Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Masterarbeit

Marco Kirschke

FPGA-basierte MPSoC-Plattform zur Integration eines Antikollisionssystems in die Fahrspurführung eines autonomen Fahrzeugs

Fakultät Technik und Informatik
Department Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Marco Kirschke

FPGA-basierte MPSoC-Plattform zur Integration eines Antikollisionssystems in die Fahrspurführung eines autonomen Fahrzeugs

Masterarbeit eingereicht im Rahmen der Masterprüfung
im Studiengang Informatik (Master) - Verteilte Systeme
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer : Prof. Dr.-Ing. Bernd Schwarz
Zweitgutachter : Prof. Dr. rer. nat Michael Schäfers

Abgegeben am 01. März 2012

Marco Kirschke

Thema der Masterarbeit

FPGA-basierte MPSoC-Plattform zur Integration eines Antikollisionssystems in die Fahrspurführung eines autonomen Fahrzeugs

Stichworte

MPSoC, FPGA, FAUST, Xilinx System Generator for DSP, Projektive Transformation, Echtzeit-Videodatenverarbeitung, autonome Fahrzeugführung, Geschwindigkeitsregelung, Laserscanner-basierte Objekterkennung, modellgetriebene RTL-Entwicklung, VHDL-Codegenerierung, MicroBlaze, Xilkernel

Kurzzusammenfassung

Diese Arbeit behandelt den Entwurf und die Entwicklung eines FPGA-basierten MPSoC zur Integration von Teilkomponenten in eine einheitliche Steuerungsplattform eines autonom agierenden Modellfahrzeugs. Eine Videodaten-basierte Bildberarbeitungspipeline wird zur Fahrspurapproximation mit der Hough-Transformation eingesetzt, aus deren Ergebnissen die Lenkwinkelstellgröße mit Algorithmen zur Fahrzeugführung bestimmt werden. Diese, durch Xilinx System Generator Blöcke realisierten, modular aufgebauten Teilkomponenten werden mit einer aus VHDL-Modulen bestehenden Geschwindigkeitsregelung in eine Hardware Accelerator Komponente zusammengefasst und als Intellectual Property in ein Multiprozessorsystem zur Fahrzeugsteuerung eingebunden. Das MPSoC besteht aus zwei MicroBlaze Prozessoren, die als AMP-basierte Prozessorarchitektur mit Shared Memory Anbindung die Parametrierung des Regelungsmoduls und die Integration von Laserscanner Abstandswerten für eine objektorientierte Hinderniserkennung umsetzen. Zur Integration von POSIX Threads wird das Xilkernel Echtzeitbetriebssystem auf den MicroBlazes eingesetzt. Darüber hinaus werden über das MPSoC Kommunikationsschnittstellen für die Aufzeichnung und Analyse von Systemparametern bei Erprobungsfahrten eingebunden.

Marco Kirschke

Title of the paper

FPGA-based MPSoC-platform to integrate a collision avoidance system into the path following system of an autonomous vehicle

Keywords

MPSoC, FPGA, FAUST, Xilinx System Generator for DSP, Projective Transformation, Realtime Image Processing, Autonomous Vehicle Control, Speed Control System, Laserscanner based Obstacle Detection, Model-Driven RTL Development, VHDL Code Generation, MicroBlaze, Xilkernel

Abstract

This thesis deals with the design and implementation of a FPGA-based MPSoC to integrate several components into an uniform control platform for an autonoumous RC vehicle. A camera-based image processing pipeline is used for lane approximation, which relies on the Hough transformation. The results are used in further components to determine the manipulated variable of the steering angle by applying various path tracking algorithms, implemented as Xilinx System Generator blocks. In combination with a VHDL based speed control module these components are encapsulated in a hardware accelerator intellectual property, which is integrated into the multiprocessor system. The MPSoC consists of two MicroBlaze softcore processors, which are assembled in an AMP-based structure and have connections to a shared memory region. The processor system handles both the communication to the hardware accelerator IP for vehicle control and the integration of laser scanner data for an obstacle detection system. The Xilkernel RTOS is used to gain access to thread functionalities of the POSIX standard. Furthermore the MicroBlaze system is used to include communication interfaces to the vehicle platform for measurement and analysis of system data during test runs.

Inhaltsverzeichnis

Abkürzungsverzeichnis

ADU ..... Analog-Digital-Umsetzter
AMP ..... Asymmetric Multiprocessing
ASCII ..... American Standard Code for Information Interchange
ASM ..... Arithmetic State Machine
BRAM ..... Bloch RAM
BSP ..... Board Support Package
CCD ..... Charge-coupled Device
CLB ..... Configurable Logic Blocks
DAC ..... Digital-to-Analog Converter
DCM ..... Digital Clock Manager
DDR2 SDRAM . Double Data Rate Synchronous Dynamic Random Access Memory
EAV ..... End Active Video
EDK ..... Xilinx Embedded Development Kit
EPP ..... Extensible Processing Platform
FAUST ..... Fahrerassistenz- und Autonome System
FFT ..... Fast Fourier Transformation
FIFO ..... First In, First Out
FIR ..... Finite Impulse Response
FPGA ..... Field Programmable Gate Array
FSL ..... Fast Simplex Link
FSM ..... Final State Machine
FTC ..... Follow-The-Carrot
FVAL ..... Frame Valid
HAW ..... Hochschule für Angewandte Wissenschaften
HDL ..... Hardware Description Language
HSYNC ..... Horizontal Synchronization
HT ..... Hough-Transformation
IP ..... Intellectual Property
ITU ..... International Telecommunication Union

JTAG ..... Joint Test Action Group
LAD ..... Look-Ahead-Distance
LAP ..... Look-Ahead-Point
LMB ..... Local Memory Bus
LVAL ..... Line Valid
MDM ..... MicroBlaze Debug Module
MHS ..... Microprocessor Hardware Specification
MPMC ..... Multi-Port Memory Controller
MPSoC ..... Multiprozessor Systems on Chip
PAL ..... Phase Alternating Line
PLB ..... Processor Local Bus
POSIX ..... Portable Operating System Interface
PP ..... Pure-Pursuit
PT ..... Projektive Transformation
PWM ..... Pulsweitenmodulation
RAM ..... Random Access Memory
RC ..... Remote-Control
RGB ..... Rot, Grün, Blau - Farbraum
ROI ..... Region-Of-Interest
ROM ..... Read Only Memory
RTL ..... Register Transfer Level
SAM ..... SystemACE Modul
SAV ..... Start Active Video
SAV ..... System on Chip Autonomous Vehicle
SCIP ..... Secure Communications Interoperability Protocol
SMP ..... Symmetric Multiprocessing
SoC ..... System on Chip
SPI ..... Serial Peripheral Interface
VHDL ..... Very High Speed Integrated Circuit Hardware Description Language
VSYNC ..... Vertical Synchronisation
XCL ..... Xilinx Cache Link

1 Einleitung

Der überwiegende Teil der heute eingesetzten Prozessoren arbeitet nicht in herkömmlichen PCs, sondern in Embedded Systems. Diese sind unter Verwendung von dedizierten Hardware- und Softwarekomponenten auf die optimale Ausführung spezieller Anwendungen ausgelegt. Sie bestehen in der Regel aus Sensorik-, Aktorik- und Steuerungselementen und sind in einer Vielzahl von industriellen und kommerziellen Anwendungen zu finden; der Medizintechnik, dem Fahrzeugbau, der Luftfahrttechnik, der Unterhaltungselektronik oder der Telekommunikationsindustrie [9]. Der Einsatz von Multiprozessorsystemen ist ein zentraler Bestandteil bei der Erstellung von Embedded Systems. Dabei werden die spezifischen Eigenschaften unterschiedlicher Prozessoren genutzt, um performante Ergebnisse zu erzielen.

Durch den stetigen Wandel in der Computerindustrie ergeben sich kontinuierlich neue Anforderungen an eingebettete Systeme. Diese bestehen vorrangig aus der Nutzung der höheren Leistungsfähigkeit von Hardwarebausteinen und zum Beispiel einer wachsenden Nachfrage an multifunktionalen Endgeräten sowie deren zunehmende Vernetzung [12]. In diesem Zusammenhang müssen die bestehenden Entwicklungsprozesse in der Form angepasst werden, dass die Hersteller schneller auf neue Trends und Entwicklungen am Markt reagieren können. Gleichzeitig dürfen die existierenden Anforderungen an die spezialisierten Systeme, wie zum Beispiel die energieeffizienten und wärmereduzierenden Leistungsmerkmale für mobile Anwendungen oder die Verarbeitung von hohen Datenraten in der digitalen Signalverarbeitung, nicht vernachlässigt werden [24].

Im Kontext dieser Ansprüche bietet der Einsatz von FPGA-basierten Systemen den Herstellern eine Reihe von Entwicklungsfreiheitsgeraden. Dazu gehört zum einen die Eigenschaft der FPGA Bausteine für Echtzeitanwendungen große Datenmengen durch Beschleunigungsmodule parallel verarbeiten zu können [7]. Auf der anderen Seite erlaubt ihre Rekonfigurierbarkeit die ständige Anpassung und Erweiterung von Entwicklungsplattformen. Durch die wachsende Anzahl an Logikressourcen innerhalb der FPGAs und der zunehmenden Verfügbarkeit von Softcore-Prozessoren, ergeben sich zudem neue Entwicklungsansätze für die Konfiguration von MPSoC (*Multiprozessor Systems on Chip*). Diese umfassen die Verlagerung bestimmter Implementierungsschritte des Hardware/Software-Codesigns zur reinen Softwareentwicklung, womit deren Konzepte adaptiert werden können und eine weitaus größere Gruppe an Entwicklern für diese Aufgaben zur Verfügung steht. Diesen Entwicklungsschritt verdeutlicht die Anfang 2011 erschienene Xilinx ZYNQ7000 EPP, ein FPGA Baustein mit einem festintegrierten *ARM dual-core Cortex A9* Multiprozessor [47].

Im Forschungsprojekt FAUST (*Fahrerassistenz- und Autonome System*) [11] des Departments Informatik an der Hochschule für Angewandte Wissenschaften Hamburg werden prototypische Fahrzeugplattformen entwickelt, um die Entwicklungsprozesse von eingebetteten Systemen zu durchdringen. Die Forschungsinhalte liegen dabei schwerpunktmäßig auf dem Einsatz von Sensorik, Telemetrie und Bildverarbeitung in Echtzeitsystemen mit unterschiedlichen Software- und Hardewarearchitekturen. Im Bereich Autonomes Fahren werden Modellfahrzeugplattformen im Maßstab 1:10 entwickelt, welche den Vorgaben des Carolo-Cup Regelwerkes entsprechen. Der Carolo-Cup [10] ist ein alljährlich von der TU Braunschweig veranstalteter Wettbewerb, bei dem Studentengruppen mit den von ihnen entwickelten Fahrzeugen in gestaffelten Disziplinen gegeneinander antreten. Zu den dynamischen Disziplinen gehören das autonome Durchfahren einer vorgegebenen Rundstrecke, sowohl mit Hindernissen als auch ohne, und das automatische Einparken in eine parallel zur Fahrspur angelegte Parklücke.

Die Hardware/Software-Codesign Gruppe der HAW Hamburg hat es sich zur Aufgabe gesetzt, ein solches Fahrzeug mit einem FPGA-basiertem Führungssystem zu entwickeln. Das SAV (*System on Chip Autonomous Vehicle*) vereinigt die Arbeitsergebnisse vorangegangener Bachelor- und Masterarbeiten mit den Schwerpunkten (vgl. Abb. 1.1):

- Bilddatenverarbeitung zur Fahrspurerkennung und Spurregelung: [17], [20], [26], [21]
- VHDL Implementierung einer Geschwindigkeitsregelung: [31], [8]
- Laserscanner-basierte Hinderniserkennung und Objekterzeugung: [32]
- automatisches Einparken durch Auswertung von Infrarot-Abstandsensoren: [1]
- Einrichtung einer Bluetooth Kommunikationsschnittstelle zur Datenaufzeichnung: [6]

Dabei wurden die einzelnen Teilkomponenten für unterschiedliche FPGA-Zielplattformen als Beschleunigermodule in VHDL, Matlab/Simulink Sytem Generator Module oder C-Softwareanwendungen für den MicroBlaze Softcore Prozessor implementiert.

Ziele des SAV Projektes

Ein Schwerpunkt der vorliegenden Arbeit bezieht sich auf die Integration der oben aufgeführten Entwicklungen in eine einheitliche Systemarchitektur und die Inbetriebnahme des SAV. Das beinhaltet den Entwurf und die Erzeugung einer Hardwareplattform, die Anordnung der einzelnen Module im System und die Anpassung der Schnittstellen. Ein zusätzlicher Bestandteil des Entwicklungsprozesses ist die Montage der Sensorik und Aktorik mit allen Daten- und Versorgungsleitungen. In Abbildung 1.2 werden die elementaren Bestandteile des SAV Gesamtsystems schematisch dargestellt. Als Ausgangsplattform dient das Xilinx Spartan-3A DSP Starter Board, dass mit einem Spartan-3A 3SD1800A FPGA

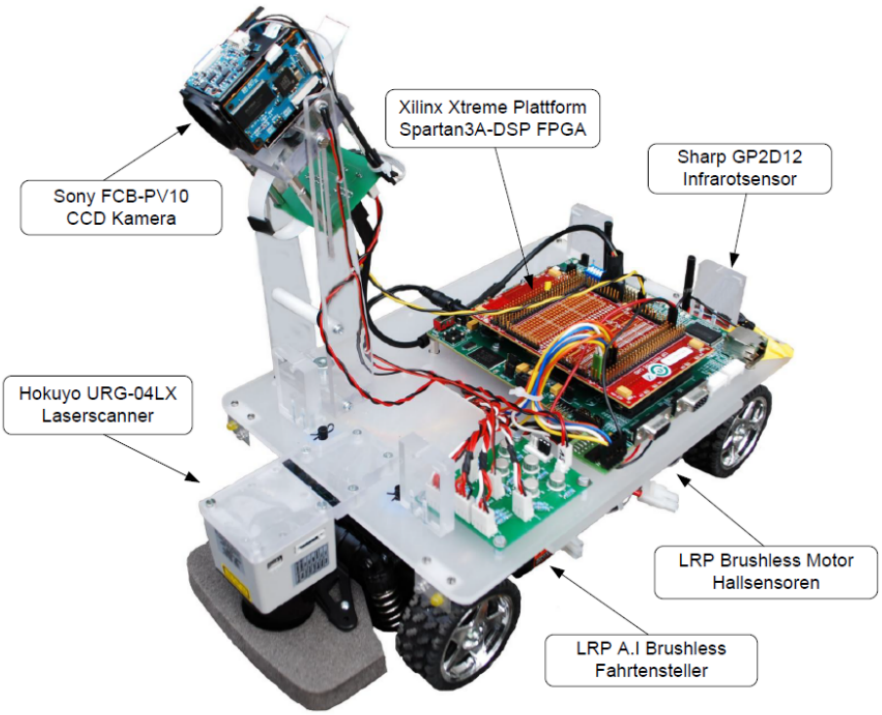


Abbildung 1.1 – Die im Zusammenhang dieser Arbeit entstandene Erprobungsplattform; das SoC-Autonomous-Vehicle (SAV).

und 128 MB DDR2 SDRAM ausgestattet ist; zusätzlich stellt es über zwei EXP Expansion Connector 168 User I/O Schnittstellen zur Anbindung der Sensorik und Aktorik zur Verfügung.

Die Architektur der SAV-Plattform zur Kontrolle des Fahrzeugs im autonomen Modus kann im Wesentlichen in drei Teile untergliedert werden (vgl. Abb. 1.2):

- Aktoren und Sensoren zur Datenerfassung bzw. Fahrzeugführung
- hardwarebeschleunigte, parallele Signaldatenverarbeitung der Kamera-, Geschwindigkeits- und Fernbedienungsdaten in Form des SAV CONTROL IP-Cores
- Verarbeitung der Sensordaten, Interpretation und Steuerung der Aktoren über Softwareanwendungen auf einem MPSoC, bestehend aus zwei MicroBlaze Prozessoren

Die Ziele der vorliegenden Arbeit lassen sich aus den oben genannten Teilkomponenten detaillierter ableiten:

- Entwurf und Entwicklung einer Hardwareplattform für ein MPSoC zur Integration der RTL-Beschleunigermodule, der Matlab/Simulink System Generator Komponenten und des DUAL-MICROBLAZE-SYSTEMS

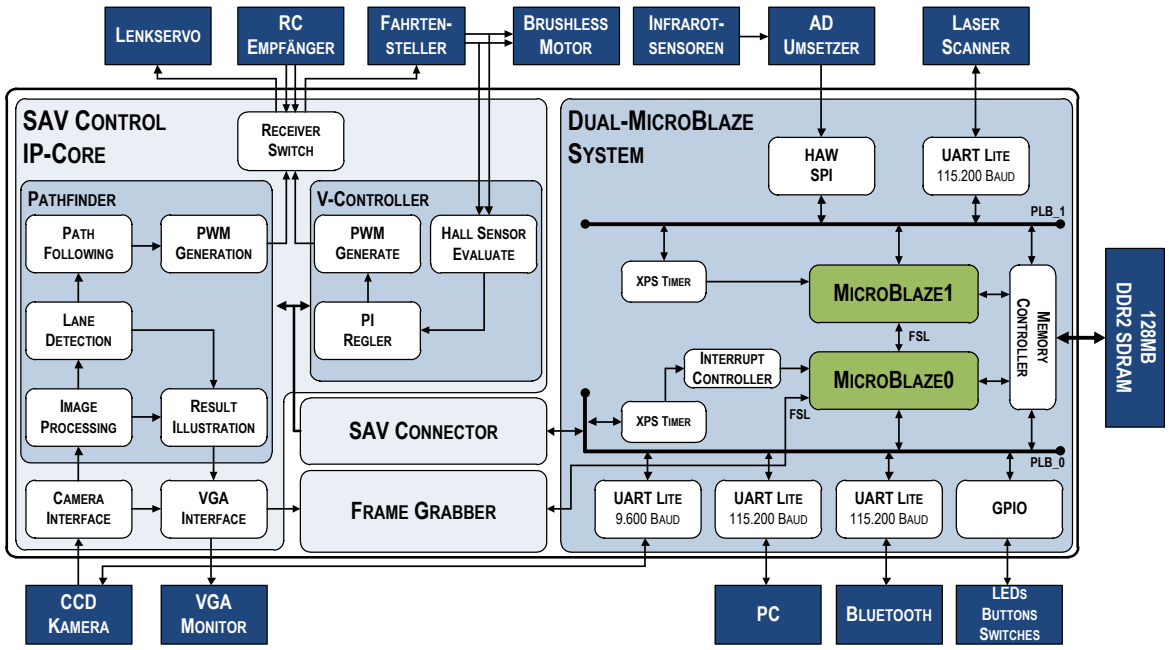


Abbildung 1.2 – Das SAV System; die Regelung der Fahrzeugführung im SAV CONTROL IP wird durch das DUAL-MICROBLAZE-SYSTEM über die Schnittstelle des SAV CONNECTORS gesteuert.

- Konfiguration des MPSoC mit zwei MicroBlaze Prozessoren, Shared-Memory Speicherarchitektur, Kommunikationsmodul zum SAV CONTROL IP-Core, User I/O Schnittstellen zur Kommunikation mit den Sensoren und Einrichtung des Xilkernel Betriebssystems
- Anpassung der Bildbearbeitungspipeline zur Aufnahme der Kameradaten und Darstellung auf einem VGA-Monitor
- Analyse und Erweiterung der System Generator Komponenten zur Fahrspurerkennung und Spurführung sowie der Bereitstellung von Parametrierungsregistern zur Steuerung der Prozesskette
- Integration der Geschwindigkeitsregelung in den SAV CONTROL IP-Core
- Eingliederung der Laserscanner-basierten Objekterkennung, Implementierung der Hindernisdetektion und Ausweichautomatisierung in Software
- Vorbereitung der Infrarot-Sensor Schnittstellen für den Einparkassistenten
- Einrichtung der Kommunikationsverbindungen über Bluetooth und serieller RS232 Schnittstelle

- Montage und Einbau der Aktoren/Sensoren, Adapterplatinen, des Entwicklungsboards und der Spannungsversorgung mit allen erforderlichen Schnittstellen

Zur Umsetzung dieser Ziele werden die bestehenden Teilkomponenten zunächst abhängig von ihren Implementierungsverfahren als VHDL-Komponenten, System Generator Module oder SoC-basierte Softwareanwendungen untersucht und eine erforderliche Vertiefung in die Entwicklungsprozesse vorgenommen. Anschließend werden die Ergebnisse der Vorarbeiten in die verwendete Entwicklungsumgebung des Xilinx EDK 13.1 portiert und sukzessive in die Entwicklungsplattform des Spartan-3A DSP FPGAs integriert. Gleichzeitig erfolgt der mechanische Aufbau der Fahrzeugplattform, um die integrierten Teilsystem in Betrieb zu nehmen; das beinhaltet neben der Herstellung von Verbindungssteckern und dem Einrichten von Versorgungsspannungen die Montage der Sensoren und Aktoren im Modellfahrzeug.

Insbesondere die Anpassung der Kamerakalibrierung an das im Zielfahrzeug bestehende Sichtfeld, ist für eine erfolgreiche Erkennung der Fahrbahnmarkierung erforderlich. Dazu werden die Pipelinestufen zur Vorverarbeitung des Kamerabildes und die Module zur Fahrzeugführung detaillierter betrachtet. Daneben bestehen weitere Ziele der vorliegenden Arbeit in der Erprobung der Fahreigenschaften im autonomen Betrieb auf einer Teststrecke und der Entwicklung einer Softwareanwendung zum Wechsel der Betriebsparameter.

Gliederung der Arbeit

In **Kapitel 2** werden Grundlagen zum modernen Entwurf von eingebetteten Systemen in Bezug auf die vorliegende Arbeit vorgestellt. So werden die Entwicklungsmethoden eines strukturierten VHDL-Entwurfes und die des Xilinx System Generators im Kontext von FPGA-basierten Anwendungen erläutert.

Die für den Entwurf des Gesamtsystems relevanten technologischen Konzepte und mathematischen Grundlagen werden in **Kapitel 3** beschrieben. Dabei wird speziell auf das Datenformat der Bildbearbeitung, den Algorithmen zur Fahrspurdetektion und Spurführung sowie auf die Konzepte zu objektbasierten Hinderniserkennung eingegangen.

Die konkreten Implementierungsergebnisse sind in **Kapitel 4** zusammengefasst. In diesem Teil wird speziell auf den Zusammenhang der technologischen Grundlagen mit dem erzeugten Entwicklungen eingegangen sowie eine detaillierte Beschreibung der Signalpfade und Kommunikationsabläufe innerhalb der Komponenten vorgenommen. Zusätzlich befindet sich in diesem Kapitel die Dokumentation der Softwareanwendung, die auf den MicroBlaze Prozessoren des Zielsystems ausgeführt wird.

Kapitel 5 analysiert die Ergebnisse des vorliegenden Systems in Bezug auf die Ressourcenauslastung und Auswirkung von Kalibrierungsparametern; zudem wird eine Bewertung des Fahrzeugverhaltens im autonomen Fahrmodus vorgenommen. Am Ende des Kapitels

werden Vorschläge für zukünftige Forschungsschwerpunkte und Weiterentwicklungen formuliert.

Den Abschluss dieser Arbeit bildet **Kapitel 6** mit einer Zusammenfassung.

2 Ansätze zum modernen Embedded Systems Entwurf

Der Entwicklungsprozess von eingebetteten Systemen unterteilt sich in zwei grundlegende Bestandteile: der Erstellung einer Hardwareplattform mit spezialisierten Eigenschaften, die der Spezifikation einer konkreten Aufgabenstellung entsprechen, und der Entwicklung von Softwareanwendungen, die den Funktionsumfang dieser Spezialplattformen ausschöpfen. Die klassische Vorgehensweise der sukzessiven Entwicklung von Hardware- und Software Komponenten wird dabei zunehmend durch den moderneren Ansatz des Hardware/Software Codesigns ersetzt. Dieser umfasst den gemeinsamen, zeitgleichen Entwurf von Hardware- und Software-Bestandteilen, wodurch Alternativen für die Implementierung erkannt werden, und eine differenziertere Entscheidung zur Realisierung von Funktionen durch Hardware- oder Softwarekomponenten getroffen werden kann. Die Entwurfsmethodik des HW/SW Codesign lässt sich durch folgende Phasen charakterisieren [23]:

- **Spezifikationsphase** - Die Aufgabenstellung wird durch eine Funktionalitäten Beschreibung abstrakt spezifiziert. Dabei werden verschiedene Realisierungsalternativen für modulare Komponenten des Systems formuliert und untersucht. Neben der Identifikation von kritischen Systemeigenschaften werden bestehende Entwürfe und Module auf deren Wiederverwendbarkeit betrachtet.
- **Explorationsphase** - In dieser Phase werden die Realisierungsalternativen auf Kosten und Leistungsfähigkeit verglichen. Die Systemfunktionen werden gruppiert und auf potentielle Komponenten verteilt, dabei werden konkrete Implementierungseigenschaften der Alternativen gegeneinander gestellt und bewertet.
- **Verfeinerungsphase** - Hier werden die spezifizierten Teilsysteme auf entsprechende Hardware- bzw. Softwarekomponenten verteilt und die Schnittstellen zwischen diesen definiert.

Der entscheidende Vorteil dieses Ansatzes liegt in der abstrakten Systemsicht vor der Verfeinerungsphase. So kann in der Explorationsphase die Realisierung einer Systemfunktion zunächst unabhängig bewertet werden. Im Übergang erfolgt dann die konkrete Festlegung auf eine bestimmte Hardware- bzw. Softwarerealisierung. Somit wird erst nach der abgeschlossenen Spezifikation entschieden, ob zum Beispiel die Integration von Sensorwerten über eine spezielle Hardwarelösung implementiert werden muss oder die Realisierung mit

Standardkomponenten genügt. Zusätzlich lassen sich die Phasen der Exploration und Verfeinerung auch im laufenden Entwicklungsprozess erneut anwenden. Dadurch entsteht die Möglichkeit auf neue Anforderungen oder die Einführung neuer Standards zeitnah zu reagieren.

Diese Entwurfsmethode lässt sich auf die Entwicklung von FPGA-basierten System-On-Chip Plattformen abbilden. Das klassische Anwendungsgebiet der FPGAs, als *Hardware Accelerator* Komponenten zur parallelen Verarbeitung von Signaldaten, lässt sich durch die steigende Anzahl an Logikblöcken innerhalb der FPGAs (vgl. Abb. 2.1) und der Verfügbarkeit von Softcore Prozessoren erweitern. So verbinden moderne FPGA-basierte Systeme Hardware- und Softwarekomponenten auf einer Plattform, deren Partitionierung als wesentlicher Bestandteil des Entwicklungsprozesses gilt. Aufgrund des modularen Entwurf Ansatzes ist die Entscheidung zur Implementierung von Teilkomponenten in HW oder SW zu jedem Zeitpunkt der Entwicklungsphase gegeben. Abhängig vom Laufzeitverhalten oder der Ressourcenauslastung eines bestehenden Systems kann die Realisierung einzelner Teilkomponenten ständig angepasst oder verändert werden.

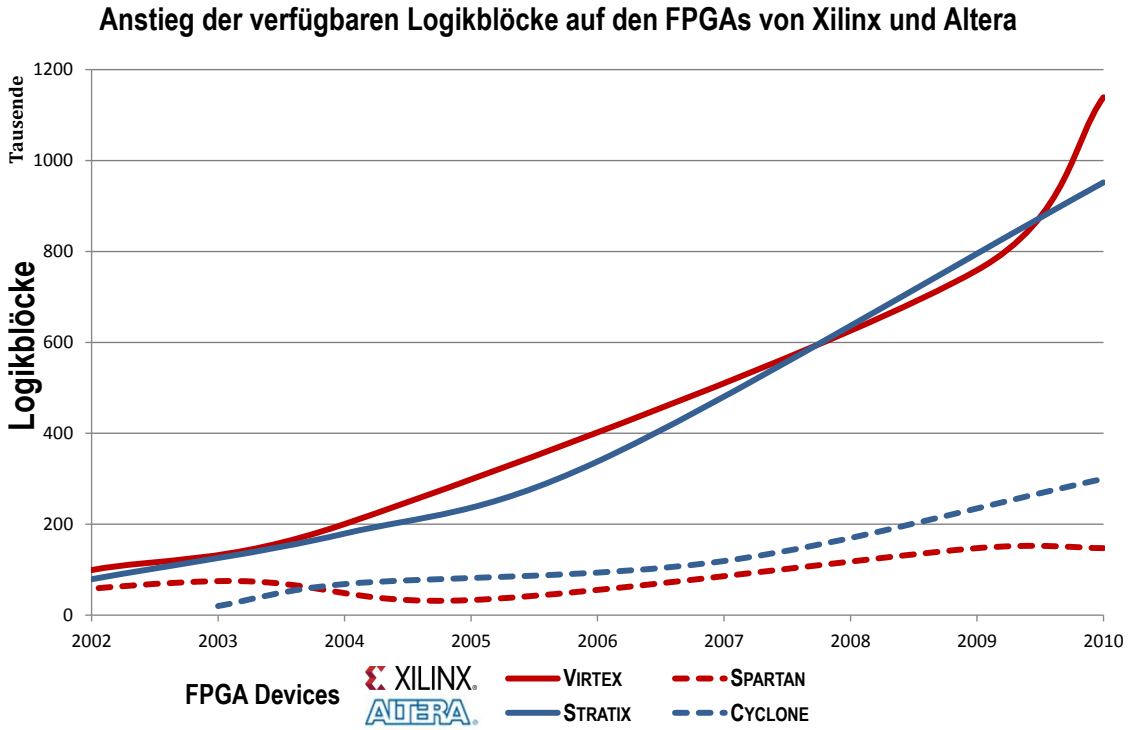


Abbildung 2.1 – Die Anzahl der auf den FPGAs verfügbaren Logikelemente der beiden marktführenden Hersteller nimmt kontinuierlich zu (vgl. Anhang E).

Die Entwurfsmethode kann auch auf den Entwicklungsprozess der vorliegenden Arbeit übertragen werden, da als Ergebnis der oben genannten Vorarbeiten bereits Teilsysteme

entstanden sind. Im ersten Entwicklungsschritt werden die bestehenden Teilsysteme analysiert, ihre Schnittstellen untereinander bestimmt und in eine einheitliche Systemumgebung integriert. Für kommende Optimierungen wird der modulare Aufbau allerdings beibehalten. Somit können Einzelkomponenten zu einem späteren Zeitpunkt gegebenenfalls angepasst oder durch neue Lösungen in HW bzw. SW implementiert werden.

Aus der Analyse der Teilsysteme ergeben sich drei unterschiedliche Implementierungsverfahren zur Erstellung FPGA-basierter Anwendungen:

- dem VHDL-basierten Ansatz zur Erstellung von *Hardware Accelerator* Komponenten,
- der Erzeugung von Synthese-fähigen Netzlisten über den Xilinx System Generator for DSP und
- dem Entwurf von SoC-Plattformen zur Integration von Softwareentwicklungen.

Im Folgenden werden die grundlegenden Strukturen dieser Entwicklungsverfahren in einem zum Verständnis der vorliegenden Arbeit entsprechendem Umfang vorgestellt.

2.1 Parallele Datenstromverarbeitung in FPGAs

Die Erstellung von FPGA-basierten *Hardware Accelerator* Modulen beruht auf der Konfiguration der Logikelemente des FPGAs zu integrierten Schaltungen durch eine Hardwarebeschreibungssprache HDL (*Hardware Description Language*). Dadurch lässt sich die Hardware-Modellierung in der Abstraktionsebene des RTL-Designs beschreiben. Dieser Ansatz beschreibt das Verhalten von digitalen Schaltungen durch Registerstufen und kombinatorischer Logik [25]. Die HDLs stellen dabei, im Gegensatz zu herkömmlichen Programmiersprachen der Softwareentwicklung, auch Notationselemente zur Beschreibung von zeitlichen Abläufen und nebenläufigen Prozessen bereit; somit kann das exakte Verhalten von Hardwarebausteinen spezifiziert werden. Durch den Einsatz von Synthesewerkzeugen wird auf Basis dieser Spezifikation eine Netzliste erstellt, die als Grundlage des physikalischen Chip-Designs dient.

Das Konzept des strukturellen VHDL-Entwurfs bezeichnet die Unterteilung eines digitalen Systems in Daten- und Steuerpfad. Durch die Strukturierung wird die Komplexität der Entwicklung auf einzelne Schritte verteilt, womit sich die Verifikation des Gesamtsystems erleichtert. Der **Steuerpfad** beeinflusst das zeitliche Verhalten des Datenpfades durch Steuersignale, die vorwiegend durch den Einsatz von Zustandsautomaten generiert werden. Die funktionalen Anforderungen einer VHDL Entwicklung werden innerhalb des **Datenpfades** realisiert. Diese Anforderungen beziehen sich auf die Verarbeitung von Eingangssignalen,

durch die Anwendung von algorithmischen Operationen, zur Erzeugung von Ausgangssignalen. Zur Synchronisation mit dem Steuerpfad erzeugt der Datenpfad Statussignale (vgl. Abb. 2.2).

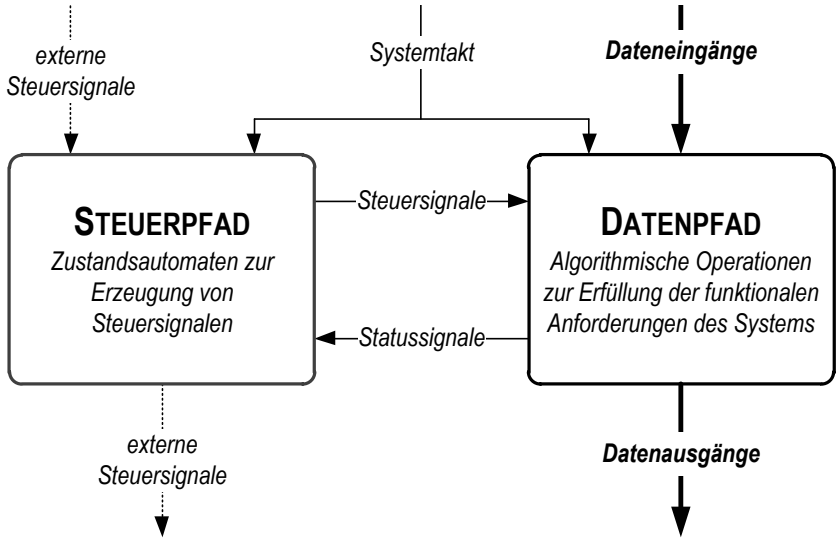


Abbildung 2.2 – Die Strukturierung in Daten- und Steuerpfad verringert die Komplexität des Gesamtsystems zur vereinfachten Verifikation in der Entwicklungsphase.

Abhängig von der Datenrate und des Systemtaktes, ergeben sich unterschiedliche Ausprägungen des Datenpfades. Bei einer hohen Datenrate, deren Frequenz im Bereich des Systemtaktes liegt, wird die Verarbeitung der Eingangsdaten in Pipelinestufen vorgenommen. Da die Zwischenspeicherung der Elemente des Datenstroms nicht realisierbar ist, werden alle Bearbeitungsschritte sequentiell und taktsynchron vorgenommen. Die dadurch entstehende Latenz der Ausgangssignale ist proportional zur Anzahl der durchlaufenen Pipelinestufen. Die Umsetzung dieses Entwurfes erfordert allerdings einen erhöhten Bedarf an Logikressourcen, da alle Signale des Datenstroms in allen Pipelinestufen parallel verarbeitet werden müssen. Ist die Datenraten jedoch deutlich geringer als die Systemfrequenz, stehen zur Verarbeitung der Eingangssignale mehrere Taktzyklen zur Verfügung. In einem sogenannten Multizyklusdatenpfad werden Arithmetik Elemente für die Berechnung von unterschiedlichen Eingangsdaten verwendet, wodurch Logikressourcen eingespart werden können. Das *Resource-Sharing* lässt sich auch auf Speicherelemente übertragen (*Register-Sharing*), bei dem Ergebnisse unterschiedlicher Berechnungen in den Speicherblöcken vorgehalten werden. Allerdings erfordert der Multizyklusdatenpfad einen erhöhten Steuerungs- und Signalisierungsaufwand des Steuerpfades, da die Eingänge der entsprechenden Speicher- und Arithmetikblöcke durch Multiplexer geschaltet werden müssen [25].

Eine weitere Strukturierung des Systementwurfs wird über die Verschachtelung von erstellten VHDL-Komponenten erreicht. Die VHDL-Entities werden durch ihre Ein- bzw. Ausgangssignale miteinander verbunden, wodurch Bottom-Up oder Top-Down Entwurfsansätze

realisiert werden können. Zusätzlich kann dadurch das Verhalten von Teilprozessen eigenständig getestet werden. Spezielle Simulationsumgebungen, wie *Mentor Graphics ModelSim*, werden zur Analyse der Laufzeitpfade von Daten- und Steuerungssignalen eingesetzt. Dazu werden die Eingangssignale in einem zeitlichen Kontext in VHDL-Testbenches definiert und die logische Wirkungsweise der VHDL-Komponente simuliert. Anschließend lassen sich die resultierenden Ausgangssignale mit der Systemspezifikation verifizieren.

2.2 Embedded System Design mit dem Xilinx System Generator for DSP

Der Xilinx System Generator ist eine Erweiterung für die Matlab/Simulink Umgebung, mit der aus der Modellierung von digitalen Fixed-Point Signalverarbeitungsalgorithmen HDL-Code für Xilinx FPGAs erzeugt wird. Dazu steht ein Xilinx DSP Blockset zur Verfügung, dessen Elemente Abstraktionen für mathematische-, logische-, Speicher- und DSP- Funktionen darstellen. Über den Simulink Ansatz, der grafischen Modellierung von Algorithmen, werden diese zusammengefügt. Dabei sind die Blöcke in ihren unterschiedlichen Funktionen sowohl *bit-true* als auch *cycle-true*, was die exakte, bitgenaue und zeitdiskrete Abbildung der Modelleigenschaften in die physische Hardwareplattform beschreibt. Dadurch kann das Verhalten des Modells durch Simulationsverfahren bereits in der Entwicklungsphase verifiziert werden. Zur Simulation wird dabei einerseits die Simulink Umgebung verwendet, auf der anderen Seite können aus den Systemmodellen VHDL-Testbenches zur Simulation in ModelSim generiert werden. Darüber hinaus ist der vollständige Matlab Funktionsumfang gegeben, wodurch Systementwürfe vor der Umsetzung in System Generator Blöcke durch Algorithmen in Matlab Code evaluiert werden können.

Die Xilinx Blockset Library stellt eine Vielzahl von Funktionsblöcken zur Modellierung zur Verfügung:

- Signalverarbeitungsblöcke (z.B. FIR-Filter, FFT)
- Speicherblöcke (z.B. Shift Register, FIFO, ROM, RAM)
- Arithmetikblöcke (z.B. Mult., AddSub, Div.)
- Logikblöcke (z.B. Logical, Relational)
- Blöcke zur Modifikation von Datenformaten (z.B. Convert, Concat, Scale, Shift)

Durch die Kombination der Funktionsblöcke und der Verbindung ihrer Ein- und Ausgangsports wird das funktionale Systemverhalten modelliert (vgl. Abb. 2.3). Dabei kann der Systementwurf modular aufgebaut werden, um einerseits die Simulation und das Testen zu vereinfachen, aber auch die Wartbarkeit und Wiederverwendbarkeit der Module zu gewährleisten.

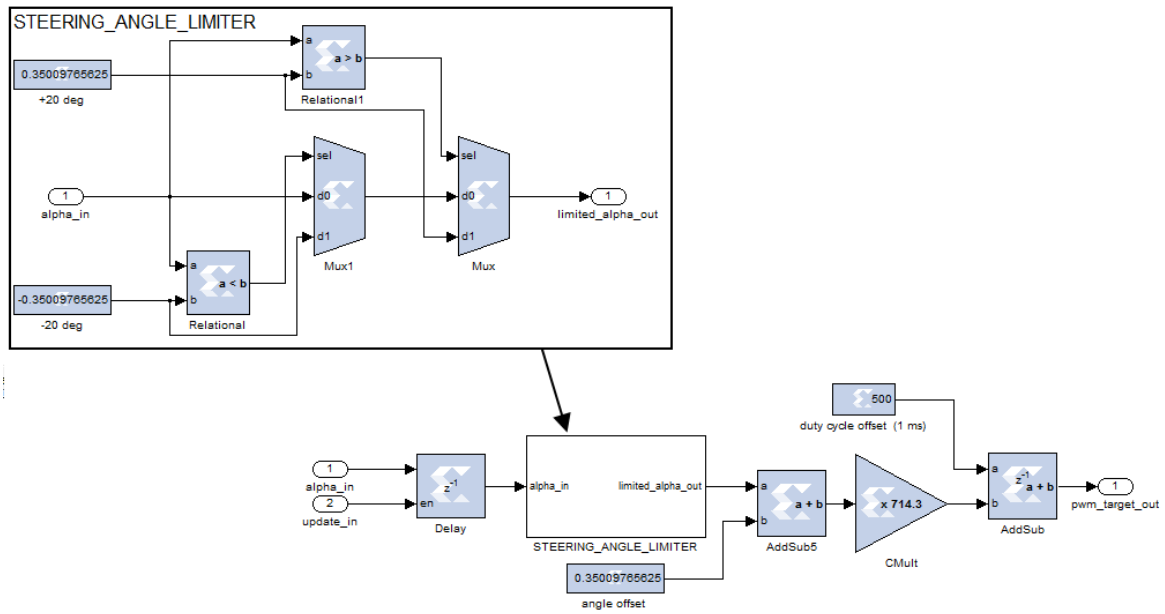


Abbildung 2.3 – Die Modellierung eines System Generator Moduls am Beispiel der Generierung des PWM Signals für den Lenkwinkel im Fahrzeugführungssystem. Über die Kombination abgeschlossener Module (hier die Limitierung der Winkeleingangsgrößen) lässt sich ein strukturierter Systementwurf durchführen.

Für die Einbindung von System Generator Modulen in größere Gesamtsysteme werden die äußeren Systemschnittstellen durch *Gateway In* bzw. *Gateway Out* Module realisiert. Diese stellen im erzeugten Hardwaredesign Top Level Input/Output Ports dar und führen eine Typkonvertierung zwischen Matlab Datentypen und dem Xilinx Fixed-Point Format durch. Die Abbildung zeitlichen Systemverhaltens wird durch die Verwendung von *Up* bzw. *Down Sample* Blöcken vorgenommen. Dadurch kann die Taktfrequenz in Teilbereichen des Systems abhängig von der Systemfrequenz erhöht bzw. verringert werden, um Multizyklus-pfade zu realisieren. Die Bereitstellung der unterschiedlichen Taktfrequenzen erfolgt wahlweise über die Verwendung eines Clock Managers, der bei der Generierung der Netzlisten vom System Generator erzeugt wird, oder kann über externe Clock Eingangssignale von außen erfolgen. Zur direkten Integration von Matlab M-Funktionen ist der MCode Block vorgesehen. Dieser unterstützt einen eingeschränkten Teil der Matlab Sprachkonstrukte wie Zuweisungen, If-Then-Else Statements, Arithmetische Operationen, Logik Operatoren und Typenkonvertierungen, um arithmetische Funktionen, Zusandsautomaten oder Steuerungslogiken zu implementieren [51]. Der Matlab Code bestimmt die Erzeugung der Ausgangssignale des MCode Blockes während der Simulink Simulation; gleichzeitig kann der System Generator die M-Funktionen in VHDL Code umsetzten, um diese in den Systementwurf zu integrieren.

Die Ein- bzw. Ausgangssignale müssen dabei im Xilinx Fixed-Point Format anliegen; dieses

bildet die Zahlendarstellung im Q-Format ab, wobei vorzeichenbehaftete Werte in Zweierkomplementdarstellung als *fix* und vorzeichenlose Zahlen als *ufix* bezeichnet werden. Das Q-Format bietet den Vorteil, dass für die Berechnungen keine Floating Point Bibliotheken eingebunden werden müssen und das Synthesetool des System Generators die Berechnungen in Integer Arithmetik anlegt. Dadurch werden im Vergleich zur Implementierung mit Floating Point Datentypen weniger Hardware Ressourcen auf dem FPGA beansprucht. Gleichzeitig wird die Verteilung von Ganzzahl- und Fließkomma Anteilen für Datentypen im Q-Format durch den Entwickler festgelegt, wodurch die Vektorbreiten abhängig von der konkreten Verwendung des Datentypens dimensioniert werden können [3].

3 Funktionskonzepte des SAV

Das Steuerungssystem des SAV für den Einsatz im autonomen Betrieb besteht aus zwei Hauptkomponenten (vgl. Abb. 3.1); einem SAV CONTROL Modul zur Regelung der Fahrzeugführung und der Geschwindigkeit, und dem DUAL-MICROBLAZE-SYSTEM, durch das Einfluss auf die Regelung genommen werden kann. Zusätzlich bindet es die Abstandssensorik zur Detektion von Objekten im Fahrzeugumfeld in die Steuerungsplattform ein.

Der SAV CONTROL IP-Core ist als modulare Anordnung von Hardwarebeschleuniger-Komponenten realisiert. Die Bestandteile zur Regelung der Motor- und Lenkwinkelsteuerung sind in Form von VHDL-Implementierungen angelegt oder durch System Generator Blöcke modelliert. Der hierarchische Aufbau der Teilkomponenten erleichtert zum einen das Testen des Gesamtsystems und lässt darüber hinaus das Einbinden von neuen Entwicklungen zu. Die Zusammenführung der Module in einen einzigen, eigenständigen IP-Core, mit definierten Ein- und Ausgangsschnittstellen, erleichtert zudem die Integration des SAV CONTROL in eine SoC-Plattform.

Die Regelung des Fahrzeugantriebs bzw. der Lenkwinkelstellung wird durch zwei unabhängig voneinander agierenden Einzelsystemen vorgenommen. Die Ermittlung des Lenkwinkels beruht auf einer Bildverarbeitungspipeline, die auf Grundlage von Videodaten einer CCD-Kamera eine Fahrspurerkennung vornimmt, die detektierte Fahrbahnmarkierung in Abhängigkeit zur Fahrzeugposition setzt und durch Spurführungsalgorithmen die Fahrzeugausrichtung neu bestimmt. Parallel wird die Regelung der Fahrzeuggeschwindigkeit in einem separaten Teilsystem vorgenommen. Dazu werden Hall-Sensorwerte zur Bestimmung der Achsenrotation des Motors ausgewertet und die Fahrzeuggeschwindigkeit über einen PI-Regelstrecke angepasst. Gleichzeitig können die beiden Regelungen mit der angeschlossenen Fernbedienungsanlage unterbrochen werden, um das Fahrzeug manuell zu steuern.

Das DUAL-MICROBLAZE-SYSTEM verfügt über Schnittstellen, mit denen die Regelstrecken des SAV CONTROLS beeinflusst und Daten aus der Bildbearbeitungspipeline abgegriffen werden können. Es besteht aus zwei MicroBlaze Prozessoren, einer Shared Memory Architektur und verschiedenen IP-Cores zur Kommunikation mit Sensorik und Entwicklungsumgebung.

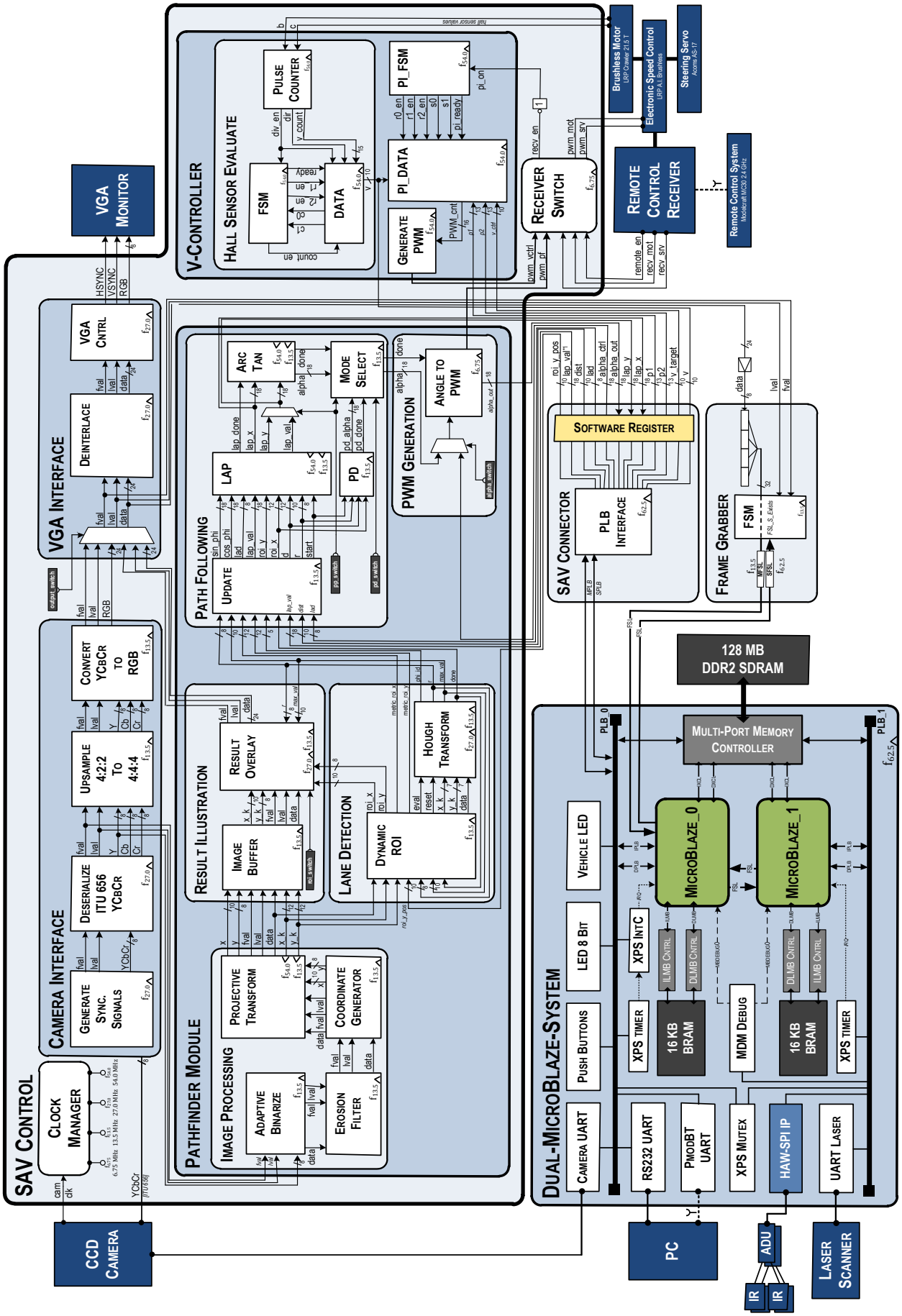


Abbildung 3.1 – Die beiden Kernkomponenten der SAV Plattform: SAV-CONTROL zur Fahrzeugregelung und DUAL-MICROBLAZE-SYSTEM zur Steuerung.

Als Grundlage zur Entwicklung von Softwareanwendungen wird das Xilkernel Betriebssystem auf den Prozessoren eingesetzt. Dieses stellt ein Echtzeitbetriebsystem dar, kann speziell auf die Konfiguration der MicroBlazes angepasst werden und unterstützt die grundlegenden Funktionen des POSIX *pthread* Standards zur Erstellung von Threads und Zugriffskontrollen auf kritische Bereiche. Darüber hinaus wird die Integration von Abstandsensoren in das Gesamtsystem im DUAL-MICROBLAZE-SYSTEM vorgenommen. Diese werden zur Detektion von Hindernissen auf der Fahrbahn oder für die Bestimmung geeigneter Parklücken, für ein automatisches Einpark-Szenario, eingesetzt.

Nachfolgend wird die grundlegende Technologie der Bestandteile beider Hauptkomponenten beschrieben, bevor in Kapitel 4 die konkrete Implementierung dargestellt wird.

3.1 Empfang und Darstellung der Kameradaten in der Bildbearbeitungspipeline

Die digitalen Kameradaten werden primär zur Detektion der Fahrbahnmarkierung verwendet. Für die Entwicklungsphase und zur Kalibrierung der Kamera ist aber auch die vollständige Darstellung des Sichtbereiches notwendig. Daher umfasst die Bildbearbeitungspipeline zusätzliche Module zur Ausgabe der Kameradaten auf einem VGA-Monitor. Die CCD Kamera liefert einen Pixeldatenstrom nach dem ITU-Rec. BT.656 Standard im YCbCr 4:2:2 Format [30], der innerhalb der Bildverarbeitungspipeline interpretiert und ins RGB-Format konvertiert wird. Die Kamera ist auf einen Interlaced Modus kalibiriert, wodurch innerhalb eines Frames zwei Halbbilder mit jeweils 640x240 Bildpunkten bei einer Frequenz von 27 MHz übertragen werden. Die Synchronisationssignale sind im Pixeldatenstrom eingebettet, um die Anzahl an Signalleitungen zum Sensor gering zu halten. Anhand der eindeutig kodierten Synchronisationssequenzen, SAV (*Start Active Video*) und EAV (*End Active Video*), wird die Verarbeitungspipeline auf den Datenstrom synchronisiert und systeminterne LVAL (*Line Valid*) bzw. FVAL (*Frame Valid*) Signale erzeugt [17].

Das YCbCr Farbmodell ist eine leichte Abwandlung des analogen YUV Farbmodells und findet seinen Ursprung in der digitalen PAL Fernsehtechnik. Durch die komprimierenden Eigenschaften diese Modells wird es auch bei der Kodierung von JPEG Bildern und MPEG Videos verwendet. Das Prinzip des Formats basiert auf der Verteilung der Farbinformationen eines Pixels auf die Grundhelligkeit im Y-Wert (Luminanz) und den zwei Farbinformationswerten Cb, für Blau zu Gelb Chrominanz, und Cr, für Rot zu Türkis Chrominanz. Angelehnt an die Verarbeitung der Farbinformation durch das menschliche Auge, bei der Helligkeitsunterschiede wesentlich deutlicher wahrgenommen werden als geringe Farbvariationen, besitzen die Luminanzwerte nur die Helligkeitsinformationen eines Pixels. Die Y-Werte eines vollständigen Bildes können somit als Grauwertbild angegeben werden (vgl. Abb. 3.2). Die Chrominanzwerte enthalten die jeweilige Abweichung der Farbigkeit vom

Y-Helligkeitswert zu Blau/Gelb bzw. Rot/Türkis. Da die Chrominanzunterschiede weniger

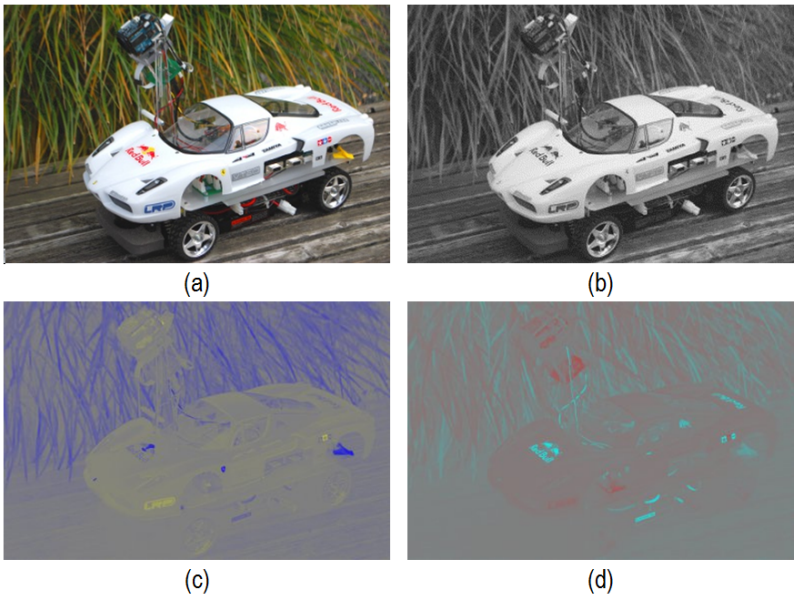


Abbildung 3.2 – Zusammensetzung des YCbCr Farbmodells:(a) das Original RGB-Bild, (b) Y-Luminanzwerte als Graustufenbild, (c) und (d) die Blau/Gelb- bzw. Rot/Türkis- Chrominanzwerte

Gewicht in der Wahrnehmung des menschlichen Auges haben, kann dieser Anteil im YCbCr Datenformat reduziert werden, ohne große Qualitätsverluste hinnehmen zu müssen. Diese Reduktion wird durch das Abtastverhältnis 4:2:2 beschrieben, was bedeutet, dass die Chrominanzanteile nur für jeden zweiten Bildpunkt geliefert werden. Der Videodatenstrom wird demnach Zeile für Zeile zu jeweils 8-Bit in der Form

$$C_B, Y, C_R, Y, C_B, Y, C_R, \text{etc.}$$

übertragen.

Das YCbCr Farbmodell bietet sich für die Verwendung in der Fahrspurerkennungspipeline an, da die Fahrbahnmarkierung durch die Auswertung von Luminanzdifferenzen ermittelt wird. Dazu wird der eingehende Pixeldatenstrom in einem späteren Verarbeitungsschritt über ein Erosions Filter binarisiert; als Eingangsdaten werden dabei ausschließlich die Y-Helligkeitswerte verwendet.

Zur Darstellung des Kamerabildes auf einem VGA Monitor wird das Abtastverhältnis auf 4:4:4 erweitert und anschließend ins RGB-Farbmodell konvertiert. Das Upsampling, wie die Erweiterung des Abtastverhätnisses allgemein genannt wird, erfolgt über Interpolation oder der Anwendung einer Faltungsmaske unter Verwendung der umgebenden Pixelinformationen [22]. Für den Anwendungsfall im SAV wurde dieses Verfahren durch die Replikation

der vorhergehenden Chrominanzwerte vereinfacht. Durch den Upsample Vorgang entsteht ein dreimal 8-Bit großer Ausgangsdatenstrom, der die drei Informationen für jeden Bildpunkt enthält; allerdings wird so die Frequenz des Pixeldatenstroms auf 13,5 MHz halbiert. Der Konvertierungsvorgang berechnet aus den drei YCbCr-Werten einen entsprechenden 24-Bit RGB-Wert, dabei wird gleichzeitig der Wertebereich der Eingangsdaten auf, alle mit 8-Bit darstellbaren, 256 Werte für Rot, Grün und Blau angepasst. Die Erweiterung des Wertebereichs erfolgt, da durch die Integration der Synchronisationssequenzen in den Pixeldatenstrom der Eingangswertebereich der Y-Werte lediglich [16,235], der Wertebereich der Chrominanzwerte nur [16,240] ist.

Die Implementierung dieses Moduls als VHDL Pipelinestufe erfordert eine Umrechnung der Konvertierungsgleichungen nach Jack(2005) in Integer-Arithmetik [16]. Demnach ergeben sich folgende Konvertierungsgleichungen [22]

$$R = (298Y + 409C_R - 57065)/256 \tag{1}$$

$$G = (298Y - 100C_B - 208C_R + 34658)/256 \tag{2}$$

$$B = (298Y + 516C_B - 70894)/256 \tag{3}$$

Die Bildbearbeitungspipeline enthält neben der Darstellung der originalen Kameradaten einen zusätzliches RESULT ILLUSTRATION Modul, um Zwischenergebnisse der Fahrspurerkennung anzuzeigen. Diese zur visuellen Verifikation genutzten Bilddaten zeigen das projektiv transformierte Bild der Fahrbahn mit der ermittelten dynamischen Region-Of-Interest. Über einen Schalter kann das gewünschte Ausgabeformat der Bildbearbeitungskette gewählt werden.

Die Frequenz von 13,5 MHz des Pixeldatenstroms ist zu gering für die Darstellung auf standardisierten VGA Monitoren. Diese sind zumeist auf eine darstellbare Bildfrequenz von 60 - 80 Hz ausgelegt; mit 13,5 MHz Pixelfrequenz und ungefähr 450.000 Takten pro Bild[1] wird ein Bild aber lediglich mit 30 Hz ausgegeben.

Zur Verdoppelung der Bildfrequenz wird die Eigenschaft des Interlace Modus verwendet, dass innerhalb eines Frames zwei Teilbilder vorliegen. Dabei wird jeweils eine Bildzeile im DEINTERLACE Modul zwischengespeichert und mit der doppelten Frequenz zweimal ausgegeben. Diese Zeilenverdopplung hat in Bezug auf die dargestellten Bildinformationen für den Betrachter keine nachteilige Auswirkung (vgl. Abb. 3.3). Da im Interlace Modus die Teilbilder aus den geraden bzw. ungeraden Bildzeilen eines Gesamtbildes bestehen, sind alle Bildinformationen des Sichtfeldes im Datenstrom vorhanden.

Zur Ausgabe des Videodatenstroms werden abschließend aus den internen LVAL/FVAL

[1] Die hohe Anzahl ergibt sich zum einen aus den gültigen Pixeldaten (640x480), und aus den Blankinganteile, also Bereichen im Datenstrom in denen keine gültigen Pixeldaten anliegen.

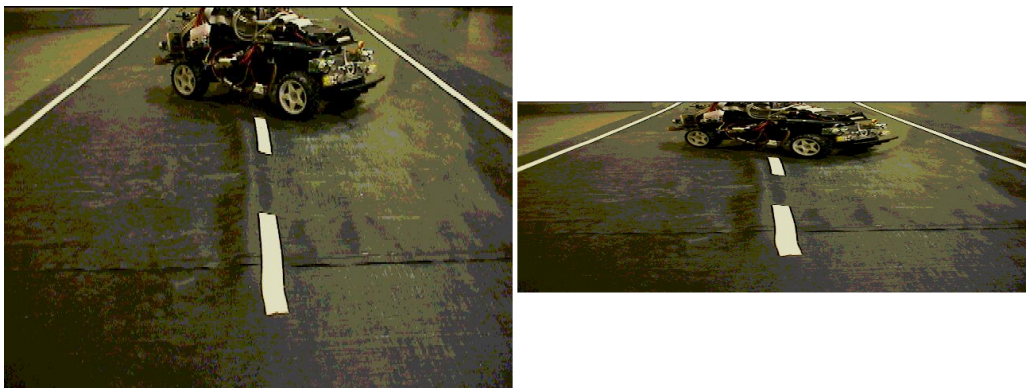


Abbildung 3.3 – Die Streckung des Bildes durch die Verdoppelung der Bildzeilen hat für den menschlichen Betrachter eine vernachlässigbare Auswirkung, da die Breite einer Bildzeile, abhänig von der Auflösung des Ausgabegeräts, im Bruchteil eines Millimeters liegt [17].

Synchronisationssignalen die entsprechenden HSYNC bzw. VSYNC Signale erzeugt; diese verwenden VGA Monitore zur Darstellung der gültigen Pixeldaten.

3.2 Fahrspurerkennung und Spurführung

Die Fahrzeugführung im autonomen Betrieb erfolgt durch die Detektion der Fahrspurmarkierungen aus den Kameradaten, um die Position des Fahrzeugs und des weiteren Fahrbahnverlaufs zu bestimmen. Dieser Vorgang erfolgt innerhalb der Bildverarbeitungskette in folgenden Teilschritten:

1. Vorverarbeitung der Kameradaten zur Datenreduktion und Hervorhebung der Fahrspurmarkierungen
2. Projektive Transformation des perspektivisch verzerrten Kamerabildes, um die Bilddaten in das Fahrzeugkoordinatensystem zu übertragen
3. Erkennung der Fahrspurmarkierung durch die Hough-Transformation
4. Anwendung von Spurführungsalgorithmen auf Grundlage der ermittelten Fahrzeugposition und des fortschreitenden Fahrbahnverlaufes

Die Anwendung von Spurführungsalgorithmen erfolgt aus der Interpretation des Kamerabildes, das Informationen über den vorausliegende Umgebung des Fahrzeugs in Form von Pixelwerten beinhaltet. Um diese Informationen als Grundlage für die Berechnung der Lenkwinkeleinstellung zu nutzen, müssen die Pixelkoordinaten in ein allgemein gültiges, metrisches Fahrzeugkoordinatensystem übertragen werden. In diesem Zusammenhang werden unterschiedliche Koordinatensysteme genutzt, auf deren Basis die schrittweise Transformation der Kameradaten vollzogen wird (vgl. Abb. 3.4):

- (X_V,Y_V) metrisches Fahrzeugkoordinatensystem
  Der Ursprung dieses Systems befindet sich auf der Hinterachse im Zentrum der Räder.
- (X',Y') Bildkoordinatensystem des perspektivisch verzerrten Kameradatenstroms
  Grundlage sind die Pixelkoordinaten, wobei der Ursprung die linke obere Ecke des Bildes bei Pixel (0,0) darstellt; der letzte Bildpunkt ist an Koordinate (640,240). Die Bilddaten sind in Bezug auf das Fahrzeugkoordinatensystem aufgrund der Kameraposition und des Neigungswinkels perspektivisch verzerrt.
- (X_P,Y_P) Bildkoordinatensystem des projektiv transformierten Datenstroms
  Analog zu (X',Y') sind Pixelkoordinaten Grundlage dieses Systems; die Bildinformationen sind allerdings in die Perspektive des Fahrzeugkoordinatensystems transformiert.
- (X_{ROI},Y_{ROI}) Koordinatensystem der ROI (*Region-Of-Interest*)
  Die Detektion der Fahrspurmarkierung wird auf einen Teilbereich des Gesamtbildes, der sogenannten Region-Of-Interest, begrenzt, um die Eingangsdaten für die diskrete Hough-Ebene im Hough-Transformations Modul gering zu halten.
- (X_H,Y_H) Hilfskoordinatensystem zur Ist-Wert Konstruktion für die Spurführung
  Das Hilfkoordinatensystem wird zur Konstruktion der Fahrzeugausrichtung verwendet. Dazu wird das Fahrzeugkoordinatensystem um den im (X_{ROI},Y_{ROI}) erkannten Winkel ϕ der Fahrspurausrichtung gedreht.

3.2.1 Vorverarbeitung des Bilddatenstroms

Die Pixeldaten durchlaufen eine adaptive Binarisierung und ein Erosionsfilter, um die Fahrbahnmarkierung vom Rest der Bildinformationen hervorzuheben und gleichzeitig eine Datenreduktion durchzuführen. Als Eingangsdaten werden die Y-Luminanzwerte der Kameradaten verwendet, die im darstellbaren 8-Bit Grauwertebereich [0,255] vorliegen. Die adaptive Binarisierung ermittelt pro Einzelbild die Grauwertmaxima und einen abhängigen Schwellwert, durch den die Pixeldaten binär interpretiert werden (vgl. Gl. 4). Im Gegensatz zur Binarisierung mit einem festen Schwellwert hat dieses Verfahren den Vorteil, dass unabhängig von den Licht- bzw. Schattenverhältnissen der Umgebung, die maximalen Helligkeitswerte eines Bildes erkannt und erhalten bleiben (vgl. Abb. 3.5). Zudem ist dieser Ansatz durch einfache Vergleichsoperationen innerhalb des Pixeldatenstroms zu lösen, ohne zeilenübergreifende Faltungsoperationen einzusetzen, wie bei anderen Kantendetektionsfiltern, beispielsweise den Sobel- oder Laplace-Operatoren.

$$f_{binarize}(g) = \begin{cases} 0, & falls \quad g > (g_{max} - (g_{max} \cdot 0.875)) \\ 1, & falls \quad g \geq (g_{max} - (g_{max} \cdot 0.875)) \end{cases} \tag{4}$$

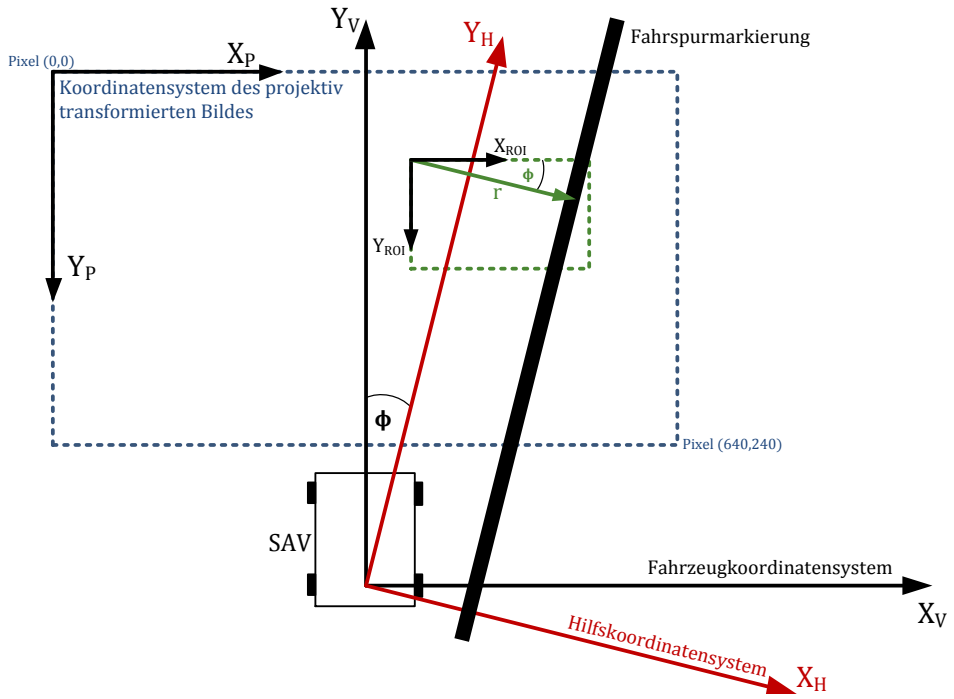


Abbildung 3.4 – Zur Anwendung von Spurführungsalgorithmen werden die Fahrspurinformationen aus dem Bildkoordinatensystem sukzessive in das Fahrzeugkoordinatensystem übertragen [26].

wobei g_{max} das fortlaufende Luminanzmaximum aus den Pixeln eines Einzelbildes darstellt und bei Beginn eines neuen Bildes zurückgesetzt wird.

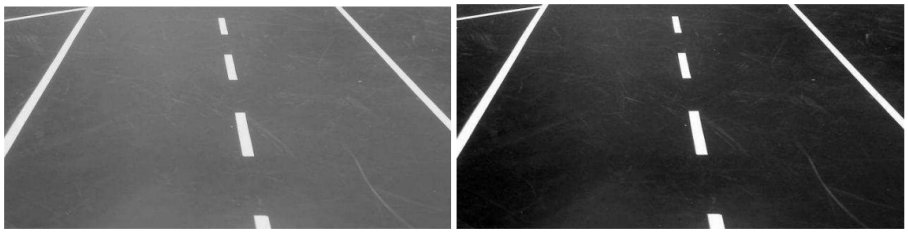


Abbildung 3.5 – Das Ergebnis der adaptiven Binarisierung zeigt die Unterdrückung von unregelmäßigen Helligkeitserscheinungen, die zum Beispiel durch Licht- bzw. Schatteneinflüsse aus der Umgebung entstehen.

Als Ergebnis der adaptiven Binarisierung zeichnet sich die Fahrspur deutlich von der restlichen Fahrbahn ab; für die Weiterverarbeitung durch die Hough-Transformation sind die Markierungen allerdings zu breit, was zu einer erhöhten Anzahl von Maxima bei der Geradenapproximation führt. Zur Verringerung der detektierten Bildpunkte wird eine morpholo-

gische Filterung auf den Pixeldatenstrom angewandt, wobei der Wert eines Pixels abhängig von den umgebenden Bildpunkten über eine Faltungsmaske bestimmt wird. In der Bilddatenvorverarbeitung wird eine 3x3 Faltungsmatrix mit Erosions-Operator eingesetzt, wodurch, zur Bereitstellung der Umgebungspixel, zwei Bildzeilen zwischengespeichert werden. Die Erosion bezeichnet die Suche nach einem lokalen Minimum und ist in Abbildung 3.6 beispielhaft dargestellt. Für den Fall des binären Bilddatenstroms bedeutet das, dass die neun korrespondierenden Pixel zur Anpassung des zentralen Pixelwertes durch einen logischen UND-Operator verknüpft werden [21].

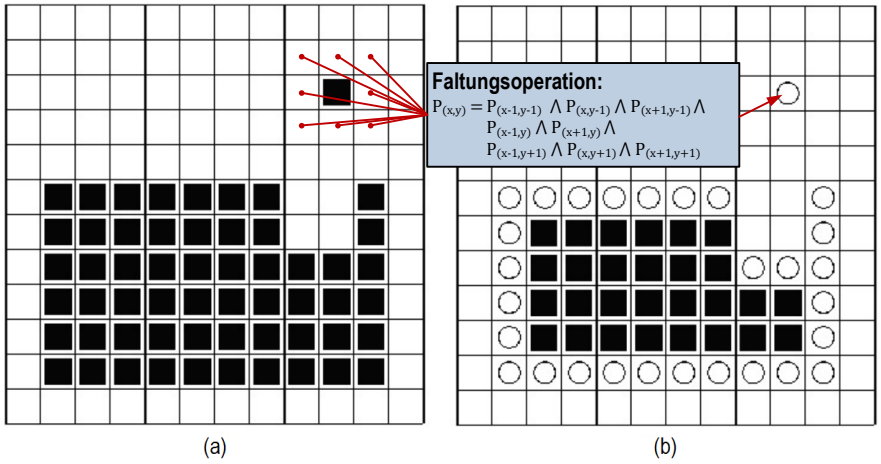


Abbildung 3.6 – Die Wirkungsweise der binären Erosion, wobei der Wert eines Pixels mit einer logischen UND-Verknüpfung der umliegenden Nachbarn erzeugt wird. (a) Eingangsbild, (b) Ergebnis der Erosion [21].

3.2.2 Projektive Transformation der Bilddaten

Die Daten des Videostroms liegen aufgrund der Kameraposition und deren Neigungswinkel in einer bestimmten Perspektive vor, die sich nicht mit der Ausrichtung des Fahrzeugkoordinatensystems (X_V,Y_V). Die Übertragung der Bildpunkte von der Bildebene(X',Y') in die Fahrzeugebene(X_P,Y_P) wird durch eine projektive Transformation vorgenommen (vgl. Abb. 3.7), um die perspektivische Verzeichnung zu entzerren. Zur Translation der Bildpunkte in die Fahrzeugebene wird der Zusammenhang von kartesischen Koordinaten und homogenen bzw. projektiven Koordinaten dargestellt. Durch die Erweiterung um eine zusätzliche Dimension wird ein Punkt des kartesischen Koordinatensystems $\vec{p}_k = (x_k, y_k)$ zu einem Punkt $\vec{p}_h = (x_h, y_h, 1)$ in homogenen Koordinaten.

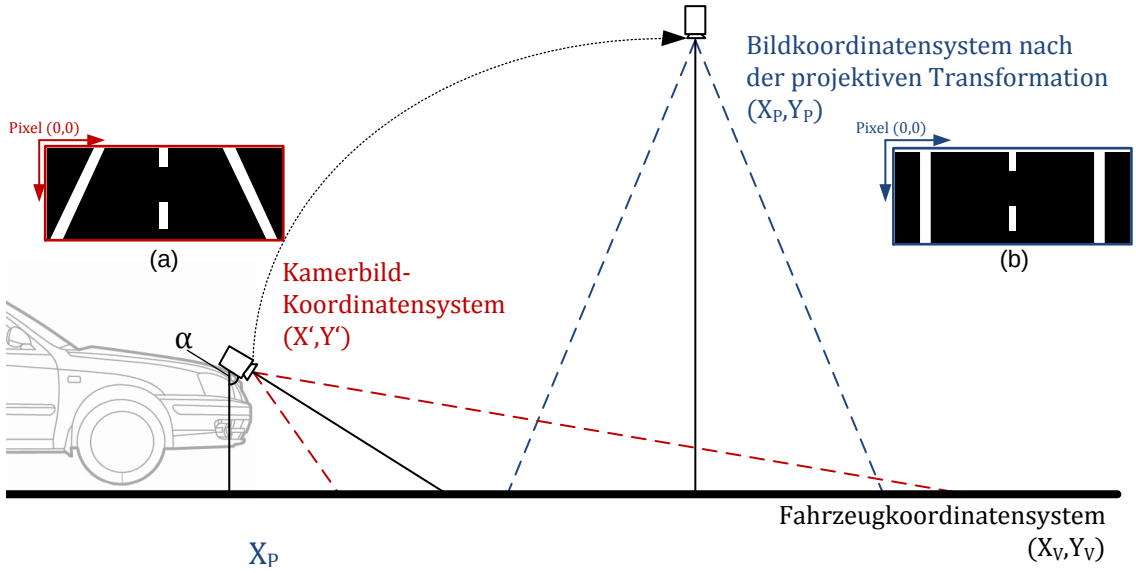


Abbildung 3.7 – Die Perspektive der Bilddaten ist durch die Kamerapositionierung auf einen Fluchtpunkt ausgerichtet (a); durch die PT wird sie in eine Fahrspuraufsichtsebene transformiert, um in das Fahrzeugkoordinatensystem übertragen werden zu können (b).

Ein Vektor ist homogen, wenn gilt:

$$\begin{pmatrix} x_h \\ y_h \\ 1 \end{pmatrix} \Leftrightarrow \begin{pmatrix} sx_h \\ sy_h \\ s \end{pmatrix} \quad \text{für alle } s \neq 0 \tag{5}$$

Analog wird ein Punkt in homogenen Koordinaten $\vec{p}_h$ durch die Division mit einem Skalierungsfaktor s in einen Punkt $\vec{p}_k$ in kartesischen Koordinaten transformiert.

$$\vec{p}_h = \begin{pmatrix} sx_h \\ sy_h \\ s \end{pmatrix} \rightarrow \vec{p}_k = \begin{pmatrix} x_h = \frac{x_h}{s} \\ y_h = \frac{y_h}{s} \end{pmatrix} \tag{6}$$

Die projektive Transformation der Ebene, eines zweidimensionalen Objektes, beschreibt eine umkehrbare Abbildung h von einer Projektionsebene π' in eine andere Projektionsebene π (vgl. Abb. 3.8). Sie ist eine lineare Transformation mit homogenen dreidimensionalen Vektoren und einer nichtsingulären 3x3 Matrix [20].

$$\vec{x_h} = H\vec{x_h'} \leftrightarrow \begin{pmatrix} x_h \\ y_h \\ 1 \end{pmatrix} = \begin{pmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{pmatrix} \cdot \begin{pmatrix} x_h' \\ y_h' \\ 1 \end{pmatrix} \tag{7}$$

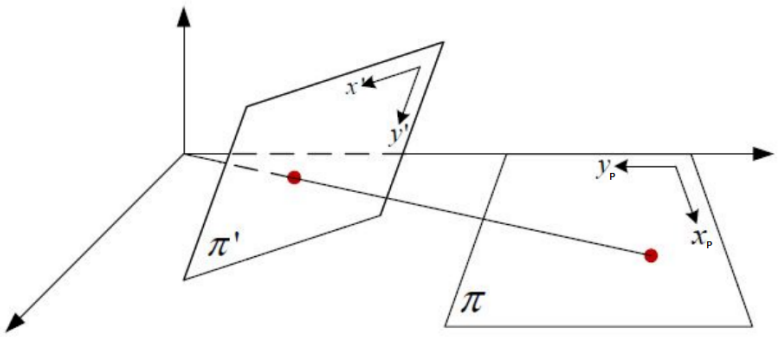


Abbildung 3.8 – Bei der Projektion in der Ebene π' auf die Projektionsebene π bleiben die Bildstrukturen projektiv äquivalent erhalten [21].

Die Transformationsmatrix H kann durch Multiplikation mit einem Skalierungsfaktor $s \neq 0$ verändert werden, ohne die eigentliche Projektion zu verändern. Eine Reduktion der unbekannten Matrixelemente auf acht erhält man durch die Division aller Elemente durch den Projektionsfaktor h_{33}.

$$\vec{x_h} = B\vec{x_h'} \leftrightarrow \begin{pmatrix} x_h \\ y_h \\ 1 \end{pmatrix} = \begin{pmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & 1 \end{pmatrix} \cdot \begin{pmatrix} x_h' \\ y_h' \\ 1 \end{pmatrix} \quad \text{, mit } b_{ij} = \frac{h_{ij}}{h_{33}} \tag{8}$$

Mit Gleichung 6 ergeben sich aus den homogenen Koordinaten (x_h, y_h) und einem Skalierungsfaktor von $s = 1$ die kartesischen Koordinaten (x_k, y_k):

$$x_k = \frac{x_h}{1} = \frac{b_{11}x_h' + b_{12}y_h' + b_{13}}{b_{31}x_h' + b_{32}y_h' + 1} \tag{9}$$

$$y_k = \frac{y_h}{1} = \frac{b_{21}x_h' + b_{22}y_h' + b_{23}}{b_{31}x_h' + b_{32}y_h' + 1} \tag{10}$$

Die angepasste Transformationsmatrix B wird abhängig zur Kamerapositionierung einmalig ermittelt und über die Gleichungen 9 und 10 auf die Bildpunkte des Videodatenstroms zur Laufzeit angewandt (vgl. Abb. 3.9).

Kalibrierung der perspektivischen Korrektur

Die Elemente der Transformationsmatrix B in Gleichung 8 werden abhängig vom Sichtbereich der Kamera bestimmt. Eine Änderung des Sichtbereiches, z.B. durch eine Veränderung des Neigungswinkels der Kamera, erfordert die Neukalibrierung von B. Die Ermittlung der Transformationsparameter $[b_{11}, ..., b_{32}]$ erfolgt durch eine Kalibrierungsmatte, die mit einer

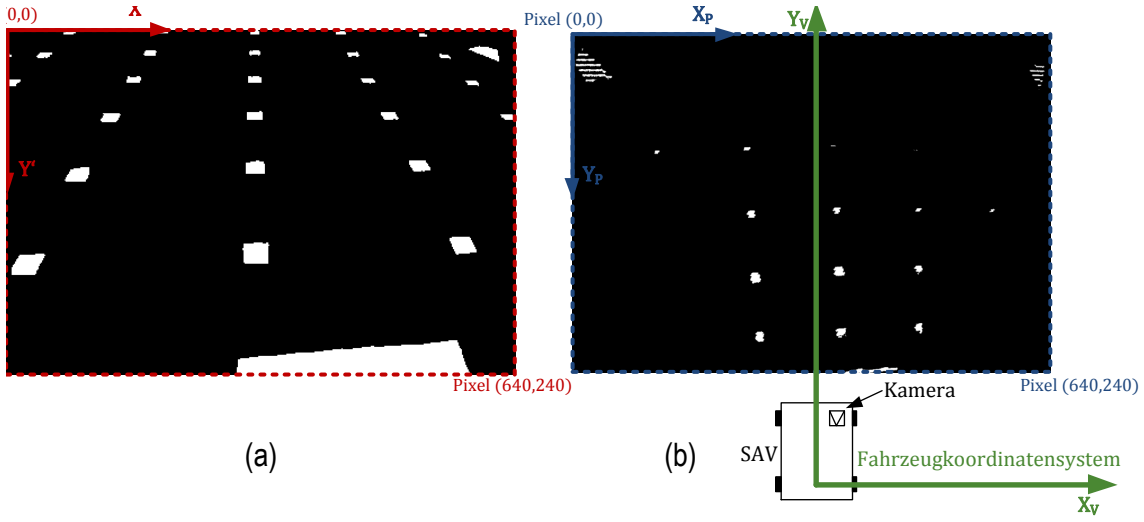


Abbildung 3.9 – Das perspektivisch verzerrte Kamerabild (a) wird über Gl. 9 und Gl. 10 in die Ebene des Fahrzeugkoordinatensystems transformiert (b); für (a) und (b) wurde die Kameraperspektive auf die Kalibrierungsmatte gerichtet.

Punktanordnung in metrischen Koordinaten der Fahrzeugebene versehen ist. Zur Aufstellung eines linearen Gleichungssystems werden die korrespondierenden Punkte aus dem Kamerabild und der Fahrzeugebene ermittelt. Die Gleichungen 9 und 10 werden so umgestellt, dass alle b_{ij} auf einer Seite stehen:

$$
\begin{aligned}
x_k = \frac{x_h}{1} &= \frac{b_{11}x'_k + b_{12}y'_k + b_{13}}{b_{31}x'_k + b_{32}y'_k + 1} \\
\Leftrightarrow x_k(b_{31}x'_k + b_{32}y'_k + 1) &= b_{11}x'_k + b_{12}y'_k + b_{13} \\
\Leftrightarrow x_k &= b_{11}x'_k + b_{12}y'_k + b_{13} - b_{31}x'_k x_k - b_{32}y'_k x_k
\end{aligned} \tag{11}
$$

und

$$
\begin{aligned}
y_k = \frac{y_h}{1} &= \frac{b_{21}x'_k + b_{22}y'_k + b_{23}}{b_{31}x'_k + b_{32}y'_k + 1} \\
\Leftrightarrow y_k(b_{31}x'_k + b_{32}y'_k + 1) &= b_{21}x'_k + b_{22}y'_k + b_{23} \\
\Leftrightarrow y_k &= b_{21}x'_k + b_{22}y'_k + b_{23} - b_{31}x'_k x_k - b_{32}y'_k x_k
\end{aligned} \tag{12}
$$

Aus den umgestellten Gleichungen 11 und 12 wird ein lineares Gleichungssystem erstellt, wobei für jedes korrespondierende Punktepaar jeweils eine Gleichung für die x- bzw. y-Komponente entsteht. Um ein Gleichungssystem mit acht Unbekannten b_{ij} zu lösen, werden mindestens acht Gleichungen verwendet. Die Lösung der acht b_{ij} ergibt sich aus $n \geq 4$

korrespondierenden Punktepaaren.

$$\begin{pmatrix} x'_{k_1} & y'_{k_1} & 1 & 0 & 0 & 0 & -x'_{k_1}x_{k_1} & -y'_{k_1}x_{k_1} \\ 0 & 0 & 0 & x'_{k_1} & y'_{k_1} & 1 & -x'_{k_1}x_{k_1} & -y'_{k_1}x_{k_1} \\ x'_{k_2} & y'_{k_2} & 1 & 0 & 0 & 0 & -x'_{k_2}x_{k_2} & -y'_{k_2}x_{k_2} \\ 0 & 0 & 0 & x'_{k_2} & y'_{k_2} & 1 & -x'_{k_2}x_{k_2} & -y'_{k_2}x_{k_2} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x'_{k_n} & y'_{k_n} & 1 & 0 & 0 & 0 & -x'_{k_n}x_{k_n} & -y'_{k_n}x_{k_n} \\ 0 & 0 & 0 & x'_{k_n} & y'_{k_n} & 1 & -x'_{k_n}x_{k_n} & -y'_{k_n}x_{k_n} \end{pmatrix} \cdot \begin{pmatrix} b_{11} \\ b_{12} \\ b_{13} \\ b_{21} \\ b_{22} \\ b_{23} \\ b_{31} \\ b_{32} \end{pmatrix} = \begin{pmatrix} x'_{k_1} \\ y'_{k_1} \\ x'_{k_2} \\ y'_{k_2} \\ \vdots \\ x'_{k_n} \\ y'_{k_n} \end{pmatrix} \tag{13}$$

Zur Aufstellung des Gleichungssystems wird das Sichtfeld der Kamera durch die Punktanordnung auf der Kalibriermatte ausgemessen. Abbildung 3.10 zeigt eine Kameraperspektive, bei der ein maximales Sichtfeld von 120 cm in x-Richtung und 103 cm in y-Richtung entsteht. Aus diesen Maxima ergeben sich die Transformationsskalierungen

$$mx = \frac{640px}{120cm} \tag{14}$$

bzw.

$$my = \frac{240px}{103cm} \tag{15}$$

Gleichzeitig werden die metrischen Koordinaten für die Darstellung als x',y'-Bildkoordinaten auf den Ursprung des Bildkoordinatensystems in der linken, oberen Bildecke [*Pixel(0,0)*] verschoben:

$$x'_{offset} = 320px \tag{16}$$

und

$$y'_{offset} = 240px \tag{17}$$

Der Vorgang zur Bestimmung der Kalibrierungsparameter erfolgt in den folgenden Teilschritten:

- Auswahl von freiwählbaren Referenzpunkten auf der Kalibriermatte und die Aufnahme ihrer metrischen Koordinaten
- Erfassung der Bildkoordinaten (X',Y') zu diesen Referenzpunkten aus dem Kamerabild. Dazu wird das Bild aus der Bearbeitungspipeline extrahiert und die Koordinaten mit einem Bildbearbeitungsprogramm manuell abgelesen.
- die korrespondierenden Koordinatenpaare, aus (X_k,Y_k) und (X'_k,Y'_k), werden in die Gleichung 13 eingesetzt

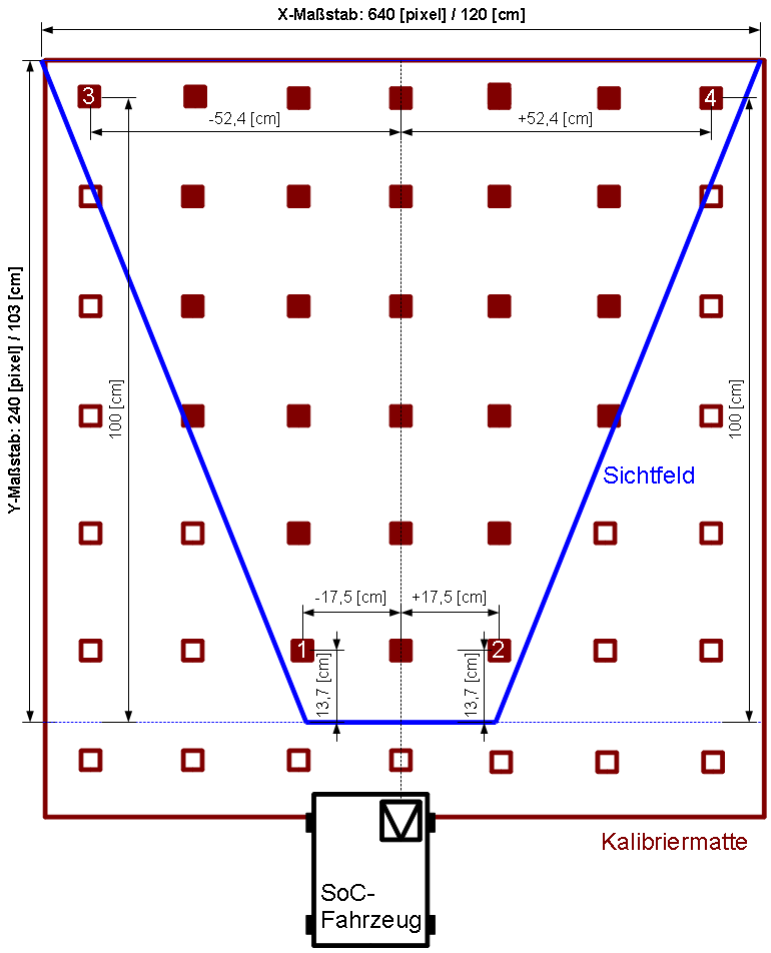


Abbildung 3.10 – Es werden vier Punkte im Kamerabild bestimmt, deren metrische Koordinaten in der Fahrzeugebene auf der Kalibriermatte gemessen werden können.

Exemplarisch heißt das für den eingezeichneten *Punkt 1* in Abbildung 3.10:

$$x_k = x_v \cdot mx + x'_{offset} = (-17,5cm) \cdot \frac{640px}{120cm} + 320px = 226,67px$$

$$y_k = y_v \cdot my + y'_{offset} = (13,7cm) \cdot \frac{240px}{103cm} + 240px = 208px$$

Die korrespondierenden Punktkoordinaten aus dem Kamerabild wurden manuell mit einem Bildbearbeitungsprogramm erfasst und haben die Bildkoordinaten:

$$x'_k = 75px \quad y'_k = 159px$$

Dieser Vorgang wird für die weiteren drei Punktepaare wiederholt, um die fehlenden Ele-

mente des Gleichungssystem zu ermitteln. Zur Lösung des Gleichungssystems steht ein Matlab Modul zur Verfügung, mit dem die Kalibrierungsparameter der Transformationsmatrix B berechnet werden (vgl. Kap. 5.1). Für das in Abbildung 3.10 beschriebene Sichtfeld sind folgende Werte berechnet worden (vgl. Tab. 3.1):

Transformationsmatrix B					
B_{11}	1,0850	B_{12}	3,9030	B_{13}	-30,2500
B_{21}	0,0338	B_{22}	4,0770	B_{23}	-34,0400
B_{31}	0,0001	B_{32}	0,0122	B_{33}	1,0000

Tabelle 3.1 – Kalibrierungsparameter der projektiven Transformation, die in Gleichung 9 und Gleichung 10 zur Entzerrung des Kamerabildes eingesetzt werden.

3.2.3 Hough-Transformation zur Fahrspurdetektion

Die Ermittlung der Fahrbahnmarkierung aus dem Pixeldatenstrom erfolgt durch das Verfahren der Hough-Transformation. Um die Speicheranforderungen gering zu halten, werden die Eingangsdaten zunächst auf einen bestimmten Teilbereich des Bildes durch eine dynamische ROI reduziert. Diese umfasst einen Bereich von 50x50 Pixeln und wird abhängig von den Koordinaten der zuletzt erkannten Fahrspurführung in x-Richtung auf dem Eingangsbild verschoben. Dieses Vorgehen verringert nicht nur das Datenvolumen, das durch die HT interpretiert werden muss, durch die Fokussierung auf einen festgelegten Zielbereich, werden Fehlinterpretationen von anderen Markierungen oder Umgebungseffekten vermindert.

Die HT beschreibt ein robustes Verfahren zur Detektion von geometrischen Figuren in vorverarbeiteten Gradientenbildern. Durch die Vorverarbeitung werden die Kanten des Eingangsbildes hervorgehoben und binarisiert, was einer Kategorisierung der Bildinforamtionen in Vorder- und Hintergrundpixel entspricht. Anschließend werden markante Parameter der Vordergrundpixel über eine Referenzstruktur in die Hough-Ebene übertragen, wobei geometrische Objekte als Punkte in der Hough-Ebene dargestellt werden (vgl. Abb. 3.11). Für die Fahrspurerkennung wird als Referenzobjekt die Gerade verwendet, welche durch die Geradengleichung eindeutig beschrieben ist:

$$y = m \cdot x + b \tag{18}$$

Diese Darstellungsform mit m als Steigung und dem Achsenabschnitt b, bietet sich nur bedingt für die Verwendung in der Fahrspurerkennung an, da das Fahrzeug in einem typischen Anwendungsfall parallel zur Fahrspur steht. Somit ist die Steigung der approximierten Fahrspurgeraden $m \to \infty$, was bei der Übertragung in die Hough-Ebene nicht darstellbar ist. Aus diesem Grund wird die Gerade durch die Hessesche Normalform beschrieben; diese

bezeichnet eine Gerade durch die Ursprungslotlänge r und dem Winkel ϕ zwischen Ursprungslot und x-Achse:

$$r = x \cdot \cos(\phi) + y \cdot \sin(\phi) \tag{19}$$

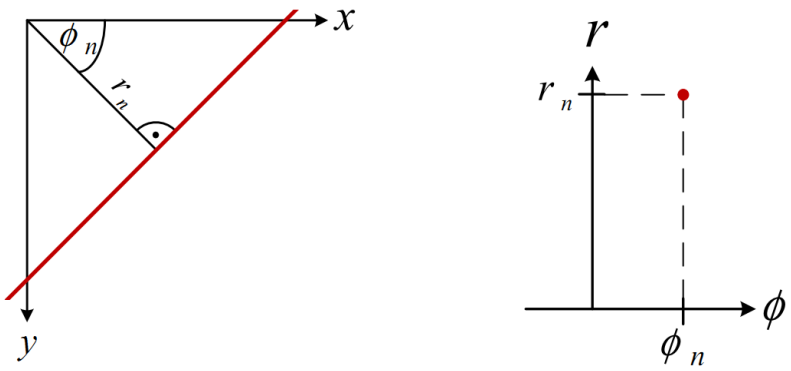


Abbildung 3.11 – Eine in Bildkoordinaten dargestellte Gerade wird mit der Übertragung in den Hough-Raum zu einem durch r und ϕ beschriebenen Punkt [20].

Zur Übertragung eines Bildpunktes in die Hough-Ebene werden die Parameter aller Geraden, die durch einen bestimmten Punkt A verlaufen, in Hessescher Normalform angegeben (vgl. Abb. 3.12). Dadurch entsteht in der Hough-Ebene eine sinusoidale Funktion, die den Bildpunkt charakterisiert. Die Übertragung von weiteren Bildpunkten erzeugt weitere Sinusoide, wobei zwei Bildpunkte A und B in x,y-Koordinaten eine gemeinsame Gerade g an der Stelle haben, wo sich ihre Abtragungen in der Hough-Ebene schneiden. Somit kann über das Schnittpunkt-Maximum in der Hough-Ebene identifiziert werden, welche Gerade die meisten Vordergundpixel zusammenfasst.

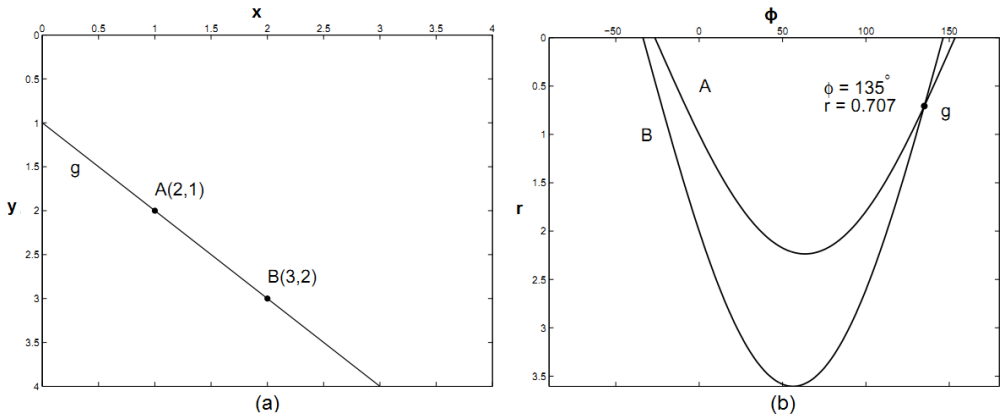


Abbildung 3.12 – Die Gerade g zweier Bildpunkte A und B in x,y-Koordinaten (a) wird in der Übertragung in die Hough-Ebene durch den Schnittpunkt ihrer sinusoidalen Funktionen identifiziert (b) [20].

Zur Approximation der Fahrspurführung wird in der HT eine Speichermatrix verwendet, in der das Auftreten der Geradenparameter mit den Abständen $r_i \in [r_{min}, r_{max}]$, mit $i \in \mathbb{Z}$ für den Winkelbereich von $\phi_j \in [\phi_{min}, \phi_{max}]$, mit $j \in \mathbb{Z}$ akkumuliert wird. Dabei wird die Deskretisierung der Hough-Ebene mit folgenden Einschränkungen nach Schneider,2011 und Mellert,2010 vorgenommen [26], [20]:

- der Winkelbereich $\phi \in [\phi_{min}, \phi_{max}]$ wird auf das Intervall von $-18°$bis $+18°$verkleinert
- das Winkelintervall $\Delta\phi$ für die Geradenhypothesen wird auf $2°$gesetzt
- die maximale Lotlänge r_{max} ist durch die Anwendung der HT auf den Bereich der ROI eingegrenzt

Durch diese Parameter wird die Gerade, welche die größte Anzahl an Vordergrundpixeln des ROI-Bereiches auf sich vereint, ermittelt und als erkannte Fahrspur an die Module zur Regelung der Fahrzeugführung in Hessescher Normalform übergeben.

3.2.4 Algorithmen der Fahrspurführung

Zur Fahrzeugsteuerung wird zunächst die approximierte Fahrbahnmarkierung in die Fahrzeugebene überführt, um im Fahrzeugkoordinatensystem (X_V,Y_V) die Regelung des Lenkwinkels über verschiedene Fahrspurführungsalgorithmen vorzunehmen. Dabei sind allgemein drei unterschiedliche Regler-basierte Ansätze üblich:

- Algorithmen die auf eine Ansteuerung eines berechneten Zielpunktes basieren; aus der Kenntnis der Fahrzeug- und Fahrspurpositionen wird ein Ansteuerungspunkt, LAP (*Look-Ahead-Point*), konstruiert, der als Grundlage für die Berechnung der Lenkbefehle dient.
- Beim regelungsbasierten Ansatz werden die Lenkfunktionen aus der Abweichung der Fahrzeugposition zu einer Sollgröße ermittelt. Dazu werden Sensordaten und Systemzustände verarbeitet, die durch Geschwindigkeitsmesser oder Abstandssensoren ermittelt werden.
- Hybride Algorithmen stellen Mischformen aus der Kombination der beiden oben genannten Ansätze dar.

Im SAV sind zwei Zielpunkt-basierte Ansätze implementiert; Follow-The-Carrot und Pure-Pursuit. Des Weiteren wird ein PD-Regler eingesetzt, um eine direkte Abstandsregelung zur Fahrspur durchzuführen, sowie ein hybrider Ansatz durch die Kombination von Pure-Pursuit und PD-Regler.

Grundlage der Konstruktion des LAP (*Look-Ahead-Point*) ist das Fahrzeugkoordinatensystem, die aktuelle Position der ROI, die Lage der detektierten Fahrspurmarkierung sowie der Sollabstand d zur Markierung und die LAD (*Look-Ahead-Distance*) (vgl. Abb. 3.13).

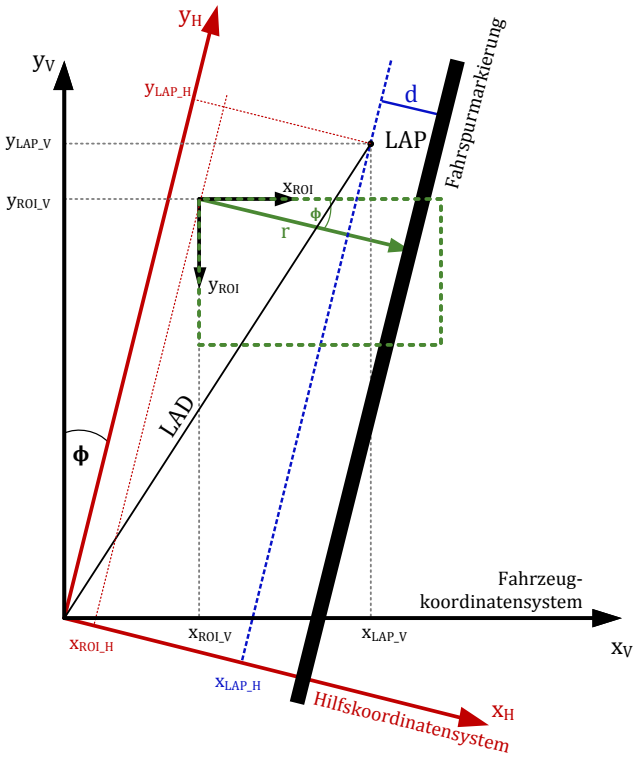


Abbildung 3.13 – Zur Konstruktion des LAP wird aus dem (X_V,Y_V) durch Drehung um den ermittelten Geradenwinkel ϕ das Hilfskoordinatensystem erzeugt. In diesem kann aus den Parametern X_{ROI_H}, dem Abstand r, dem Offset d und der LAD die Position des LAP bestimmt werden [26].

Dabei werden der Abstand r und der Winkel ϕ der erkannten Fahrspurmarkierung in Bezug auf den Koordinatenursprung von (X_{ROI},Y_{ROI}) angegeben. Zur Konstruktion wird das Hilfskoordinatensystem (X_H,Y_H) um den Winkel ϕ rotiert und somit an der approximierten Fahrspurmarkierung ausgerichtet.

$$\begin{pmatrix} x_{ROI_H} \\ y_{ROI_H} \end{pmatrix} = \begin{pmatrix} \cos(\phi) & \sin(\phi) \\ -\sin(\phi) & \cos(\phi) \end{pmatrix} \cdot \begin{pmatrix} x_{ROI_V} \\ y_{ROI_V} \end{pmatrix} \tag{20}$$

Die x-Komponente des LAP wird in Abhängigkeit zum Abstand r der detektierten Fahrspurgeraden, dem Abstandsoffset d und der x-Position der ROI ermittelt:

$$x_{LAP_H} = x_{ROI_H} + r - d \tag{21}$$

Aus x_{LAP_H} kann mit der vorgegebenen LAD der Look-Ahead-Point bestimmt werden

$$y_{LAP_H} = \sqrt{LAD^2 - x_{LAP_H}^2} \tag{22}$$

Zur Übertragung der Position des LAP ins Fahrzeugkoordinatensystem, wird der Punkt aus dem Hilfskoordinatensystem zurückrotiert:

$$\begin{pmatrix} x_{LAP_V} \\ y_{LAP_V} \end{pmatrix} = \begin{pmatrix} cos(\phi) & -sin(\phi) \\ sin(\phi) & cos(\phi) \end{pmatrix} \cdot \begin{pmatrix} x_{LAP_H} \\ y_{LAP_H} \end{pmatrix} \tag{23}$$

Follow-The-Carrot

Das Prinzip des Follow-The-Carrot Algorithmus besteht darin, den ermittelten LAP direkt anzusteuern. Dazu wird der Lenkwinkel

$$\alpha = \arctan\left(\frac{x_{LAP_V}}{y_{LAP_V}}\right) \tag{24}$$

bestimmt und das Fahrzeug dementsprechend ausgerichtet (vgl. Abb. 3.14 (a)). Seinen Namen hat dieser Algorithmus aus der sinnbildlichen Karotte an einer Angel, die zur Steuerung einem Esel vor die Nase gehalten wird [26]. Die Angel wird im vorliegenden Fall durch die LAD dargestellt. Der Vorteil des Follow-The-Carrot liegt in seiner einfachen Berechnung. Allerdings führt er gerade bei höheren Geschwindigkeiten in engen Kurven oder Kurvenkombinationen zu Aufschwing- und Auspendelverhalten [26].

Pure-Pursuit

Bei der Pure-Pursuit Methode wird Annäherung an den LAP auf einer vorgegebenen Kreisbahn mit der Krümmung γ angestrebt. Diese wird durch einen Rotationspunkt Z so gewählt, dass sie den Ursprung des Fahrzeugkoordinatensystems und den LAP tangiert (vgl. Abb. 3.14 (b)).

Der einzustellende Lenkwinkel α ergibt sich unter Berücksichtigung der Kinematik des Fahrzeugs, die hier durch ein Einspur-Fahrzeugmodell abstrahiert wird [20]. Dabei wird der konstante Achsabstand L_F zwischen dem lenkbarem Vorderrad und festem Hinterrad und der Abstand r_Z zur Fahrzeuglängsachse in die Berechnung aufgenommen.

$$\alpha = \arctan\left(\frac{L_F}{r_Z}\right) = \arctan\left(\frac{2 \cdot x_{LAP_V} \cdot L}{LAD^2}\right) \tag{25}$$

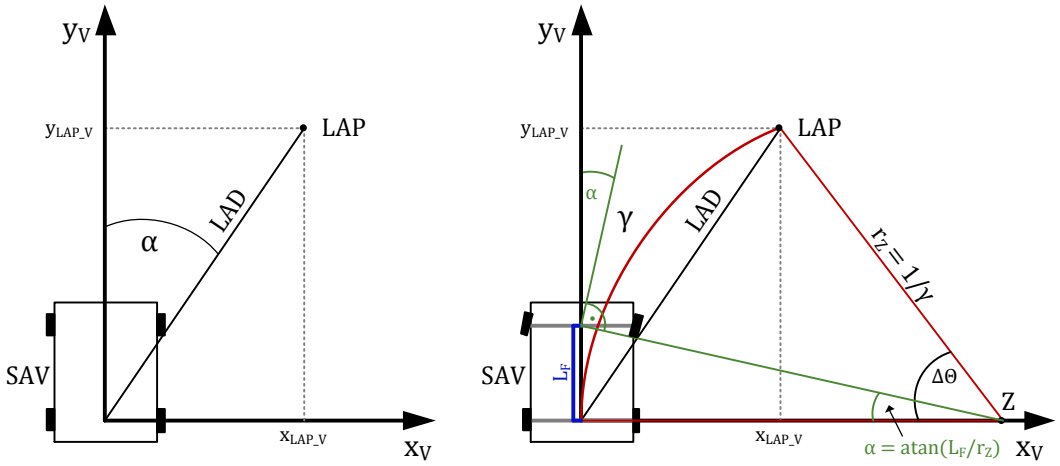


Abbildung 3.14 – Lenwinkelstrategien Follow-The-Carrot (a) und Pure-Pursuit (b); beim FTC wird direkt der Winkel α zur Ausrichtung angestrebt, die Ermittlung des Lenkwinkels bei der PP Methode erfolgt durch eine Annäherung an den LAP auf einer vorgegebenen Kreisbahn γ unter Berücksichtigung der kinematischen Eigenschaften eines Fahrzeugmodells [26].

Die Bestimmung von γ erfolgt über r_Z, den Kehrwert von γ. Das Rotationszentrum Z befindet sich auf der x-Achse und repräsentiert die Änderung der Fahrzeugausrichtung θ im Verhältnis zur zurückgelegten Wegstrecke s auf dem Kreisbogen:

$$\gamma = \frac{1}{r_Z} = \frac{d\theta}{ds} \tag{26}$$

Die Bestimmung der LAP-Koordinaten erfolgt über den Radius r_Z und die Änderung der Fahrzeugausrichtung $\Delta\theta$:

$$x_{LAP_V} = r_Z - r_Z \cdot \cos(\Delta\theta) = \frac{1-\cos(\Delta\theta)}{\gamma} \tag{27}$$

und

$$y_{LAP_V} = r_Z \cdot \sin(\Delta\theta) = \frac{\sin(\Delta\theta)}{\gamma} \tag{28}$$

Mit der Position des LAP aus 26 und 27 und der LAD lässt sich die Krümmung über

$$LAD^2 = x_{LAP_V}^2 + x_{LAP_V}^2 = \frac{2-2\cdot\cos(\Delta\theta)}{\gamma^2} = \frac{2\cdot x_{LAP_V}}{\gamma} \tag{29}$$

und der Umstellung nach γ ermitteln, um in Gleichung 25 verwendet zu werden:

$$\gamma = \frac{1}{r_Z} = \frac{2 \cdot x_{LAP_V}}{LAD^2} \tag{30}$$

Durch die Berücksichtigung der kinematischen Eigenschaften des Fahrzeugs werden bei der Pure-Pursuit Methode starkes Überschwingen bzw. lange Einschwingphasen vermieden. Eine weitere Verbesserung der Fahrzeugführung werden über die Kombination mit Reglerbasierten Ansätzen erzielt, indem zum Beispiel eine geschwindigkeitsabhängige Anpassung der LAD vorgenommen wird.

PD-Regler zur Abstandsregelung

Die Grundfunktion eines Reglers ist die Anpassung einer oder mehrerer physikalischer Größen an einen definierten Soll-Wert unter Berücksichtigung und Rückführung gemessener Ist-Größen in einem zeitkontinuierlich System. Der Standardregelkreis besteht dabei aus dem vorgegebenen Soll-Wert $w(t)$, dem Regler zur Kompensation der Regelabweichung $e(t)$, einer Regelstrecke, auf welche die Stellgröße $u(t)$ angewandt wird und der Rückführung der negativen Regelgröße $y(t)$ als Ist-Wert. Auf die Regelstrecke wirken darüber hinaus äußerliche Störgrößen $d(t)$, deren Auftreten den Einsatz von Regelkreisen motiviert. Abbildung 3.15 zeigt einen Standardregelkreis, der auf die Größen des SAV angepasst ist.

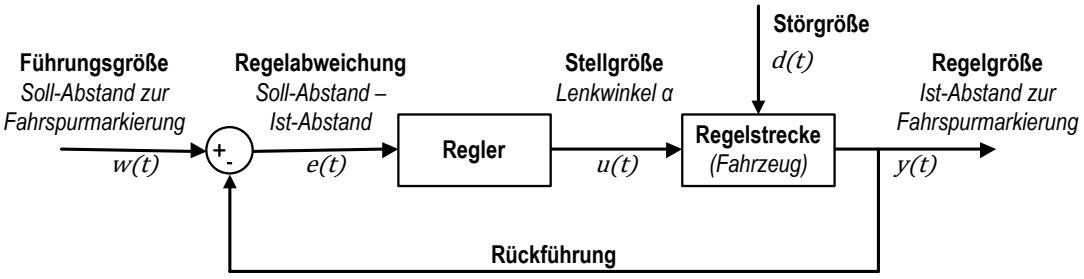
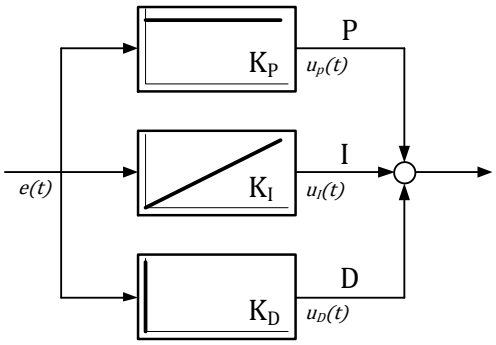


Abbildung 3.15 – Standardregelkreis zur Abstandsregelung des SAV zur Fahrspurmarkierung; über die Rückführung von $y(t)$ wird die Regelabweichung zu $w(t)$ bestimmt und im Regler die entsprechende Stellgröße $u(t)$ in Form des Lenkwinkels α ermittelt.

Die Ermittlung der Stellgröße $u(t)$ wird vom Regler mit proportionalem, integralem oder differentialem Verhalten vorgenommen (vgl. Abb. 3.16). Das Übergangsverhalten des P-Reglers beschreibt eine zum Eingangssignal $e(t)$ proportionale Stellgröße $u(t)$ die aufgrund des fehlenden Zeitverhalten unmittelbar anliegt. Zur zeitlichen Integration der Regelabweichung werden daher I-Regler eingesetzt, die $u(t)$ mit einer zeitlichen Komponente T_N gewichten. Diese Regelung ist genau, durch die zeitliche Näherung aber langsam. Ein differentialer Anteil wird über ein D-Glied in Verbindung mit den beide erstgenannten in die

Regelung aufgenommen. Das D-Glied reagiert nicht auf die Höhe der Regelabweichung, wodurch es nicht regelt, sondern aufgrund der Änderungsgeschwindigkeit der Regelabweichung differenziert. Durch Kombination dieser Anteile kann die Reaktion auf wechselnde Regelabweichungen optimiert werden.



(a)

$$u_P(t) = K_P \cdot e(t) \qquad \frac{u(s)}{e(s)} = K_P$$

$$u_I(t) = \frac{1}{T_N}\int_o^t e(\tau)d\tau \qquad \frac{u(s)}{e(s)} = \frac{1}{T_N \cdot s} = \frac{K_I}{s}$$

$$u_D(t) = T_V \frac{d}{dt} \cdot e(t) \qquad \frac{u(s)}{e(s)} = T_V \cdot s$$

(b) (c)

Abbildung 3.16 – Strukturbilder der P-, I-, und D-Glieder (a) für stetige Regler mit den entsprechenden Übergangsverhalten von $e(t)$ zu $u(t)$ (b) und ihre Übertragungsfunktionen (c).

Zur Abstandsregelung im SAV wird ein PD-Regler eingesetzt, dessen differentieller Anteil durch ein PT_1 Glied gedämpft wird. Somit kann ein ungewolltes Ausbrechen und Aufschwinken der Lenkfunktion auch bei starken Änderungen der Regelabweichung verhindert werden. Darüber hinaus ist der minimale und maximale Radeinschlag des SAV begrenzt, wodurch der Wertebereich der Lenkwinkelstellgröße zusätzlich eingeschränkt wird.

Zur Ermittlung der Stellgröße zu einem Zeitpunkt $u(k)$ wird zunächst die Übertragungsfunktion des PID-Reglers umgestellt. Für deren zeitkontinuierliche Darstellung gilt [26]:

$$G_R(s) = \frac{u(s)}{e(s)} = \left[K_P + \left(\frac{K_I}{s}\right) + K_D\left(\frac{Ns}{s+N}\right)\right] \tag{31}$$

Die Kamera liefert die Bilddaten mit 30 Bildern pro Sekunde, was für einer Abtastzeit T_s von 33,30 ms entspricht. Im diskretisierten PD-Regler wird hier der Integralanteil durch $K_I = 0$ kompensiert; gleichzeitig wird die Diskretisierung mit der z-Transformation durch Einsatz der Trapezregel vorgenommen [8]. So lautet die diskretisierte z-Übertragungsfunktion:

$$G_R(z) = \frac{u(z)}{e(z)} = K_P + K_D\left[\frac{N}{1 + N\frac{T_s}{2}\frac{1+z^{-1}}{1-z^{-1}}}\right] \tag{32}$$

Durch Umstellung nach $u(z)$ und Erweiterung des Hauptbruchs mit $(1-z^{-1})$ folgt:

$$u(z) = K_P e(z) + \frac{K_D N(1-z^{-1})}{1 - z^{-1} + N\frac{T_s}{2}(1+z^{-1})} \tag{33}$$

$$u(z) = K_P e(z) + \frac{K_D N(1-z^{-1})}{\left(N\frac{T_s}{2}+1\right)+\left(N\frac{T_s}{2}-1\right)z^{-1}} \tag{34}$$

Zur Vereinfachung wird $a = (N\frac{T_s}{2}+1) = \left(\frac{NT_s+2}{2}\right)$ und $b = (N\frac{T_s}{2}-1) = \left(\frac{NT_s-2}{2}\right)$ substituiert:

$$(a+bz^{-1})u(z) = K_P(a+bz^{-1})e(z) + K_D N(1-z^{-1})e(z) \tag{35}$$

Die Darstellung der zeitlichen Abhängigkeit von den vorangegangenen Zustandswerten der Stellgröße $u(z)z^-1$ und der Regelabweichung $e(z)z^-1$ zur Berechnung einer neuen Stellgröße $u(k)$ wird durch die Transition $x(z)z^-1 \rightarrow x(k-1)$ konkretisiert

$$a \cdot u(k) + b \cdot u(k-1) = K_P(a \cdot e(k) + b \cdot e(k-1)) + K_D N(e(k) - e(k-1)) \tag{36}$$

und nach $u(k)$ umgestellt:

$$u(k) = K_P\left(e(k) + \frac{b}{a}e(k-1)\right) + \frac{K_D N}{a}(e(k) - e(k-1)) - \frac{b}{a}u(k-1) \tag{37}$$

$$u(k) = \left(K_P + \frac{K_D N}{a}\right)e(k) + \left(K_P\frac{b}{a} - \frac{K_D N}{a}\right)(e(k-1) - \frac{b}{a}u(k-1) \tag{38}$$

Durch eine weitere Substitution erhält man die Dimensionierung des PD-Reglers

$$u(k) = c_1 \cdot e(k) + c_2 \cdot e(k-1) - c_3 \cdot u(k-1) \tag{39}$$

mit $c_1 = K_P + \frac{K_D N}{N\frac{T_s}{2}+1}$, $c_2 = \frac{K_P(N\frac{T_s}{2}-1)-K_D N}{N\frac{T_s}{2}+1}$ und $c_3 = \frac{N\frac{T_s}{2}-2}{N\frac{T_s}{2}+2}$.

Gleichung 39 dient als Grundlage zur Erstellung eines Algorithmus zur Berechnung der Stellgröße des PD-Reglers und wird in Form eines System Generator Moduls in der PATH FOLLOWING Komponente der Spurführung umgesetzt (vgl. Kap. 4.1.3).

3.3 Segmentierung der Abstandsdaten zur Objekterkennung

Die Erkennung von Hindernissen im Frontbereich des SAV wird durch die Auswertung von Abstandswerten vorgenommen. Diese werden als Rohdaten über eine serielle Schnittstelle von einem 2D-Laserscanner mit einer Frequenz von 10 Hz bereitgestellt. Der Laserscanner ist im Zentrum der Fahrzeugfront, 2 cm über der Fahrbahn montiert und wurde auf einen Scanbereich von 90°konfiguriert, der abhängig von der Fahrzeugausrichtung den Großteil der vorausliegenden Fahrbahn abdeckt (vgl. Abb. 3.17).

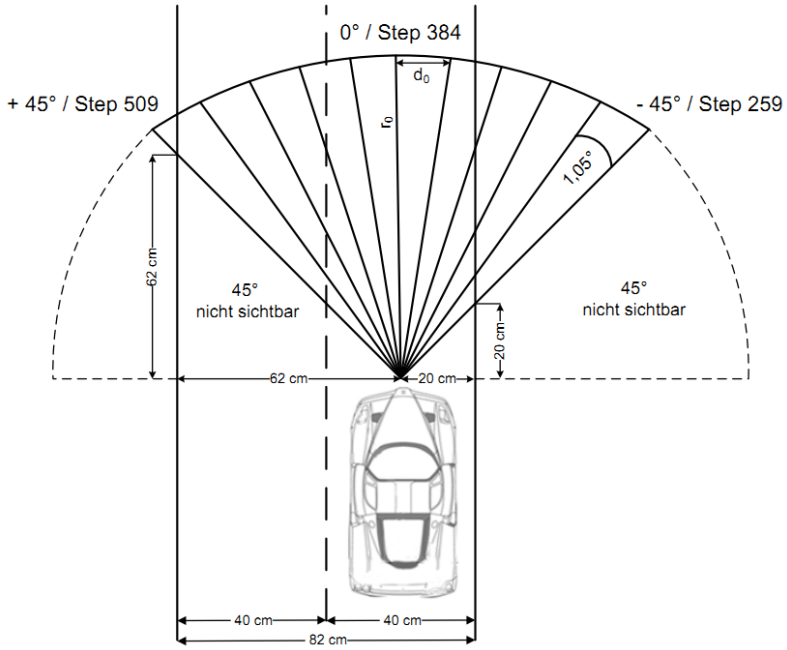


Abbildung 3.17 – Der 90°Scanbereich des Laserscanners zur Erfassung von Hindernissen auf der vorausliegenden Fahrbahn; die Abbildung zeigt das SAV auf einer, dem Carolo Cup Reglement entsprechenden Fahrbahn-Dimensionierung.

Der Laserscanner besitzt eine minimale Winkelauflösung von 0,35°; zur Abtastung des 90°Bereiches wird der Scanner mit einem *Cluster Count* von 3 konfiguriert, wodurch die Winkelauflösung auf 1.05°verringert wird und 84 Abstandswerte für einen Scanvorgang, als 12-Bit kodierte ASCII- Werte innerhalb des SCIP 2.0 Kommunikationsprotokolls, übertragen werden. Für die Objekterkennung werden die dekodierten Entfernungswerte mit den Winkelangaben zu zweidimensionalen Polarkoordinaten zusammengefasst und über Schwellwertverfahren segmentiert [32]. Durch Anwendung des „Successive Edge Following“ Algorithmus werden die Distanzen benachbarter Messwerte d_x sukzessive miteinander verglichen und zu Segmenten zusammengeführt, wenn die Differenz der Abstandswerte

einen festgelegten Schwellwert t nicht überschreitet [28]:

$$d_x \in SegmentX, \begin{cases} \text{TRUE}, & falls \quad (t \leq (d_{x-1} - d_x)) \\ \text{FALSE}, & falls \quad (t > (d_{x-1} - d_x)) \end{cases} \tag{40}$$

Die Bestimmung des Schwellwertes wird abhängig von der Entfernung des Objektes zum Laserscanner durch vier Schwellwerte dynamisch angepasst. Diese wurden aufgrund der Fahrbahn- und Hindernisspezifikation des Carolo Cup Reglements für einen Sichtbereich von 200 cm vor dem SAV bestimmt (vgl. Tab. 3.2).

Entfernung	Schwellwert
0 -50 cm	20 mm
50 -100 cm	35 mm
100 - 150 cm	50 mm
150 - 200 cm	70 mm

Tabelle 3.2 – Die Segmentierung der Messwerte erfolgt durch die dynamische Anwendung von Schwellwerten in Abhängigkeit zur Entfernung.

Die erkannten Segmente werden aus den 84 Abstandswerten erzeugt, Entfernungen die keinem Segment zugeordnet werden konnten, werden zur Datenreduktion verworfen. Ein Segment wird durch drei Punkte in Polarkoordinaten $P(\phi, d)$ beschrieben, wobei ϕ durch die Schrittweite des 90°Scanbereichs und d durch den Abstandswert gegeben ist. Die drei Punkte geben zu jedem Segement den linken und rechten äußeren Punkt, $R_{Sx}(\phi_R, d_R)$ und $L_{Sx}(\phi_L, d_L)$, sowie den Punkt mit der geringsten Distanz zum Segment $N_{Sx}(\phi_N, d_{minSx})$ an (vgl. Abb. 3.18).

Die Identifikation von Objekten erfolgt aus den segmentierten und reduzierten Messwerten. Dazu werden die Polarkoordinaten $P(\phi, d)$ in kartesische Koordinaten x_P, y_P umgerechnet:

$$x_P = d \cdot \cos(\phi) \tag{41}$$

$$y_P = d \cdot \sin(\phi) \tag{42}$$

Die drei Punkte des Segments werden nach Gleichung 41 und Gleichung 42 in kartesische Koordinaten übertragen und dem Steuerungsprozess zur Verfügung gestellt. Zusätzlich wird der Abstand d_LR zwischen dem linken und rechten Endpunkt des Segements, (x_L, y_L) und (x_R, y_R), mit

$$d_LR = \overline{LR} = \sqrt{(x_R - x_L)^2 + (y_R - Y_L)^2} \tag{43}$$

berechnet, um die maximale Breite des Objektes abschätzen zu können. Mit die Charakterisierung der Objekte über drei Koordinaten und des maximalen Breitenwertes kann der Steuerungsprozess die Objektinformationen in die Fahrspurführung aufnehmen. In Kapitel

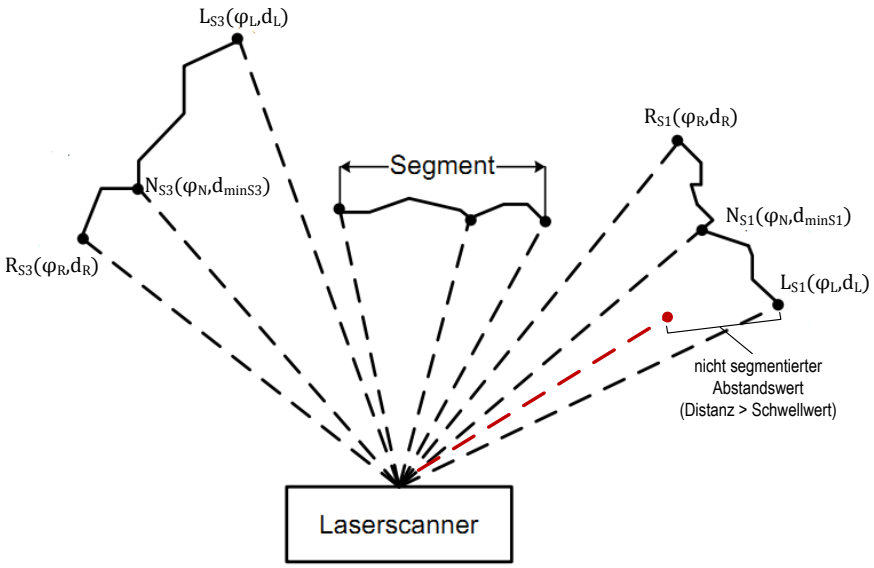


Abbildung 3.18 – 3-Punkt Segmentierung; die Abstandsdaten werden durch ein Schwellwertverfahren in Segmente zusammengefasst und durch drei Punkte beschrieben [32]

4.2.3 wird die Implementierung der Software-Komponenten zur Segementierung zur Objekterkennung beschrieben.

4 Modellierung und Implementierung der SAV Plattform

Die Implementierung des SAV Systems erfolgt basierend auf den Ergebnissen der Vorarbeiten in zwei Teilschritten; dem Zusammenführen der Fahrzeugregelungskomponenten in das Hardware-Beschleunigermodul SAV CONTROL und der Erstellung des DUAL-MICROBLAZE-SYSTEM zur Parametrierung der Regelungsprozesse und der Integration der Abstandssensorik.

4.1 Der IP-Core zur Regelung der Fahrzeugführung

Der SOC CONTROL IP-Core setzt sich aus einzelnen VHDL-Komponenten zusammen, die in einem strukturierten Aufbau die Aufnahme der Sensordaten, deren Verarbeitung und die Generierung von Ausgangssignalen zur Motor- und Lenkwinkelregelung realisieren. Die Regelungssysteme für den autonomen Betrieb sind in folgende Bestandteile gegliedert (vgl. Abb. 4.1):

- Ein **CLOCK MANAGER** zur Erzeugung unterschiedlicher Systemtakte auf Basis der Eingangsfrequenz des Kameradatenstroms von 27 MHz.
- Das **CAMERA-INTERFACE** erzeugt interne Synchronisationssignale, die die gültigen Pixeldaten im Kameradatenstrom kennzeichnen. Außerdem werden die Pixeldaten einerseits auf ihre Grauwertanteile für die Fahrspurerkennung reduziert und zweitens in das RGB-Format konvertiert, um auf einem VGA Monitor dargestellt werden zu können (vgl. Kap. 3.1).
- Das **PATHFINDER** Modul verbindet mehrere System Generator Blöcke zur Fahrspurdetektion und Berechnung des Lenkwinkels; dabei werden die Funktionskonzepte aus Kapitel 3.2 angewandt:

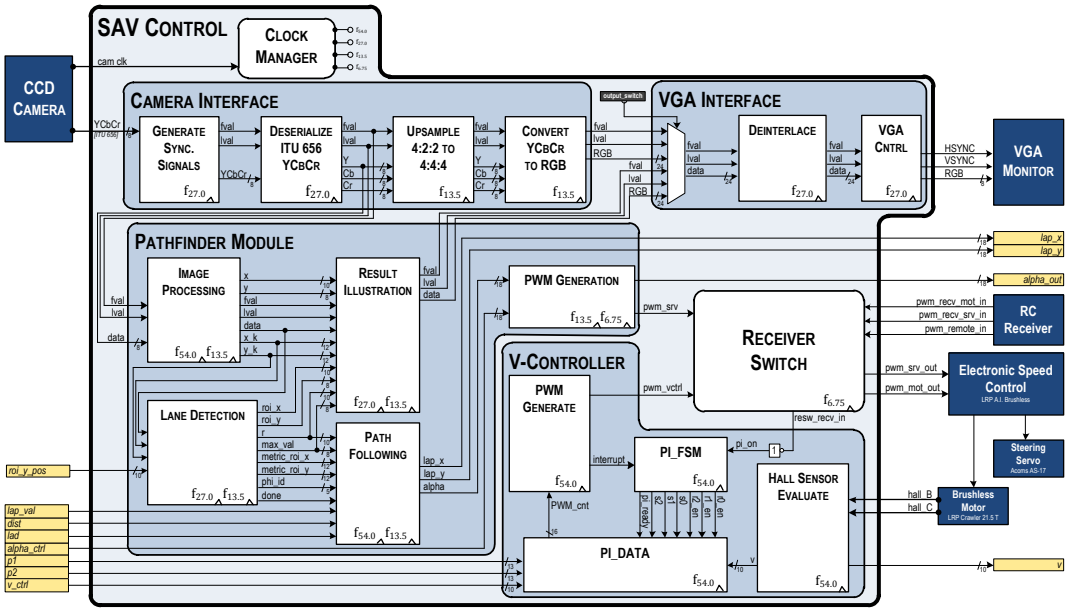


Abbildung 4.1 – Der SAV CONTROL IP-Core vereint zwei unabhängig voneinander agierende Regelungssysteme für die Fahrspurführung und die Geschwindigkeit, die über Software-Register (gelbe Kennzeichnung) vom MPSoC zur Laufzeit parametriert werden; gleichzeitig kann über das RECEIVER SWITCH Modul zwischen manuellem und autonomen Betrieb umgeschaltet werden.

- IMAGE PROCESSING: Vorverarbeitung der Bilddaten in Form einer adaptiven Binarisierung, eines Erusions Filters zur Kantenextraktion, der Generierung von Pixelkoordinaten und der projektiven Transformation der Pixeldaten in das Fahrzeugkoordinatensystem (vgl. Kap. 3.2.1 und Kap. 3.2.2).
- LANE DETECTION: Datenreduktion durch Einsatz einer dynamischen ROI und Erkennung der Fahrspurmarkierung durch die Hough-Transformation (vgl. Kap. 3.2.3).
- RESULT ILLUSTRATION: Bilddatenspeicher zur Darstellung des projektiv transformierten Kamerabildes und der erkannten Fahrspurmarkierung.
- PATH FOLLOWING: Berechnung des Lenkwinkels durch die Fahrspurführungsalgorithmen FTC und PP in Abhängigkeit von der Fahrzeugposition und der Fahrspurmarkierung, sowie die zusätzliche Implementierung eines Abstandsreglers, dessen Ergebnisse wahlweise in die Berechnung einfließen können.
- PWM GENERATION: Erzeugung eines PWM Signals auf Basis der ermittelten Lenkwinkelstellgröße zur Ansteuerung des Lenkservos.

- Das **VGA-INTERFACE** erhöht die Ausgangsfrequenz der Bilddaten durch eine Zeilenverdopplung und erzeugt die standardisierten Synchronisationssignale für VGA-Monitore.

- Das **V-CONTROLLER** Modul, zur Regelung der Fahrzeuggeschwindigkeit; die Ermittlung der aktuellen Geschwindigkeit erfolgt auf Basis von Hall Sensor Signalen. Der Entwurf und die Umsetzung der Geschwindigkeitsregelung ist in Anhang C beschrieben.
- Ein **RECEIVER SWITCH** Modul zum Umschalten zwischem manuellem und autonomen Betriebsmodus.

4.1.1 Die Bereitstellung der Systemtakte

Die Verarbeitung der Kameradaten erfolgt innerhalb der Pipelinestufen der SAV CONTROL Komponenten mit unterschiedlichen Taktfrequenzen. Dadurch können Teilberechnungen, wie die Divisionsstufen der Hough-Transformation, durch Multizyklusdatenpfade vorgenommen und somit Logikressourcen eingespart werden (vgl. Kap. 2.1). Die Bereitstellung der Taktfrequenzen im SAV CONTROL wird über die CLOCK MANAGER Komponente realisiert, der durch den Xilinx Core Generator als VHDL-Komponente erstellt wurde. Die Eingangsfrequenz von 27 MHz wird von der Kamera über eine separate Signalleitung bezogen. Diese ist über einen dedizierten *Clock Buffer Input Pin* direkt mit einem fest integrierten DCM (*Digital Clock Manager*) des FPGAs verbunden. Die Erzeugung der Systemfrequenzen erfolgt über zwei kaskadierte DCMs, um die Systemtakte erzeugen zu können (vgl. Tab. 4.1)

Signal	Frequenz	Verwendung
`U2_CLKFX_OUT`	**6,75 MHz**	Zur Erzeugung des PWM Signals im Pathfinder Module und der Abtastrate im Receiver Switch
`U1_CLKDV_OUT`	**13,5 MHz**	Takt des gesamten Kameradatenstroms nach der Deserialisierung in 24-Bit Pixelwerte
`U1_CLK0_OUT`	**27,0 MHz**	Pixelfrequenz der Kameradaten am Eingang der Bildverarbeitungspipeline und erforderliche Ausgangsfrequenz zur Darstellung auf einem VGA Monitor
`U1_CLK2X_OUT`	**54,0 MHz**	Implementierung von Multizyklusdatenpfaden innerhalb der Hough-Transformation und des Path Following Moduls; und der Systemtakt für die V-Controller Prozesskette

Tabelle 4.1 – Zur Erzeugung der vier Systemtakte werden zwei DCMs verwendet [48], was durch die Anordnung von jeweils zwei DCMs an einer I/O Bank des FPGAs keine langen Signalpfade erzeugt (vgl. Anhang E).

4.1.2 Aufnahme und Vorverarbeitung der Kameradaten

Die Aufnahme der Kameradaten in die Bildbearbeitungspipeline erfolgt innerhalb des CAMERA INTERFACES über vier Verarbeitungsstufen, die durch VHDL-Komponenten modular aufgebaut sind (vgl. Abb 4.2).

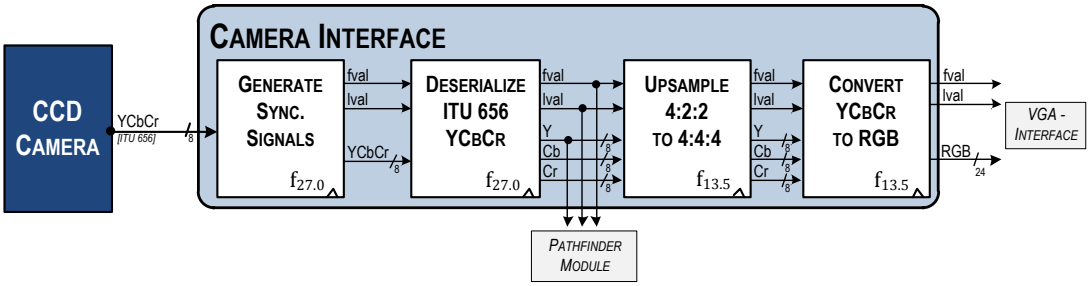


Abbildung 4.2 – Das CAMERA INTERFACE zur Synchronisation auf den Kameradatenstrom und der Vorverarbeitung der Pixeldaten für die PATHFINDER und VGA INTERFACE Module.

Zunächst werden die Synchronisationssignale `LVAL` und `FVAL` direkt aus dem Pixeldatenstrom erzeugt; diese werden in der weiteren Prozesskette zur Signalisierung von gültigen Pixelwerten verwendet. Der Kameradatenstrom enthält nach ITU Rec. BT656 [30] Kontrollsequenzen, mit denen die eigentlichen Pixeldaten von den Blankinganteilen unterschieden werden können. Durch die Interpretation der Sequenzen kann der Zeitpunkt eines gültigen Pixelwertes für eine entsprechende Zeile (`LVAL`) und eines gesamten Bildes (`FVAL`) angegeben werden. Die in Tabelle 4.2 dargestellten Kontrollsequenzen werden durch die VHDL Implementierung eines Zustandsautomaten aus dem Datenstrom erkannt. Die Timing-Übersicht für die Übertragung eines vollständigen Kamerabildes mit Blankingphasen und dem Auftreten der Kontrollsequenzen ist im Anhang E zu finden.

Durch das Auftreten von `SAV` nach `EAV_BLANK` wird der Beginn eines neuen Gesamtbildes vom Zustandsautomaten erkannt. Über einen horizontalen Zähler für die Pixel einer Zeile wird das `LVAL` Signal für 1280 Takte[1] aufrecht erhalten (*high*-aktiv). Das `FVAL` Signal wird entsprechend mit einem Zähler über alle 240 Zeilen eines Teilbildes generiert (vgl. Kap. 3.1 und [17]).

Im YCbCr-Datenstrom am Ausgang des Synchronisationsmoduls sind die Farbinformationen eines Pixels seriell angeordnet. Zur Darstellung der Kamerabilder auf einem VGA Monitor wird allerdings das parallele RGB-Format verwendet. Zudem ist für die Verarbeitung der Luminanzwerte zur Fahrspurerkennung ein kontinuierlicher Datenstrom erforderlich. Aus diesem Grund transformiert das DESERIALIZE Modul den seriellen Datenstrom in einen 24-Bit breiten parallelen Datenstrom, der aus jeweils 8-Bit breiten Informationen für

[1] Durch die Anordnung der Pixeldaten werden 640 Y-Werte, 320 Cb- und 320 Cr Werte pro Zeile übertragen.

		8-Bit Datenwort							
Bezeichnung	**Hex**	**MSB**	**6**	**5**	**4**	**3**	**2**	**1**	**LSB**
1. Wort	FF_H	1	1	1	1	1	1	1	1
2. Wort	00_H	0	0	0	0	0	0	0	0
3. Wort	00_H	0	0	0	0	0	0	0	0
4. Wort	XY_H	1	F	V	H	P_3	P_2	P_1	P_0
SAV	80_H	1	F	**0**	**0**	P_3	P_2	P_1	P_0
EAV	$9D_H$	1	F	**0**	**1**	P_3	P_2	P_1	P_0
SAV_BLANK	AB_H	1	F	**1**	**0**	P_3	P_2	P_1	P_0
EAV_BLANK	$B6_H$	1	F	**1**	**1**	P_3	P_2	P_1	P_0

F[0,1]	beschreibt das Field 1 bzw. Field 2 eines Frames im Interlaced-Modus
V[0,1]	Datum befindet sich im vertikalen Blankingbereich '1', sonst '0'
H[0,1]	identifiziert SAV bzw. EAV
P_3 - P_0[0,1]	Protection Bits als XOR Verknüpfung der F,H und V Signale zur Fehlererkennung

Tabelle 4.2 – Aufbau der SAV und EAV Kontrollsequenzen zur Signalisierung von gültigen Pixeldaten im Kameradatenstrom [30].

die Y-Werte, die Cb- und die Cr- Werte besteht. Das datenreduzierte 4:2:2 Abtastverhältnis des Datenstroms liefert nur für jedes zweite Pixel einen Cb- bzw. Cr- Wert, was den Pixeltakt auf 13,5 MHz verringert. Die Synchronisationssignale werden dementsprechend verzögert und stehen am Ausgang des Deserialisierungsmoduls mit den Y-Helligkeitswerten zur Verarbeitung im Fahrspurerkennungsmodul bereit. Die fehlenden Chrominanzinformationen werden zunächst mit Nullwerten aufgefüllt und innerhalb der parallelen Pixeldaten zusammen mit den Synchronisationssignalen an die nächste Pipelinestufe übergeben. Diese realisiert den Upsamplingvorgang indem die Chrominanzwerte der Vorgängerpixel dupliziert werden [22]. Am Ausgang des Upsample Moduls liegen für jedes Pixel gültige 8-Bit Farbinformationen an, die mit den, in den Gleichungen 1 bis 3 beschriebenen, Transformationen im CONVERT Modul in den RGB-Farbraum umgesetzt werden.

4.1.3 Die Kameradaten-basierte Fahrspurregelung

Die Ausrichtung des SAV auf die Fahrspur erfolgt durch die Auswertung des Pixeldatenstroms im PATHFINDER Modul. Dieses besteht aus einer Bildverarbeitungspipeline zur Aufbereitung der Pixel (IMAGE PROCESSING, der Fahrspurerkennung (LANE DETECTION), der Anwendung von Fahrspurführungsalgorithmen zur Lenkwinkelbestimmung (PATH FOLLOWING) und der Erzeugung des PWM Signals für den Fahrtensteller (PWM GENERATION) (vgl. Abb. 4.3). Zusätzlich sieht das Modul einen Bildspeicher zur Darstellung von Zwischenergebnissen der Fahrspurdetektion (RESULT ILLUSTRATION) vor. Die Bestandteile des PATHFINDER Moduls sind durch System Generator Blöcke realisiert und werden als generierte Netzliste in die Top Entity des SAV CONTROLS integriert.

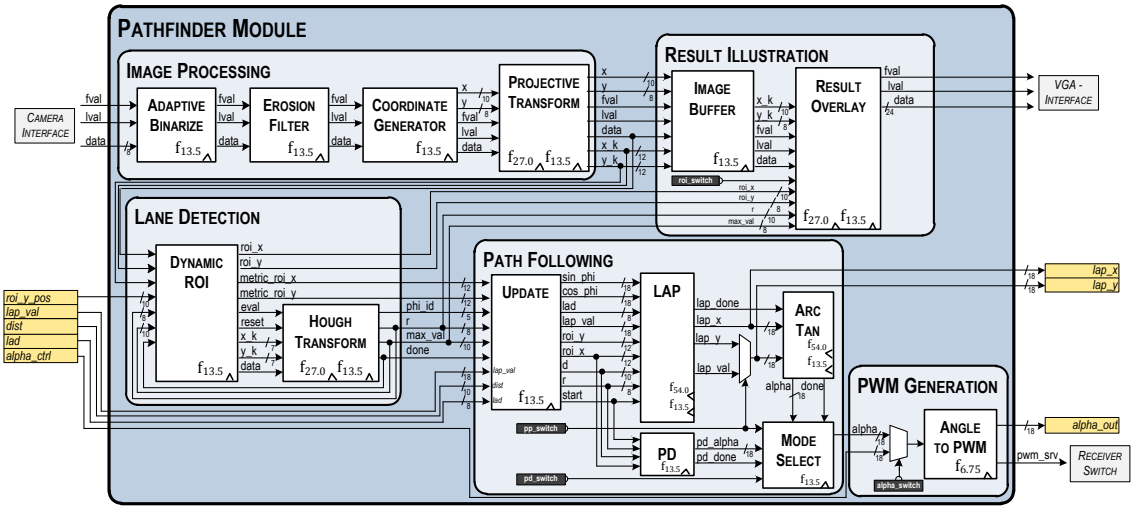


Abbildung 4.3 – Die Komponenten des PATHFINDER Moduls sind in einer vorsynthetisierten Netzliste durch den System Generator implementiert und werden in die VHDL-Top Entity des SAV CONTROL IPs integriert.

Die Kameradaten liegen am Eingang des PATHFINDER Moduls in der Pixelfrequenz von 13,5 MHz an und werden zunächst in Pipelinestufen verarbeitet. Die Approximation der Fahrspurgeraden durch das HOUGH TRANSFORM Modul erfolgt unabhängig von der Pixelfrequenz auf Basis der Bildfrequenz. Da die Pixeldaten nicht mehr sukzessive verarbeitet werden, bietet sich der Einsatz von Multizyklusdatenpfaden für die Implementierung der weiteren Komponenten an, um Arithmetik Elemente mehrfach zu verwenden und somit weniger Logik Resourcen zu beanspruchen.

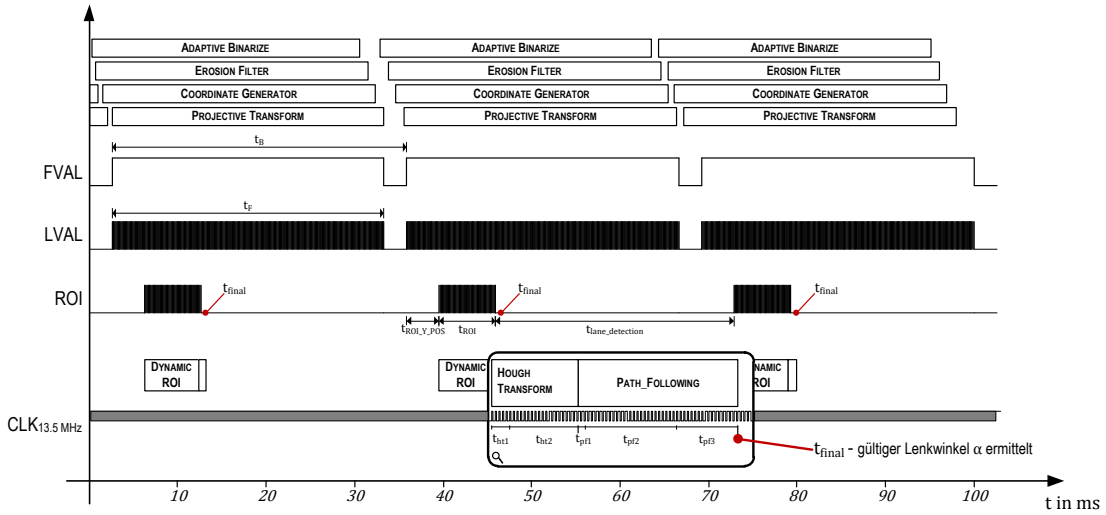


Abbildung 4.4 – Übersicht zum Timing des Pixeldatenstroms; die Verarbeitung der Pixeldaten eines Bildausschnitts werden der HT durch das DYNAMIC ROI übergeben, darauf erfolgt die Ermittlung der Lenkwinkelstellgröße unabhängig von der Pixelfrequenz in Multizykluspfaden zum *Resource-Sharing* (vgl. Kap. 5).

Vorverarbeitung der Kameradaten zur Fahrspurerkennung

Die Lumminanzwerte des Bilddatenstroms liegen in einem 8-Bit breiten Vektor zusammen mit den Synchronisationssignalen LVAL und FVAL und einer Pixelfrequenz von 13,5 MHz am IMAGE PROCESSING Modul an. Die Binarisierung erfolgt durch die in Gleichung 4 in Kapitel 3.2.1 vorgestellte Bestimmung des dynamisch angepassten Schwellwertes. Dazu werden die Grauwertmaxima eines Teilbildes in einem Register gespeichert, mit einer prozentualen Grauwertschwelle versehen und mit den Eingangspixelwerten verglichen (vgl. Abb. 4.5).

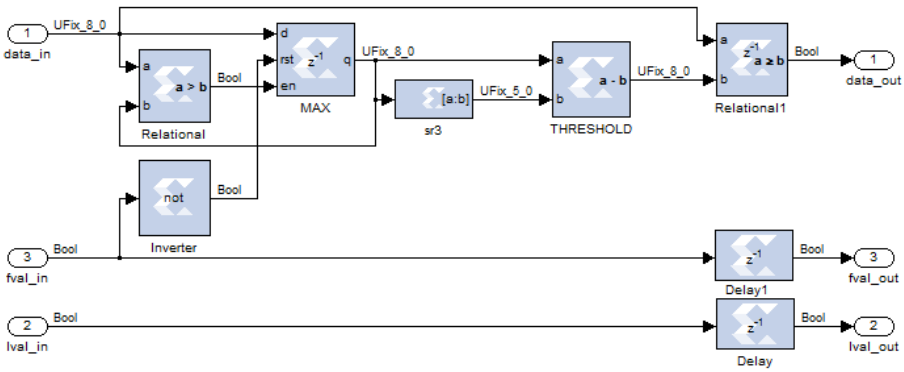


Abbildung 4.5 – Der adaptive Binarisierung; über das Shift Register **sr3** und die Subtraktion wird der Schwellwert für g_{max} bei jedem Systemtakt neu ermittelt.

Zur Berechnung der prozentualen Grauwertschwelle in der Form $(g_{max} - (g_{max} \cdot 0.875))$ in Integer-Arithmetik, wird der Subtrahend durch eine dreifache bitweise Rechtsverschiebung von g_{max} erzeugt, wodurch eine direkte Multiplikation umgangen wird. Die Rückführung mit dem Komparator bewirkt eine gleitende Max-Wertbestimmung für jedes Pixel das im Bild erfasst wird. Das Zurücksetzen des Schwellwertmaximums für jedes Einzelbild erfolgt über ein invertiertes FVAL Signal. Die Ermittlung des binären Ausgangswertes erfolgt innerhalb eines Taktes.

Eine Verringerung der markanten Pixelwerte nach der Binarisierung wird über ein Erosionsfilter vorgenommen. Die Berechnung des neuen Pixelwertes wird als Faltungsoperation auf den laufenden Bilddatenstorm angewandt (vgl. Abb. 3.6 und [17]). Dazu werden zwei Zeilen eines Bildes aus dem Pixelstrom zwischengespeichert und der Wert des aktuellen Pixels unter Berücksichtigung der umgebenden Pixelwerte ermittelt. Die Erosion wird im binären Datenformat durch eine logische UND-Verknüpfung realisiert (vgl. Abb. 4.6). Das Ausgangssignal des EROSION FILTER Moduls wird abhängig von der Speicherung von drei Pixelwerten pro Zeile und der logischen Verknüpfung um vier Takte verzögert, wodurch die Synchronisationsignale über Delay Blöcke zum Laufzeitausgleich geführt werden.

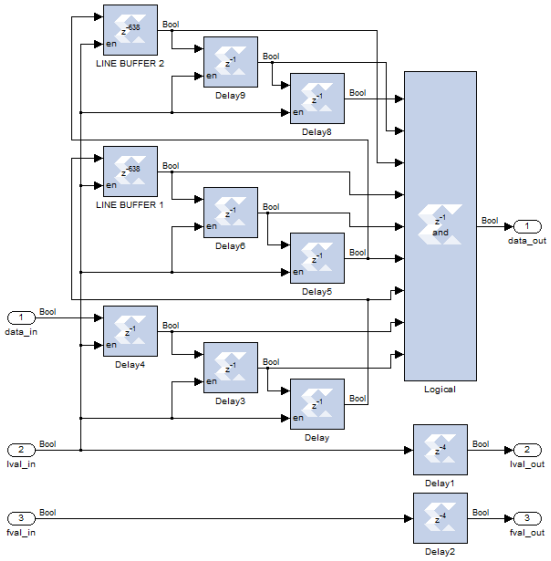


Abbildung 4.6 – Implementierung des Erosions-Filters mit zwei LINE BUFFER Elementen zur Speicherung von jeweils einer gesamten Bildzeile, die zusammen mit den Registern die neun Umgebungspixel zur Ermittlung des neuen Pixelwertes bereithalten.

Nachdem die Pixeldaten binär und reduziert vorliegen, werden für jedes Pixel durch den COORDINATE GENERATOR Bilddatenkoordinaten erzeugt, um die Pixelinformationen zur Auswertung der Fahrspurführung zu verwenden. Diese Auswertung erfolgt in den folgenden Teilschritten (vgl. Kap. 3.2.2):

- Erzeugung von Bildkoordinaten für jedes Pixel des Kameradatenstroms
- Projektive Transformation der Kamera-Bildkoordinaten in Bildkoordinaten der Fahrzeugperspektive
- Übertragung dieser Bildkoordinaten in das metrische Fahrzeugkoordinatensystem
- Erzeugung der Lenkwinkelstellgröße α auf Basis von Fahrspurführungsalgorithmen

Die Erzeugung der Bildkoordinaten basiert auf einem Zustandsautomaten, der die Synchronisationssignale interpretiert und mit der Systemfrequenz von 13,5 MHz für jedes Pixel eine Koordinate als Ausgangswerte `x_out` und `y_out` erzeugt. Durch Zähler werden die x-Koordinaten als 10-Bit breites Ausgangssignal und die y-Koordinaten als 8-Bit breiter Vektor generiert. Somit sind Wertebereiche für `x_out` bis 640 und für `y_out` bis 240 darstellbar. Grundlage des COORDINATE GENERATOR Moduls ist dabei ein MCode System Generator Block, der den Zustandsautomaten als Matlab Funktion *pixcount.m* realisiert (vgl. Listing 4.1).

```
[...]
  case WORKING %continue incrementing x_reg until lval is false
    if lval_reg == true && fval_reg == true
        x_reg = x_reg + 1;
        state = WORKING;
    elseif lval_reg == false && fval_reg == true
        state = LINEDONE;
    else
        state = WAITING;
    end
[...]
```

Listing 4.1 – Auschnitt aus der *pixcount.m* Matlab Funktion, die die Bildkoordinaten mit einer Zähler-FSM-Kombination als MCode Block im COORDINATE GENERATOR Modul erzeugt.

Die Bildkoordinaten bilden die Eingangsgrößen der projektiven Transformation (vgl. Kap. 3.2.2). Dazu steht mit dem PROJECTIVE TRANSFORM Modul ein System Generator Block zur Verfügung, der mit den acht Kalibrierungsparameter aus Tabelle 3.1 die transformierten Pixelkoordinaten nach Gleichung 9 und Gleichung 10 berechnet. x_k und y_k werden dabei in Teilschritten erzeugt, bei denen die beiden Dividenden $(b_{11}x'_h + b_{12}y'_h + b_{13})$ und $(b_{21}x'_h + b_{22}y'_h + b_{23})$ sowie der Divisor $(b_{31}x'_h + b_{32}y'_h + 1)$ nebenläufig ausgewertet werden (vgl. Abb. 4.7 (a)).

Zur Bestimmung der projektiven Koordinaten wird der Divisionsschritt durch eine Multiplikation mit dem Divisorkehrwert realisiert, wobei der Divisionsschritt für

$$\frac{1}{b_{21}x'_h + b_{22}y'_h + b_{23}} \tag{44}$$

mit einem erhöhten Systemtakt von 54 MHz erfolgt (vgl. Abb. 4.7 (b)). Am Ausgang der PROJECTIVE TRANSFORM werden somit die Pixelkoordinaten des urspünglichen Kamerabildes `x_out` und `y_out`, die projektiv transformierten Pixelkoordinaten `x_k_out`

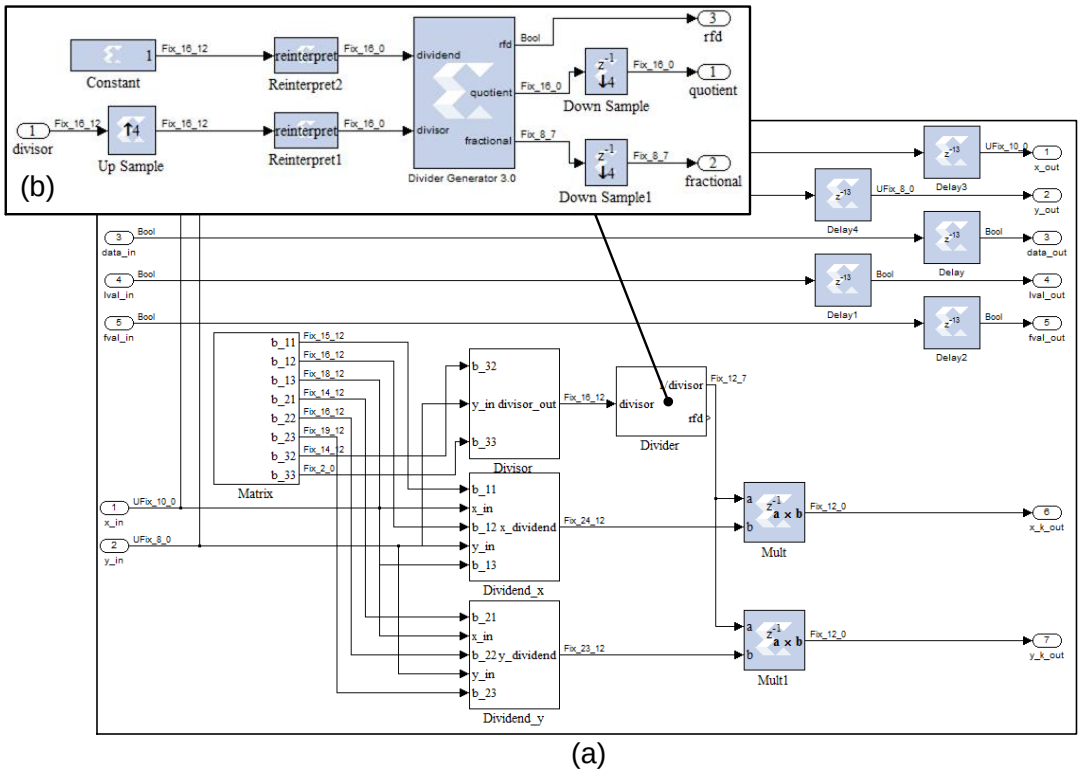


Abbildung 4.7 – PROJECTIVE TRANSFORM Modul (a) in dem Gl. 9 und Gl. 10 mit einem auf 54 MHz übertakteten Dividierer Block (b) realisiert werden. Der Pixelwert, die x',y' Koordinaten und die Synchronisationssignale werden mit Laufzeitausgleich weitergeleitet.

und `y_k_out`, der Pixelwert `data_out` sowie die Synchronisationssignale `fval_out` und `lval_out` mit 13,5 MHz bereit gestellt.

Erkennung der Fahrspurmarkierung

Die Approximation der Fahrspurmarkierung setzt sich innerhalb des LANE DETECTION Moduls aus zwei Teilschritten zusammen (vgl. Abb. 4.3). Dazu wird die Auswertung der Bildinformationen auf einen dynamisch angepassten Bildausschnitt begrenzt und dieser in einem zweiten Schritt durch das Verfahren der Hough-Transformation auf das maximale Auftreten von r, ϕ Parameterpaaren durchsucht (vgl. Kap. 3.2.3).

Das DYNAMIC ROI Modul passt die ROI in ihrer x-Ausrichtung zur Laufzeit an; dazu wird der Abstand `r` zur erkannten Fahrbahnmarkierung aus dem HOUGH TRANSFORM Modul zurückgeführt. Die Eingangssignale des Moduls bestehen darüber hinaus aus den projektiv transformierten Koordinaten und dem Pixelwert. Weiterhin signalisieren die Signale `max_val` eine erfolgreiche Detektion im HT Modul und das `done` Signal das Ende

der Berechnung. Zusätzlich kann für Erprobungsfahrten die y-Position der ROI mit dem `roi_y_pos` Signal zur Fahrtzeit verändert werden. Die Implementierung der dynamischen ROI erfolgt durch die Matlab Funktion *roi_fsm.m*, die über einen MCode Block in das Gesamtsystem eingebunden ist. Die Funktion realisiert einen Zustandsautomaten zur Anpassung der ROI und Erzeugung der Ausgangssignale für die Hough-Transformation (vgl. Abb. 4.8).

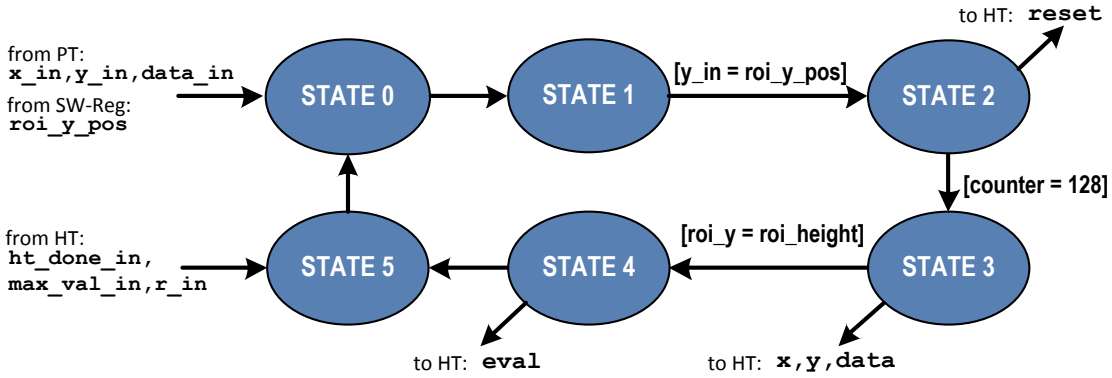


Abbildung 4.8 – Das Zustandsdiagramm zur dynamischen ROI; State 1 wird verlassen, sobald die y-Werte der Pixeldaten der ROI y-Position entsprechen; nachdem alle Bilddaten im Bereich der ROI übertragen wurden, erzeugt das `eval` Signal die Maximumsuche in der HT; sobald diese abgeschlossen ist, kann durch r die ROI neu ausgerichtet werden.

Die Anpassung der x-Ausrichtung der ROI erfolgt abhängig vom Abstand zur erkannten Fahrspurmarkierung r und der roi_width von 50 Pixeln:

$$roi\_x_p = roi\_x_p - roi\_width/2 + r \tag{45}$$

Darüber hinaus erfolgt die Umrechnung der projektiv transformierten Bildkoordinaten (X_P,Y_P) in die metrischen Koordinaten des Fahrzeugkoordinatensystem (X_V,Y_V) in der *roi_fsm.m* Funktion (vgl. Listing 4.2). Mit der Umrechnung der ROI Koordinaten in das metrische System wird die in der Hough-Transformation erkannte Gerade in das Fahrzeugkoordinatensystem zur Implementierung von Fahrspurverfolgungsalgorithmen überführt. Die Umrechnung erfolgt mit den Skalierungsparametern mx für die x- und my für die y-Koordinaten, der Offsets zur Verschiebung des Koordinatenursprungs ins Zentrum des unteren Bildrandes und dem Versatz der Kamera auf der Fahrzeugplattform zum Ursprung des Fahrzeugkoordinatensystems (vgl. Kap. 3.2.2). Demnach ergeben sich für die metrischen Koordinaten roi_x_{metric}, roi_y_{metric}:

$$roi\_x_{metric} = (roi\_x_p - x_{offset})/mx) - cam\_pos_x \tag{46}$$

und

$$roi\_y_{metric} = y_{offset} - (roi\_y_p/my) - cam\_pos_y \tag{47}$$

```
[...]
%ROI in Bildkoordinaten für die Darstellung in der Result Illustration ausgeben
roi_x_pos = roi_x;
roi_y_pos = xfix({xlUnsigned, 8, 0}, roi_y_in);
%projektiv transformiert Bildkoordinaten ausgeben
x_out = x;
y_out = y;
%Zuweisung der Daten- und Steuer- Ausgangssignale
data_out = data;
eval_out = eval;
reset_out = reset;
%Festlegung von Konstanten und Startwerten
size_x = 640;
roi_height = 50;
roi_width = 50;
roi_x_init = 295;
cam_pos_x = 27.0;
cam_pos_y = 5.5;
mx = 640/120;
my = 240/103;
%Berechnung der metrischen Koordinaten
roi_metric_x_pos = xfix({xlSigned, 12, 4}, (roi_x_pos - size_x/2)/mx) - cam_pos_x;
roi_metric_y_pos = xfix({xlSigned, 12, 4}, 103 - roi_y_pos/my) - cam_pos_y;
[...]
```

Listing 4.2 – Die *roi_fsm.m* Matlab Funktion reduziert den Bildbereich für die Hough-Transformation und übernimmt die Umrechnung von Bildkoordinaten in das metrische Fahrzeugkoordinatensystem.

Die Implementierung der Hough-Transformation übernimmt die Bildkoordinaten und den Pixelwert aus der dynamischen ROI und ermittelt aus diesen eine Gerade, die den Fahrspurverlauf wiederspiegelt. Die HT verläuft in folgenden Teilschritten:

- Die FSM des DYNAMIC ROI Moduls signalisiert der HT eine neue Berechnung über das `reset` Signal, worauf alle Akkumulatoren Register gelöscht werden.
- Darauf werden für die Koordinaten eines hellen Bildpunktes die Lotlängen r für alle diskreten Winkel Φ zwischen -18°und +18°in 2°-Schritten nach Gleichung 19 berechnet.
- Den Ergebnissen r wird ein Offset $|r_{min}|$ angefügt, wodurch diese als Adressen für die Akkumulatoren verwendet werden.
- Der mit dem ermittelten r adressierbare Akkumulator wird erhöht.
- Nachdem alle Bildpunkte transformiert wurden, wird mit dem `eval` Signal die Maximasuche der Parameterpaare ausgelöst.
- Nach 19 Vergleichsoperationen ist das Maximum gefunden und wird als Abstand `r`, der Winkel ID `phi_id` und dem `max_val` zur Bestimmung der Fahrzeuglage in das

PATH FOLLOWING Modul übergeben; die Signalisierung erfolgt dabei über das `done` Signal.

Die Hough-Transformation wird mit System Generator Blöcken im Pipeliningverfahren umgesetzt. Dabei wird für jede zu berechnende Winkelstufe ein Houghtransformatelement *ht_phi_x* implementiert, die ihre Ergebnisse nebenläufig berechnen. Die Elemente sind so aufgebaut, dass durch Übertaktung und Multizykluspfade das Konzept des *Resource-Sharing* genutzt wird (vgl. Abb. 4.9).

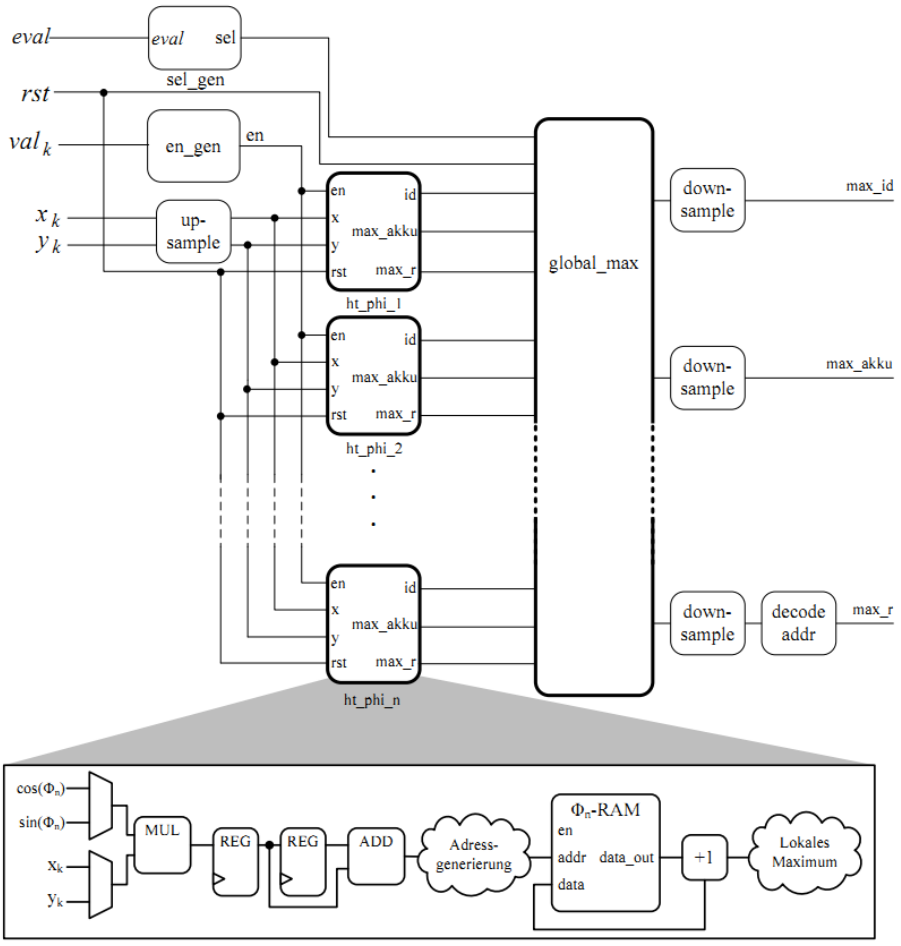


Abbildung 4.9 – Die Pipelinestufen des HOUGH TRANSFORM Modules; zur Umsetzung des *Resource-Sharing* verarbeitet der Multiplizierer eines Hough-Transformator Elements unterschiedliche Eingangswerte [20].

Darstellung von Fahrspurerkennungsergebnissen

Die Darstellung von Zwischenergebnissen der Fahrspurerkennung ist für den Entwicklungsprozess ein entscheidener Vorteil und wird durch die RESULT ILLUSTRATION Komponente realisiert. Dazu werden die Pixelwerte, die Synchorinsationssignale und sowohl die Kamerabildkoordinaten `x` und `y` als auch die projektiv transformierten Koordinaten `x_k` und `y_k` in einem IMAGE BUFFER verarbeitet. Die `x_k`, `y_k`-Koordinaten werden dabei zur Berechnung einer Adresse verwendet, an der die binären Pixelinformationen in einem *Dual Port RAM* abgelegt werden. Dadurch entsteht ein Abbild des projektiv transformierten Bildes im Speicher. Über die geordneten Bildkoordinaten, die ursprünglich für das Kameraeingangsbild im COORDINATE GENERATOR erzeugt wurden, werden die Pixelinformationen über den zweiten Anschluss des *Dual Port RAMs* wieder ausgelesen [21]. Somit liegen die Pixelinformationen des projektiv transformierten Bildes in darstellbaren Bildkoordinaten vor. Das *Dual Port RAM* ist auf die Größe eines Bildes von 640x240 Pixeln angelegt und wird durch BlockRAM Einheiten auf dem FPGA implementiert. Dadurch können die Lese- und Schreibvorgänge innerhalb von zwei Takten innerhalb des 13,5 MHz Systemtaktes erfolgen. Die Synchronisationssignale werden entsprechend verzögert und bilden zusammen mit dem Pixelwert `data` und den x,y-Koordinaten die Ausgangssignale des IMAGE BUFFERS (vgl. Abb. 4.10).

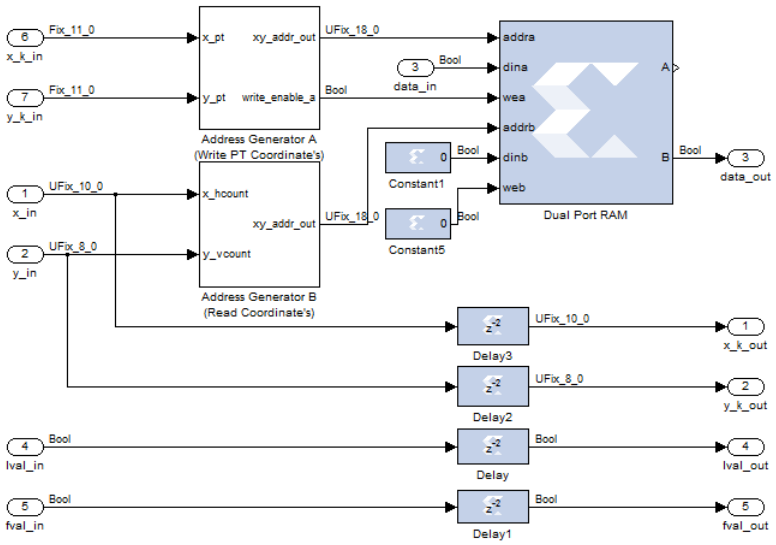


Abbildung 4.10 – Der IMAGE BUFFER setzt auf Grundlage von Pixelkoordinaten die binären Pixelinformationen des projektiv transformierten Bildes in ein darstellbares Ausgangssignal um.

Eine Anpassung des IMAGE BUFFER Moduls, zur Darstellung eines eingeschränkten Sichtbereiches oder die Verminderung von benötigten BRAM Ressourcen, erfolgt zum einen über

die Anpassung der Dimensionierung des *Dual Port RAM* System Generator Blockes, und zum anderen werden die Konstanten zur Erzeugung der Adressbereiche auf die Änderungen abgestimmt; die Konstanten sind innerhalb der *Address Generator* Blöcke zu finden.

Neben der Darstellung des perspektivisch entzerrten Bildes, kann über den externen Schalter `roi_switch` die aktuelle Position der ROI im Ausgangsbild angezeigt werden. Darüber hinaus wird im RESULT OVERLAY Modul die detektierte Fahrspur innerhalb der ROI markiert und der errechnete Winkel Φ durch eine Gerade angezeigt. Durch dieses Vorgehen lässt sich die Regelung des Fahrspurerkennungssystem zu Testzwecken nachvollziehen. Um die beschriebenen Informationen darstellen zu können, werden die Bildkoordinaten aus dem IMAGE BUFFER, die Koordinaten der aktuellen ROI aus dem DYNAMIC ROI Modul sowie der Abstand zur Fahrspurmarkierung `r` und die ermittelte Winkel ID `phi_id` aus dem HOUGH TRANSFORM Modul verwendet. Die binären Pixelinformationen werden auf den 24-Bit RGB-Farbbereich erweitert, damit die ROI in Kontrast zum binären Hintergrundbild dargestellt werden kann. Die ROI-Pixelkoordinaten bestimmen dabei, ob das bestehende Bild mit einem Pixelwert zur Darstellung der ROI überschrieben wird.

Als Ausgangssignale des RESULT ILLUSTRATION Moduls wird der 24-Bit RGB-Pixeldatenstrom zusammen mit den Synchronisationssignalen `FVAL` und `LVAL` an das VGA Interface zur Darstellung auf einem VGA Monitor übergeben. Die Bilddaten liegen hierbei noch im Interlaced Format mit 640x240 Pixeln und einer Frequenz von 13,5 MHz vor.

Ausrichtung des Fahrzeugs auf der Fahrspur

Zur Regelung der Fahrzeugführung werden aus den Ergebnissen der Fahrspurdetektion Algorithmen zur Spurführung angewandt, um die Stellgröße α des SAV zu berechnen. Das PATH FOLLOWING Modul wurde zur Vereinfachung der Funktionstests in mehrere Komponenten unterteilt. Am Eingang werden innerhalb des UPDATE Blockes der erkannte Abstand zur Fahrspurmarkierung `r` und der korrespondierende Winkel Φ als `phi_id` in einen Zusammenhang mit den metrischen Koordinaten der ROI gebracht. Dadurch lässt sich die Position der Fahrspur in das Fahrzeugkoordinatensystem (X_P,Y_P) übertragen und ein angestrebter Look-Ahead-Point ermitteln (vgl. Abb. 3.13).

Die Sinus- und Cosinus Koeffizienten werden mit der `phi_id` aus einer Tabelle bezogen, die als *Single Port RAM* die Werte für den implementierten Winkelbereich von -18° bis +18° in 2°-Schritten bereithält. Der Abstand `r` ist in Pixeln angegeben und wird über eine Multiplikation mit dem Kehrwert des Skalierungsfaktors für die x-Koordinaten in das metrische System überführt.

$$r_{metric} = r_{in} \cdot \frac{1}{mx} = r_{in} \cdot 0.1875 \tag{48}$$

Die Berechnung des LAP im Fahrzeugkoordinatensystem erfolgt in der LAP Komponente, in der Daten- und Steuerpfad in Form einer ASM (*Arithmetic State Machine*) implementiert wurden [26]. Der Steuerpfad ist dabei als MCode Block realisiert, um die Multiplexer und Register des Datenpfades zu setzen. Die Bestimmung des LAP erfolgt nach Gleichung 22, wobei die y-Koordinate den Einsatz der Wurzelfunktion erfordert. Daher wurde eine Approximation der Wurzelfunktion in der LAP Komponente realisiert. Darüber hinaus wird die Position des LAP durch folgende Eingangsparameter beeinflusst (vgl. Abb. 3.13):

- `d`: gibt die Distanz des Fahrzeugmittelpunktes zur Fahrbahnmarkierung in cm an.
- `LAD`: beschreibt den Abstand des LAP zum Ursprung des Fahrzeugkoordinatensystems in cm.

Aus den Gleichungen 20 bis 23 ergibt sich die Vorgehensweise zur Konstrunktion des LAP. Dabei werden die ROI-Koordinaten in das Hilfskoordinatensystem transformiert, die entsprechenden x,y-Werte für den LAP berechnet und zurück ins Fahrzeugkoordinatensystem rotiert. Die nachfolgende Matlab Funktion wurde zur Evaluation der Arithmetik durch die Simulation erstellt und bildet die Rechenschritte zur Ermittlung des LAP ab (vgl. Listing 4.3).

```
function [LAPX LAPY] = LookAheadPoint(ROIX, ROIY, phi, R, D, LAD)
    % x-Wert der ROI im Hilfskoordinatensystem berechnen
    ROIX_H = cos(phi)\cdot ROIX + sin(phi)\cdot ROIY;

    % Berechnen des x-Wertes des LAP im Hilfskoordinatensystem
    LAPX_H = ROIX_H + (R - D);
    LAPY_H = sqrt(LAD \cdot LAD - LAPX_H \cdot LAPX_H);

    % Rücktransformation der LAP-Hilfskoordinaten in das FZG-Koordinatensystem
    LAPX = cos(phi)\cdot LAPX_H - sin(phi)\cdot LAPY_H;
    LAPY = sin(phi)\cdot LAPX_H + cos(phi)\cdot LAPY_H;
end
```

Listing 4.3 – Die Berechnung des LAP durch die Matlab Funktion, die als Basis für den Entwurf der LAP Prozessorelements verwendet wurde.

Das Ergebnis der LAP Berechnung wird als x_v,y_v-Koordinaten des Fahrzeugkoordinatensystems zur Verifikation an zwei 18-Bit breite Ausgangsvektoren an die Software-Register übermittelt. Darüber hinaus wird das Ergebnis zur Anwendung der in Kapitel 3.2.4 vorgestellten Fahrspuhrführungsalgorithmen verwendet. Sowohl bei der FTC- als auch bei der PP- Methode wird der Lenkwinkel α nach Gleichung 24 und Gleichung 25 über die Arkustangens Funktion ermittelt. Dazu wird diese im ARCTAN Prozessorelement approximiert. Mit dem externen Schalter `pp_switch` lassen sich die Eingangssignale multiplexen und die FTC- bzw. PP-Methode als Grundlage der Fahrspurführung auswählen. Für den Fall der FTC-Methode werden die x_v,y_v-Koordinaten des LAPs an die ARCTAN Komponente übergeben, wobei die Division innerhalb des Prozessorelements erfolgt. Zur Realisierung der PP-Methode wird Gleichung 25 umgeformt, um dem Aufbau der implementierten Division

zu entsprechen:

$$\alpha = \arctan\left(\frac{L}{r}\right) = \arctan\left(\frac{2 \cdot x_{LAP} \cdot L}{LAD^2}\right) = \arctan\left(\frac{x_{LAP}}{\frac{LAD^2}{2 \cdot L}}\right) \tag{49}$$

Die Parametrierung der LAD wird für die Entwicklungsphase dynamisch gehalten und erfolgt über einen 8-Bit breiten Wert, der aus einem Software-Register vom MPSoC bezogen wird. Dementsprechend muss die Bestimmung des Nenners aus Gleichung 49 ebenfalls dynamisch konfigurierbar sein. Die entsprechende Konstante $\frac{LAD^2}{2 \cdot L}$ wird daher über einen 18-Bit breiten Vektor mit vier Fließkommastellen in das PATH FOLLOWING Modul aufgenommen (vgl. Abb. 4.3). Die Approximation der Arkustangens Funktion wird analog zur Approximation der Wurzelfunktion in der LAP Komponente durch eine ASM und die Trennung von Steuer- und Datenpfad realisiert [26].

Neben den Fahrspurführungsalgorithmen ist zusätzlich ein PD-Regler für die Abstandsregelung des Fahrzeugs zur Fahrspurmarkierung implementiert. Dabei wird die Regelungsfunktion durch Gleichung 39 aus Kapitel 3.2.4 durch ein Prozessorelement realisiert (vgl. Abb. 4.11). Die Stellgröße $u(k)$ ist durch das Eingangssignal `d` gegeben. Die Ermittlung der Regelabweichung $e(k)$ erfolgt durch den Vergleich des vorgegebenen `d` Wertes mit dem ermittelten Abstand `r` und der aktuelle x- Ausrichtung der ROI `roi_x`.

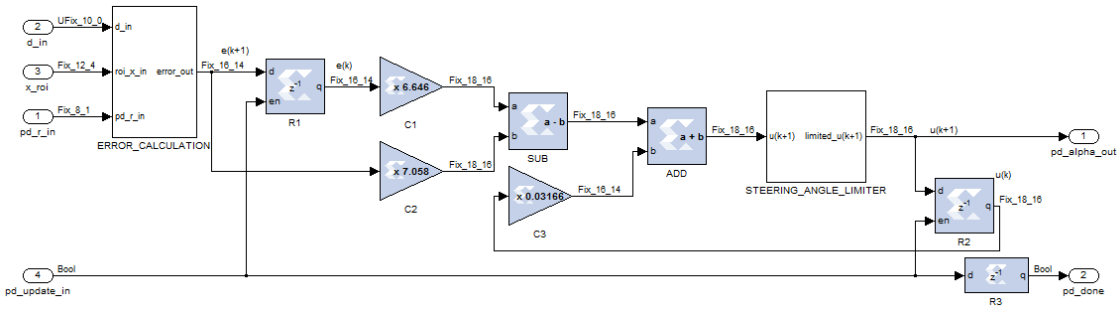


Abbildung 4.11 – Der PD-Regler zur Abstandsregelung; durch den STEERING_ANGLE_LIMITER wird der Lenkwinkelausgang `pd_alpha` auf den Wirkungskreis des Lenkservos von -20° bis +20° eingeschränkt.

Der Lenkwinkelsollwert `alpha` kann somit durch drei verschiedene Fahrspurführungsalgorithmen erzeugt werden. Über den Funktionsblock MODE SELECT und den zwei externen Schaltern `pd_switch` und `pp_switch` lässt sich der α Wert für die Generierung des PWM Signals bestimmen.

Die Erzeugung des PWM Signals erfolgt in dem Prozessorelement PWM GENERATION. Das Eingangssignal kann mit einem Multiplexer über einen externen Schalter `alpha_switch` aus der Fahrspurerkennung oder durch den externen `alpha_ctrl` Vektor

aus den Software-Registern bezogen werden. Dadurch kann die Regelung des Lenkwinkels für Ausweichmanöver vom MicroBlaze Steuerungssystem unterbrochen und unabhängig von der Regelstrecke gesetzt werden. Um Beschädigungen des Lenkwinkelservos zu verhindern, wird der Einstellwinkel vor der Erzeugung des PWM Signals auf den Wertebereich von -20° bis +20° beschränkt. Das PWM Signal wird über einen Zähler für eine dem Lenkwinkel entsprechende Periode generiert[2]. Um die Vektorbreite des Zählers gering zu halten wird das Prozessorelement über einen *Down Sampling* Block auf 6,75 MHz getaktet.

4.1.4 Darstellung des Pixeldatenstroms

Die Darstellung des Pixeldatenstroms auf einem VGA Monitor ist während der Entwicklungsphasen zur optischen Verifikation von Teilergebnissen erforderlich. Über das `output_switch` Signal kann dabei der Eingangsdatenstrom des VGA INTERFACES gewählt werden. So werden entweder die von der Kamera bereitgestellten, ins RGB-Format transformierten, Bilder oder die Ergebnisse der RESULT ILLUSTRATION des PATHFINDER Moduls, mit der Darstellung des projektiv transformierten Kamerabildes und der erkannten Fahrspurgeraden in der dynamischen ROI, dargestellt (vgl. Abb. 4.12). Das

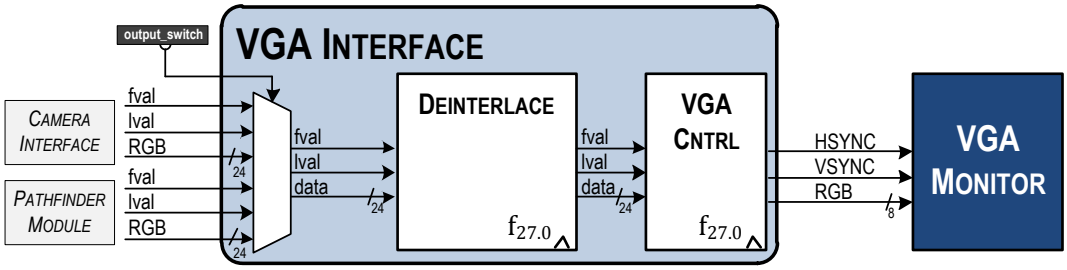


Abbildung 4.12 – Das VGA INTERFACE; über einen externen Schalter lässt sich der Eingangsdatenstrom zwischen Kameradaten und perspektivisch entzerrtem Bild umschalten.

`output_switch` Signal ist über das FPGA Pinout (vgl. Anhang E) direkt auf die DIP-SWITCH Bank des Entwicklungsboards gelegt, womit die Ausgabe zur Laufzeit beeinflusst werden kann. Die Pixeldaten liegen in beiden Ausprägungen in einem einheitlich Format mit den drei 8-Bit breiten Farbwerten für Rot, Grün und Blau, in Bildgrößen von 640x240 Pixeln und einer Frequenz von 13,5 MHz am Eingang des VGA INTERFACES an. Im DEINTERLACE Modul werden die Pixeldaten aufgrund der in Kapitel 3.1 erläuterten, standardisierten Anforderungen von VGA Monitoren auf eine Ausgangsfrequenz von 27 MHz durch eine Zeilenverdoppelung „deinterlaced“. Dazu stehen zwei Shift Register zur Verfügung, die alternierend die Pixeldaten genau einer Bildzeile mit der Eingangsfrequenz von 13,5 MHz

[2]Die Spezifikation zur Erzeugung gültiger PWM Perioden ist im Anhang B in Tabelle B.1 zu finden

aufnehmen. Der Flankenwechsel des LVAL Signals, der das Anliegen der nächsten Bildzeile signalisiert, wird im Zustandsautomaten erkannt und die nächste Bildzeile im zweiten Shift Register gespeichert. Zeitgleich erfolgt die Ausgabe der zuvor gespeicherten Bildzeile mit der doppelten Frequenz. Das FVAL Signal bleibt für die Verarbeitungszeit von 240 Eingangszeilen bzw. 480 Ausgangszeilen *high*-aktiv [22].

Zur Darstellung der Bilddaten auf einem Monitor werden im VGA CONTROLLER durch horizontale und vertikale Zähler HSYNC und VSYNC Synchronisationssignale erzeugt [22]. Darüber hinaus werde Werte des Datenstroms in den Blankingphasen auf RGB_H= #000000 (Schwarz) gesetzt. Die auf dem Entwicklungsboard verwendete DAC VGA-Schnittstelle stellt jeweils 4-Bit breite Eingangssignale für die RGB-Farbinformationen zur Verfügung [34], daher werden die Farbwertausgänge des VGA INTERFACES auf die ersten vier Bit reduziert.

4.1.5 Wechsel der Betriebsmodi des SAV

Die Vorgaben des Carolo Cup sehen unter anderem vor, dass eine Unterbrechung des autonomen Betriebes zur Laufzeit erfolgen muss, sobald das Fahrzeug die Teststrecke verlässt. Das Umschalten zwischen den Betriebsmodi erfolgt über die Auswertung des freien PWM Signals der Handfernbedienung (vgl. Kap. B). Dieses Signal wird je nach Schalterstellung

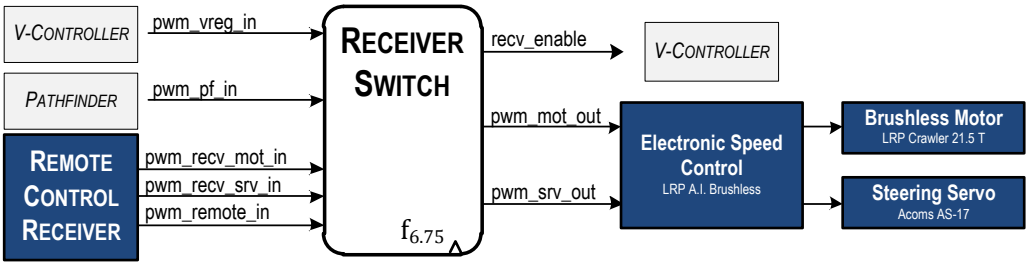


Abbildung 4.13 – Für die Umschaltung der Betriebsmodi werden die Eingangssignale der Funk Fernbedienung durch das RECEIVER SWITCH Modul interpretiert und die Ausgänge zum Fahrtensteller entsprechend geschaltet.

der Fernbedienung für 1,5 ms bzw. 2 ms innerhalb der 20 ms PWM Periode aktiv gesetzt und in den RECEIVER SWITCH als Eingangssignal PWM_REMOTE_IN aufgenommen (vgl. Abb. 4.13). Daneben werden sowohl die generierten PWM Signale der Fahzeugregelungskomponenten, als auch die von der Fernbedienung an den RC-Empfänger übertragenen PWM Signale zur Motor- und Lenkwinkelsteuerung als Eingangssignale in das RECEIVER SWITCH Modul geführt. Durch die Auswertung der Länge des aktiven PWM_REMOTE_IN Signals

werden die entsprechenden Eingänge zu den Ausgängen an den Motor bzw. Lenkservo geschaltet. Die Auswertung erfolgt über die Implementierung eines Zustandsautomaten, der bei Auftreten des aktiven `PWM_REMOTE_IN` einen Zähler startet. Dieser ist auf die Konstante `GEN_PWM_NORM_PERIOD` beschränkt, welche einen Zeitraum von 1,7 ms abbildet. Dazu wird ein VHDL-Generic auf 11.475 festgelegt; $11.475 \cdot 6{,}75\ \text{MHz} = 1{,}7\ \text{ms}$.

Die 6,75 MHz Taktfrequenz des Moduls wird vom CLOCK MANAGER bezogen; die Wahl der niedrigsten Taktfrequenz bietet den Vorteil, dass die Vektorbreite des Zählers auf 14-Bit begrenzt werden kann. Überschreitet der Zähler die angegebene Konstante, ist davon auszugehen, dass die Breite des PWM Signals 2 ms beträgt und die Fernsteuerung den manuellen Betriebsmodus signalisiert (vgl. Listing 4.4). Dadurch werden die vom RC-Empfänger übermittelten Steuersignales an die Aktoren übertragen. Entsprechend werden die Eingangswerte der Regelungssysteme an den Ausgang des SAV CONTROL gelegt, wenn die Dauer des aktiven PWM Signals unter 1,7 ms liegt.

```
[...]
when COUNT=>
      if PWM_REMOTE_IN = '0' then
        if pwm_counter > GEN_PWM_NORM_PERIOD then
          --finished counting for 1,7 ms, pwm is longer, so remote is active
          pwm_recv_dis <= '1';
          recv_cnt_en <= '0';
          NEXT_STATE <= START;
        else
          --pwm period less than 1,7 ms -> stay in pathfinder mode
          pwm_recv_dis <= '0';
          recv_cnt_en <= '0';
          NEXT_STATE <= START;
        end if;
      else
        recv_cnt_en <= '1';
        NEXT_STATE <= COUNT;
      end if;
[...]
```

Listing 4.4 – Im COUNT Zustand der FSM wird sobald das `PWM_REMOTE_IN` nicht mehr aktiv ist, der Zählerstand abgefragt und abhängig davon die Ausgangssignale mit `pwm_recv_dis` gesetzt. Diese Zuweisung findet im getakteten Prozess statt.

4.2 Das MPSoC zur Fahrzeugsteuerung

Mit dem Einsatz eines Multiprozessorsystems als Kernkomponente der SAV Plattform erweitern sich die Möglichkeiten zur Integration von zusätzlichen Funktionalitäten in das Gesamtsystem. Über spezielle IPs werden Schnittstellen zum Datenaustausch und der Verarbeitung von zusätzlichen Sensorwerten in den Systementwurf übernommen. Darüber hinaus wird der Entwicklungsprozess, für die Ansteuerung der auf dem Entwicklungsboard verfügbaren Hardwareperipherie, durch die Verwendung von Softwaretreiber um ein Vielfaches erleichtert. Auf die gleiche Weise wird das SAV CONTROL IP zur Fahrzeugregelung in das System integriert, wodurch deren Parameter über Software-Register zur Laufzeit angepasst werden können.

4.2.1 Bestandteile des Hardwaredesigns

Das MPSoC besteht aus zwei MicroBlaze Softcore Prozessoren, die über getrennte PLB Bussysteme mit unterschiedlichen Peripherie-Komponenten des Entwicklungsboards verbunden sind (vgl. Abb. 4.14 und Anhang D). Das System ist als AMP-basierte Architek-

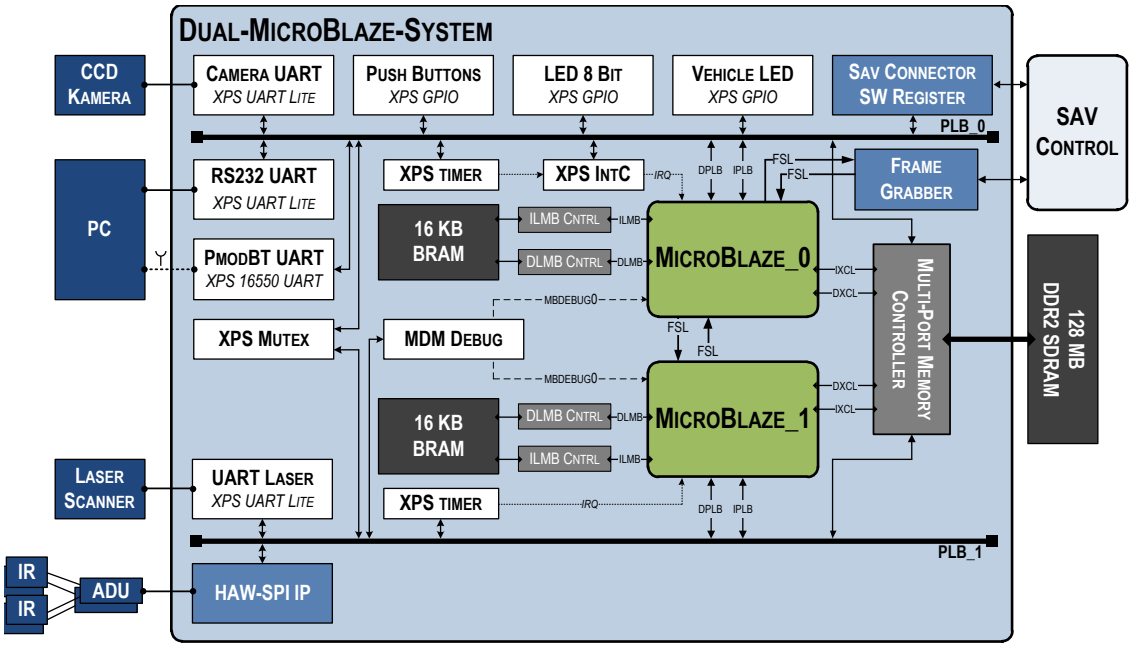


Abbildung 4.14 – Das DUAL-MICROBLAZE-SYSTEM zur Parametrierung der SAV-Regelung. Zusätzlich stehen Schnittstellen zur Kommunikation mit dem Entwicklungsrechner und der Einbindung weiterer Peripherie-Komponenten zur Verfügung.

tur angelegt, wobei **microblaze_0** als *Master* eingesetzt wird, um die Fahrzeugregelung zu

steuern und die Kommunikation mit externen Schnittstellen zur Datenaufzeichnung vornimmt. **microblaze_1** verfügt über Schnittstellen zu den im System eingesetzte Abstandssensoren und bereitet deren Daten für die Auswertung auf dem *Master* vor. Beide Prozessoren sind durch Daten- und Instruktionsverbindungen des LMB an separate 16 kB große BRAM Speicherblöcke angeschlossen. Zusätzlich sind beide durch entsprechende Daten- und Instruktionsleitungen des XCL mit einem MPMC verbunden, der die Speicherzugriffe auf das 128 MB große DDR2 SDRAM vornimmt. Obwohl der Oszillator des Entwicklungsboard eine Frequenz von 125 MHz bereitstellt, wird eine Systemfrequenz von 62,5 MHz verwendet, da der MPMC die Speicherzugriffe mit der doppelte Systemfrequenz vornehmen muss. Innerhalb des DUAL-MICROBLAZE-SYSTEMS werden diese Taktfrequenzen durch ein *clock generator* IP bereitgestellt und die 62,5 MHz Systemfrequenz über die PLB-Anschlüssen an die entsprechenden IPs verteilt.

Darüber hinaus wird für die Implementierung des Xilkernel Betriebssystems an beide MircoBlaze Prozessoren ein *XPS Timer* IP angebunden. Diese erzeugen, die für das Scheduling der Software-Threads erforderlichen, Interrupts, wobei die Interrupt Ausgänge der Timer Module direkt mit den `INTERRUPT` Eingangsports der MicroBlaze verbunden sind. Die Einrichtung der Timer/Counter wird mit der Xilkernel Konfiguration bei Systemstart durchgeführt, indem deren Status- und Load Register entsprechend beschrieben werden. Im DUAL-MICROBLAZE-SYSTEMS werden die Timer Module ausschließlich zur Erzeugung der Interrupts verwendet; in dieser Funktionsweise wird der Wert des Load Registers mit jedem Takt bis zum Nulldurchgang dekrementiert, anschließend das Ausgangssignal `Interrupt` für einen Takt aktiviert und der Wert des Zählers wieder auf den Initialwert des Load Registers zurückgesetzt.

Die Übertragung des bitfiles zur Programmierung des FPGAs erfolgt über eine JTAG Schnittstelle, die gleichzeitig über das MDM (*MicroBlaze Debug Module*) mit beiden Prozessoren verbunden ist. Über diese Verbindung erfolgt zudem die Verteilung der unterschiedlichen elf Files an die MicroBlazes. Das MDM erlaubt zusätzlich das simultane Debuggen der beiden unterschiedlichen MicroBlaze Anwendungen.

Als *Master* übernimmt **microblaze_0** vorrangig die Kommunikation mit externen Schnittstellen, um die Parameter des SAV CONTROL IPs anpassen zu können und die Systemzustände zu dokumentieren. Die Anpassung der Regelungsparameter erfolgt über das SAV CONNECTOR IP. Dieses stellt zehn 32-Bit breite Software-Register zur Verfügung, über die Daten zwischen dem MicroBlaze System und der SAV CONTROL *Hardware Accelerator* Komponente ausgetauscht werden können. Somit kann das Regelungsverhalten zur Laufzeit angepasst werden und erfordert keine erneute Hardwaresynthese für die Erprobung von unterschiedlichen Reglerparametrierungen. Dabei übernimmt das SAV CONNECTOR IP die Aufgabe den Inhalt der Software-Register zwischen den beiden unterschiedlichen Frequenzdomänen des MicroBlaze Systems und der Fahrzeugregelungskomponente vorzunehmen. Das SAV CONTROL arbeitet, abhängig von der Bilddatenfrequenz, in einem Bereich von 27 MHz, was für die Übernahme der Werte aus den Software-Registern, die über den PLB

mit 62,5 MHz versorgt werden, zu instabilen Registerinhalten führen kann. Dazu wird mit den vom PLB Interface erzeugten Synchronisationssignalen im SAV CONNECTOR IP in ein verzögertes SW_REGS_EN Signal erzeugt, das zur Signalisierung von gültigen Registerinhalten im SAV CONTROL interpretiert wird (vgl. Listing 4.5).

```
ENABLE_READ_FOR_IP : process(Bus2IP_Clk) is
 begin
   if Bus2IP_Clk'event and Bus2IP_Clk = '1' then
     if Bus2IP_Reset = '1' then
       sig_sw_regs_en <= '0';
       counter_sw_regs_en <= 0;
     else
       if slv_write_ack = '1' then --plb sync signals detected
         sig_sw_regs_en <= '1';    --save status for delay
       end if;
       if sig_sw_regs_en = '1' then
         counter_sw_regs_en <= counter_sw_regs_en + 1;
       else
         counter_sw_regs_en <= 0;
       end if;
       if counter_sw_regs_en = 5 then --end of delay -> reset
         sig_sw_regs_en <= '0';
         counter_sw_regs_en <= 0;
       end if;
     end if;
   end if;
 end process ENABLE_READ_FOR_IP;

SW_REGS_EN  <= sig_sw_regs_en; --sig_sw_regs_en is valid for at least 5 clks
```

Listing 4.5 – SAV CONNECTOR; die Verzögerung der PLB Synchronisationssignale über 5 Takte in der 62,5 MHz Domäne lässt dem SAV CONTROL 80 ns zur Interpretation des SW_REGS_EN Signals.

Die Implementierung der zehn Software-Register sieht vier Augangsregister vom SAV CONTROL zur Übertragung von Berechnungsergebnissen, wie zum Beispiel dem ermittelten Lenkwinkel oder die Position des LAPs, und sechs Eingangsregister zur Parametrierung der Fahrzeugregelung vor. Diese Aufteilung kann in der weiteren Erprobungsphase angepasst werden, wobei allerdings zu Beachten ist, dass alle Änderungen sowohl in den Quellen der SAV CONTROL und SAV CONNECTOR IPs als auch in der implementierten Software-Anwendung durchgeführt werden müssen. Die Übersicht zu den Belegungen der Software-Registern befindet sich in Kapitel 4.2.3 im Rahmen der Beschreibung der Steuerungssoftware.

Neben dem SAV CONNECTOR gibt es ein zweites IP zum Datenaustausch mit der *Hardware Accelerator* Komponente. Für die Ermittlung der Transformationsparameter zur perspektivischen Entzerrung der Bilddaten wird ein Bild der Kalibrierungsmatte aus der Kameraperspektive verwendet (vgl. Kap. 3.2.2). Das FRAME GRABBER Modul wird eingesetzt, um ein solches Bild aus der Bildbearbeitungspipeline zu extrahieren. Das Modul ist über zwei FSL Anschlüsse mit **microblaze_0** verbunden, wobei ein FSL zur Signalisierung der Bildanfrage verwendet wird und der andere die Datenübertragung eines gesamten Bildes

mit 640x240 Bildpunkten realisiert. Die Bildpunkte werden auf ein 8-Bit RGB-Format reduziert und jeweils vier Bildpunkte in den 32-Bit breiten FIFO des FSL geschrieben. Dabei wird die Eigenschaft des FSLs, mit unterschiedlichen Eingangs- und Ausgangsfrequenz arbeiten zu können, ausgenutzt, um die Daten zwischen den beiden Clockdomänen von 62,5 MHz und 13,5 Mhz auszutauschen. Der FRAME GRABBER ist als einfacher Zustandsautomat implementiert, der sobald er eine Bildanfrage über das `FSL_S_Exists` erkannt hat, die Synchronisationssignale des Datenstroms auf den Beginn eines neuen Bildes überprüft. Sobald das erste Pixel eines Bildes anliegt, fasst er jeweils vier Pixel zu einem 32-Bit Wert zusammen und schreibt diesen mit dem `FSL_M_WRITE` Signal in den FSL.

Über den plb_0 ist der **microblaze_0** zudem mit unterschiedlichen UART Schnittstellen verbunden. Dabei wird zur Kalibrierung der Kamera ein *XPS UART Lite* Modul eingesetzt, um die Eigenschaften der Kamera zur Laufzeit anpassen zu können. Eine weitere *XPS UART Lite* Schnittstelle dient zur Kommunikation mit dem Entwicklungsrechner, um die Steuerungssoftware zu bedienen und Informationen über den Systemzustand zu erhalten. Um diese Ausgaben auch während der Erprobungsfahrten zu erhalten, bei denen keine direkte Verbindung mit dem Entwicklungsrechner vorhanden ist, steht zusätzlich ein *XPS UART 16550* IP zur Verfügung. Dieses übernimmt die Kommunikation mit dem PmodBT Adapter, wodurch die Systemdaten über eine Bluetooth Verbindung zum Entwicklungrechner übermittelt werden können. Der Vorteil des *XPS UART 16550* IP gegenüber des *XPS UART Lite* besteht darin, dass die Verbindungsparameter durch Softwarefunktionen zur Laufzeit angepasst werden können. Weiterhin sind über plb_0 drei *XPS GPIO* IPs integriert, um einerseits die Pushbuttons und LEDs des Entwicklungsboards ansprechen zu können und zweitens die Ansteuerung der Fahrzeugbeleuchtung vorzunehmen. Somit können die Systemparameter der Regelung auch über die Pushbuttons direkt auf dem Entwicklungsboard zur Laufzeit beeinflusst werden; die Board LEDs zeigen davon abhängig den gewählten Systemzustand an.

Der zweite Prozessor **microblaze_1** realsiert die Integration der Abstandssenorik. Dazu wird ein separater plb_1 verwendet, um die Daten der Sensoren ohne Beeinflussung des Steuersystems auf **microblaze_0** verarbeiten zu können. Die Abstandswerte des Laserscanners werden in ihrem Rohdatenformat über ein *XPS UART Lite* IP bezogen und von der Software-Anwendung des **microblaze_1** auf Basis der Entfernungswerte segmentiert. Neben der Segmentierung wird weiterführend eine Objekterkennung durchgeführt, um Hindernisse auf der Fahrspur des SAV zu detektieren (vgl. Kap. 4.2.3). Des Weiteren erfolgt über plb_1 die Anbindung des HAW SPI IPs an **microblaze_1** . Dieses erfasst die Daten der Infrarot-Sensoren zur seitlichen Abstandsmessung und errechnet aus den digitalisierten Eingangsspannungen die korrespondierenden Abstandswerte [1]. Die vier Messdaten werden in zwei Software-Registern abgelegt und können über den PLB vom **microblaze_1** bezogen werden, zur Initialisierung und Überwachung des HAW SPI IPs stehen zwei weitere Software-Register zur Verfügung.

Damit auch vom zweiten MicroBlaze Prozessor Ausgaben zu Testzwecken an den Entwick-

lungsrechner übermittelt werden können, ist dieser über plb_1 zusätzlich mit dem *MicroBlaze Debug Modul* verbunden, das eine JTAG UART Schnittstelle für die Systemausgaben bereitstellt.

Die Parametrierung der IPs des DUAL-MICROBLAZE-SYSTEMS erfolgt über das MHS File; die Einstellungen für die verschiedenen UART Schnittstellen sind in der nachfolgenden Tabelle 4.3 zusammengefasst.

	Camera UART XPS UART Lite	RS232 UART XPS UART Lite	PmodBT UART XPS UART 16550	UART Laser XPS UART Lite
Baud Rate	9600	115.200	9600	115.200
Data Bits	8	8	8	8
Use parity	true	false	true	true
Parity	odd	even	odd	odd
PLB	plb_0	plb_0	plb_0	plb_1

Tabelle 4.3 – Die Parametrierung der UART Interfaces für **microblaze_0** und **microblaze_1**; die Einstellungen der PMODBT UART Schnittstelle können dabei zur Laufzeit in der Software geändert werden.

Beide Prozessoren sind zudem über den XCL und die PLBs mit dem MPMC verbunden. Die XCL Leitungen werden für Zugriffe auf Speicherbereiche des DDR2 SDRAMs verwendet, wobei für jeden MicroBlaze ein 8-kB Daten- und ein 8-kB Instruktionscache implementiert wurden. Der Zugang zum Speicher über den PLB ist ohne die Verwendung von Caches umgesetzt. Dabei werden die Speicherbereiche so partitioniert, dass die Prozessoren über die XCL Verbindungen auf Speicherbereiche zugreifen, die ausschließlich für sie zugängig sind. Der Zugriff über die PLBs erfolgt auf einen gemeinsam verwendeten Speicherbereich. Diese Vorgehensweise besitzt den Vorteil, dass keine Synchronisation der Cacheinhalte vorgenommen werden muss, da keine überlappenden Speicherbereiche für die Caches verwendet werden. Zwar ist der Zugriff auf das Shared Memory über den PLB-Zugriff langsamer, es müssen aber keine Cache Kohärenz Mechanismen implementiert werden [32]. Dadurch können über das Shared Memory Daten zwischen **microblaze_0** und **microblaze_1** ausgetauscht werden. Zur Synchronisation der Zugriffe wird das *XPS Mutex* IP eingesetzt, was eine Hardware-basierte Implementierung des „Wechselseitigen Ausschluss“-Verfahrens darstellt. Bei der Erstellung von Software-Anwendungen muss allerdings beachtet werden, dass die Zugriffe auf den gemeinsamen Speicher über die Sicherungsfunktionen des *XPS Mutex* Treibers geschützt werden.

Zusätzlich werden zwei FSL Verbindungen eingesetzt, um eine direkte Kommunikationschnittstelle zwischen **microblaze_0** und **microblaze_1** einzurichten. Dabei sind die FSL Verbindungen so angelegt, dass die Prozessoren jeweils einmal als Master- und einmal als

Slaveteilnehmer fungieren. Durch die direkte Verbindung können Daten ausgetauscht werden, ohne die Zugriffe auf das Shared Memory synchronisieren oder die Arbitrierungszeiten des MPMCs berücksichtigen zu müssen. Als exemplarischer Anwendungsfall für die Verwendung dieser zusätzlichen Schnittstelle, lässt sich die Signalisierung für die Detektion von Hindernissen angeben. Nachdem **microblaze_1** ein Objekt im Toleranzbereich vor dem Fahrzeug erkannt hat, benachrichtigt er **microblaze_0** über den FSL. Die ermittelten Objektkoordinaten werden dann von **microblaze_0** aus dem Shared Memory gelesen und zur Auswertung in das Fahrzeugkoordinatensystem überführt.

4.2.2 Konfiguration der Hardwareplattform

Das erstellte Hardwaredesign wird für die Implementierung der Software Komponenten speziell konfiguriert, um die Treiber zu den Hardware Schnittstellen zu integrieren, die angepasste Speicherpartitionierung vorzunehmen und den Funktionsumfang des Xilkernel Betriebssystems zu bestimmen. Dazu wird für jeden MicroBlaze ein BSP (*Board Support Package*) angelegt, das die speziellen Funktionen konfiguriert. Mit den BSPs werden die Treiberdateien in das Softwareprojekt integriert, wodurch das Ansprechen der Hardwareperipherie über spezielle Softwarefunktionen oder über speicherbezogene Adressierung (Memory Mapped) erfolgt. Das Xilkernel Betriebssystem kann speziell auf die Anforderungen der Systemeigenschaften konfiguriert werden, wodurch die Größe des erstellten .elf Files entscheidend beeinflusst werden kann, um die Belegung von Speicherressourcen zu verringern. Die Konfiguration der BSPs wird für das DUAL-MICROBLAZE-SYSTEM für jeden Prozessor separat vorgenommen. So werden Shedulingverfahren, Threadunterstützung, Software Timer oder zusätzliche OS-Funktionen gemäß den Anforderungen der Software-Anwendung konfiguriert (vgl. Tab. 4.4).

Parameter	microblaze_0	microblaze_1
stdin	RS232_UART_1	mdm_0
stdout	RS232_UART_1	mdm_0
sysintc_spec	none	none
systmr_spec	true	true
systmr_dev	xps_timer_plb_0	xps_timer_plb_1
config_pthread_support	true	true
max_pthreads	10	10
pthread_stack_size	1000	1000
config_pthread_mutex	true	true
max_pthread_mutex	10	10
max_pthread_mutex_waitq	10	10
config_shed	true	true
shed_type	SCHED_RR	SCHED_PRIO
n_prio	32	32
max_readyq	10	10

Parameter	microblaze_0	microblaze_1
config_time	true	false
max_tmrs	10	10
config_sema	false	false
config_shm	false	false
config_bufmalloc	false	true
copyoutfiles	false	false
config_debug_support	false	false
enhanced_features	true	false
config_kill	true	false
config_yield	true	false

Tabelle 4.4 – Übersicht der Xilkernel Parameter für die Board Support Packages der beiden MicroBlaze Prozessoren

Ein weiterer Bestandteil der Konfiguration des Hardwaredesigns ist die Speicherpartitionierung. Dazu werden die Adressen der Speichersektionen den physikalisch zur Verfügung stehenden Speicherbereichen zugeordnet (vgl. Abb. 4.15). Die reset.-, interrupt.- und exception.vectors für die Softwareausführung müssen im lokalen BRAM der MicroBlaze Prozessoren liegen, da in diesen die Startadressen für die Software-Anwendungen hinterlegt sind. Die schnellen Zugriffszeiten auf das BRAM begünstigen zudem die Implementierung der .heap und .stack Sektionen in diesem Speicherbereich. Damit wird zudem die getrennte Verarbeitung der unterschiedlichen Anwendungen auf den Prozessoren sichergestellt. Heap und Stack sind für beide Prozessoren auf eine Größe von 4-kB angelegt.

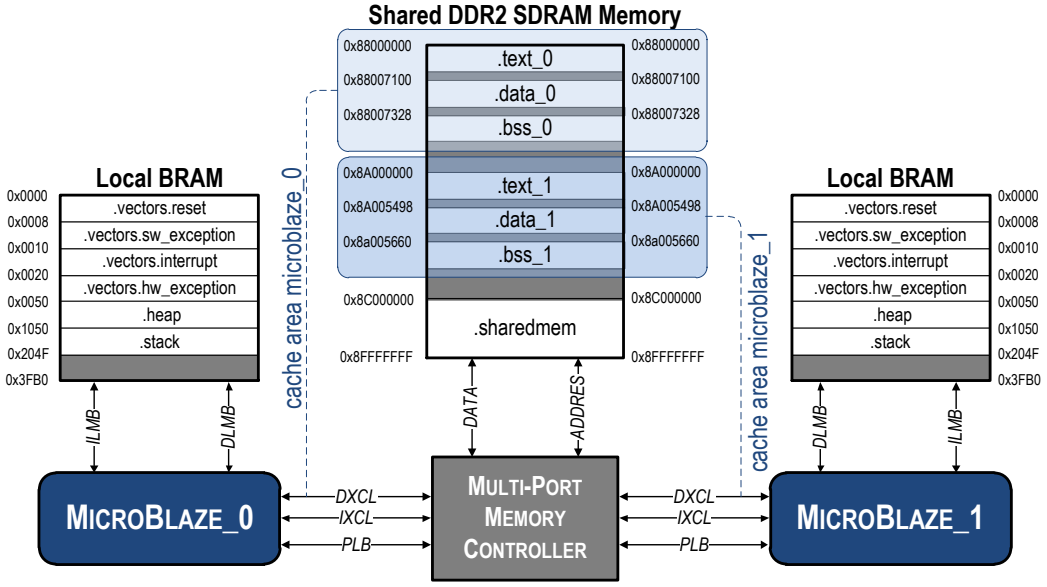


Abbildung 4.15 – Die Speicherzuteilung für beide MicroBlaze Prozessoren, wobei die Zugriffe auf die .text, .data und .bss Sektionen durch den XCL vorgenommen werden und somit mit Caches versehen sind.

Die Programmcodes in den .text und .data Sektionen sind aufgrund der Einbindung von Treiber- und Xilkernel-Bibliotheken, zu groß für eine direkte Platzierung im BRAM. Daher werden sie für beide MicroBlazes in getrennten Bereichen des DDR2 SDRAMs abgelegt. Der Zugriff auf diese Bereich wird durch den XCL vorgenommen, der gleichzeitig die Daten und Instruktionen in getrennten Caches zwischenspeichert um die Zugriffszeiten zu optimieren. Daneben steht beiden Prozessoren ein gemeinsamer Speicherbereich zu Verfügung, in dem Daten ausgetauscht werden können. Der Zugriff auf diesen Bereich erfolgt über die PLB Anschlüsse des MPMC und muss über die Sicherungsmechanismen des Hardware Mutex Moduls in der Anwendungsebene synchronisiert werden. Die Partitionierung der Speicherbereiche wird über das *Linker Script* für jeden MicroBlaze separat vorgenommen. Dabei werden zunächst die Speicherbereiche mit ihren Startadressen und Größen spezifiziert, um die Sektionen diesen Bereichen zuordnen zu können (vgl. Lst. 4.6).

```
[...]
_STACK_SIZE = DEFINED(_STACK_SIZE) ? _STACK_SIZE : 0x1000;
_HEAP_SIZE = DEFINED(_HEAP_SIZE) ? _HEAP_SIZE : 0x1000;
/* Define Memories in the system */
MEMORY
{
   ilmb_cntlr_mb_0_new_dlmb_cntlr_mb_0_new : ORIGIN = 0x00000050, LENGTH = 0x00007FB0
   DDR2_SDRAM_MPMC_BASEADDR : ORIGIN = 0x88000000, LENGTH = 0x02000000
   DDR2_SDRAM_SHARED_MEM_BASEADDR : ORIGIN = 0x8C000000, LENGTH = 0x04000000
}
/* Specify the default entry point to the program */
ENTRY(_start)
/* Define the sections, and where they are mapped in memory */
SECTIONS
[...]
.sharedmem : {
   __sharedmem_start = .;
   *(.sharedmem)
   __sharedmem_end = .;
} > DDR2_SDRAM_SHARED_MEM_BASEADDR

_SDA_BASE_ = __sdata_start + ((__sbss_end - __sdata_start) / 2 );
_SDA2_BASE_ = __sdata2_start + ((__sbss2_end - __sdata2_start) / 2 );
/* Generate Stack and Heap definitions */
.heap : {
   . = ALIGN(8);
   _heap = .;
   _heap_start = .;
   . += _HEAP_SIZE;
   _heap_end = .;
} > ilmb_cntlr_mb_0_new_dlmb_cntlr_mb_0_new

.stack : {
   _stack_end = .;
   . += _STACK_SIZE;
   . = ALIGN(8);
   _stack = .;
   __stack = _stack;
} > ilmb_cntlr_mb_0_new_dlmb_cntlr_mb_0_new

_end = .;
}
```

Listing 4.6 – Das *Linker Script* zur Speicherpartitionierung für **microblaze_0** ; Die Adresse für den Shared Memory Bereich wird in der gleichen Form im *Linker Script* von **microblaze_1** eingepflegt.

4.2.3 Die Anwendungssoftware der SAV Plattform

Die Implementierung der Software auf dem DUAL-MICROBLAZE-SYSTEM wird gemäß der Prozessorstruktur in zwei eigenständige Anwendungen unterteilt. Zur Steuerung der Fahrzeugführung übernimmt **microblaze_0** die Kommunikation mit dem Regelungssystem und den externen Schnittstellen. Die Anwendung von **microblaze_1** dagegen nimmt die Sensorwerte des Laserscanners auf und führt die Segmentierung zur Objektidentifikation durch. Der grundlegende Aufbau der `main`-Methoden besteht dabei aus den folgenden Teilschritten:

- Initialisierung der HW-Plattform durch Aktivierung der Caches
- Einrichtung der Peripherie-Komponenten; durch die Zuweisung von globalen Variablen für den *Memory Mapped* Zugriff oder der Vorbelegung der Software-Register Inhalte für das SAV CONTROL IP
- Initialisierung des Xilkernel, dabei werden die Einstellungen aus den BSPs übernommen und eingerichtet
- Instanziierung eines „Master“ Threads, aus dem die Threads zur Umsetzung der Funktionalität gestartet werden
- Start des Xilkernel; dazu wird ein Idle Task erzeugt, der mit kleinster Priorität im Hintergrund läuft

Zwischen den beiden „Master“ Threads der SAV Plattform erfolgt keine Synchronisation, dadurch kann in der Entwicklungsphase jeweils nur ein MicroBlaze gestartet werden, um die Funktionsweise isoliert zu verifizieren.

Steuerungssoftware zur Fahrzeugführung

Die Software-Anwendung von **microblaze_0** übernimmt die Auswertung und Darstellung der Fahrzeugkennwerte. Dazu werden die vom PATHFINDER bereit gestellten Fahrzeugdaten aus den Software-Registern des SAV CONNECTORS gelesen bzw. neue Vorgabewerte in diese geschrieben. Jedes der 32-Bit breiten Software-Register nimmt dabei zwei auf 16-Bit reduzierte Werte auf. Die Übersicht in Abbildung 4.5 zeigt die Verteilung der Werte auf die Software-Register und deren Darstellungsform im Q-Format bzw. in Integer Darstellung.

Die Steuerung der Fahrzeugplattform erfolgt für die Erprobung der Regelungsparameter zunächst in einem einzigen Thread, der die Pushbuttons des Entwicklungsboards auf Benutzereingaben überwacht. Über die Pushbuttons können die Parameterwerte zur Laufzeit angepasst werden. Die Auswahl des anzupassenden Wertes erfolgt über *PB 4*, mit *PB 3* und *PB 2* wird der Wert erhöht bzw. verringert, wobei das Intervall der Erhöhung/Verringerung abhängig vom gewählten Wert ist. Gleichzeitig sind die Wertebereich durch die

Register	Wort	Wert	Darstellung	Beschreibung
sw_reg_0	val_1	lap_pos_x	S[G14|F2]	die x-Position des errechneten *LAP*
	val_0	lap_pos_y	S[G14|F2]	die y-Position des errechneten *LAP*
sw_reg_1	val_1	v_ist	S{G9|F0]	ermittelte IST-Geschwindigkeit aus dem V-CONTROLLER
	val_0	alpha_ist	S[G14|F2]	der Lenkwinkel α, der an den Fahrtensteller übertragen wird
sw_reg_2	val_1	-	-	*frei*
	val_0	-	-	*frei*
sw_reg_3	val_1	-	-	*frei*
	val_0	-	-	*frei*
sw_reg_4	val_1	-	-	*frei*
	val_0	-	-	*frei*
sw_reg_5	val_1	v_soll	S[G9|F0]	Vorgabewert für die Geschwindigkeitsregelung
	val_0	alpha_soll	S[G14|F2]	für Ausweichmanöver, kann der Lenkwinkel in Zusammenspiel mit dem `alpha_switch` vorgegeben werden
sw_reg_6	val_1	roi_y_pos	U[G10|F0]	beschreibt den Vorgabewert der ROI y-Position in
	val_0	distance	U[G10|F0]	Vorgabewert für den PD-Regler für die
sw_reg_7	val_1	vreg_p1	U[G5|F8]	p_1 aus Gleichung 47 für die Geschwindigkeitsregelung
	val_0	vreg_p2	U[G5|F8]	p_2 aus Gleichung 47 für die Geschwindigkeitsregelung
sw_reg_8	val_1	lap_val	S[G14|F2]	nach Gleichung 57: $\frac{LAD^2}{2*L}$
	val_0	LAD	U[G8]	beschreibt den Vorgabewert für den Abstand des LAP zum Ursprung des Fahrzeugkoordinatensystems (vgl. Abb. 4.13)
sw_reg_9	val_1	-	-	*frei*
	val_0	pathfinder_ctrl	bits	Das Kontrollregister für das \modze{Pathfinder} Modul, mit dem die Algorithmen zur Fahrspurführung und die Vorgabe des Lenkwinkels ausgewählt werden kann.

Tabelle 4.5 – Inhalte der Software-Register zur Steuerung und Auswertung der Fahrzeugregelung; der Wert des Wortes bezieht sich auf Bit 31 bis 16 für val_1 und Bit 15 bis 0 für val_0, die Darstellungsform wird in signed/unsigned [U/S], der Bitanzahl für den Ganzzahlbereich [G] und der Anzahl der Bits für die Fließkommadarstellung [F] angegeben.

Software Implementierung begrenzt, um ungültige Werte auszuschließen. Die Veränderungen der Einstellungen werden zusätzlich über die Standardausgabe von **microblaze_0** über den RS232 UART ausgegeben. Die folgende Übersicht zeigt die Einstellungsoptionen der Steuerungssoftware mit den gültigen Wertebereichen und Intervallen (vgl. Abb. 4.6):

Die Änderung von Inhalten der Software-Register erfolgt über eine *Memory Mapped*-Adressierung, wobei die direkten Zugriffe durch C-Makros implementiert sind (vgl. Listing 4.7). Dabei müssen die Formate der Datentypen abhängig von der Verwendung der

Mode*	Beschreibung	Intervall	Wertebereich	Einheit	SW-Register
DIST	Abstand zur Fahrspurmarkierung	5	[0 bis 50]	cm	sw_reg_6 / val_0
ROI_Y	y-Position der dynamischen ROI	5	[0 bis 175]	Pixel	sw_reg_6 / val_1
LAD	Look-Ahead-Distance	5	[30 bis 90]	cm	sw_reg_8 / val_0
SPEED	Vorgabewert der Geschwindigkeitsregelung	10	[0 bis 300]	cm/s	sw_reg_5 / val_1
ALPHA	Vorgabewert des Lenkwinkels	1	[-20 bis 20]	°	sw_reg_5 / val_0
FGRAB	Frame Grabber Modus: PB3 = startet Bildübertragung des Framegrabbers PB2 = - PB1 = Fortlaufende Ausgabe aller Registerwerte ein/aus				
CTRL	Pathfinder Kontrol-Registers: PB3 = alpha_switch ein/aus PB2 = sende Ausgabe über Bluetooth ein/aus PB1 = Ausgabe des akutellen Wertes				sw_reg_9 / val_0

* PB4 zum Wechsel der Modi

Tabelle 4.6 – Übersicht zur Benutzereingabe über die Pushbuttons; *PB 3* und *PB 2* erhöhen bzw. verringern die Werte in den Registern um den in der Intervall-Spalte angegebenen Wert.

Software-Register Inhalte im SAV CONTROL IP eingehalten werden. Die Erweiterung oder Änderung der Software-Register Belegung muss stets in beiden Teilsystemen erfolgen; zum einen in der Software des MicroBlaze Systems und zum anderen in der VHDL Top Entity des SAV CONTROL Moduls.

```
void savCtrl_dispAlpha(){
  /* alpha-FORMAT  15   |14    |13-       -0|
   *                sign| fixed|  fractional|
   * alpha is in range from -20° to 20° (-0.35.. to 0.35..)*/
  //fract_part and int_part are used twice;
  //for bit conversion and to display the degree (double) value
  u16 alpha,fract_part, fix_part;
  double degrees;
  char signa = ' ';
  //alpha in swreg1 val_0
  alpha = savCtrl_getCurrentAlpha(); //performs READ_MEMORY(SAV_CON_SWREG1_ADDR,alpha);
  if (alpha > 0x8000){
    signa = '-';
    alpha = ~alpha;
  }
  fix_part = (alpha >> 14);
  fract_part = (alpha << 2) >> 2;
  degrees = (atan( fract_part * qf_fract_pow14) * 180 / PI);
  fix_part = (u16)degrees;
  fract_part = (u16)((degrees*1000.0) - (fix_part*1000));
  xil_printf("A:%c%2d.%03d°\r\n",signa,fix_part,fract_part);
}
```

Listing 4.7 – C-Funktion zur Ausgabe des Lenkwinkels; mit dem `READ_MEMORY` Makro wird der Wert aus Software-Register 1 gelesen und zur Ausgabe die Q-Format Darstellung aufgelöst.

Empfang und Verarbeitung der Laserscanner Daten

Die Bereitstellung der Laserscanner Daten für die Verwendung in der Spurführung erfolgt durch die Anwendungssoftware auf **microblaze_1**, die eine Liste der erkannten Objekte im Shared Memory für die Auswertung durch **microblaze_0** ablegt. Dazu werden folgende Teilfunktionen innerhalb des `laser_measurement` Threads zur Erzeugung der Objekt-Liste durchlaufen(vgl. Abb. 4.16):

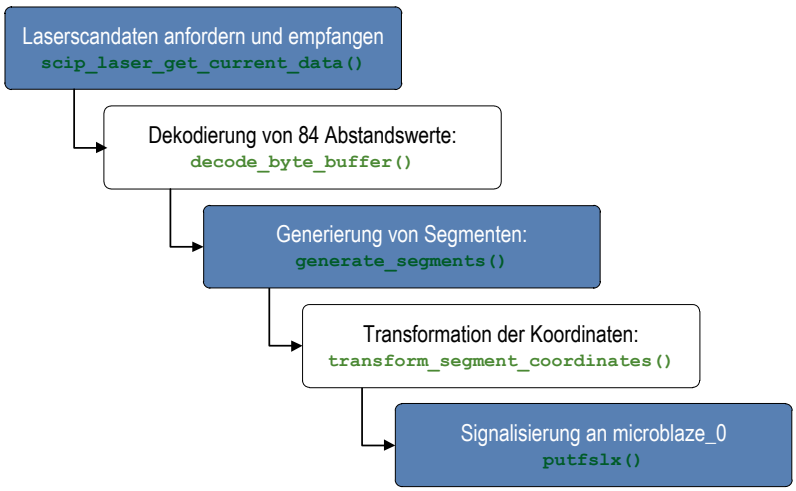


Abbildung 4.16 – Der Aufbau des Threads zur Aufnahme und Segmentierung der Laserscannerdaten.

Zur Kommunikation mit dem Laserscanner wird das SCIP 2.0 Protokoll verwendet [13], über das einerseits die Parametrierung des Laserscanners erfolgt und zweitens die Entfernungswerte empfangen werden. Der Scanner wird auf einen Scanbereich von 90° mit einem *Cluster Count* von 3 eingestellt, wodurch der Laserscanner die Scanwerte 259 bis 509 in 84 Messwerten mit einer Auflösung von 1,05° überträgt (vgl. Abb. 3.17 und [32]). Die Entfernungswerte werden als 12-Bit ASCII Rohdaten von **microblaze_1** über den UART LASER mit einer Baudrate von 115.200 Bits/s empfangen. Jeder Messwert wird durch zwei ASCII Zeichen dargestellt, wobei sich der Entfernungswert in Millimeter Darstellung aus jeweils 6-Bit eines Zeichens zusammensetzt. Die Dekodierung der Rohdaten erfolgt durch folgende Schritte (vgl. Tab. 4.7) in der `decodeData()` Funktion:

Die dekodierten Daten werden in einem `decoded_data_buffer[84]` Array abgelegt, wobei die Eingangsreihenfolge der Daten erhalten wird, um die Zuordnung der Messwerte zum Winkelbereich von -45° bis +45° für die Erstellung von Polarkoordinaten zu verwenden. Zusätzlich werden am Ende des Dekodierungsschrittes Messwerte, die einen definierten Schwellwert von 2.000 mm überschreiten und somit außerhalb des Aufmerksamkeitsbereiches liegen, aus der Berechnung durch die Zuweisung des Wertes Null genommen.

Dekodierungsschritt		Beispiel
1.	ASCII Darstellung	`C B`
2.	Hexadezimale Darstellung	$43_H 42_H$
3.	ASCII Offset abziehen (-30H)	$13_H 12_H$
4.	Binäre Darstellung von jeweils 6-Bit	$010011_2\ 010010_2$
5.	zu 12-Bit zusammenführen	010011010010_2
6.	Darstellung in mm	`1234 mm`

Tabelle 4.7 – Dekodierung der ASCII Werte aus dem SCIP 2.0 Protokoll; die zwei ASCII Zeichen enthalten jeweils 6-Bit des Abstandswertes vom Laserscanner zum Objekt [13].

Die Segmentierung der Messwert erfolgt durch den „Successive Edge Following“ Algorithmus, wobei das Array schrittweise durchschritten wird und die Entfernungswerte nach der Vorschrift 40 durch die in Tabelle 3.2 beschriebenen Schwellwerte gruppiert werden (vgl. Kap. 3.3). Aufgrund der Reihenfolge der Messwerte im Array lassen sich die Polarkoordinaten der detektierten Punkte ermitteln, wobei der Scanbereich beginnend bei +45° gegen den Uhrzeigersinn abgearbeitet wird.

Für eine neues Segment wird der erste Messwert als rechter Punkt des Segments R_x gespeichert und dessen Differenz mit dem folgenen Entfernungswert R_{x+1} mit dem dynamisch angepassten Schwellwert verglichen. Ist die Differenz geringer als der Schwellert, wird R_{x+1} zum neuen rechten Punkt und der alte R_x Wert als linker Punkt des Segments detektiert. Dieser Vorgang wird fortgesetzt, bis die Differenz der Abstandswerte den Schwellwert überschreitet; gleichzeitig wird die kürzeste Distanz zu einem Punkt für ein Segment zwischengespeichert. Das vollständig erkannte Segment wird durch die drei Punkte (linker Punkt, rechter Punkt und nächstgelegener Punkt (vgl. Kap. 3.3)) dargestellt, mit einer ID versehen und in der `Segment` Struktur in einer verketteten Liste abgespeichert (vgl. Listing 4.8).

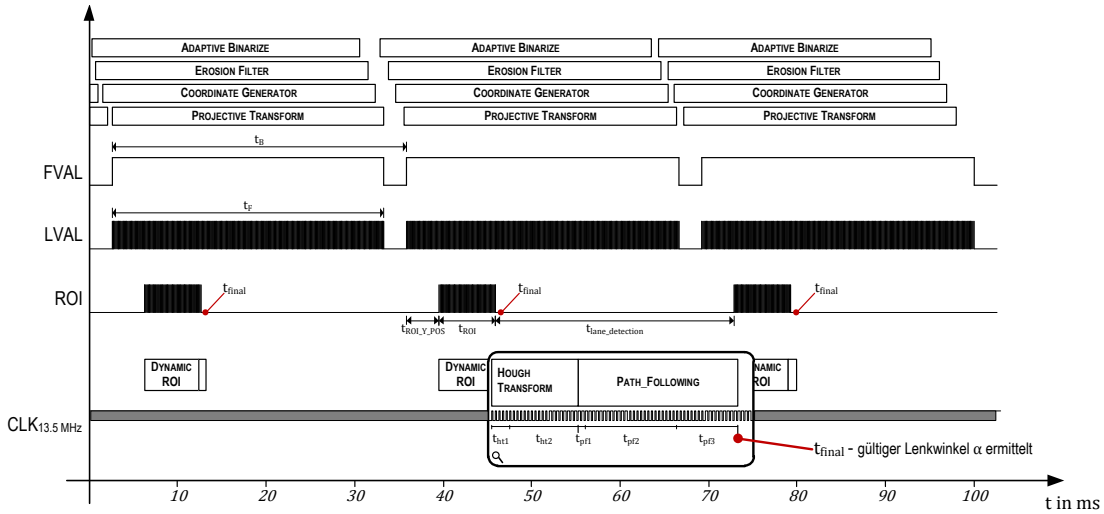
```
typedef struct
{
  double Distance;
  double Angle;
}SegmentPointPolar;
typedef struct
{
  double X;
  double Y;
}SegmentPointCartesian;
typedef struct
{
  unsigned short ID;                    // unique ID of the segment
  double Width ;                        // max. width of the segment
  SegmentPointPolar Left;               // polarcoordinate of the left segment point
  SegmentPointPolar Right;              // polarcoordinate of the right segment point
  SegmentPointPolar Nearby;             // polarcoordinate of the nearest segment point
  SegmentPointCartesian LeftCartesian;  // cartesiancoordinate of the left segment point
  SegmentPointCartesian RightCartesian; // cartesiancoordinate of the right segment point
  SegmentPointCartesian NearbyCartesian;// cartesiancoordinate of the nearest segment point
}Segment;
```

Listing 4.8 – Die Struktur zur Speicherung der Segmentparameter enthält die Position der drei Punkte sowohl in Polar- als auch in kartesischen Koordinaten.

Segmente, die identische L_x und R_X Punkte haben, bestehen nur aus einem Punkt und werden zur Datenreduktion verworfen. Mit dem Funktionsaufruf von `transform_segment_coordinates()` werden für alle Einträge der Segment-Liste die kartesischen Koordinaten nach Gleichung 41 und Gleichung 42 berechnet und in die Struktur aufgenommen. Abschließend wird die maximale Breite zwischen dem linken und dem rechten Punkt des Segments nach Gleichung 43 berechnet und als `Width` in die Segment-Struktur übernommen. Die Liste der erkannten Segmente wird in einer Sektion im gemeinsamen Speicher abgelegt, damit die Objekte von **microblaze_0** zur Anpassung der Steuerungsparameter verwendet werden können. Gleichzeitig signalisiert **microblaze_1** dem *Master* Prozessor über die FSL Schnittstelle, dass die Berechnung von neuen Objektdaten beendet wurde.

5 Ergebnisanalyse

Als Grundlage für die Integration der Teilkomponenten in ein einheitliches Gesamtsystem wurden die Ergebnisse und Konfigurationen der Vorarbeiten aus [20] und [26] zur Fahrspurerkennung und Lenkwinkel Bestimmung untersucht, um die Vorgehensweise innerhalb der Teilschritte zu durchdringen und die Parameterierung für das SAV zu übernehmen bzw. entsprechend anzupassen. Zur Analyse wurden die Berechnungsschritte in Matlab Funktionen abgebildet sowie deren Funktionsweise innerhalb abgeschlossener Teilsysteme mit dem System Generator simuliert. Zudem wurde das Zeitverhalten der Prozessorelemente der Bildbearbeitungspipeline detaillierter untersucht (vgl. Abb. 5.1). Die Kamerabilder, mit



Name	Zeit	Beschreibung	
t_P	74,074 ns	Pixel	13,5 Mhz
t_Z	127,11 µs	Zeile (mit Blanking)	1716 * 13,5 Mhz
t_B	33,3031 ms	Frame (mit Blanking)	262 * t_Z
t_F	30,56 ms	Frame	240 * t_Z
t_{ROI}	6,355 ms	ROI	50 * t_Z
$t_{lane\_detection}$	26,9475 ms	Start nächstes Bild	t_B -t_{ROI}
$t_{ROI\_Y\_POS\_start}$	3,177 ms*	Anfang der ROI nach FVAL	ROI_Y_POS * t_Z
$t_{ROI\_Y\_POS\_end}$	9,533 ms*	Ende der ROI	(ROI_Y_POS + 50) * t_Z

* für ROI_Y_POS = 25

Name	Modul / Funktion	Takte	Zeit	Komponente
t_{ht1}	Berechnung & Speicher	5	0,370 µs	HOUGH TRANSFORM
t_{ht2}	Maximasuche	19	1,407 µs	
t_{pf1}	UPDATE	2	0,148 µs	PATH FOLLOWING
t_{pf2}	LAP	25	1,851 µs	
t_{pf3}	ARCTAN	17	1,259 µs	
t_{final}		68	5,035 µs	

Abbildung 5.1 – Das Timingdiagram des PATHFINDER Moduls; die Ermittlung des Lenkwinkels ist bereits abgeschlossen, bevor alle Pixeldaten des aktuellen Bildes von der IMAGE PROCESSING-Pipeline verarbeitet wurden.

Blanking Anteilen, werden mit einer Bildrate von t_B = 33,30 ms von der Prozesskette des

IMAGE PROCESSING Moduls verarbeitet, dabei werden die projektiv transformierten Bildkoordinaten kontinuierlich im Pixeldaten Takt von 13,5 MHz an die dynamische Ermittlung der ROI weitergeleitet. Diese nimmt eine Datenreduktion der Eingangswerte auf einen eingeschränkten Bildbereich innerhalb von t_{ROI} = 6,355 ms vor. Somit entsteht ein Zeitfenster zur Detektion der Fahrspur und zur Ermittlung der Lenwinkelstellgröße α von $t_{lane\_detection}$ = 26,94 ms. Die Hough-Transformation zur Fahrspurerkennung und die Komponenten zur Anwendung der Fahrspurführungsalgorithmen können unabhängig von der Pixelfrequenz ausgeführt werden. Diese Prozessorelemente sind mit Multizyklusdatenpfaden implementiert und werden für Teilberechnungen übertaktet. Somit liegt als Ergebnis der PATHFINDER Komponente, die Lenkwinkelstellgröße α nach t_{final} = 5,035μs vor.

5.1 Kalibrierungsparameter für die PT

Die Festlegung der Kalibrierungsmatrix B für die projektive Entzerrung des Kamerabildes wird durch die Entwicklung eine Matlab Moduls unterstützt (vgl. Kap. 3.2.2; Abb. 5.2). Eine Neukalibrierung ist erforderlich, sobald sich die Ausrichtung oder Position der Ka-

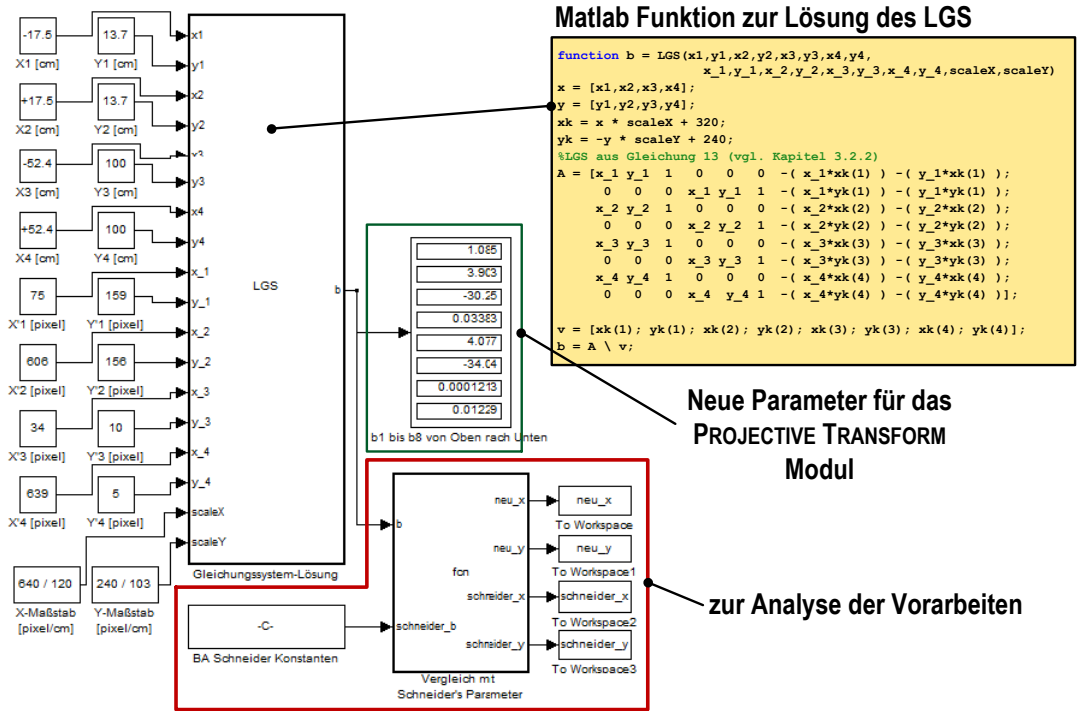


Abbildung 5.2 – Das Matlab Modul zur Ermittlung der Kalibrierungsparameter für die PT wurde zudem zur Analyse der bestehenden Parametereinstellungen aus [20] und [26] verwendet.

mera auf der Fahrzeugplattform ändert, da die projektive Transformation die Translation

der Pixeldaten aus der Kameraperspektive in eine dem Fahrzeugkoordinaten entsprechende Aufsicht vornimmt (vgl. Abb. 3.7). Das Matlab Modul nimmt dazu vier manuell ermittelte Punktpaare aus dem metrischen Fahrzeugkoordinatensystem (X_k,Y_k) und den korrespondierenden Bildpunkten (X'_k,Y'_k) sowie die Skalierungsfaktoren zur Lösung des linearen Gleichungssystems auf. Die erzeugten Kalibrierungsparameter werden in den PROJETIVE TRANSFORM System Generator Block der Bilddatenvorverarbeitung (vgl. Kap. 4.1.3) als Konstanten eingetragen und für die Transformation in die entzerrten Koordinaten innerhalb des Pixeldatenstroms verwendet.

Gleichzeitig wurde das Matlab Modul zur Analyse der bestehenden Arbeiten von [20] und [26] verwendet, um die Parametrierung des Transformationsmoduls zu untersuchen und die Auswirkung von verschiedenen Einstellungen auf das Ausgangsdatenbild zu testen. Die mit den ermittelten Kalibrierungsparametern darstellbaren Koordinatenbereiche wurden dazu in Testdateien ausgegeben, um einen optischen Eindruck über den Pixelbereich der Zielbilder zu bekommen (vgl. Abb. 5.3 (a)). Dabei werden darstellbare Koordinaten als weiße Pixel des 640x240 Pixel großen Bildes angezeigt.

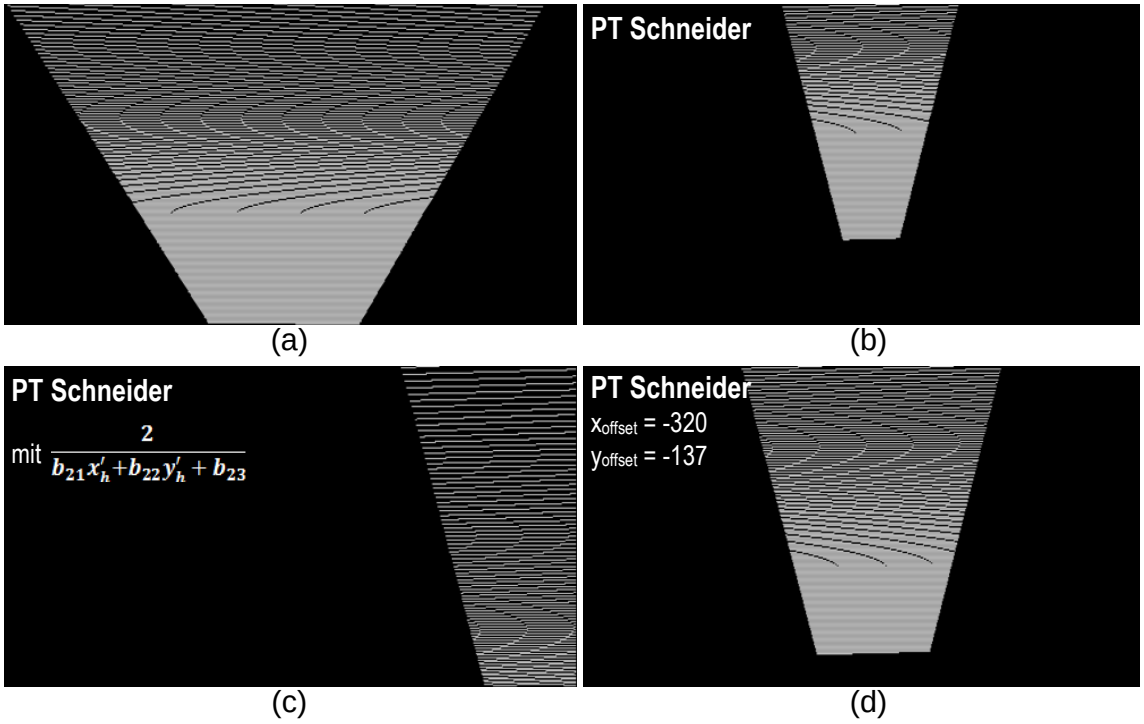


Abbildung 5.3 – Der Vergleich der Einstellung zur projektiven Transformation; (a) zeigt den Darstellungsbereich für die PT des SAV, (b), (c) und (d) zeigen die PT-Ergebnisse nach [26] in Teilschritten. (b) mit einheitlichem Skalierungsfaktor, (c) reduzierter Darstellungsbereich und (d) Verschiebung des Bereiches durch Offsets.

Des Weiteren wurden die Berechnungsschritte der Vorarbeiten innerhalb der Systemgenera-

tor Blöcke in einer Matlab Funtkion nachgebildet, um auch deren Darstellungsbereiche zu visualisieren und die Abhängigkeiten zur Fahrspurerkennung offen zu legen. Die Skalierung wurde in [26] durch einheitliche Faktoren von 5mm pro Pixel für die x- und y-Koordinaten vorgenommen, wodurch eine Stauchung des Darstellungsbereichs in x-Richtung erfolgt (vgl. Abb. 5.3 (b)). Aus der Nachbildung der Berechnungsschritte konnte zudem eine Datenreduktion erkannt werden, die den Koordinatenbereich der Kameradaten auf ein Viertel des Gesamtbildes durch die Anpassung des Divisor Kehrwerts (vgl. Gl. 44) um den Faktor 2 reduziert (vgl. Abb. 5.3 (c)). Zusätzlich wurden in [26] Offsets verwendet, durch die der reduzierte Darstellungsbereich in x- und y- Richtung verschoben wird (vgl. Abb. 5.3 (d)).

Die Anaylse der projektiven Transformation aus [26] ergab, dass eine Fokussierung auf einen eingeschränkten Bildbereich bereits vor der Anwendung der dynamische ROI innerhalb des PROJECTIVE TRANSFORM Moduls erfolgt. Dadurch liegen die Pixelabstände des Kamerabildes in einem vergrößerten Verhältnis zueinander vor, was sich direkt auf die Ermittlung der Fahrzeugposition und der Fahrbahnmarkierung sowie auf die Berechnung des LAP auswirkt. In der Implementierung des SAV sind diese Abstände somit wesentlich kleiner dimensioniert, wodurch gegebenenfalls eine Anpassung der Datentypen innerhalb der Module zur Ermittlung des Lenkwinkelsollwerts erforderlich ist.

5.2 System Generator Analyse der Fahrspurführungsmodule

Die Verifikation der RTL Modelle zur Ermittlung des Lenkwinkels α wurde in einer Analyse der PATH FOLLOWING Module vorgenommen [26]. Dazu wurden die Teilkomponenten des Fahrspurführungssystems UPDATE, LAP und ARCTAN kaskadiert und die Auswertung der Laufzeiten für die Teilberechnungen simuliert (vgl. Abb. 5.4). Dabei werden die `start`

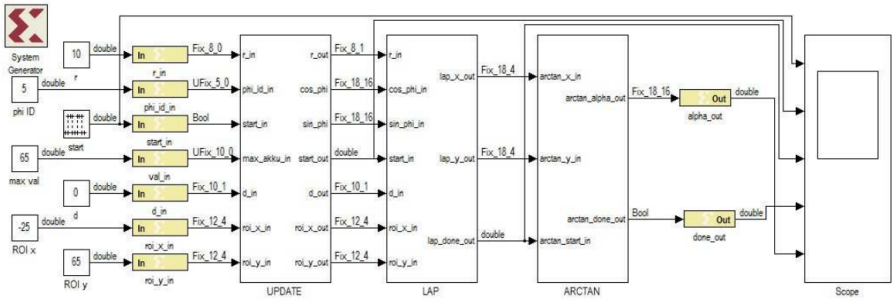


Abbildung 5.4 – Kaskadierte Teilkomponenten des PATH FOLLOWING Moduls in einem System Generator Simulationsaufbau; die Eingangssignale werden über die Input/Output Ports in das Xilinx Fixed-Point Format konvertiert.

bzw. `done` Signale der Teilschritte über einen Scope - System Generator Block visualisiert

und über die einheitliche Darstellung ausgewertet. Die Simulation der Laufzeiten ergab, dass bereits 44 Takte nachdem ein neues Ergebnis des LANE DETECTION Moduls durch `r` und `phi_id` anliegt, die Berechnung des entsprechenden Lenkwinkels α abgeschlossen ist (vgl. Abb. 5.5). Dabei werden die Sinus- und Kosinuswerte innerhalb von zwei Takten (t = 12) nach Auslösen des Startsignals (t = 10) vom UPDATE Modul aus den Look-Up-Tabellen über das `phi_id` Signal ausgelesen und zusammen mit dem Soll- und Istabständen `r` und `d` an das LAP Modul übergeben. Die Berechnung des LAP erfolgt innerhalb der ASM in 25 Takten und liefert die x- und y-Koordinaten bei t = 37 an die Eingänge des ARCTAN Moduls. Dieses benötigt weitere 17 Takte zur Berechnung der Lenkwinkelstellgröße, bevor das Ergebnis bei t = 54 am Ausgang angezeigt wird.

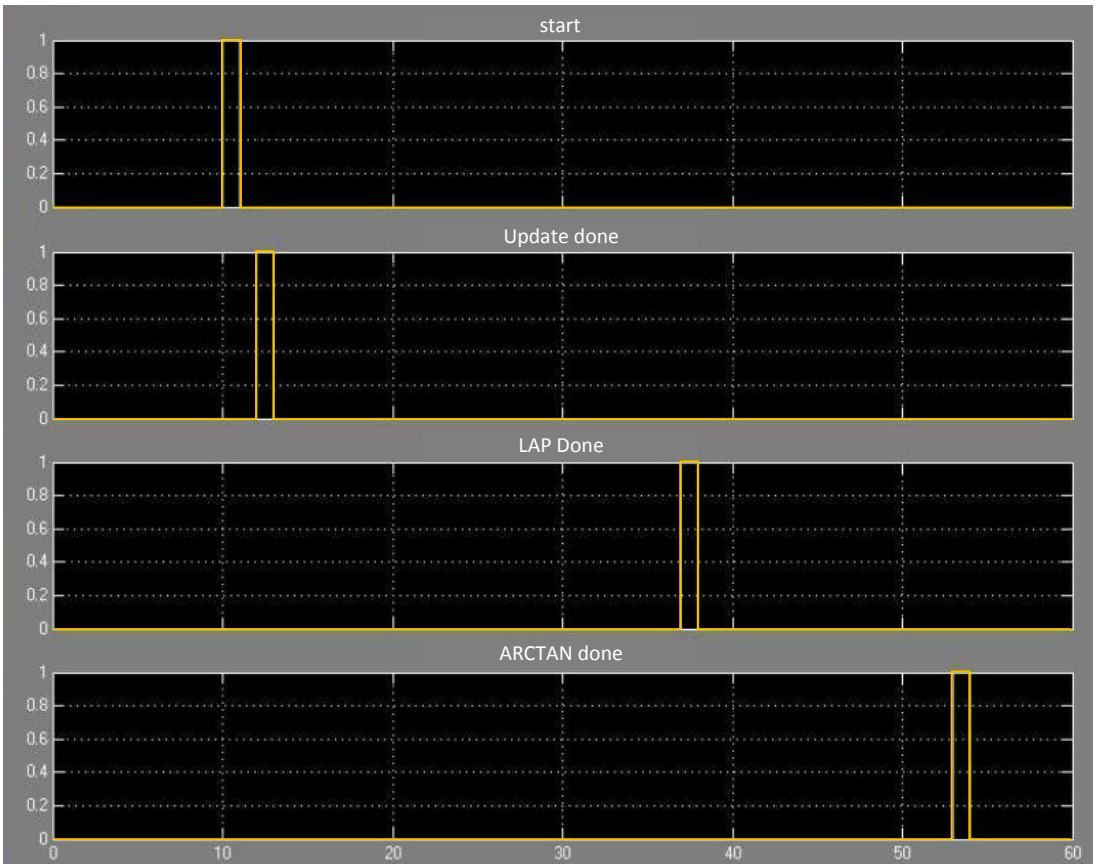


Abbildung 5.5 – Scope Ansicht zu den Laufzeiten der kaskadierten Teilkomponenten des PATH FOLLOWING Moduls.

5.3 Ressourcenbedarf des Gesamtsystems

Die Generierung der Teilkomponenten zu einem Gesamtsystem wurde für das Xilinx Spartan-3A DSP FPGA durchgeführt und ergab eine Ressourcenauslastung, bei der Verwendung von 20.610 der insgesamt 33.280 zur Verfügung stehenden LUTs, von 61% (vgl. Tab. 5.1).

Device Utilization Summary (actual values)			
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slice Flip Flops	13,259	33,280	39%
Number of 4 input LUTs	20,610	33,280	61%
Number of occupied Slices	13,932	16,640	83%
Total Number of 4 input LUTs	21,873	33,280	65%
Number of bonded IOBs	160	519	30%
Number of ODDR2s used	48		
Number of BUFGMUXs	9	24	37%
Number of DCMs	3	8	37%
Number of BSCANs	1	1	100%
Number of DSP48As	26	84	30%
Number of RAMB16BWERs	80	84	95%
Number of BSCAN_SPARTAN3As	1	1	100%
Number of RPM macros	1		
Average Fanout of Non-Clock Nets	3.10		

Tabelle 5.1 – Der Summary Report zeigt für die Verwendung der Ressourcen des Spartan-3A DSP FPGA, dass ausreichend Logikelemente zur Verfügung stehen, um eine spätere Integration von zusätzlichen Entwicklungen vorzunehmen.

Die LUTs werden in 83% der verfügbaren Slices implementiert, was durch die Modifikation der Place and Route Einstellungen für die Integration von weiteren Bestandteilen optimiert werden kann. Demnach bildet die ausgewählte FPGA-Plattform ausreichend Ressourcen für die Weiterentwicklung des SAV durch die Einbindung zusätzlicher Module.

Eine detailliertere Betrachtung der Ressourcenverteilung zeigt, dass der größte Ressourcenbedarf für die Umsetzung des MPMC zur Speicheranbindung, den MicroBlaze Prozessoren und dem SAV CONTROL IP beansprucht wird (vgl. Tab. 5.2). Im SAV CONTROL werden die Ressourcen hauptsächlich von der Realisierung der Multizykluspfad-Elemente zur projektiven Transformation, der Hough-Transformation und der LAP Bestimmung verwendet, wogegen die Bildbearbeitungskomponenten im Pipelinepfad nur in den Zeilen- und Bildzwischenspeichern einen erhöhten Ressourcenbedarf aufweisen.

Des Weiteren lassen sich aus der Übersicht der Ressourcenbelegung Optimierungen ableiten. Das HOUGH TRANSFROM Modul verwendet in diesem Zusammenhang lediglich vier der DSP48A Elemente, die mit Mulitiplizierer Einheiten für die Implementierung der rechenintensiven Hough-Transformation prädestiniert sind. Durch eine manuelle Zuweisung dieser Elemente in den System Generator Einstellungen kann die Ressourcenverteilung angepasst werden.

Module Name	Slice Reg	LUTs	BRAM	DSP48A
system	**13259**	**20610**	**80**	**26**
MPMC DDR2_SDRAM	3515	2221	17	0
LEDs_8Bit	106	65	0	0
PmodBT_UART	498	582	0	0
Push_Buttons	82	52	0	0
RS232_Uart_plb_0	124	117	0	0
boot_bram	0	0	32	0
cam_uart	133	127	0	0
clock_generator_0	4	0	0	0
debug_module	127	148	0	0
dlmb	6	12	0	0
framegrabber_0	28	9	0	0
fsl_framegrabber	78	71	1	0
fsl_microblaze	14	86	0	0
ilmb	6	2	0	0
interrupt_controller	105	78	0	0
microblaze_0	1334	2828	10	3
microblaze_1	1344	2795	10	3
mutex_0	99	91	0	0
plb_0	89	309	0	0
plb_1	83	248	0	0
proc_sys_reset_0	35	22	0	0
sav_connector	361	398	0	0
system	0	1	0	0
vehicle_LEDs	223	114	0	0
xps_timer	596	698	0	0

Module Name	Slice Reg	LUTs	BRAM	DSP48A
sav_control	4269	9279	10	20
CAM_IF	216	220	0	8
converter	14	30	0	8
deserialize	68	23	0	0
syncer	115	166	0	0
upsample	19	1	0	0
CLKMGR	4	1	0	0
VGA_if	406	1342	0	0
deinterer	384	1281	0	0
VGA_CTRL	22	61	0	0
VREG	157	402	0	1
pathfinder	3268	7243	10	11
image_processing	613	780	0	5
adaptive_binarize	11	24	0	0
coordinate_generator	23	39	0	0
erosion_filter	85	80	0	0
projective_transform	494	637	0	5
lane_detection	1383	3683	0	4
dynamic_roi	39	173	0	0
hough_transform	1344	3510	0	4
path_following	1002	2113	0	0
arctan	551	838	0	0
lap	251	693	0	0
mode_select	0	20	0	0
pd	35	427	0	0
update	108	191	0	0
pwm generation	81	191	0	0
result_illustration	189	424	10	2
image_buffer	31	161	10	2
result_overlay	158	263	0	0
receiver_switch	34	48	0	0

Tabelle 5.2 – Verteilung der Ressourcen Auslastung; neben der Speicheranbindung und den MicroBlaze Prozessoren wird der Hauptanteil der FPGA Ressourcen für die Pipeline- und Multizykluspfad Stufen des SAV CONTROL verwendet.

5.4 Vorschläge für weitere Technologieerprobungen

Durch die Integration der Fahrspurführung, der Geschwindigkeitsregelung und der Objekterkennung in ein einheitliches Gesamtsystem, steht eine Erprobungsplattform zur Verfügung, die als Ausgangsplattform für die Teilnahme am Carolo Cup dient. Für die weitere Inbetriebnahme des SAV ergeben sich neben der Einführung von dynamisch angepassten Parametern zur Geschwindigkeitsregelung und der weiteren Erprobung der Fahrzeugführungsalgorithmen in unterschiedlichen Fahrszenarien, folgende Ansätze:

- **Die Integration der ermittelten Objektdaten in das Fahrzeugkoordinatensystem:** Auf Grundlage der LAP Position und des berechneten Lenkwinkels wird das Fahrzeugkoordinatensystem über die Software-Register in die Steuerungsanwendung von

microblaze_0 übernommen; die Positionen der vom Laserscanner erkannten Objekte sind über das Shared Memory abzurufen. Auf Grundlage der Fahrzeugausrichtung werden die Objektdaten in das Fahrzeugkoordinatensystem übernommen, um durch die Berechnung von Ausweichpfaden den Lenkwinkel direkt anzupassen. Darüber hinaus ist die Aufnahme der IR-Sensor Daten für die seitliche Abstandsmessung in den Steuerungsprozess geplant, um eine detailliertere Modelldarstellung der Fahrzeugumgebung zu erzeugen.

- **Steuerungs- und Datenaufzeichnungssoftware über Bluetooth:**
  Die Inbetriebnahme der Bluetooth Verbindung zur Aufzeichnung von Systemdaten bei Erprobungsfahrten, erfordert die Spezifikation und Realisierung eines Protokolls zum Austausch von Daten zwischen dem SAV und dem Entwicklungsrechner; gleichzeitig muss dieses Protokoll in die Steuerungsplattform integriert werden. Zudem steht die Entwicklung einer Fahrzeugführungssoftware aus, die das Protokoll zur Steuerung des SAV einsetzt und zusätzlich Funktionen zur Aufzeichnung und Darstellung der Systemdaten bereithält.

- **Die Integration eines Einparkassistenten:**
  Die Software zur Erfassung der seitlichen Abstandswerte aus der IR-Sensor Auswertung und der in [1] implementierte automatische Einparkassistent werden in das SAV System integriert. In einem zweiten Schritt ist dessen Optimierung durch die Verwendung der Informationen der Fahrspurführung denkbar.

- **Erweiterungen zur Bestimmung der ROI:**
  Aktuell verläuft die Ausrichtung des Fahrzeugs durch die Erkennung der rechten Fahrbahnmarkierung. Durch das Auftreten von Hindernissen im Frontbereich des SAV, ergeben sich Situationen, in denen die Fahrspurmarkierung ausserhalb des Sichtbereiches der Kamera liegt; das erfolgt durch die Verdeckung der Fahrspurmarkierung durch das Objekt oder der angepassten Ausrichtung des Fahrzeugs. Um diese Situationen, in denen der Fahrspurerkennung keine gültigen Werte zur Bestimmung der Fahrzeugposition vorliegen, zu vermeiden, sind unterschiedliche Ansätze denkbar.
  Zunächst könnte die ROI bei Detektion eines Hindernisses im Kollisionsbereich des SAV durch einen festen Offset auf die linke Fahrspurmarkierung verschoben werden. Die Position ist durch die aktuelle Position der rechten Fahrspurmarkierung und der festgelegten Fahrbahnbreite der Carolo Cup Strecken im Fahrzeugkoordinatensystem zu bestimmen. Dieser Vorgang kann auch durch die Erzeugung einer zweiten ROI und einer zweifachen Auswertung durch die Hough-Transformation erfolgen, wobei die Berechnungsergebnisse durch die Steuerungssoftware für die linke oder rechte Fahrspurmarkierung an das PATH FOLLOWING Modul übertragen werden. Zusätzlich würde der Einsatz einer Kamera mit einem größeren Sichtbereich eine genauere Ermittlung von Umgebungsdaten zulassen.

6 Zusammenfassung

Diese Arbeit dokumentiert den Entwurf und die Implementierung einer FPGA-basierten MPSoC Plattform zur Steuerung eines autonomen Fahrzeugs, das im Rahmen des FAUST Projektes an der HAW Hamburg entwickelt wurde. Basierend auf der Analyse der Ergebnisse aus den Vorarbeiten in [1], [6], [8], [20], [17], [21], [26], [31] und [32] wurden die erzeugten Teilsysteme in die einheitliche Entwicklungsplattform des SAV integriert. Als Kernkomponente dient ein MicroBlaze-basiertes MPSoC, in das die *Hardware Accelerator* Komponenten zur Fahrzeugregelung und die Software Anwendung zur Steuerung der Fahrzeugplattform und Integration der Umgebungssensoren eingebunden wurden. Neben dem Entwurf und der Implementierung der Fahrzeugführungsplattform wurden der Aufbau und die Montage der Aktoren und Sensoren auf der Fahrzeugplattform für die Durchführung von Erprobungsfahrten vollzogen.

Die in den Vorarbeiten erzeugten Teilkomponenten wurden in verschiedenen Implementierungsweisen durch VHDL-Module, synthesefähige System Generator Blöcke oder als Softwareimplementierungen für ein MicroBlaze-basiertes SoC umgesetzt und dementsprechend in die SAV Plattform aufgenommen. Die Regelungsfunktionen für die Fahrzeuggeschwindigkeit und der Lenkwinkelstellgröße wurden in einem SAV CONTROL IP konzentriert, um die Integration in das Multiprozessorsystem zu erleichtern. Zur Anpassung von Systemparametern des SAV CONTROL zur Laufzeit wurde ein SAV CONNECTOR Modul erzeugt, das den Datenaustausch zwischen dem MicroBlaze-System und der Beschleunigerkomponente über Software-Register realisiert. Gleichzeitig werden Abstandswerte von einem Laserscanner durch einen zweiten MicroBlaze empfangen und für die objektbezogene Hinderniserkennung verarbeitet.

Die Ermittlung der Lenkwinkelstellgröße wird auf Basis von Kameradaten in einer Bildverarbeitungspipeline vorgenommen, die den Pixeldatenstrom durch eine adaptive Binarisierung und ein Erosionsfilter reduziert, eine projektive Transformation der Pixelkoordinaten aus der Kameraperspektive in die Aufsicht des Fahrzeugkoordinatensystems vornimmt und die Fahrspurmarkierung durch eine Hough-Transformation aus den Daten eines ROI-Bildausschnittes detektiert. Die Position der erkannten Fahrbahnmarkierung als Ergebnis der HT wird zur Bestimmung der Fahrzeugausrichtung und zur Anwendung der Fahrspurführungsalgorithmen Pure-Pursuit und Follow-the-Carrot verwendet, was in Teilkomponenten durch Multizyklusdatenpfade mit *Resource-Sharing* realisiert wurde. Die ermittelte Lenkwinkelstellgröße wird am Ende der Verarbeitungskette als PWM Signal an den

Fahrtensteller des SAV übermittelt. Daneben umfasst das SAV CONTROL Modul eine Geschwindigkeitsregelung, die durch die VHDL Implementierung eines PI-Reglers realisiert wurde und deren Parameter über die Software-Register zur Laufzeit in Abhängigkeit zur Ist-Geschwindigkeit angepasst werden können. Die Geschwindigkeitsstellgröße wird ebenfalls in ein PWM Signal umgesetzt und an den Fahrtensteller des SAV übertragen. Einen weiterer Bestandteil des SAV CONTROL IP stellt das RECEIVER SWITCH Modul dar, das als Sicherheitsmechanismus zum Wechsel zwischen manuellem und autonomen Betriebsmodus dient.

Zur Ermittlung von Kalibrierungsparametern für die Transformation des perspektivisch verzerrten Kamerabildes wurde der Aufbau des PROJECTIVE TRANSFORM Moduls detaillierter analysiert und für einen allgemeineren Einsatz angepasst. Zudem wurde ein Matlab Modul zur Ermittlung der Kalibrierungswerte erstellt, durch das gleichzeitig eine Auswertung des projektiv transformierten Darstellungsbereichs vorgenommen wird. In Abhängigkeit der transformierten Pixelkoordinaten wurden ferner die Koordinatensysteme zur Bestimmung der Fahrzeugposition auf Grundlage der Bilddaten und zur Anwendung der Fahrspurführungsalgorithmen genauer untersucht.

Als Entwicklungsplattform wurde das Xilinx Spartan-3A DSP Starter Board ausgewählt, dessen Spartan-3A DSP FPGA die Grundlage des MPSoC zur Fahrzeugsteuerung bildet. Das DUAL-MICROBLAZE-SYSTEM umfasst zwei MicroBlaze Softcore Prozessoren die in einer AMP-basierten Prozessorarchitektur für spezielle Teilaufgaben konfiguriert wurden. Als Basis der Softwareanwendungen wird das Xilkernel Echtzeitbetriebssystem eingesetzt, um die Verwendung von Threadabläufen nach dem POSIX-Standard zu realisieren. Darüber hinaus werden die Peripherie-Komponenten des Entwicklungsboards über das Multiprozesssorsystem eingebunden und stellen die Kommunikation mit einem Entwicklungsrechner zur Bedienung des SAV und zur Datenaufzeichnung her. Die Anpassung der Software-Register Inhalte für die Parametrierung des SAV CONTROL IPs erfolgt dabei über die Steuerungssoftware auf **microblaze_0**. Der zweite MicroBlaze stellt die Verbindung zum Hokuyo URG-04LX Laserscanner her, nimmt dessen Abstandswerte auf, dekodiert diese und verarbeitet die Entfernungen zur objektbasierten Hinderniserkennung. Die Ergebnisse der Objekterkennung werden in einer separaten Speichersektion des externen DDR2 SDRAM abgelegt, die als Shared Memory realisiert wurde, womit **microblaze_0** ebenfalls Zugriff auf die Daten hat. Die Synchronisation der Zugriffe erfolgt über einen Hardware Mutex. Des Weiteren wurden für die MicroBlaze Prozessoren, neben einem 16-kB großem lokalen BRAM Speicher, eigene Speichersektionen im SDRAM angelegt, die mit jeweils 8-kB großen Daten- und Instruktionscaches für einen optimierten Speicherzugriff implementiert sind.

Mit der SAV Plattform wurde eine Entwicklungsplattform erzeugt, die in Erprobungsfahrten den autonomen Fahrbetrieb vollzogen hat. Die integrierten Kommunikationskomponenten lassen für weitere Testfahrten die Aufzeichnung von Daten zur Analyse der Systemparameter und Anpassung der Fahrzeugkonfiguration zu. Zudem stehen ausreichend Logik Res-

sourcen auf dem verwendeten Spartan-3A DSP FPGA Baustein zur Verfügung, um zusätzliche Funktionalitäten in die Plattform zu integrieren und die Entwicklung des Gesamtsystems fortzuführen.

Literaturverzeichnis

[1] AHAD, Walli: *HW-/SW-Codesign eines Einparkassistenten mit Infrarot-Abstandssensorik für eine SoC-Plattform eines autonomen Fahrzeugs*, Hochschule für Angewandte Wissenschaften - Hamburg, Bachelor Thesis, 2011

[2] AMDAHL, Gene M.: Validity of the single processor approach to achieving large scale computing capabilities. In: *AFIPS spring joint computer conference* (1967)

[3] ASHENDEN, Peter: *Digital Design: An Embedded Systems Approach Using VHDL*. Frankfurt am Main : Morgan Kaufmann, 2008. – ISBN 9978-0-12-369528-4

[4] AVNET: *EXP Expansion Connector - Specification*, 2007. – URL `http://www.em.avnet.com/en-us/design/linecard/Documents/Xilinx/exp_specification_v1_4.pdf`. – from companies webpage [Jan 2012]

[5] BENGEL, Günther ; BAUN, Christian ; KUNZE, Marcel ; STUCKY, Karl-Uwe: *Masterkurs Parallele und Verteilte Systeme*. Wiesbaden : Vieweg+Teubner, 2008. – ISBN 978-3-8348-0394-8

[6] BERNGARDT, Roman: *Bluetooth-Schnittstelle zur Parametrierung und Betriebsdatenerfassung eines SoC-basierten autonomen Fahrzeugs*, Hochschule für Angewandte Wissenschaften - Hamburg, Bachelor Thesis, 2012

[7] BILSKI, Göran ; MOHAN, Sundararajarao ; WITTIG, Ralph: *Designing Soft Processors for FPGAs*. San Fransisco, CA : Morgan Kaufman, 2007. – ISBN 0-12-369526-0

[8] BORDASCH, Heiko: *VHDL-Modellierung einer Geschwindigkeitsregelung für ein autonomes Fahrzeug implementiert auf einer SoC - Plattform*, Hochschule für Angewandte Wissenschaften - Hamburg, Bachelor Thesis, 2009

[9] BUNDESMINISTERIUM FÜR BILDUNG UND FORSCHUNG: *Allgemeine Hinweise zur BMBF-Förderung von Embedded Systems*. Januar 2012. – URL `http://www.pt-it.pt-dlr.de/de/1852.php`

[10] CAROLO CUP: *Studenten entwickeln autonome Modellfahrzeuge, Regelwerk*. Januar 2012. – URL `http://www.carolo-cup.de`

[11] HAW HAMBURG, DEPARTMENT INFORMATIK: *FAUST Fahrerassistenz- und Autonome Systeme*. Januar 2012. – URL `http://www.informatik.haw-hamburg.de/faust.html`

[12] HESSELL, Fabiano ; KENT, Kenneth ; PNEVMATIKATOS, Dionisios: Embedded Systems - New Challenges and Future Directions. In: *ACM Transactions on Embedded Computing Systems* (2008)

[13] HOKUYO AUTOMATIC CO., LTD.: *Communication Protocol Specification SCIP 2.0 Standard*, 2006. – URL `http://http://www.hokuyo-aut.jp/02sensor/07scanner/download/data/URG_SCIP20.pdf`. – from companies webpage [Jan 2012]

[14] HOKUYO AUTOMATIC CO., LTD.: *Scanning Laser Range Finder URG-04LX - Specifications*, 2008. – URL `http://www.hokuyo-aut.jp/02sensor/07scanner/download/data/URG-04LX_spec.pdf`. – from companies webpage [Jan 2012]

[15] HUERTA, Pablo ; CASTILLO, Javier ; PEDRAZA, Cesar ; CANO, Javier ; MARTINEZ, Jose I.: *Symmetric multiprocessor systems on FPGA*. 2009. – URL `http://ieeexplore.ieee.org`. – IEEE Digital Library [April 2010]

[16] JACK, Keith: *Video Demystified: A Handbook for the Digital Engineer*. 4. Newnes, 2005. – ISBN 9780750678223

[17] KIRSCHKE, Marco: *Echtzeitbildverarbeitung mit einer FPGA-basierten Prozessorelementkette in einem Fahrspurführungssystem*, Hochschule für Angewandte Wissenschaften - Hamburg, Bachelor Thesis, 2009

[18] LRP ELECTRONIC GMBH: *Crawler Brushless 21,5 Turns und A.I. Brushless Pro Reverse Digital*. Januar 2012. – URL `http://www.lrp.cc`

[19] LUTZ, Holger ; WENDT, Wolfgang: *Taschenbuch der Regelungstechnik*. Frankfurt am Main : Harri Deutsch, 2007. – ISBN 978-3-8171-1807-6

[20] MELLERT, Dennis: *Modellierung einer Video-basierten Fahrspurerkennung mit dem Xilinx System Generator für eine SoC-Plattform*, Hochschule für Angewandte Wissenschaften - Hamburg, Bachelor Thesis, 2010

[21] NURETTIN PÜSKÜL Özgür: *SoC-basierte Bildverarbeitungspipeline für eine Fahrspurerkennung und -visualisierung*, Hochschule für Angewandte Wissenschaften - Hamburg, Bachelor Thesis, 2010

[22] PETERS, Falko: *FPGA-basierte Bildverarbeitungspipeline zur Fahrspurerkennung eines autonomen Fahrzeugs*, Hochschule für Angewandte Wissenschaften - Hamburg, Bachelor Thesis, 2009

[23] PLESSL, Christian: *Hardware/Software Codesign*. Universität Paderborn, 2010. – Paderborn Center for Parallel Computing

[24] RAUBER, Thomas ; RÜNGER, Gudula: *Parallele Programmierung*. Heidelberg : Springer, 2007. – ISBN 3-540-46549-2

[25] REICHARDT, Jürgen ; SCHWARZ, Bernd: *VHDL-Synthese - Entwurf digitaler Schaltungen und Systeme*. 5. München : Oldenbourg Verlag, 2009. – ISBN 978-3-486-58987-0

[26] SCHNEIDER, Christian: *Ein System-on-Chip-basiertes Fahrspurführungssystem*, Hochschule für Angewandte Wissenschaften - Hamburg, Bachelor Thesis, 2011

[27] SHARP CORPORATION: *GP2D12 Optoelectronic Device - Datasheet*, 2005. – URL `http://www.sharpsma.com/webfm_send/1203`. – from companies webpage [Jan 2012]

[28] SIADAT, Ali ; KASKE, Axel ; KLAUSMANN, Siegfried ; DUFAUT, Michael ; HUSSON, Rene: An Optimized Segmentation Method For A 2D Laser-Scanner Applied To Mobile Robot Navigation. In: *Proceedings of the 3rd IFAC Symposium on Intelligent Components and Instruments for Control Applications* (1997)

[29] SONY CORPORATION: *Color Camera Module - Technical Manual FCB-PV10*, 2006. – URL `http://pro.sony.com/bbsccms/assets/files/mkt/indauto/manuals/FCB-PV10_Technical_Manual.pdf`. – from companies webpage [Jan 2012]

[30] THE ITU RADIOCOMMUNICATION ASSEMBLY: *Interface for digital component video signals in 525-line and 625-line television systems operating at the 4:2:2 level of Recommendation ITU-R BT.601*. Dezember 2007. – URL `http://www.itu.int/rec/R-REC-BT.656/en`

[31] TIMOSCHENKO, Alexander: *Implementierung einer Geschwindigkeitsregelung als Prozessor-Element auf einer SoC-Plattform für ein autonomes Fahrzeug*, Hochschule für Angewandte Wissenschaften - Hamburg, Bachelor Thesis, 2011

[32] WILKEN, Heiko: *Laserscanner-basierte Objekterkennung für ein Fahrspurführungssystem auf einer Multiprozessor FPGA-Plattform*, Hochschule für Angewandte Wissenschaften - Hamburg, Master Thesis, 2012

[33] XILINX: *Local Memory Bus (LMB) V10 (v1.00a) - DS445*, 2009. – URL `http://www.xilinx.com/support/documentation/ip_documentation/lmb_v10.pdf`. – from companies webpage [May 2011]

[34] XILINX: *Spartan-3A DSP Starter Platform User Guide - UG454*, 2009. – URL http://www.xilinx.com/support/documentation/boards_and_kits/ug454_sp3a_dsp_start_ug.pdf. – from companies webpage [Dec 2011]

[35] XILINX: *XPS General Purpose Input/Output (GPIO) (v2.00.a) - DS569*, 2009. – URL http://www.xilinx.com/support/documentation/ip_documentation/xps_gpio.pdf. – from companies webpage [May 2011]

[36] XILINX: *XPS UART Lite (v1.01a) - DS571*, 2009. – URL http://www.xilinx.com/support/documentation/ip_documentation/xps_uartlite.pdf. – from companies webpage [May 2011]

[37] XILINX: *LogiCORE IP Fast Simplex Link (FSL) V20 Bus (v2.11c) - DS449*, 2010. – URL http://www.xilinx.com/support/documentation/ip_documentation/fsl_v20.pdf. – from companies webpage [Jan 2012]

[38] XILINX: *LogiCORE IP Mutex (v1.00a) - DS775*, 2010. – URL http://www.xilinx.com/support/documentation/ip_documentation/mutex.pdf. – from companies webpage [May 2011]

[39] XILINX: *LogiCORE IP Processor Local Bus(PLB)v4.6 (v1.05a) - DS531*, 2010. – URL http://www.xilinx.com/support/documentation/ip_documentation/plb_v46.pdf. – from companies webpage [May 2011]

[40] XILINX: *LogiCORE IP XPS Timer/Counter (v1.02a) - DS573*, 2010. – URL http://www.xilinx.com/support/documentation/ip_documentation/xps_timer.pdf. – from companies webpage [May 2011]

[41] XILINX: *MicroBlaze Debug Module (MDM) (v2.00.a) - DS641*, 2010. – URL http://www.xilinx.com/support/documentation/ip_documentation/mdm.pdf. – from companies webpage [May 2011]

[42] XILINX: *MicroBlaze Processor Reference Guide (v11.4) - UG081*, 2010. – URL http://www.xilinx.com/support/documentation/sw_manuals/xilinx12_4/mb_ref_guide.pdf. – from companies webpage [May 2011]

[43] XILINX: *Spartan-3A DSP FPGA Family Data Sheet - DS610*, 2010. – URL http://www.xilinx.com/support/documentation/data_sheets/ds610.pdf. – from companies webpage [Jan 2012]

[44] XILINX: *Xilkernel (v5.00.a) - UG708*, 2010. – URL http://www.xilinx.com/support/documentation/sw_manuals/xilinx12_4/oslib_rm.pdf. – from companies webpage [May 2011]

[45] XILINX: *IP Processor Block RAM (BRAM) Block (v1.00a) - DS444*, 2011. – URL http://www.xilinx.com/support/documentation/ip_documentation/bram_block.pdf. – from companies webpage [May 2011]

[46] XILINX: *Multi-Port Memory Controller(MPMC) (v6.03.a) - DS643*, 2011. – URL http://www.xilinx.com/support/documentation/ip_documentation/mpmc.pdf. – from companies webpage [May 2011]

[47] XILINX: *Product Brief - ZYNQ 7000 Extensible Processing Platform*, 2011. – URL http://www.xilinx.com/publications/prod_mktg/zynq7000/Product-Brief.pdf. – from companies webpage [June 2011]

[48] XILINX: *Spartan-3 Generation FPGA User Guide - UG331*, 2011. – URL http://www.xilinx.com/support/documentation/user_guides/ug331.pdf. – Section 1, Chapter 3: Using Digital Clock Managers

[49] XILINX: *Xilinx AXI Reference Guide - UG761*, 2011. – URL http://www.xilinx.com/support/documentation/ip_documentation/ug761_axi_reference_guide.pdf. – from companies webpage [Dec 2011]

[50] XILINX: *XPS 16550 UART (v3.00a) - DS577*, 2011. – URL http://www.xilinx.com/support/documentation/ip_documentation/xps_uart16550.pdf. – from companies webpage [Jan 2012]

[51] XILINX: *System Generator for DSP - Reference Guide - UG639*, 2012. – URL http://www.xilinx.com/support/documentation/sw_manuals/xilinx13_4/sysgen_ref.pdf. – from companies webpage [Jan 2012]

A FPGA-basierte Multiprozessor Technologie im Kontext eingebetteter Systeme

Eingebettete Systeme sind in ihrer Struktur und Implementierung auf eine klar definierte Aufgabenstellung abgestimmt, um ein bestmögliches Verhältnis von Systemleistung und Ressourcenverbrauch auf dem Zielsystem zu gewährleisten. Um wirtschaftliche Produkte anbieten zu können, müssen die Hersteller zusätzlich die Entwicklungs- und Herstellungskosten ohne Vernachlässigung der spezifizierten Anforderungen gering halten. Einen zentralen Ansatz für den Entwurf von eingebetteten Systemen stellt der Einsatz von Mikrocontrollern oder Mikroprozessoren dar, da diese nicht nur speziell auf diese Anforderungen abgestimmt werden können, sondern im Gegensatz zu General Purpose CPUs kostenkünstiger sind und einen geringeren Energiebedarf vorweisen.

Mit der Verfügbarkeit von Softcore Prozessoren für FPGA SoCs ergibt sich eine interessante Alternative zur klassischen Verwendung von Mikroprozessoren in eingebetteten Systemen. Die Rekonfigurierbarkeit der FPGAs lässt die Erweiterung bestehender Systementwürfe in der Entwicklungsphase oder für spätere Produktversionen mit zusätzlichen Hardwarekomponenten zu. Dazu werden bestehende oder eigens entwickelte IP-Cores (*Intellectual Properties*) in die Systeme aufgenommen, da diese über definierte Schnittstellen ansprechbar sind. Darüber hinaus können zusätzlichen Funktionalitäten durch weitere Softcore Prozessoren in bestehende Plattformen eingegliedert werden, ohne ein neues Platinenlayout entwerfen zu müssen. In diesem Kontext ist die Anordnung und Interaktion der Prozessoren von zentraler Bedeutung.

Homogene und Heterogene Prozessor Architekturen

Parallele Prozessorarchitekturen lassen sich in zwei Kategorien unterteilen. **Homogene Architekturen** bestehen aus baugleichen Prozessortypen, die den Ansatz des SMP (*Symmetric Multiprocessing*) verfolgen. Dabei wird eine Aufgabenstellung in einzelne Berechnungsschritte unterteilt und auf die zur Verfügung stehenden Recheneinheiten verteilt. In Bezug auf die Ausführungszeit ergibt sich im Vergleich zur sequentiellen Berechnung durch die

Parallelisierung ein Geschwindigkeitsvorteil, der auch als *SpeedUp* bezeichnet wird. Allerdings erfordert diese Vorgehensweise die Partitionierung der Operation in Teilprozesse und das Zusammenführen der Ergebnisse nach deren Berechnung. Dieser sequentielle Anteil ist nicht parallelisierbar und beschränkt den *SpeedUp* unabhängig von der verwendeten Anzahl an Prozessoren; diese Eigenschaft wird durch das Amdahlsche Gesetzt [2] beschrieben. Die Prozessoren in SMP-fähigen Systemen sind mit den gleichen Funktionalitäten ausgestattet und verfügen über eine identische Sicht auf das Gesamtsystem [24], d.h. sie verwenden einheitliche Schnittstellen, um auf Speicherbereiche oder Peripherie zugreifen zu können. Die homogenen Architekturen finden ihren Einsatz zunehmend in General Purpose CPUs, die durch die Bereitstellung mehrerer Berechnungseinheiten (*Cores*) die Berechnungszeit verkürzen und nebenläufige Operationen unterstützen. Durch die besonderen Anforderungen von SMP-Systemen in Bezug auf parallelisierbare Softwareanwendungen und dem Einsatz von SMP unterstützenden Betriebssystemen, ist deren Verbreitung auf FPGA-basierten Plattformen derzeit nicht weit vorangeschritten. Allerdings existieren generelle Studien mit Machbarkeitsanalysen und Betrachtungen zur Laufzeitoptimierung [15]. Zudem stellt Xilinx mit der ZYNQ-7000 EPP seit Anfang 2011 einen FPGA Baustein zur Implementierung von symmetrischen Multiprozessoranwendungen bereit [47].

Für das Anwendungsgebiet der eingebetteten System bietet sich die Verwendung von **heterogen organisierten Prozessorarchitekturen** an, da diese den charakteristischen, modularisierten Aufbau unterstützen. Das sogenannte AMP (*Asymmetric Multiprocessing*) beschreibt das Zusammenspiel eigenständiger Berechnungseinheiten, die einen gewissen Funktionsumfang unabhängig von einander bearbeiten. Die Synchronisation der Verarbeitungsergebnisse in den Kontext eines Gesamtsystems wird vorwiegend durch einen Masterprozessor übernommen. Dieser Ansatz bietet zudem den Vorteil, dass die Einzelprozesse eigenständig entwickelt und getestet werden können. Zur Integration in das Gesamtsystem müssen lediglich spezifizierte Schnittstellen und Kommunikationsverfahren eingehalten werden. Dadurch können auf den Prozessoren unterschiedliche Betriebssysteme und Softwareimplementierungen eingesetzt werden. Die Konfiguration des Prozessors kann somit gezielt auf die Spezifikation erfolgen[5].

Interprozessorale Kommunikation und Speicherpartitionierung in FPGA-basierten AMP-MPSoC

Die Kommunikationsverfahren die zur Synchronisierung und dem Datenaustausch zwischen Prozessoren in heterogen organisierten Architekturen eingesetzt werden, sind, abhängig von den Anforderungen des Zielsystems, durch unterschiedliche Ansätze realisiert. Dazu stehen in FPGA-basierten Systemen spezielle Bussysteme zur Verfügung, über die ein Nachrichtenaustausch realisiert werden kann. Diese Bussysteme stellen Punkt-zu-Punkt Verbindungen zwischen den Prozessoren her, wodurch während der Kommunikation keine Beeinflussung der übrigen Systemkomponenten auftritt. In der Xilinx MicroBlaze Umgebung wird

so ein Bussystem zum Beispiel durch den FSL (*Fast Simplex Link* - vgl. Anhang D) realisiert, der eine dirkete Kommunikationsleitung zwischen zwei Prozessoren herstellt. Über diesen können sowohl Steuersignale zur Synchronisation der Prozessoren als auch größere Datenmengen als Datenstrom (*streaming interface*) ausgetauscht werden.

Einen anderen Ansatz zum Austausch von Daten zwischen den Prozessoren stellt die Speicher-basierte Kommunikation dar. Dabei können die Prozessoren über ein Bussystem lesend und schreibend auf gemeinsame Speicherbereiche zugreifen. Dieser Ansatz eignet sich zum Beispiel für die sequentielle Bearbeitung von großen Datenmengen durch unterschiedliche Prozessoren. Die Daten sind an definierten Speicherstellen im Shared Memory abgelegt und müssen zwischen den Bearbeitungsschritten nicht über zusätzliche Protokolle und Bussysteme im System verteilt werden. Für den Zugriff auf diese Speicherstellen wird ein globaler *direct-mapped* Adressraum bereitgestellt und von allen Prozessoren verwendet. Die Zugriffe auf den kritischen Speicherbereich müssen allerdings synchronisiert werden, um gleichzeitig Datenmanipulationen zu verhindern. Diese Zugriffskontrollen werden in der Regel über Mutex-, Semaphore oder Monitorverfahren realisiert. Ein weiterer Nachteil liegt in der leistungsbegrenzenden physikalischen Anbindung des Speichers an das Gesamtsystem. Bei steigender Anzahl an Zugriffen auf den Speicher wird dieser, auch als „von-Neumann-Flaschenhals“ bekannter, Engpass zusätzlich belastet. Zur Reduktion der Speicherzugriffe werden die Prozessoren daher mit Caches implementiert, in denen Kopien des Hauptspeichers aus vorangegangenen Speicherzugriffen abgelegt werden. Ohne einen globalen Speicherbus passieren zu müssen, sind die Zugriffe auf Cachebereiche wesentlich schneller realisierbar. Allerdings führen *Cache Misses*, für Daten die noch nicht im Cache vorhanden sind und aus dem Shared Memory nachgeladen werden müssen, zu einem nicht deterministischem Zeitverhalten für Speicherzugriffe [5]. Darüber hinaus erfordert der Einsatz von Caches für gemeinsame Speicherbereiche in Multiprozessorsystemen die Synchronisation der Caches, um zu verhindern, dass Anfragen auf die gleiche Speicheradresse unterschiedliche Daten zurückliefern. Zur Einhaltung der Cache-Kohärenz werden die Zugriffe auf bestimmte Speicherstellen protokolliert, um den Caches die Manipulation von Daten aus den von ihnen vorgehaltenen Speicherbereichen zu signalisieren [24].

B Hardware-Elemente des SAV

Im folgenden Abschnitt werden die Hardware-Komponenten des SAV beschrieben. Der technische Aufbau gliedert sich in drei Bereiche; den Aktoren zur Fahrzeugbewegung, den Sensoren zur Erfassung von Umgebungsvariablen und dem Steuerungssystem für den autonomen Fahrbetrieb (vgl. Abb. B.1).

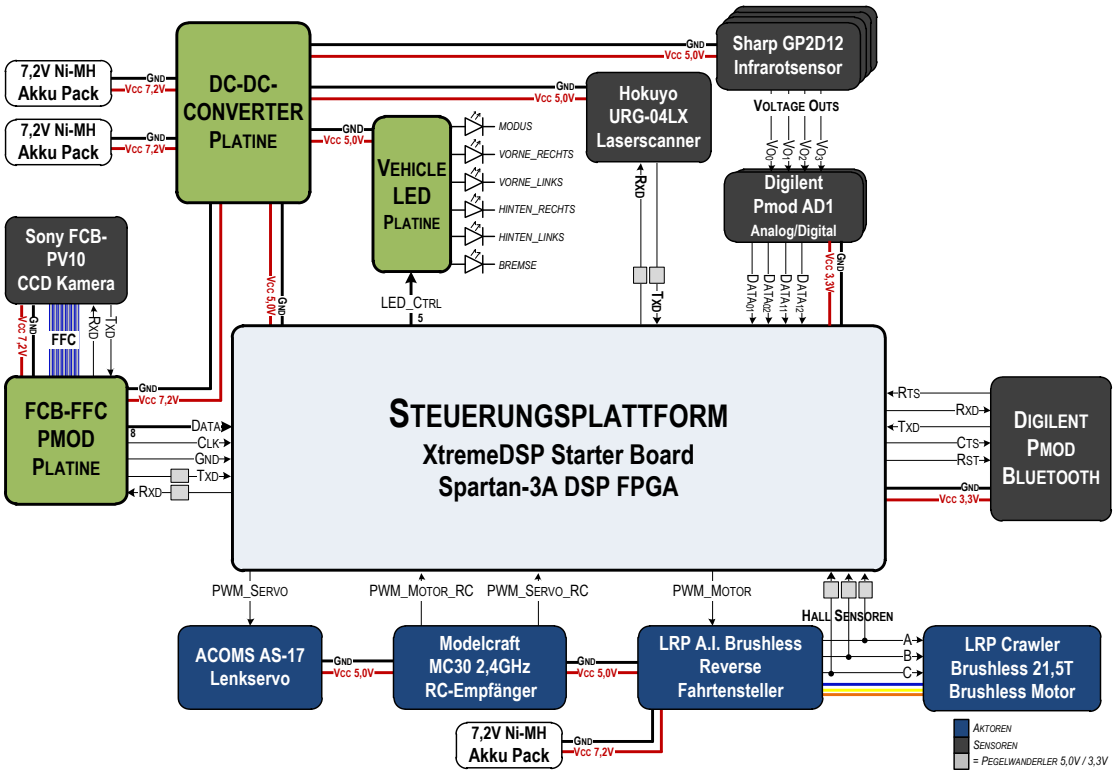


Abbildung B.1 – Übersicht zur Spannungsversorgung und den Schnittstellen zu Aktoren und Sensoren des SAVs.

Das Fahrzeug wurde gemäß den Vorgaben des aktuellen Carolo Cup Reglements [10] in einem Maßstab von 1:10 entworfen. Als Ausgangsplattform des SAV dient ein handelsüblicher RC-Modell Bausatz, das Tamiya TT-01 Chassis. Dieses enthält ein Differentialgetriebe, welches die Motorleistung über eine Antriebswelle auf alle vier Räder überträgt. Der manuelle Betrieb des Fahrzeugs erfolgt über die Modelcraft 3-Kanal MC30 Fernsteueranlage, die

neben einem Pistolen-Fernsteuersender auch einen Empfänger beinhaltet (vgl. Abb. B.2). Die Fahr- und Lenkfunktionen werden als PWM Signal unabhängig voneinander mit einer Frequenz von 2,4 GHz an den Empfänger übertragen, der diese an den Antriebsmotor und den Lenkservo weiterleitet. Der dritte Kanal stellt einen zusätzlichen Schaltkanal für Sonderfunktionen zur Verfügung; dieser wird im SAV zum Umschalten zwischen manuellem und autonomen Betrieb genutzt.

Abbildung B.2 – Lieferumfang der Fernsteueranlage MC30; der ursprünglich zur Kalibrierung und dem Feintuning der Fernsteuerung vorgesehene dritte PWM Kanal wird zum Wechsel der Betriebsart verwendet[18].

Der LRP A.I. Brushless Pro Reverse Digital Fahrtensteller verwendet die PWM Signale des Empfängers, um die Drehzahl des Antriebsmotors über die Versorgungsanschlüsse der drei Wicklungen einzustellen und diese an den Lenkservo zu übertragen. Die PWM Signale liegen mit einer Spannung von 3,3V in einem Tastverhältnis von 20 ms vor (vgl. Tab. B.1). Der Fahrtensteller versorgt zudem den RC-Empfänger und den Lenkservo mit der Versorgungsspannung von 5V.

Timing Spezifikation der PWM Signale		
Pulspegel	Vcc	3,3V
Periodendauer	T_{PWM}	20 ms
Tastverhältnis - Vorwärts	T_{max_vor}	10% = 2 ms
Tastverhältnis - Rückwärts	T_{max_rueck}	5% = 1 ms
Tastverhältnis - Stillstand	T_{still}	7,5% = 1,5 ms
die Werte sind ebenfalls für die Lenkwinkel PWM Signale gültig		

Tabelle B.1 – Kennwerte der PWM Signale; die Grenzen von 5% bis 10% für das Tastverhältnis dürfen nicht unter - bzw. überschritten werden, um Beschädigungen am Fahrtensteller zu vermeiden [31].

Die Stromversorgung des Fernsteuersenders erfolgt über acht 1,2V Ni-MH Mignon Akku Zellen. Die für den Carolo Cup vorgesehene Fahrzeugbeleuchtung wird über eine entwickelte LED Platine gesteuert. Dabei können die Vorder-, Hinter- und Blinker-LEDs durch eine fünfpolige Steuerungsleitung unterschiedlich angesteuert werden (vgl. Schaltbild im Anhang E).

Aktorik des SoC-Fahrzeugs

Die Aktoren des SAV sind an eine seperate Stromversorgung angeschlossen, um einen störungsfreien manuellen Steuerungsbetrieb zu gewährleisten. Somit kann die Steuerung des SAV unabhängig vom Zustand des Sensor-Regelungssystems manuell erfolgen. Ein 7,2V Ni-MH Akkupack im Rumpf des SAV ist an den Fahrtensteller angeschlossen, der den Antriebsmotor und den Lenkservo mit der Betriebsspannung von 5V versorgt.

Brushlessmotor und Fahrtensteller

Das SAV wird über den LRP Crawler Brushless 21,5T mit drei Wicklungen angetrieben. Dieser bürstenlose Elektromotor wurde speziell für präzise Regelbarkeit bei niedrigen Geschwindigkeiten entwickelt und besitzt eine Maximalgeschwindigkeit von 5 m/s. Im Gegensatz zu herkömmlichen Gleichstrommotoren wird die Drehbewegung eines bürstenlosen Gleichstrommotors nicht über direkte Kontakte (Bürsten) zwischen dem Stator und dem Rotor erzeugt. Die Bewegung ensteht durch einen Permanentmagneten im Rotor, der über elektronisch angesteuerte Spulen des Stators zur Rotation gebracht wird. Der Vorteil dieser Technik liegt in einem geringeren Verschleiß, da in der Drehbewegung kein Kontakt zwischen Rotor und Stator besteht; außerdem entstehen auf diese Weise keine Funken an den Bürsten, die hochfrequente Störungen für das Gesamtsystem erzeugen können. Der LRP A.I. Brushless Pro Reverse Digital Fahrtensteller verwendet das PWM Signal des Empfängers, um die Drehzahl des Antriebsmotors über die Versorgungsanschlüsse der drei Wicklungen einzustellen. Zudem ist der Fahrtensteller für die Bereitstellung der Versorgungsspannung des Empfängers und des Lenkservos von 5V zuständig.

Hall Sensoren

Die tatsächliche Drehbewegung des Rotors erhält der Fahrtensteller über die drei im Motor integrierten Hall Sensoren, wodurch er diese gegebenenfalls korrigieren kann. Die Perioden der Hall Sensoren erfassen eine vollständige Rotation, wodurch sich je nach Dauer der Perioden die Fahrgeschwindigkeit ableiten lässt. Ferner kann durch die Kombination der phasenverschobenen Hall Sensor Werte die Fahrtrichtung identifiziert werden. Die Hall Sensor

Abbildung B.3 – Der Crawler Brushless Antriebsmotor mit dem A.I. Brushless Pro Reverse Digital für präzise Regelung bei geringen Geschwindigkeiten [18].

Werte werden zusätzlich für die autonome Geschwindigkeitsregelung an das Steuerungssystem übertragen. Um die zulässige Eingangsspannung des Steuerungssystems nicht zu überschreiten, werden die Hall Sensor Signale durch einen Texas Instruments SN74LVC245A Pegelwandler von 5V auf 3,3V umgesetzt.

Lenkwinkelsteller

Der Lenkwinkel des SAV wird über den ACOM AS-17 Standardservo eingestellt. Dieser ist direkt mit dem Empfänger verbunden und bezieht von diesem neben dem PWM Signal für die Winkelstellung auch die Versorgungsspannung von 5V. Der maximale Lenkwinkel des AS-17 beträgt $\pm 20^{\circ}$.

Sensorik des SoC-Fahrzeugs

Die Sensoren werden, wie auch die Steuerungsplattform, über eine zweite Spannungsversorgung gespeist. Dazu steht ein zweites Akkufach zwischen Trägerplattform und Fahrwerk bereit, in dem zwei weitere 7,2V Ni-MH Akkupacks Platz finden. Durch eine DC-DC-Converter Platine werden diese in Reihe geschaltet und die resultierende Eingangsspannung von 14,4V durch einen Spannungswandler auf vier 5V Anschlußstecker umgesetzt. Zusätzlich verfügt die Platine über zwei Anschlußstecker, über die 7,2V eines angeschlossenen Akkupacks bezogen werden können. Das Schaltbild der DC-DC-Converter Platine ist im Anhang E angegeben.

Sony FCB-PV10 CCD Kamera

Die FCB-PV10 Kamera von Sony ist mit einem 1/4 Type Interline-Transfer CCD Sensor ausgestattet und liefert digitale Bilddaten über eine FFC (*flexible flat cable*) Schnittstelle mit einer Bildrate von 29,97 Bildern pro Sekunde. Die FCB-PV10 stellt eine VGA Auflösung mit 640x480 Pixeln pro Bild bereit und kann auf verschiedene Betriebsmodi konfiguriert werden. Weiterhin enthält sie einen integrierten Quarz, der mit einer Taktrate von 27 MHz die Bilddaten bereitstellt. Für die Fahrspurerkennung des SAV wurde die Kamera auf den 8-Bit Interlace Betriebsmodus eingestellt, in dem jeweils 8-Bit Pixelinformationen nach dem ITU-Rec. BT.656 Standard im YCbCr 4:2:2 Format übertragen werden [30]. Diese Konfiguration ist notwendig, da die Bilddaten zur Unterstützung in der Entwicklungs- und Testphase zusätzlich auf einem Monitor dargestellt werden sollen. Das YCbCr 4:2:2 hingegen erfordert innerhalb der Bildverarbeitungspipeline die Interpolation der Chrominanzwerte, um ein darstellbares RGB 4:4:4 Format zu erzeugen. Durch diesen Vorgang wird die Taktrate innerhalb der Verarbeitungskette auf 13,5 MHz halbiert; welche allerdings für die Ausgabe auf standardisierten VGA Monitoren zu niedrig ist. Aus diesem Grund wurde der Interlace Modus gewählt, in dem nicht die gesamten 480 Zeilen, sondern alternierend die geraden bzw. ungeraden Zeilen des Bildes übertragen werden. Dadurch kann am Ende der Bildverarbeitungskette durch eine einfache Zeilenverdopplung die ursprüngliche Taktrate von 27 MHz wieder erreicht werden. Dieser Vorgang hat für die Fahrspurerkennung keinerlei, für die Betrachtung durch das menschliche Auge nur marginale Nachteile [17]. Die Konfiguration

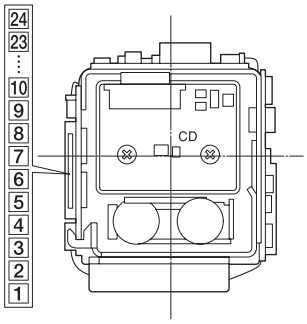
Technische Daten	
Auflösung	640 x480
Bildrate	29,97 fps
Blickwinkel	46°
Optischer Zoom	10fach
Video Output	YCrCb 4:2:2 / 8-Bit
Arbeitstakt	27,0 MHz / intern

Abbildung B.4 – Die Sony FCB-PV10 CCD Kamera zur Erfassung der Fahrspurmarkierung liefert zusätzlich die Eingangsfrequenz für den CLOCK MANAGER des SAV CONTROL IP-Cores [29].

der Kamera erfolgt über eine RS232 UART Schnittstelle, über die neben den Betriebsmodi auch Zoom, Fokussierung, Belichtungsautomatik und Blendeneinstellung zur Laufzeit angepasst werden können. Die Kamera arbeitet mit einer Versorgungsspannung zwischen 6V bis 12V und wird von der DC-DC-Converter Platine mit 7,2V versorgt. Für den Anschluss der FCB-PV10 an das Steuerungssystem wurde eine Kamera Platine entwickelt, um die Eingangssignale des FFC auf kompatible Digilent Pmod Anschlüsse umzusetzen (vgl. Schaltbild im Anhang E). Abbildung B.5 zeigt eine Übersicht der Signale der FCB-PV10.

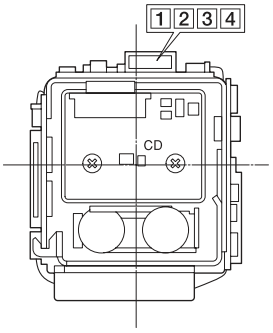
Die Platine ist zusammen mit der Kamera auf einem 25 cm hohen Träger im rechten Vorderbereich der Fahrzeugplattform angebracht. Höhe und Neigungswinkel der Kamera sind verstellbar, um einen optimalen Sichtbereich ermitteln zu können.

CN501
24P FFC (0.5mm)
KYOCERA ELCO Co. 046240024006848+



Pin No.	Name	16 bit data bus	Level	8 bit data bus	Level
1	GND	Signal Ground			
2	Y0	Digital Y-Out 0	0 - 3.2 Vp-p	Digital Out 0	0 - 3.2 Vp-p
3	Y1	Digital Y-Out 1	0 - 3.2 Vp-p	Digital Out 1	0 - 3.2 Vp-p
4	Y2	Digital Y-Out 2	0 - 3.2 Vp-p	Digital Out 2	0 - 3.2 Vp-p
5	Y3	Digital Y-Out 3	0 - 3.2 Vp-p	Digital Out 3	0 - 3.2 Vp-p
6	Y4	Digital Y-Out 4	0 - 3.2 Vp-p	Digital Out 4	0 - 3.2 Vp-p
7	Y5	Digital Y-Out 5	0 - 3.2 Vp-p	Digital Out 5	0 - 3.2 Vp-p
8	Y6	Digital Y-Out 6	0 - 3.2 Vp-p	Digital Out 6	0 - 3.2 Vp-p
9	Y7	Digital Y-Out 7	0 - 3.2 Vp-p	Digital Out 7	0 - 3.2 Vp-p
10	GND	Signal Ground			
11	C0	Digital C-Out 0	0 - 3.2 Vp-p	Hi imp	
12	C1	Digital C-Out 1	0 - 3.2 Vp-p	Hi imp	
13	C2	Digital C-Out 2	0 - 3.2 Vp-p	Hi imp	
14	C3	Digital C-Out 3	0 - 3.2 Vp-p	Hi imp	
15	C4	Digital C-Out 4	0 - 3.2 Vp-p	Hi imp	
16	C5	Digital C-Out 5	0 - 3.2 Vp-p	Hi imp	
17	C6	Digital C-Out 6	0 - 3.2 Vp-p	Hi imp	
18	C7	Digital C-Out 7	0 - 3.2 Vp-p	Hi imp	
19	GND	Signal Ground			
20	VSYNC	Vertical SYNC	0 - 3.2 Vp-p		0 - 3.2 Vp-p
21	HSYNC	Horizontal SYNC	0 - 3.2 Vp-p		0 - 3.2 Vp-p
22	GND	Signal Ground			
23	CLOCK	Clock signal	0 - 3.2 Vp-p		0 - 3.2 Vp-p
24	GND	Signal Ground			

CN701
4P connector
J.S.T. Mfg Co. S4B-ZR-SM4A-TF(LF)



Pin No.	Name		Level
1	UNREG	Power Input	6 - 12 V (dc)
2	GND	Ground	
3	TD		TTL Level (0 - 5.0 Vp-p)
4	RD		TTL Level (0 - 5.0 Vp-p)

Abbildung B.5 – Pinbelegungen des Kamera FFC CN501 Anschlusses und des CN701 für die Spannungsversorgung und die serielle Schnittstelle, die direkt mit der FFC-Pmod Adapter Platine verbunden werden.

Hokuyo URG-04LX Laserscanner

Der URG-04LX ist ein einstrahliger 2D-Laserscanner mit einem Wirkungsradius von 240°. Die Wellenlänge der Laserdiode beträgt 785 nm und ist vergleichbar mit Lasern, die in Lichtschranken oder Laserdruckern eingesetzt werden. Die maximale Sichtweite, um zuverlässig Abstände zu ermitteln, liegt bei 4 Metern, was die Dimensionen des SAV-Einsatzgebietes abdeckt. Als Basis für die Abstandsermittlung wird der reflektierte Laserstrahl durch eine Photodiode empfangen und die Phasenverschiebung mit einer Auflösung von 1 mm ermittelt. Die detektierten Entfernungen im Sichtfeld werden mit einer Frequenz von 10 Hz im Form von Polarkoordinaten über eine RS232 UART Schnittstelle an das Steuerungssystem übertragen, um dort in Segmente und Objekte verarbeitet zu werden. Die Stromversorgung des Laserscanners erfolgt über die DC-DC-Converter Platine mit 5V. Die Kommunikationssignale der RS232 Verbindung werden über einen Pericom PI74LVC3245A bidirektionalen Pegelwandler zwischen dem Laserscanner und der Steuerungsplattform von 3,3V auf 5V und v.v. transformiert. Eine detaillierte Übersicht zur Leistung und Konfiguration des URG-04LX ist in [32] zu finden. Damit der Laserscanner den größtmöglichen Sichtbereich nutzen kann, ohne dabei von anderen Fahrzeugteilen behindert zu werden, ist dieser vor der Frontachse am Fahrzeug angebracht.

Technische Daten	
Reichweite	60 - 4.095 mm
Scanfrequenz	10 Hz
Scanwinkel	240°
Anzahl Scanschritte	682
Auflösung	1 mm
Interface	RS232 & USB

Abbildung B.6 – Zur Detektion von Hindernissen auf der Fahrbahn wird der Hokuyo URG-04LX Laserscanner eingesetzt [14].

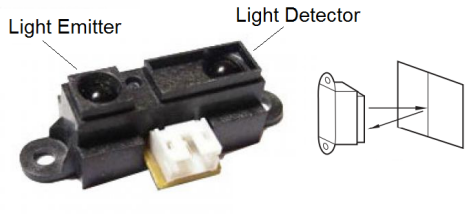
Digilent Pmod Bluetooth Modul

Das PmodBT verbindet einen seriellen National Semiconductor LMX9838 Bluetooth Baustein mit den standardisierten Pmod Schnittstellen der Firma Digilent. Das Bluetooth Modul wird über ein einfaches serielles Interface angesprochen, das zur Datenübertragung und Konfiguration dient. Es stehen zwei Betriebsarten zur Verfügung; im *Transparent Mode* wird die Kommunikation einer seriellen Verbindung durch den LMX9838 zu einem einzigen Bluetooth Zielgerät transformiert. Der *Command Mode* bietet dagegen die Möglichkeit die GAP (*Generic Access Profile*), SDAP (*Service Discovery Application Profile*), SPP (*Serial Port Profile*) Übertragungsprofile des Bluetooth 2.0 Stacks zu verwenden. Dadurch können

Anwendungen entwickelt werden, die mehrere Verbindungen enthalten oder an standardisierte Übermittlungsverfahren anknüpfen. Im Kontext des Einsatzes im SAV wird die Bluetooth Verbindung lediglich zum Austausch von Steuersignalen mit einem Host-PC oder der Datenaufzeichnung bei Erprobungsfahrten eingesetzt; was auch bei Verwendung des transparenten Modus realisiert werden kann. Das PmodBT wird direkt vom Entwicklungsboard mit der Betriebsspannung von 3,3V versorgt.

Sharp GP2D12 Infrarot-Sensor

Für den Anwendungsfall des automatischen Einparkens, ist es erforderlich, dass auch die Bereiche neben dem Fahrzeug auf Objekte überprüft werden. Zur seitlichen Abstandserkennung werden vier Infrarot-Sensoren vom Typ Sharp GP2D12, mit einem Einsatzbereich zwischen 10 cm und 80 cm, eingesetzt.



Technische Daten	
Reichweite	100 - 800 mm
Ausgangsspannung	0,25 - 2,6 V
Versorgungsspannung	4,5 - 5,5 V
Typische Antwortzeit	39 ms
Stromverbrauch	33 mA

Abbildung B.7 – Die seitlichen Bereiche des SAV werden für ein automatisches Einparkszenario mit vier Sharp GP2D12 Infrarot-Sensoren überwacht [27].

Die Ermittlung des Abstandes erfolgt durch einen gerichteten Infrarotlichtstrahl, dessen Reflexion von einer Photodiode empfangen wird. Durch ein Triangulationsverfahren kann so der Abstand ermittelt werden, der durch eine analoge Spannung zwischen 0,4V und 2,6V die Detektion eines Objektes anzeigt. Niedrigere Ausgangsspannungen stellen hierbei eine größere Entfernung dar. Da die analogen Werte nicht linear sind, muss eine konkrete Kalibrierung jedes einzelnen Sensors erfolgen, um zuverlässige Abstände zu ermitteln. Die GP2D12 Sensoren werden mit einer Betriebsspannung von 5V über die DC-DC-Converter Platine versorgt.

Komponenten der Steuerungsplattform

Die Steuerungsplattform basiert auf einem Entwicklungsboard, das ein Spartan-3A DSP FPGA bereithält, auf dem die Implementierung des Steuerungssystems erfolgt. Gleichzeitig stellt das Entwicklungsboard Peripherieschnittstellen zur Verfügung, mit denen die Kommunikation zu den Sensoren und dem Entwicklungsrechner realisiert wird.

Xilinx Spartan-3A DSP Starter Board

Als Grundlage für das SAV Steuerungssystem dient die Xilinx Spartan-3A DSP Starter Plattform. Neben einem Spartan 3SD1800A-FG676 FPGA ist dieses Entwicklungsboard mit diversen physikalischen Schnittstellen und Peripherie-Komponenten ausgestattet, die den Hardware- und Softwareentwicklern zahlreiche Varianten für den Entwicklungsprozess von FPGA-basierten Prototypsystemen bereithalten (vgl. Abb. B.8).

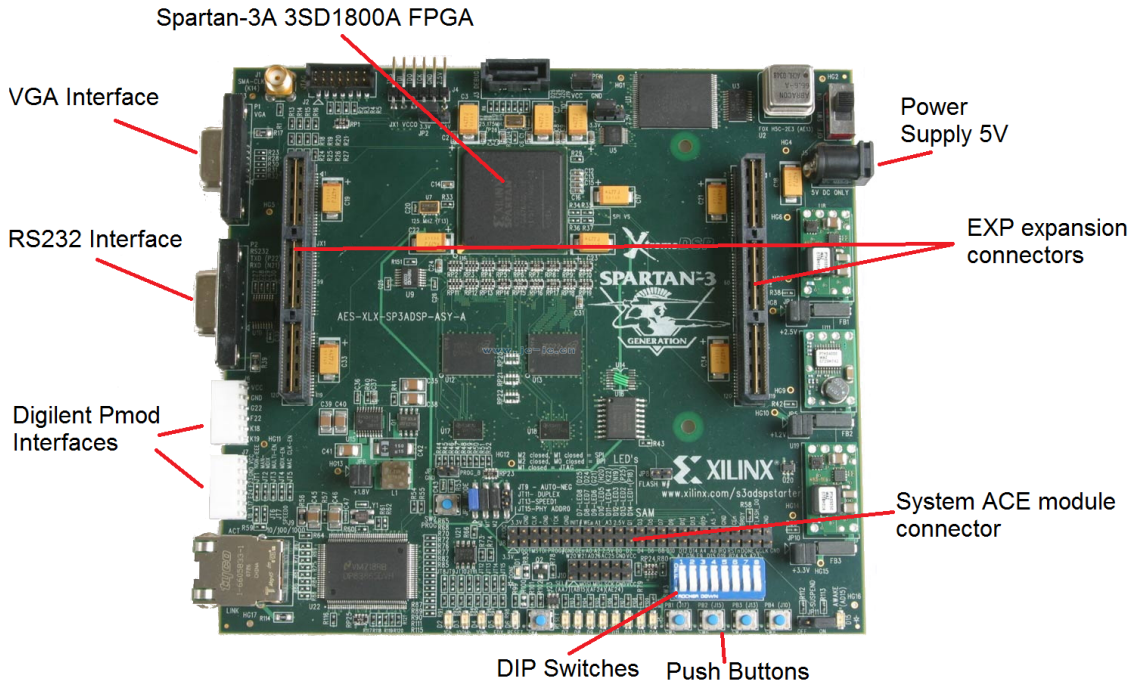


Abbildung B.8 – Das Xilinx Spartan-3A DSP Starter Board als Grundlage der Steuerungsplattform für das SAV.

In der nachfolgend Übersicht sind die zentralen Komponenten des Entwicklungsboards zusammengefasst [34]:

- 125 MHz LVTTL SMT Oszillator

- 128 MB Micron DDR2 SDRAM
- 128 MBit Intel Parallel BPI Flash und 64 MBit SPI Flash
- 2 EXP Expansion Connectors mit jeweils 84 I/O Pins
- Ethernet PHY, RS232 Port, VGA, JTAG, SystemACE Modul
- 8 LEDs, 8 DIP Switches, 4 Push Buttons
- 2 Digilent 6-pin Header (Pmod)

Die DSP Starter Plattform erhält die Betriebsspannung von 5V über die DC-DC-Converter Platine; zudem enthält sie vier Regulatoren, um ihrerseits Versorgungsspannungen von 3,3V, 2,5V, 1,8V und 1,2V bereit zu stellen. Die Integration der Kameraschnittstelle erfolgt über das SAM (*SystemACE Modul*). Dieses stellt ein 50 Pin umfassendes Interface dar, das ursprünglich zum Anschluß einer Compact Flash Speicherkarte vorgesehen ist. Im SAV ist eine solche Schnittstelle nicht geplant, daher werden über die Datenleitungen `SAM_D0-SAM_D7` die Kamerabilddaten aufgenommen. Gleichzeitig wird der von der Kamera bereitgestellte Bilddatentakt über den Clock-Eingangspin `SAM_CLK` aufgenommen. Das Bluetooth Modul wird über die beiden Digilent Pmod Schnittstellen direkt an das Board angeschlossen, wobei dieses zugleich die erforderliche Betriebsspannung von 3,3V für den PmodBT Baustein liefert. Die Belegung der 168 EXP Connector Pins erfolgt über Dual-Slot Adapterplatine AVNET EXP Bus Prototyping Board, die auf die beiden EXP Anschlüsse des Entwicklungsboards gesteckt wird. An diese sind sowohl die PWM Ein- und Ausgangssignale für die Motorensteuerung als auch die UART Schnittstellen der Kamera, des Laserscanners und die Ausgänge der Hall Sensoren angeschlossen. Die Signale der Letzteren werden dabei über die oben beschriebenen Pegelwandlermodule von 5V auf 3,3V umgesetzt. Für die Entwicklungsphase wird zusätzlich ein Monitor an den Standard VGA Anschluß der Starter Plattform sowie eine serielle Verbindung über die integrierte RS232 Schnittstelle angeschlossen. Über diese werden einerseits die Konfigurationsdatei und die Software auf das FPGA übertragen; zweitens dient diese Schnittstelle zur Kommunikation der Anwendungssoftware mit einem Entwicklungsrechner. Zur Steuerung und Konfiguration der Softwareanwendung werden die 4 Pushbuttons und die DIP Switches der Starter Plattform verwendet. Eine detailierte Übersicht der Pinbelegungen befindet sich im Anhang E.

Spartan 3SD1800A-FG676 FPGA

Die DSP Variante des Xilinx Spartan-3A FPGAs ist mit zusätzlichen 250MHz XtremeDSP DSP48A Slices und erweitertem Speicher pro Logikzelle ausgestattet, um diesen Baustein für den Einsatz in DSP-basierten Systemen zu spezialisieren. Dazu enthalten die DSP48A Slices 18-Bit x 18-Bit Multiplizierer mit 48-Bit Akkumulatoren sowie einem integrierten Addierer, um performante Multiplikationsoperationen umsetzten zu können. Nachfolgend sind die entscheidenden Attribute des Spartan-3A DSP aufgelistet [43]:

- Gesamtanzahl von 16.640 Slices aufgetielt in 4.610 CLBs
- 84 DSP48A Slices mit je einem 18-Bit x 18-Bit Multiplizierer, einem 18-Bit Pre-Adder und einem 48-Bit Akkumulator für Pipelineoperation mit einer Geschwindigkeit von mindestens 250 MHz
- 84 18-kBit Dual-Port BRAM Blöcke (1.512 kBit On-Chip BRAM)
- 8 Digitalen Clock Managern (DCM) zur Signalverarbeitung in unterschiedlichen Frequenz-Domänen
- 519 User I/O Pins für die Integration externer Peripherie

C Die Geschwindigkeitsregelung des SAV

Die Fahrzeugführung des SAV im automen Betriebsmodus basiert im ersten Ansatz auf festgelegten Soll-Geschwindigkeiten; das Durchfahren der Erprobungsstrecken erfolgt mit konstanter Geschwindigkeit. Durch unterschiedliche Lenkbewegungen in Kurvenfahrten und die ständige Ausrichtung an der Fahrspurmarkierung entstehen unterschiedliche Reibungsverhältnisse und Kontaktflächen zwischen Rädern und Fahrbahn. Um die daraus resultierenden Unregelmäßigkeiten auszugleichen, wird eine Geschwindigkeitsregelung eingesetzt. Darüber hinaus ist in einem späteren Entwicklungsschritt die adaptive Anpassung der Geschwindigkeit an die Streckenverhältnisse denkbar.

Regelungstechnischer Entwurf des Geschwindigkeitsreglers

Die im SAV eingesetzte Geschwindigkeitsregelung basiert auf einem PI-Regler[1]. Die Kombination mit einem D-Anteil wurde vermieden, da die daraus resultierende schnellere Regelung wesentlich höhere Anlaufströme des Motors hervorruft. Bei stark variierenden Regelabweichungen führt das zu einem erhöhten Energiebedarf und der Verkürzung der Betriebszeit des SAV. Zusätzlich führen die höheren Anlaufströme zu einer größeren Kraftumsetzung, was bei erhöhter Beschleunigung zu Übersteuerungen der Räder und Kontaktverlust mit der Fahrbahn führen kann.

Zur Festlegung der Regelungsparameter ist eine Anaylse der Fahrzeugregelstrecke durch die Sprungantwortmethode durchgeführt worden [31]. Die Funktion des Übergangsverhaltens für einen Regler mit proportionalem und integralem Anteil ist mit folgender Gleichung für die zeitkontinuierliche Darstellung beschrieben:

$$u(t) = K_P \cdot \left(e(t) + \frac{1}{T_N} \cdot \int e(\tau) d\tau \right) \tag{50}$$

[1] Zum Aufbau und der allgemeinen Wirkungsweise von Regelkreisen siehe 3.2.4.

Demnach lautet die Übertragungsfunktion des PI-Reglers:

$$G_R(s) = \frac{u(s)}{e(s)} = K_P \cdot \left(1 + \frac{1}{s \cdot T_N}\right) \tag{51}$$

Durch Veränderung des Proportionalbeiwertes K_P oder der Nachstellzeit T_N kann Einfluss auf die Ermittlung der Stellgröße genommen werden. Zur Diskretisierung von Gleichung 51 wird auch hier die z-Transformation angewandt. Dabei wird die Transformation von der s- in die zeitdiskrete z-Domäne über die Tustin Formel vorgenommen. Die Tustin Formel ist beschrieben durch [19]:

$$s = \frac{2(z-1)}{T_a(z+1)} \tag{52}$$

Daraus ergibt sich die diskretisierte Übertragungsfunktion des PI-Reglers:

$$G_R(z) = \frac{u(z)}{e(z)} = K_P \cdot \left(1 + \frac{1}{\frac{2(z-1)}{T_a(z+1)} \cdot T_N}\right) = K_P \cdot \left(1 + \frac{T_a(z+1)}{2T_N(z-1)}\right) \tag{53}$$

Die Modellierung der Regler Funktion in einem Simulink Funktionsblock setzt eine weitere Umformung der diskretisierten Übertragungsfunktion voraus. Dabei werden die Konfigurationsparameter T_N und K_P zusammen mit der Abtastrate T_a durch Konstanten in der Form „$bz+a$“ beschrieben:

$$G_R(z) = \frac{u(z)}{e(z)} = \frac{2T_NK_P(z-1)+T_aK_P(z+1)}{2T_N(z-1)} = \frac{\overbrace{(2T_NK_P+T_aK_P)}^{b_0}z - \overbrace{(2T_NK_P-T_aK_P)}^{a_0}}{\underbrace{2T_N}_{b_1}z - \underbrace{2T_N}_{a_1}} \tag{54}$$

Zur Übertragung des Reglers in eine digitale Systemumgebung, wird eine zeitliche Zustandsvariable z^-1 in Gleichung 54 eingefügt, die einen Zustandsspeicher darstellt und somit den Zusammenhang zu vorangegangenen Messungen herstellt.

$$G_R(z) = \frac{u(z)}{e(z)} = \frac{b_0z-a_0)z^-1}{(b_1z-a_1)z^-1} = \frac{b_0-a_0z^-1}{b_1-a_1z^-1} \tag{55}$$

Die Berechnung einer neuen Ausgangsgröße $u(k)$ erfordert zunächst die Freistellung von $u(z)$ aus Gleichung 55. Da der Nenner keine Bezug zur Stellgröße $u(z)$ hat, entfällt er in der weiteren Betrachtung (vgl. Gl. 57).

$$\frac{u(z)b_1-a_1z^-1)-e(z)(b_0-a_0z^-1)}{e(z)(b_1-a_1z^-1)} = 0 \tag{56}$$

$$u(z)b_1 - u(z)a_1z^-1 = e(z)b_0e(z) - a_0z^-1 \tag{57}$$

Die Abbildung der Veränderung von konkreten Zustandswerte für die Stellgröße u und die Regelabweichung e wird über die Transitionen $x(z)z^{-}1 \rightarrow x(k-1)$ vorgenommen.

$$b_1 u(k) - a_1 u(k-1) = b_0 e(k) - a_0 e(k-1) \tag{58}$$

Die Stellgröße zum Zeitpunkt k wird somit in den Zusammenhang mit der aktuellen und vorangegangenen Regelabweichung, $e(k)$ und $e(k-1)$, sowie der vorangegangenen Stellgröße $u(k-1)$ gebracht.

$$u(k) = \frac{b_0}{a_0} e(k) - \left(\frac{a_0}{b_0} e(k-1) - \frac{a_1}{b_1} u(k-1) \right) \tag{59}$$

Durch Resubstitution ergibt sich die Gleichung zur Berechnung der neuen Stellgröße. Die Parameter p_1 und p_2 sind dabei Konstanten, die abhängig vom Verhalten des Regelstrecke und den umgebenden Einflüssen bestimmt werden.

$$u(k) = \frac{\overbrace{2T_N K_P + T_a K_P}^{p_1}}{2T_N} e(k) - \left(\frac{\overbrace{2T_N K_P - T_a K_P}^{p_2}}{2T_N} e(k-1) - u(k-1) \right) \tag{60}$$

$$u(k) = p_1 \cdot e(k) - (p_2 \cdot e(k-1) - u(k-1)) \tag{61}$$

Aufgrund der Ergebnisses der Sprungantwortanalyse wurden zwei Verfahren zur Bestimmung von p_1 und p_2 betrachtet; das Wendetangenten Verfahren sowie die Anwendung der T-Summen-Regel nach Kuhn. Zur Analyse dieser Verfahren wurde ein Matlab Simulink Modul entworfen und unterschiedliche Parametrierungsempfehlungen untersucht. Als Ergebnis dieser Analyse wird im weiteren Verlauf die T-Summen-Regel beschrieben, da deren simulierte Sprungantwort weniger Überschwingungen und allgemein eine größere Übereinstimmung mit der expirementell-ermittelten Sprungantwort des Fahrzeugregelkreises ergab [31].

T-Summen-Regel nach Kuhn zur PI-Regler-Parametrierung

Die T-Summen-Regel ist eine heuritische Einstellregel, nach der zwei Streckenkennwerte genügen, um die Parametrierung eines PI-Reglers für Regelstrecken, die eine S-förmige Sprungantwort aufweisen, vorzunehmen; den Proportionalbeiwert oder Verstärkungsfaktor K_s und die Summenzeitkonstante T_Σ. Der Verstärkungsfaktor ergibt sich aus dem angestrebten Soll-Wert. T_Σ setzt sich aus der Streckentotzeit T_t und allen Zeitkonstanten mit regelungstechnisch verzögernden Eigenschaften T_1 bis T_n, abzüglich der differenzierenden Zeitkonstanten T_{D1} bis T_{Dm}, zusammen. Die Übertragungsfunktion für Regelstrecken mit

S-förmiger Sprungantwort wird beschrieben durch

$$G(s) = K_s \frac{(1+T_{D1}s)(1+T_{D2}s)...(1+T_{Dm}s)}{(1+T_1s)(1+T_2s)...(1+T_ns)} e^{-sT_t} \tag{62}$$

Daraus ergibt sich die Summe der verzögernden Zeitkonstanten als

$$T_\Sigma = T_1 + T_2 + ... + T_n - T_{D1} - T_{D2} - ... - T_{Dm} + T_t \tag{63}$$

Die Konstante kann auch geometrisch mit der Sprungantwort konstruiert werden, indem eine vertikale Gerade in Abbildung C.1 so positioniert wird, dass die Flächen F_1 und F_2 gleich groß werden. Der Schnittpunkt der Geraden mit der Zeitachse t bezeichnet die Summenzeitkonstante T_Σ.

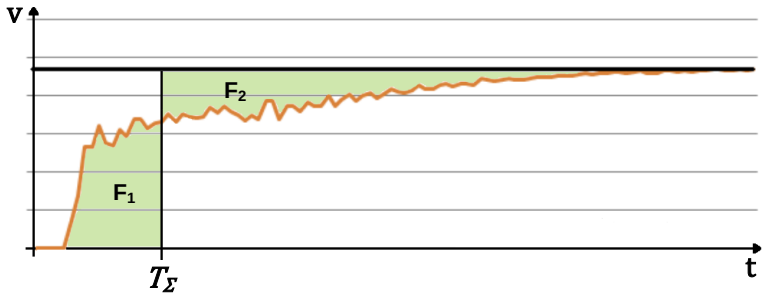


Abbildung C.1 – Geometrische Konstruktion der T-Summen-Regel nach Kuhn.

Für die Reglerparametrierung gilt nach Kuhn folgende Empfehlung:

$$K_P = 0,5/K_s \tag{64}$$

$$T_N = 0,5 \cdot T_\Sigma \tag{65}$$

Diese Empfehlungen werden in Gleichung 60 zur Berechnung der Stellgröße übernommen, um die Parameter p_1 und p_2 zu bestimmen. Die Sprungantworten der Regelstrecke besitzen bei unterschiedlichen Geschwindigkeiten verschiedene Werte für K_s und T_Σ, sodass die Bestimmung der Parameter für ein optimales Regelverhalten adaptiv an die aktuelle Geschwindigkeit angepasst werden muss.

RTL-Modellierung der Geschwindigkeitsregelung

Der V-CONTROLLER übernimmt die Regelung der Fahrzeuggeschwindigkeit auf eine vorgegebene Soll-Wert und ermittelt dazu die bestehende Ist-Geschwindigkeit. Zudem wird als Ausgangssignal des Moduls eine PWM zur Ansteuerung des Motors erzeugt. Die Ermittlung des Ist-Wertes erfolgt über die Auswertung von Hall Sensor Signalen, die direkt vom Crawler Brushless Motor bezogen werden (vlg. Abb. C.2). Ein PI-Regler wird verwendet, um mit der ermittelten Geschwindigkeit und der vorgegebenen Stellgröße `v_ctrl` die Geschwindigkeit des Fahrzeugs zu regeln. Die Systemfrequenz von 54 MHz wird vom CLOCK MANAGER bezogen.

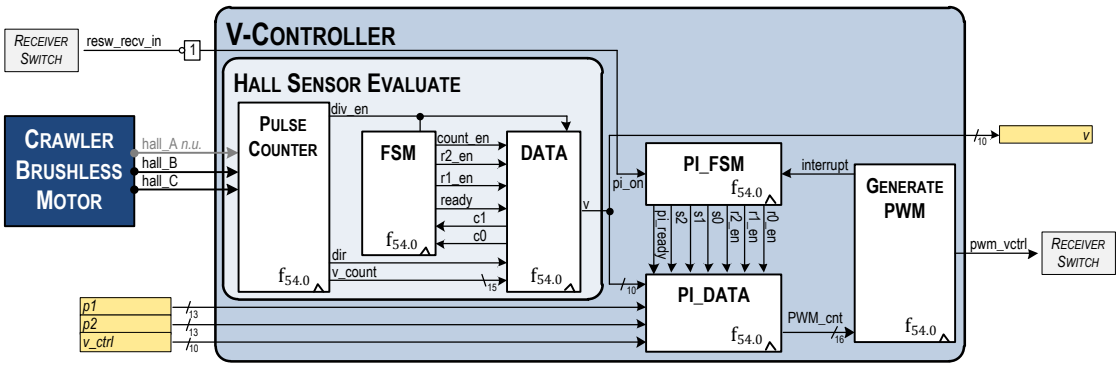


Abbildung C.2 – Das V-CONTROLLER Modul verfügt zusätzlich über Eingangsvektoren für die Parameter der PI-Regelung, wodurch eine Anpassung des Regelkreises zur Laufzeit vorgenommen werden kann.

Bestimmung der Ist-Geschwindigkeit durch Hall Sensor Daten

Im Gegensatz zur Drehgeber-basierten Geschwindigkeitsmessung, bei der die Messung direkt an der Radachse vorgenommen wird, sind die Hall Sensoren in regelmäßigen Abständen direkt um die Motorachse platziert. Gleichzeitig ist an der Achse ein Magnet angebracht, auf den die Sensoren durch elektrische Rechteckimpulse induktiv reagieren. Über drei Signalleitungen werden die Impulse an das V-CONTROLLER Modul übertragen, in dem über die Reihenfolge der Flankenübergänge die Drehrichtung des Motors bestimmt werden kann; eine vollständige Achsumdrehung ist somit als feste Folge von Pegelflankenwechseln beschrieben (vgl. Abb. C.3). Die Periodendauer der Flankenwechsel gibt darüber hinaus Auskunft über die Rotationsgeschwindigkeit. Zur Ermittlung der Fahrzeuggeschwindigkeit wird die Strecke S_{wu} ermittelt, die bei einer vollständigen Rotation der Motorachse zurückgelegt wird und in Abhängigkeit zur benötigten Zeit gesetzt. Die Strecke ergibt sich aus dem Übertragungsverhältnis des Motors, [8,35:1] (vgl. Herstellerangaben [18]), und dem

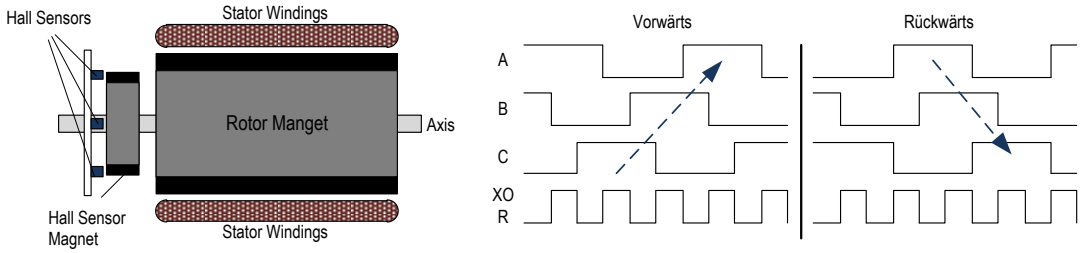


Abbildung C.3 – Durch die Ausgangssignale der Hall Sensoren kann die Rotationsrichtung und die Geschwindigkeit der Antriebsachse direkt am Motor ermittelt werden[31].

Umfang der Antriebsräder; in der Standard-Bereifung des SAVs entspricht dieser 0,2042 m:

$$S_{wu} = \frac{U_r}{i_{Antrieb}} = \frac{0,2042m}{8,35} = 0,024m \tag{66}$$

Zur Festlegung der maximalen und minimalen Geschwindigkeit wurden die Periodendauern der Hall Sensor Signale mit Hilfe der Fernbedienungsanlage gemessen [31]. Aus den Periodenlängen von 5 ms bzw. 180 ms ergeben sich

$$v_{max} = \frac{S_{wu}}{t_{HS\_min}} = \frac{0,024m}{0,005s} = 4,8\frac{m}{s} \tag{67}$$

$$v_{min} = \frac{S_{wu}}{t_{HS\_max}} = \frac{0,024m}{0,18s} = 0,13\frac{m}{s} \tag{68}$$

Die Auswertung der Hall Sensor Pegel erfolgt im V-CONTROLLER durch die VHDL-Komponente HALL SENSOR EVALUATE. Aufgrund der konstant auftretenden Periodenverschiebungen zwischen den Signalen kann die Rotationsrichtung bereits mit zwei Sensorwerten bestimmt werden. In einer FSM werden die Flankenwechsel der Hall Sensor Signale `B` und `C` als Zustandsübergänge abgebildet (vgl. Abb. C.4).

Gleichzeitig wird ein Zähler `v_count` verwendet, um die Länge der Signalperioden und somit die zurückgelegte Wegstrecke zu erfassen. Zur Einsparung von Systemressourcen wird der Zähler mit einer intern generierten 100 kHz Frequenz inkrementiert. Dadurch ergeben sich mit den gemessenen Geschwindigkeitsmaxima von 5 ms bzw. 180 ms, Zählerstände im Bereich von 500 bis 18.000. In Abhängigkeit zur Abtastrate wird `v_count` in die Geschwindigkeitsberechnung aufgenommen $t_{HS} = \frac{v\_count}{100kHz}$:

$$v_{ist} = \frac{S_{wu}}{t_{HS}} = \frac{S_{wu}}{v\_count \cdot 10\mu s} = \frac{S_{wu}/10\mu s}{v\_count} \tag{69}$$

Die Darstellungsform mit `v_count` im Nenner hat den Vorteil, dass der Zählerwert

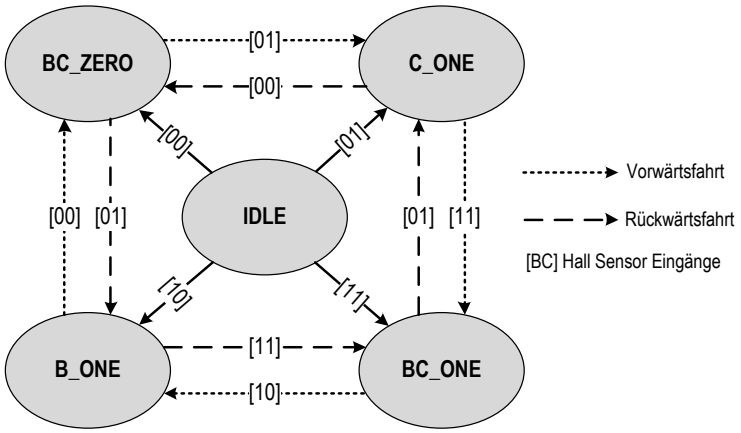


Abbildung C.4 – Durch die Übertragung der Eingangssignale `B` und `C` in einen Zustandsautomaten kann in der PULSE COUNTER Komponente die Fahrtrichtung erkannt werden.

$S_{wu}/10\mu s$ als Konstante behandelt werden kann.

$$\frac{S_{wu}}{10\mu s} = \frac{0,024m}{0,00001s} = 240.000\frac{cm}{s} \tag{70}$$

Der Divisionsschritt zur Geschwindigkeitsermittlung ist als VHDL Prozessorelement umgesetzt, wobei die Division als vorzeichenlose Integer Implementierung nach dem *Trial Subtraction* Algorithmus durchgeführt wird [31]. Dabei wird die Division in Form von Schiebeoperationen und einer Reihe von Subtraktionen realisiert. Das angeführte Listing C.1 zeigt den Algorithmus exemplarisch als C Code.

```
/* Trial subtraction:  q = n / d; r = n % d;  N length of bit vector d */
unsigned udiv_simple(unsigned d, unsigned n, unsigned N){
  unsigned q=0, r=n;
  do{
    N--;              /*next bit*/
    if ((r>>N) >=d){  /* if r>=(1<<N)*/
      r -= (d << N);  /*update remainder*/
      q += (1 << N);  /*update quotient*/
    }
  }while(N);
  return q;
}
```

Listing C.1 – Ein iterativer Ansatz der Division nach dem *Trial Subtraction* Algorithmus als Implementierung in Integer Arithmetik innerhalb der DATA Komponente im HALL SENSOR EVALUATE Modul

Zur Bestimmung der Geschwindigkeit wird das vorgestellte Verfahren in ein Prozessorelement mit Daten- und Steuerpfad überführt (vgl. Abb. C.5), in dem zyklisch die Streckenkonstante $\frac{S_{wu}}{10\mu s}$ durch den Wert des Periodenzählers `v_count` dividiert wird. Die Berechnung

der Geschwindigkeit wird in Zentimeter pro Sekunde vorgenommen; der im Algorithmus angegebene Restwert r wird nicht weiter verwendet.

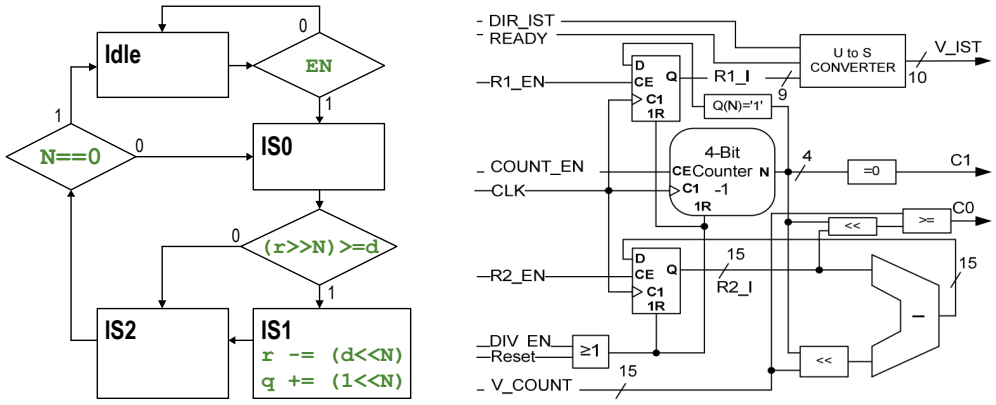


Abbildung C.5 – Überführung des C-Codes für den *Trial Subtraction* Algorithmus in ein Verhaltensmodell, in welchem die Operationensschritte durch Zustandsübergänge realisiert werden (a); (b) zeigt den entsprechenden Datenpfad, in dem zusätzlich das Vorzeichen des Ausgangsvektors durch die erkannte Rotationsrichtung `DIR_IST` angefügt wird.

PI-Regelung und PWM Generierung der Soll-Geschwindigkeit

Mit der ermittelten Ist-Geschwindigkeit `v` und des vorgegebenen Soll-Wertes `v_ctrl` nimmt der V-CONTROLLER die Geschwindigkeitsregelung in Form eines PI-Reglers vor. Dazu wird Gleichung 61 mit den im Anhang C vorgestellten Empfehlungen zur Reglerparametrierung nach Kuhn versehen. Die Regelung erfolgt im skalierten Wertebereich der SAV Umgebung, dazu müssen p_1 und p_2 ebenfalls in diesen Bereich überführt werden. Dabei darf das PWM Ausgangssignal nur 5%[2] der 20 ms PWM Periode aktiv sein, um die Fahrzeugmechanik nicht zu beschädigen. Bei einer Systemfrequenz von 54 MHz entspricht eine Abtastperiode 1.080.000 Systemtakten; Systemtakte = Abtastperiode / Frequenz = 20 ms / $\frac{1}{54MHz} = 1.080.000$. Nach Übertragung in den zulässigen Wertebereich von 5%, stehen für die Abbildung des gesamten Geschwindigkeitsbereiches von $\pm$480 cm/s 54.000 Takte zur Verfügung.

Zur weiteren Bestimmung der Parameter werden die Ergebnisse der Sprungantwortanalyse für unterschiedliche Geschwindigkeiten verwendet. Dabei sind T_Σ und K_P abhängig von den Sprungantworteigenschaften für unterschiedliche Geschwindigkeiten in den skalierten

[2] Wertebereich zwischen 5% und 10%; vgl. Kap. 3.

Wertebereich zu übertragen. Tabelle C.1 zeigt in diesem Zusammenhang die Berechnungsschritte und die skalierten Werte für p_1 und p_2 bei einer Geschwindigkeit von 480 m/s.

Skalierung der Regelungsparameter	
K_s	0,0177 cm/s
K_p nach Kuhn	28,125 s/cm
T_N nach Kuhn	0,136 s
$2T_N K_{P+} T_a K_P$	8,2125 s²/m
$2T_N K_P - T_a K_P$	7,0875 s²/m
p_1	30,19301 s/cm
p_2	26,05699 s/cm

Tabelle C.1 – Die von Kuhn empfohlene Regelungsparametrierung wird mit der PWM Signal Spezifikation (vgl. Tab. B.1) auf die Eigenschaften des SAV skaliert.

Zur Anpassung der Parameter zur Laufzeit werden mit `p1` und `p2` zwei 13-Bit breite Vektoren am Eingang des V-CONTROLLERS bereitgestellt. Diese erlauben eine Integration von Reglerparametern, die an die aktuelle Geschwindigkeit angepasst sind. Die Parameterwerte werden im Q-Format mit einem 5-Bit Ganzzahlanteil und 8-Bit für die Fließkommadarstellung innerhalb der Reglerkomponente verarbeitet. Mit den skalierten Parametern lässt sich die Gleichung 61 in einen Algorithmus überführen, um in einem weiteren Schritt die Regelung in einer VHDL-Komponente zu realisieren:

```
//aktuelle Regelabweichung ermitteln:
e = v_soll - v_ist
//neue Stellgröße berechnen:
u = p1 \cdot e - (p2 \cdot e_alt - u_alt)
//aktuelle Regelabweichung und Stellgröße für die nächste Berechnung speichern:
e_alt = e
u_alt = u
```

Listing C.2 – Der Algorithmus zur Umsetzung des Reglers in Pseudocode

Aus dem Algorithmus geht hervor, dass das Prozessorelement mit nur einem Multiplizierer und einem Subtrahierern umgesetzt werden kann. Dazu werden die Operationen in Teilaufgaben zerlegt und zur optimalen Auslastung der Arithmetik-Komponenten in aufeinanderfolgenen Takten berechnet (vgl. Abb. C.6 (a)). Dies erfolgt durch die Strukturierung der VHDL Komponenten in einen Steuer- und einen Datenpfad; PI_FSM und PI_DATA. Das Signal `pi_on` schaltet die Regelung frei, es wird vom RECEIVER SWITCH bezogen und verhindert die Fortführung der Regelungsfunktion im manuellen Betrieb. Abbildung C.6 (b) zeigt den Datenpfad des PI-Reglers, wobei die Eingänge der Arithmetikoperatoren über Multiplexer und die Register über getaktete Enable Signale aus dem Steuerpfad geschaltet werden.

Für das Ausgangssignal des PI-Reglers werden die Fließkommaanteile der Berechnung aus dem Ergebnis entfernt, wodurch ein 16-Bit breiter `PWM_cnt` Ausgangsvektor entsteht. In diesen 16-Bit lässt sich der zulässige Wertebereich von 5% des PWM Signals (54.000 Takte)

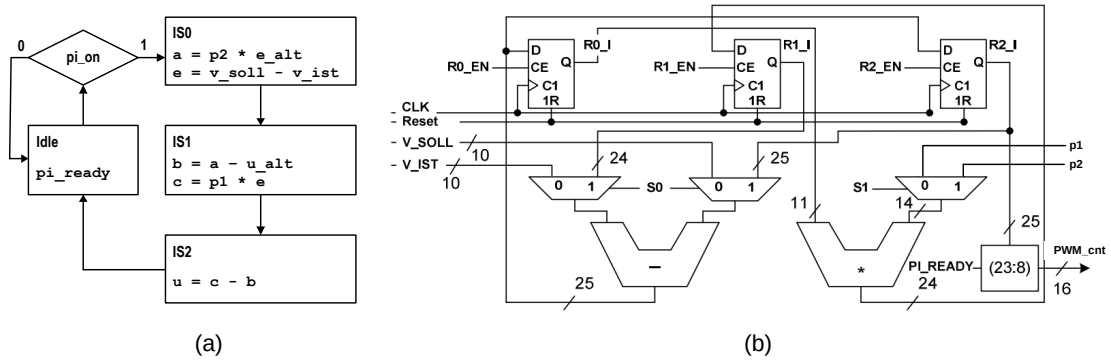


Abbildung C.6 – Die Abbildung des Algorithmus in ein Verhaltensmodell zeigt, dass sich die Berechnung in vier Takten durchführen lässt (a); im Datenpfad werden zum *Resource-Sharing* nur zwei Arithmetikblöcke verwendet (b).

vollständig abbilden.Der Ausgangsvektor wird im GENERATE PWM Modul verwendet, um als Ausgangssignal des V-CONTROLLER ein PWM Signal zu erzeugen (vgl. Abb. C.7).

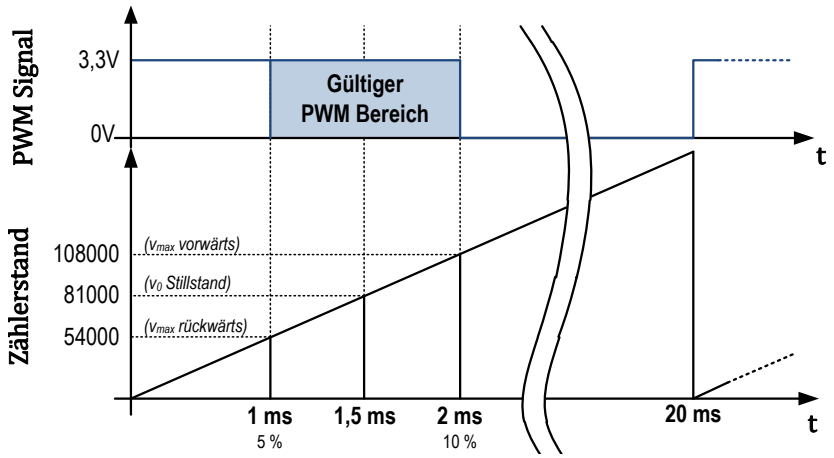


Abbildung C.7 – Das PWM Signal wird mit einem Offset von 54.000 versehen, um den Toleranzbereich der Fahrzeugmechanik nicht zu unterschreiten. Gleichzeitig werden kein Zählerstände über 1.080.000 erzeugt, um die 10% Begrenzung einzuhalten.

Dieses kann vom Fahrtensteller interpretiert und direkt zur Motorsteuerung verwendet werden. Der zulässige Wertebereich zwischen 5% und 10% stellt in Systemtakten bei 54 MHz den Bereich zwischen 54.000 und 108.000 Takten dar. Die Generieung erfolgt über einen Zähler, der die 20 ms PWM Periode abbildet und am Ende ein `INTERRUPT` Signal erzeugt, das als Enable Signal der PI-Regler FSM fungiert. Abhängig vom Ergebnis der Regelung, dem `PWM_cnt` Signal, wird das Ausgangssignal auf *high* gesetzt. Dabei wird die Skalierug auf den zulässigen Wertebereich durch einen Offset von 54.000 Takten umgesetzt.

D Komponenten der MPSoC Plattform

Im autonomen Betrieb des SAV werden die Regelung der Fahrspurführung sowie die Geschwindigkeitsregelung als nebenläufige Prozesse außerhalb der Steuerungsplattform ausgeführt, wodurch diese unabhängig voneinander getestet und optimiert werden können. Zur Koordinierung und Steuerung dieser Module wird ein Multiprozessor System eingesetzt, dass über instanzierte Software-Register mit den Modulen kommuniziert, um deren Regelung zur Laufzeit zu parametrieren. Darüber hinaus wird die Steuerungsplattform zur Integration von Umgebungssensoren eingesetzt; dazu gehören Infrarot-Sensoren zur seitlichen Abstandserkennung und eine Laserscanner-basierte Objekterkennung für den Bereich vor dem SAV (vgl. Abb. 4.14). Ein wesentlicher Punkt bei der Erstellung von Prototypsystemen ist die Darstellung und Aufzeichnung von Systemdaten zur Laufzeit; dazu umfasst die Steuerungsplattform ein JTAG Debug Modul, eine serielle Schnittstelle zum Entwicklungsrechner, ein Bluetooth Modul sowie unterschiedliche General Purpose-Verbindungen zur Ansteuerung von LEDs und Buttons.

Zentraler Bestandteil des Steuerungssystems sind zwei MicroBlaze Softcore Prozessoren, die über zwei getrennte PLB mit peripheren IP-Cores verbunden sind. Dabei übernimmt **microblaze_0** die Koordinations- und Steuerungsaufgaben des SAV, **microblaze_1** wird zur Integration der Abstandssensorik eingesetzt. Als Speicherbausteine werden lokale FPGA Ressourcen in Form von BRAM Blöcken und ein externes DDR2 SDRAM verwendet. Die Kommunikation der MicroBlaze Prozessoren untereinander erfolgt über das externe Speichermodul oder direkt über zwei FSL Verbindungen. Zur Synchronisation der Zugriffe auf den externen Speicher wird ein Hardware Mutex IP verwendet.

Bestandteile des SoC Entwurfs

Der Aufbau der Steuerungsplattform lässt sich in zwei Bereiche gliedern. Das Hardwaredesign behandelt die Integration und Konfiguration der Prozessoren, Bussysteme und IP-Cores in eine geschlossene Systemumgebung. Der zweite Teil umfasst die Implementierung von

Betriebssystemen und Softwareanwendungen, die auf die Ressourcen der erstellten Hardwareplattform zugreifen. Das System zur Steuerung des SAV umfasst dabei eine Reihe von bereitgestellten IP-Cores, die im Folgenden kurz beschrieben werden:

- **MicroBlaze (8.00.b)** [42]
  Der MicroBlaze ist ein 32-Bit RISC Softcore Prozessor, der speziell auf die konkreten Anforderungen und Eigenschaften der unterschiedlichen Xilinx FPGAs angepasst werden kann. Er besitzt zweiunddreißig 32-Bit breite General Purpose Register und verarbeitet 32-Bit Instruktionen in einer konfigurierbaren drei bis fünf stufigen Pipeline (vgl. Abb. D.1). Er stellt Schnittstellen für die Anbindung von Bussystemen

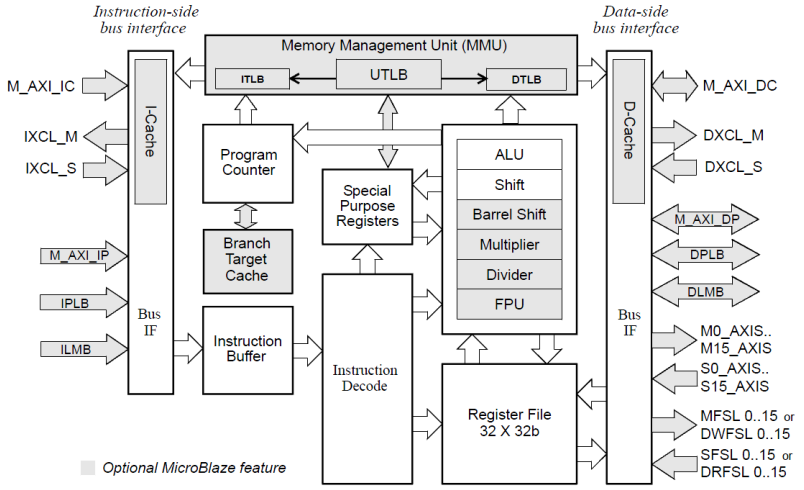


Abbildung D.1 – Der konfigurierbare MicroBlaze Softcore Prozessor, mit seinen Schnittstellen zu den verfügbaren Bussystemen [42]

bereiht, über die Speicherzugriffe und die Kommunikation mit Peripheriemodulen realisiert werden. Dazu gehören der LMB (*Local Memory Bus*) zur Anbindung von Daten- und Instruktionspfaden an den FPGA internen BRAM Speicher, der PLB (*Processor Local Bus*) zur Kommunikation mit anderen IP-Cores und der XCL (*Xilinx Cache Link*) zur Verbindung mit einem Speicherkontroller für Zugriffe auf externe Speicherbausteine. In der aktuellen Version des MicroBlaze sind zudem die neuen AXI4[1] (*Advanced eXtensible Interface*) Schnittstellen implementiert. Darüber hinaus stehen bis zu 16 Stream Link Interfaces zur Verfügung, über die Daten von Hardwarebeschleunigungsmodulen verarbeitet werden können. Diese können entweder über FSL- oder AXI4-Stream Schnittstellen angesprochen werden. Für Speicherzugriffe kann der MicroBlaze mit separat konfigurierbaren Caches für Daten und Instruktionen versehen werden. Die Einrichtung virtueller Speicheradressen ist

[1] Das AXI4 Protokoll ist Teil der AMBA (*Advanced Microcontroller Bus Architecture*) Protokollfamilie, einem von der Firma ARM betriebenem, offenen Standard für Mikrokontroller Bussysteme, und wurde für die aktuellsten Versionen der Xilinx FPGAs als Bussystem eingeführt [49].

über den Einsatz einer Memory Management Unit gegeben. Zusätzlich stellt er einen bis zu 64-Bit breiten Hardware Multipizierer, einen optional konfigurierbaren Hardware Divider, ein Barrel Shifter zur verbesserten Ausführung von Schiebeoperationen und eine Floating Point Unit zur Verfügung. Der Aufbau des Softcore Prozessors ist modular, so dass diese Eigenschaften je nach Anforderung an die Zielplattform in das Systemdesign aufgenommen werden können.

- **Local Memory Bus (LMB) V10 (v1.00a)** [33]
  Der LMB dient als Schnittstelle zur Anbindung separater Daten- und Instruktionsleitungen für Speicherzugriffe des MicroBlaze auf das on-chip BRAM (vgl. hierzu [45]) des FPGAs. Er ist als „Single Master Bus“ implementiert, wodurch kein Arbiter zur Zugriffskontrolle eingesetzt werden muss. Gleichzeitig ist er auf die Verwendung möglichst weniger FPGA Ressorusen optimiert; dadurch entstehen kurze Signalpfade und lokale Speicherzugriffe können innerhalb von nur zwei Taktzyklen erfolgen. Diese schnellen Zugriffszeiten bieten die Implementierung der *.heap* und *.stack* Sektionen zur Softwareausführung innerhalb des BRAMs an.

- **LogiCORE IP Processor Local Bus (PLB) v4.6 (v1.05a)** [39]
  Der PLB stellt eine umfangreiche Schnittstelle für die Kommunikation unterschiedlicher IP-Cores bereit. Er besteht aus einem zentralen Bus Arbiter, der die Zugriffskontrolle auf die Busleitungen nach Round-Robin oder Prioritäten-basiertem Zugriff für eine konfigurierbare Anzahl an Master und Slave Teilnehmern regelt. Dabei stehen voneinander getrennte *address*, *read* und *write* Verbindungen zur Verfügung, was die gleichzeitige Ausführung von Lese- und Schreiboperationen ermöglicht. Der Informationsaustausch ist über Register geregelt, was die speicherbezogene Adressierung *Memory Mapped* des MicroBlaze auf die angeschlossenen IP-Cores unterstützt.

- **LogiCORE IP Fast Simplex Link (FSL) V20 Bus (v2.11d)** [37]
  Der FSL ist ein unidirektionaler Kommunikationskanal zum Datentransfer von einem Master- zu einem Slave Teilnehmer. Er stellt ein Streaming Interface dar, das auch hohe Datenraten kontinuierlich übertragen kann. Typischerweise wird er zur Kommunikation zwischen einem MicroBlaze und Beschleunigungsmodulen auf dem FPGA verwendet. Damit neben dem FSL keine weiteren Verbindungen für Steuersignale zwischen den Teilnehmern implementiert werden müssen, besteht die Möglichkeit die Daten mit Kontrollflags zu versehen. Dadurch kann der entsprechende Kommunikationspartner Synchronisations- und Steuersignale detektieren. Der Aufbau des FSL entspricht dem FIFO Prinzip, wobei die Lese- und Schreibzugriffe mit unterschiedlichen Taktfrequenzen asynchron erfolgen können. Im SAV sind die beiden MicroBlaze Prozessoren über zwei FSL miteinander verbunden. Dadurch kann eine

direkte Kommunikation für Synchronisations- und Kontrollbenachrichtigungen stattfinden, ohne den PLB bzw. den MPMC zu blockieren.

- **Multi-Port Memory Controller (MPMC) (6.02.a)** [46]
  Der MPMC ist ein frei konfigurierbarer Memory Controller zur Anbindung von externen SDRAM/DDR/DDR2/DDR3/LPDDR Speichermodulen an die MicroBlaze bzw. PowerPC Prozessoren. Dazu stellt er insgesamt acht Ports zur Verfügung, die mit den prozessor-spezifischen Bussystemen verbunden werden können; für den MicroBlaze stehen die Anbindung über PLB, XCL oder AXI4 zur Verfügung. Die Einrichtung der Schnittstellen zum externen Speicher erfolgt über eine ausführliche Parametrierung; weiterhin lässt sich der Arbitrierungsalgorithmus frei konfigurieren. Über die unterschiedlichen Ports lassen sich mehrere Prozessoren mit dem externen Speicher verbinden, um einen gemeinsamen Speicher in Multiprozessor Systeme zu integrieren.

- **LogiCORE IP Mutex (v1.00a)** [38]
  Der Mutex Core wird in Multiprozessor Systemen verwendet, um den Zugriff auf gemeinsame Ressourcen zwischen den Prozessoren zu synchronisieren. Dazu stehen bis zu acht PLB bzw. AXI4 Schnittstellen zur Verfügung, über die auf eine konfigurierbare Anzahl von 32-Bit Memory Mapped Mutex Registern zugegriffen wird. Obwohl jede der angeschlossenen Schnittstellen auf alle 32 Mutex Register zugreifen kann, ist immer nur der Zugriff über einen Port zur Zeit erlaubt; alle weiteren Zugriffe werden blockiert. Jedes Mutex Register besteht aus drei Bitfeldern, dem `LOCK` Bit, der `CPUID` und einem nicht vom Benutzer zugänglichen `HWID` Bereich. Mit dem `LOCK` Bit wird die Belegung des Mutex festgehalten; bei einem Zugriff, prüft der zugreifende Prozessor, ob das `LOCK` Bit gesetzt ist. Wenn es nicht gesetzt ist, schreibt der Prozessor seine ID in das `CPUID` Feld und setzt das `LOCK` Bit, um den Mutex zu belegen. Das `HWID` Feld wird durch den Parameter `C_ENABLE_HW_PROT` aktiviert und wird im Hintergrund mit der ID des verwendeten PLB bzw. AXI4 Ports belegt. Da die `CPUID` vom Anwender manipuliert werden kann, stellt dies einen zusätzlichen Sicherungsmechanismus dar.

- **MicroBlaze Debug Module (MDM) (v2.00.a)** [41]
  Das MDM erlaubt das Debuggen von bis zu acht MicroBlaze Prozessoren über die JTAG Schnittstelle. Zudem beinhaltet es eine UART Schnittstelle, die über den PLB mit dem System verbunden wird. Das JTAG UART Interface dient ebenfalls zur Kommunikation mit dem XMD (*Xilinx Microprocessor Debug*) Tool, über welches das Laufzeitverhalten mit Debugbefehlen beeinflusst wird. Die JTAG UART Verbindung wird weiterhin verwendet, um die Bitfiles sowie den Programmcode in den

ELF[2] Dateien auf das FPGA zu übertragen. Im Header der ELF Dateien sind dazu die Adressen beschrieben, an denen die einzelnen Software Sektionen im Speicher hinterlegt werden.

- **LogiCORE IP XPS Timer/Counter (v1.02a)** [40]
  Das XPS Timer Modul ist ein 32-Bit Timer der über den PLB in das System integriert wird. Es besteht aus zwei identischen Timer Modulen, die jeweils mit einem Load Register und einem Counter Register versehen sind. Der Timer/Counter kann auf drei Betriebsarten konfiguriert werden; Generate Mode, zur Erzeugung von Ereignissen bei Erreichen eines im Load Register gespeicherten Zählers. Dem Capture Mode zur Wiedergabe des Zählers bei einem externen Capture Signal und dem PWM Mode, für eine Pulsweitenmodulation. Die Erzeugung eines Interrupts wird im Generate Mode bei Erreichen des im Load Register gesicherten Zählers realisiert; hierzu wird bei der Initialisierung des Timers das `INIT` Flag gesetzt. Im SAV System werden die Timer hauptsächlich zur Erzeugung der Interrupts für die Synchronisation des Xilkernel Betriebssystems verwendet.

- **XPS UART Lite (v1.01a) und XPS 16550 UART (v3.00a)** [36], [50]
  Das XPS UART Lite IP wird zur Einrichtung einer seriellen Kommunikationsschnittstelle, beispielsweise einer RS232 Verbindung, eingesetzt und verbindet diese mit dem PLB. Dabei werden die parallelen bis zu 8-Bit breiten Datenpakete vom PLB entgegengenommen, serialisiert und übertragen. Auf der anderen Seite empfängt es die seriellen Daten und wandelt diese in einen parallelen Datenstrom um. Der UART Lite enthält zwei voneinander getrennte Eingangs- und Ausgangs-FIFOs, eine Kontrolleinheit zur Erzeugung von Interrupts und einen frei konfigurierbaren Baudraten Generator. Die Datenübertragung kann nach den standardisierten Verfahren mit einem Stop Bit sowie einem *odd/even/no* Paritätsbit versehen werden.

  Der XPS 16550 UART bietet im Vergleich zu der Lite Version eine vollständige Implementierung der seriellen Schnittstelle. Neben den Eigenschaften die auch beim UART Lite zu finden sind, stellt dieser UART ein zusätzliches Modem Interface zur Verfügung. Der XPS 16550 UART stellt eine Softcore Implementierung des National Semiconductor PC16550D UART Bausteins dar, und ist mit den gleichen Eigenschaften ausgestattet. Wie der UART Lite, wird auch dieses IP an den PLB angeschlossen, um eine Verbindung mit einer externen seriellen Schnittstelle herzustellen. In Abbildung D.2 werden die Blockschaltbilder der beiden UARTs gegenübergestellt. Im MPSoC für die Fahrzeugsteuerung werden die UART Module verwendet, um Daten von den Sensoren aufzunehmen, die Sensoren parametrieren zu können oder als Schnittstellen zum Entwicklungsrechner für Statusanzeigen und Aufzeichnungen.

[2] Die ELF (*Executable and Linkable Format*) Files sind binäre Dateien, die vom Compiler erzeugt und auf den Zielprozessoren ausgeführt werden.

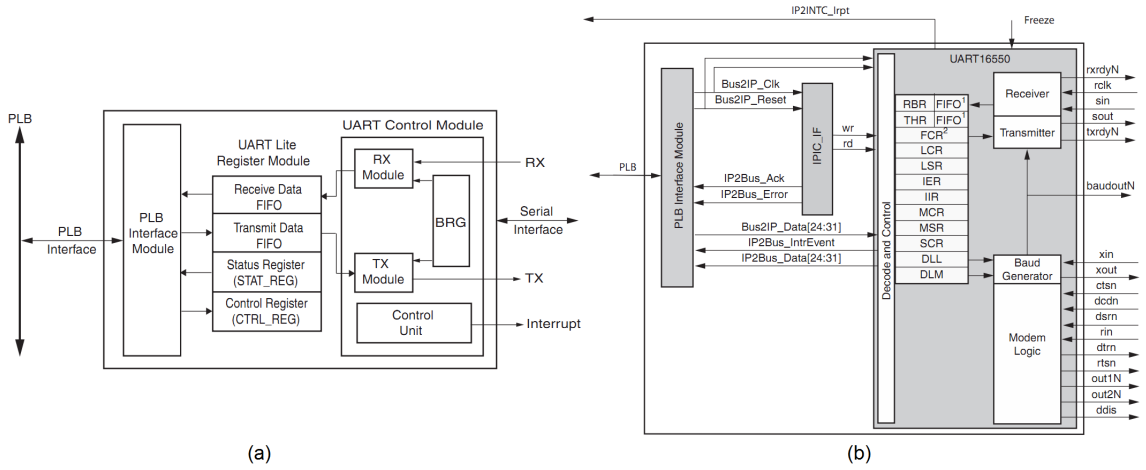


Abbildung D.2 – Das XPS UART Lite IP (a) enthält lediglich die Implementierung einer standardisierten seriellen Verbindung, während das XPS 16550 UART Modul (b) einer Softcore-Umsetzung des National Semiconductor PD16550 Bausteins entspricht.

- **XPS General Purpose Input/Output (GPIO) (v2.00.a)** [35]
  Das XPS GPIO Peripheral ist als ein 32-Bit Slave Modul an den PLB angeschlossen, dessen Hauptaufgabe die Bereitstellung von User I/O Pins aus dem Microprozessor System nach Außen ist. Dabei besteht die Möglichkeit jedes Bit, der 32-Bit breiten Kanäle, dynamisch als Eingangs- oder Ausgangssignal zu definieren. Darüber hinaus kann ein gesamter Kanal als uni- oder bidirektionale Verbindung angelegt werden. Das GPIO Modul ist mit einer Kontrollstruktur ausgerüstet, die es erlaubt Interrupts zu erkennen und auszulösen.

Neben den bereitgestellten Xilinx IP-Cores werden im MPSoC des SAV Systems auch eigene Module verwendet, um Daten- und Steuerungssignale zwischen den Beschleunigungsmodulen und der Steuerungsanwendung austauschen zu können:

- FRAME GRABBER
  Das Frame Grabber Modul realisiert die Übertragung eines Kamerabildes aus der Bildbearbeitungspipeline in das Steuerungssystem. Dort wird es zunächst im externen DDR Speicher abgelegt und anschließend über die RS232 Schnittstelle an den Entwicklungsrechner zu Dokumentationszwecken übertragen. Die Integration in das Steuerungssystem verläuft über **microblaze_0** , der über zwei FSL Verbindungen mit dem Framegrabber kommuniziert (vgl. Abb. D.3); wodurch eine bidirektionale Kommunikation erfolgen kann. Der erste FSL vom MicroBlaze (Master) zur Bildverarbeitungspipeline (Slave) dient zur Signalisierung einer neuen Bildanfrage. Der zweite FSL ist entgegengesetzt verbunden und überträgt jeweils 4 Bilddatenpunkte auf einmal. Die Realisierung der Datenverbindung über den FSL hat seine Begründung in der großen Datenmenge, die für ein Bild übertragen werden muss. Ein Bild

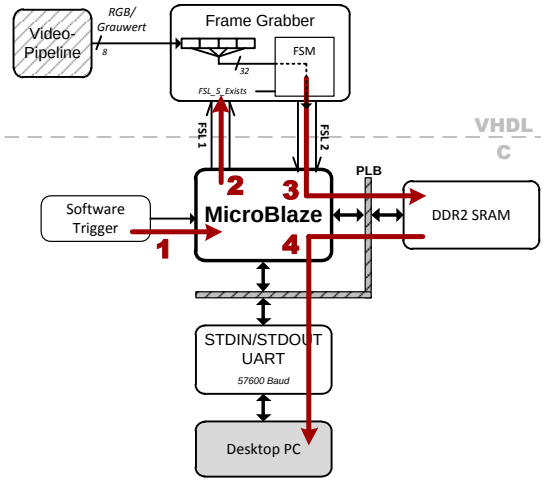


Abbildung D.3 – Das Abspeichern eines Bildes aus der Bildverarbeitungspipeline verläuft in vier Schritten: (1)Erzeugung der Bildanfrage(z.B. durch User-Input), (2)Signalisierung an das Frame Grabber Modul, (3)Übertragung der Daten an den MicroBlaze und Speicherung im externen RAM und (4) Übertragung der Daten aus dem RAM über die UART Schnittstelle an den Entwicklungsrechner.

besteht aus 640x240 Bildpunkten[3], die jeweils 8-Bit umfassen, was eine Gesamtmenge von 150 kB ausmacht. Diese Daten müssen zwischen zwei unterschiedlichen Clockdomänen ausgetauscht werden, was bei einer Realisierung über den PLB eine aufwendigen Synchronisationsstrategie voraussetzt und eine andauernde Blockierung des Busses nach sich ziehen würde. Die Eigenschaft des FSL, mit unterschiedlichen Frequenzen gelesen und beschrieben werden zu können, umgeht diesen Aufwand; zumal die direkte Verbindung zwischen HW-Modul und MicroBlaze zumindest zum Zeitpunkt der Datenaufnahme keine anderen Komponenten blockiert.

- SAV CONNECTOR
  Zur Steuerung der Fahrzeugführungs- und Geschwindigkeitsregler wird das SAV Connector IP verwendet. Dieses stellt eine Anbindung von Software-Registern bereit, die auf der einen Seite über den PLB vom **microblaze_0** gelesen und beschrieben werden können, und auf der anderen Seite im SAV Control Modul zur Parametrierung der Regler verwendet werden. Auch hier muss eine Synchronisation zwischen den Clockdomänen erfolgen, um keine instabilen Registerzustände zu erzeugen. Der SAV Connector verzögert dafür die Acknowledge-Signale, die vom PLB Master nach dem Beenden des Schreibevorgangs gesetzt werden. Somit wird sichergestellt, dass diese Information in der niedriger getakteten Clockdomäne des HW-Beschleunigermoduls

[3]Die Bilddaten werden mit 240 Zeilen aus der Pipeline übertragen bevor sie das Deinterlace Modul durchlaufen; dieser Vorgang kann auch im Übertragungsvorgang zum Entwicklungsrechner vorgenommen werden.

nicht verloren geht.

- Das HAW-SPI IP Modul
  Zur Anbindung der Infrarot-Sensorik an das Steuerungssystem wird der HAW-SPI IP-Core verwendet. Die IR-Sensoren werden zur seitlichen Abstandsermittlung eingesetzt und liefern analoge Spannungen, abhängig von der ermittelten Distanz. Über zwei Pmod AD1 Module, die jeweils zwei ADU (*Analog-Digital-Umsetzter*) vereinen, werden die Messwerte der vier IR-Sensoren digitalisiert und durch eine Polynomapproximation in entspechende Abstandsdistanzen umgesetzt (vgl. Abb. D.4). Die Ermittlung der Koeffizienten für die Festlegung der Polynome wurde durch einen Kalibrierungsschritt vorgenommen [1]. Da sich diese Koeffizienten zur Laufzeit nicht ändern, sind sie als konstante Werte im HAW-SPI angelegt. Die vier ermittelten Abstandsergebnisse werden in zwei Software-Registern zur Verfügung gestellt, von wo sie über den PLB vom **microblaze_1** bezogen werden. Zur Signalisierung von errechneten Messwerten, erzeugt der SPI_Controller einen Interrupt.

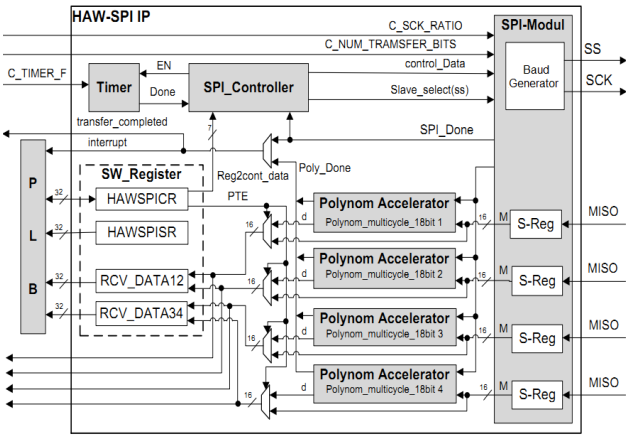


Abbildung D.4 – Das HAW-SPI Modul enthält zwei Softwareregister, die die Ergebnisse der Abstandsmessungen, nach deren Transformation in den Polynom Accelerator Modulen, zur Verfügung stellen[1].

Das Xilkernel RTOS als Grundlage der Steuerungssoftware

Die Anwendungssoftware für das MPSoC zur Fahrzeugsteuerung basiert auf dem Einsatz des Echtzeitbetriebssystems Xilkernel. Dieses besitzt einen modular aufgebauten Kernel, der eine Anpassung auf die spezielle Systemkonfiguration des SAV erlaubt (vgl. Abb. D.5). Xilkernel bietet die Unterstützung der meisten POSIX Funktionen zur Implementierung von Threads und dem synchronisierten Zugriff auf kritische Bereiche, wie *mutex* und *semaphore*. Der grundlegedende Aufbau einer Xilkernel Anwendung erfolgt durch die Instanziierung eines Masterthreads innerhalb des Kernel-Bootvorganges, aus dem die übrigen Softwarethreads gestartet werden.

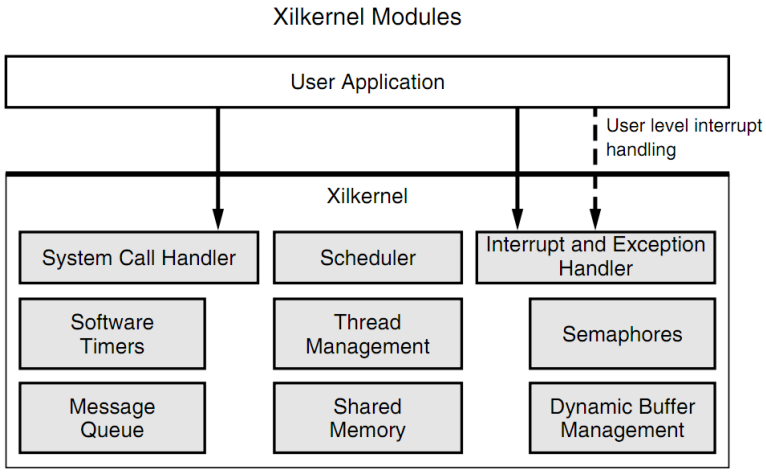


Abbildung D.5 – Das modular aufgebaute Xilkernel Betriebsystem wird speziell auf die Systemanforderungen mit unterschiedlichen Komponenten konfiguriert [44].

Die Threadfolge lässt sich wahlweise auf Round-Robin oder Prioritäten-basiertes Scheduling festlegen. Die Interprozesskommunikation erfolgt über *message queues* oder dem geregelten Zugriff auf geteilte Speicherbereiche. Der Interrupt eines externen HW-Timers erzeugt die Kernelticks für den Xilkernel Systemtakt. Dieser bildet die Grundlage der Schedulingperioden und wird zur Generierung von Softwaretimern verwendet. Der modulare Aufbau des Betriebssystems begünstigt zudem die Integration von zusätzlichen Softwarebibliotheken, mit denen zum Beispiel ein TCP/IP Stack, ein FAT32 Dateisystem oder Treiber zur Verwendung von Seriellen Flash Bausteinen in den Funtkionsumfang des Systems aufgenommen werden können.

E Anhang

Anstieg der verfügbaren Logik Ressourcen auf FPGAs

Vendor	Product Family	Name	Device	Year of Release	Logic Cells	CLB*1	Multiplier	DCM / MMCM or PLL*2	User I/O pins	BRAM Bits (K =1024)	References	
											Document ID	Date
XILINX	Spartan	Spartan 2	XC2S200	2000	5292	1176	-	-	264	75K	DS001	Jun 08
		Spartan 3	XC3S5000	2003	74880	8320	104	4	633	1872K	DS099	Dec 09
		Spartan 3E	XC3S1600E	2005	33192	3688	36	8	376	648K	DS312	Aug 09
		Spartan6	XC6SLX150	2009	147443	11519	180	12	576	4824K	DS160	Oct 11
	Virtex	Virtex II	XC2VP100	2002	99216	11024	444	12	1164	7992K	DS083	Jun 11
		Virtex 4	XC4VLX200	2004	200448	22272	96	12	960	6048K	DS112	Aug 10
		Virtex6	XC6VLX760	2009	758784	59280	864	36	1200	25K	DS150	Jan 12
		Virtex7	XC7VX1140T	2010	1139200	89000	3360	24	1100	67680K	DS180	Jan 12
ALTERA	Cyclone	Cyclone	EP1C20	2003	20060	-	-	2	301	288K	C51001-1.5	May 08
		Cyclone II	EP2C70	2004	68416	-	150	4	622	1125K	CII51001-3.2	Feb 08
		Cyclone III	EP3C120	2007	119088	-	288	4	531	3888K	CIII51001-2.3	Dec 11
		Cyclone V	5CEB9	2011	300000	-	812	4	488	12760K	AIB-01016-1.2	May 11
	Stratix	Stratix	EP1S80	2002	79040	-	176	12	1238	7253K	S51001-3.2	Jul 05
		Stratix II	EP2S180	2004	179400	-	384	12	1170	9163K	SII51001-4.2	May 07
		Stratix III	EP3SL340	2006	338000	-	576	12	1120	16272K	SIII51001-1.8	Mar 10
		Stratix V	5SGXAB	2010	952000	-	1062	28	696	52000K	SV51001-2.2	Dec 11

*1 CLB (Configurable Logic Block)

*2 DCM (Digital Clock Manager) / MMCM (Mixed-mode Clock Manager) / PLL (Phase-Locked Loops)

Tabelle E.1 – Die Fortschritte der Mikroelektronik wirken sich auch auf die Logikressourcen der FPGA Technologie aus, was immer größere Systemkonfigurationen zulässt.

Timingdiagramm eines gesamten Bildes im Interlaced-Modus

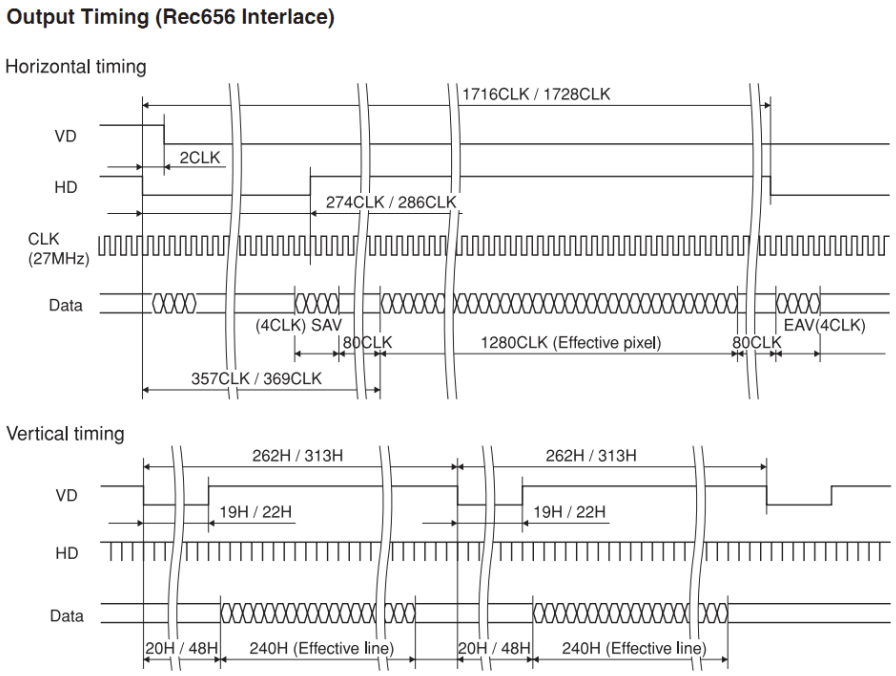
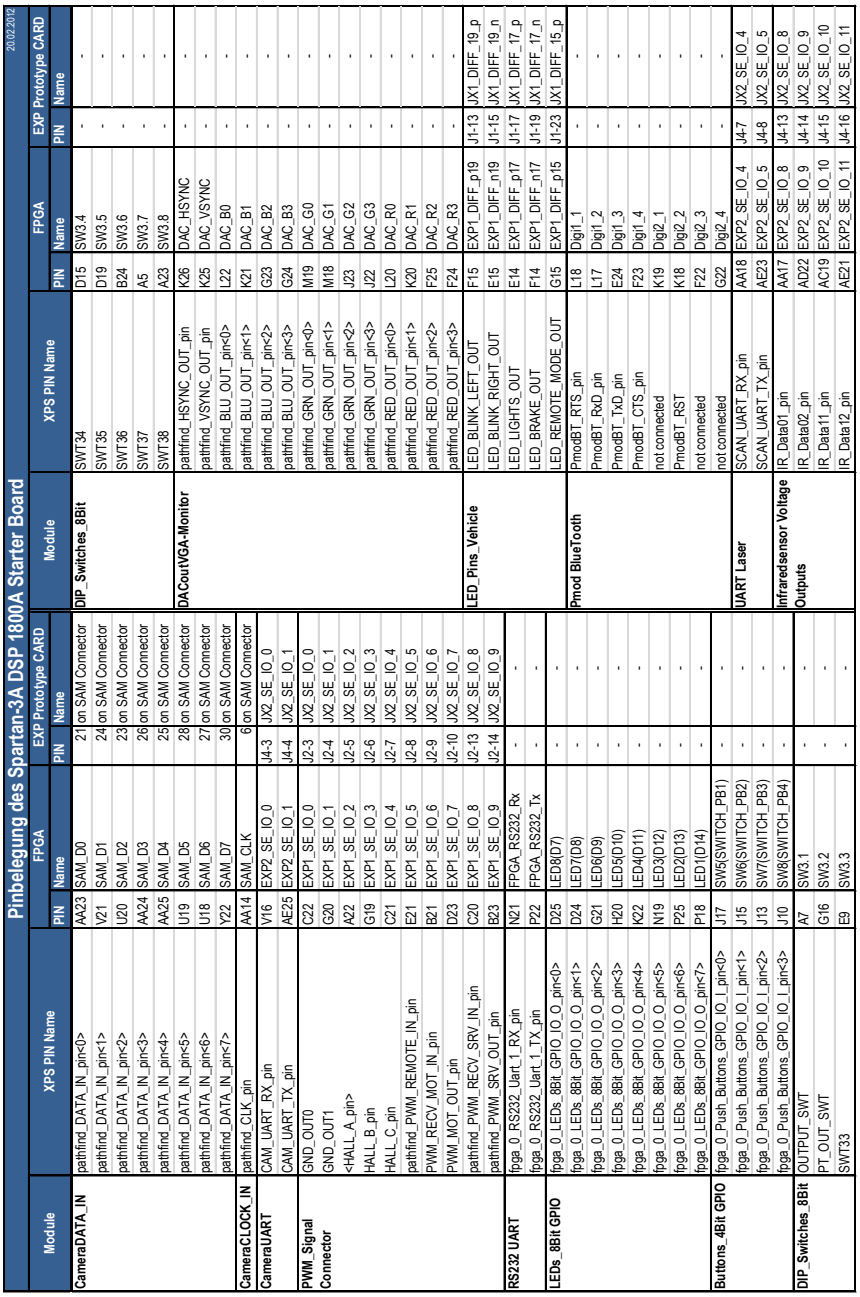


Abbildung E.1 – Im Interlace Modus nach ITU-R BT.656 enthält der Videodatenstrom zwei Halbbilder pro Frame, wobei im ersten die geraden Zeilen eines Bildes, im zweiten die ungeraden Zeilen, enthalten sind.

Vollständige Pinbelegung des SAV



Pinbelegung des Spartan-3A DSP 1800A Starter Board												20.02.2012
Module	XPS PIN Name	FPGA		EXP Prototype CARD		Module	XPS PIN Name	FPGA		EXP Prototype CARD		
		PIN	Name	PIN	Name			PIN	Name	PIN	Name	
CameraDATA_IN	pathfind_DATA_IN_pin<0>	AA23	SAM_D0	21	on SAM Connector	DIP_Switches_8Bit	SWT34	D15	SW3.4	-	-	
	pathfind_DATA_IN_pin<1>	V21	SAM_D1	24	on SAM Connector		SWT35	D19	SW3.5	-	-	
	pathfind_DATA_IN_pin<2>	U20	SAM_D2	23	on SAM Connector		SWT36	B24	SW3.6	-	-	
	pathfind_DATA_IN_pin<3>	AA24	SAM_D3	26	on SAM Connector		SWT37	A5	SW3.7	-	-	
	pathfind_DATA_IN_pin<4>	AA25	SAM_D4	25	on SAM Connector		SWT38	A23	SW3.8	-	-	
	pathfind_DATA_IN_pin<5>	U19	SAM_D5	28	on SAM Connector	DACoutVGA-Monitor	pathfind_HSYNC_OUT_pin	K26	DAC_HSYNC	-	-	
	pathfind_DATA_IN_pin<6>	U18	SAM_D6	27	on SAM Connector		pathfind_VSYNC_OUT_pin	K25	DAC_VSYNC	-	-	
	pathfind_DATA_IN_pin<7>	Y22	SAM_D7	30	on SAM Connector		pathfind_BLU_OUT_pin<0>	L22	DAC_B0	-	-	
CameraCLOCK_IN	pathfind_CLK_pin	AA14	SAM_CLK	6	on SAM Connector		pathfind_BLU_OUT_pin<1>	K21	DAC_B1	-	-	
CameraUART	CAM_UART_RX_pin	V16	EXP2_SE_IO_0	J4-3	JX2_SE_IO_0		pathfind_BLU_OUT_pin<2>	G23	DAC_B2	-	-	
	CAM_UART_TX_pin	AE25	EXP2_SE_IO_1	J4-4	JX2_SE_IO_1		pathfind_BLU_OUT_pin<3>	G24	DAC_B3	-	-	
PWM_Signal Connector	GND_OUT0	C22	EXP1_SE_IO_0	J2-3	JX2_SE_IO_0		pathfind_GRN_OUT_pin<0>	M19	DAC_G0	-	-	
	GND_OUT1	G20	EXP1_SE_IO_1	J2-4	JX2_SE_IO_1		pathfind_GRN_OUT_pin<1>	M18	DAC_G1	-	-	
	<HALL_A_pin>	A22	EXP1_SE_IO_2	J2-5	JX2_SE_IO_2		pathfind_GRN_OUT_pin<2>	J23	DAC_G2	-	-	
	HALL_B_pin	G19	EXP1_SE_IO_3	J2-6	JX2_SE_IO_3		pathfind_GRN_OUT_pin<3>	J22	DAC_G3	-	-	
	HALL_C_pin	C21	EXP1_SE_IO_4	J2-7	JX2_SE_IO_4		pathfind_RED_OUT_pin<0>	L20	DAC_R0	-	-	
	pathfind_PWM_REMOTE_IN_pin	E21	EXP1_SE_IO_5	J2-8	JX2_SE_IO_5		pathfind_RED_OUT_pin<1>	K20	DAC_R1	-	-	
	PWM_RECV_MOT_IN_pin	B21	EXP1_SE_IO_6	J2-9	JX2_SE_IO_6		pathfind_RED_OUT_pin<2>	F25	DAC_R2	-	-	
	PWM_MOT_OUT_pin	D23	EXP1_SE_IO_7	J2-10	JX2_SE_IO_7		pathfind_RED_OUT_pin<3>	F24	DAC_R3	-	-	
	pathfind_PWM_RECV_SRV_IN_pin	C20	EXP1_SE_IO_8	J2-13	JX2_SE_IO_8	LED_Pins_Vehicle	LED_BLINK_LEFT_OUT	F15	EXP1_DIFF_p19	J1-13	JX1_DIFF_19_p	
	pathfind_PWM_SRV_OUT_pin	B23	EXP1_SE_IO_9	J2-14	JX2_SE_IO_9		LED_BLINK_RIGHT_OUT	E15	EXP1_DIFF_n19	J1-15	JX1_DIFF_19_n	
RS232 UART	fpga_0_RS232_Uart_1_RX_pin	N21	FPGA_RS232_Rx	-	-		LED_LIGHTS_OUT	E14	EXP1_DIFF_p17	J1-17	JX1_DIFF_17_p	
	fpga_0_RS232_Uart_1_TX_pin	P22	FPGA_RS232_Tx	-	-		LED_BRAKE_OUT	F14	EXP1_DIFF_n17	J1-19	JX1_DIFF_17_n	
LEDs_8Bit GPIO	fpga_0_LEDs_8Bit_GPIO_IO_O_pin<0>	D25	LED8(D7)	-	-		LED_REMOTE_MODE_OUT	G15	EXP1_DIFF_p15	J1-23	JX1_DIFF_15_p	
	fpga_0_LEDs_8Bit_GPIO_IO_O_pin<1>	D24	LED7(D8)	-	-	Pmod BlueTooth	PmodBT_RTS_pin	L18	Digi1_1	-	-	
	fpga_0_LEDs_8Bit_GPIO_IO_O_pin<2>	G21	LED6(D9)	-	-		PmodBT_RxD_pin	L17	Digi1_2	-	-	
	fpga_0_LEDs_8Bit_GPIO_IO_O_pin<3>	H20	LED5(D10)	-	-		PmodBT_TxD_pin	E24	Digi1_3	-	-	
	fpga_0_LEDs_8Bit_GPIO_IO_O_pin<4>	K22	LED4(D11)	-	-		PmodBT_CTS_pin	F23	Digi1_4	-	-	
	fpga_0_LEDs_8Bit_GPIO_IO_O_pin<5>	N19	LED3(D12)	-	-		not connected	K19	Digi2_1	-	-	
	fpga_0_LEDs_8Bit_GPIO_IO_O_pin<6>	P25	LED2(D13)	-	-		PmodBT_RST	K18	Digi2_2	-	-	
	fpga_0_LEDs_8Bit_GPIO_IO_O_pin<7>	P18	LED1(D14)	-	-		not connected	F22	Digi2_3	-	-	
Buttons_4Bit GPIO	fpga_0_Push_Buttons_GPIO_IO_I_pin<0>	J17	SW5(SWITCH_PB1)	-	-		not connected	G22	Digi2_4	-	-	
	fpga_0_Push_Buttons_GPIO_IO_I_pin<1>	J15	SW6(SWITCH_PB2)	-	-	UART Laser	SCAN_UART_RX_pin	AA18	EXP2_SE_IO_4	J4-7	JX2_SE_IO_4	
	fpga_0_Push_Buttons_GPIO_IO_I_pin<2>	J13	SW7(SWITCH_PB3)	-	-		SCAN_UART_TX_pin	AE23	EXP2_SE_IO_5	J4-8	JX2_SE_IO_5	
	fpga_0_Push_Buttons_GPIO_IO_I_pin<3>	J10	SW8(SWITCH_PB4)	-	-	Infraredsensor Voltage	IR_Data01_pin	AA17	EXP2_SE_IO_8	J4-13	JX2_SE_IO_8	
DIP_Switches_8Bit	OUTPUT_SWT	A7	SW3.1	-	-	Outputs	IR_Data02_pin	AD22	EXP2_SE_IO_9	J4-14	JX2_SE_IO_9	
	PT_OUT_SWT	G16	SW3.2	-	-		IR_Data11_pin	AC19	EXP2_SE_IO_10	J4-15	JX2_SE_IO_10	
	SWT33	E9	SW3.3	-	-		IR_Data12_pin	AE21	EXP2_SE_IO_11	J4-16	JX2_SE_IO_11	

Tabelle E.2 – Vollständiges Pinout des SAV Systems; die XPS Pin Bezeichnungen entstammen dem UCF File, die FPGA-PIN Namen sind in [34], und die Bezeichnungen der Pins des Prototyping Boards in [4] zu finden.

Schaltbilder zu den entwickelten SAV Platinen

DC-DC-Converter Platine

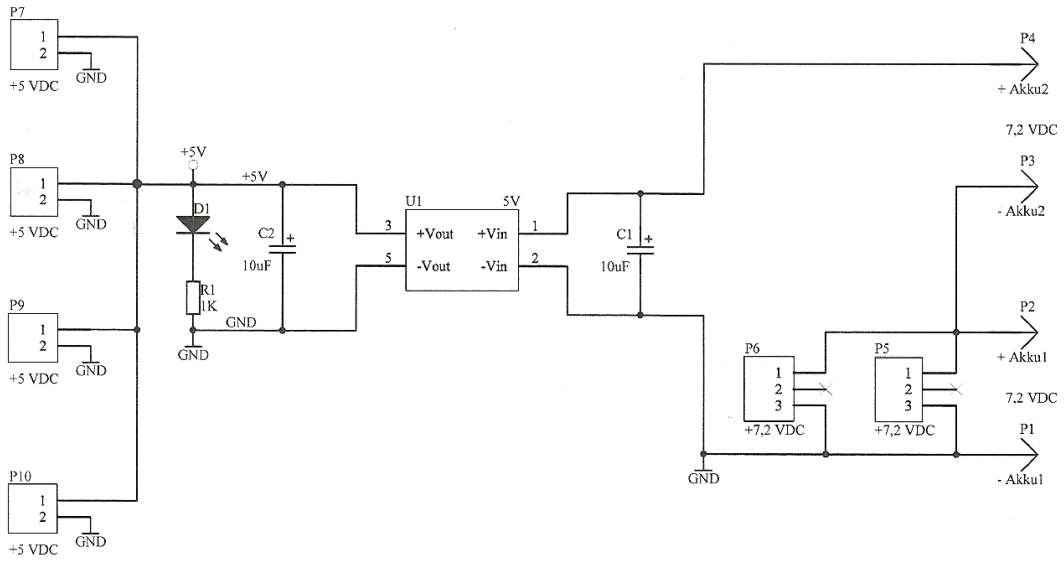


Abbildung E.2 – Die DC-DC-Converter Platine mit zwei 7,2V Anschlüssen für die Eingangsspannung, dem Spannungswandler und vier 5V bzw. zwei 7,2V Ausgangsspannungs-Anschlüssen

FCB-FFC-Pmod Adapter Platine

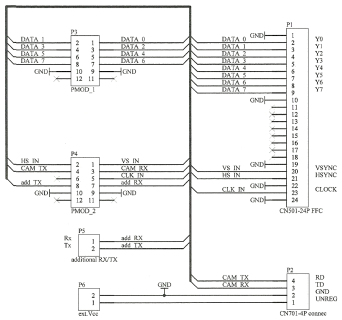


Abbildung E.3 – Die FCB-FFC-Pmod Adapter Platine setzt die Kamerasignale auf zwei Digilent Pmod Anschlüsse um und enthält zudem einen Anschluss für die Versorgungsspannung der Kamera.

LED-Platine

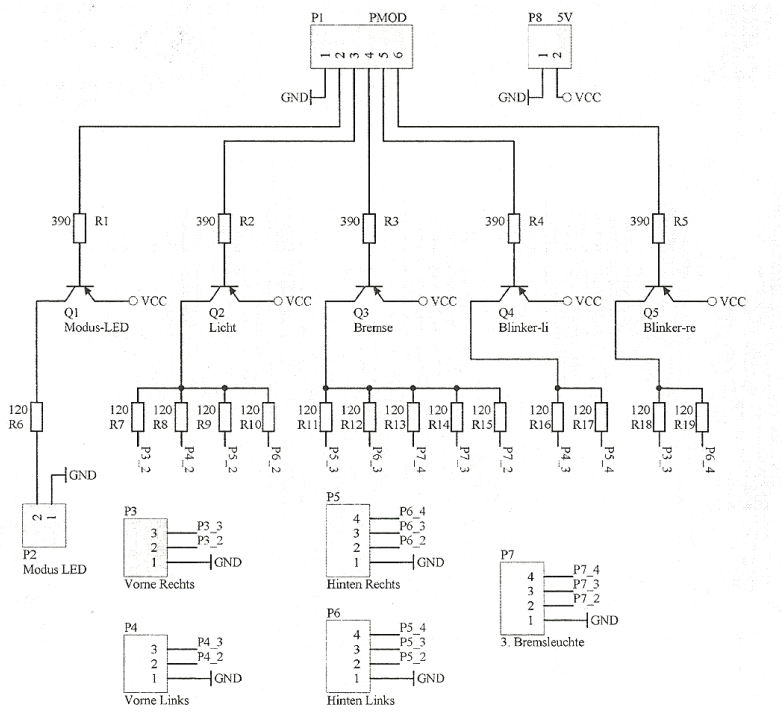
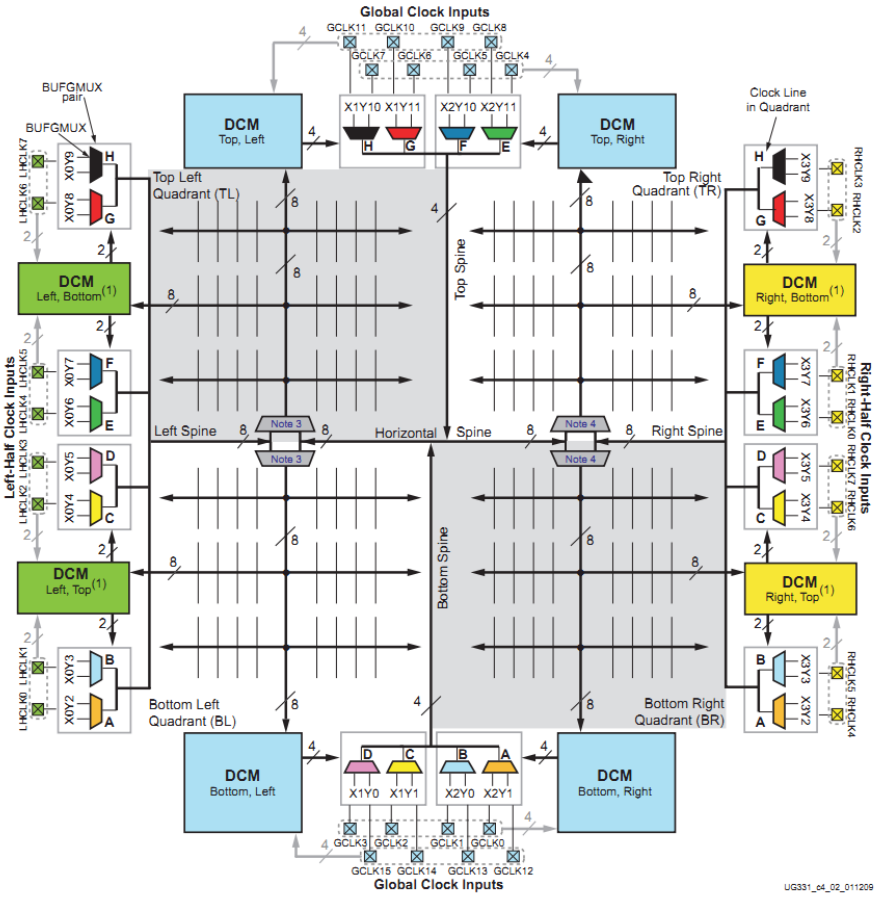


Abbildung E.4 – Das Schaltbild der LED-Platine; über den fünfpoligen Stecker werden die Eingangssignal für die Ansteuerung der einzelnen LEDs vom Steuerungssystem übermittelt.

Clock Ressourcen der Extended Spartan-3A Device Family



Notes:

1. The diagram presents electrical connectivity. The diagram locations do not necessarily match the physical location on the device, although the coordinate locations shown are correct.
2. Number of DCMs and locations of these DCM varies for different device densities. See Table 2-1.
3. See Figure 2-13a, which shows how the eight clock lines are multiplexed on the left-hand side of the device.
4. See Figure 2-13b, which shows how the eight clock lines are multiplexed on the right-hand side of the device.
5. For best direct clock inputs to a particular clock buffer, not a DCM, see Table 2-7.
6. For best direct clock inputs to a particular DCM, not a BUFGMUX, see Chapter 3, "Using Digital Clock Managers (DCMs)."

Abbildung E.5 – Die Anordnung von jeweils zwei Digitalen Clock Managern an einer I/O Bank des FPGAs, lässt deren gemeinsame Verwendung ohne Erzeugung von langen Signalpfaden zu. Dadurch können externe Eingangsfrequenzen innerhalb einer Clockdomain optimal verteilt werden[48].
Anm.: Die Abbildung wurde komplett aus dem „Spartan-3 Generation FPGA User Guide“ übernommen; die *Notes* beziehen sich auf das Dokument, sind hier aber zur Verdeutlichung übernommen worden.

CD: System Generator und EDK Projektdateien, MatLab-, VHDL- und C-Code

Versicherung über Selbstständigkeit

Hiermit versichere ich, dass ich die vorliegende Arbeit im Sinne der Prüfungsordnung nach §24(5) ohne fremde Hilfe selbstständig verfasst und nur die angegebenen Hilfsmittel benutzt habe.

Hamburg, 29. Februar 2012

Ort, Datum

Unterschrift