulsen

Abbildung A.2: Schaltplan Seite 2



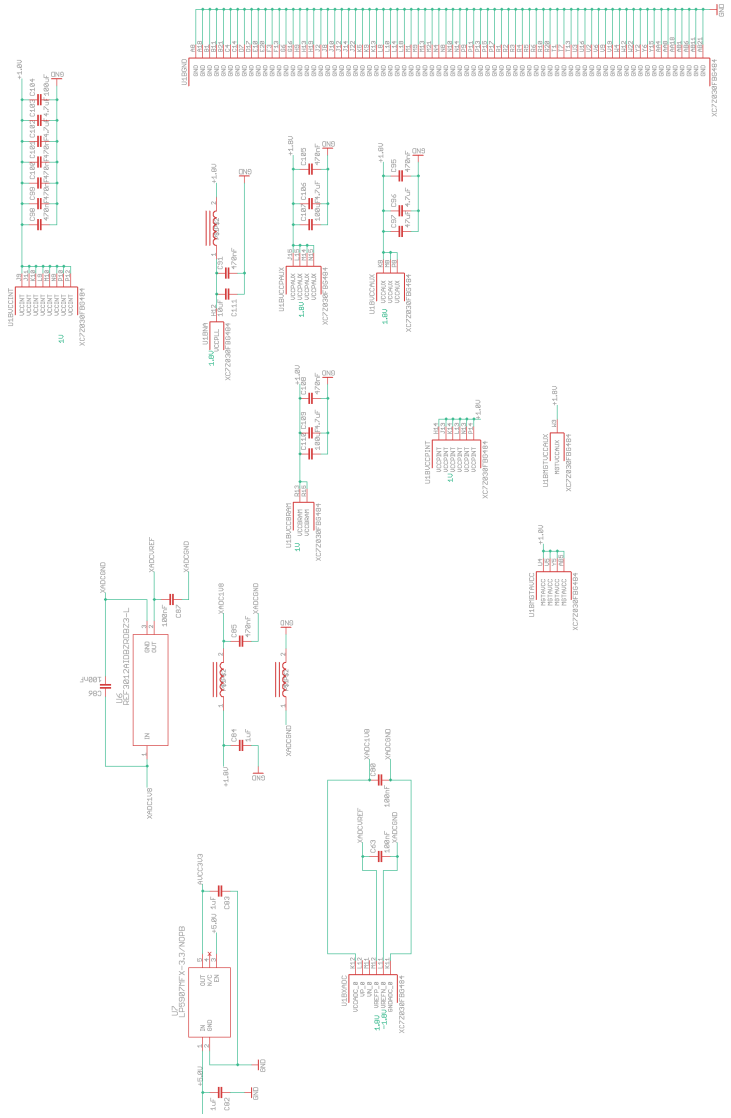
Quelle: Sebastian Paulsen

Abbildung A.3: Schaltplan Seite 3



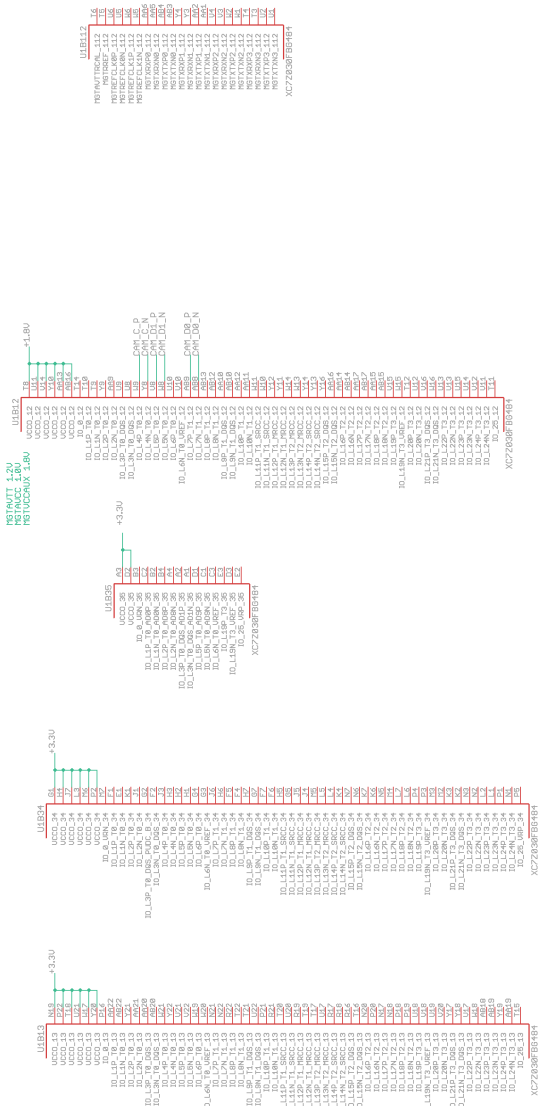
Quelle: Sebastian Paulsen

Abbildung A.4: Schaltplan Seite 4



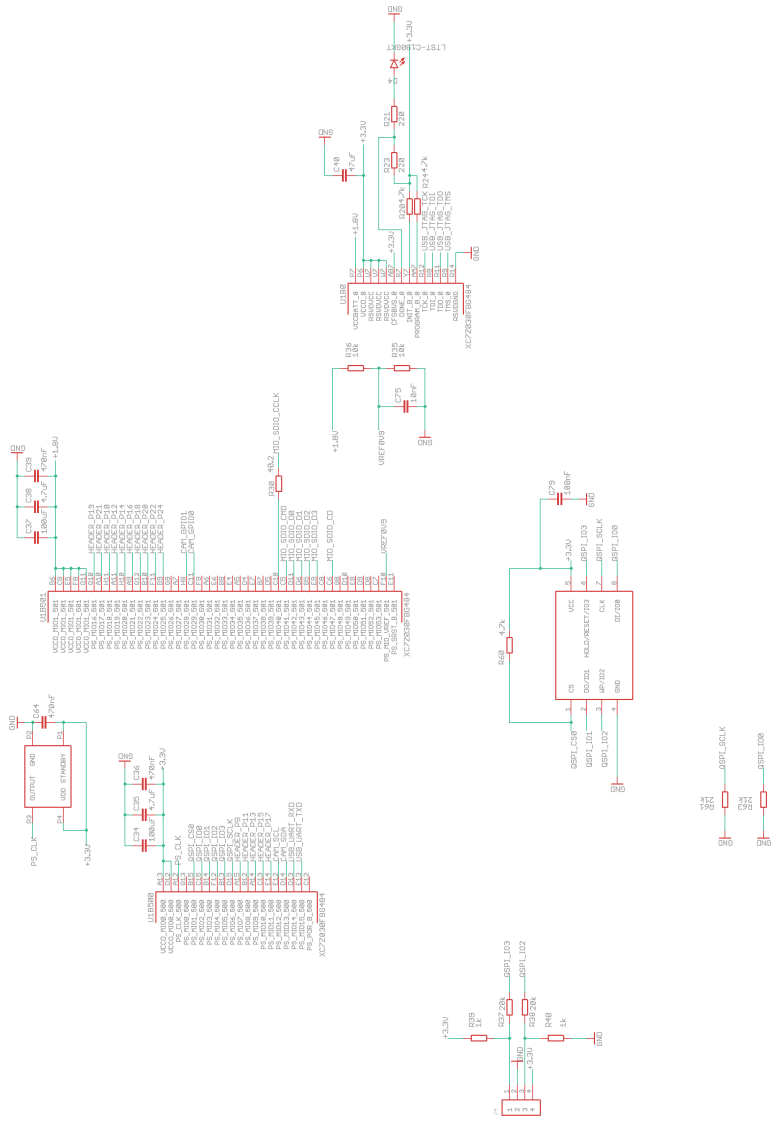
Quelle: Sebastian Paulsen

Abbildung A.5: Schaltplan Seite 5



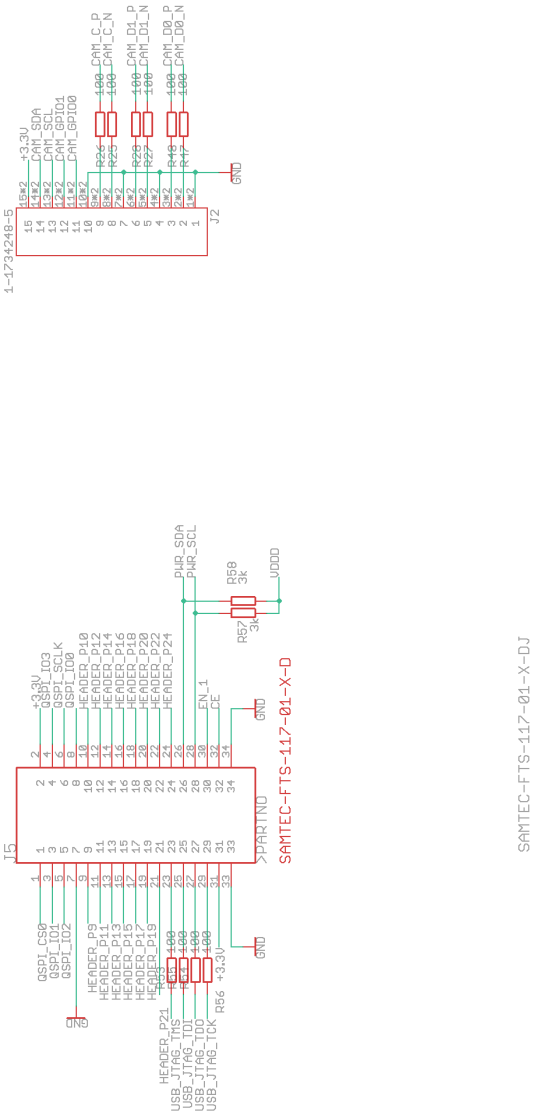
Quelle: Sebastian Paulsen

Abbildung A.6: Schaltplan Seite 6



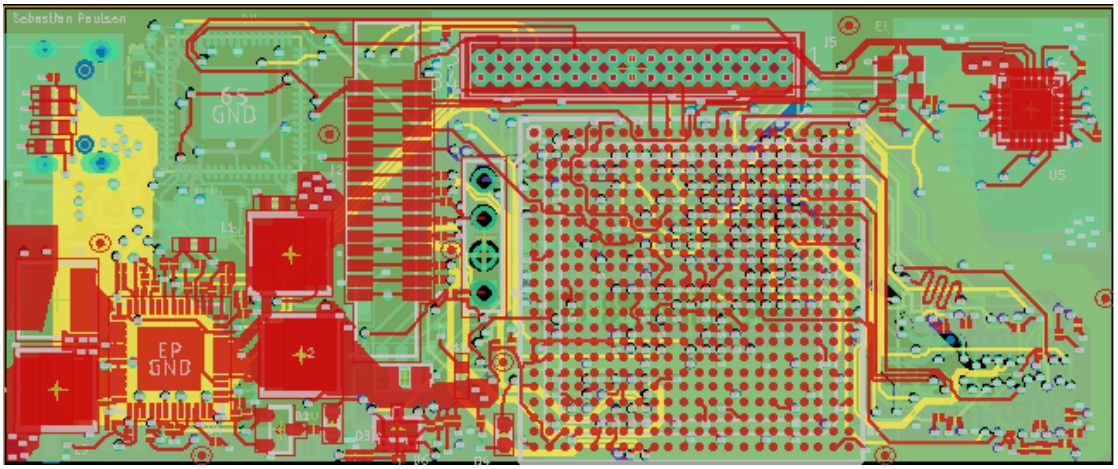
Quelle: Sebastian Paulsen

Abbildung A.7: Schaltplan Seite 7



Quelle: Sebastian Paulsen

Abbildung A.8: Schaltplan Seite 8



Quelle: Sebastian Paulsen

Abbildung A.9: Layout der Platine

Erklärung zur selbstständigen Bearbeitung einer Abschlussarbeit

Gemäß der Allgemeinen Prüfungs- und Studienordnung ist zusammen mit der Abschlussarbeit eine schriftliche Erklärung abzugeben, in der der Studierende bestätigt, dass die Abschlussarbeit „– bei einer Gruppenarbeit die entsprechend gekennzeichneten Teile der Arbeit [(§ 18 Abs. 1 APSO-TI-BM bzw. § 21 Abs. 1 APSO-INGI)] – ohne fremde Hilfe selbständig verfasst und nur die angegebenen Quellen und Hilfsmittel benutzt wurden. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen sind unter Angabe der Quellen kenntlich zu machen."

Quelle: § 16 Abs. 5 APSO-TI-BM bzw. § 15 Abs. 6 APSO-INGI

Erklärung zur selbstständigen Bearbeitung der Arbeit

Hiermit versichere ich,

Name: _____

Vorname: _____

dass ich die vorliegende Bachelorarbeit – bzw. bei einer Gruppenarbeit die entsprechend gekennzeichneten Teile der Arbeit – mit dem Thema:

Entwicklung und Evaluierung kompakter Hardware zur Realisierung künstlicher neuronaler Netze im Bereich autonomer Fahrzeuge

ohne fremde Hilfe selbständig verfasst und nur die angegebenen Quellen und Hilfsmittel benutzt habe. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen sind unter Angabe der Quellen kenntlich gemacht.

_____	_____	_____
Ort	Datum	Unterschrift im Original