Bachelorarbeit

Ein System-on-Chip-basiertes Fahrspurführungssystem

Christian Schneider

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

Betreuender Prüfer: Prof. Dr.-Ing. Bernd Schwarz
Zweitgutachter: Prof. Dr.-Ing. Andreas Meisel

Abgegeben am 16.08.2011

Christian Schneider

Thema der Bachelorarbeit
Ein System-on-Chip-basiertes Fahrspurführungssystem

Stichworte
Fahrspurführung, System on Chip, Echtzeit-Videoverarbeitung, Fahrspurerkennung, Pure-Pursuit, Follow-The-Carrot, PD-Regelung, modellgetriebene RTL-Entwicklung, VHDL-Codegenerierung

Kurzzusammenfassung
Diese Arbeit behandelt die Entwicklung einer SoC-basierten Echtzeit-Bildverarbeitungs-plattform, die die Fahrspurerkennung und Spurführung eines Modellfahrzeugs realisiert. Mit einem FPGA wird ein Prozessorsystem konfiguriert, das den Videostrom mit Beschleunigermodulen verarbeitet, die je nach Datenrate mit Pipelinestufen oder mit Multizyklus-Datenpfaden aufgebaut sind. Für Parameter- und Fahrverhaltensstudien wird ein entwickelter MatLab-Simulator verwendet, der die Fahrspurführungsalgorithmen Follow-The-Carrot, Pure-Pursuit und PD-Regelung implementiert und visualisiert. Die Analyse der Vorentwicklungsstufe führt zu Modifikationen der Videopipeline, die eine Pixelmengenreduktion durch Grauwerteskalierung, prozentualer Binarisierungsschwelle und morphologischer Erosion durchführt. Die Hough-Transformation, welche die Fahrspur im Videodatenstrom approximiert, wird um eine dynamische Region-Of-Interest ergänzt. Mit Algorithmic-State-Machines werden die Spurführungsalgorithmen in RTL-Modelle überführt und mit den Simulator-Ergebnissen parametrisiert.

Christian Schneider

Title of the paper
A System-on-Chip based path following system

Keywords
Path following, System on Chip, real-time video processing, lane detection, Pure-Pursuit, Follow-The-Carrot, PD controller, model-driven RTL development, VHDL code generation

Abstract
This thesis deals with the development of a SoC-based real-time video-processing platform, which realizes a lane detection and path following system of an autonomous RC car. Using a FPGA the processor system is configured, which processes the video stream with accelerated peripheral modules with pipelines or multi cyclic data paths, according to their data rate. A developed MatLab simulator, which implements and visualizes the path following algorithms Follow-The-Carrot, Pure-Pursuit and PD control, is used for parameter and behavioral studies. The analysis of the pre-developed system leads to modifications of the video pipeline, which reduces the amount of pixels through rescaling of gray values, percentaged binarizing level and morphologic erosion. A dynamic region of interest supplements the Hough Transformation, which approximates the lane within the video data stream. The RTL models of the path following algorithms are realized as algorithmic state machines and parameterized with the simulator results.

Inhaltsverzeichnis

1 Einleitung

Eingebettete Systeme werden häufig in heterogenen Umgebungen eingesetzt, sodass deren dedizierte Funktionen angepasst auf spezielle Anwendungsfelder entwickelt werden, z.B. in Fahrzeugen, Flugzeugen und in der Medizintechnik. Die systematische Entwicklung, Modellierung und Validierung solcher Systeme wirft in diesen Forschungsgebieten aktuelle Fragen mit hoher Anwendungsrelevanz auf [1] [2].

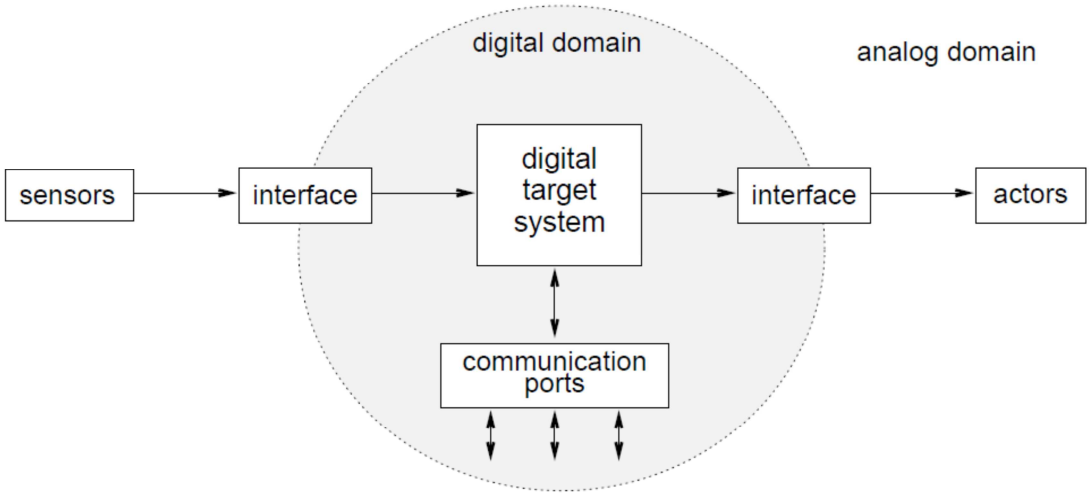

Abbildung 1: Typischer Anwendungsbereich für eingebettete Systeme. Sensoren erfassen digitalisierte Messwerte des technischen Prozesses. Diese Werte gehen z.B. als Regelabweichung in eine Echtzeitberechnung von Stellgrößen für die analoge Aktoren ein [3].

Das in dieser Arbeit konfigurierte Fahrspurführungssystem basiert auf einem System-on-Chip-Design und dient der Erprobung der SoC-Technologie mit Bildverarbeitungsanwendungen im Rahmen des FAUST-Projekts (vgl. Abbildung 2). SoCs vereinigen die Vorzüge von Mikrocontrollern und FPGAs, da z.B. durch die re-konfigurierbaren FPGA-Ressourcen rechenintensive Algorithmen in paralleler Hardware beschleunigt und weniger zeitkritische Aufgaben in Software von einem Mikrocontroller bearbeitet werden. Zudem stellt dieser umfangreiche Softwarebibliotheken zur Bedienung der Peripherie, z.B. zur Kommunikation mit der Außenwelt oder Parametrisierung der Sensoren, bereit und regelt den Austausch von Beschleunigermodulen durch partielle Re-Konfiguration.

Ziel dieser Arbeit ist eine Technologieerprobung einer System-on-Chip-Plattform in den für sie typischen Anwendungen der Signal- und Bildverarbeitung. Für die autonome Fahrspurführung des Fahrzeugs werden mit einer Kamera-FPGA-Kombination die Lageinformationen der Fahrbahnmarkierungen bestimmt. Die Datenmenge des Kameradatenstroms wird mit einer Binarisierung reduziert, um durch eine Hough-Transformation die Fahrspurmarkierung durch Pixelmengenreduktion als Gerade mit den Parametern r und Φ zu approximieren. Mit den Geradenparametern werden die Istgrößen für eine Positionierung des Fahrzeugs bestimmt, damit die Lage des Fahrzeugs zur Fahrbahn mit den Algorithmen Pure-Pursuit oder Follow-The-Carrot oder per Abstandsregelung korrigiert werden kann. Dazu wurde ein Digilent Nexys2 FPGA-Board und eine Sony FCB Kamera in ein RC-Modellfahrzeug integriert (vgl. Kapitel 2.5).

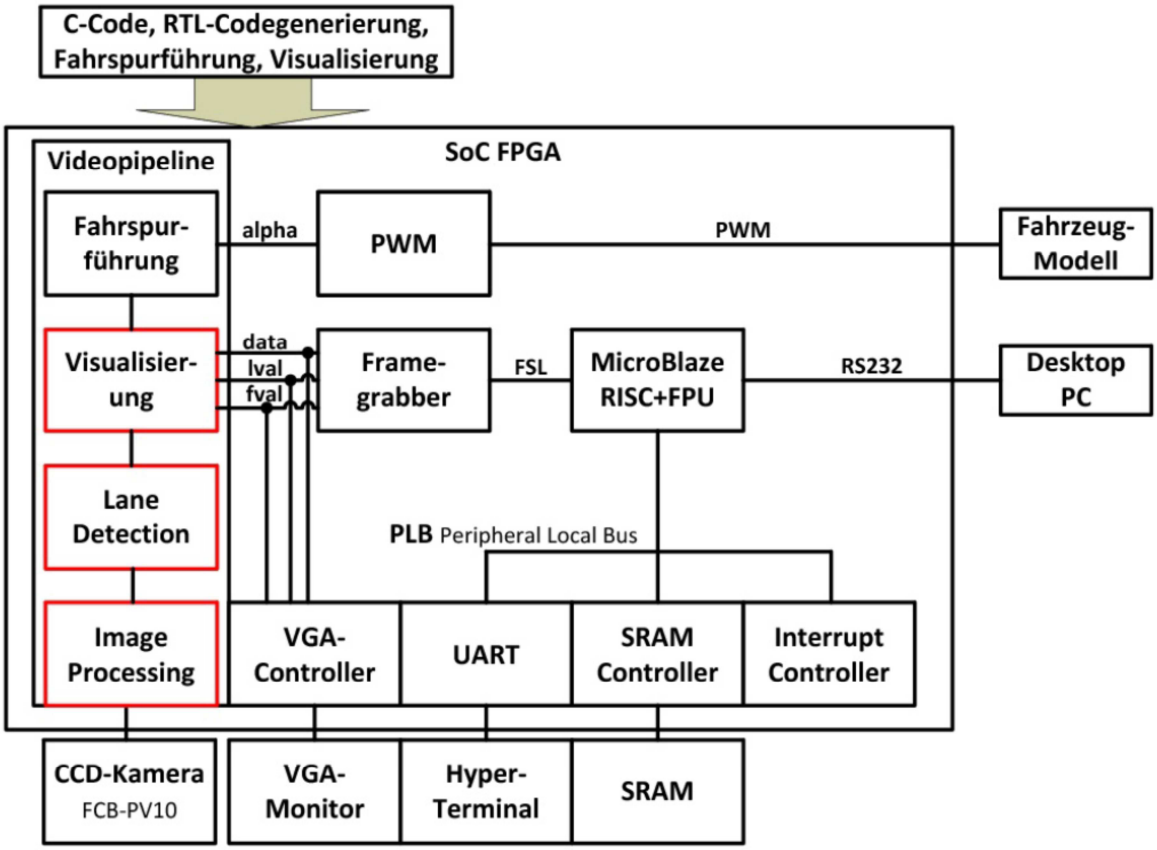

Abbildung 2: SoC-Konfiguration mit Softcore Mikrocontroller, Beschleunigermodulen und Standardschnittstellen. Der Kameradatenstrom wird in der Videopipeline verdichtet, sodass eine Fahrspurerkennung durchführt und aus den gewonnenen Informationen Lenkwinkelstellgrößen über PWM an den Lenkservo des Fahrzeugs übergeben werden können.

Die Schwerpunkte des Projekts zur Entwicklung eines SoC-basierten Fahrspurführungssystems sind:

- Geometrische Konstruktion des Look-Ahead-Points (LAP) für die Fahrspurführung mit Pure-Pursuit und Follow-The-Carrot, sowie des Istabstands des Fahrzeugs zur Fahrspurmarkierung für eine zeitdiskrete Abstandsregelung mit einem PD-Regler.
- MatLab-Simulation und Verifikation dieser Algorithmen zur funktionellen Untersuchung der Fahrspurführung eines Einspur-Fahrzeugmodells.
- Modifikation des Gesamtsystems auf die aus MatLab-Simulation und Funktionsanalyse der Vorentwicklungsstufe abgeleiteten Anforderungen an die Fahrspurführung.
- Modellgetriebene RTL-Modellierung der Istgrößenbestimmung, der Fahrspur-führungsalgorithmen Follow-The-Carrot, Pure-Pursuit und PD-Abstandsregelung.
- Überprüfung und Reduzierung des FPGA-Ressourcenbedarfs der entwickelten RTL-Modelle.

In **Kapitel 2** wird die Ausgangssituation des Entwicklungsprojektes zum SoC-Fahrzeug beschrieben und ein Überblick der Arbeiten gegeben. Es werden die Entwicklungsplattformen und der Ablauf einer modellgetriebenen RTL-Entwicklung dargestellt und der resultierende, strukturelle Systemüberblick vorgestellt.

Kapitel 3 beschreibt Algorithmen zur Fahrspurführung und die mathematischen Zusammenhänge zur Berechnung des anzusteuernden Look-Ahead-Points und des Istabstands des Fahrzeugs zur Fahrspurmarkierung, sowie Approximationsalgorithmen für trigonometrische und Wurzelfunktionen.

Die MatLab-Simulation zur Verhaltensanalyse des mathematischen Fahrzeugmodells sowie die Simulationsergebnisse der Parameterstudien werden in **Kapitel 4** beschrieben.

Auf die Modifikationen der Bildverarbeitungskette zur Verbesserung der Fahrspurerkennung wird in **Kapitel 5** eingegangen. Der Aufbau dieser angepassten Module, sowie der Hintergrund der Optimierung werden vorgestellt.

In **Kapitel 6** wird auf die Umgestaltung der Fahrspurerkennung eingegangen. Es werden die Simulationsergebnisse mit der Parametrisierung der Region-Of-Interest und der Diskretisierung der Hough-Ebene in Verbindung gebracht.

Die RTL-Modellierung ausgewählter Spurführungsmethoden wird in **Kapitel 7** vorgestellt. Neben der Realisierung dieser Algorithmen als Algorithmic State Machines wird das PWM-Modul zur Servo-Ansteuerung beschrieben.

Die Ergebnisse dieser Arbeit bezogen auf Funktion, Verhalten und Ressourcenbedarf werden in **Kapitel 8** präsentiert. Es wird ebenfalls laufende und folgende Erweiterungen des SoC-Systems eingegangen.

2 Methodik, Konzepte und SoC-Plattform

In diesem Abschnitt werden die Analyseergebnisse der Vorentwicklungsstufe des Fahrspurerkennungssystems vorgestellt und die Modifikationen und zusätzlichen Implementierungen genannt (vgl. [4] und [5]). Das resultierende SoC wird vorgestellt, um einen Überblick zur Funktionsweise und zum Zusammenspiel der angepassten und neu entwickelten Module zu geben. Des Weiteren wird die FPGA- und Entwicklungsplattform beschrieben und auf den Konfigurations- und Entwicklungsablauf mit modellgetriebener RTL-Entwicklung eingegangen.

2.1 Funktionsweise der Vorentwicklungsstufe zur Fahrspurerkennung

In dem in [4] entwickelten Fahrspurerkennungssystem wird der perspektivisch verzerrte Kameradatenstrom vom **Sync**-Modul mit Synchronisierungssignalen versehen, die als Enable-Signale in den Folgestufen der Videopipeline verwendet werden, um Übertragungspausen zwischen Zeilen und Bildern zu erkennen (vgl. Abbildung 3).

Der Videodatenstrom wird mit dem **Deserialize**-Modul deserialisiert, um mit zwei 3x3 Pixel großen **Sobel**-Masken die Kantenfilterung zur Pixelmengenreduktion durchzuführen. Zur weiteren Reduktion der Informationsmenge wird das kantengefilterte Bild vom **Binarize**-Modul mit einer festen Grauwertschwelle binarisiert.

Ein Koordinatengenerator generiert mit einem **Pixelzähler** die x- und y- Koordinaten der einkommenden Pixel, welche dann auf die Zugehörigkeit zu einer fest positionierten Region-Of-Interest (ROI) überprüft werden.

Alle in der **ROI** liegenden Pixelkoordinaten werden zur Approximation der Fahrspurmarkierung an das Modul zur **Hough-Transformation** (HT) weitergereicht, welches die Lage der Fahrspur als Gerade in Hesse'scher Normalform mit dem Abstand r und dem Winkel Φ des Lotfußpunktes zum Koordinatenursprung der ROI approximiert.

Die Geradenparameter r und Φ der perspektivisch verzerrten Geraden werden in **Softwareregister** gespeichert und dem **Mikrocontroller** per Interrupt-Flag bekannt gegeben, damit eine Interrupt-Service-Routine die perspektivische Korrektur der approximierten Fahrspurmarkierung durchführt. Aufgrund des Übergangs auf den Mikrocontroller wird bei der gewählten Berechnungsreihenfolge eine Reduktion des FPGA-Ressourcenbedarfs durch Vermeiden eines zusätzlichen Beschleunigermoduls zur projektiven Transformation (PT) erzielt.

Die perspektivisch korrigierten Geradenparameter werden über **Softwareregister** dem **Overlay**-Modul zur Verfügung gestellt, welches die Pixel auf Zugehörigkeit zur Geraden überprüft. Alle Binärpixel werden in das RGB-Format umgewandelt und entsprechend der Zugehörigkeit zur Geraden eingefärbt und an das VGA-Interface weitergereicht. Durch das **Deinterlace**-Modul werden die eingehenden Halbbilder verdoppelt und anschließend vom **VGA-Controller** an die VGA-Schnittstelle des FPGA-Boards übergeben und am Monitor dargestellt.

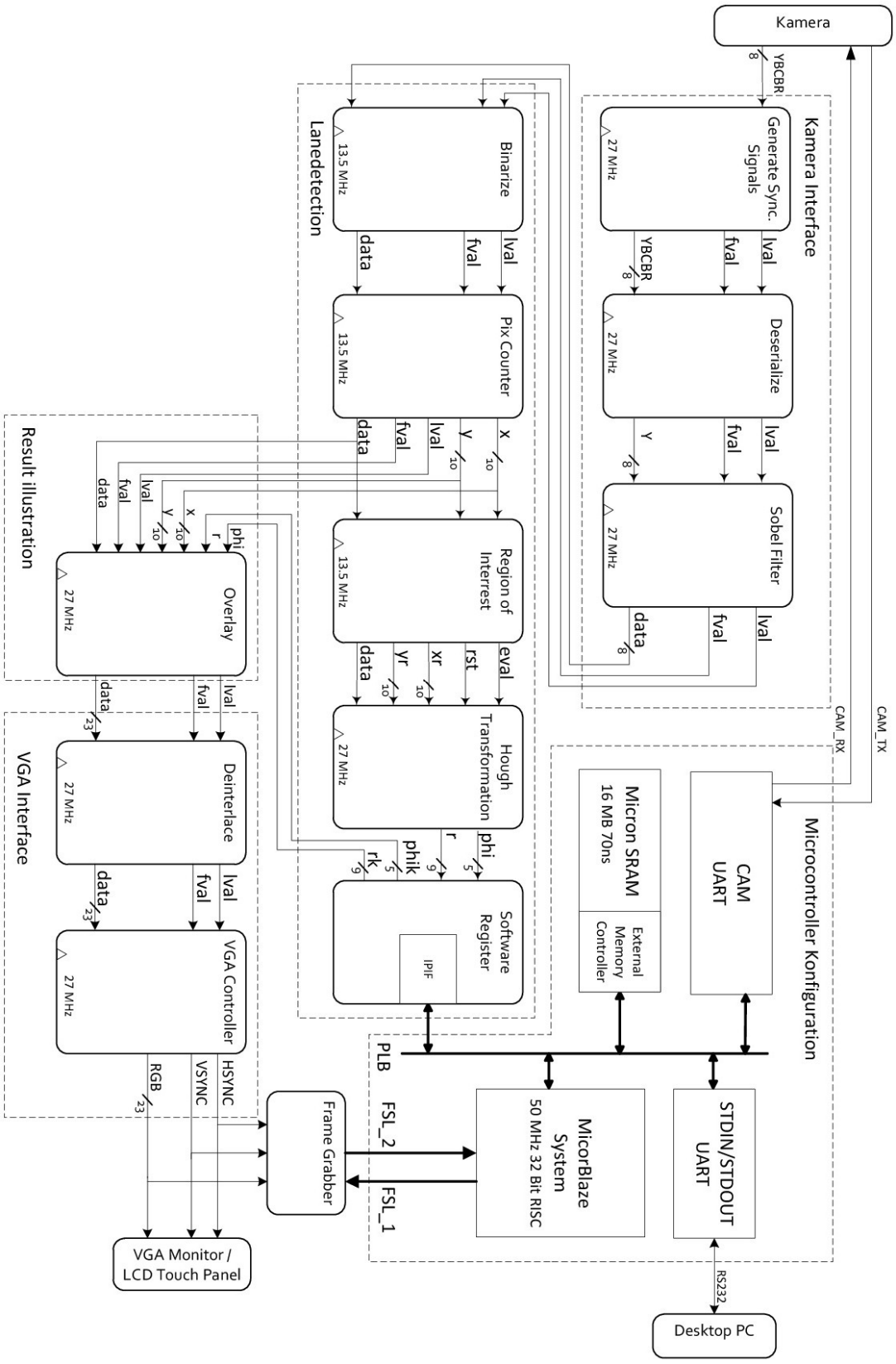

Abbildung 3: Vorentwicklungsstufe der in **[4]** entwickelten Fahrspurerkennung

2.2 Analyse des Fahrspurerkennungssystems

Das zu entwickelnde Fahrspurführungssystem ergänzt die vorhandene System-On-Chip-Konfiguration zur Fahrspurerkennung und wertet die ermittelten Geradenparameter aus, um das Fahrzeug durch Lenkwinkelsollwerte auf einer miniaturisierten Fahrbahn zu führen.

Zur Analyse der Eigenschaften und Anforderungen an die Genauigkeit der Fahrspurerkennung wurden die FPGA-Plattform und die Kamera auf dem Fahrzeug installiert und in dessen Spannungsversorgung integriert. Unter Verwendung der in vorangegangenen Arbeiten entwickelten Module sind erstmals reale Bilder aus Sicht einer auf dem Fahrzeug installierten Kamera extrahiert worden. Aus der Analyse des bestehenden Gesamtsystems nach Abbildung 3 ergaben sich folgende **Feststellungen:**

- Die Fahrzeugrotation führt zu starken Lageänderungen der Fahrspurmarkierungen im perspektivisch verzerrten Bilddatenstrom, sodass in Kurven keine Fahrbahnmarkierung innerhalb der fest positionierten ROI liegt.
- Die von der ROI an die Hough-Transformation (HT) weitergegebenen Bildpunkte führen zu fehlerbehafteten und teilweise nicht mit der Fahrspurmarkierung im Zusammenhang stehenden Geradenparametern.
- Aufgrund der Berechnungsreihenfolge werden fehlerbehaftete Geradenparameter durch die projektive Transformation (PT) in Software weiter verfälscht, sodass sich diese nicht für eine korrekte Spurführung eignen.
- Die feste Grauwertschwelle führt bei wechselndem Lichteinfall zu qualitativ schlechter Binarisierung und damit zu geringen Erkennungsraten.
- Die durch das Sobelfilter entstandenen Kanten führen zu oszillierenden Geraden.

Ausgehend von der Systemanalyse wurden Anforderungen an die Fahrspurerkennung erstellt, um eine gezielte Modifikation vorzunehmen:

- Die Binarisierung des Graubildes soll in Abhängigkeit des Lichteinfalls durchgeführt werden.
- Die Geradenapproximation soll auf einem perspektivisch korrigierten Bild erfolgen.
- Die Geradenparameter sollen in einem metrischen Zusammenhang zur Fahrzeugebene stehen.
- Die ROI soll die Fahrspurmarkierung dynamisch verfolgen.

2.3 Konzeption des Fahrspurführungssystems

Eine Literaturrecherche lieferte das Know-How zu in aktuellen Aufsätzen untersuchten Fahrspurführungsalgorithmen (vgl. Kapitel 3). Die in dieser Arbeit betrachteten Algorithmen beschränken sich auf Follow-The-Carrot, Pure-Pursuit und die Abstandsregelung des Fahrzeugs zur Fahrspurmarkierung.

Es wurde ein Matlab-Simulator entwickelt, um Parameterstudien für die RTL-Modellierung der ausgewählten Fahrspurführungsalgorithmen durchzuführen. Dadurch konnte das Fahrzeugverhalten untersucht werden, ohne Beschädigungen der Hardware und des Fahrzeugs in Kauf zu nehmen. Die MatLab-Implementierung dient als Entwurf für Algorithmic State Machines zur Entwicklung der RTL-Modelle. Des Weiteren wurde die

Arkustangensfunktion zur Berechnung des Lenkwinkels approximiert und auf Genauigkeit untersucht, sowie die Parameter für den entworfenen PD-Regler abgeschätzt.

Die Modulreihenfolge in der Bildverarbeitungspipeline wurde so angepasst, dass die HT zur Geradenapproximation auf perspektivisch korrigierten Bildern durchgeführt wird. Die Position der zur Bereichsreduktion verwendeten ROI wird dynamisch für jedes Bild neu bestimmt, indem eine Rückführung der gewonnenen Geraden zum Tracking der Fahrspurmarkierung benutzt wird.

Zur Vereinfachung der Fehler- und Systemverhaltensanalyse wurde ein Bildzwischenspeicher in die Videoverarbeitungspipeline integriert, mit dem die Korrektheit der erkannten Geraden visuell überprüft werden kann [6].

Um das Schwingen der erkannten Geraden zu verringern wurde die Kantenfilterung durch und ein morphologisches Filter ersetzt, welches durch UND-Verknüpfungen benachbarter Binärpixel zu einer Erosion der Fahrspurmarkierung führt. Die Daten werden zwar nicht so stark reduziert wie mit Sobel - dennoch führt dies zu einer genaueren Geraden-approximation.

Insgesamt wurden folgende zusätzlichen Implementierungen und Modifikationen am vorhandenen System vorgenommen:

- RTL-Modellierung, Test und Codegenerierung
  - der Binarisierung mit einer adaptiven Grauwertschwelle.
  - eines morphologischen Erosionsfilters.
  - ausgewählter Spurführungsalgorithmen.
  - des PWM-Moduls.
  - einer dynamischen ROI.
- Modifikation
  - der HT nach Verkleinerung der Hough-Ebene.
  - des Hardwaremoduls zur Projektiven Transformation (PT).

2.4 Systemüberblick zum entwickelten Fahrspurführungssystem

Das resultierende Gesamtsystem unterteilt sich in die drei Spuren **Bildverarbeitungspipeline, Fahrspurerkennungs- und –führungssystem und Parametrisierungs- und Kommunikationssystem,** welche modular aufgebaut wurden, um die Austauschbarkeit der Komponenten durch vorgegebene Schnittstellen sicherzustellen und einzelne Systemabschnitte überprüfen zu können und Fehler einzugrenzen.

Das Datenlogging, sowie die Parametrisierung des Kamerasensors wird durch die Kombination aus MicroBlaze – Mikrocontroller und dem Frame Grabber Modul durchgeführt, welches den verabeiteten Datenstrom eines Bildes zwischenspeichert und im Bitmap-Format an einen per RS232-Schnittstelle angekoppelten PC überträgt.

Die Bildverarbeitungspipeline besteht aus den Modulen Camera Interface, Image Processing, Result Illustration und VGA Interface und dient der Pixelmengenreduktion, sowie der Ergebnisdarstellung der Fahrspurerkennung.

Im Modul „Camera Interface“ wird der aus Halbbildern bestehende Videodatenstrom mit Synchronisierungssignalen versehen und die Pixelfarbwerte im YCbCr-Format in 8-bit Grauwerte umgewandelt.

Zur Datenreduktion wird eine Grauwertskalierung durchgeführt und das Bild mit einem prozentualen Schwellwert im Modul „Image Processing“ binarisiert. Die zusammenhängenden hellen Flächen im Binärbild werden durch morphologische Operationen auf den Binärpixeln erodiert. Nach der Filterung werden durch einen Pixelzähler x- und y-Koordinaten der Pixel generiert, die zur perspektivischen Korrektur projektiv in die Fahrzeugebene transformiert werden.

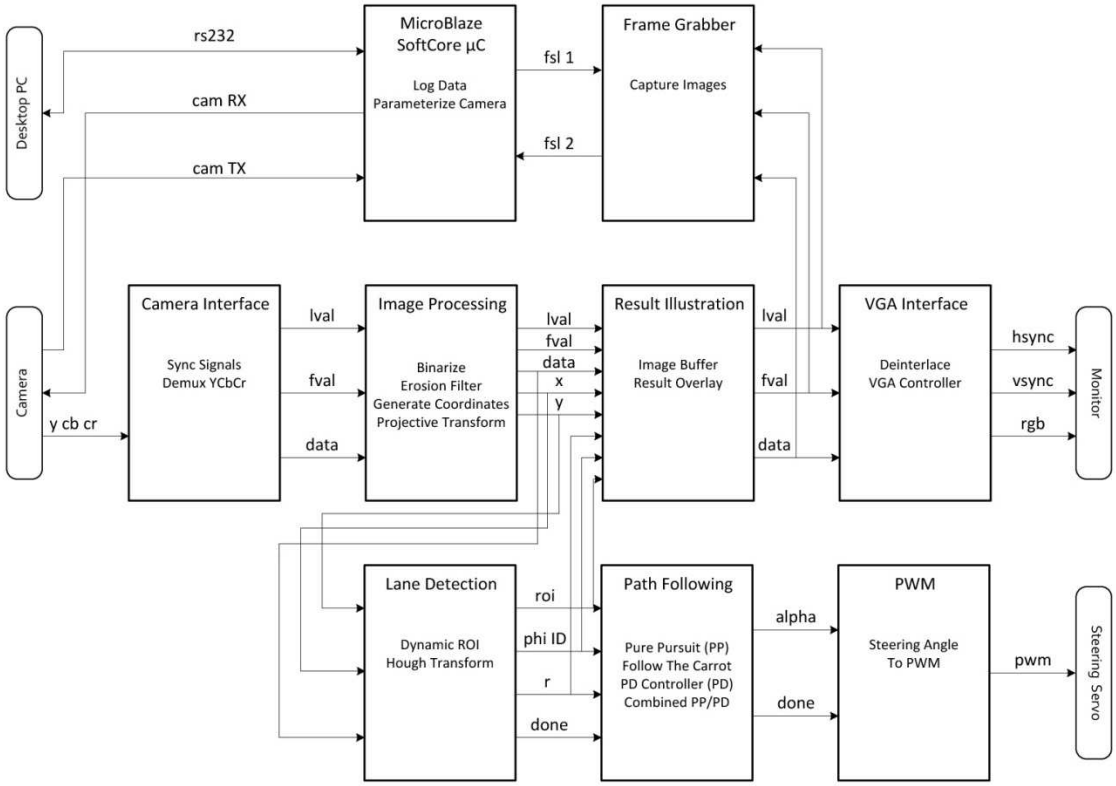

Abbildung 4: Schnittstellen und Module des SoC-Systems: Der Kameradatenstrom wird binarisiert und gefiltert, sodass die Lage der Fahrspur im „Lane Detection“-Modul als Abstand r und Winkel Φ approximiert werden kann. Diese Istgrößen werden zur Berechnung des Lenkwinkels genutzt. Zur Ergebnisanalyse kann der verarbeitete Datenstrom am Monitor angezeigt oder auf einen PC übertragen werden.

Die erkannten Geraden werden im Modul „Result Illustration“ mit dem perspektivisch entzerrten Bild überlagert, indem alle projektiv veränderten Pixel in einen Bildzwischenspeicher eingetragen und mit den Synchronisationssignalen des nächsten Bildes sortiert ausgegeben werden [6].

Zur Anzeige der bearbeiteten Halbbilder werden diese im Modul „VGA Interface“ zeilenverdoppelt und im Format 640 x 480 Pixel durch den VGA-Controller an die Hardware-Schnittstelle des VGA-Anschlusses übergeben und am Monitor darstellt.

Zur Fahrspurerkennung und –führung wird der datenreduzierten Videodatenstrom für eine Geradenapproximation verwendet. Die Positionierung des Fahrzeugs erfolgt mit den durch die Fahrspurführungsalgorithmen berechneten Lenkwinkelbefehle.

Im Modul „Lane Detection“ passieren die Pixel eine dynamische ROI, die feststellt, ob deren Koordinaten im zu interessierenden Bereich liegen und bezieht diese Koordinaten auf ein ROI-Koordinatensystem. Die HT selbst trägt die veränderten Koordinaten in die Hough-Ebene ein und gibt die Parameter der approximierten Geraden in Hesse’scher Normalform als Abstand r und Winkel Φ des Lotfußpunktes aus.

Der Look-Ahead-Point (LAP) und der Fahrzeugabstand zur Fahrspurmarkierung werden im Modul „Path Following“ durch die Geradenparameter und die ROI- Position bestimmt, um die Lenkwinkelsollwerte durch die Spurführungsalgorithmen Follow-The-Carrot, Pure-Pursuit und Combined Pure-Pursuit, sowie durch PD-Regelung zu berechnen, die an das PWM-Modul weitergereicht und in einen PWM-Duty-Cycle übersetzt werden.

2.5 Entwicklungsplattform für SoC-Systeme

2.5.1 SoC-Plattform

Das Digilent Nexys2 Entwicklungsboard dient als Zielhardware für das SoC-System und stellt einen Spartan-3E FPGA und zusätzliche Peripherie wie I/O Schnittstellen und externen Speicher zur Verfügung.

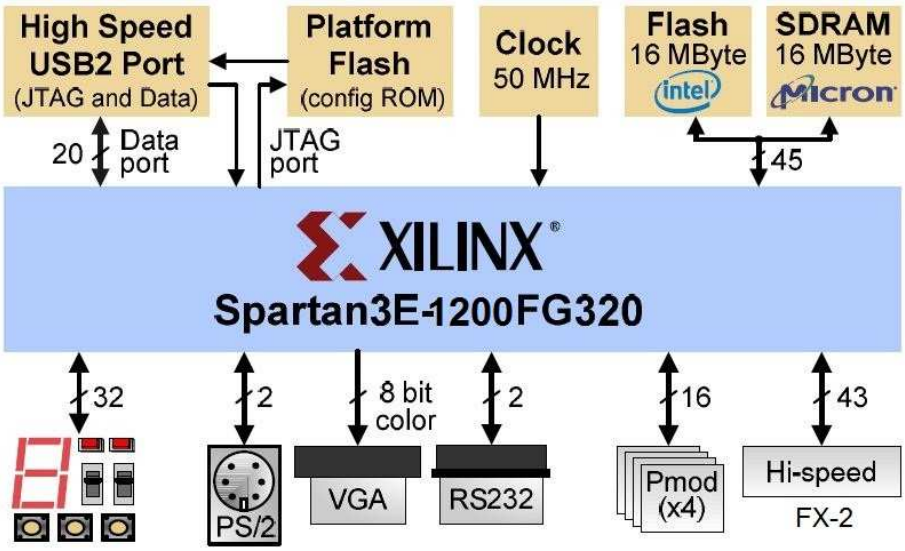

Abbildung 5: Digilent Nexys2 Entwicklungsboard mit dem Spartan-3E-1200K FPGA und Schnittstellen **[7]**.

Das Xilinx Spartan-3E XC3S1200E FPGA bietet die Ressourcen [7]:

- 2168 Configurable Logic Blocks (CLBs): Die CLBs bestehen aus vier miteinander verbundenen Slices, die in Paaren gruppiert sind. Die zwei Slice-Paare sind aufgeteilt auf die linke und rechte Seite des CLBs. Das linke Paar unterstützt sowohl Logik-und Speicher-Funktionen und wird als SLICEM bezeichnet. Das rechte Paar unterstützt nur Logik und wird als SLICEL gekennzeichnet.

- 8672 SLICEMs und SLICELs (Slices): Die Slices enthalten 4-Input LUTs, zwei Daten-Flipflops, sowie Carry- und Kontrolllogik. Es werden zwei Multiplexer zur Bestimmung des Signalpfades genutzt. Der SLICEM hat außerdem noch zwei 16x1 gekennzeichnete RAM-Blocks (RAM16) und zwei 16-Bit Shift-Register (SRL16).

- 17344 Daten-Flip-Flops (D-FF)

- 17344 Look-Up-Tables (LUT): Die Look-Up-Table ist ein RAM-basierter Funktionsgenerator und wird für die Umsetzung von logischen Funktionen genutzt, dabei werden die vorberechneten Werte in eine Wahrheitstabelle eingetragen.

- 250 Input/Output Blocks (IOB): IOBs sind Kontrollblöcke zwischen den I/O Pins und den CLBs. Der Input/Output Block ist eine programmierbare uni- oder bidirektionale Schnittstelle und unterstützt den Tristate-Betrieb. Für eine schnellere Datenübertragung wird ein spezieller Multiplexer eingesetzt, in dem zwei D-FFs die synchronisierten Daten einmal über die steigende und die fallende Taktflanke abwechselnd an den Multiplexer weiterleiten.

- 28 Dedicated Multipliers (DM) (18x18 Bit): Die Multiplizierer berechnen das Produkt aus zwei bis zu 18 Bit breiten Faktoren und liefern ein 36 Bit Ergebnis.

- 28 Block-RAMs (BRAM): Der BRAM ist ein synchroner RAM in Form von 18-kBit Blöcken, die als Single- oder Dual-Port eingesetzt werden. Folgende Lese- und Schreibzugriffe sind während des Dual-Port-Modus vorhanden:

  - Schreiben und lesen von Port A
  - Schreiben und lesen von Port B
  - Datentransfer von Port A nach Port B
  - Datentransfer von Port B nach Port A

- 8 Digital Clock Managers (DCM): Der Digital Clock Manager bietet eine vollständige Kontrolle über Taktfrequenz, Phasenverschiebung und Neigung, die mit einem Delay-Locked Loop (DLL) erreicht wird. Der DLL ist eine digitale Steuerung, die die Feedback-Eigenschaft des Taktsignals zur hohen Präzision nutzt. Folgende Funktionen stehen zur Verfügung:

  - Clock-Skew Elimination: Die Erhöhung der Setup- und Hold-Time führt zu einer Ausgleichung von unterschiedlichen Taktverzögerungen der Clock, da die Ankunftszeiten an verschiedenen Stellen auf dem Chip nicht gleich sind.

  - Frequenz-Synthese: Der DCM kann mehrere Ausgangstaktfrequenzen aus dem ankommenden Taktsignal ableiten. Durch die Multiplikation und / oder Teilung der Frequenz des Eingangstaktsignals mit ganzen Zahlen, die zwischen 0 und 32 sind.

  - Phasenverschiebung: Eine Verschiebung der Phase wird über den ankommenden Eingangstaktsignalen auf die Ausgangstaktsignale vorgenommen.

2.5.2 CCD-Kamera

Die Sony FCB-PV10 umfasst einen ¼“ Interline-Transfer-CCD Bildsensor, der Bilder in einer VGA-Auflösung mit 640x480 Pixeln und einer Bildrate von 29.97 Bildern pro Sekunde über ein FFC direkt an die Pmods des Entwicklungsboards liefert. Das Datenformat entspricht der Form YCbCr 4:2:2. Die Kamera wird für das entwickelte System im 8-Bit Interlace Modus konfiguriert.

Modus	Output	SYNC	Frame Rate	Clock
16 Bit Progressive	YUV 16 Bit 4:2:2	HSYNC/VSYNC	29.97 fps	13.5 MHz
8 Bit Progressive	YUV 8 Bit 4:2:2	HSYNC/VSYNC SAV/EAV	29.97 fps	27 MHz
8 Bit Interlace Scan	YUV 8 Bit 4:2:2	HSYNC/VSYNC SAV/EAV	29.97 fps	27 MHz

Tabelle 1: Digital Image Output Modes der Sony FCB-PV10 Kamera **[8]**.

Die Kamera ist per CN701- und die CN501-Schnittstelle mit der SoC-Plattform verbunden, wobei die CN501-Schnittstelle der Übertragung des Datenstroms, der Synchronisationssignale und des Taktsignals dient und die CN701-Schnittstelle zur Versorgung mit der Betriebsspannung von 6 Volt bis 12 Volt und Parametrisierung über das VISCA-Protokoll genutzt wird [8].

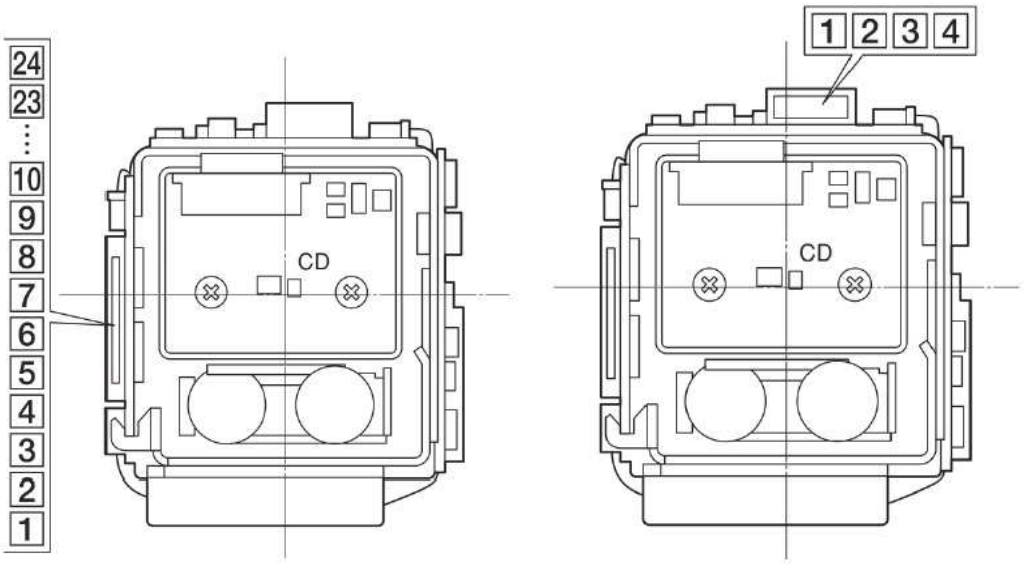

Abbildung 6: CN501 24 FFC Pin Connector (links) und CN701 4 Pin Connector (rechts) **[8]**. Pin-Belegung: vgl. Anhang D.

2.5.1 SoC-Konfiguration und modellgetriebene RTL-Entwicklung mit automatischer VHDL-Code-Generierung

Die Konfiguration des SoC-Systems wurde aufbauend auf den Spezifikationen des Entwicklungsboards mit dem Base System Builder Wizard des Xilinx Platform Studios generiert (vgl. Abbildung 7 und Anhang C). Zur Anbindung der Videoverarbeitungspipeline und der Fahrspurerkennung und -führung wurden diese in

einem Beschleunigermodul zusammengefasst und über den Create-and-Import-Peripheral-Wizard an den PLB-Bus des MicroBlaze-Systems angebunden (vgl. Abbildung 8).

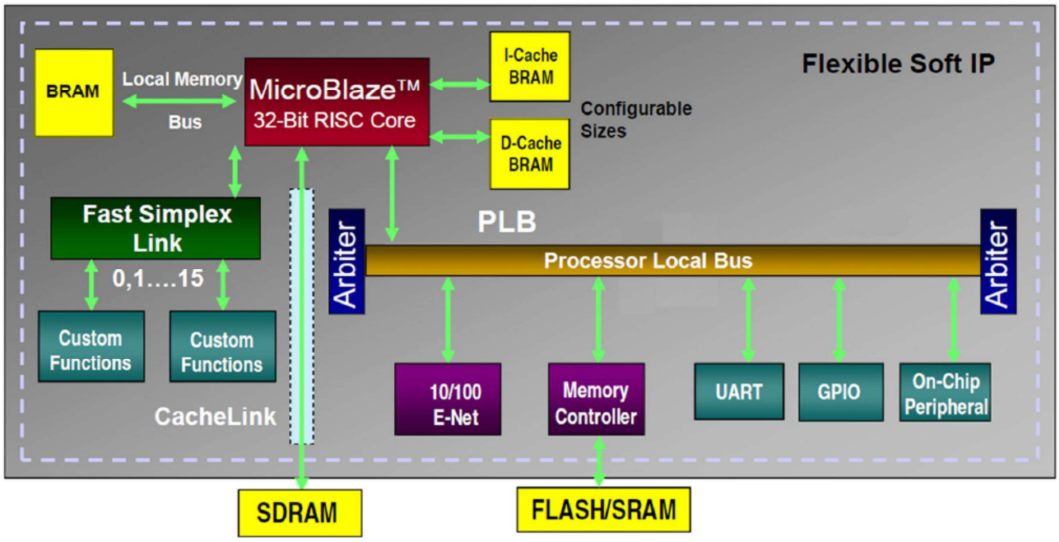

Abbildung 7: Konfiguration eines MicroBlaze-basierten Mikrocontroller-Systems **[9]**.

Komponenten dieses Beschleunigers, z.B. die Hough-Transformation, die projektive Transformation und das Fahrspurführungssystem wurden mit dem System-Generator for DSP entwickelt [10]. Durch den modellgetriebenen Ansatz wird die Umsetzung der Fahrspurführungsalgorithmen in ein RTL-Modul entscheidend verkürzt. Der System Generator vereinfacht die Modellierung von DSP-Systemen, da diese Anwendungen als Modell in Simulink mithilfe des Xilinx Blockset erstellt und dessen Funktion durch Softwaresimulation vorab überprüft werden kann. Aus den entstandenen RTL-Verhaltensmodellen werden automatisch VHDL-Code generiert oder zielplattformspezifische Xilinx IP Cores erstellt. [10].

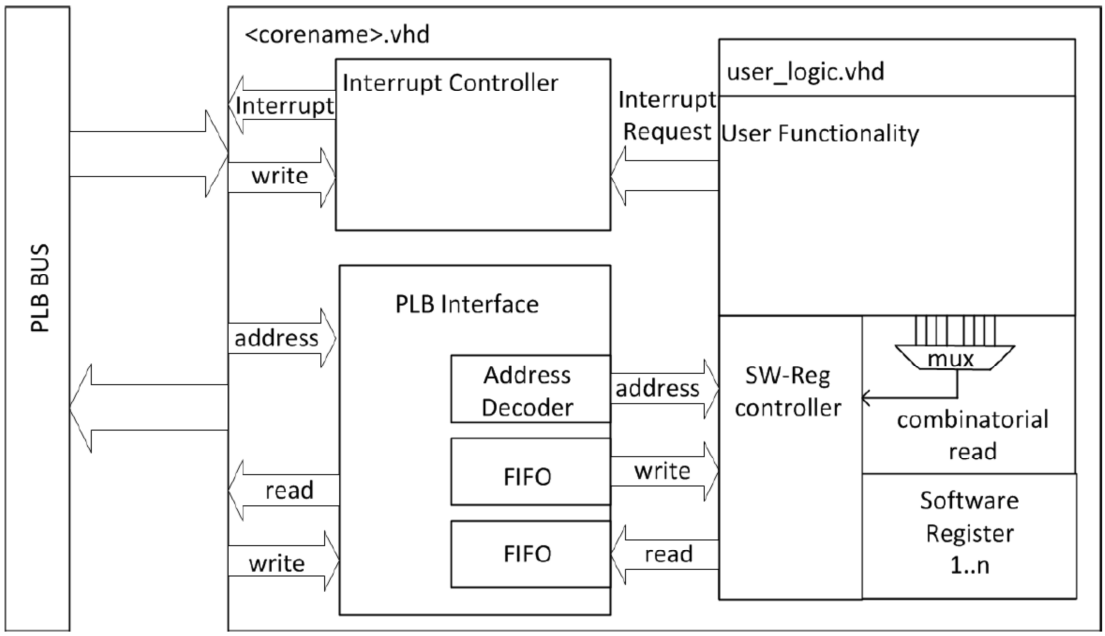

Abbildung 8: Beschleunigermodul mit Softwareregistern zum getakteten Schreiben und kombinatorischen Lesen **[4]**.

Der System Generator stellt in der Simulink Library Funktionsblöcke zur Signalfilterung (z.B. FIR, FFT), Arithmetik-Blöcke, Speicherblöcke (z.B. FIFO, RAM, ROM) und Logikelemente zur Verfügung. Der Simulink-Blocksatz benutzt für die Simulationen eine 64-Bit Fließkommadarstellung während die System Generator Funktionsblöcke eine durch den Entwickler im Q-Format festgelegte Festkommadarstellung verwenden [11], weshalb an den Schnittstellen zwischen diesen Blöcken eine Typkonversion durch Gateway IN/Out Blöcke ausgeführt wird.

Abbildung 9: Gateway In/Out Blöcke zur Typkonversion **[4]**.

Mit Up- und Downsample Blöcken lassen sich Systeme mit verschiedenen Taktfrequenzen modellieren, um z.B. vorhandene Hardwareressourcen durch Mehrfachverwendung höher auszulasten (vgl. Multiratensysteme [5]).

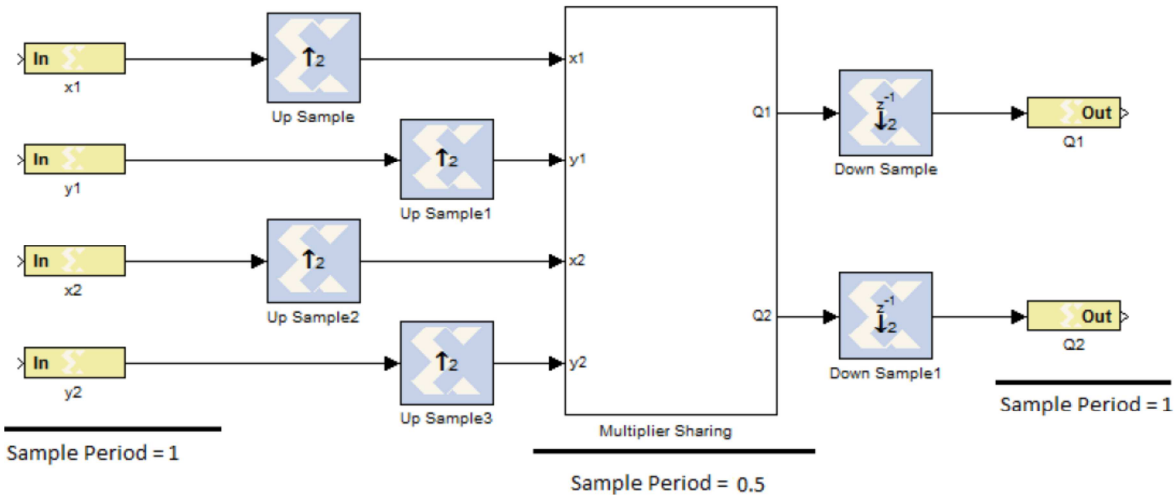

Abbildung 10: Beispiel für ein Multiratensystem: Der Block „Multiplier Sharing“ arbeitet mit der doppelten Frequenz der Eingangssignale **[4]**.

Für die Simulation eines Modells mit benutzergenerierten Eingangssignalen werden mit dem Block „From Workspace“ zeitskalierte Daten aus dem MatLab Workspace gelesen, die zuvor in einem zweidimensionalen Spaltenvektor aufgestellt werden. In der ersten Spalte wird der Simulationszeitpunkt und in der zweiten der Wert abgelegt. Die Ausgangswerte der Simulation werden mit dem „To Workspace“ Block zurückgeschrieben. Zur weiteren Funktionsüberprüfung werden die entstandenen Module in einer Hardware Co-Simulation mit dem Simulink-Modell gleichzeitig simuliert und auf der Zielhardware ausgeführt.

3 Algorithmen der Fahrspurführung

Dieses Kapitel behandelt ausgewählte Algorithmen zur Spurführung, die in aktuellen Aufsätzen zur Erforschung von autonome Fahrzeugen und Robotern ausführlich diskutiert wurden [12], [13], [14], [15], [16] (vgl. Kapitel 3.4).

Viele dieser Algorithmen basieren auf der Ansteuerung eines Zielpunktes, dem Look-Ahead-Point (LAP) oder auch Goal-Points (vgl. Kapitel 3.3) [14]. Der LAP wird zur Steuerung von nicht holonomisch bewegenden Fahrzeugen verwendet und befindet sich in einer definierten Entfernung, der Look-Ahead-Distance (LAD), im Vorausbereich. Die nicht holonomische Eigenschaft von Fahrzeugen besagt, dass sich das autonome Fahrzeug nicht auf der Stelle drehen kann und somit im Gegensatz zu sich holonomisch bewegenden Fahrzeugen, wie z.B. raupenähnlichen Fahrzeugen, einen Wendekreis besitzt [12]. Die LAD kann entweder fest vorgegeben oder in Abhängigkeit von der Geschwindigkeit oder dem Fehler zum Sollpfad bestimmt werden [13].

Auch die Regelung des Fahrzeugabstands zu einer erkannten Fahrspurmarkierung wurde untersucht, implementiert und für brauchbar gehalten. Im Falle der Abstandsregelung wird der Istabstand als Regelgröße mit der Vorgabe des Sollabstands als Führungsgröße verglichen und aus dieser Regelabweichung die Lenkwinkelstellgröße bestimmt (vgl. Kapitel 3.3).

3.1 Koordinatensysteme für die Spurführung

Für die Bestimmung der Fahrzeuglage, die Konstruktion der Positionierungsgrößen, sowie die Implementierung der Fahrspuralgorithmen werden die folgenden Koordinatensysteme in Zusammenhang gebracht:

- (X‘, Y‘) Bildkoordinatensystem des perspektivisch verzerrten Kameradatenstroms.
- (X_P, Y_P) Bildkoordinatensystem des projektiv transformierten Datenstroms.
- (X_{ROI}, Y_{ROI}) Koordinatensystem der Region-Of-Interest.
- (X_V, Y_V) metrisches Fahrzeugkoordinatensystem.
- (X_H, Y_H) Hilfskoordinatensystem zur Istwerte-Konstruktion für die Fahrspurführung.

Das Koordinatensystem des Kameradatenstroms liegt in der Bildebene, deren Pixel aufgrund der Positionierung der Kamera auf dem Fahrzeug perspektivisch verzerrt sind. Die Koordinaten dieser Punkte werden für die weiteren Berechnungen zur Fahrspurführung korrigiert in ein Koordinatensystem des projektiv entzerrten Bildes (X_P, Y_P) übertragen, welches einen Ausschnitt der Fahrzeugebene beschreibt (vgl. Abbildung 13).

Das (X_P, Y_P) - Koordinatensystem ist mit dem Fahrzeugkoordinatensystem (X_V, Y_V) metrisch verknüpft: Durch die Kalibrierung der Ausgleichsmatrix hat ein 1 Pixel des Bildes eine Dimension von 5 mm x 5 mm und perspektivisch korrigierte Koordinaten in der Fahrzeugebene (vgl. Kapitel 5).

Im Koordinatensystem des entzerrten Bildes (X_P, Y_P) liegt auch das (X_{ROI}, Y_{ROI}) - Koordinatensystem der ROI (vgl. Abbildung 12), welches das Wertebereich des (X_P, Y_P) -

Koordinatensystems zur Datenreduktion verkleinert, um den Hardwarebedarf für die diskrete Hough-Ebene im Hough-Transformations-Modul gering zu halten (vgl. Kapitel 6).

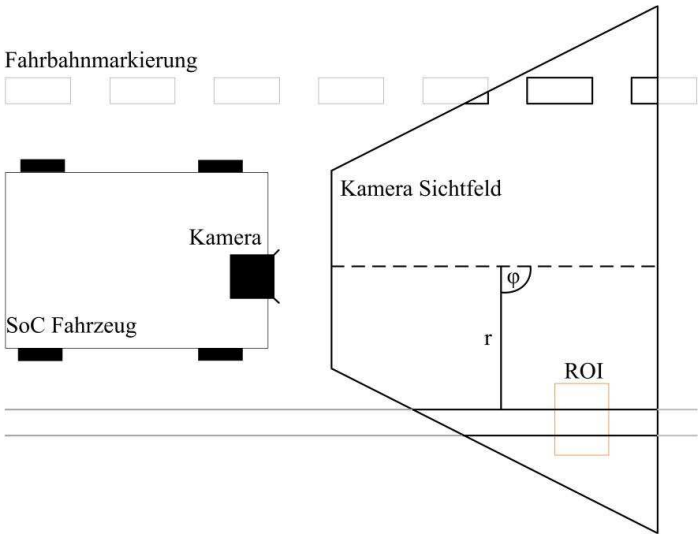

Abbildung 11: Fahrzeugaufsicht. Der Winkel **Φ** und Abstand r der Geraden wird in Bezug auf das ROI-Koordinatensystem **(X_{ROI}, Y_{ROI})** ermittelt und in das Fahrzeugkoordinatensystem **(X_V, Y_V)** übersetzt.

Das Fahrzeugkoordinatensystem (X_V, Y_V) wird zur Konstruktion des LAPs und des Istabstandes zur Fahrspurmarkierung genutzt. Hierbei liegt Y_V auf der Längsachse des Fahrzeugs und zeigt in dessen Vorausbereich. X_V liegt auf der Hinterachse des Fahrzeugs und zeigt auf dessen in Fahrtrichtung liegende rechte Seite, sodass der Schnittpunkt von Translation und Rotation des Fahrzeugs den Koordinatenursprung bildet.

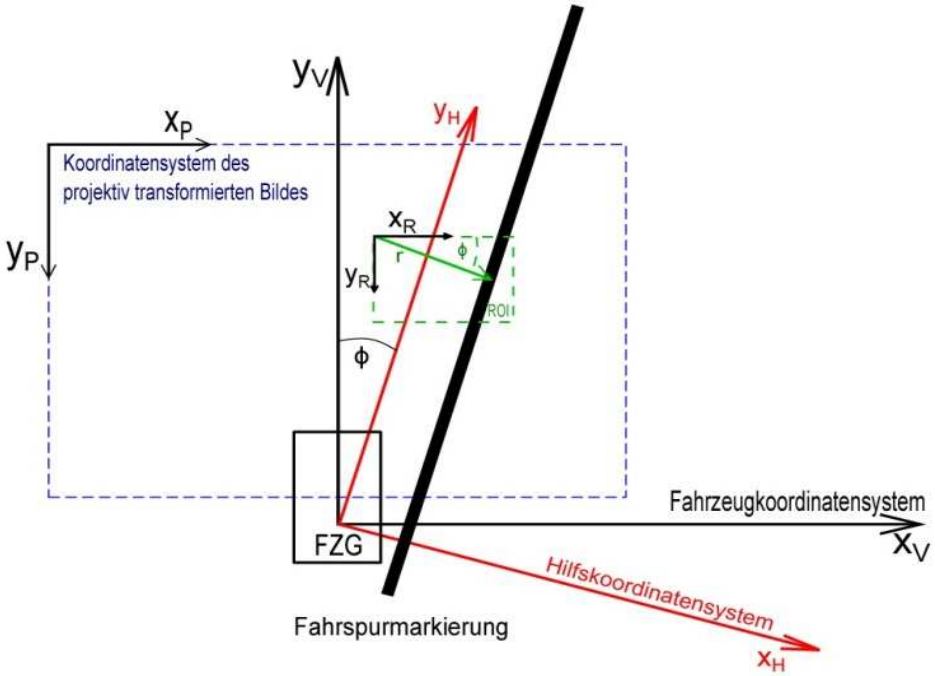

Abbildung 12: Verknüpfung der Koordinatensysteme. Das Bildkoordinatensystem des projektiv transformierten Bildes (X_P, Y_P) ist metrisch mit dem Fahrzeugkoordinatensystem (X_V, Y_V) verknüpft, sodass die Parameter der erkannten Geraden im ROI-Koordinatensystem (X_{ROI}, Y_{ROI}) in das Fahrzeugkoordinatensystem übertragen und in dessen Hilfskoordinatensystem (X_H, Y_H) ausgewertet werden können.

Durch die Verwendung des Hilfskoordinatensystems, dessen Y-Achse parallel zur Fahrspurmarkierung liegt und in den Fahrzeugvorausbereich zeigt, kann bei der Konstruktion der Positionierungsgrößen auf aufwendige mathematische Operationen verzichtet und somit die Berechnung durch Multiplikationen mit in Tabellen gespeicherten Sinus/Cosinus-Werten und Addition/Subtraktion realisiert werden.

3.2 Projektive Transformation zur perspektivischen Entzerrung

Die Punkte in der Kamerabildebene sind aufgrund der Kameraperspektive in einer anderen Ebene angeordnet als die Punkte in der Fahrzeugebene und werden zur Entzerrung der perspektivischen Verzeichnung durch eine projektive Transformation (PT) von der Bildebene in die Fahrzeugebene projiziert.

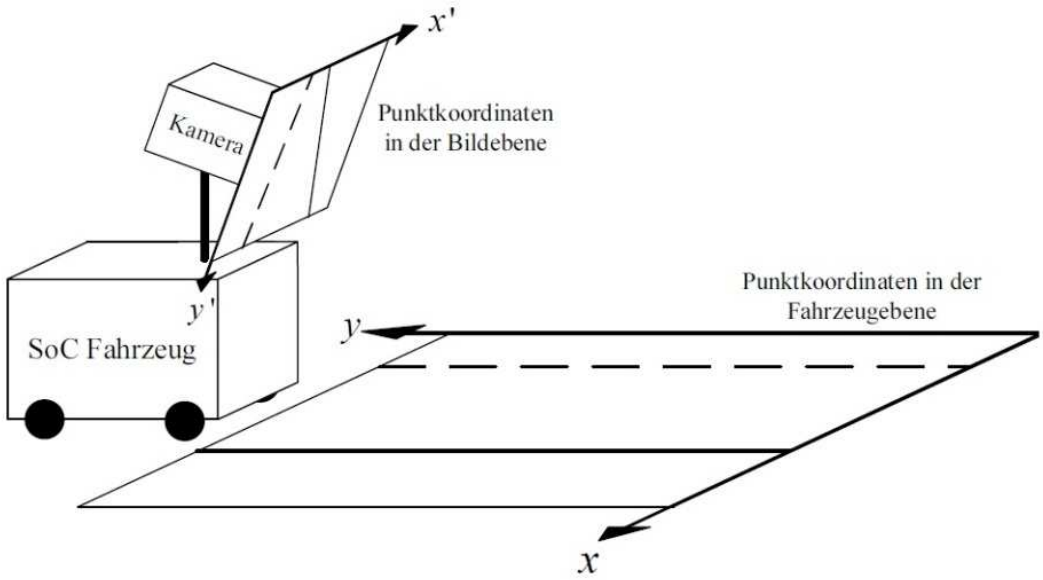

Abbildung 13: Projektion der Kamerabildpunkte in die Fahrzeugebene. Der Kameradatenstrom wird in das Koordinatensystem des perspektivisch entzerrten Bildes projiziert, welches zur Darstellung mit einem Bildkoordinatensystem vereinigt wurde.

Eine projektive Transformation der Ebene beschreibt eine umkehrbare Abbildung h von einer Projektionsebene π' in eine Projektionsebene π und ist eine lineare Transformation mit homogenen dreidimensionalen Vektoren und einer nichtsingulären 3x3-Matrix (vgl. Abbildung 14) [5].

$$\vec{p}_h = H\overrightarrow{p'_h} \leftrightarrow \begin{pmatrix} x_h \\ y_h \\ 1 \end{pmatrix} = \begin{pmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{23} & h_{33} \end{pmatrix} \cdot \begin{pmatrix} x'_h \\ y'_h \\ 1 \end{pmatrix} \qquad (1)$$

Die Transformationsmatrix H kann durch Multiplikation eines Skalierungsfaktors $s \neq 0$ verändert werden ohne die Projektion zu verändern und wird durch Division aller Matrixelemente durch den Projektionsfaktor h_{33} zur Matrix B mit acht unbekannten Elementen [5].

$$\vec{p}_h = B\overrightarrow{p'_h} \leftrightarrow \begin{pmatrix} x_h \\ y_h \\ 1 \end{pmatrix} = \begin{pmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{23} & 1 \end{pmatrix} \cdot \begin{pmatrix} x'_h \\ y'_h \\ 1 \end{pmatrix}, mit\ b_{ij} = \frac{h_{ij}}{h_{33}} \qquad (2)$$

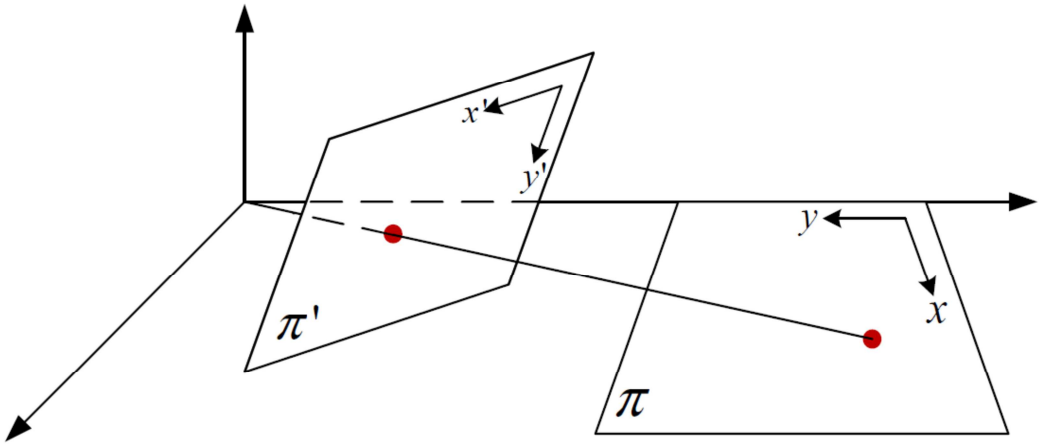

Abbildung 14: Projektive Transformation eines Punktes. Eine Projektion bildet Punkte in der Ebene $\boldsymbol{\pi}'$ auf Punkte in der Ebene π unter Erhalt projektiv äquivalenter Bildstrukturen ab.

Die kartesischen Koordinaten (x_k; y_k) ergeben sich mit Gleichung (1) und dem Skalierungsfaktor s = 1 aus den homogenen Koordinaten (x_h; y_h) [5].

$$x_k = \frac{x_h}{1} = \frac{b_{11}x'_h + b_{12}y'_h + b_{13}}{b_{31}x'_h + b_{32}y'_h + 1} \tag{3}$$

$$y_k = \frac{y_h}{1} = \frac{b_{21}x'_h + b_{22}y'_h + b_{23}}{b_{31}x'_h + b_{32}y'_h + 1} \tag{4}$$

Unter Verwendung dieser Gleichungen wird für jeden Punkt in der Bildebene der in die Fahrzeugebene projizierte Punkt bestimmt.

3.3 Bestimmung der Fahrzeuglage und Berechnung der Positionierungsgrößen

Für die Fahrspurführung wird die Lage des Fahrzeugs zur Fahrspur bestimmt. Betrachtet wird dabei die Lage einer Fahrspurapproximation in der Fahrzeugebene, bzw. im Fahrzeugkoordinatensystem (X_V, Y_V), um den Look-Ahead-Point (LAP) und den Istabstand zur Fahrspurmarkierung für die Steuer- und Regelalgorithmen nach Kapitel 3.3 zu berechnen.

Die Konstruktion des LAP (vgl. Abbildung 15) für die Spurführung nach den Methoden Follow-The-Carrot und Pure-Pursuit basiert den Eingangsgrößen:

- Lage der Fahrspurmarkierung
- Position der Region-Of-Interest (ROI)
- Sollabstand zur Fahrspurmarkierung
- Look-Ahead-Distance (LAD)

Der Abstand r und Winkel Φ der als Geraden approximierten Fahrspurmarkierung beziehen sich auf den Koordinatenursprung der ROI, deren Koordinaten in die Konstruktion des LAP einfließen. Der Offset d definiert den Sollabstand der Fahrzeuglängsachse zur erkannten Fahrbahnmarkierung.

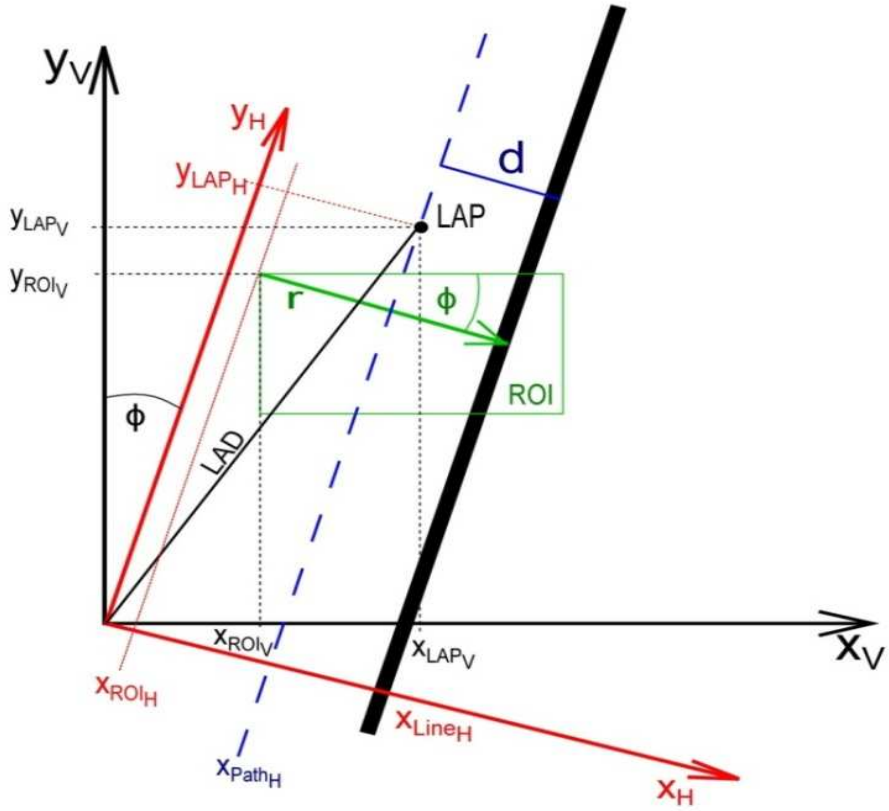

Abbildung 15: Konstruktion des LAP im Hilfskoordinatensystem (Xh, Yh),das durch Drehung des Fahrzeugkoordinatensystems (Xv, Yv) um den Geradenwinkel **Φ** entsteht. Die x-Koordinate in (Xh, Yh) ergibt sich aus der Differenz des x-Abstands der Geraden (x_{ROI} + r) und dem Offset d. Nach Berechnung der y-Koordinate über die LAD und x_{LAP} wird der LAP zurück in das Fahrzeugkoordinatensystem rotiert.

Zur geometrischen Konstruktion des LAP werden die Koordinaten der ROI aus dem Fahrzeugkoordinatensystem nach „rechts“ um den Winkel Φ der erkannten Geraden in das Hilfskoordinatensystem gedreht:

$$\begin{pmatrix} x_{ROI_H} \\ y_{ROI_H} \end{pmatrix} = \begin{pmatrix} \cos(\Phi) & \sin(\Phi) \\ -\sin(\Phi) & \cos(\Phi) \end{pmatrix} \cdot \begin{pmatrix} x_{ROI_V} \\ y_{ROI_V} \end{pmatrix} \quad (5)$$

Die x-Komponente errechnet sich aus der x-Position der ROI, dem Abstand der erkannten Geraden und dem Sollabstand d. Für x_{LAP} gilt im Hilfskoordinatensystem:

$$x_{LAP_H} = x_{ROI_H} + r - d \quad (6)$$

Mit der vorgegebenen LAD und x_{LAP} kann y_{LAP} bestimmt werden

$$y_{LAP_H} = \sqrt{LAD^2 - x_{LAP_H}^2} \quad (7)$$

und beide Koordinaten zurück in das Fahrzeugkoordinatensystem gedreht werden:

$$\begin{pmatrix} x_{LAP_V} \\ y_{LAP_V} \end{pmatrix} = \begin{pmatrix} \cos(\Phi) & -\sin(\Phi) \\ \sin(\Phi) & \cos(\Phi) \end{pmatrix} \cdot \begin{pmatrix} x_{LAP_H} \\ y_{LAP_H} \end{pmatrix} \quad (8)$$

Für der Berechnung des Istabstandes als Regelgröße für eine Abstandsregelung wird die Regelabweichung e ist genutzt, welche genau dann 0 ist, wenn das Fahrzeug parallel im Sollabstand d zur Fahrbahnmarkierung steht. Mit der Führungsgröße w = d gilt:

$$e = w - y = w - (x_{ROI_H} + r) \quad (9)$$

Durch einen Variablenvergleich ergibt sich der Istabstand y der Fahrzeuglängsachse zur Fahrbahnmarkierung als:

$$y = x_{ROI_H} + r \quad (10)$$

Störungen der Regelstrecke des Fahrzeugmodells durch Schlupf der Räder und Verzögerung der Lenkwinkelregelung wird hier nicht berücksichtigt.

3.4 State-of-the-Art-Algorithmen zur Fahrspurführung

Die in aktuellen Aufsätzen häufig genutzten Algorithmen zur Fahrspurführung können in drei Kategorien unterteilt werden [12], [13], [14] , [15], [16]:

I. Algorithmen, deren Lenkwinkelbefehle auf die Ansteuerung eines Zielpunktes basieren (z.B. Pure-Pursuit)
II. Algorithmen, deren Lenkbefehle aus einer Abweichung einer Sollgröße heraus entstehen (z.B. Fuzzy-basierte Regelung)
III. Hybride Algorithmen als Mischform der obigen Algorithmen (z.B. Combined Pure-Pursuit)

Die bekanntesten Methoden zur Steuerung nach Kategorie I sind Follow-The-Carrot und Pure-Pursuit [13]. Aus den Fahrzeuglageinformationen wird ein Ansteuerpunkt konstruiert und der anzulegende Lenkwinkel z.B. wie in Pure-Pursuit auf Basis eines kinematischen Einspur-Fahrzeugmodells berechnet (vgl. Abbildung 18).

Zur Steuerung des Fahrzeugs über eine Abstandsregelung nach Kategorie II werden PD- und PID– Regler, aber auch Neuronale Netze, Vector Support Machines, welche z.B. im softwarebasierten FAUST Projekt auf dem Fahrzeug Nebula verwendet, oder Fuzzy-Regler eingesetzt [17].

In Kategorie III werden Algorithmen der Kategorie I um die Eigenschaften von Kategorie II – Algorithmen ergänzt. In Combined Pure-Pursuit wird z.B. der Lenkwinkel aus dem Mittelwert der aus Pure-Pursuit und PD-Regelung resultierenden Stellgröße gebildet [15]. In Adaptive Pure-Pursuit wird die Look-Ahead-Distance , die Einfluss auf die Berechnung des Ansteuerpunkts hat, durch einen Fuzzy-Regler in Abhängigkeit von der aktuellen Geschwindigkeit und dem aktuellen Fehler zum geplanten Pfad bestimmt [13]. Die Berechnung des Lenkwinkelsollwerts erfolgt weiterhin mittels Pure-Pursuit.

3.4.1 Follow-The-Carrot

Als Eingangsgröße wird ein aus den Lageinformationen konstruierter Zielpunkt genutzt, der Look-Ahead-Point (LAP). Das Prinzip dieser Steuerung basiert auf der sprichwörtlichen Karotte an einer Angel, welche die Look-Ahead-Distance (LAD)

darstellt, die einem Esel in der Richtung vor die Nase gehalten wird, in die er sich drehen soll.

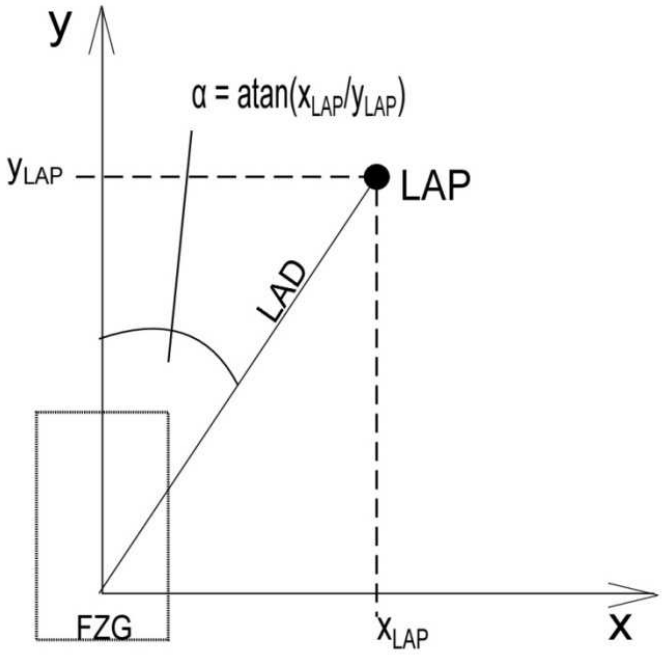

Abbildung 16: Lenkwinkelstrategie der Follow-The-Carrot-Methode. Der Winkel zwischen der Fahrzeugvorauslinie (y-Achse) und LAP entspricht dem einzustellenden Lenkwinkel α.

Der Orientierungsfehler zur Fahrspurmarkierung ergibt sich aus dem Winkel α zwischen der Fahrzeuglängsachse und dem LAP und wird direkt als Lenkwinkelstellgröße genutzt, um das Fahrzeug in dessen Richtung zu steuern:

$$\alpha = tan^{-1}\left({}^{x_{LAP}}/_{y_{LAP}}\right) \qquad (11)$$

Follow-The-Carrot zeichnet sich durch die einfache Berechnung des Lenkwinkels aus, bringt jedoch Nachteile mit sich: Durch die Vernachlässigung der Fahrzeugkinematik und das direkte Ansteuern des Zielpunktes schneidet das zu steuernde Fahrzeug enge Kurven und Kurvenkombinationen. In der Erprobungsphase wurde mit Simulationen und Fahrversuchen ebenfalls eine lange Einschwingphase festgestellt, wenn z.B. von Kurven in Geraden eingebogen wurde (vgl. Kapitel 4.3). Das Aufschwingen und das Auspendeln lassen sich insbesondere auf zu hohe Geschwindigkeiten zurückführen.

3.4.2 Pure-Pursuit

Das Verfahren nach Pure-Pursuit bildet die Grundlage weiterer Algorithmen zur Spurführung [13]. Der Name Pure-Pursuit entstand aus der Vorstellung, dass ein Fahrzeug einen in der Look-Ahead-Distance im Voraus liegenden Look-Ahead-Point auf dem erkannten Pfad verfolgt und stammt aus dem militärischen Bereich [14].

Die Methode ähnelt der Abwehr von Luftfahrzeugen und Flugkörpern, in der Raketen und Lenkflugkörper das zu zerstörende Ziel verfolgen. Dabei ist der Geschwindigkeitsvektor des abwehrenden Flugkörpers immer auf die aktuelle Position des zu zerstörenden Ziels ausgerichtet.

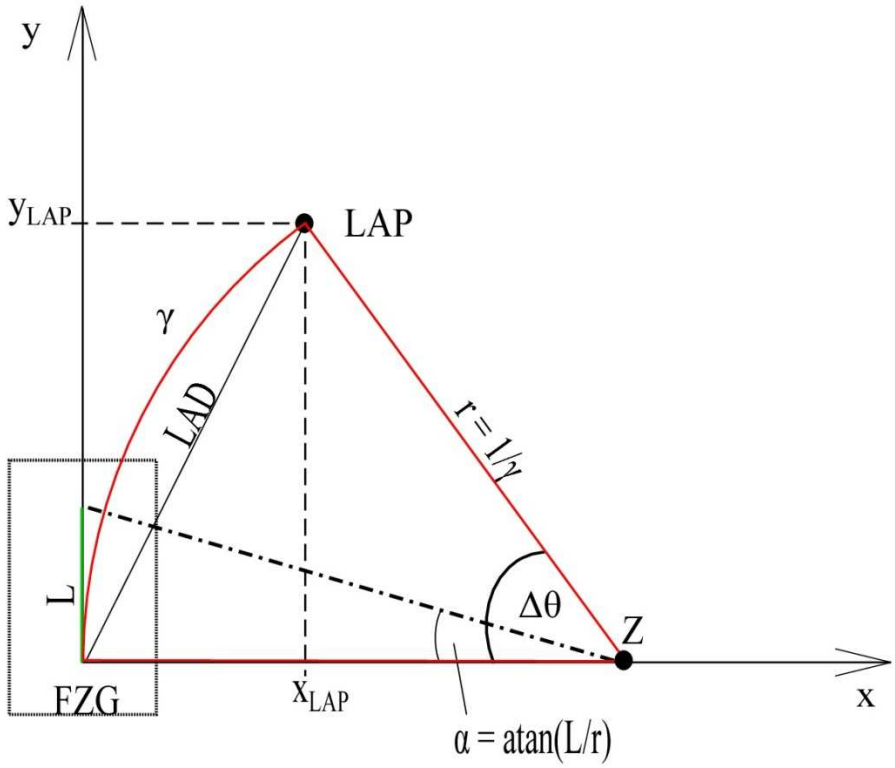

Abbildung 17: Lenkwinkelstrategie des Pure-Pursuit-Algorithmus. Der Kreisbogen durch den Ursprung des Fahrzeugkoordinatensystems und LAP ergibt die Sollkrümmung γ, auf der sich das Fahrzeug bewegen soll. Der Lenkwinkel α wird über den Achsabstand L des Fahrzeugs und den Drehkreisradius r um das Rotationszentrum Z bestimmt.

Die Achsen des zur Berechnung benutzten Koordinatensystems sind so gewählt, das die y-Achse die Fahrzeuglängsachse und die x-Achse die Hinterachse des Fahrzeugs repräsentieren und der Schnittpunkt von Translation und Rotation des Fahrzeugs den Koordinatenursprung bildet.

Die Krümmung γ des Kreisbogens, auf der sich das Fahrzeug zum LAP bewegen soll, ist reziprok zum Abstand r zwischen Fahrzeuglängsachse und dem Punkt Z, um den das Fahrzeug rotiert und repräsentiert die Änderung der Fahrzeugorientierung θ im Verhältnis zur zurückgelegten Strecke s auf dem Kreisbogen:

$$\gamma = {}^{1}/_{r} = {}^{d\theta}/_{ds} \qquad (12)$$

Aufgrund der Auswahl der Achsenanordnung folgt, dass sich das Rotationszentrum Z auf der x-Achse (y_Z = 0) befindet. Die Sollkrümmung γ wird so bestimmt, dass ein zur y-Achse tangentialer Kreisbogen durch den Koordinatenursprung und LAP gelegt wird.

Zur geometrischen Konstruktion der Krümmung γ werden die LAP-Koordinaten in Abhängigkeit des Drehkreisradius r und der Änderung der Orientierung des Fahrzeugs Δθ entlang des Kreisbogens beschrieben:

$$x_{LAP} = r - r \cdot \cos(\Delta\theta) = \frac{1 - \cos(\Delta\theta)}{\gamma} \qquad (13)$$

$$y_{LAP} = r \cdot \sin(\Delta\theta) = \frac{\sin(\Delta\theta)}{\gamma} \tag{14}$$

Mit (13) und (14) ergibt sich:

$$LAD^2 = x_{LAP}^2 + y_{LAP}^2 = \frac{2 - 2 \cdot \cos(\Delta\theta)}{\gamma^2} = \frac{2 \cdot x_{LAP}}{\gamma} \tag{15}$$

Stellt man (15) nach γ um, ergibt sich die anzustrebende Krümmung:

$$\gamma = \frac{1}{r} = \frac{2x_{LAP}}{LAD^2} \tag{16}$$

Die Kinematik des Fahrzeugs wird hier durch ein Einspur-Fahrzeugmodell mit lenkbarem Vorderrad und festem Hinterrad abstrahiert, dessen Translations- und Rotationsschwerpunkt durch den Schnittpunkt der Fahrzeuglängsachse und Hinterradachse definiert wird [5].

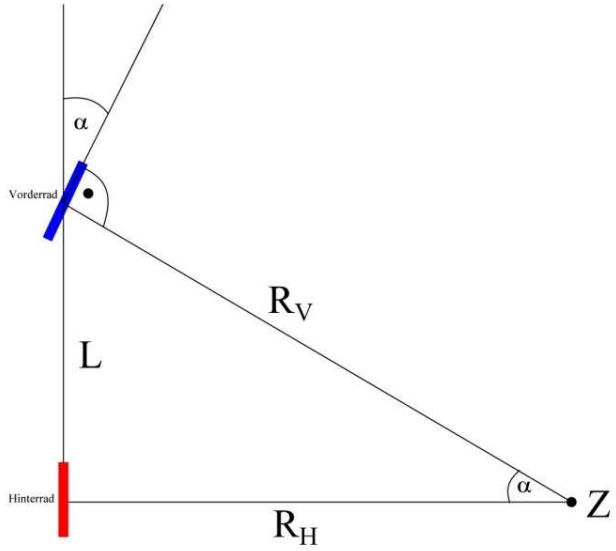

Abbildung 18: Lenkwinkelkonstruktion für das kinematische Einspur-Fahrzeugmodell. Vorder- und Hinterrad des Fahrzeugmodells haben den Abstand L und bewegen sich bei eingestelltem Lenkwinkel α auf den Radien R_V und R_H um das Rotationszentrum Z.

Soll sich das Hinterrad des Fahrzeugs auf dem Radius R_H um das Rotationszentrum Z bewegen, ergibt sich mit r = R_H und dem Achsabstand L:

$$\alpha = \tan^{-1}\left(\frac{L}{r}\right) = \tan^{-1}\left(\frac{2x_{LAP} \cdot L}{LAD^2}\right) \tag{17}$$

Pure-Pursuit zeichnet sich gegenüber Follow-The-Carrot durch die Einbeziehung der kinematischen Eigenschaften eines Fahrzeugs aus, wodurch starkes Überschwingen und lange Einschwingphasen vermieden, bzw. abgeschwächt werden sollen. In aktuellen Aufsätzen werden Anpassungen und Erweiterungen von Pure-Pursuit untersucht, so dass das Lenkverhalten z.B. durch eine geschwindigkeitsabhängige, Fuzzy-geregelte LAD oder, wie in Vector-Pursuit, durch Einbeziehen der Orientierung des LAP zum geplanten Pfad verbessert wird [13].

3.4.3 PD-Regler für die Abstandsregelung

In einem Regelkreis wird die Regelabweichung e(t) durch die Differenz der Regelgröße y(t) und der Führungsgröße w(t) gebildet, um durch die Stellgröße u(t) die Regelabweichung im eingeschwungenen Zustand zu minimieren.

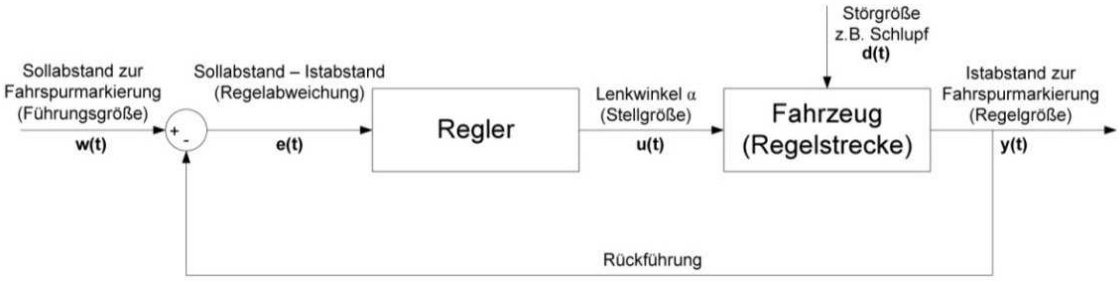

Abbildung 19: Regelkreis zur Positionierung des Fahrzeugs. Über den Sollabstand w(t) und den Istabstand y(t) des Fahrzeugs zur der Fahrspurmarkierung wird die Lenkwinkelstellgröße u(t) bestimmt.

Die durch einen minimalen und maximalen Radeinschlag begrenzte Lenkwinkelstellgröße u(t) = α(t) wird über die Regelabweichung e(t), welche sich aus der Differenz des Sollabstands w(t) = d und des Istabstandes y(t) ergibt, bestimmt (vgl. Kapitel 3.2).

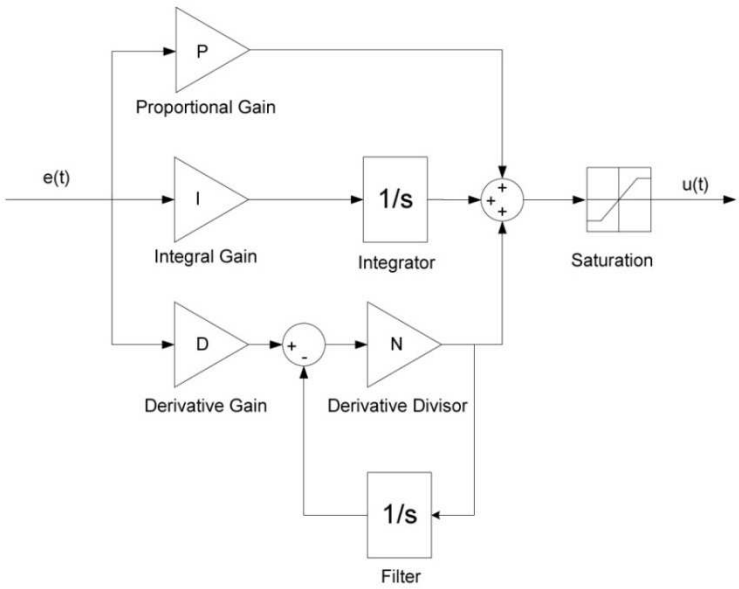

Abbildung 20: Simulink-Darstellung eines zeitkontinuierlichen PID-Reglers mit Glättung des differenzierenden Anteils und Stellgrößenbegrenzung **[18]**.

Die Stellgröße kann sich je nach Wahl des Reglers aus einem proportionalen, einem integralen und einem differenzierenden Anteil zusammensetzen .Aufgrund zahlreicher Dimensionierungshilfsmittel wurde der in MatLab 2010A beschriebene Regleraufbau [18] verwendet und dabei die maximal und minimal einstellbaren Lenkwinkel durch eine Sättigungsfunktion berücksichtigt. Der differentielle Anteil wird durch ein PT1-Glied gedämpft, um trotz starker Änderung der Regelabweichung ein ungewolltes Ausbrechen des Fahrzeugs durch hektische Lenkwinkelbefehle zu vermeiden.

Für die zeitkontinuierliche Darstellung des PID-Reglers gilt folgende Übertragungs-funktion:

$$G_R(s) = \frac{u(s)}{e(s)} = \left[P + I\left(\frac{1}{s}\right) + D\left(\frac{Ns}{s+N}\right)\right] \quad (18)$$

Der Kamerasensor überträgt pro Sekunde 60 Bilder der Fahrbahn, woraus eine Abtastzeit T_s von 16,67 ms resultiert. Die z-Übertragungsfunktion des diskretisierten Reglers lautet:

$$G_R(z) = \frac{u(z)}{e(z)} = P + Ia(z) + D\left[\frac{N}{1 + Nb(z)}\right] \quad (19)$$

wobei a(z) die Integrationsfunktion und b(z) die Filterfunktion zur Dämpfung des differentiellen Anteils darstellt. Je nach erforderlicher Genauigkeit können für beide Funktionen unterschiedliche Integrationsmethoden gewählt werden. Aufgrund der höheren Genauigkeit gegenüber einer Rechteckintegration mit den Forward- und Backward-Eulerverfahren die wurde hier die Trapezintegration verwendet.

Integrationsmethode	Forward Euler	Backward Euler	Trapezodial
a(z) und b(z)	$\frac{T_s}{z-1} = \frac{T_s z^{-1}}{1-z^{-1}}$	$\frac{T_s z}{z-1} = \frac{T_s}{1-z^{-1}}$	$\frac{T_s}{2}\frac{z+1}{z-1} = \frac{T_s}{2}\frac{1+z^{-1}}{1-z^{-1}}$

Tabelle 2: a(z) und b(z) in Abhängigkeit der Integrations- und Filtermethode **[18]**.

Ohne integralen Anteil (mit I = 0) und b(z) als Trapezintegration ergibt sich:

$$C(z) = \frac{u(z)}{e(z)} = P + D\left[\frac{N}{1 + N\frac{T_s}{2}\frac{1+\mathrm{z}^{-1}}{1-\mathrm{z}^{-1}}}\right] \quad (20)$$

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

$$u(z) = Pe(z) + \frac{DN(1-\mathrm{z}^{-1})}{1-\mathrm{z}^{-1} + N\frac{T_s}{2}(1+\mathrm{z}^{-1})}e(z) \quad (21)$$

$$u(z) = Pe(z) + \frac{DN(1-\mathrm{z}^{-1})}{\left(N\frac{T_s}{2}+1\right) + \left(N\frac{T_s}{2}-1\right)\mathrm{z}^{-1}}e(z) \quad (22)$$

Mit $a = \left(N\frac{T_s}{2}+1\right) = \left(\frac{NT_s+2}{2}\right)$ und $\mathrm{b} = \left(N\frac{T_s}{2}-1\right) = \left(\frac{NT_s-2}{2}\right)$ ergibt sich:

$$(a + b\mathrm{z}^{-1})u(z) = P(a + b\mathrm{z}^{-1})e(z) + DN(1-\mathrm{z}^{-1})e(z) \quad (23)$$

Zur zeitdiskreten Berechnung der Stellgröße wird (23) von der z-Domäne in die Zeit-Domäne transformiert:

$$au(k) + bu(k-1) = P(ae(k) + b\mathrm{e}(\mathrm{k}-1)) + DN(e(k) - \mathrm{e}(\mathrm{k}-1)) \quad (24)$$

$$u(k) = P\left(e(k) + \frac{b}{a}\mathrm{e(k-1)}\right) + \frac{DN}{a}\left(e(k) - \mathrm{e(k-1)}\right) - \frac{b}{a}u(k-1) \quad (25)$$

$$u(k) = \left(P + \frac{DN}{a}\right)e(k) + \left(P\frac{b}{a} - \frac{DN}{a}\right)\mathrm{e(k-1)} - \frac{b}{a}u(k-1) \quad (26)$$

Nach der Substitution von a und b ergibt sich folgende Bauvorschrift für ein zu implementierendes Software- oder Hardwaremodul:

$$u(k) = c_1 e(k) + c_2\mathrm{e(k-1)} - c_3 u(k-1) \quad (27)$$

mit $C_1 = P + \frac{DN}{N\frac{T_S}{2}+1}$, $C_2 = \frac{P\left(N\frac{T_S}{2}-1\right)-DN}{N\frac{T_S}{2}+1}$ und $C_3 = \frac{NT_S-2}{NT_S+2}$

Da der Lenkwinkel baulich begrenzt ist wird die Übertragungsfunktion nach links (α_{min}) und rechts (α_{max}) durch eine Sättigungsfunktion eingeschränkt:

$$u_{gesättigt}(k) = \begin{cases} \alpha_{\min}, falls\ u(k) < \alpha_{\min} \\ \alpha_{\max}, falls\ u(k) > \alpha_{\max} \\ sonst\ u(k) \end{cases} \quad (28)$$

3.5 Approximationsalgorithmen für trigonometrische und Wurzelfunktionen

Die SoC-Plattform wurde zur FPGA-Ressourceneinsparung so konfiguriert, dass keine Floating-Point-Libraries eingebunden wurden. Zur Berechnung der Wurzel und des Arkustangens werden in den folgenden Abschnitten Approximationen dieser Funktionen in Integer-Arithmetik vorgestellt.

3.5.1 Berechnung der Wurzel mit Integer-Arithmetik

Für die Berechnung der Wurzel in Integer-Arithmetik werden in aktuellen Aufsätzen u.a. Restoring- und Non-Restoring-Algorithmen verglichen [19]. Beide Algorithmen basieren auf einer 2n-Bit-Integer-Darstellung des Operanden D:

$$D = D_{2n-1}D_{2n-2}\ldots D_1 D_0 \quad (29)$$

Jedes Bitpaar des Operanden D ergibt ein Bit des n-Bit-Radikanten Q:

$$Q = \sqrt{D} = Q_{n-1}Q_{n-2}\ldots Q_1 Q_0 \quad (30)$$

Der in den einzelnen Iterationen verwendete partielle Rest benötigt mindestens n+1 Bits:

$$R = R_n R_{n-1}\ldots R_1 R_0 \quad (31)$$

Für jede neue Berechnung der Wurzel wird Q auf Q=0 gesetzt und für n = 16 Iterationen von k = 15 bis 0 durchgeführt. In jeder dieser Iterationen wird Qk zu Qk = 1 gesetzt und

$$\mathrm{R = D - Q\ x\ Q} \quad (32)$$

als partieller Rest berechnet. Wird R dadurch negativ ist das gesuchte Q zu groß und Qk muss auf Qk = 0 zurückgesetzt und der partielle Rest wieder hergestellt werden. Diese Vorgehensweise wird als Restoring-Algorithmus bezeichnet, weil jedes Bit des Radikanten Q pro Iteration zweimal gesetzt und gegebenenfalls restauriert werden muss. Implementierungsbeispiele können in [20] nachvollzogen werden.

Ein Non-Restoring-Algorithmus modifiziert jedes Q-Bit genau einmal, startet mit einem geratenen $Q = Q_{15}Q_{14} \dots Q_1Q_0 = \ 100\ldots000_2$ als partielle Wurzel und iteriert dann von k = 14 bis 0. In jeder Iteration wird wiederum (32) gebildet. Basierend auf dem Vorzeichen vom so entstandenen R wird eine 1 an der k-ten Stelle von Q addiert oder subtrahiert. Das folgende Beispiel zeigt die einzelnen Zwischenergebnisse der 4 dazugehörigen Iterationen:

$D\ = 01\ 11\ 11\ 11_2 = \ 127_{10}$
$Q_3 = 10\ 00_2$ mit $R = D - Q \times Q = 00\ 11\ 11\ 11_2$ R nicht negativ, initialer Guess
$+\ 01\ 00_2$
$Q_2 = 11\ 00_2$ mit $R = D - Q \times Q = 11\ 10\ 11\ 11_2$ R negativ
$-\ 00\ 10_2$
$Q_1 = 10\ 10_2$ mit $R = D - Q \times Q = 00\ 01\ 10\ 11_2$ R nicht negativ
$-\ 00\ 01_2$
$Q_0 = 10\ 11_2 = 11_{10}$ mit $R = D - Q \times Q = 00\ 00\ 01\ 10_2 = \ 6_{10}$

Implementierungsbeispiele sind in [21] und [22] zu finden.

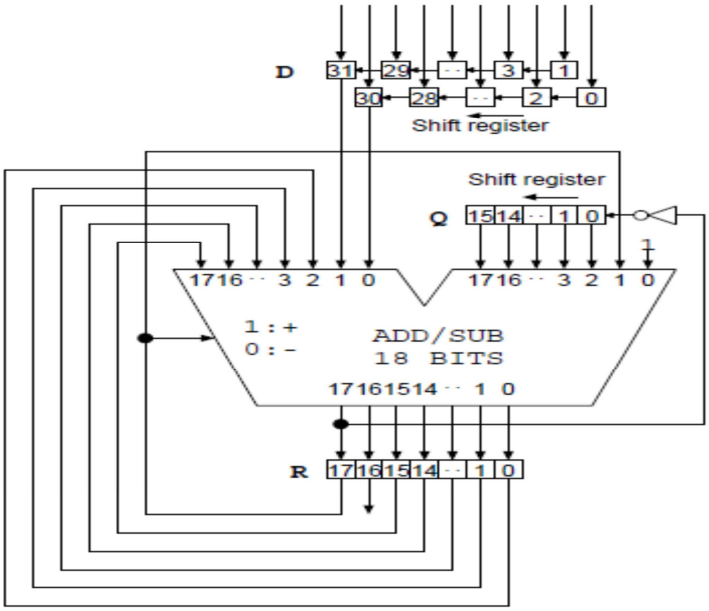

Abbildung 21: RTL-Modell des Non-Restoring-Wurzelalgorithmus in Integer-Arithmetik. Der 32-bit Operand wird in Register D gespeichert und pro Takt um zwei Stellen nach links geschoben. Register Q speichert den Radikanten, wird mit 0 initialisiert und pro Takt um ein Bit nach links geschoben. Register R speichert den partiellen Rest und wird zu Beginn mit 0 initialisiert **[19]**.

Die in dieser Arbeit genutzte Quadratwurzel-Implementierung basiert auf einem Non-Restoring-Algorithmus [19] und benutzt den partiellen Rest einer Iteration für die nächste Iteration, auch wenn dieser negativ ist. Auf diese Weise muss der Rest nicht durch den

Vorherigen ersetzt werden, wie es in den Restoring-Algorithmen über einen Multiplexer (if R >= 0 then continue, else restore Qk and R) realisiert wird.

Die RTL-Implementierung nach [19] verwendet einen Addierer/Subtrahierer und generiert pro Takt ein resultierendes Radikanten-Bit. Bei einer Bitbreite des Operanden von 32 Bit stehen der Radikant Q und der Rest R nach 16 Takten als Ergebnisse zur Verfügung.

3.5.2 Approximation der Arkustangensfunktion

Für die Bestimmung des Lenkwinkels nach Follow-The-Carrot und Pure-Pursuit wird die Arkustangensfunktion approximiert [23]**.** Betrachtet man eine komplexe Zahl z, so erfordert die Berechnung des Phasenwinkels

$$\varphi = \tan^{-1}\left(\frac{y}{x}\right) \qquad \text{mit } x = \mathrm{Re}(z) \text{ und } y = \mathrm{Im}(z) \tag{33}$$

mit x und y als Festkommazahlen eine Normalisierung, da die Division zu Werten außerhalb des darstellbaren Bereichs [-1, 1] führen kann. In Abhängigkeit der Quadrantenzugehörigkeit wird deshalb ein selbstnormalisierendes Seitenverhältnis r eingeführt.

Für Quadrant I und IV gilt für den Phasenwinkel φ:

$$r = \frac{x - y}{x + y} \tag{34}$$

$$\varphi_{QI} = -\varphi_{QIV} = 0.1963 \cdot r^3 - 0.9817 \cdot r + \frac{\pi}{4} \tag{35}$$

Befindet sich die komplexe Zahl z in Quadrant II oder III gilt:

$$r = \frac{x + y}{y - x} \tag{36}$$

$$\varphi_{QII} = -\varphi_{QIII} = 0.1963 \cdot r^3 - 0.9817 \cdot r + \frac{3\pi}{4} \tag{37}$$

Vergleicht man die Ergebnisse von (30) und (32), bzw. (34) ergibt sich ein maximaler Fehler von +/- 0,6 Grad [23].

4 Spurführungssimulation mit einem Einspur-Fahrzeugmodell

Für die Simulation der ausgewählten Fahrspuralgorithmen wurde ein MatLab-Simulator entwickelt, der auf einem Einspur-Fahrzeugmodell mit einem PT1-Glied als Lenkwinkelverzögerung basiert [5]. Das Verhalten dieses Modells wurde für Studien zur Spurführung mit variierten LADs und ROI-Abständen zur Hinterachse des Fahrzeugs genutzt, um die so gewonnenen Parameter für die Implementierung auf der Zielhardware zu nutzen (vgl. Kapitel 4.3).

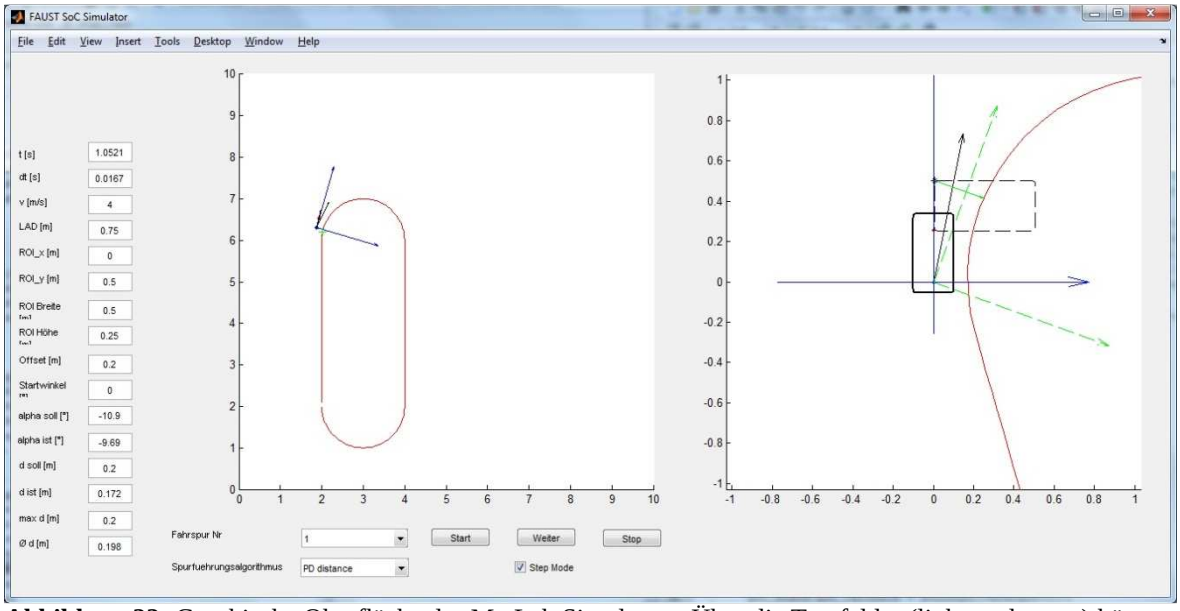

Abbildung 22: Graphische Oberfläche des MatLab Simulators. Über die Textfelder (links und unten) können die Geschwindigkeit v, die LAD und ROI-Position verändert und die implementierten Fahrspuralgorithmen ausgewählt werden. Das Verhalten des Einspur-Fahrzeugmodells wird in der Draufsicht (Vogelperspektive) bezogen auf das Weltkoordinatensystem (mitte) und Fahrzeugkoordinatensystem (rechts) dargestellt.

Basierend auf der durch die Bildfrequenz der Kamera von 60 Hz resultierenden Abtastperiode T_A = 16,7 ms wird die Position des Fahrzeugmodells in einem Weltkoordinatensystem zu jedem Abtastzeitpunkt aktualisiert, die Lage der Fahrspurmarkierung in der ROI bestimmt und ins Fahrzeugkoordinatensystem übertragen. Die Berechnungsabfolge orientiert sich dabei an dem in der geplanten RTL-Implementierung vorgesehenen Ablauf:

- Approximation der Fahrspur als Gerade mit Abstand r und Winkel Φ in der ROI
- Transformation der Geradenparameter in das Fahrzeugkoordinatensystem.
- Berechnung des LAP und des Abstands zur Fahrspurmarkierung als Istgrößen für die Fahrspurführungsalgorithmen im Hilfskoordinatensystem (vgl. Abbildung 15, sowie Gleichung (5)-(8) und (10)).
- Bestimmung der Lenkwinkelstellgröße aus den berechneten Istgrößen über die Arkustangensapproximation, bzw. mit der diskretisierten Übertragungsfunktion des PD-Reglers (vgl. Abbildung 16 und Abbildung 17, sowie Gleichung (11), (17) und (28)).
- Verschiebung der ROI für den nächsten Abtastschritt durch Nutzung des Istabstandes r zur Betimmung der optimalen ROI-Position (vgl. Gleichung (63) und Abbildung 48).

4.1 Kinematisches Einspur-Fahrzeugmodell mit Lenkwinkelverzögerung

Zur Simulation des Fahrverhaltens des Modellfahrzeugs wird dessen geographische Position in der Fahrzeugebene näherungsweise über ein kinematisches Ein Einspur-Fahrzeugmodell bestimmt. Dieses basiert auf der Geschwindigkeit $v_m(t)$ und dem Lenkwinkel $\alpha(t)$, die zur Berechnung der Fahrzeugorientierung $\Theta(t)$ in der Fahrzeugebene, der Vorderradposition (x_m; y_m) des lenkbaren Vorderrades M und der Hinterradposition (x_p; y_p) des starren Hinterrades P über Integralgleichungen verwendet werden [5].

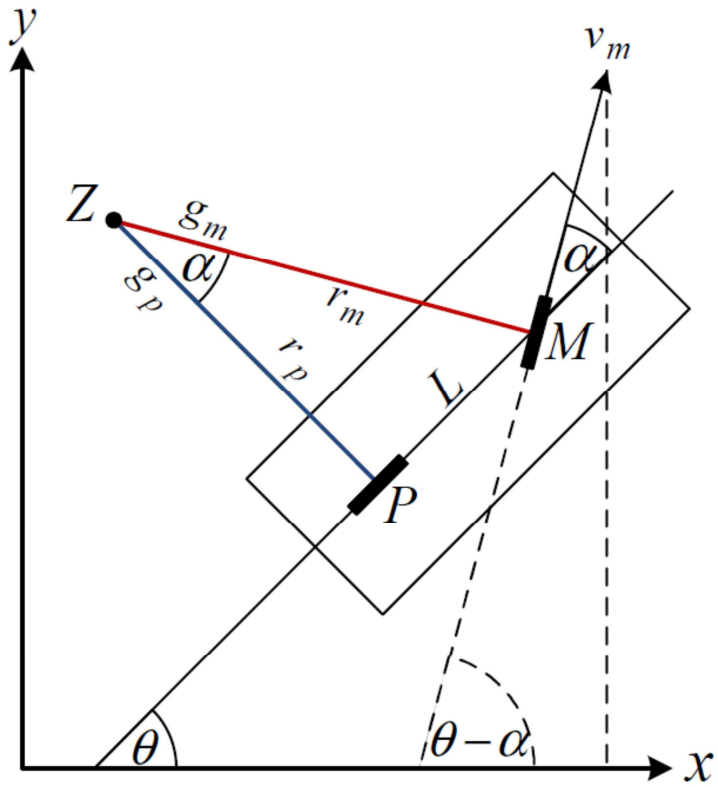

Abbildung 23: Einspur-Fahrzeugmodell mit gelenktem Vorderrad M und festem Hinterrad P. Das Modell bewegt sich bei einstellbarer Geschwindigkeit v_m, auf den Radien r_m und r_p um das Rotationszentrum Z und verändert aufgrund des Lenkwinkels α seine Orientierung Θ im Koordinatensystem (X, Y), das einen festen Bezug zur Fahrzeugebene hat.

Das Fahrzeugmodell bewegt sich im (X; Y) – Koordinatensystem, dessen Ursprung und Lage einen festen Bezug zur Fahrzeugebene darstellt, und wird durch folgende Integralgleichungen beschrieben [5]:

$$\Theta = \frac{1}{L}\int v_m(t)\cdot\sin(\alpha)\cdot dt \tag{38}$$

$$x_m = \int v_m(t)\cdot\cos(\Theta+\alpha)\cdot dt \tag{39}$$

$$y_m = \int v_m(t)\cdot\sin(\Theta+\alpha)\cdot dt \tag{40}$$

$$x_p = \int v_m(t)\cdot\cos(\alpha)\cdot\cos(\Theta)\cdot dt \tag{41}$$

$$x_p = \int v_m(t)\cdot\cos(\alpha)\cdot\sin(\Theta)\cdot dt \tag{42}$$

Da die Integralgleichungen (38) bis (42) gekoppelt und aufgrund der trigonometrischen Funktionen, die die zeitveränderliche Anregungsfunktion α(t) als Argument enthalten, massiv nichtlinear sind, gibt es keine analytische Lösung. Zur nummerischen Auswertung des Gleichungssystem wird dieses in ein Zustandsdifferentialgleichungssystem n-ter Ordnung überführt, welches mathematisch mit dem Zustandsvektor x, dem Zeitargument t, einer vektorwertigen Funktion f und dem Anfangszustand x_0 zum Zeitpunkt t_0 beschrieben wird:

$$\dot{x} = \frac{dx}{dt} = f(x,t) \quad mit \quad x(t_0) = x_0$$

Durch Umstellung der Gleichungen (38) bis (42) und Substitution ergibt sich bei konstanter Geschwindigkeit v_m und Lenkwinkel α folgendes Differentialgleichungssystem 1. Ordnung:

$$\dot{x}_1 = \frac{\mathrm{d}\Theta}{\mathrm{dt}} = \frac{1}{L} v_m \cdot \sin(\alpha) \tag{43}$$

$$\dot{x}_2 = \frac{\mathrm{dx}_m}{\mathrm{dt}} = v_m \cdot \cos(\mathrm{x}_1 + \alpha) \tag{44}$$

$$\dot{x}_3 = \frac{\mathrm{dy}_m}{\mathrm{dt}} = v_m \cdot \sin(\mathrm{x}_1 + \alpha) \tag{45}$$

$$\dot{x}_4 = \frac{\mathrm{dx}_p}{\mathrm{dt}} = v_m \cdot \cos(\alpha) \cdot \cos(\mathrm{x}_1) \tag{46}$$

$$\dot{x}_5 = \frac{\mathrm{dy}_p}{\mathrm{dt}} = v_m \cdot \cos(\alpha) \cdot \sin(\mathrm{x}_1) \tag{47}$$

Dieses Gleichungssystem wurde genutzt, um die Bewegung des Fahrzeugmodells in der Fahrzeugebene zu simulieren. Der verwendete Solver löst dieses Differentialgleichungssystem 1. Ordnung durch das Runge-Kutta-Verfahren ODE45 mit automatischer Schrittweitensteuerung.

Für die Verhaltensbeschreibung des Servomotors wurde eine Lenkverzögerung implementiert, welche die Dauer beschreibt, die vom Einstellen eines Soll-Lenkwinkel bis zum Erreichen dieses Soll-Lenkwinkels vergeht.

Mit der Abtastperiode $\Delta t = T_A$ und der Zeitkonstante T_1 ergibt sich für den Istwert des Lenkwinkels α_{ist} zum Zeitpunkt k [5]:

$$\alpha_{ist}(k) = \left(\alpha_{soll}(k-1) + \alpha_{soll}(k)\right) \frac{\Delta t}{2T_1 + \Delta t} - \alpha_{ist}(k-1) \frac{\Delta t - 2T_1}{2T_1 + \Delta t} \tag{48}$$

Die Zeitkonstante T_1 dieses Verzögerungsglieds erster Ordnung (PT_1) wurde als Näherungswert über Durchführung von Messreihen ermittelt und beträgt für eine Lenkwinkeländerung von 0° auf 30° ca. 250 ms [24].

4.2 Pfadmodellierung und Spurerkennung

Für die Simulation der Fahrzeugbewegung wird ein ovaler Sollpfad als Spurreferenz vorgegeben, die sich aus einer Liste von Punkten eines in der Fahrzeugebene liegenden Weltkoordinatensystems (X_W; Y_W) zusammensetzt. Zwei aufeinanderfolgende Punkte bilden ein Liniensegment, welches zur Bestimmung des Abstands r und des Winkels Φ des Fahrzeugs zum Sollpfad verwendet wird.

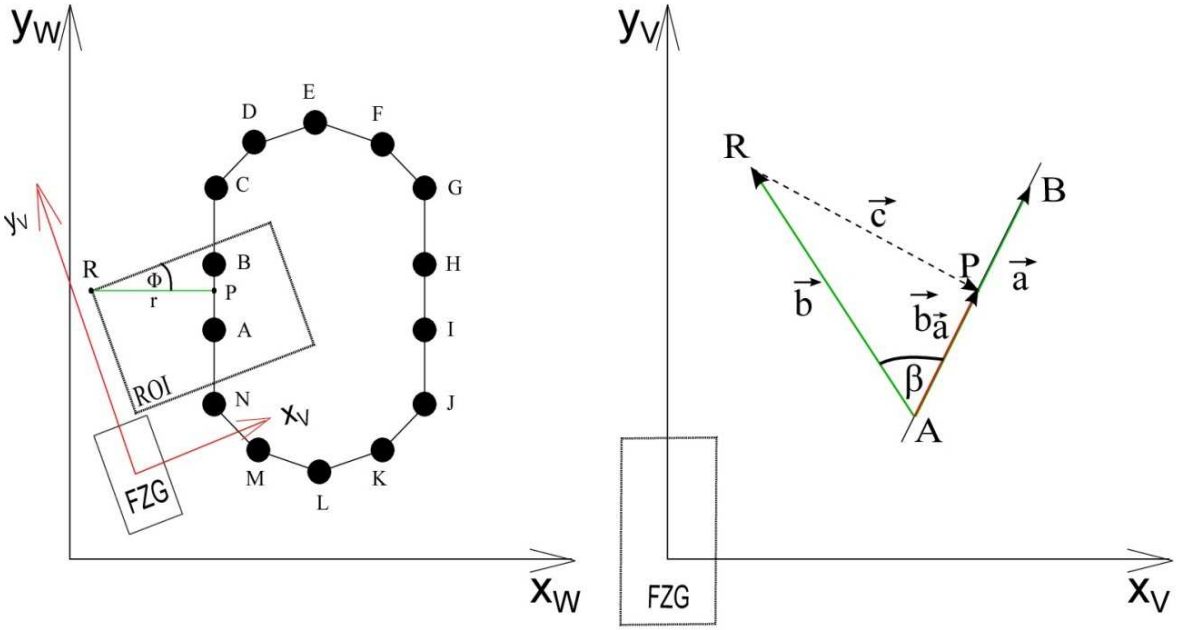

Abbildung 24: Sollpfad im Weltkoordinatensystem (XW; YW) und im Fahrzeugkoordinatensystem (X_V; Y_V). Die Punkte A bis N bilden Liniensegmente, mit denen die Länge r und der Winkel Φ des Vektors $\vec{c}$ in (X_V; Y_V) ermittelt wird.

Zur Berechnung von r und Φ werden alle Punkte des Pfades aus (X_W; Y_W) über den Zustandsvektor x des Einspur-Fahrzeugmodells durch Rotation und Verschiebung in das Fahrzeugkoordinatensystem (X_V; Y_V) transformiert und dort auf Zugehörigkeit zur ROI überprüft.

Um den Abstand $r = |\vec{c}|$ vom Punkt R, der den Koordinatenursprung der ROI darstellt, zum dichtesten Punkt P auf dem Liniensegment $\overrightarrow{AB} = \vec{a}$ zu bestimmen, wird das Skalarprodukt der Vektoren $\vec{a}$ und $\overrightarrow{b} = \overrightarrow{AR}$ gebildet:

$$\vec{a} \cdot \vec{b} = |\vec{a}| \cdot \left|\vec{b}\right| \cdot \cos(\beta) = |\vec{a}| \cdot \left|\vec{b}_{\vec{a}}\right| \tag{49}$$

Der Vektor $\vec{b}_{\vec{a}}$ beschreibt die orthogonale Projektion des Vektors $\vec{b}$ auf die durch $\vec{a}$ bestimmte Richtung und endet im Punkt P. Setzt man den Betrag von $\vec{b}_{\vec{a}}$ und $\vec{a}$ ins Verhältnis s, so gilt mit (49):

$$s = \frac{s_1}{s_2} = \frac{\left|\vec{b}_{\vec{a}}\right|}{|\vec{a}|} = \frac{\vec{a} \cdot \vec{b}}{|\vec{a}| \cdot |\vec{a}|} \tag{50}$$

Wenn $s1 \leq 0$ ist, dann ist der von $\vec{a}$ und $\vec{b}$ eingeschlossene Winkel $\beta \geq \frac{\pi}{2}$, und P somit nicht auf $\vec{a}$, sondern auf dem vorhergehenden Liniensegment. Wenn s1 ≥ s2, dann liegt P auf dem nachfolgenden Segment. Nur wenn $0 < s < 1$ gilt, kann P auf $\vec{a}$ liegen und über s bestimmt werden:

$$P = A + s \cdot \vec{a} \tag{51}$$

Der Winkel Φ und der Abstand r wird über den Vektor $\vec{c} = \overrightarrow{RP}$ berechnet:

$$\Phi = \tan^{-1}\left(\frac{y_P - y_R}{x_P - x_R}\right) \tag{52}$$

$$r = |\vec{c}| = \sqrt{(y_P - y_R)^2 + (x_P - x_R)^2} \tag{53}$$

4.3 Simulationsergebnisse

Die Berechnung des LAP und des Istabstands für die Abstandsregelung wird von der LAD und der y-Entfernung der ROI y_R im Fahrzeugkoordinatensystem beeinflusst (vgl. Abbildung 15). Eine grundsätzliche Voraussetzung für alle Spurführungsalgorithmen ist, dass neben den günstigen LAD-yR-Wertekombinationen alle zur Berechnung genutzten Parameter, wie z.B. die (x; y)-Position der ROI und die Istwerte der Fahrspurapproximation in das Fahrzeugkoordinatensystem übertragbar sind (vgl. Kapitel 3.3).

Für die Auswahl geeigneter LAD-y_R-Wertekombinationen für die Fahrspuralgorithmen Follow-The-Carrot, Pure-Pursuit und Abstandsregelung wurde deshalb eine Parameterstudie mit den Geschwindigkeiten v = 1, 2 und 4 m/s und festem Sollabstand d = 20 cm der Fahrzeuglängsachse zur Fahrspurmarkierung auf einer vorgegebenen Simulationsstrecke durchgeführt (vgl. Tabelle 9 in Anhang A).

Folgende Sollkriterien wurden für die Parameterauswahl definiert:

- die maximal zulässige Differenz Δd_{max} zwischen der Fahrzeuglängsachse d_{Ist} und vorgegebenen Sollabstand zur Fahrbahnmarkierung d_{Soll} beträgt maximal 4 cm.

- die durchschnittliche Differenz Δd_{mean} zwischen d_{Ist} und d_{Soll} beträgt maximal 2 cm

Der PD-Regler berechnet die einzustellenden Lenkwinkelbefehle nur auf Basis des Istabstands des Fahrzeugs zur Fahrbahnmarkierung, weshalb die LAD keinen Einfluß auf den Lenkwinkel hat (Abbildung 25).

Die Fahrspurführungssimulation mit Abstandsregelung hat gezeigt, dass eine zum Fahrzeugkoordinatenursprung dichtere ROI zu einer genaueren Spurverfolgung führt. Aufgrund des geringen Maximalfehlers von 3,09 cm wurde für die RTL-Modellierung des PD-Reglers eine ROI-Entfernung von 50 cm zur Hinterachse ausgewählt (vgl. Abbildung 25 und Tabelle 9).

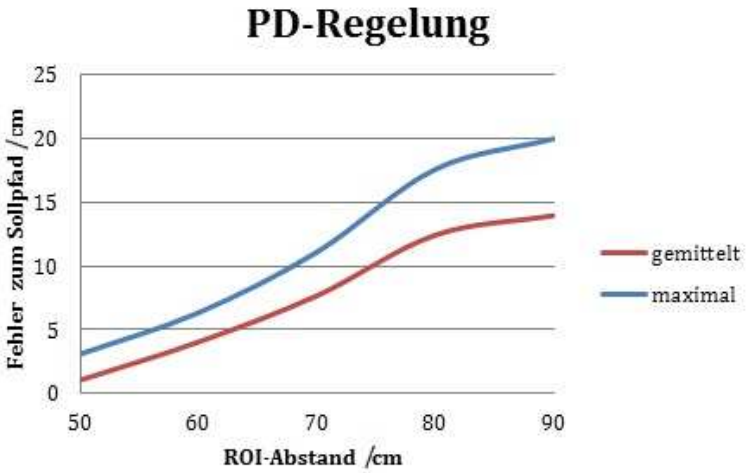

Abbildung 25: Fehler zum Sollpfad in Abhängigkeit des ROI-Abstands. Je dichter die ROI an der Fahrzeughinterachse ist, desto genauer ist die PD-Abstandsregelung.

Für die Bestimmung der ROI-Position y_R und LAD für Follow-The-Carrot und Pure-Pursuit wurden die ROI-Abstände zur Hinterachse im Intervall [50 cm, 80 cm], sowie die LAD im Intervall [50 cm, 90 cm] untersucht.

Bei der Spurführung mit Follow-The-Carrot hat sich gezeigt, dass die LAD maximal gleich, besser aber: kürzer als die Entfernung yR der ROI zur Hinterradachse sein soll, um geringe Positionsfehler zu garantieren. LADs, die mehr als 10 cm länger als die ROI-Entfernung sind, führen zu einem sehr instabilen, aufschwingenden System, was zum Übertreten der Fahrspurmarkierung führt (vgl. Abbildung 26 und Tabelle 9).

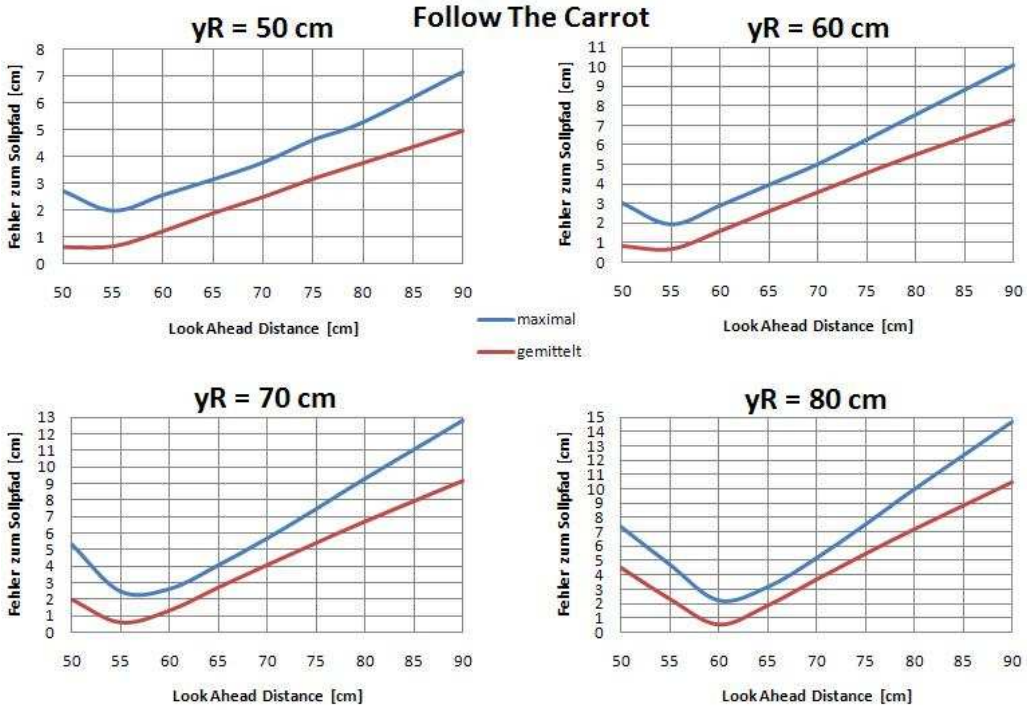

Abbildung 26: Fehlerdiagramme zur Bestimmung günstiger LAD-yR-Kombinationen für Follow-The-Carrot.

Die Simulation von Pure-Pursuit ergab, dass die Spurführung am genauesten ist, wenn die LAD ungefähr gleich groß oder kleiner als yR ist (vgl. Abbildung 27 und Tabelle 9).

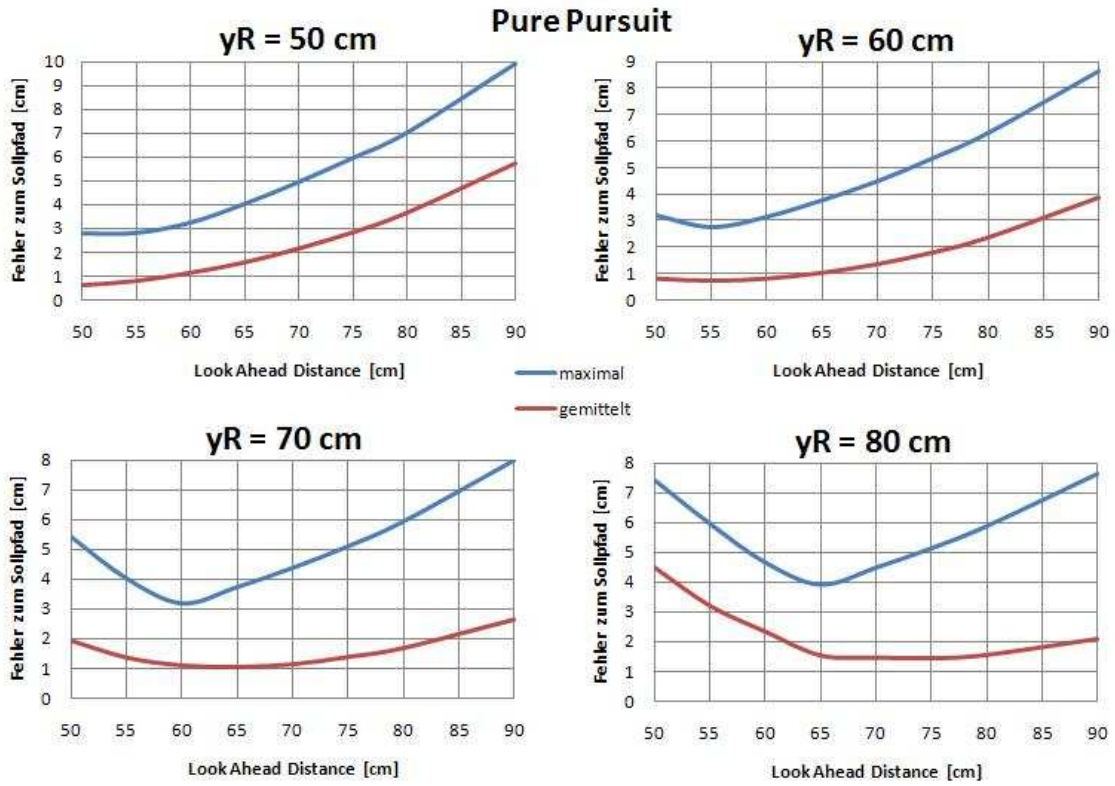

Abbildung 27: Fehlerdiagramme zur Bestimmung günstiger LAD-yR-Kombinationen für Pure-Pursuit.

Fahrspuralgorithmus	LAD [cm]	yR [cm]	Δd_{mean} [cm]	Δd_{max} [cm]
Follow-The-Carrot	55	50	0,66	2,0
	55	60	0,69	1,97
	55	70	0,55	2,44
	60	80	0,53	2,23
Pure-Pursuit	55	50	0,78	2,83
	55	60	0,75	2,74
	60	70	1,12	3,21
	65	80	1,56	3,93
PD-Regelung	-	50	1,04	3,09

Tabelle 3: Günstige LAD-yR-Wertekombinationen für die Fahrspurführung mit Follow-The-Carrot, Pure-Pursuit und PD-Regelung.

5 Modifikationen der Bildverarbeitungskette

Die in [4] entwickelten Verbesserungsalternativen für die Bildverarbeitungskette wurden analysiert und mit Anforderungen aus der Ausgangsanalyse und der Fahrspurführungssimulation überprüft. Für die auf Sobel-Kantenfilterung mit fester Binarisierungsschwelle basierende Geradenerkennung im verzerrten Bild mit anschließender perspektivischer Korrektur der Fahrspurapproximation in einem RTOS Task ergaben sich dabei folgende für die Istgrößenberechnung zur Fahrspurführung ungünstige Eigenschaften:

- Nur die Istwerte r und Φ der Fahrspurapproximation werden transformiert
- Die Position und die Dimensionen der ROI werden nicht perspektivisch korrigiert und stehen der Fahrspurführung nicht zur Verfügung
- Die Kanten des Sobel-Bildes führen zu oszillierenden r- und Φ-Istwerten, da unvorhersehbar abwechselnd die Kante des Hell-Dunkel- oder Dunkel-Hell-Übergangs als längste Gerade erkannt wird.
- Die feste Binarisierungsschwelle führt bei ungünstigen Lichtverhältnissen zum Verlust für die Hough-Transformation wichtiger Bildpunkte
- Die Ausgleichsmatrix zur perspektivischen Entzerrung wurde auf den Bilddarstellungsbereich optimiert und kein den Genauigkeitsanforderungen der Fahrspurführung entsprechender metrischer Zusammenhang hergestellt.

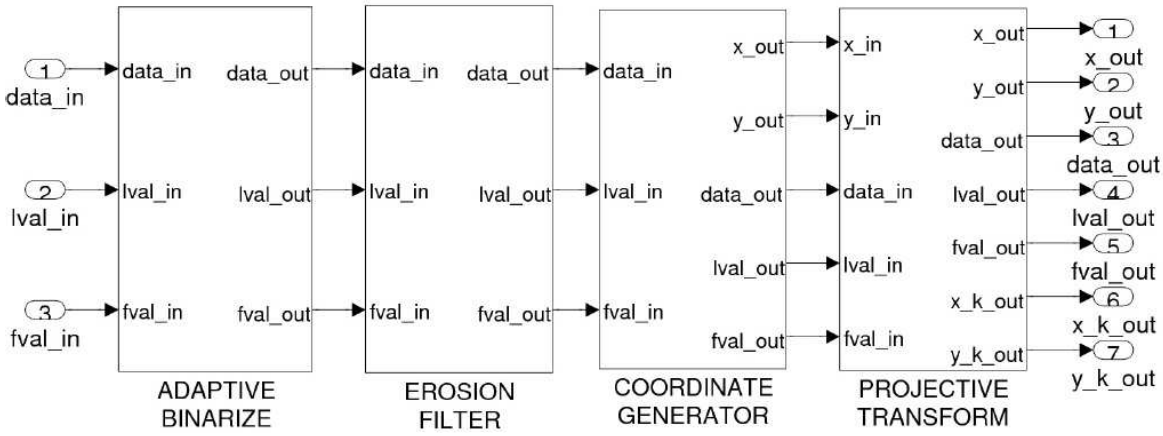

Abbildung 28: Modifizierte Bildverarbeitungskette. Die Binarisierungsschwelle wird pro Bild anhand des bisher höchsten und niedrigsten Grauwertes bestimmt. Der Erosionsfilter unterdrückt Bildrauschen und verkleinert Konturen. Zur Entzerrung des Kameradatenstroms werden durch lval und fval Koordinaten generiert in die Fahrzeugebene projiziert.

Zur Verbesserung der Vorentwicklungsstufe für die Fahrspurerkennung wurden die Submodule „Adaptive Binarize“ und „Erosion Filter“ entwickelt und mit den vorhandenen Komponenten „Coordinate Generator“ und „Projective Transform“ der bisherigen Bildverarbeitungskette im Modul „Image Processing“ gekoppelt.

Das Submodul „Adaptive Binarize“ spreizt den Grauwertebereich des Pixeldatenstroms auf den vollen 8-bit-Bereich auf und berücksichtigt somit die Lichtverhältnisse auf der Fahrbahn bei der Binarisierung. Der Erosionsfilter erodiert breite Pixelbereiche durch morphologische Operationen und ersetzt die Kantenfilterung mit Sobel-Operator, um doppelte Kanten in der ROI zu vermeiden, welche zuvor zu oszillierenden

Geradenapproximationen führten. Die für die perspektivische Korrektur genutzte Ausgleichsmatrix des Submoduls „Projective Transform“ wurde auf die Position der Kamera kalibriert und bringt die Pixel der Bilddatenstromebene in einen metrischen Zusammenhang mit der Fahrzeugebene.

5.1 Adaptive Grauwertschwelle zur Binarisierung des Pixeldatenstroms

Die Bestimmung der im Modul „Adaptive Binarize“ genutzten Grauwertschwelle basiert auf einer Grauwertskalierung des Pixeldatenstroms durch homogene Punktoperationen, d.h. durch nur vom Grauwert abhängige Veränderung der Pixel unabhängig von den Nachbarpixeln.

5.1.1 Grauwertskalierung und Binarisierung zur Datenreduktion

Durch die lineare Skalierung wird der Grauwerteumfang des Bildes gleichmäßig gestreckt, sodass der Teilbereich $[g_1, g_2]$ mit $0 \leq g1 < g2 \leq 255$ auf die gesamte, darstellbare 8-bit Grauwertskala [0, 255] projiziert wird [25].

Diese Spreizung des Grauwertebereichs erfolgt durch die Skalierungsfunktion f(g):

$$f(g) = \begin{cases} 0 & falls\ g < g_1 \\ \dfrac{g - g_1}{g_2 - g_1} \cdot 255 & falls\ g_1 \leq g \leq g_2 \\ 255 & falls\ g_2 < g \end{cases} \tag{54}$$

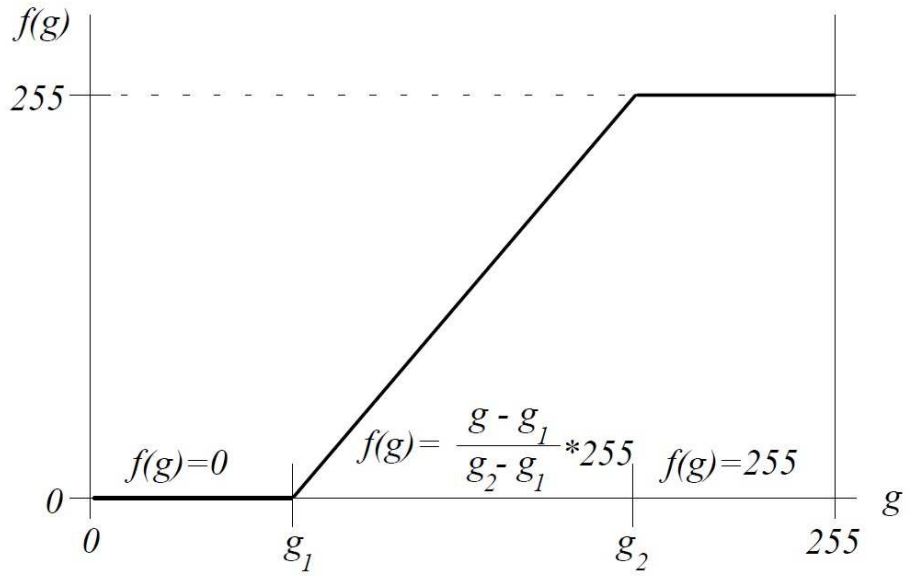

Abbildung 29: Skalierungsfunktion f(g) zur Grauwertebereichsspreitzung.

Für die Variablen g_1 und g_2 wurden der minimale im Bild auftretende Grauwert g_{min} und der maximale im Fahrbahnabbild auftretende Grauwert g_{max} genutzt, sodass mit $g_1 = g_{min}$ und $g_2 = g_{max}$ gilt:

$$f'(g) = \frac{g - g_{\min}}{g_{\max} - g_{\min}} \cdot 255 \tag{55}$$

Nicht im Bild auftretenden Grauwerte werden mit f´(g) eliminiert und das für die Fahrspur typische Histogramm gespreizt, sodass die Fahrspurmarkierungen deutlich hervorgehoben werden (vgl. Abbildung 31). Die Bestimmung von g_{max} und g_{min} erfolgt pro Kamerabild, wobei durch die Verarbeitung im Videodatenstrom nicht die für das gesamte Bild geltenden Grauwertextreme, sondern nur der zum aktuellen Zeitpunkt minimale bzw. maximale Grauwert betrachtet werden (vgl. Abbildung 33). Durch die wiederkehrende Struktur der Kamerabilder führte dies in der Erprobung zu keinerlei Nachteil.

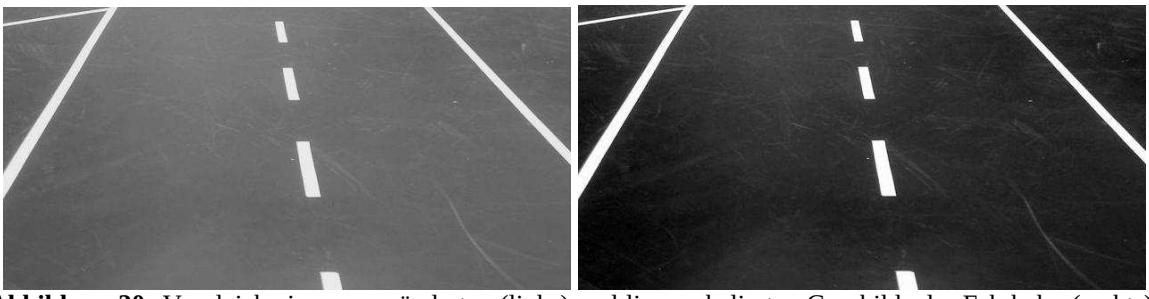

Abbildung 30: Vergleich eines unveränderten (links) und linear skalierten Graubilds der Fahrbahn (rechts). Die starke Lichtquelle führt zu Reflektionen auf der Fahrbahn und geringeren Grauwertunterschieden zwischen Fahrbahn und Fahrspurmarkierung.

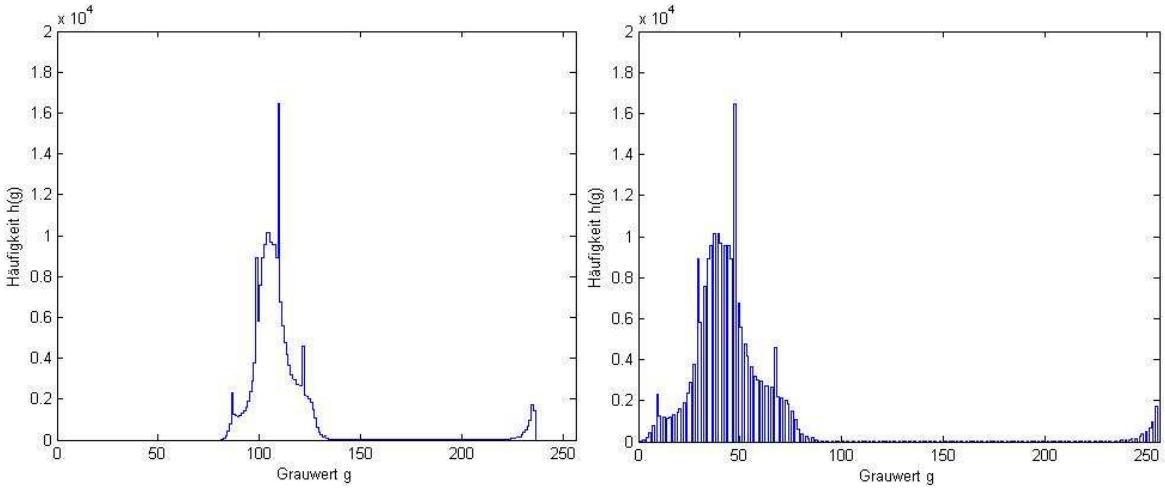

Abbildung 31: Histogramme des unveränderten Bildes (links) und linear skalierten Bildes (rechts). Die markante Struktur bleibt erhalten, ist aber voneinander besser unterscheidbar.

Da die Kamerabilder zur Datenreduktion nach der Skalierung binarisiert werden sollen, wird Gleichung (55) so umgestellt, dass anstelle einer neuen Grauwertzuweisung eine direkte Binarisierung mit einer prozentualen Grauwertschwelle p bezogen auf die Differenz $\Delta g = g_{max} - g_{min}$ erfolgt:

$$f_{binarize}(g) = \begin{cases} 0, & falls\ g - g_{min} < \Delta g \cdot p \\ 1, & falls\ g - g_{gmin} \geq \Delta g \cdot p \end{cases} \quad (56)$$

5.1.2 RTL-Modell des Binarisierungsmoduls mit Grauwertskalierung

Das „Adaptive Binarize“-Modul wird mit einer Frequenz $f_{13.5}$ = 13.5 MHz betrieben, welche durch den Clock-Manager zur Verfügung gestellt wird. Die eingehenden Synchronisierungssignale lval und fval, sowie das 8-bit Grauwertsignal data werden vom Modul „Camera Interface“ angelegt.

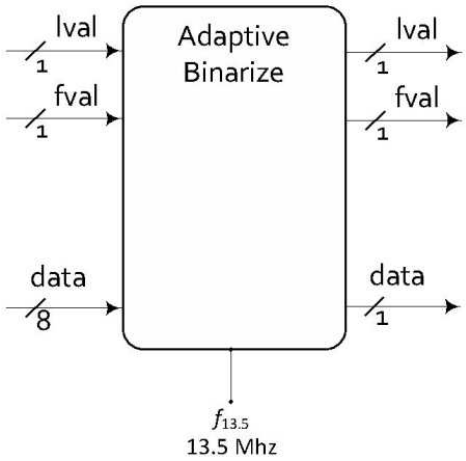

Abbildung 32: Blackbox des Moduls zur Binarisierung mit adaptiver Grauwertschwelle. Die Binarisierung der 8-bit Grauwerte erfolgt mit der Pixelfrequenz $f_{13.5.}$

Nach erfolgter Binarisierung liegt das bearbeitete Pixel mit einem Takt Latenz am Ausgang an. Die Register g_max und g_min werden während der Bildpausen (fval = false) mit 0 bzw. 255 reinitialisiert, sodass ab dem ersten Pixel des nächsten Bildes der aktuelle Minimal- und Maximal-Grauwert über Komparatoren bestimmt wird.

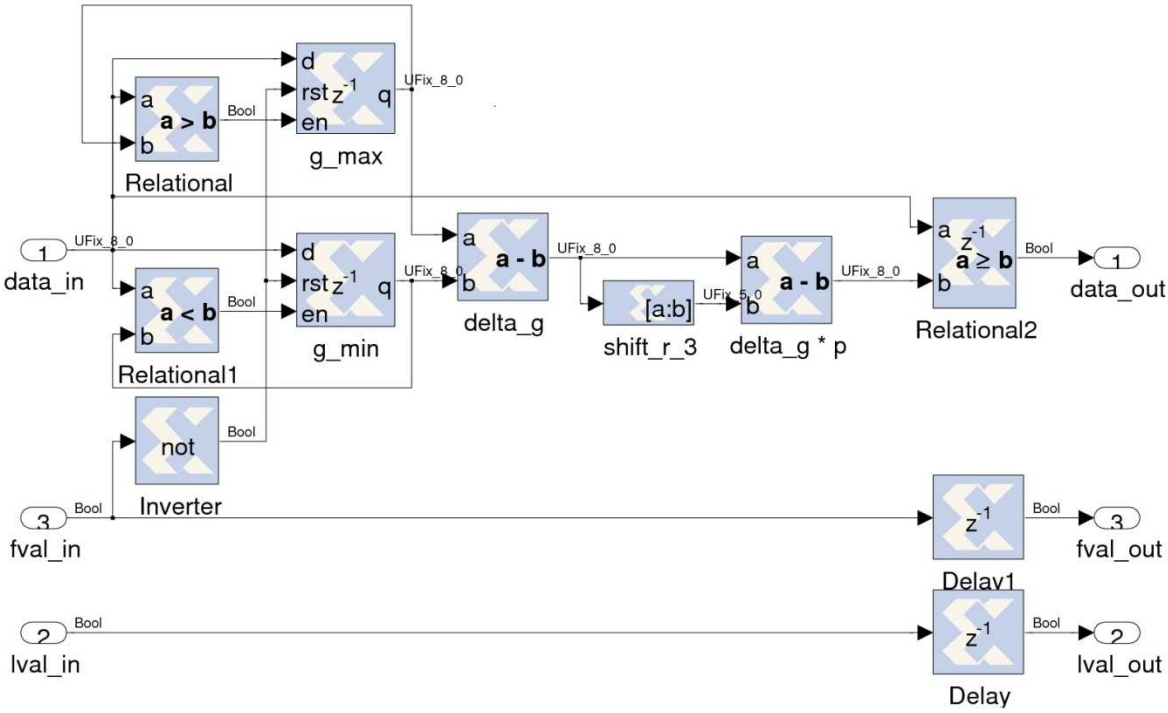

Abbildung 33: Pipelinedatenpfad zur Binarisierung mit adaptiver Grauwertschwelle. Grauwerte, die mindestens 87,5 % der Differenz delta_g vom aktuellen Maximum g_max und Minimum g_min erreichen, werden als helles, alle anderen als dunkles Binärpixel weitergereicht.

Die prozentuale Grauwertschwelle wurde mit p = 0.875 so gewählt, dass durch Schieben von Δg um 3 Bits nach rechts und Subtraktion dieses Wertes von Δg eine Multiplikation vermieden wird. Durch einen Vergleich des um g_min reduzierten Eingangsgrauwerts mit dem so entstandenen Schwellwert wird das binäre Ausgangssignal erzeugt.

5.2 Morphologische Filterung des Pixeldatenstroms

Durch die Binarisierung des Pixeldatenstroms unterscheiden sich die Fahrspurmarkierungen deutlich von der restlichen Fahrbahn, sind aber für eine genaue Fahrspurerkennung durch die HT zu breit, da die Geradenapproximation in der Hough-Ebene zu mehrdeutigen Maxima mit unterschiedlichen Abständen r und Winkeln Φ führt (vgl. Abbildung 34 und Kapitel 6).

5.2.1 Morphologische Operatoren

Morphologische Operatoren nutzen die Umgebungsinformationen eines Pixels und lassen sich in ihrer Wirkung durch die Form der Umgebungsmaske steuern, wodurch eine Richtungsabhängigkeit erreicht wird (vgl. Abbildung 35 und Abbildung 36).

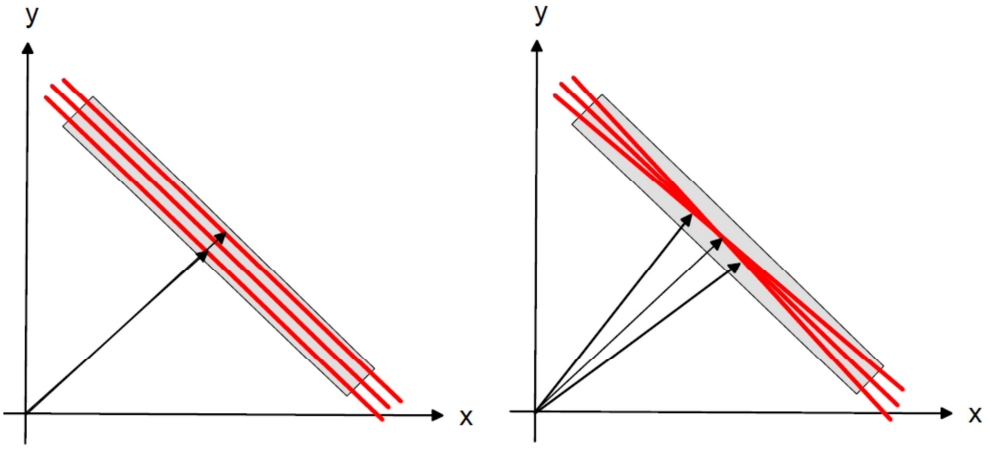

Abbildung 34: Geradenhypothesen von breiten Fahrspurmarkierungen. Geraden mit unterschiedlichen Abständen r und Winkeln Φ passen in die Fahrspurmarkierung (grau) **[26]**.

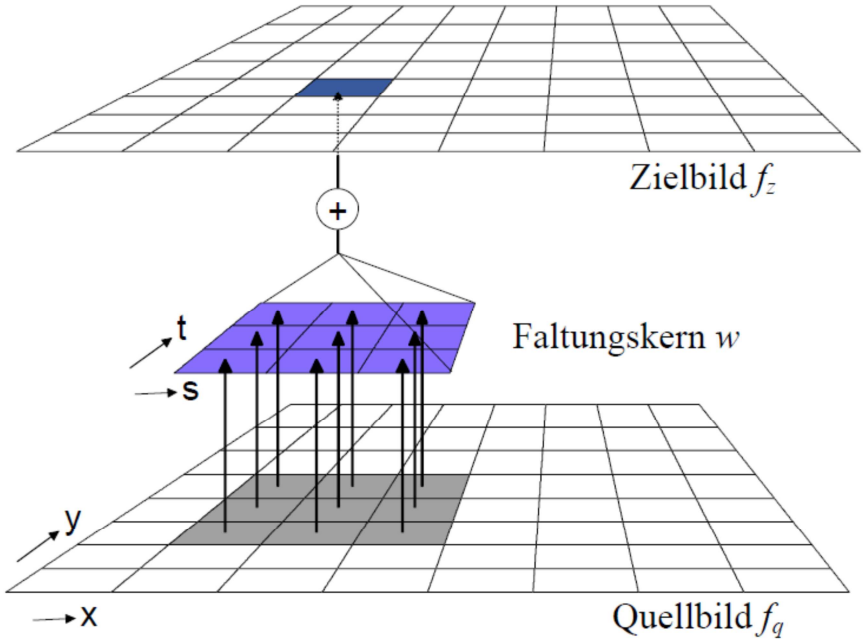

Abbildung 35: Prinzip einer Faltung mit quadratischer Maske. Auf der Umgebung eines Pixels (x;y) im Quellbild fq wird ein quadratischer Faltungskern w angewendet und das Ergebnis der Faltung an der Stelle (x;y) in das Zielbild fz eingetragen.

Wichtige morphologische Operatoren sind [27]:

- Dilatations-Operator
  Aus der geordneten Folge der Umgebungspixel wird der maximale Grauwert ausgewählt, sodass sich helle Gebiete auf Kosten der dunklen ausdehnen. Sei p die Anzahl der in einem binären Eingangsbild g(x,y) gesetzten Pixel, dann gilt:

$$g'(x,y) = \begin{cases} 1, & \text{für } p > 0 \\ 0, & sonst \end{cases} \quad (57)$$

- Erosions-Operator
  Aus der geordneten Folge der Umgebungspixel wird der minimale Grauwert ausgewählt, sodass sich dunkle Gebiete auf Kosten der hellen ausdehnen. Sei p die Anzahl der in einem binären Eingangsbild g(x,y) gesetzten Pixel und k die Anzahl der Pixel in der Faltungskern w, dann gilt:

$$g'(x,y) = \begin{cases} 1, & \text{für } p = k \\ 0, & \text{für } p < k \end{cases} \quad (58)$$

- Closing-Operator
  Es wird zuerst eine Dilatation und danach eine Erosion auf die Umgebungspixel ausgeführt, sodass kleine Lücken an den Objekträndern geschlossen werden.

- Opening-Operator
  Es wird zuerst eine Erosion und danach eine Dilatation auf die Umgebungspixel ausgeführt, sodass ausgefranste Objektränder geglättet und kleine Pixelstörungen oder dünne Linien unterdrückt werden.

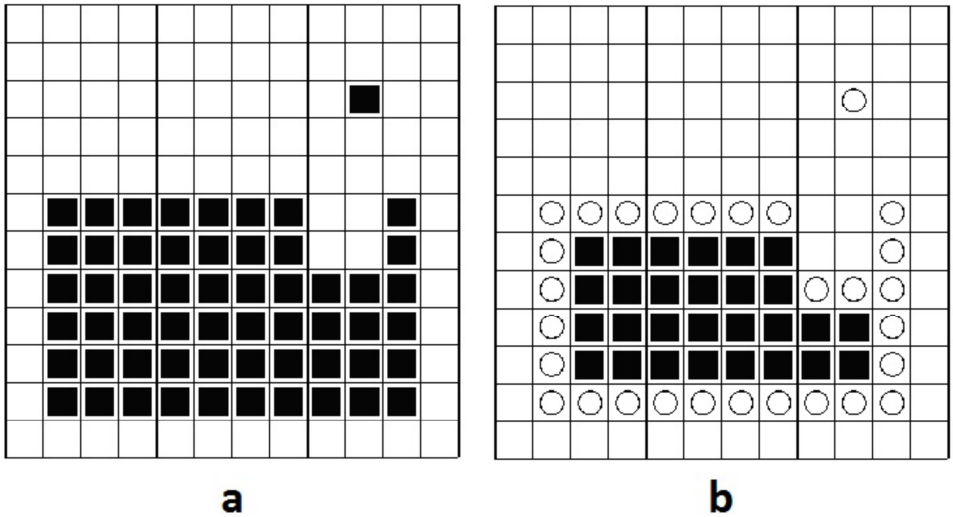

Abbildung 36: Wirkungsweise des Erosions-Operators. Die Kreise markieren die Stellen der erodierten Pixel. a) Eingangsbild, b) Erosionsgrenze = 8, also eine UND-Verknüpfung der 3x3-Faltungsmatrix **[27]**.

Alle morphologischen Operatoren lassen sich auf Grauwert- und Binärbildern anwenden, wobei bei der Faltung eines binären Bildes das Sortieren der Umgebungspixel entfällt und die Minimum-, bzw. Maximumsuche durch logische UND-, bzw. ODER-Verknüpfungen auf den Binärpixeln ersetzt werden (vgl. Gleichungen (57) und (58)).

5.2.2 RTL-Modell des Erosionsfilters

Der binarisierte Pixeldatenstrom wird am data-Eingang vom „Adaptive Binarize“ Modul angelegt und mit der Pixelfrequenz $f_{13.5}$ übertragen.

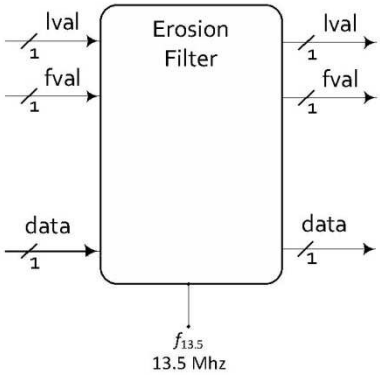

Abbildung 37: Blackbox des Erosionsfilter-Moduls. Der eingehende Binärpixeldatenstrom wird mit der Frequenz $f_{13.5}$ durch UND-Verknüpfung einer quadratischen 3x3 Faltungsmaske erodiert.

Zur Unterscheidung der Bildpausen von der eigentlichen Pixelübertragung wird das lval-Signal als Enable für die Zeilenzwischenspeicher genutzt, welche den Datenstrom deserialisieren und die Umgebungspixel als quadratische 3x3-Matrix zur Verfügung stellen. Nach der Erosion durch UND-Verknüpfung liegt das Ergebnispixel mit einer Latenz von 4 Takten am data-Ausgang an.

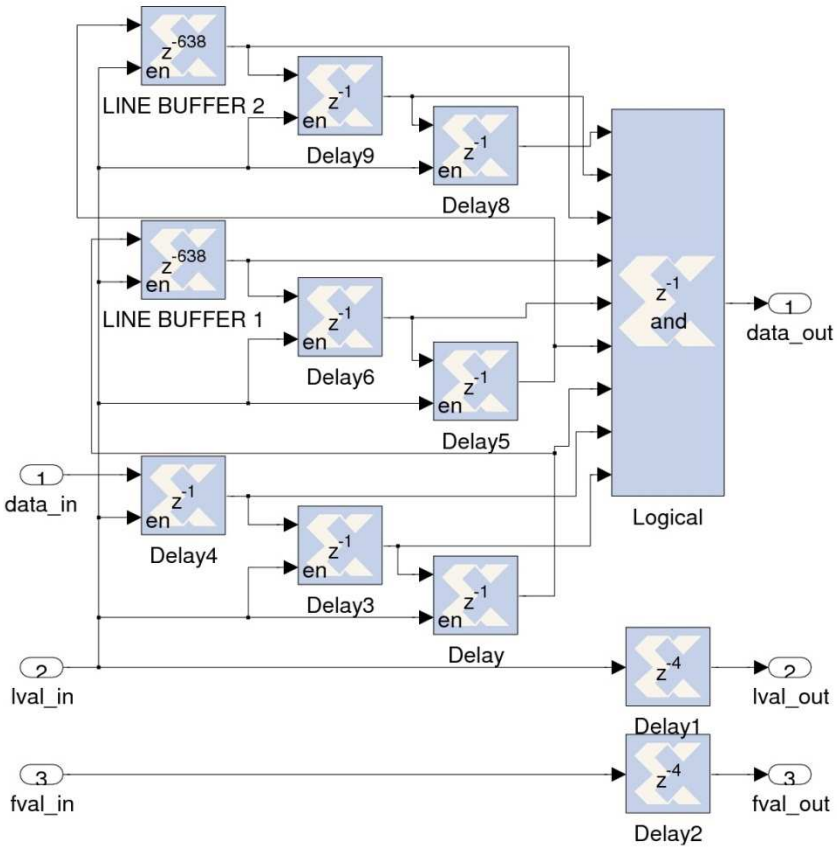

Abbildung 38: RTL-Modell zur morphologischen Filterung des binären Pixeldatenstroms. Durch die Schieberegisterketten LINE BUFFER 1 und 2 wird der Datenstrom deserialisiert und durch UND-Verknüpfung eines 3x3-Faltungskerns erodiert.

5.3 Kalibrierung der Transformationsmatrix B für die perspektivische Korrektur

Die Elemente der Transformationsmatrix B in Gleichung (2) werden durch eine Kalibrierung für eine konstante Kameraausrichtung bestimmt, sodass das Verhältnis der Bildebene zur Fahrzeugebene zu jedem Zeitpunkt identisch mit dem Verhältnis zum Zeitpunkt der Kalibrierung ist.

Zur Kalibrierung wird eine Punktanordnung mit bekannten Koordinaten in der Fahrzeugebene durch Ermitteln der korrespondierenden Punkte im Kamerabild und der Kalibrieranordnung erstellt, sodass nach Umstellen der Gleichungen (3) und (4) ein lineares Gleichungssystem aufgestellt wird, bei dem alle unbekannten Elemente b_{ij} der Matrix B auf eine Seite stehen (vgl. Abbildung 39) [5].

$$x_k = b_{11}x'_k + b_{12}y'_k + b_{13} - b_{31}x'_k x_k - b_{32}y'_k x_k \tag{59}$$

$$y_k = b_{21}x'_k + b_{22}y'_k + b_{23} - b_{31}x'_k y_k - b_{32}y'_k y_k \tag{60}$$

$$\begin{pmatrix} x'_{k1} & y'_{k1} & 1 & 0 & 0 & 0 & -x'_{k1}x_{k1} & -y'_{k1}x_{k1} \\ 0 & 0 & 0 & x'_{k1} & y'_{k1} & 1 & -x'_{k1}y_{k1} & -y'_{k1}y_{k1} \\ x'_{k2} & y'_{k2} & 1 & 0 & 0 & 0 & -x'_{k2}x_{k2} & -y'_{k2}x_{k2} \\ 0 & 0 & 0 & x'_{k2} & y'_{k2} & 1 & -x'_{k2}y_{k2} & -y'_{k2}y_{k2} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x'_{kn} & y'_{kn} & 1 & 0 & 0 & 0 & -x'_{kn}x_{kn} & -y'_{kn}x_{kn} \\ 0 & 0 & 0 & x'_{kn} & y'_{kn} & 1 & -x'_{kn}y_{kn} & -y'_{kn}y_{kn} \end{pmatrix} \cdot \begin{pmatrix} b_{11} \\ b_{12} \\ b_{13} \\ b_{21} \\ b_{22} \\ b_{23} \\ b_{31} \\ b_{32} \end{pmatrix} = \begin{pmatrix} x_{k1} \\ y_{k1} \\ x_{k2} \\ y_{k2} \\ \vdots \\ x_{kn} \\ y_{kn} \end{pmatrix} \tag{61}$$

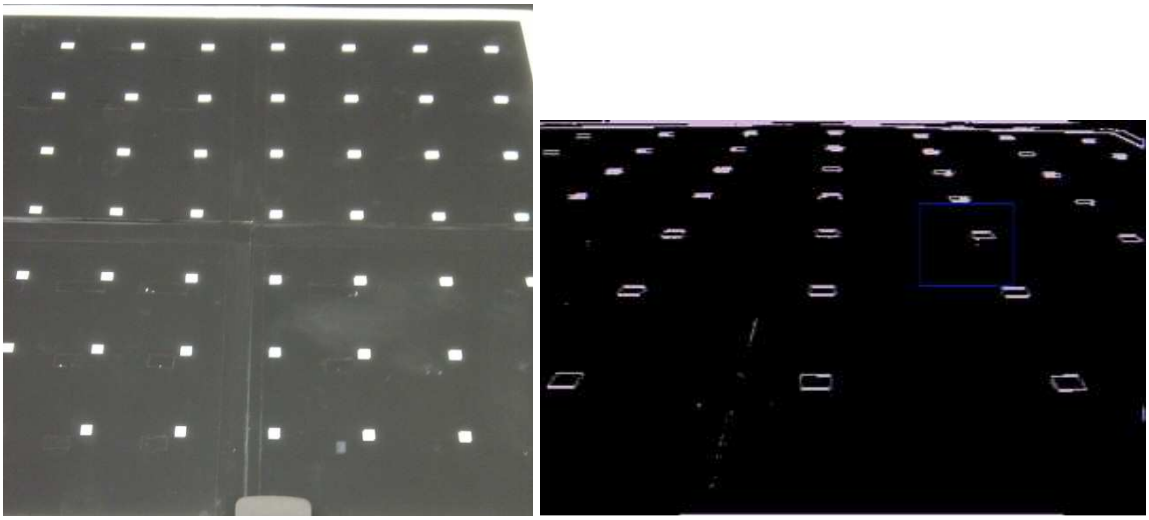

Abbildung 39: Draufsicht der Kalibrieranordnung in der Fahrzeugebene. Die Punkte in x- und y-Richtung liegen im Abstand von 17,5 cm zueinander (links). Das Sobel-gefilterte und perspektivisch verzerrte Kamerabild der Kalibrieranordnung (rechts).

Da die Fahrspurerkennung mit der Hough-Transformation zur besseren Darstellung auf dem Monitor auf Basis von Pixelkoordinaten erfolgen soll, wird das projektiv transformierte Bild als Ausschnitt der Fahrzeugebene dargestellt. Die zur Verfügung stehenden 640 x 240 Pixel in x-, bzw. y-Richtung bekommen durch eine Maßstabszuweisung von 5 mm in der Fahrzeugebene = 1 Pixel im projektiv

transformierten Bild, sodass in x-Richtung 3200 mm und in y-Richtung 1200 mm dargestellt werden.

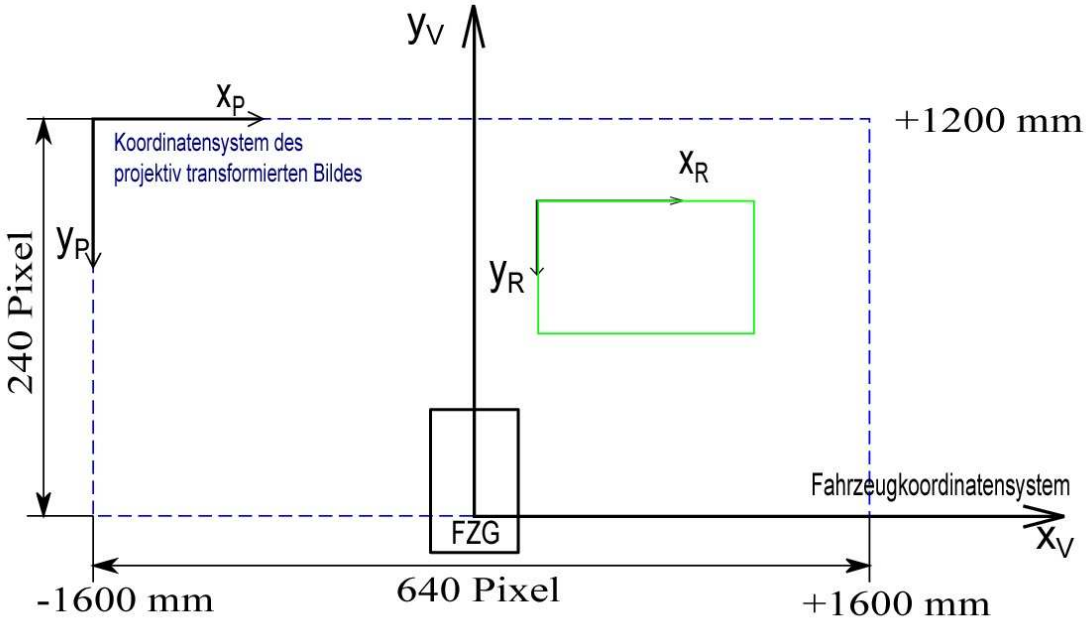

Abbildung 40: Das projektiv transformierte Bild im definierten Ausschnitt der Fahrzeugebene. Die Pixelkoordinaten im Bildkoordinatensystem (xP; yP) lassen sich in metrische Koordinaten im Fahrzeugkoordinatensystem (xV; yV) umrechnen, sodass z.B. die Dimensionen und die Position der ROI (xR; yR) in beiden Koordinatensystemen bestimmbar ist.

Die Kamera wurde so positioniert, dass sie sich genau auf dem Ursprung des Fahrzeugkoordinatensystems befindet und ca. 1200 mm der im Vorausbereich des Fahrzeugs liegenden Fahrbahn aufnimmt.

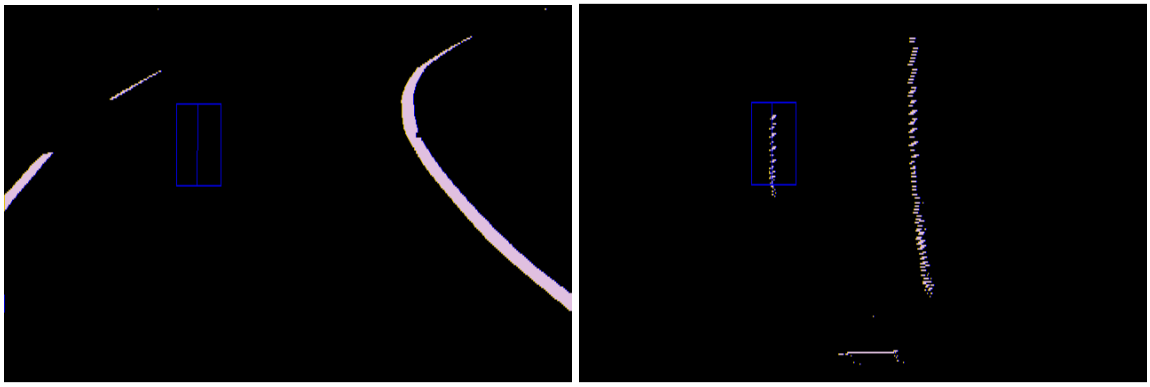

Abbildung 41: Vergleich der Kamera- und Fahrzeugdraufsicht: Die Verzerrung des Kamerabildes (links) führt zu nicht-linearen Stauchungen im oberen und Streckungen des Bildes im unteren Bereich. Eine korrekte Spurerkennung basiert auf projektiv transformierten Bildern (rechts).

Mit der neu kalibrierten Ausgleichsmatrix wurde das in [4] und [5] entwickelte Modul zur projektiven Transformation neu parametrisiert, kompiliert und in das Gesamtsystem eingefügt (vgl. Anhang E). Der Pixeldatenstrom der binarisierten und projektiv transformierten Ergebnisbilder bildet die Basis für die Fahrspurerkennung (vgl Kapitel 8 und Abbildung 41).

6 Fahrspurerkennung und –darstellung

Das Modul zur Approximation der Fahrspurmarkierung setzt sich aus einer dynamisch positionierten ROI und der Hough-Transformation zusammen, deren Aufbau und Parametrisierung im folgenden Abschnitt beschrieben wird.

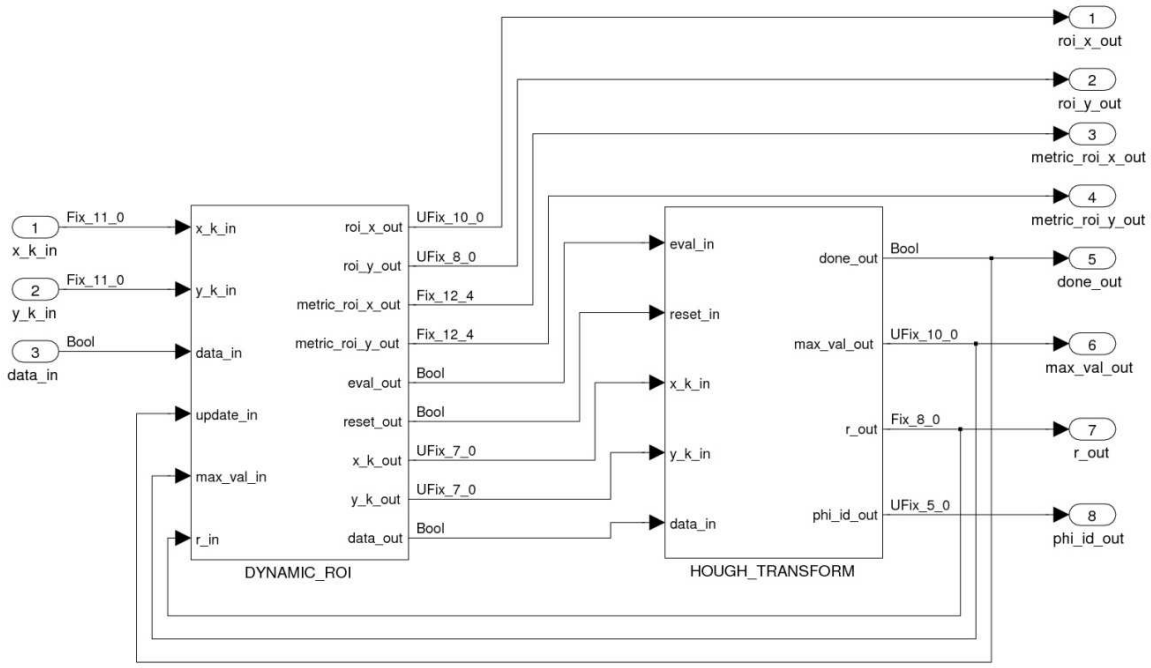

Abbildung 42: Fahrspurerkennungsmodul mit dynamischer ROI und Hough-Transformation. Die dynamische ROI reduziert den Wertebereich der projizierten Koordinaten, um die Speicheranforderungen des folgenden HT-Moduls zu reduzieren. Die Verschiebung der ROI basiert auf dem zuletzt ermittelten Abstand (r) der Geraden (vgl. Kapitel 6.2).

Beide Module werden mit der Pixelfrequenz $f_{13.5}$ betrieben und haben als Eingangswerte die projektiv transformierten x-y-Koordinaten des Pixeldatenstroms, sowie die binarisierten Farbinformationen der zugehörigen Pixel. Als Ergebnisse der Fahrspurerkennung werden folgende Werte zur Verfügung gestellt:

- Abstand r des Lotfußpunktes zum ROI-Koordinatenursprung
- ID des Winkels Φ zwischen x-Achse der ROI und dem Zeiger auf den Lotfußpunkt
- Metrische x-y-Position der ROI im Fahrzeugkoordinatensystem
- Pixel-Position der ROI im perspektivisch entzerrten Bild (zur Darstellung des Ergebnisses)
- Akkumulatorstand max_val_out des Hough-Ebenen-Maximums
- Done-Bit, welches das Bereitstehen des Approximationsergebnisses signalisiert

Für die Darstellung des Approximationsergebnisses auf einem Monitor wurde das in [4] entwickelte Overlay-Modul mit dem in [6] entwickelten Bildzwischenspeicher-Modul kombiniert, welcher die ungeordneten, projektiv transformierten Pixel-Koordinaten in einen Dual-Port-RAM einsortiert und durch Vergleich mit den Koordinaten des perspektivisch verzerrten Pixeldatenstroms des Folgebildes geordnet ausgibt [6].

Nach Erweiterung der Farbinformation auf 24 bit (RGB-Farbmodell) und der Überlagerung der ROI und des HT-Ergebnisses des letzten Approximationszyklus mit dem projektiv transformierten Bild werden die modifizierten Pixel und die zugehörigen Synchronisierungssignale dem VGA-Interface zur Verfügung gestellt (vgl. Abbildung 4).

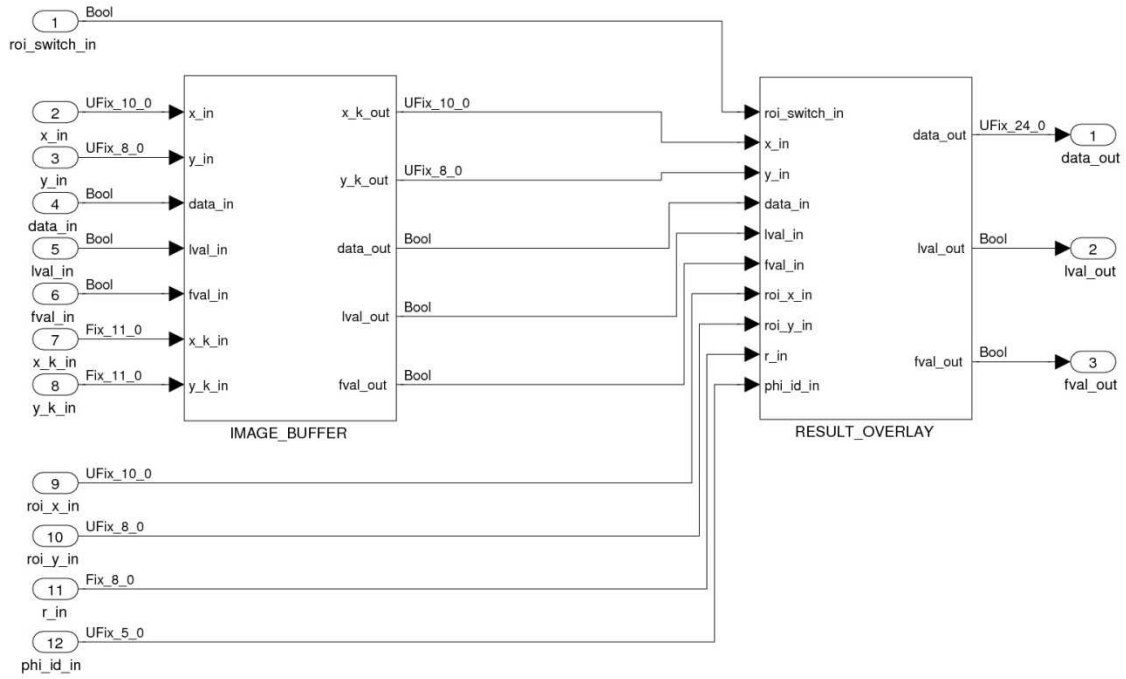

Abbildung 43: „Result Illustration"-Modul zur Darstellung der ermittelten Geradenparameter. Der projektiv transformierte, ungeordnete Pixeldatenstrom (x_k, y_k) wird sortiert in ein Dual Port RAM zwischengespeichert. Gleichzeitig wird aus diesem Speicher das vorangegangene Bild sortiert ausgelesen und mit der erkannten Geraden überlagert. Die nicht entzerrten Koordinaten (x, y) werden als HCOUNT und VCOUNT benutzt (vgl. **[6]**).

6.1 Diskretisierte Hough-Transformationen zur Fahrspurerkennung

Die Approximation der Fahrspurmarkierung durch die HT erfolgt in zwei Schritten [4]:

1. Jeder helle Bildpunkt, der sich auf der Fahrspurmarkierung befindet, wird in die Hough-Ebene übertragen. Diese Bildpunkte entsprechen in der Hough-Ebene (Lotlänge r, Winkel Φ) jeweils einer Sinuskurve.
2. In der Hough-Ebene wird der Punkt gesucht, in dem sich die meisten Sinuskurven schneiden. Dieser Punkt entspricht in der xy-Ebene der Geraden, auf dem die meisten Eingangsbildpunkte liegen (vgl. Abbildung 44).

Die Parameter einer als Geraden approximierten Fahrspurmarkierung liegen in der Hesse'schen Normalform (HNF) vor, wobei die Gerade g durch die Länge r und den Winkel Φ des Ortsvektors des Lotfußpunktes beschrieben wird (vgl. Abbildung 44).

$$g: r = \mathrm{x}\cos(\Phi) + y\ \sin(\Phi) \qquad (62)$$

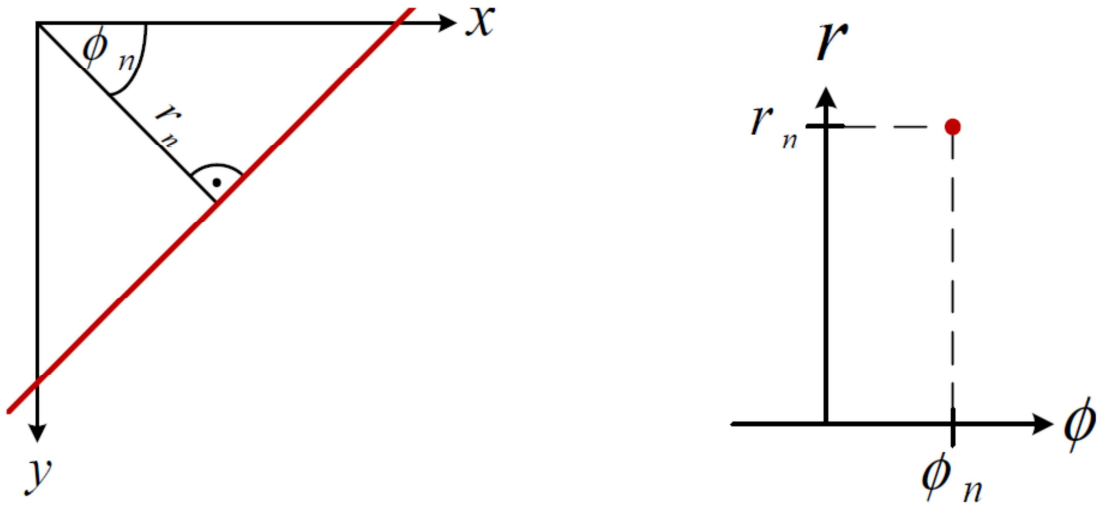

Abbildung 44: Geradendarstellung in Hesse'scher Normalform. Der Abstand r und der Winkel **Φ** des Lotfußpunktes der Geraden werden in der Hough-Ebene als Punkt dargestellt **[26]**.

Die HT verwendet zur Erkennung der Geraden eine Speichermatrix als diskrete Hough-Ebene, in der iterativ die Anzahl aller Geradenhypothesen mit den Abständen $r_i \in [r_{min}, r_{max}], i \in \mathbb{Z}$ im Winkelbereich $\Phi_j \in [\Phi_{min}, \Phi_{max}], j \in \mathbb{Z}$ akkumuliert wird. Die zur Diskretisierung der Hough-Ebene in [5] getroffenen Einschränkungen wurden folgendermaßen verändert:

- Der abgebildete Winkelbereich $\Phi \in [\Phi_{min}, \Phi_{max}]$ von -45° bis +45° wird auf den Bereich -18° bis +18° verkleinert.
- Das Winkelintervall ΔΦ von 5° wird auf 2° verkleinert.
- Zur Begrenzung der maximalen Lotlängen wird die HT auf einen 50 x 50 Pixel großen Ausschnitt des Bildes angewendet.

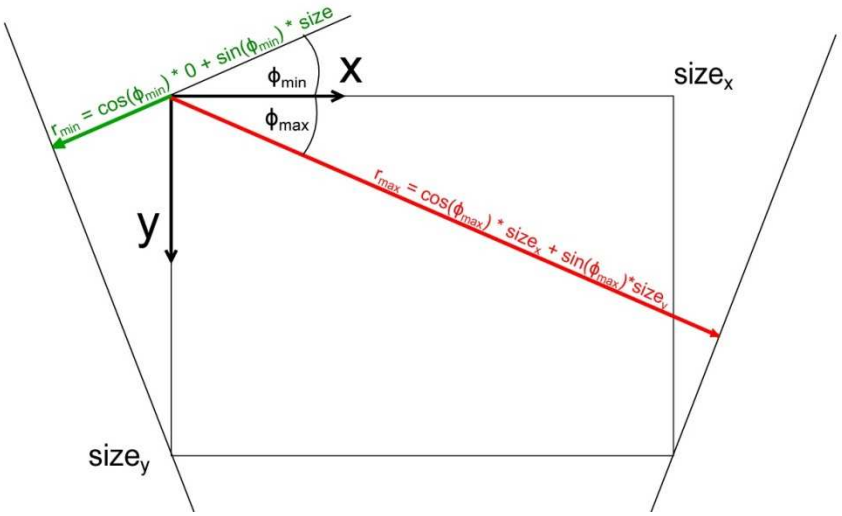

Abbildung 45: Minimaler und maximaler Abstand der Geraden. Aufgrund der Beschränkung auf den Winkelbereich Φ_{min} bis Φ_{max} reduziert sich der Wertebereich der Abstände, der zur Diskretisierung der Hough-Ebene berücksichtigt werden muss **[5]**.

Durch die gewählten Einschränkungen ergeben sich die Integerwerte der maximalen und minimalen Lotlänge r_{max} = 64 Pixel und r_{min} = -16 Pixel.

Die Gleichung (62) beschreibt die Transformation der Vordergrundbildpunkte in die Hough-Ebene, welche in folgenden Schritten erfolgt [4]:

- Für die Koordinaten eines Vordergrundbildpunktes wird zu jedem diskreten Winkel φ der Integerwert der Lotlänge r berechnet.
- Durch den Offset $|r_{min}|$ wird r zu einer Akkumulatoradresse.
- Der mit r adressierte Akkumulator wird erhöht.
- Wird dadurch der höchste Akkumulatorstand des Winkels erreicht, wird r als lokales Maximum vorgehalten.
- Nach dem alle Bildpunkte transformiert wurden, löst das *eval*-Signal die globale Maximumsuche aus.
- Das globale Maximum ist nach 19 Vergleichsoperationen ermittelt.

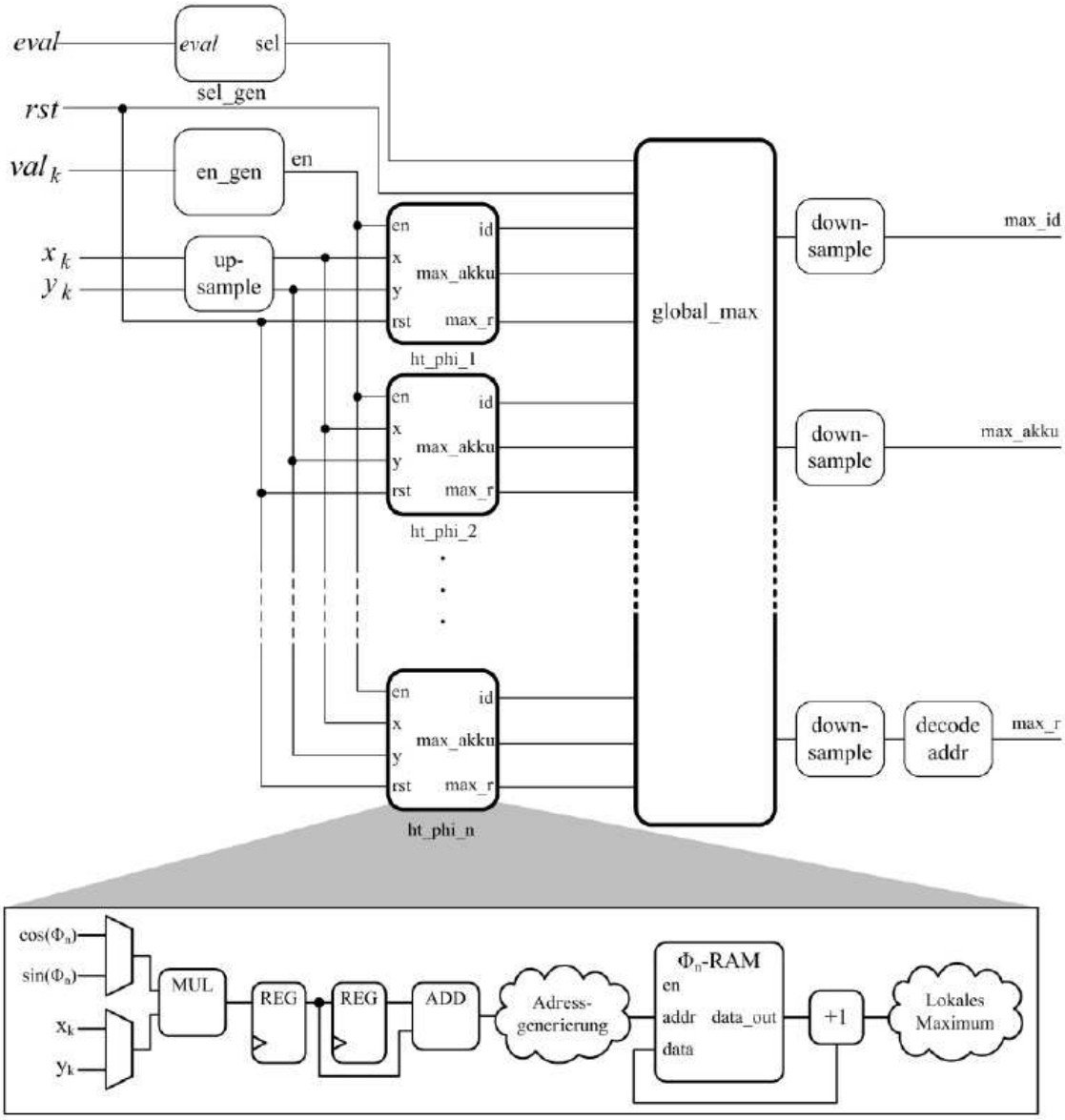

Abbildung 46: Modellierung der diskreten Hough-Transformation für den Winkelbereich Φ1 bis Φn mit Resource Sharing in einer übertakteten Pipelinestufe. Die Hough-Ebene wird durch die den Winkeln zugehörigen Akkumulatoren dargestellt, deren Maximum die erkannte Geradenapproximation ist **[5]**.

6.2 Dynamische Region-Of-Interest zur Stabilisierung der Fahrspurerkennung

Die ROI steuert das Verhalten des Fahrspurerkennungsmoduls und führt eine Datenreduktion durch Beschränkung auf definierten Teil des Binärbildes durch. Die Höhen- und Breitenparameter $size_y$ und $size_x$ sind durch die Wahl der Hough-Ebenen-Dimensionen vorgegeben. Neben der Generierung des Reset-Signals reset_out für die Hough-Ebene wird deren Evaluierung durch das Signal eval_out von der ROI gestartet.

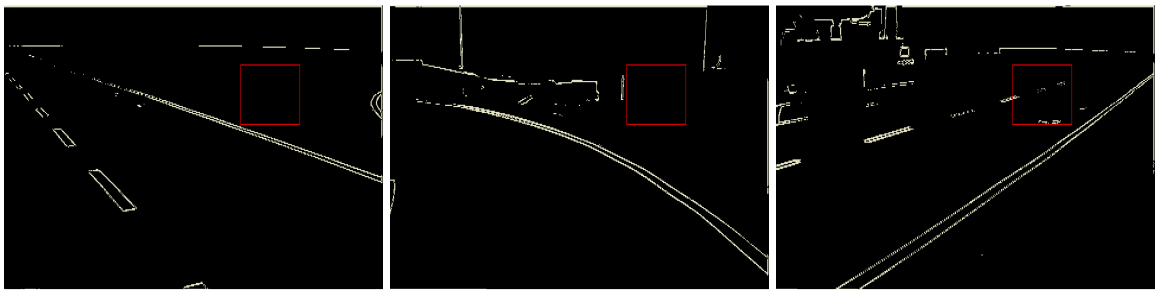

Abbildung 47: Folgen der festen Positionierung der ROI im perspektivisch verzerrten Sobel-Bild. Aufgrund der Fahrzeugbewegung in Kombination mit der Kameraperspektive treten Situationen auf, in denen keine gültigen Ergebnisse der Spurerkennung ausgewertet werden können.

Während der Analyse der Vorentwicklungsstufe fiel auf, dass die Rotation des Fahrzeugs dazu führt, dass die Fahrbahnmarkierung in Kurvenfahrten nicht innerhalb der ROI liegt und somit kein gültiges Ergebnis durch die HT bereitgestellt wird (vgl. Abbildung 47). Mit der Annahme, dass die zu verfolgende Fahrspurmarkierung stetig ist, wird die x-Position der ROI durch Rückführung des ermittelten Abstands r neu bestimmt, sodass sich die zu erkennende Fahrspurmarkierung im folgenden Auswertezyklus garantiert innerhalb der ROI befindet. Mit der ROI-Breite $size_x$ gilt:

$$ROI = \begin{pmatrix} x_{ROI} - \frac{size_x}{2} + r \\ y_{ROI} \end{pmatrix} \tag{63}$$

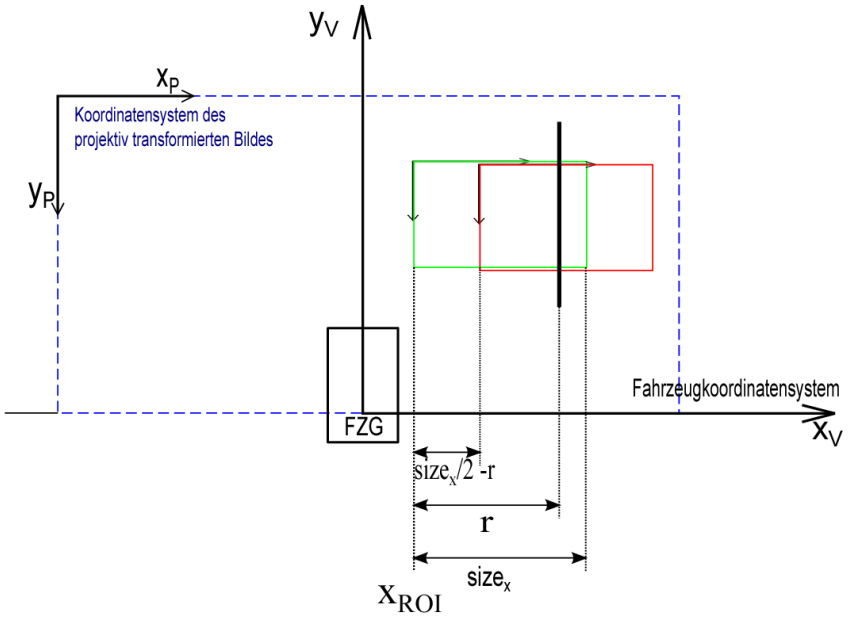

Abbildung 48: Verschiebung der ROI durch Rückführung des Abstands. Die Position der ROI x_{ROI} zur aktuellen Abtastzeit (grün) wird für das nächste Bild (rot) um die Differenz der halben ROI-Breite $size_x$ und dem ermittelten Abstand r verschoben.

7 Implementierung der Fahrspurführungsalgorithmen

Bei der Modellierung des Fahrspurführungssystems kommen die in Kapitel 3 vorgestellten mathematischen Konzepte der Berechnung des LAPs, des Fahrzeugabstands zur Fahrspurmarkierung und die Spurführungsalgorithmen, sowie die Arkustanges- und Wurzelfunktionsapproximation unter Berücksichtigung der in Kapitel 4 dargestellten Simulationsergebnisse zur Anwendung. Der Datenpfad wird als Algorithmic State Machine (ASM) mit Multizykluseigenschaften entworfen, in dem übertaktete Teilsysteme zur Beschleunigung der Berechnung unter gleichzeitiger FPGA-Ressourceneinsparung eingebettet sind. Die Multizyklusrealisierung wurde ausgewählt, da die Geradenistwerte genau einmal pro Bild ermittelt werden und durch die ROI-Beschränkung auf einen 50 x 50 Pixel großen Bildausschnitt sehr viel mehr Takte bis zum Beginn des nächsten Bildes zur Verfügung stehen, als zur Berechnung des Lenkwinkelsollwerts benötigt werden.

7.1 Funktionalität des Fahrspurführungsmoduls

Die Blackbox des „Path Following“-Moduls zur Berechnung des Lenkwinkelsollwerts umfasst folgende Eingangssignale (vgl. Abbildung 49), die vom Modul zur Fahrspurerkennung und dem Clock-Manager angelegt werden (vgl. Abbildung 4):

- **Clk**: Die mit $f_{13.5}$ = 13,5 MHz getaktete Clock wird auf f_{54} = 54 MHz übertaktet, um das Ergebnis der Divisionsfunktion innerhalb der Arkustangensapproximation, welche so konfiguriert wurde, dass weniger FPGA-Ressourcen genutzt werden, in weniger Takten zur Verfügung gestellt zu bekommen.
- **select_switch**: Zur Auswahl der Fahrspuralgorithmen Pure-Pursuit, Follow-The-Carrot, PD-Abstandsregelung und Combined Pure-Pursuit wird eine 2-bit Schalter-Kombination verwendet
- **r**, **phi_id** und **max_val**: Der Abstand r sowie die ID des Winkels Φ der erkannten Geraden, sowie das dazugehörige globale Maximum max_val in der Hough-Ebene werden zur Bestimmung der Fahrzeuglage und Berechnung des LAPs genutzt.
- **d:** Der Sollabstand der Fahrzeuglängsachse zur Fahrspurmarkierung
- **roi_x** und **roi_y**: Die x- und y-Position der ROI im metrischen Fahrzeugkoordinatensystem geht ebenfalls in die Fahrzeuglagenbestimmung ein
- **start**: Mit dem Start-Signal werden die Ergebnisse der HT und die Position der ROI aus dem „Lane Detection“-Modul als Eingangswerte synchronisiert übernommen und anschließend die Berechnung des Lenkwinkelsollwerts entsprechend der Algorithmenauswahl (select_switch) gestartet.

Am Ausgang des Moduls stehen nach abgeschlossener Lenkwinkelbestimmung folgende Signale an:

- **lap_x** und **lap_y**: Die LAP-Koordinaten im metrischen Fahrzeugkoordinatensystem werden zum Datenlogging bereitgestellt.
- **alpha**: Der Lenkwinkelsollwert α wird an das nachfolgende PWM-Modul angelegt.
- **done**: Das done-Bit signalisiert einen neuen Lenkwinkelsollwert.

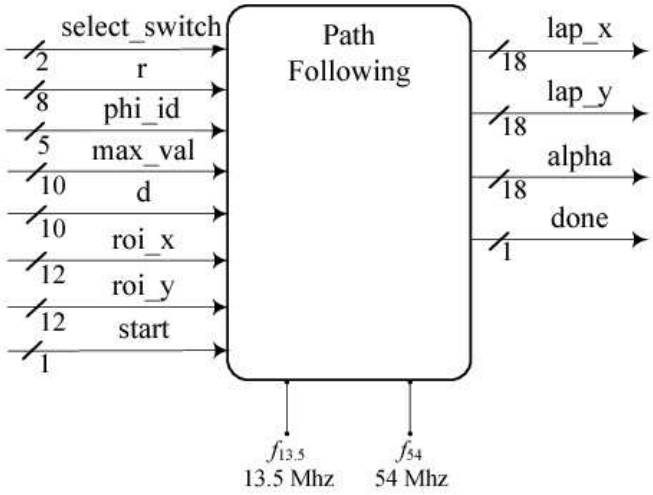

Abbildung 49: Blackbox des Path-Following-Moduls mit zwei Takteingängen.

Das Fahrspurführungssystem wurde zur Vereinfachung der Funktionstests in mehrere Komponenten unterteilt, deren Aufgabe im Folgenden beschrieben wird (vgl. Abbildung 50). Alle Berechnungen erfolgen im metrischen Koordinatensystem mit der Längeneinheit cm:

- **UPDATE**: Diese Komponente dient der Synchronisierung der ROI-Position und der Geradenparameter, deren Güte durch das Signal **max_val** angezeigt wird: Ist der Wert größer als eine vorgegebene Schwelle, werden der Abstand **r** und die ID des Winkels **Φ** übernommen. Durch Rechtsschieben (1 Pixel ≡ 0,5 cm) wird das in Pixeln angegebene r als metrischer Abstand den Folgekomponenten zur Verfügung gestellt. Mit der ID des Winkels Φ werden Sinus- und Cosinus-Tabellen, implementiert als Single-Port-RAM, adressiert und deren Speicherinhalt am Komponentenausgang angelegt.

- **LAP**: Steuer- und Datenpfad der ASM zur Berechnung des LAPs: Diese Komponente nutzt die Wurzelfunktionsapproximation, welche auch durch einen Multizyklusdatenpfad realisiert wurde (vgl. Abbildung 70).

- **ARCTAN**: Komponente zur Kapselung der Arkustangensapproximation mit übertakteter Divisionsfunktion (vgl. Abbildung 51).

- **PD**: RTL-Modell des PD-Reglers mit integrierter Lenkwinkelbegrenzung gemäß den Gleichungen (27) und (28).

- **MODE_SELECT**: Auswahlfunktionsblock zum Umschalten zwischen den Lenkwinkelsollwerten der verschiedenen Fahrspuralgorithmen über User-Switches.

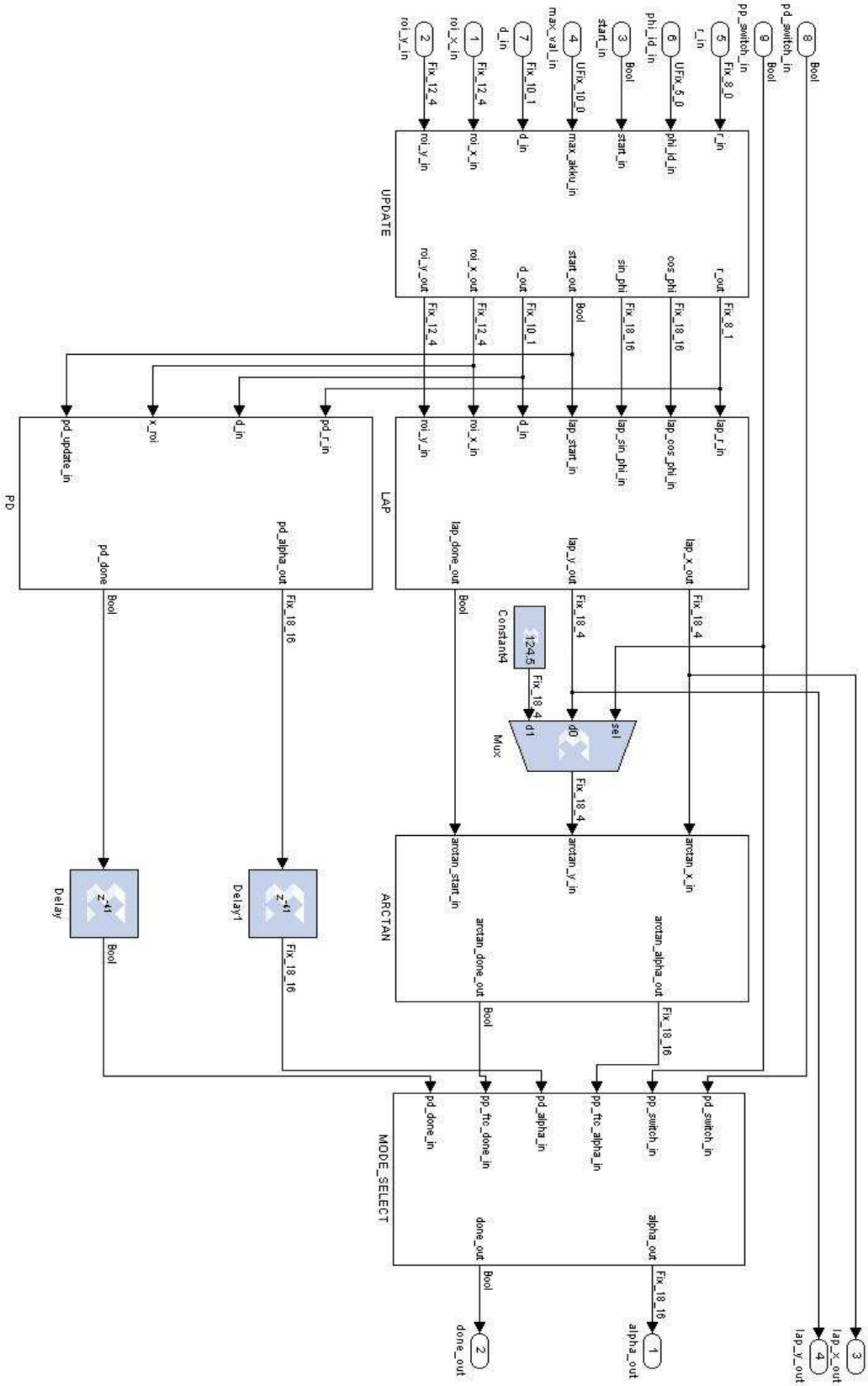

Abbildung 50: Fahrspurführungssystem mit Prozessorelementen zur LAP- und ARCTAN-Berechnung, sowie PD-Regelung. Aus den Geradenparametern wird der LAP bzw. der Istabstand des Fahrzeug berechnet und ein Lenkwinkelsollwert über die Spurführungsalgorithmen Follow-The-Carrot, Pure-Pursuit und PD-Regelung bestimmt.

7.2 Realisierung der Algorithmic State Machines am Beispiel der Arkustangensfunktion

Der ASM-Entwurf ist eine Methode zum Designen von Zustandsautomaten, wobei anstelle eines Zustandsdiagramms ein informelleres, aber leichter verständlicheres ASM-Chart genutzt wird, das die sequentiellen Operationen zu beschreibt. Die ASM zur Berechnung des Arkustangens unterteilt sich in einen Steuer- und einen Datenpfad, der sich auf die Nutzung von genau einem Multiplizierer und Addierer beschränkt, und wird in folgenden Schritten entwickelt [28]:

1. Umwandlung des Algorithmus (vgl. Gleichungen (34) bis (37)) in Pseudocode, der die Sequenz der durch die Arithmetik-Blockbeschränkung resultierenden Zwischenschritte widerspiegelt (vgl. Listing 1).
2. Umwandlung des Pseudocodes in ein ASM-Chart (vgl. Abbildung 53).
3. Design des Datenpfades basierend auf der Signal-Lifetime-Tabelle und Register-Sharing (vgl. Tabelle 4 und Tabelle 6).
4. Ergänzung des ASM-Charts um die Steuersignale resultierend aus dem Register-Sharing (vgl. Abbildung 53).
5. Entwurf der Steuerlogik als Zustandsautomat anhand des ergänzten ASM-Charts (vgl. Anhang B).

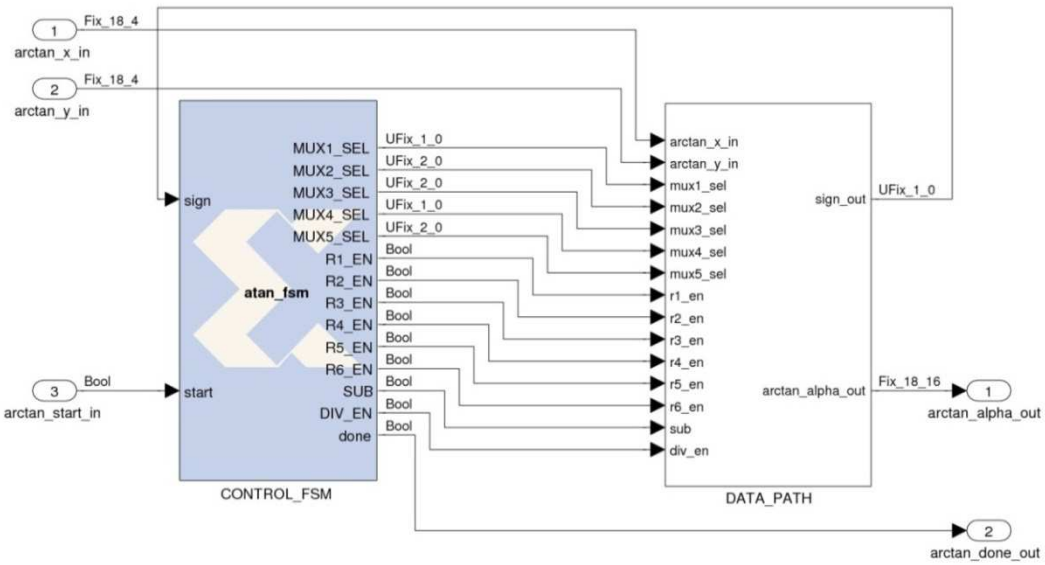

Abbildung 51: Kopplung des Daten- und Steuerpfads der ARCTAN-ASM. Je nach Vorzeichen der Eingangswerte schaltet die Steuer-FSM der Arkustangens-ASM die Register und Multiplexer des Datenpfads, sodass zusätzliche Multiplikationen zur Korrektur des Vorzeichens vermieden werden.

Der nachfolgende MatLab-Code beschreibt den Algorithmus zur Umsetzung der Arkustangensapproximation nach Gleichungen (34) bis (37), wobei der in [23] vorgestellte atan2-Algorithmus zu einem atan-Algorithmus umgewandelt wurde, dessen Ergebnis angle auf den Wertebereich $-\pi$ bis π abgebildet wird.

Durch Ausklammern des Ausdrucks „0.9817 * r“ wird die quadrantenabhängige Berechnung vereinfacht, sodass die Bildung des Arkustanges für Punkte im Quadranten I und III sich von der für Punkte im Quadranten II und IV nur durch einen Operatortausch unterscheiden (vgl. Listing 1 und Tabelle 4).

```
function [ angle ] = arctan(y, x)
% approximates arctan(y/x)
    if(y>0)
        if(x>0)         % Quadrant I, y > 0 && x > 0
            r = (x-y)/(x+y);
            %angle = 0.1963 * r^3 - 0.9817 * r + pi/4;
            angle = 0.9817 * r * (0.2 * r^2 - 1) + pi/4;
        else            % Quadrant II, y > 0 && x < 0
            r = (x+y)/(x-y);
            %angle = -0.1963*r^3 + 0.9817*r - pi/4;
            angle = 0.9817 * r * (1 - 0.2 * r^2) - pi/4;
        end
    elseif (y<0)
        if (x>0)        % Quadrant IV, y < 0 && x > 0
            r = (x+y)/(x-y);
            %angle = -0.1963*r^3 + 0.9817*r - pi/4;
            angle = 0.9817 * r * (1 - 0.2 * r^2) - pi/4;
        else            % Quadrant III, y < 0 && x < 0
            r = (x-y)/(x+y);
            %angle = 0.1963*r^3 - 0.9817*r + pi/4;
            angle = 0.9817 * r * (0.2 * r^2 - 1) + pi/4;
        end
    else
        angle = 0;
    end
end
```

Listing 1: MatLab-Code zur Berechnung des Arkustangens mit dem Wertebereich -π/2 bis π/2

Durch Nutzung einer parallelen Multiplizierer- und Addierereinheit können die 12 Zwischenschritte in 9 Zuständen abgearbeitet werden.

Zustand	Variable		Operand 1	Operator	Operand 2	Bedeutung
Z0	sign	←	MSB(x)	XOR	MSB(y)	Vorzeichenermittlung
	x	←	x_in	SLL 12		Linksschieben des Dezimalpunktes
	y	←	y_in	SLL 12		Linksschieben des Dezimalpunktes
Z1	t1	←	x	-/+	y	Operatortausch
Z2	t2	←	x	+/-	y	Operatortausch
Z3	t3	←	t1	/	t2	
Z4	t4	←	t3	*	t3	
Z5	t5	←	t4	*	0.2	
Z6	t6	←	t3	*	0.9817	
	t7	←	t5 | 1	-	1 | t5	Operandentausch
Z7	t8	←	t7	*	t6	
Z8	alpha	←	t8	+ | -	Pi/4	Operatortausch

Tabelle 4: Aufteilung und Zustandszuordnung der einzelnen Schritte der Arkustangensapproximation nach **Listing 1**

Aufgrund der Berechnung in Fixpoint-Arithmetik wurde eine Wertebereichsbetrachtung durchgeführt und die Vektorbreiten aller Variablen und Zwischenergebnisse auf das Format Fix_18_16 festgelegt, sodass Werte von -2 bis +2 – 1 LSB mit einer Genauigkeit von 1 LSB = 2^{-16} dargestellt werden können. Das Verschieben des Dezimalpunktes der

Variablen x und y hat keinen Einfluss auf das Zwischenergebnis r = t3, da es einer kürzbaren Erweiterung des Zählers und Nenners entspricht:

$$r = \frac{x+y}{x-y} = \frac{\frac{x}{2^{-12}} + \frac{y}{2^{-12}}}{\frac{x}{2^{-12}} - \frac{y}{2^{-12}}} = \frac{\frac{1}{2^{-12}}(x+y)}{\frac{1}{2^{-12}}(x-y)} = r \tag{64}$$

Variable	Wertebereich RESULT	Precision OP1	Operator	Precision OP2	Full Precision RESULT	User defined Precision
sign	BOOL	BOOL	XOR	BOOL	BOOL	BOOL
x	<=2	FIX_18_4	SLL 12		FIX_18_16	FIX_18_16
y	<=2	FIX_18_4	SLL 12		FIX_18_16	FIX_18_16
t1	<=2	FIX_18_16	-/+	FIX_18_16	FIX_19_16	FIX_18_16
t2	<=2	FIX_18_16	+/-	FIX_18_16	FIX_19_16	FIX_18_16
t3	<=1	FIX_18_16	/	FIX_18_16	FIX_18_16	FIX_18_16
t4	<=1	FIX_18_16	*	FIX_18_16	FIX_36_32	FIX_18_16
t5	<=1	FIX_18_16	*	FIX_18_16	FIX_36_32	FIX_18_16
t6	<=1	FIX_18_16	*	FIX_18_16	FIX_36_32	FIX_18_16
t7	<=2	FIX_18_16	-	FIX_18_16	FIX_19_16	FIX_18_16
t8	<=2	FIX_18_16	*	FIX_18_16	FIX_36_32	FIX_18_16
alpha	<=2	FIX_18_16	+/-	FIX_18_16	FIX_19_16	FIX_18_16

Tabelle 5: Quantisierung der Zwischen- und Endergebnisse. Die Vektorbreiten der Zwischenergebnisse ergeben sich durch Zuschneiden der Full-Precision-Ergebnisse auf den tatsächlich genutzten Wertebereich der einzelnen Rechenschritte.

	z0	z1	z2	z3	z4	z5	z6	z7	z8	Register	ALU
sign		x	x	x	x	x	x	x	x	R1	XOR
x		x	x							R2	
y		x	x							R3	
t1			x	x						R4	ADD/SUB
t2				x						R3	ADD/SUB
t3					x	x	x			-	DIV
t4						x				R5	MUL
t5							x			R5	MUL
t6								x		R5	MUL
t7								x		R4	ADD/SUB
t8									x	R5	MUL
alpha	x	x	x	x	x	x	x	x	x	R6	ADD/SUB

Tabelle 6: Register-Sharing der Arkustangens-ASM. Die externe Divisionsoperation stellt ein eigenes Register zur Ausgangssynchronisation zur Verfügung, sodass dessen Ergebnis nicht zwischengespeichert werden muss.

Neben der Einsparung von FPGA-Ressourcen durch Festlegen der Vektorbreiten werden die Zwischenschritte auf ein Register-Sharing analysiert. Für die Berechnung der 12 Zwischen- und Endergebnisse werden nur 6 Register genutzt, da diese nur so lange gespeichert werden, bis sie in folgenden Schritten nicht mehr benötigt werden. Dabei wird die Zuweisung der genutzten Register zu genau einem Arithmetik-Block berücksichtigt, um lange Signallaufzeitpfade durch zusätzliche Multiplexer hinter den Arithmetik-Blöcken zu vermeiden (vgl. Tabelle 5 und Tabelle 6).

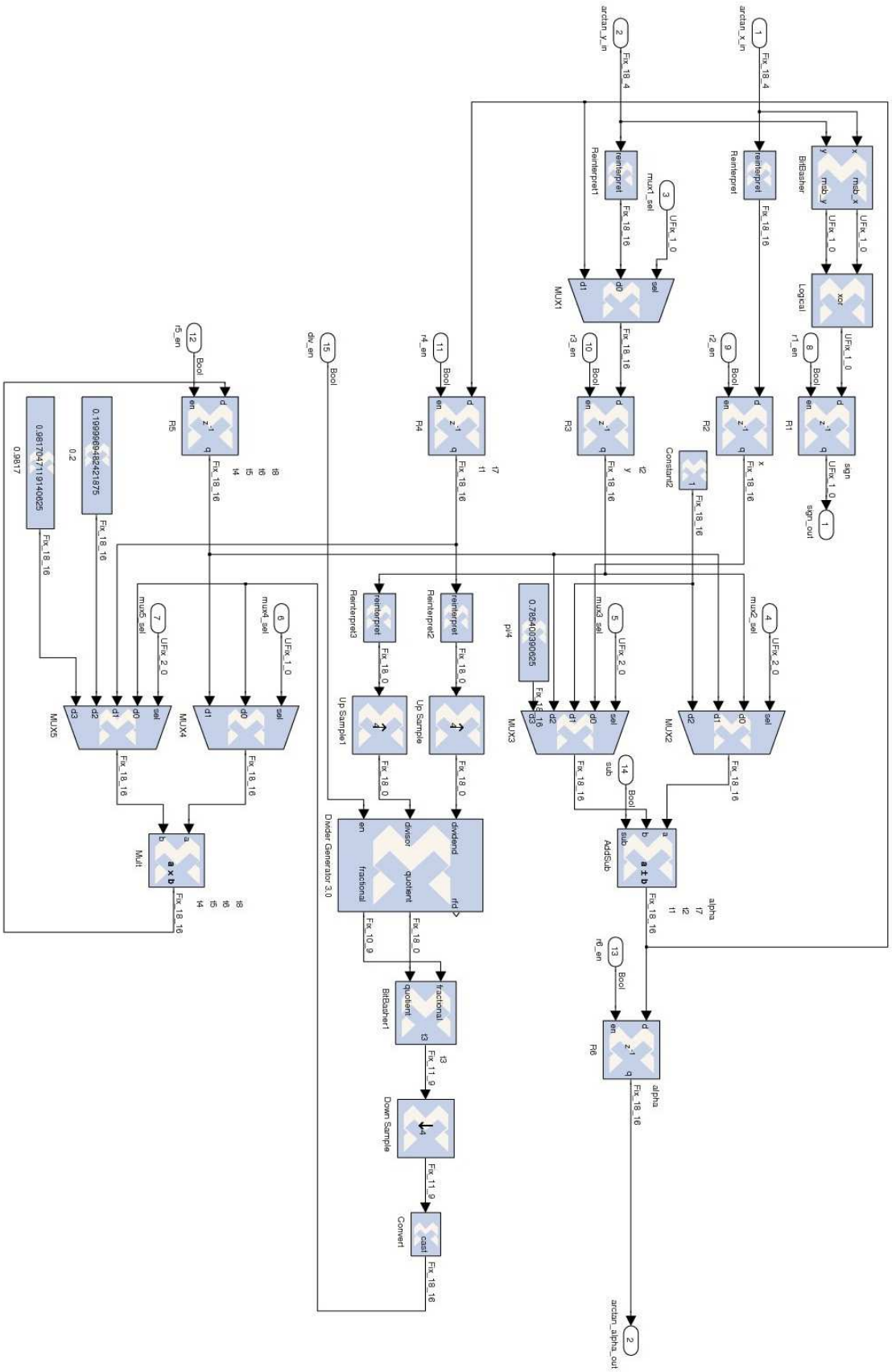

Abbildung 52: Datenpfad der Arkustangens-ASM. Das Ergebnis wird mithilfe von jeweils einem Addierer, Multiplizierer und Dividierer berechnet ist auf den Wertebereich $-\pi/2$ bis $+\pi/2$ beschränkt.

Nach der RTL-Modellierung des Datenpfades unter Berücksichtigung des Register-Sharings und der Signalvektorbreiten das ASM-Chart um die Steuersignale ergänzt (vgl. Abbildung 52 und Abbildung 53). Neben der Sequenz der Berechnungsschritte bildet dieses auch die zu schaltenden Steuersignale ab und die stellt die Grundlage für das Programmieren der Steuer-FSM dar (vgl. Anhang B).

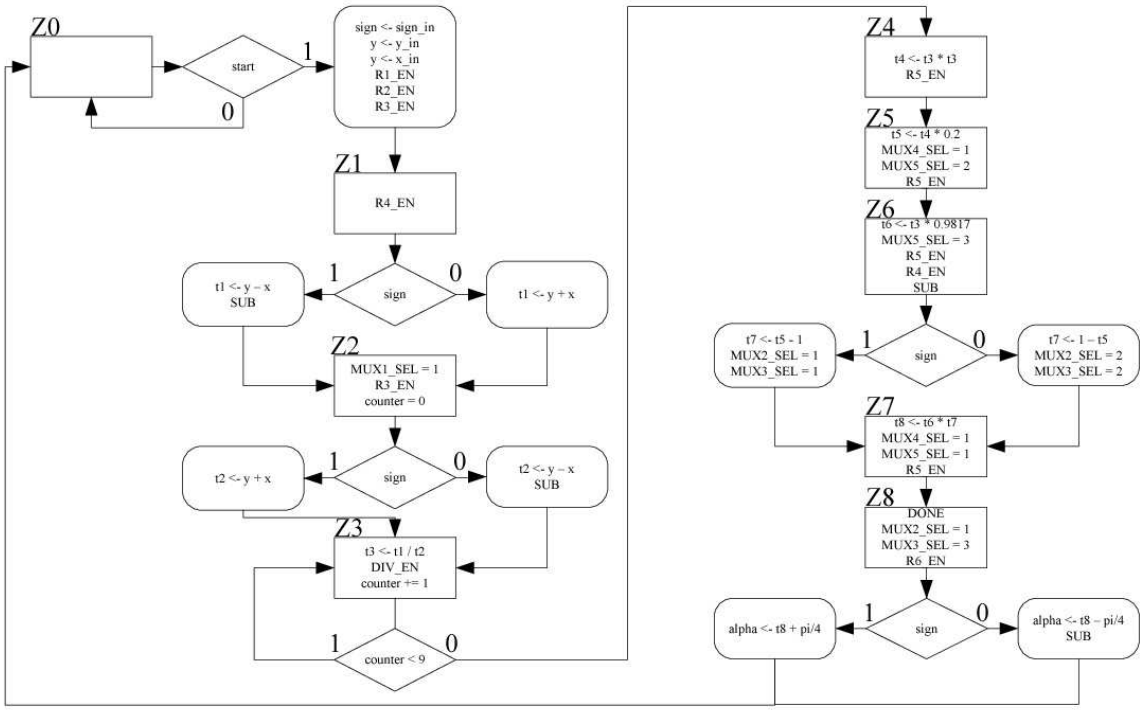

Abbildung 53: Detailliertes ASM-Chart der Arkustangens-ASM mit Berechnungsschritten und Steuersignalen. Das Ergebnis eines Rechenzyklus liegt nach 17 Takten vor, da ein externes Divisionsmodul genutzt wird.

7.3 PWM-Modul zur Umsetzung der Lenkwinkelstellgröße

Die Schnittstelle zwischen digitaler Steuerlogik und dem analogen Servomotor wird durch das mit einer Frequenz $f_{13.5}$ = 13,5 MHz getaktete PWM-Modul gebildet, welches aus der Lenkwinkelstellgröße α einen Sollwert für die Dauer des Duty Cycles ermittelt und diesen dann mit Zähler- und Komparator-Logik umsetzt (vgl. Abbildung 55 und Abbildung 56).

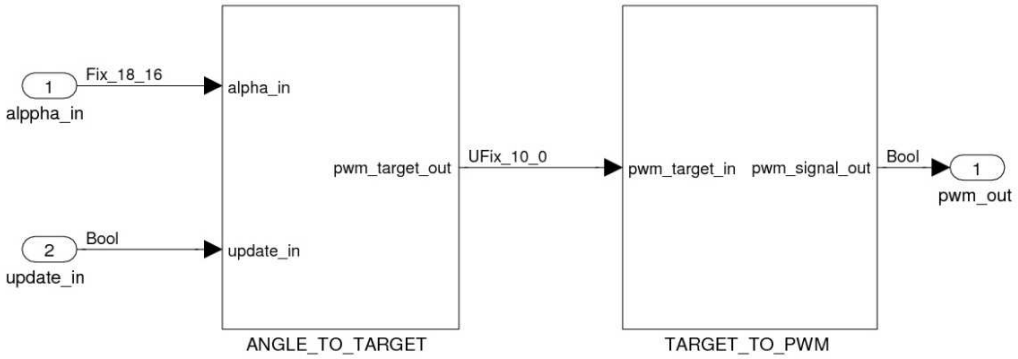

Abbildung 54: PWM-Modul zur Umsetzung der Lenkwinkelstellgröße: Der anzulegende Lenkwinkel wird in einen Vergleichswert für die Dauer des Duty Cycles des PWM-Signals umgerechnet.

Der im Modellfahrzeug genutzte Servomotor wird mit Pulslängen zwischen 1 und 2 ms betrieben, die mit einer Wiederholfrequenz f_{pwm} = 50 Hz angelegt werden. Ein Duty Cycle von 1 ms resultiert in einem Lenkwinkel von -20°, 1,5 ms entsprechen 0° und 2 ms führen zu einem Lenkwinkel von +20°.

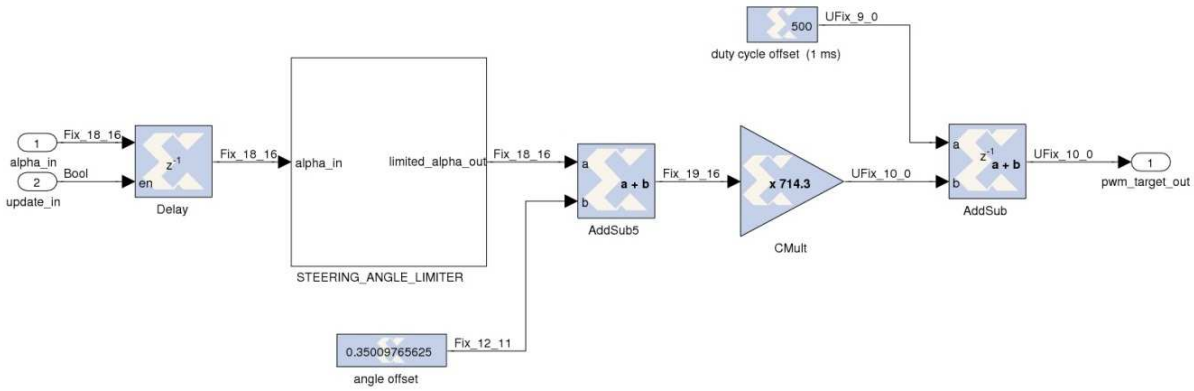

Abbildung 55: Datenpfad zur Berechnung des PWM-Vergleichswerts. Die eingehenden Lenkwinkel aus Pure-Pursuit, Follow-The-Carrot und Combined Pure-Pursuit werden auf den Stellbereich des Servomotors begrenzt.

Im Submodul „Angle to Target“ wird der Lenkwinkelbereich in 500 Stellschritte aufgelöst, sodass sich ein Winkel von 0,08° pro Schritt ergibt. Der in Radiant angegebene Lenkwinkel wird um den Offset α_{offset} = +0,35 rad ≡ +20° verschoben und durch Multiplikation mit der Konstanten c = 714,3 auf den Wertebereich 0 bis 500 abgebildet. Werte außerhalb des Bereichs -20° bis +20° werden über Vergleichsoperationen auf den Stellbereich begrenzt. Zusammen mit dem Duty Cycle Offset dc_{offset} = 500 ≡ 1 ms wird das Pulslängenintervall von 1 bis 2 ms realisiert, sodass sich ein Vergleichswert pwm_{target} von 500 bis 1000 ergibt (vgl. Abbildung 55):

$$pwm_{target} = (\alpha_{limited} + \alpha_{offset}) \cdot c + dc_{offset} \qquad (65)$$

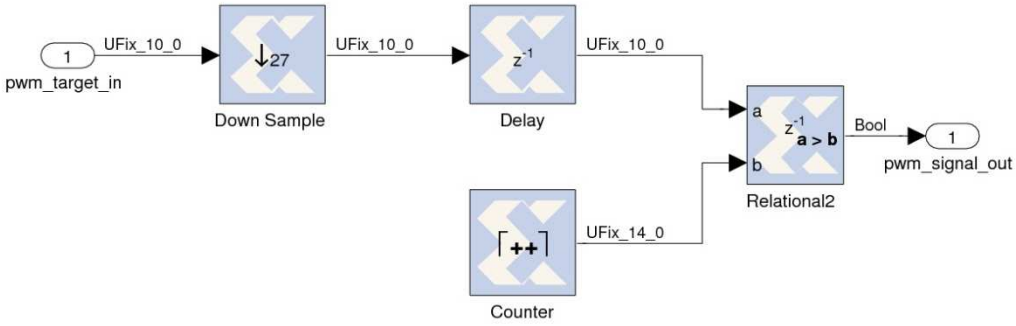

Abbildung 56: Umwandlung des Vergleichswerts in ein PWM-Signal. Die eingehenden Sollwerte werden von 13,5 MHz auf 500 kHz downgesampled. Mittels Zähler- und Komparator-Logik wird der Sollwert auf die Dauer des PWM Duty Cycles von 1 bis 2 ms mit einer Widerholfrequenz von 50 Hz abgebildet.

Zur Umwandlung in einen Impuls wird der Vergleichswert pwm_{target} im Submodul „Target to PWM „ von $f_{13.5}$ auf $f_{0.5}$ = 500kHz downgesampled und mit dem Zählerstand einem ebenfalls mit $f_{0.5}$ getaktetem Zählers verglichen. Durch Limitierung auf den Zählbereich 0-10000 ergibt sich die erforderliche PWM-Periodendauer von $T_{PWM} = \frac{1}{50\ Hz} = 20\ ms$ (vgl. Abbildung 56).

8 Ergebnisanalyse der Fahrspurerkennung und –führung

Für die Fahrspurerkennung werden die Zwischen- und Endergebnisse der aufbereiteten Bilder erläutert und analysiert (vgl. Kapitel 2 und 5). Zudem werden die RTL-Modelle der Fahrspurführungsmodule validiert, indem deren Ergebnisse mit denen der Software-Implementierung des Simulators gegenübergestellt werden (vgl. Kapitel 3, 4 und 7). Nach Vorstellung des FPGA-Ressourcenbedarfs des SoC's werden weitere Technologieerprobungen vorgeschlagen.

8.1 Modifikation der BV-Kette

Die feste Grauwertschwelle führt durch Lichtreflektionen der Fahrbahn zu Störungen im oberen Bereich, da deren Grauwerte oberhalb des Schwellwerts liegen (vgl. Abbildung 57, links). Der Vorteil der Binarisierung mit prozentualer Schwelle auf den tatsächlich auftretenden Grauwertebereich ist, dass sie unempfindlicher gegenüber starker und auch schwacher Lichtintensität ist. Das Verschieben der den Fahrbahnmarkierungen zugehörigen Grauwerte in den hellsten Bereich eliminiert Störungen durch Reflektionen auf dem Fahrbahnuntergrund (vgl. Abbildung 57, rechts).

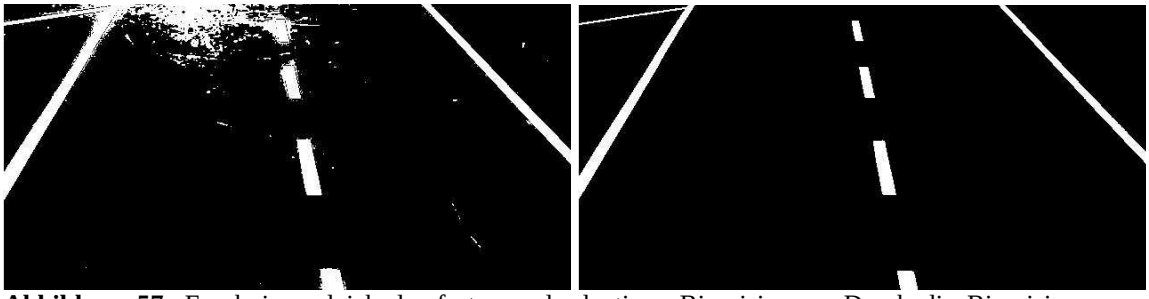

Abbildung 57: Ergebnisvergleich der festen und adaptiven Binarisierung. Durch die Binarisierung nach linearer Grauwertskalierung mit prozentualer Grauwertschwelle p = 0,875 (rechts) entstehen bei hoher Lichtintensität weniger Störungen aufgrund von Reflektionen, als bei der Binarisierung mit fester Grauwertschwelle ($g_{Schwelle}$=125, links)

Das Ergebnisbild der Binarisierung wird mit einem quadratischen 3x3 Erosions-Faltungskern zur Verschmälerung der Fahrspurmarkierung verarbeitet (vgl. Abbildung 58). Die Fahrspurmarkierung wird an jeder Seite um 2 Pixel verkleinert und führt zu eindeutigen Maxima in der Hough-Ebene und somit zu genaueren Abständen r und Winkeln Φ der erkannten Geraden (vgl. Abbildung 59, sowie Kapitel 5.1 und 5.2).

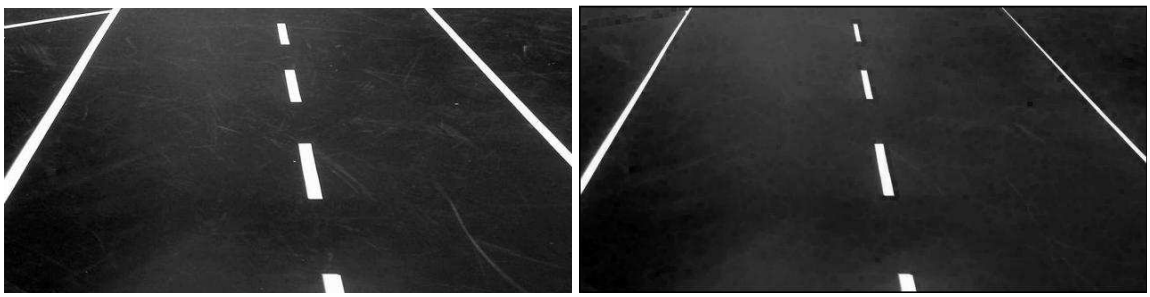

Abbildung 58: Wirkung des Erosionsfilters. Die Fahrspurmarkierungen im grauwertskalierten Bild (links) wurden durch Anwendung eines quadratischen 3x3 Erosions-Faltungskern verschmälert, sodass diese in der HT zu weniger Maxima und somit eindeutigen Geradenparametern r und Φ führt.

Die Anwendung der HT erfolgt nach der perspektivischen Entzerrung durch Projektive Transformation (PT) des Binärbildes auf eine pro Bild dynamisch festgelegte ROI. Diese wird so positioniert, dass die erkannte Gerade verfolgt wird (vgl. Abbildung 59 (rechts), blau).

Vor der Umstellung der Bildaufbereitungsmodule führte die Kombination aus Sobel-Filterung mit fester Binarisierungsschwelle und fest positionierter ROI besonders in Kurvenfahrten zu widersprüchlichen, und damit für die Spurführung unbrauchbaren Geradenparametern (vgl. Abbildung 60 (links), sowie Kapitel 2.1 und 5).

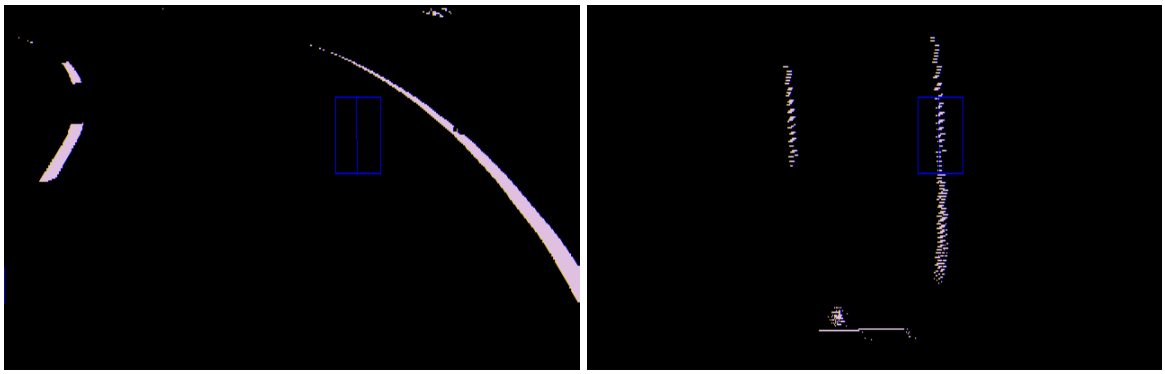

Abbildung 59: Ergebnis der projektiven Transformation eines grauwertskalierten und mit einer 3x3 Faltungsmatrix erodierten Bildes. Die Anwendung der HT im Bereich der dynamisch platzierten ROI (blau) führt zu Geraden, die die Fahrspurmarkierung sehr genau approximieren.

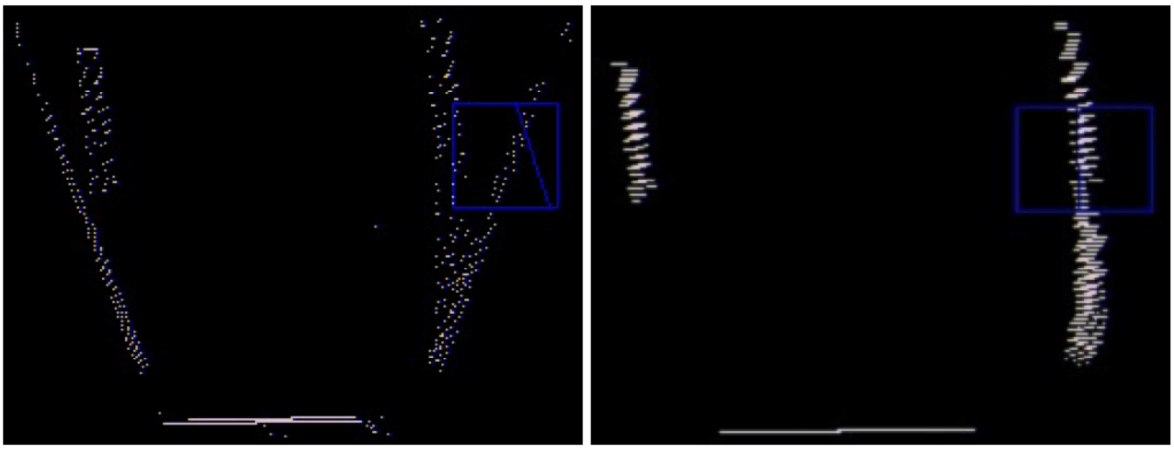

Abbildung 60: Vergleich der Projektiven Transformation auf Sobel-gefiltertem Bild und nicht-kantengefiltertem Bild. Durch Projektion des Sobelbildes entsteht eine pixelhafte Anordnung, die in Kombination mit der festen ROI (blau) zu fehlerbehafteten und oszillierenden Geradenergebnissen führt (links). Durch Erosion und Nutzung einer dynamischen ROI werden eindeutige und korrekte Geraden approximiert (rechts).

8.2 Validierung der Fahrspurführungsmodule

Zur Analyse der Genauigkeit der entwickelten RTL-Modelle zur Look-Ahead-Point-Berechnung und Arkustangensapproximation werden diese mit der im MatLab-Simulator genutzten Software-Implementierung verglichen (vgl. Abbildung 61). Über MatLab-Skripte werden Eingangsstimuli erzeugt und die Ergebnisse für eine Gegenüberstellung mit dem Fahrzeug-Simulator im Workspace gespeichert (vgl. Kapitel 4, Anhang B).

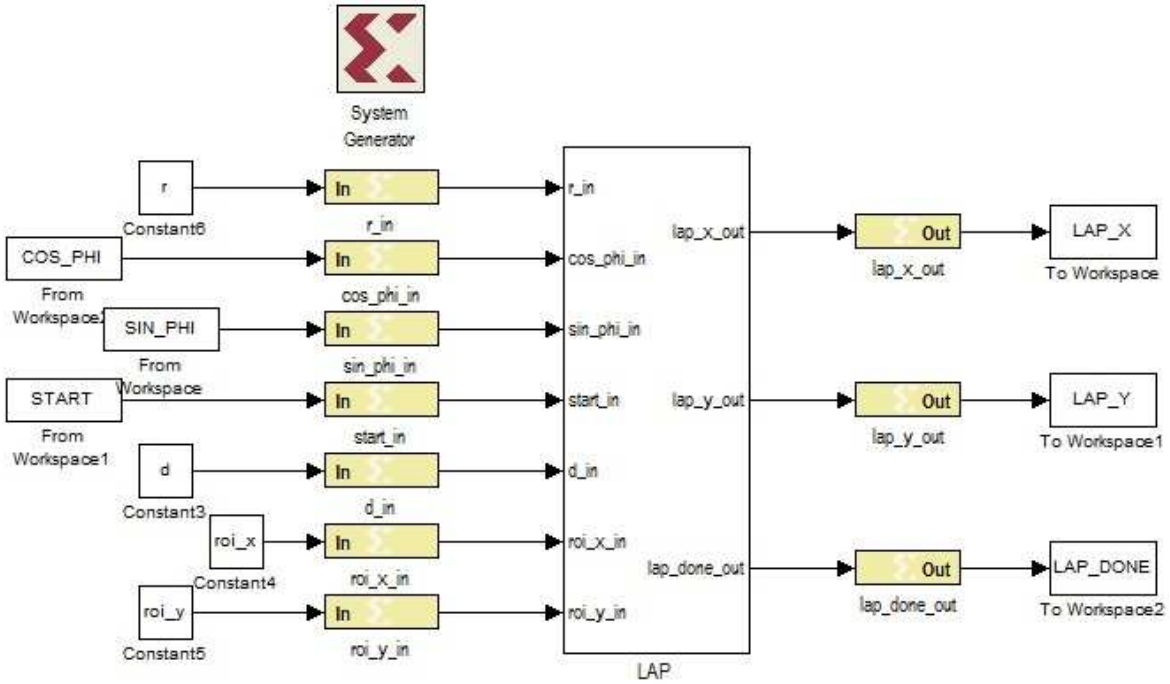

Abbildung 61: Simulation des RTL-Modells der LAP-Berechnung. Eingangsgrößen: konstanter Istabstand r, Sollabstand d, feste ROI-Position (roi_x, roi_y), Sinus- und Kosinuswerte des Winkelintervalls Φ.

Die durch das RTL-Modell berechneten LAP-Koordinaten weichen infolge der Umwandlung in das Q-Format von denen in Software berechneten Koordinaten in Fließkommadarstellung ab. Mit der gewählten Quantisierung ergibt sich eine euklidische Distanz von weniger als 3,5 mm (vgl. Abbildung 62).

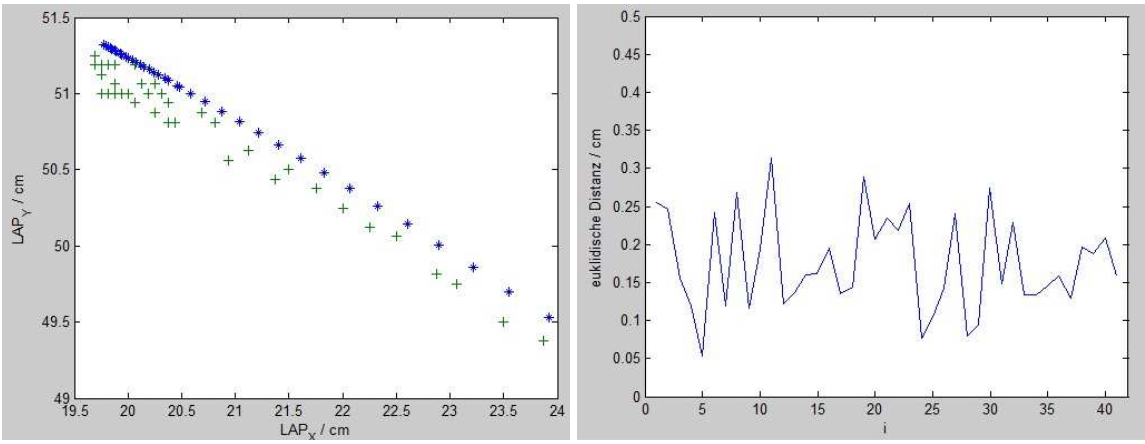

Abbildung 62: Gegenüberstellung RTL-Modell (+) und Fahrzeug-Simulator (*). Geradenwinkel Φ [-20°, +20°], ΔΦ = 1°, Istabstand r = 10 cm, ROI-Position (roi_x = -10 cm, roi_y = 55 cm), Sollabstand d = 0 cm. Euklidische Distanz < 3,5 mm.

Die resultierenden LAP-Koordinaten werden als Eingangsstimuli für die Analyse des ARCTAN-RTL-Modells verwendet, um die Pure-Pursuit-Lenkwinkel nach Gleichung (17) zu bestimmen (vgl. Abbildung 63, sowie Kapitel 3.4 und 7.2).

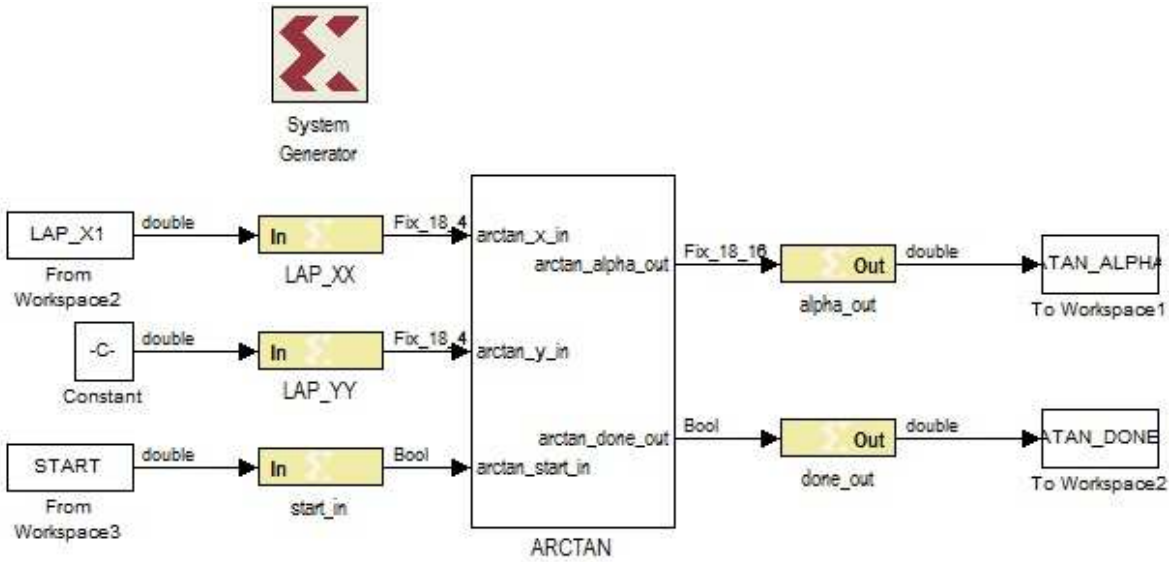

Abbildung 63: Simulation des RTL-Modells der atan-Approximation durch Pure-Pursuit-Lenkwinkelbestimmung nach Gleichung **(17)**. Eingangsgrößen: x-Koordinate des LAP, Konstante C = $LAD^2/2L$ mit konstanter LAD und konstantem Achsabstand L.

Durch die Approximation des Arkustangens in Integer-Arithmetik ergeben sich erneut Abweichungen von der Fließkommaberechnung der MatLab-atan-Funktion. Die resultierende Differenz zwischen den Lenkwinkelsollwertpaaren beträgt -0,5° bis 0,65° und ist für die Fahrspurführung akzeptabel (vgl. Abbildung 64).

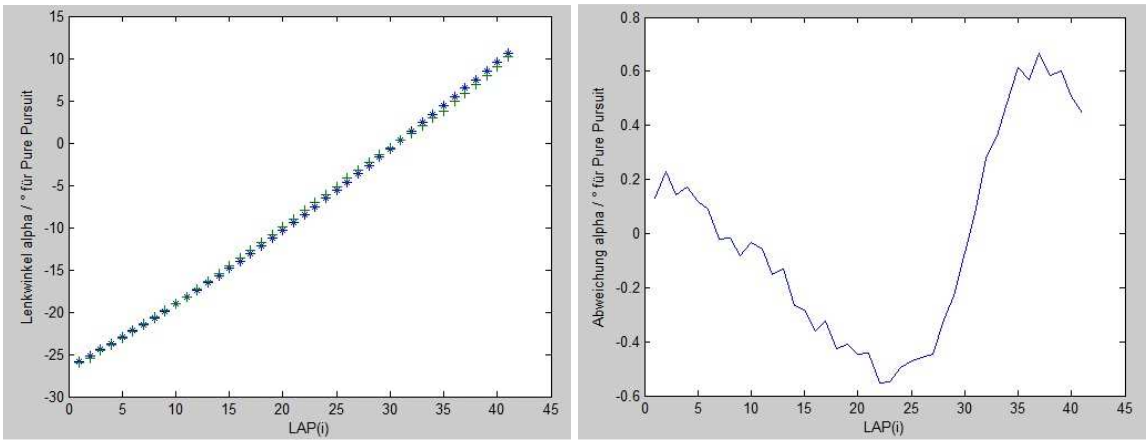

Abbildung 64: Gegenüberstellung RTL-Modells (+) und der atan-Funktion. Geradenwinkel Φ [-20°, +20°], Istabstand r = 10 cm, ROI-Position (-10 cm, 55 cm), Sollabstand d = 0 cm, L = 25,7 cm, LAD = 50 cm. -0,6° < Fehler < 0,7°.

Zur Analyse des zeitlichen Verhaltens der kaskadierten ASMs des Fahrspurführungssystems wurde ein Teilsystem betrachtet, das nur aus den Submodulen „Update“, „LAP“ und „ARCTAN“ besteht (vgl. Abbildung 50 und Abbildung 65) und die Schaltlogik zur Auswahl des Spurführungsalgorithmus vernachlässigt. Das Modul „Update“ wandelt dabei den Geradenistwinkel der HT über eine Look-Up-Tabelle in die entsprechenden Sinus- und Kosinuswerte um. Ab dem Zeitpunkt, ab dem ein gültiges HT-Ergebnis vorliegt, vergehen 44 Takte, bis der Lenkwinkelsollwert zur Verfügung steht (vgl. Abbildung 66).

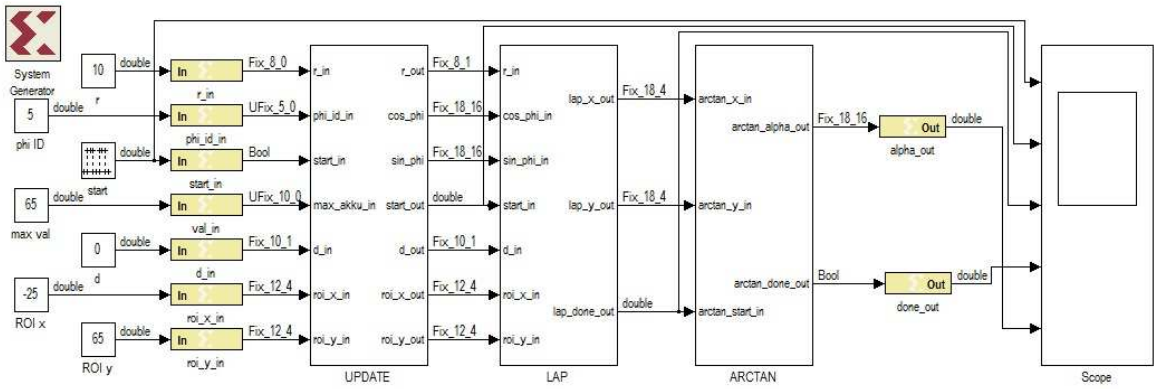

Abbildung 65: Simulation der kaskadierten ASMs. Das Update-Module wandelt den Geradenwinkel Φ über dessen ID (phi_ID) in der Hough-Ebene in korespondierende Sinus- und Kosinuswerte um, die von den ASMs „LAP“ und „ARCTAN“ sukzesive in einen Lenkwinkelsollwert umgewandelt werden.

Durch Auslösen des Startsignals speichert das Update-Modul die ROI-Position (t = 10), den Soll- und Istabstand d und r der Geraden und nutzt phi_ID als Adresse, um die entsprechenden Sinus- und Kosinuswerte zur Verfügung zu stellen, die nach 2 Takten am Ausgang anliegen (t = 12). Das durchgereichte Startsignal startet die LAP-ASM, die nach 25 Takten (t = 37) die Koordinaten des LAPs an den Eingang des ARCTAN-Moduls anlegt. Die ASM zur Approximation der atan-Funktion benötigt weiter 17 Takte bis die Lenkwinkelstellgröße berechnet und das Ergebnis am Ausgang angezeigt wird (t = 54).

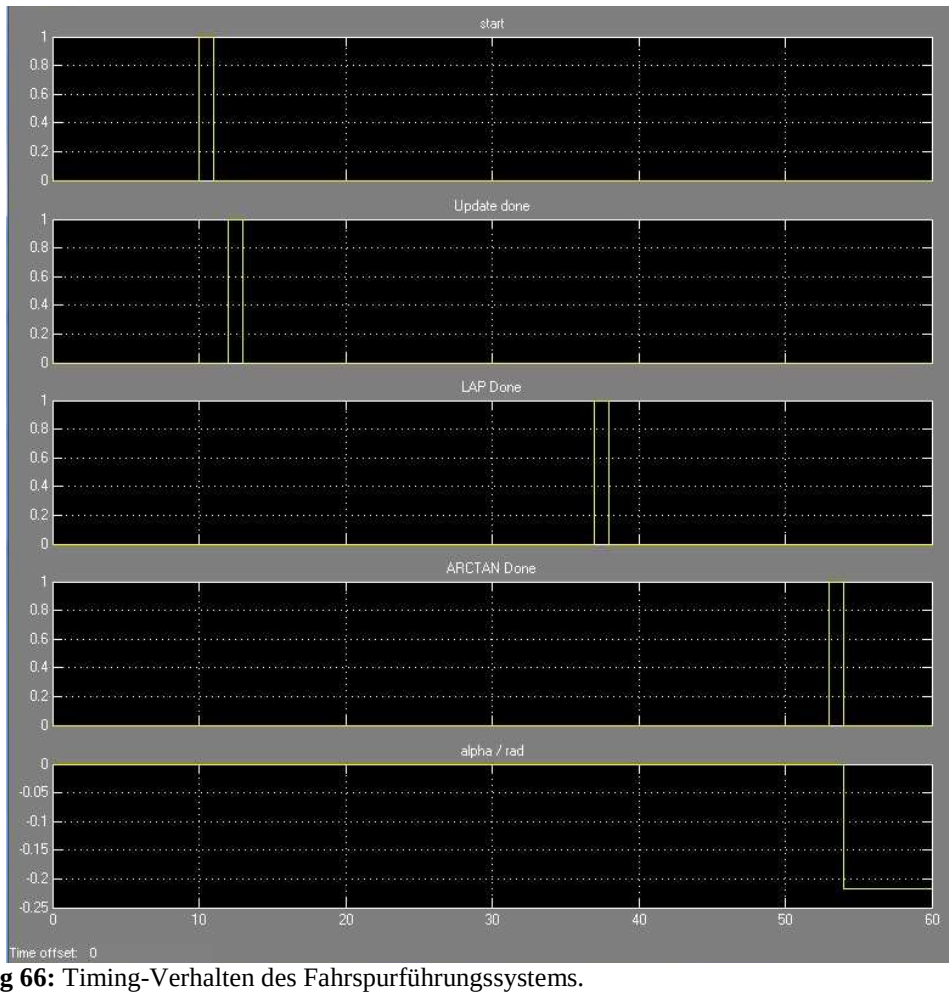

Abbildung 66: Timing-Verhalten des Fahrspurführungssystems.

8.3 Ressourcenbedarf des SoC-Systems

Der FPGA-Ressourcenbedarf dieser SoC-Konfiguration mit modifizierter Bildverarbeitungskette und Implementierung des Fahrspurführungssystem sind in Tabelle 8 und Tabelle 8 angegeben. Aufgrund der hohen Slice-Belegung von 98% wird in parallelen Abschlussarbeiten eine Ankopplung des zusätzlichen SoC-Boards TE-320 mit Spartan-3A-DSP FPGA angestrebt, das ein Vielfaches der z.Z. verfügbaren Logik- und zusätzliche DSP-Ressourcen zur Verfügung stellt [29] (vgl. [6] und [31]). Dadurch können z.B. die Bildverarbeitungsmodule auf dem DSP-Board effizienter gestaltet und das Nexys2-Board ausschließlich für die Spurführung und den künftigen Einparkassistenten genutzt werden.

Modul	**Slices**	**BRAM**	**MULT**
Camera Interface	94	0	0
Image Processing	754	0	5
Lane Detection	2550	0	4
Pathfollowing	1477	0	0
PWM	122	0	0
Result Illustration	406	4	2
VGA Interface	682	0	0

Tabelle 7: Slice-, Block-RAM- und Multiplizierer-Belegung der Beschleunigermodule (vgl. Abbildung 4).

Device Utilization Summary (actual values)				[-]
Logic Utilization	**Used**	**Available**	**Utilization**	**Note(s)**
Number of Slice Flip Flops	8,686	17,344	50%	
Number of 4 input LUTs	14,745	17,344	85%	
Number of occupied Slices	8,512	8,672	98%	
Number of Slices containing only related logic	8,512	8,512	100%	
Number of Slices containing unrelated logic	0	8,512	0%	
Total Number of 4 input LUTs	15,765	17,344	90%	
Number used as logic	11,562			
Number used as a route-thru	1,020			
Number used for Dual Port RAMs	256			
Number used for 32x1 RAMs	760			
Number used as Shift registers	2,167			
Number of bonded IOBs	119	250	47%	
IOB Flip Flops	48			
Number of RAMB16s	15	28	53%	
Number of BUFGMUXs	5	24	20%	
Number of DCMs	2	8	25%	
Number of BSCANs	1	1	100%	
Number of MULT18X18SIOs	14	28	50%	
Average Fanout of Non-Clock Nets	2.92			

Tabelle 8: Der FPGA-Ressourcenbedarf der SoC-Konfiguration mit modifizierter Bildverarbeitungskette und Fahrspurführungssystem inklusive des MicroBlaze und des in **[6]** entwickelten Bildzwischenspeichers. Die Differenz zwischen den 8512 Slices und den in **Tabelle 7** genutzten Resourcen werden vom MicroBlaze-Mikrocontroller genutzt.

8.4 Vorschläge für weitere Technologieerprobungen

Durch die Modifikation der vorentwickelten Bildverarbeitungspipeline mit der Adaptiven Binarisierung basierend auf eine Grauwerteskalierung und der morphologischen Erosion und Ergänzung dieses SoC-Systems um die Fahrspurführungskomponente wurde das RC-Fahrzeug auf der miniaturisierten Fahrbahn in Betrieb genommen (vgl. Kapitel 5 und 7). Aufgrund der laufenden Systemerweiterung mit der Kamera und dem TE0320-Board ergeben sich neue Perspektiven und zu betrachtende Aspekte.

Logging-Funktionalität mit Datenübertragung via Bluetooth: Für eine datenbasierte Analyse des Fahrverhaltens und zum Vergleich mit den in Kapitel 4.3 erarbeiteten Simulationsergebnissen wird eine Komponente zur Erfassung und kabellosen Übertragung der Geradenistparameter, der daraus resultierenden Ansteuerpunkte (LAP), sowie der berechneten Lenkwinkelstellwerte benötigt. Die Auswertung dieser Daten ermöglicht ein Feintuning des Einspur-Fahrzeugmodells und der PD-Regelung, sowie die Optimierung der Look-Ahead-Distance und des Abstands der ROI zur Fahrzeughinterachse (vgl. Kapitel 4). Zum jetzigen Zeitpunkt kann nur eine optische Bewertung des Fahrverhaltens durchgeführt werden.

HW/SW Partitionierung des Fahrspurführungsmoduls: Das derzeitige Gesamtsystem nutzt den MicroBlaze nur zum Export von Bildern auf einen Desktop-PC, welche eine aktuelle Sicht der Kamera anzeigen. Die in dieser Arbeit entwickelten Fahrspurführungsalgorithmen wurden lediglich als Beschleunigermodule entworfen, sodass die Umsetzung und Analyse einer Software-Implementierung der in Kapitel 3 vorgestellten Verfahren Follow-The-Carrot, Pure-Pursuit und PD-Regelung auf dem MicroBlaze aussteht, z.B. unter weiterer Nutzung der Wurzel- und Arkustangens-ASM als Beschleunigermodule.

Entscheidungslogik für die Fahrspurführungsalgorithmen: Die Fahrspurführung wurde mit den Algorithmen Follow-The-Carrot, Pure-Pursuit und PD-Regelung simuliert und erprobt, wobei sich das Fahrverhalten je nach Geschwindigkeit unterschiedlich auswirkt (vgl. Kapitel 4.3). Eine Entscheidungslogik würde je nach Geschwindigkeit und Fahrsituation den besten Fahrspurführungsalgorithmus bestimmen und sich der Situation anpassen.

Einführung eines Cordic-Algorithmus für die Hough-Transformation: Ein Cordic-Algorithmus nutzt für die Berechnung nur Addier- und Shift-Blöcke, die für die Geradenapproximation in der Gesamtbetrachtung weniger Multiplizierer und Slices verbraucht, als die derzeitig verwendete Hough-Transformation.

Einparkassistent: Das Einparkassistenzsystem verwendet Infrarotsensoren, um Lücken zwischen geparkten Fahrzeugen zu vermessen und mittels Einspur-Fahrzeugmodell über den „Virtuelle Deichsel“-Algorithmus einen s-förmigen Pfad zu berechnen, auf dem sich das Fahrzeug bewegen muss, um in die Parklücke zu gelangen.

Hinderniserkennung und Kollisionsvermeidung: Zur Hinderniserkennung wird zusätzlich zu den Infrarotsensoren ein Laserscanner genutzt, um die Umgebung des Fahrzeugs abzutasten über ein Ausweichsystem Vorkehrungen zur Vermeidung von Kollisionen vorgenommen.

9 Zusammenfassung

Diese Arbeit dokumentiert die Entwicklungsschritte und -ergebnisse einer SoC-basierten Echtzeit-Bildverarbeitungsplattform, die die Fahrspurerkennung und Spurführung eines Modellfahrzeugs realisiert. Zur Durchdringung der SoC-Technologie ist dazu mit einem FPGA ein Prozessorsystem konfiguriert worden, das den Videostrom mit Beschleuniger-Peripheriemodulen verarbeitet, die je nach Datenrate mit Pipelinestufen oder mit Multizyklus-Datenpfaden aufgebaut sind.

Basierend auf einer Literaturrecherche wurden für die Fahrspurführung die Algorithmen Follow-The-Carrot, Pure-Pursuit und eine PD-Abstandsregelung ausgewählt. Eine Methodik zur Erfassung des Istabstandes für die Abstandsregelung und die Berechnung des Ansteuerpunkts (LAP) für Follow-The-Carrot und Pure-Pursuit ist entwickelt worden. Die Konstruktion dieser Istwerte wurde auf die Fahrspurapproximation mit Hough-Transformation abgestimmt. Da sich die resultierende Berechnung der Lenkwinkelstellgröße aus Wurzel- und Arkustangensfunktionen zusammensetzt, sind für die Beschleuniger-Modellierung mit dem SystemGenerator Approximationen in Integer-Arithmetik entwickelt worden.

Ein entwickelter MatLab-Simulator nutzt ein Einspur-Fahrzeugmodell, um die Wirkung der Spurführungsalgorithmen und das resultierende Fahrverhalten visuell und datenbezogen zu analysieren. Aus Studien zur Spurführung mit variierten LADs und ROI-Abständen wurden die Parameter und Genauigkeitsanforderungen für die Modifikation der Vorentwicklungsstufe des SoCs und die FPGA-Implementierung gewonnen. Der MatLab-Simulator wurde zudem für die Validierung der sequentiellen Berechnung der Funktionsapproximationen genutzt. Der MatLab-Code der Spurführungsalgorithmen und Approximationsfunktionen diente als Bildungsvorschrift für die modellgetriebene RTL-Entwicklung.

Die Bildvorverarbeitungskette wurde umgestellt und einzelne Module ersetzt, um präzise Ergebnisse der Fahrspurerkennung zu erzielen. Das Modul „Adaptive Binarize“ spreizt den Grauwertebereich des Kameradatenstroms und binarisiert diesen mit einem prozentualen Schwellwert, der auf den Charakteristika des für die Fahrbahn typischen Grauwert-Histogramms basiert. Durch die Grauwertskalierung passt sich die Binarisierungsschwelle an die Lichtverhältnisse an, sodass z.B. Störungen durch Lichtreflexionen der Fahrbahn vermieden werden. Ein Erosionsfilter kontrahiert der Fahrspurmarkierungen und führt zu weniger Maxima in der Hough-Ebene und somit eindeutigen Geradenparametern r und Φ.

Das ROI-Modul des Fahrspurerkennungssystems wurde modifiziert, sodass die Hough-Transformation approximierte Fahrspurmarkierungen durch Verschieben der ROI-Position verfolgt. Diese Dynamisierung nutzt den Geradenistabstand r und führt zu einer Reduktion der ROI-Dimensionen und der resultierenden Hough-Ebene. Trotz kleinerer Winkeldiskretisierung $\Delta\Phi = 2°$ und genauerer Fahrspurapproximation wurde der Speicherbedarf zur Abbildung der Hough-Ebene um 40% reduziert. Durch Anwendung der HT in der Fahrzeugebene (Vogelperspektive) werden Spurkurven durch Tangenten approximiert, sodass die Spurführung auch auf stetigen Rundkursen ohne Einschränkungen funktioniert.

Im Fahrspurführungssystem wurde die Berechnung des LAPs und die Wurzel- und Arkustangensapproximation als Kombination mehrerer ASMs entworfen, um die mit der Bildfrequenz einfließenden Ergebnisse der HT mit Ressourcen-Sharing zu auszuwerten. Die Entwicklung der Daten- und Steuerpfade einer ASM wurde beispielhaft mit der Arkustangensapproximation vorgestellt, indem aus Pseudo-Code das ASM-Chart, die Signal-Lifetime-Tabelle, das Register-Sharing und der Daten- und Steuerpfad entwickelt wurde. Eine ASM-Kombination bildet die Spurführungsalgorithmen Follow-The-Carrot und Pure-Pursuit auf RTL ab, deren Ausgangsgröße, der Lenkwinkelsollwert, nach 44 Takten vorliegt. Die Ergänzung um einen diskreten PD-Regler und Auswahl über User-Switches liefert eine Kombination aus Pure-Pursuit und PD-Regelung (Combined Pure-Pursuit) zur Spurführung.

Die Validierung des Fahrspurführungssystems erfolgte mit einer MatLab/Simulink-Simulation, deren Eingangs-Stimuli Initialisierungs- und Auswertungsskripte erzeugen. Die Funktion der ASMs wurde durch Gegenüberstellung des RTL-Modells mit dem Fahrzeug-Simulator überprüft und das Timing-Verhalten analysiert. Die euklidische Distanz der LAP-Ergebnisse ist kleiner als 3,5 mm und resultiert in Lenkwinkel des RTL-Modells, die zwischen -0,5° und 0,65° vom Fahrzeug-Simulator abweichen.

Literaturverzeichnis

[1] Bundesministerium für Bildung und Forschung, „Allgemeine Hinweise zur BMBF-Förderung von Embedded Systems," 2010.

[2] R. Sass und A. G. Schmidt, „Embedded Systems Designwith Platform FPGAs," Burlington,MA 01803, USA, 2010.

[3] C. Plessl, „Hardware/Software Codesign," Paderborn Center for Parallel Computing, Universität Paderborn, 2010.

[4] S. A. Horsinka, „Integration einer Hough-Transformation zur Fahrspurerkennung in eine SoC-basierte Videoverarbeitungspipeline," Hochschule für Angewandte Wissenschaften Hamburg, 2010.

[5] D. Mellert, „Modellierung einer Video-basierten Fahrspurerkennung mit dem Xilinx SystemGenerator für eine SoC-Plattform," Hochschule für Angewandte Wissenschaften Hamburg, 2010.

[6] Ö. N. Püskül, „SoC-basierte Bildverarbeitungspipeline für eine Fahrspurerkennung und- visualisierung," Hochschule für Angewandte Wissenschaften Hamburg, 2011.

[7] Xilinx Inc, „Spartan-3E," [Online]. Available: http://www.xilinx.com/support/documentation/spartan-3e.htm. [Zugriff am 1 Juni 2011].

[8] Sony, „FCB-PV10 Color Camera Modul," 2006.

[9] Xilinx Inc., „MicroBlaze Processor Reference Guide - Embedded Development Kit 12.4," 2010. [Online]. Available: http://www.xilinx.com/support/documentation/sw_manuals/xilinx12_4/mb_ref_guide.pdf. [Zugriff am 01 August 2011].

[10] Xilinx Inc, „System Generator for DSP," [Online]. Available: http://www.xilinx.com/tools/sysgen.htm. [Zugriff am 1 Juni 2011].

[11] Xilinx Inc, „System Generator for DSP - Reference Guide," März 2008. [Online]. Available: http://www.xilinx.com/support/sw_manuals/sysgen_ref.pdf. [Zugriff am 1 Juni 2011].

[12] J. Morales, J. Martinez, M. Martinez und A. Mandow, „Pure-Pursuit Reactive Path Tracking for Nonholonomic Mobile Robots with a 2D Laser Scanner," Universidad de Malaga, Spain, 2009.

[13] J. S. Wit, „Vector Pursuit Path Tracking For Autonomous Ground Vehicles," University Of Florida, 2000.

[14] R. C. Coulter, „Implementation of the Pure Pursuit Pathtracking Algorithm," Carnegie Mellon University, Pittsburgh, Pennsylvania, 1992.

[15] M. Lundgren, „Path Tracking for a Miniature Robot," Umea University, Sweden, 2003.

[16] O. Ringdahl, „Path Tracking And Obstacle Avoidance Algorithms For Autonomous Forest Machines," Umea University, Sweden, 2003.

[17] I. Nikolov, „Verfahren zur Fahrbahnverfolgung eines autonomen Fahrzeugs mittels „Pure Pursuit" und „Follow-the-carrot"," Hochschule für Angewandte Wissenschaften Hamburg, 2009.

[18] The MathWorks Inc, „MatLab 2010a PID Controller," 2010.

[19] Y. Li und W. Chu, „A New Non-Restoring Square Root Algorithm and Its VLSI Implementations," Aizu-Wakamatsu 965-80 Japan, 1996.

[20] K. Johnson, „Efficient Square Root Implementation on the 68000," 1987.

[21] A. V. J. Bannur, The VLSI Implementation of A Square Root Algorithm, Washington D.C.: IEEE Computer Society Press, 1985.

[22] J. O'Leary, M. Leeser, J. Hickey und M. Aagaard, „Non-Restoring Integer Square Root: A Case Study in Design by Principled Optimization," 2nd International Conference on Theorem Provers in Circuit Design, 1994.

[23] J. Shima, „DSP Trick: Fixed-Point Atan2 With Self Normalization," 23 April 1999. [Online]. Available: http://www.dspguru.com/dsp/tricks/fixed-point-atan2-with-self-normalization. [Zugriff am 1 Juni 2011].

[24] D. Schetler, „Automatischer Ausweichassistent mit einer Laserscanner-basierten Abstandsregelung für ein fahrerloses Transportsystem," Hochschule für Angewandte Wissenschaften Hamburg, 2007.

[25] H. Kopp, Bildverarbeitung Interaktiv, Stuttgart: B.G. Teubner, 1997.

[26] A. Meisel, „Robot Vision," Hochschule für Angewandte Wissenschaften Hamburg, 2011.

[27] A. Erhardt, Einführung in die Digitale Bildverarbeitung - Grundlagen, Systeme und Anwendungen, Wiesbaden: Vieweg+Teubner | GWV Fachverlage GmbH, 2008.

[28] J. Reichardt und B. Schwarz, VHDL-Synthese: Entwurf digitaler Schaltungen und Systeme, München: Oldenbourg Wissenschaftsverlag GmbH, 2009.

[29] Trenz Electronic GmbH, „TE0320 (Spartan-3A DSP) Industrial Micromodule Series," 2010. [Online]. Available: http://www.trenz-electronic.de/de/produkte/fpga-boards/trenz-electronic/te0320-spartan-3a-dsp-series.html. [Zugriff am 1 Juni 2011].

[30] Xilinx Inc, „Spartan-3A DSP," [Online]. Available: http://www.xilinx.com/support/documentation/spartan-3a_dsp.htm. [Zugriff am 1 Juni 2011].

[31] DIGILENT Inc, „Nexys2 FPGA Board," Digilent, 2008. [Online]. Available: http://www.digilentinc.com/Products/Detail.cfm?NavPath=2,400,789&Prod=NEXYS2. [Zugriff am 1 Juni 2011].

Tabellenverzeichnis

Abbildungsverzeichnis

A Simulations-Ergebnisse der Parameterstudie

zu weit aussen | zu weit innen | Markierung übertreten | schlingern

				PD-Controller		Pure-Pursuit		Follow-The-Carrot	
V in m/s	ROI_y in cm	LAD in cm	d_{soll} in cm	max error in cm	mean error in cm	max error in cm	mean error in cm	max error	mean error
1	50	50	20	2,59	0,961	2,11	0,601	2,17	0,625
1	60	50	20	5,59	3,95	2,23	0,791	2,33	0,814
1	70	50	20	10,7	7,47	2,7	1,82	2,72	1,85
1	80	50	20	17,4	12	6,6	4,41	6,51	4,44
1	90	50	20	20	14,1	10,3	7,09	10,4	7,1
1	50	55	20			2,62	0,79	2	0,756
1	60	55	20			2,6	0,775	1,97	0,803
1	70	55	20			2,85	1,32	1,93	0,548
1	80	55	20			4,9	3,12	3,34	2,23
1	90	55	20			8,04	5,41	6,94	4,7
1	50	60	20			3,26	1,12	2,59	1,4
1	60	60	20			3,13	0,907	2,93	1,78
1	70	60	20			3,21	1,14	2,58	1,45
1	80	60	20			3,44	2,24	1,74	0,515
1	90	60	20			6,13	4,05	3,56	2,31
1	50	65	20			4,05	1,55	3,17	2,03
1	60	65	20			3,76	1,14	3,99	2,77
1	70	65	20			3,76	1,16	4,02	2,82
1	80	65	20			3,93	1,78	3,16	2,01
1	90	65	20			4,52	2,9	1,7	0,492
1	50	70	20			4,97	2,12	3,8	2,66
1	60	70	20			4,48	1,47	5,03	3,73
1	70	70	20			4,4	1,3	5,65	4,15
1	80	70	20			4,5	1,58	5,18	3,78
1	90	70	20			4,75	2,15	3,81	2,59
1	50	75	20			5,98	2,76	4,47	3,29
1	60	75	20			5,35	1,86	6,3	4,67
1	70	75	20			5,12	1,63	7,43	5,47
1	80	75	20			5,15	1,63	7,53	5,54
1	90	75	20			5,33	1,86	6,57	4,71
1	50	80	20			7,05	3,58	5,21	3,87
1	60	80	20			6,31	2,4	7,57	5,58
1	70	80	20			5,95	1,88	9,27	6,77
1	80	80	20			5,89	1,82	9,99	7,27
1	90	80	20			6	1,87	9,55	6,9
1	50	90	20			9,94	5,63	6,93	5,09

1	60	90	20			8,65	3,86	10,1	7,36
1	70	90	20			7,99	2,76	12,7	9,22
1	80	90	20			7,64	2,39	14,7	10,6
1	90	90	20			7,59	2,33	15,5	11,1
2	50	50	20	2,83	0,994	1,59	0,483	1,66	0,513
2	60	50	20	5,9	4,03	2,05	0,733	2,07	0,757
2	70	50	20	10,8	7,46	2,96	1,89	2,96	1,89
2	80	50	20	17,4	12,6	6,78	4,43	6,88	4,47
2	90	50	20	19,9	14	10,4	7,13	10,4	7,12
2	50	55	20			2,07	0,702	1,53	0,589
2	60	55	20			2,04	0,659	1,49	0,633
2	70	55	20			2,38	1,31	1,3	0,463
2	80	55	20			5,04	3,16	3,58	2,25
2	90	55	20			8,07	5,41	6,96	4,68
2	50	60	20			2,65	1,04	1,9	1,24
2	60	60	20			2,53	0,757	2,24	1,62
2	70	60	20			2,67	1,04	1,9	1,27
2	80	60	20			3,57	2,25	0,961	0,403
2	90	60	20			6,17	4,09	3,61	2,43
2	50	65	20			3,45	1,48	2,53	1,88
2	60	65	20			3,1	0,958	3,55	2,6
2	70	65	20			3,11	1,01	3,62	2,67
2	80	65	20			3,25	1,7	2,61	1,84
2	90	65	20			4,56	2,96	1,45	0,328
2	50	70	20			4,22	2,09	3,43	2,49
2	60	70	20			3,77	1,3	4,9	3,57
2	70	70	20			3,68	1,11	5,54	4,04
2	80	70	20			3,77	1,44	5,02	3,65
2	90	70	20			3,99	2,15	3,75	2,42
2	50	75	20			5,41	2,79	4,34	3,16
2	60	75	20			4,6	1,75	6,25	4,57
2	70	75	20			4,37	1,33	7,42	5,38
2	80	75	20			4,41	1,42	7,51	5,47
2	90	75	20			4,52	1,77	6,45	4,6
2	50	80	20			6,43	3,59	5,27	3,8
2	60	80	20			5,6	2,35	7,57	5,53
2	70	80	20			5,16	1,65	9,23	6,67
2	80	80	20			5,05	1,55	9,97	7,18
2	90	80	20			5,15	1,67	9,5	6,8
2	50	90	20			9,4	5,7	6,93	5,04
2	60	90	20			7,83	3,82	10,1	7,3
2	70	90	20			7,02	2,54	12,8	9,2
2	80	90	20			6,72	2,08	14,7	10,5

2	90	90	20			6,66	2,02	15,4	11
4	50	50	20	3,09	1,16	2,81	0,72	2,73	0,729
4	60	50	20	6,34	4,16	3,18	0,964	3,06	0,989
4	70	50	20	11,1	8,08	5,43	2,09	5,31	2,11
4	80	50	20	17,6	12,7	7,43	4,59	7,39	4,57
4	90	50	20	20	13,8	10,4	7,08	10,5	7,21
4	50	55	20			2,83	0,858	1,96	0,622
4	60	55	20			2,74	0,814	1,93	0,641
4	70	55	20			4,04	1,52	2,44	0,649
4	80	55	20			5,97	3,33	4,75	2,48
4	90	55	20			8,1	5,48	6,98	4,78
4	50	60	20			3,21	1,22	2,15	1,03
4	60	60	20			2,53	0,835	2,54	1,43
4	70	60	20			3,18	1,19	1,97	1,09
4	80	60	20			4,67	2,56	2,23	0,666
4	90	60	20			6,24	4,12	3,78	2,46
4	50	65	20			3,7	1,65	3,06	1,79
4	60	65	20			2,65	1,05	3,71	2,47
4	70	65	20			2,53	1,05	3,78	2,51
4	80	65	20			3,78	1,19	2,76	1,69
4	90	65	20			4,6	3,09	1,56	0,615
4	50	70	20			4,43	2,24	3,76	2,36
4	60	70	20			3,03	1,35	5,04	3,45
4	70	70	20			2,62	1,08	5,56	3,9
4	80	70	20			2,96	1,45	5,07	3,55
4	90	70	20			3,41	2,36	3,76	2,16
4	50	75	20			5,31	2,94	4,64	3,1
4	60	75	20			3,66	1,82	6,27	4,45
4	70	75	20			3,06	1,25	7,39	5,23
4	80	75	20			3,2	1,34	7,5	5,32
4	90	75	20			3,69	1,85	6,49	4,46
4	50	80	20			6,31	3,8	5,3	3,67
4	60	80	20			4,26	2,36	7,54	5,37
4	70	80	20			3,66	1,56	9,21	6,53
4	80	80	20			3,7	1,36	9,92	7,02
4	90	80	20			3,83	1,6	9,39	6,63
4	50	90	20			9	5,87	7,18	4,82
4	60	90	20			6,85	3,91	10,1	7,08
4	70	90	20			5,53	2,58	12,6	8,94
4	80	90	20			4,97	1,85	14,6	10,2
4	90	90	20			4,99	1,67	15,3	10,8

Tabelle 9: Simulationsergebnisse der Parameterstudie

B RTL-Modelle, ASMs und Auswertung des Fahrspurführungssystems

B.1 MatLab-Code der Arkustangens-Steuer-FSM

```
function [ MUX1_SEL, MUX2_SEL, MUX3_SEL,  ...
           MUX4_SEL, MUX5_SEL, ...
           R1_EN, R2_EN, R3_EN, R4_EN, R5_EN, R6_EN, ...
           SUB, DIV_EN, done] = atan_fsm(start, sign)

    persistent state, state = xl_state(0,{xlUnsigned, 4, 0});
    persistent counter, counter = xl_state(0,{xlUnsigned, 4, 0});

    MUX1_SEL = 0;
    MUX2_SEL = 0;
    MUX3_SEL = 0;
    MUX4_SEL = 0;
    MUX5_SEL = 0;
    R1_EN = false;
    R2_EN = false;
    R3_EN = false;
    R4_EN = false;
    R5_EN = false;
    R6_EN = false;
    SUB = false;
    DIV_EN = false;
    done = false;

    switch state
        case 0
            if start == true
                MUX1_SEL = 0;
                R1_EN = true;  % sign = MSB(x) XOR MSB(y) -> R1
                R2_EN = true;  % x <- x_in -> R2
                R3_EN = true;  % y <- y_in -> R3
                state = 1;
             else
                state = 0;
             end
        case 1                 %t1 <- y -/+ x
            MUX2_SEL = 0;
            MUX3_SEL = 0;
            if sign == 0
                SUB = true;    %t1 <- y - x
            else
                SUB = false;   %t1 <- y + x
            end
            R4_EN = true;      %t1 -> R4
            state = 2;
        case 2                 %t2 <- y +/- x -> R3
            MUX1_SEL = 1;
            MUX2_SEL = 0;
            MUX3_SEL = 0;
            if sign == 0
                SUB = false;   %t2 <- y + x
            else
                SUB = true;    %t2 <- y - x
```

```
            end
            R3_EN = true;       %t2 -> R3
            counter = 0;
            state = 3;
        case 3                  %t3 <- t1 / t2
            DIV_EN = true;
            counter = counter + 1;
            if counter == 9     %div fertig
                state = 4;      %t3 -> ist im DIVIDER OUT REG gespeichert
            else
                state = 3;      %warten bis div fertig
            end
        case 4
            MUX4_SEL = 0;
            MUX5_SEL = 0;
            R5_EN = true;       %t4 = t3 * t3 -> R5
            state = 5;
        case 5
            MUX4_SEL = 1;
            MUX5_SEL = 2;
            R5_EN = true;       %t5 = t4 * 0.2 -> R5
            state = 6;
        case 6
            MUX4_SEL = 0;
            MUX5_SEL = 3;
            R5_EN = true;       %t6 = t3 * 0.9817 -> R5
            if sign == 0        %subtrahierer beschalten
                MUX2_SEL = 1;   %t7 = t5 - 1
                MUX3_SEL = 1;
            else
                MUX2_SEL = 2;   %t7 = 1 - t5
                MUX3_SEL = 2;
            end
            SUB = true;
            R4_EN = true;       %t7 -> R4
            state = 7;
        case 7
            MUX4_SEL = 1;
            MUX5_SEL = 1;
            R5_EN = true;       %t8 = t6 * t7 -> R5
            state = 8;
        case 8
            MUX2_SEL = 1;
            MUX3_SEL = 3;
            if sign == 0        %ADD/SUB beschalten
                SUB = false;
            else
                SUB = true;
            end
            R6_EN = true;       %alpha = t8 +/- pi/4 -> R6
            state = 9;
        case 9
            done = true;
            state = 0;
    otherwise
            state = 0;
    end
end
```

B.2 RTL-Modell der PD-Abstandsregelung

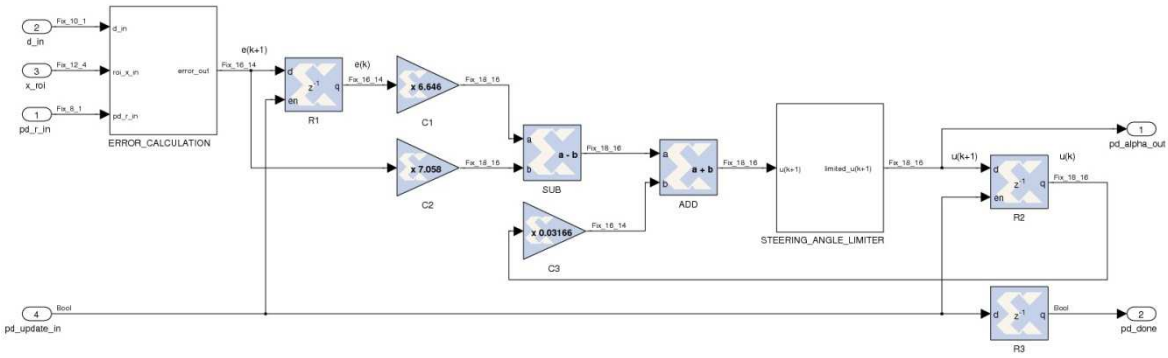

Abbildung 67: RTL-Modell des PD-Reglers: Die Regelabweichung wird aus den eingehenden Daten generiert. Zur Vermeidung einer verfälschten Stellgrößenrückführung muss bereits hier der Lenkwinkel begrenzt werden.

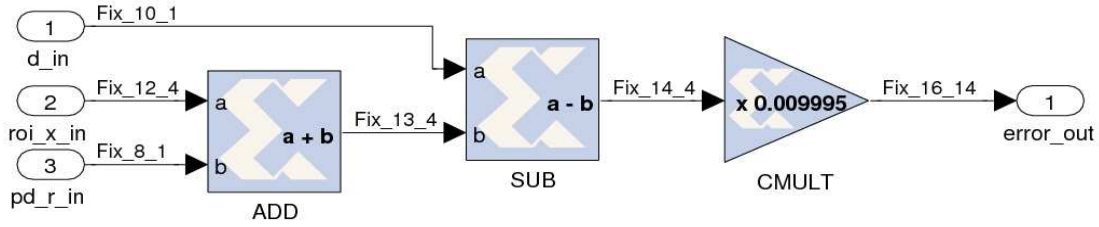

Abbildung 68: Berechnung der Regelabweichung: Die Regelabweichung setzt sich aus dem Sollabstand d, der x-Position der ROI und dem ermittelten Abstand r der Geraden zusammen.

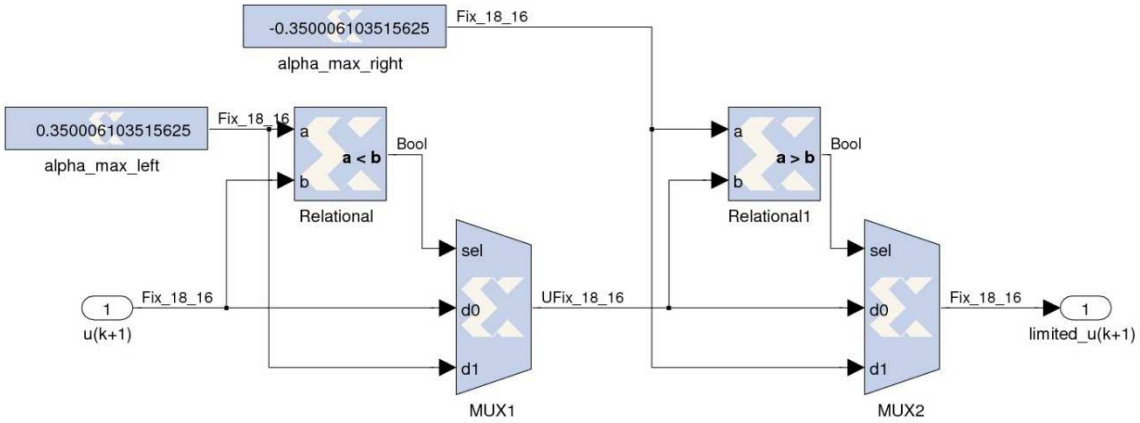

Abbildung 69: Begrenzung der Lenkwinkelstellgröße: Der Wertebereich des anzulegenden Lenkwinkels wird auf den Lenkwinkelbereich des Steuerservomotors begrenzt.

B.3 ASM zur LAP-Berechnung

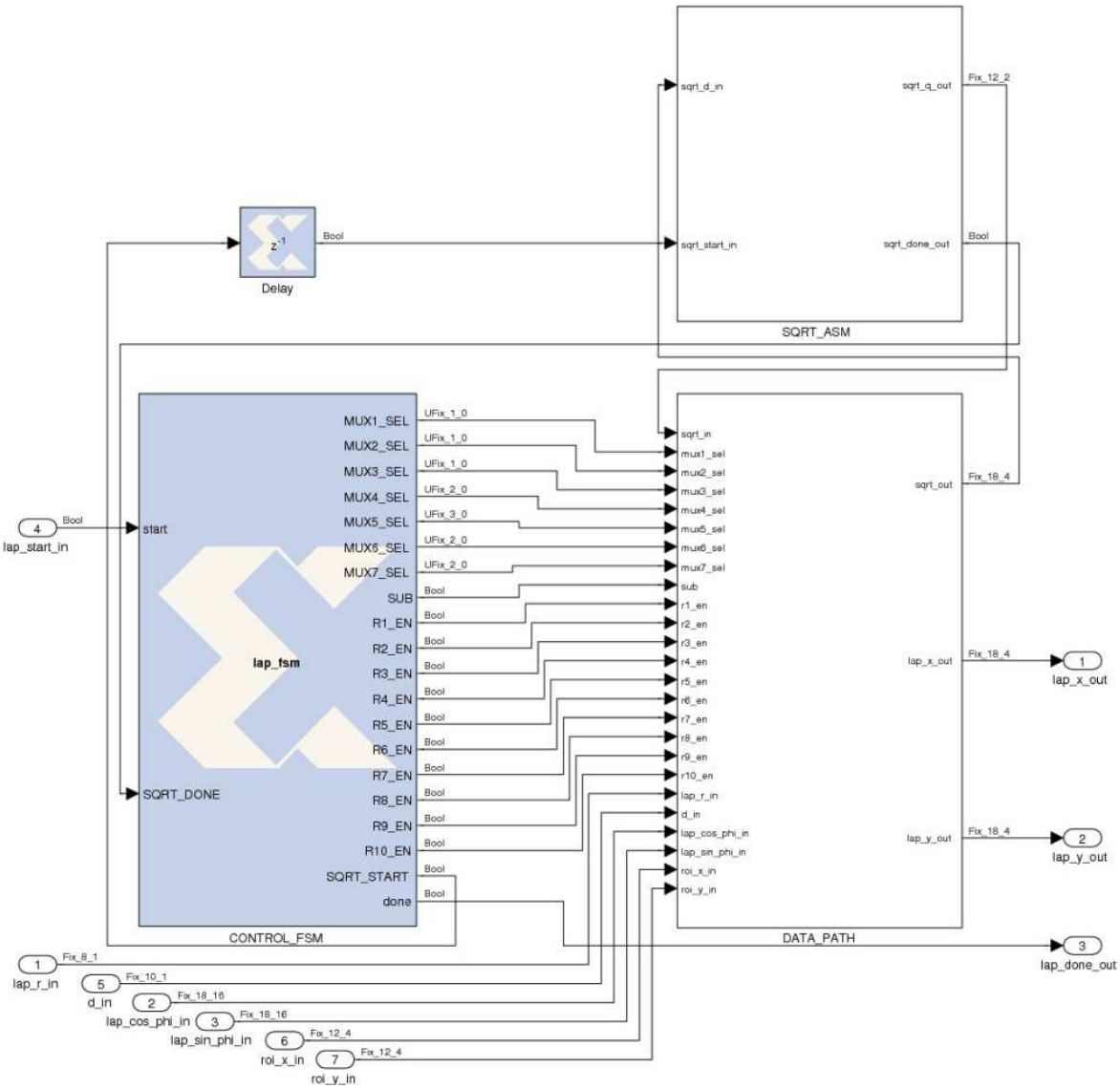

Abbildung 70: Steuer- und Datenpfad der LAP-Berechnung: Die Steuer-FSM regelt das Zusammenspiel des Datenpfades mit der Wurzel-ASM.

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

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

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

Listing 2: MatLab-Code zur Berechnung des Look-Ahead-Points

Zustand	Variable		OP1	Operator	OP2	Bedeutung
z0	t1	<-	R			Eingangssynchronisation
	t2	<-	cos(Φ)			
	t3	<-	sin(Φ)			
z1	t4	<-	t1	-	D	R-D
	t5	<-	t2	*	ROIX	cos(Φ)*ROIX
z2	t6	<-	t3	*	ROIY	sin(Φ)*ROIY
z3	t7	<-	t5	+	t6	ROIX_H = cos(Φ) * ROIX + sin(Φ) ROIY
z4	t8	<-	t4	+	t7	LAPX_H = ROIX_H + (R – D)
z5	t9	<-	t8	*	t8	LAPX_H² = LAPX_H * LAPX_H
z6	t10	<-	t2	*	t8	cos(Φ)*LAPX_H
	sqrt_out	<-	LAD²	-	t9	LAPY_H² = (LAD²-LAPX_H²)
z7	t11	<-	t3	*	t8	sin(Φ)*LAPX_H
	t12	←	sqrt_in			LAPY_H
z8	t13	<-	t3	*	t12	sin(Φ)*LAPY_H
z9	t14	<-	t2	*	t12	cos(Φ)*LAPY_H
	LAPX	<-	t10	-	t13	LAPX = cos(Φ) LAPX_H – sin(Φ) LAPY_H
z10	LAPY	<-	t14	+	t11	LAPY = sin(Φ) LAPX_H + cos(Φ) LAPY_H

Tabelle 10: Aufteilung und Zustandszuordnung der einzelnen Schritte der LAP-Berechnung.

Variable	Precision OP1	Operator	Precision OP2	Full Precision	User defined
t1	Fix_10_0	cast			Fix_18_4
t2	Fix_18_16				Fix_18_16
t3	Fix_18_16				Fix_18_16
t4	Fix_18_4	-	Fix_18_4	Fix_19_4	Fix_18_4
t5	Fix_18_16	*	Fix_18_4	Fix_36_20	Fix_18_4
t6	Fix_18_16	*	Fix_18_4	Fix_36_20	Fix_18_4
t7	Fix_18_4	+	Fix_18_4	Fix_19_4	Fix_18_4
t8	Fix_18_4	+	Fix_18_4	Fix_19_4	Fix_18_4
t9	Fix_18_4	*	Fix_18_4	Fix_36_8	UFix_18_4
t10	Fix_18_16	*	Fix_18_4	Fix_36_20	Fix_18_4
sqrt_out	Ufix_18_4	-	Ufix_18_4	Ufix_19_0	Ufix_18_0
t11	Fix_18_16	*	Fix_18_4	Fix_36_20	Fix_18_4
t12	Ufix_9_0	SRL 2 / CAST			Fix_18_4
t13	Fix_18_16	*	Fix_18_4	Fix_36_20	Fix_18_4
t14	Fix_18_16	*	Fix_18_4	Fix_36_20	Fix_18_4
x_lap	Fix_18_4	-	Fix_18_4	Fix_19_4	Fix_18_4
y_lap	Fix_18_4	+	Fix_18_4	Fix_19_4	Fix_18_4

Tabelle 11: Quantisierung der LAP-Zwischen- und -Endergebnisse.

	z0	z1	z2	z3	z4	z5	z6	z7	z8	z9	z10	Register
t1		x										R3
t2		x	x	x	x	x	x	x	x	x		R1
t3		x	x	x	x	x	x	x	x			R2
t4			x	x	x							R3
t5			x	x								R5
t6				x								R6
t7					x							R4
t8						x	x	x				R4
t9							x					R5
t10								x	x	x		R5
sqrt_out								x				R10
t11									x	x	x	R6
t12									x	x		R4
t13										x	x	R7
t14											x	R5
x_lap	x	x	x	x	x	x	x	x	x	x	x	R8
y_lap	x	x	x	x	x	x	x	x	x	x	x	R9

Tabelle 12: Signal-Lifetime-Table zum Register-Sharing der LAP-ASM

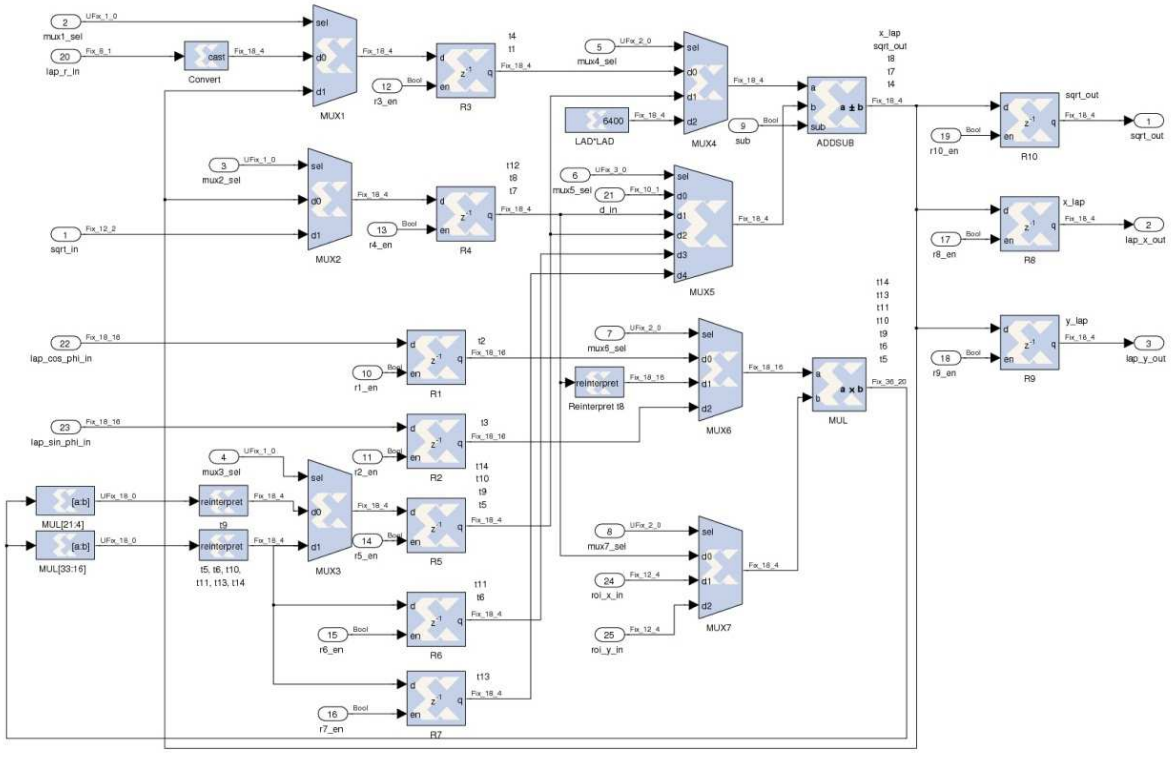

Abbildung 71: Datenpfad zur LAP-Berechnung: Durch Register Sharing und Zuweisen von Registern zu bestimmten Arithmetik-Blöcken wurden Multiplexer hinter diesen ALUs eingespart, Laufzeitpfade verkürzt und Register eingespart.

```
function [ MUX1_SEL, MUX2_SEL, MUX3_SEL, MUX4_SEL, MUX5_SEL, MUX6_SEL,
MUX7_SEL, SUB, R1_EN, R2_EN, R3_EN, R4_EN, R5_EN, R6_EN, R7_EN, R8_EN,
R9_EN, R10_EN, SQRT_START, done] = lap_fsm(start, SQRT_DONE)
persistent state, state = xl_state(0,{xlUnsigned, 4, 0});
%defaults
MUX1_SEL = 0; MUX2_SEL = 0;
MUX3_SEL = 0; MUX4_SEL = 0;
MUX5_SEL = 0; MUX6_SEL = 0;
MUX7_SEL = 0; SUB = false;
R1_EN = false; R2_EN = false;
R3_EN = false; R4_EN = false;
R6_EN = false; R5_EN = false;
R8_EN = false; R7_EN = false;
R10_EN = false; R9_EN = false;
SQRT_START = false; done = false;

switch state
    case 0
        if start == true
            MUX1_SEL = 0;
            R3_EN = true;   %t1 = r -> R3
            R1_EN = true;   %t2 = cos(phi) -> R1
            R2_EN = true;   %t3 = sin(phi) -> R2
            state = 1;
        else
            state = 0;
        end
    case 1
        done = false;
        MUX4_SEL = 0;
        MUX5_SEL = 0;
        SUB = true;
        MUX1_SEL = 1;
        R3_EN = true;       %t4 = t1 - d -> R3
        MUX6_SEL = 0;
        MUX7_SEL = 1;
        MUX3_SEL = 1;
        R5_EN = true;       %t5 = t2 * x_roi ->R5
        state = 2;
    case 2
        R3_EN = false;
        MUX6_SEL = 2;
        MUX7_SEL = 2;
        R6_EN = true;       %t6 = t3 * y_roi -> R6
        state = 3;
    case 3
        MUX4_SEL = 1;
        MUX5_SEL = 3;
        SUB = false;
        MUX2_SEL = 0;
        R4_EN = true;       %t7 = t5 + t6 -> R4
        state = 4;
    case 4
        MUX4_SEL = 0;
        MUX5_SEL = 1;
        SUB = false;
        MUX2_SEL = 0;
        R4_EN = true;       %t8 = t4 + t7 -> R4
        state = 5;
```

```
    case 5
        MUX6_SEL = 1;
        MUX7_SEL = 0;
        MUX3_SEL = 0;
        R5_EN = true;       %t9 = t8 * t8 -> R5
        state = 6;
    case 6
        MUX6_SEL = 0;
        MUX7_SEL = 0;
        MUX3_SEL = 1;
        R5_EN = true;       %t10 = t2 * t8 -> R5
        MUX4_SEL = 2;
        MUX5_SEL = 2;
        SUB = true;
        R10_EN = true       %sqrt_out = LAD^2 - t9 -> R10
        SQRT_START = true;  %sqrt-Automat starten
        state = 7;
    case 7
        MUX6_SEL = 2;
        MUX7_SEL = 0;
        R6_EN = true;       %t11 = t3 * t8 -> R6
        if SQRT_DONE == true
            MUX2_SEL = 1;
            R4_EN = true;   %t12 = sqrt_in ->R4
            state = 8;
        else
            state = 7;
        end
    case 8
        MUX6_SEL = 2;
        MUX7_SEL = 0;
        R7_EN = true;       %t13 = t3 * t12 -> R7
        state = 9;
    case 9
        MUX6_SEL = 0;
        MUX7_SEL = 0;
        MUX3_SEL = 1;
        R5_EN = true;       %t14 = t2 * t12 -> R5
        MUX4_SEL = 1;
        MUX5_SEL = 4;
        SUB = true;
        R8_EN = true;       %x_lap = t10 - t13 -> R8
        state = 10;
    case 10
        MUX4_SEL = 1;
        MUX5_SEL = 3;
        SUB = false;
        R9_EN = true;       %y_lap = t14 + t11 -> R9
        state = 11;
    case 11
        done = true;
        state = 0;
    otherwise
        state = 0;
end
end
```

Listing 3: Steuer-FSM der LAP-ASM

B.4 MatLab-Code zur Simulation und Auswertung der ASMs

```
clc;
clear all;
ts = 1;
% Konstanten erzeugen
roi_size_x = 50;
roi_size_y = 50;
L = 25.7;
roi_x = -10;
roi_y = 55;
ROI = [roi_x; roi_y];
phi = -20:1:20;
r = 30;
d = 0;
LAD = 55;
anz_takte = 26;
sim_size = length(phi)*anz_takte;
% STIMULI INIT
SIN_PHI = zeros(sim_size,2);
COS_PHI = zeros(sim_size,2);
START = zeros(sim_size,2);
t = 0;
%Stimuli erzeugen
for y = 1:sim_size
    START(y,1) = y;
    if mod(y, anz_takte) == 1
        START(y,2) = 1;
        t = t + 1;
    else
        START(y,2) = 0;
    end;
    SIN_PHI(y,1) = y;
    SIN_PHI(y,2) = sind(phi(t));
    COS_PHI(y,1) = y;
    COS_PHI(y,2) = cosd(phi(t));
end
%Modell simulieren
sim_length = sim_size;
sim('lap_analyse2');
%SW-Berechnung
lap_res_sw = zeros(length(phi),2);
for x = 1:length(phi)
    LAP = LookAheadPoint(ROI, deg2rad(phi(x)), r, d, LAD);
    lap_res_sw(x,1) = LAP(1);
    lap_res_sw(x,2) = LAP(2);
end
%RTL-Modell-Ergebnisse zusammenstellen
LAP_RES = zeros(length(phi),2);
t = 0;
for y = 1:sim_length+1
    if(LAP_DONE(y)==1)
      t = t + 1;
      LAP_RES(t,1) = LAP_X(y);
      LAP_RES(t,2) = LAP_Y(y);
    end;
end;
```

```
%Euklidische Distanz Simulator zu RTL-Modell
error_vec = zeros(length(phi),1);
for z = 1:length(phi)
    a = [lap_res_sw(z,1) lap_res_sw(z,2)] - [LAP_RES(z,1) LAP_RES(z,2)];
    error_vec(z) = norm(a);
end
%plotten
figure(1);
plot(lap_res_sw(:,1),lap_res_sw(:,2),'*', LAP_RES(:,1),LAP_RES(:,2),'+');
axis([19.5 24 49 52]);
z=1:1:41;
figure(2);
plot(z, error_vec(z));
axis([0 42 0 0.5]);
```

Listing 4: MatLab-Code zur Stimulierzeugung und Auswertung der LAP-ASM

```
% STIMULI INIT
anz_takte = 20;
sim_size = length(LAP_RES)*anz_takte;
LAP_X1 = zeros(sim_size,2);
LAP_Y1 = zeros(sim_size,2);
START = zeros(sim_size,2);
t = 0;
for y = 1:sim_size
    START(y,1) = y;
    if mod(y, anz_takte) == 1
        START(y,2) = 1; t = t + 1;
    else
        START(y,2) = 0;
    end;
    LAP_X1(y,1) = y; LAP_Y1(y,1) = y;
    LAP_X1(y,2) = LAP_RES(t,1);
    LAP_Y1(y,2) = LAP_RES(t,2);
end
%Simulation des ATAN-Modells
sim_length = sim_size;
sim('atan_analyse_PP');
%SW-Berechnung
atan_res_sw = zeros(length(LAP_RES),1);
for x = 1:length(LAP_RES)
    atan_res_sw(x) = rad2deg(atan(lap_res_sw(x,1)/(LAD^2/(2*L))));
end
%RTL-Modell-Berechnung
ATAN_RES = zeros(length(LAP_RES),1);
t = 0;
for y = 1:sim_length+1
    if(ATAN_DONE(y)==1)
      t = t + 1;
      ATAN_RES(t) = rad2deg(ATAN_ALPHA(y));
    end;
end;
%Abweichung Simulator - RTL-Modell
z=1:1:41;
figure(1); plot(z, atan_res_sw(z)-ATAN_RES(z));
figure(2); plot(z, atan_res_sw(z),'*', z, ATAN_RES(z),'+');
```

Listing 5: MatLab-Code zur Stimulierzeugung und Ergebnisanalyse der ARCTAN-ASM

C EDK-Projekt der SoC-Konfiguration

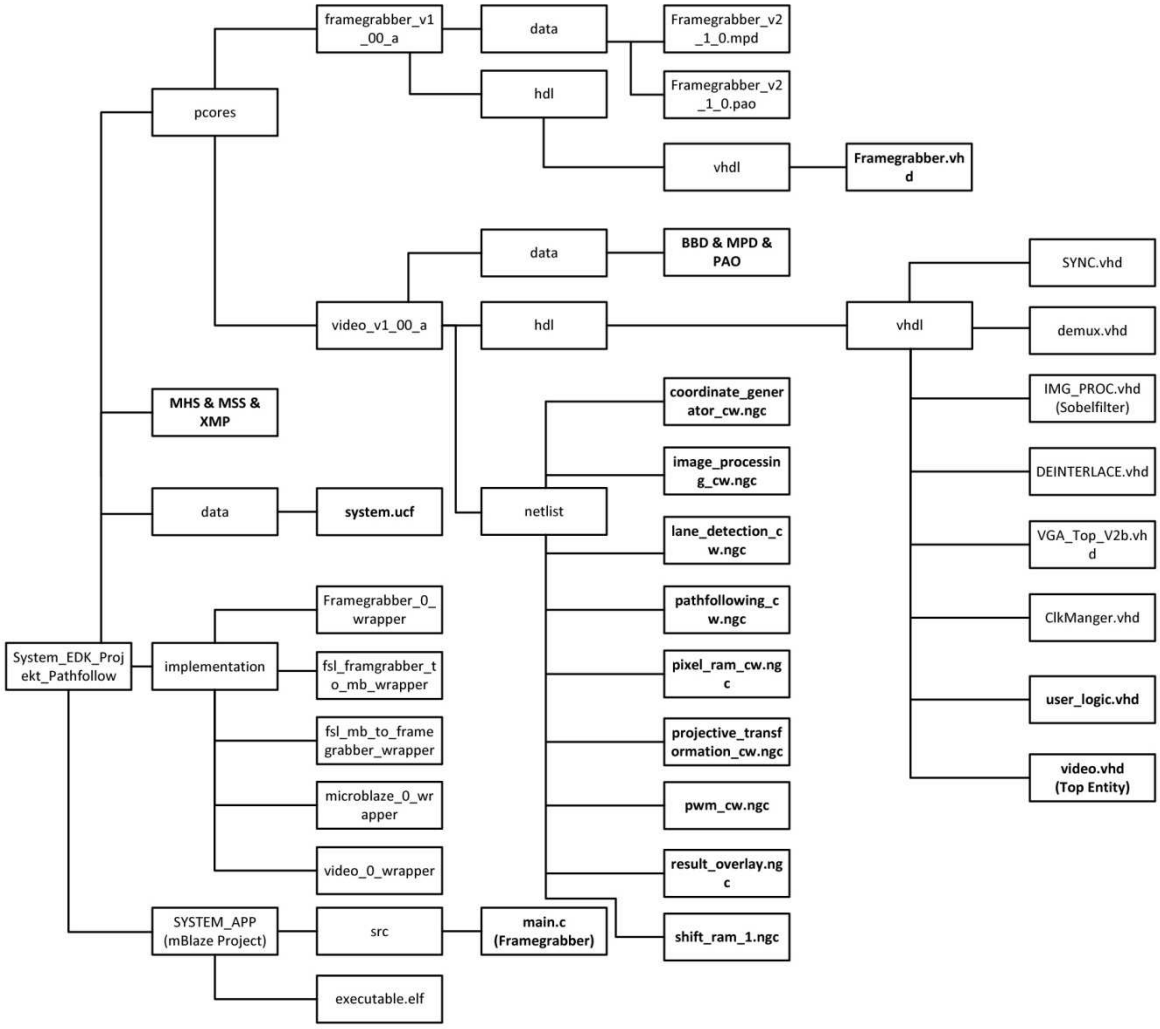

Abbildung 72: Dateibaum des XPS-Projekts. Die Dateipfade und enthaltenen Dateien des XPS-Projektes, die zur Erstellung der SoC-Konfiguration genutzt werden.

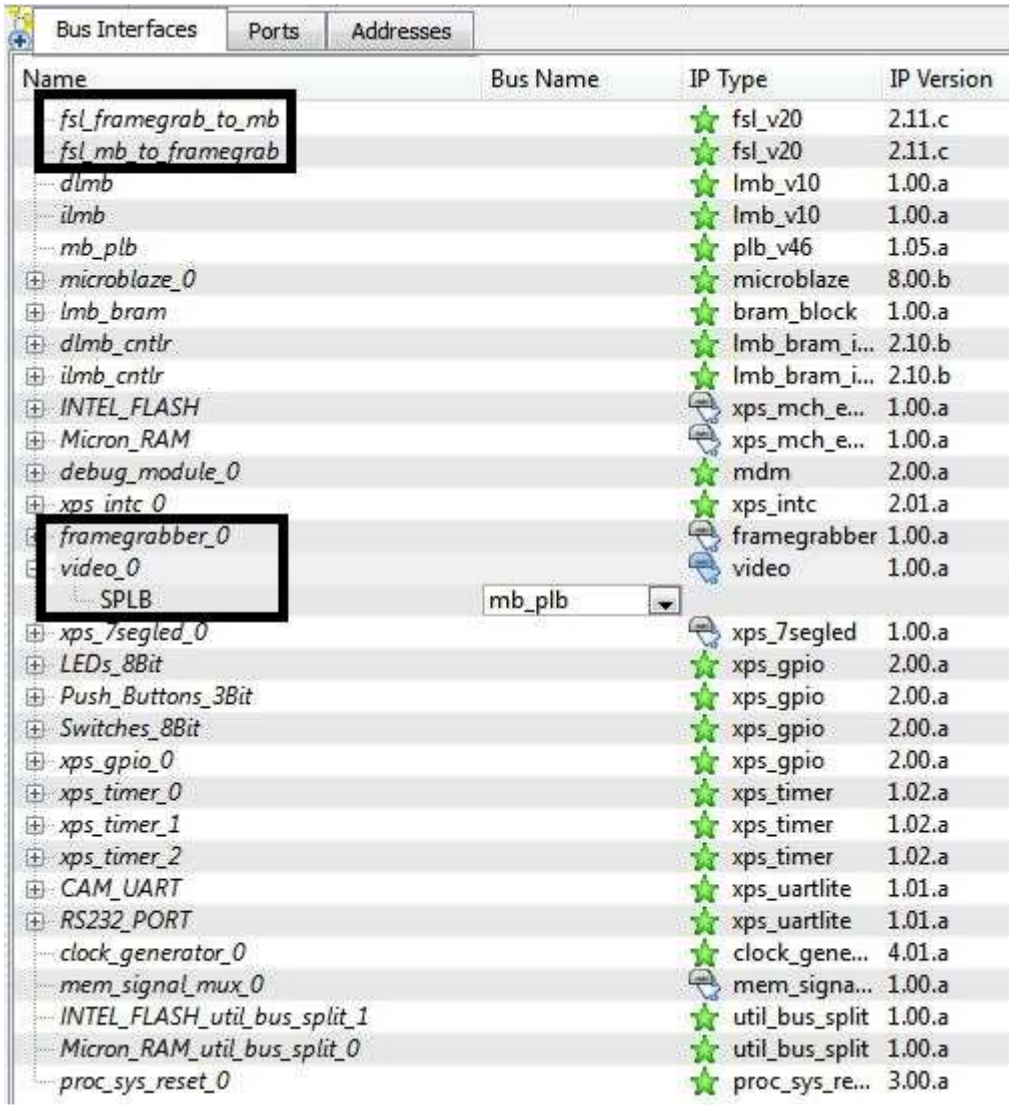

Abbildung 73: System Assembly View. Die IP-Cores des EDK-Projektes werden im System Assembly View angezeigt und deren Verknüpfungen über den mb_plb-Bus, dabei sind Framegrabber und Video Anwender-IP-Cores. Der System Assembly View ist an die MHS-Datei angebunden.

D Pin-Belegung Sony FCB

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

Tabelle 13: CN501 Pinbelegung bei 8-Bit-Datenbuskonfiguration

Pin No.	Name	Bedeutung	Level
1	UNREG	Power Input	6-12 V (DC)
2	GND	Signal Ground	
3	TD	Transmit Data	TTL Level (0-5 V)
4	RD	Receive Data	TTL Level (0-5 V)

Tabelle 14: CN701 Pinbelegung für die Spannungsversorgung und Parametrisierung der Kamera über RS232

E CD: SystemGenerator- und EDK-Projektdateien, MatLab- und VHDL-Code