Bachelorarbeit

SoC-basierte Bildverarbeitungspipeline für eine Fahrspurerkennung und -visualisierung

Özgür Nurettin Püskül

Bachelorarbeit eingereicht im Rahmen der Bachelorpüfung
im Studiengang Technischen 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. rer. nat. Thomas Canzler

Abgegeben am 01. Juli 2010

Özgür Nurettin Püsköl

Thema der Bachelorarbeit
SoC-basierte Bildverarbeitungspipeline für eine Fahrspurerkennung und- visualisierung

Stichworte
Echtzeit-Bildverarbeitung, System-on-Chip, RTL-Pipeline, videobasierte Fahrspurerkennung, Visualisierung, VHDL-Codegenerierung, Leiterplattendesign, LCD-Touch-Panel

Kurzzusammenfassung
Diese Arbeit behandelt die Entwicklung einer Fahrspurerkennung für eine autonome Fahrzeugführung, die als Bildverarbeitungspipeline auf einer SoC-Plattform realisiert ist. Zur Visualisierung der Echzeit-Videodaten einer CCD-Kamera wurde die Bildverarbeitungsplattform an einen VGA-Monitor angeschlossen. Durch die Visualisierung wurden Erkenntnisse über das Fahrverhalten des Fahrzeugs erlangt, die zum Austausch des Sobel-Filters zu einer Schwellwertanpassung mit Graubildbinarisierung mit einer zusätzlichen morphologischen Erosion geführt hat. Weiterhin wurde die projektive Entzerrung des Bildes auf ein metrisches Maß überführt und die Transformationsmatrix der Kameraposition angepasst. Zur Kopplung der SoC-Plattform mit einer CCD-Kamera, einem LCD-Touchpanel und einem FPGA-Miniboard sind Adapterplatinen mit miniaturisierten Steckverbindern entwickelt worden. Die Konzeptionierung der elektrischen Leiterplatten und deren Entwicklung wird in dieser Arbeit dargestellt.

Özgür Nurettin Püsköl

Title of the bachelor thesis
SoC-based image processing pipeline for a lane detection and visualization

Keywords
Real-time image processing, System-on-Chip, RTL-Pipeline, video-based lane detection, visualization, VHDL-Code generation, PCB design, LCD-Touch-Pane

Abstract
The thesis deals with the development of a lane detection for autonomous vehicle guidance, which is realized as an image processing pipeline on a SoC-platform. The visualization of a real-time videostream via a CCD-Camera is realized with an image processing platform connected to a VGA-monitor. The Knowledge of the visualization were used for the performance of the vehicle, which exchanges the Sobel-filter to an adaptive binarize and a morphological erosion. Furthermore, the projektive rectification of the image transferred to a metric measure and the transformation-matrix was adjusted to the camera position. To couple the SoC-platform with a CCD-Camera, an LCD-Touch-Panel and a FPGA-miniboard, an adapter board have been developed with miniaturized connectors. The conceptual design of the electrical circuit boards and their development is presented in this thesis.

Inhaltsverzeichnis

1. Einleitung **5**

2. Systemübersicht zur Bildverarbeitungsplattform **8**
2.1. CCD-Kamera FCB-PV10 8
2.2. SoC-Plattform mit einem Spartan3E1200 FPGA 10
2.3. 5.7" LCD-Touch-Panel mit integriertem Timing-Controller 12
2.3.1. Gegenüberstellung von Displayparametern 12
2.3.2. Ansteuerung des LCD-Touch-Panels 13
2.4. Trenz-Electronic Spartan 3A DSP 3400K 14
2.5. Gekoppelte SoC-Plattform 16
2.6. Systemübersicht zur Vorentwicklungsstufe 17

3. Konzept und Kohärenz **19**
3.1. Systemübersicht zur weiterentwickelten Bildverarbeitungsplattform 19
3.2. Zeilenzwischenspeicher-RAM für eine Rekonstruktion von Bildkoordinaten 21
3.3. LCD-Controller 21
3.4. Erweiterungsadapter für die Bildverarbeitungsplattform 22
3.4.1. Entwicklung der Adapterplatinen mit dem Altium Designer 23
3.4.2. Massekonzept und Signallaufzeitpfade der Adapterplatinen 26
3.5. Simulation, RTL-Codegenerierung und SoC-Konfiguration 26

4. Modifizierung und Erweiterung der Bildverarbeitungspipeline **29**
4.1. Schwellwertanpassung zur Graubildbinarisierung 29
4.2. Morphologische Erosion zur Kontraktion der Fahrbahnmarkierung 32
4.3. Projektive Entzerrung der Kameraperspektive 38
4.3.1. Kalibrierung der perspektivischen Transformation 40
4.4. Entwurf eines Zwischenspeicher-RAMs zur Wiederherstellung der Pixelreihenfolge 41
4.4.1. Bestimmung des zu speichernden Bildbereiches 41
4.4.2. Entwicklung eines dynamischen Adressgenerators 42
4.4.3. Visualisierungsergebnisse und Auswertung der Filterfunktion 44

5. Adapterplatinen der SoC-Plattformerweiterung **45**
5.1. FFC-IDC-Adapter und IDC-PMOD-Adapter für die Pixelstromübertragung zur Bildverarbeitungspipeline 45
5.2. FFC-FX2-Adapter für eine Visualisierung des Echtzeit-Videodatenstroms . 46
5.3. Kopplung des Nexys2-Boards mit einem zusätzlichem Spartan3A DSP FPGA 47

6. Verifikation der Bildverarbeitungspipeline-Modelle mit Matlab/Simulink **50**
6.1. Bildverarbeitungsmodelle zur Aufbereitung des Videodatenstroms 50
6.2. Projektive Transformation mit Zeilenzwischenspeicher für die Visualisierung des projektiv Entzerrten Bildes 52

7. Ergebnisanalyse der Fahrspurerkennung **57**

8. Weiterentwicklung **59**

9. Zusammenfassung **60**

10. Literaturverzeichnis **1**

11. Abbildungsverzeichnis **3**

12. Tabellenverzeichnis **8**

Anhang **9**

A. Tabelle zu den Pin-Connections zwischen den Adapterplatinen und der Hardwareplattform **10**

B. EDK-Projekt, VHDL-Code und Matlab-Skript der Bildverarbeitungskomponenten **15**

B.1. Matlab-Skript zur Initialisierung der gesamten Simulink-Modelle 15

B.2. Matlab-Skript zur Neukalibrierung der Transformationsmatrix B und die Auswertung der projektiven Transformation 18

B.3. Initialisierung der Zeit t zur Auswertung der generierten Adressen des Adressgenerators 19

B.4. Das System EDK Projekt der Bildverarbeitungspipeline mit integriertem Framegrabber 20

B.5. VHDL Code der User-Logic des aktuellen EDK-Projektes 23

1. Einleitung

Embedded Systeme sind vertreten in der Robotik, Fahrzeugen, Flugzeugen, Haushaltstechniken, in der Medizintechnik und sind in vielen weiteren Anwendungsgebieten des täglichen Lebens von Bedeutung. Die systematische Entwicklung, Modellierung, Validierung, Verifikation und Gestaltung der Software solcher Systeme bietet eine Fülle aktueller Forschungsfragen mit hoher Anwendungsrelevanz [DLR10]. In einem System-on-Chip werden Datenverarbeitungsfunktionen in Beschleunigermodulen und ein Softcore-Prozessor *(µBlaze)* [Xil10a] integriert, der als HW/SW-Codesign in einem FPGA-basierten System zum Einsatz kommt (vgl. Abb 1.1).

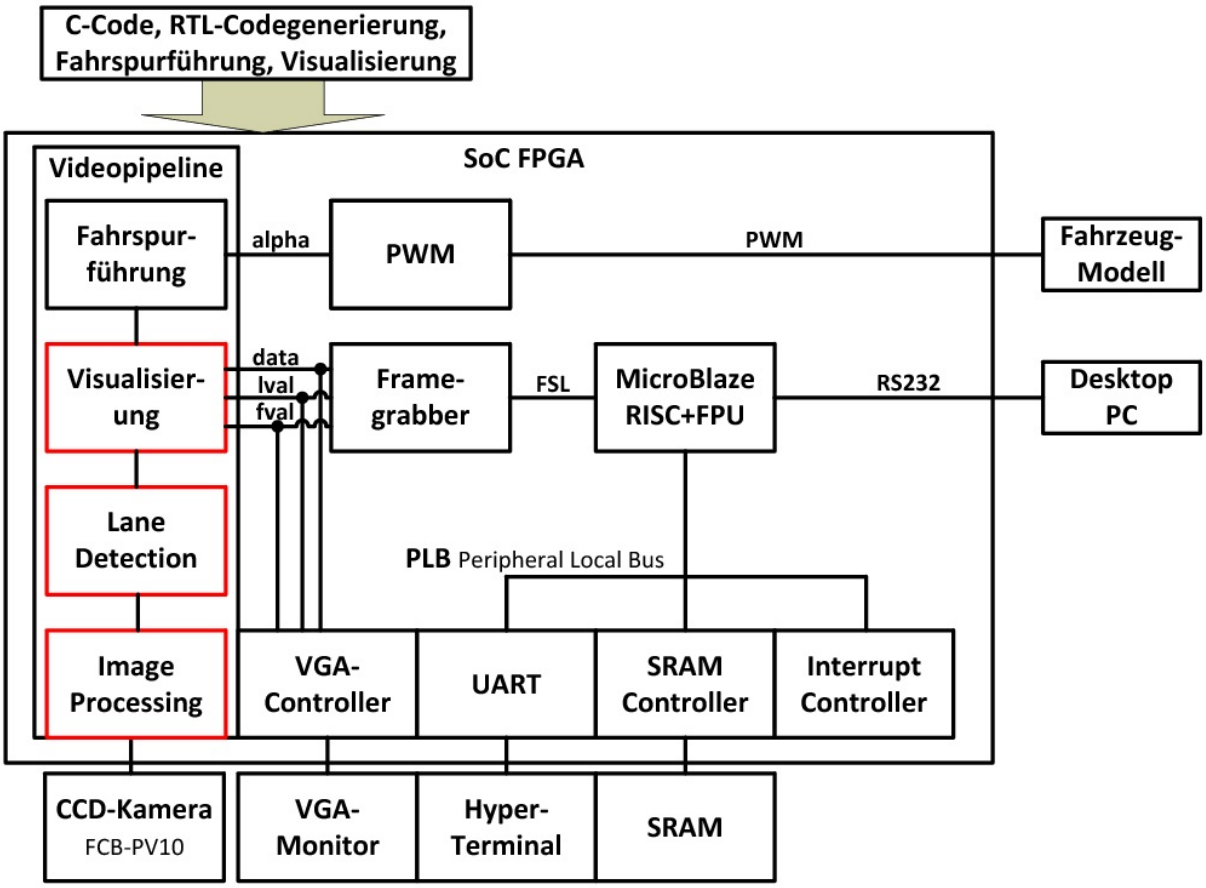

Abbildung 1.1.: SoC-Plattform für die Entwicklung eines Fahrspurführungssystems für ein autonomes Modellfahrzeug mit der Visualisierung der Bildverarbeitungsergebnisse über einen VGA-Monitor. Der Softcore-Prozessor *(µBlaze)* wird als Daten-Logger des Bildstromes mit einem Desktop-PC über die RS232-Schnittstelle verbunden (vgl. Anhang B.7.

Im Forschungsprojekt FAUST, *(Fahrerassistenz- und Autonome Systeme)*, der Hochschule für Angewandte Wissenschaften Hamburg werden Hardware/Software-Co-Designs für Echtzeitsysteme am Beispiel eines Fahrerassistenzsystem über die RTL-Codegenerierung von Beschleunigermodulen und der RTL-Modellierung über die Hardwarebeschreibungssprache VHDL erprobt.

Die Visualisierung von Echtzeit-Videodatenströmen mit einem VGA-Monitor eines kameraunterstützenden Fahrerassistenzsystems liefert zusätzliche qualitative Einblicke in das Fahrverhalten und in die Assistenzfunktionen. Der VGA-Monitor kann während der

Videoübertragung ausgewertete Informationen, der jeweiligen Betriebssituation extrahieren und darstellen. Ein Anwendungsbeispiel ist die Anzeige des projektiv entzerrten Bildes und der Vergleich von Filterfunktionen für die Fahrspurerkennung während des Betriebs.

Diese Arbeit dokumentiert die Entwicklungsergebnisse einer Fahrspurerkennung und -visualisierung für einen Fahrspurführungssystems über eine Kopplung der Bildverarbeitungsplattform mit der CCD-Kamera, dem LCD-Touch-Panel und einem weiteren Field Programmable Gate Array (FPGA)-Board, die durch Adapterplatinen mit miniaturisierten Steckverbindern miteinander realisiert ist. Die Visualisierung der Fahrspur über kaskadierte Filtervarianten wird zur Veranschaulichung der Effizienz dargestellt und verglichen. Dabei wird die Schwellwertanpassung zur Graubildbinarisierung mit einer morphologischen Erosion und der Sobel-Operator zur Kantendetektion je nach Anzeigemodi visualisiert, um die Fahrsituation zu erfassen und auszuwerten. Weiterhin wird sowohl die Modellierung mit dem System Generator, als auch die RTL-Modellierung mit der Hardwarebeschreibungssprache VHDL behandelt und diese Modellierungsvarianten gegenüber gestellt.

Schwerpunkte dieser Arbeit:

Filtervarianten: Für die Fahrspurerkennung wurde eine Schwellwertanpassung mit einer Graustufenbinasierung vorgenommen, um die Fahrspurmarkierung aus dem Bild zu extrahieren. Anschließend wurde eine morphologische Erosion zur Kontraktion der Fahrspur ausgeführt. Die beiden kaskadierten Filteroperationen werden mit dem Sobel-Operator gegenübergestellt, die zur Kantendetektion genutzt wird.

Projektive Transformation: Anpassung der projektiven Transformationsmatrix in Bezug auf die angebrachte CCD-Kamera, um ein perspektivisch korrigiertes Bild zu erzeugen und die Optimierung der korrigierten Bildkoordinaten für eine Darstellung der projektiven Transformation.

Zeilenzwischenspeicher-RAM: Der Zeilenzwischenspeicher-RAM wird zur Rekonstruktion der perspektivisch korrigierten Pixelkoordinaten genutzt, um eine Darstellung des entzerrten Bildes zu realisieren. Eine Visualisierungsvariante ist die Anzeige des verzerrten Bildes, während sich das entzerrte Bild in der Region of Interest befindet und die komplette Darstellung des perspektivisch korrigierten Bildes, die über ein User-Switch gewechselt wird.

Erweiterungsadapter für die Bildverarbeitungsplattform: Die Bildverarbeitungsplattform verbraucht einen Ressourcenbedarf von 98% (vgl. Anhang B.6), um weitere Funktionen zu implementieren wird ein FPGA-Board mit der SoC-Plattform durch eine Adapterplatine gekoppelt. Die Kopplung weiterer Adapterplatinen mit der Bildverarbeitungsplattform wird für die Datenübertragung der Kamera bis zur Bildverarbeitungspipeline und für den Anschluss eines LCD-Touch-Panels genutzt.

Matlab/Simulink Simulation: Die Verifikation des System-Generator Models wurde über das Aufrufen eines Matlab-Skripts, die für die Erstellung von zeitabhängigen Eingangsdaten genutzt wird und die anschließende Ausführung der Simulink-Simulation getestet. Anhand der gelieferten Ausgangsdaten der Modelle wurden die Daten mit einem weiteren Skript ausgewertet.

LCD-Touch-Panel: Eine Gegenüberstellung von vier LCD-Touch-Panels und die daraus folgende Auswahl, für eine kompakte Visualisierung der Bildverarbeitungskette und die Kopplung des Displays mit der SoC-Plattform über eine Adapterplatine.

Die Systemübersicht zur Bildverarbeitungsplattform wird in ***Kapitel 2*** vorgestellt. Hierbei werden die Komponenten der Bildverarbeitungsplattform und die gekoppelte SoC-Plattform mit dem Spartan3E 1200K des Nexys2- und dem Spartan3A DSP 3400K des TE0320-Boards erläutert. Weiterhin wird die vorentwickelte Bildverarbeitungspipeline vorgestellt, damit ein Bezug auf die weiterentwickelte Bildverarbeitungskette entsteht.

In ***Kapitel 3*** werden auf die Konzepte und Zusammenhänge dieser Arbeit eingegangen. Es wird hier die weiterentwickelte Bildverarbeitungspipeline und die Komponenten der BV-Kette vorgestellt, die modifiziert bzw. erweitert wurden. Außerdem werden die Erweiterungsadapter der Bildverarbeitungsplattform erläutert und auf die Matlab/Simulink-Simulation mit der anschließenden RTL-Codegenerierung eingegangen.

Auf die Modifikationen und Erweiterungen der Bildverarbeitungskette zur Verbesserung der Fahrspurerkennung wird in ***Kapitel 4*** eingegangen. Die angepassten und hinzugefügten Module werden in Bezug auf den Hintergrund der Optimierung vorgestellt und mit den Analyseergebnissen erläutert.

Kapitel 5 stellt die Erweiterungsadapter für die Bildverarbeitungsplattform vor und referenziert auf die Signallaufzeitpfadtabellen im Anhang A.

Die MATLAB-Simulation zur Verhaltensanalyse der Filterfunktionen sowie die Simulationsergebnisse der Ausgangswerte werden in ***Kapitel 6*** beschrieben.

Die Ergebnisse dieser Arbeit bezogen auf die Funktion und das Verhalten der Bildverarbeitungsmodule werden in ***Kapitel 7*** präsentiert.

In ***Kapitel 8*** wird eine konzeptionelle Weiterentwicklung vorgestellt, die als Schwerpunkte nachfolgender Bachelor- und Masterthesen zur Verfügung stehen.

Den Abschluss der Arbeit bildet ***Kapitel 9*** mit einer Zusammenfassung.

2. Systemübersicht zur Bildverarbeitungsplattform

Dieser Abschnitt erläutert die Komponenten der Bildverarbeitungsplattform bestehend aus dem Nexys2-Entwicklungsboard, einer Sony FCB-PV10 CCD-Farbbildkamera [Son06] und einem Hitachi TX14D12VM1CAA LCD-Touch-Panel [Hit06] mit einem integrierten TimeSync-Controller (vgl. Abb. 2.1). Der Erweiterungsadapter(FFC-IDC und IDC-PMOD) koppelt die CCD-Kamera mit der PMOD-Schnittstelle des Nexys2-Boards [Dig08]. Das LCD wird an die FX2-Schnittstelle über einen weiteren Adapter(LCD-Touch-Panel-FX2) angeschlossen, sodass die Videodaten, die mit der Bildverarbeitungskette der SoC-Plattform verarbeitet wurden, mit dem LCD-Touch-Panel anzeigbar werden.

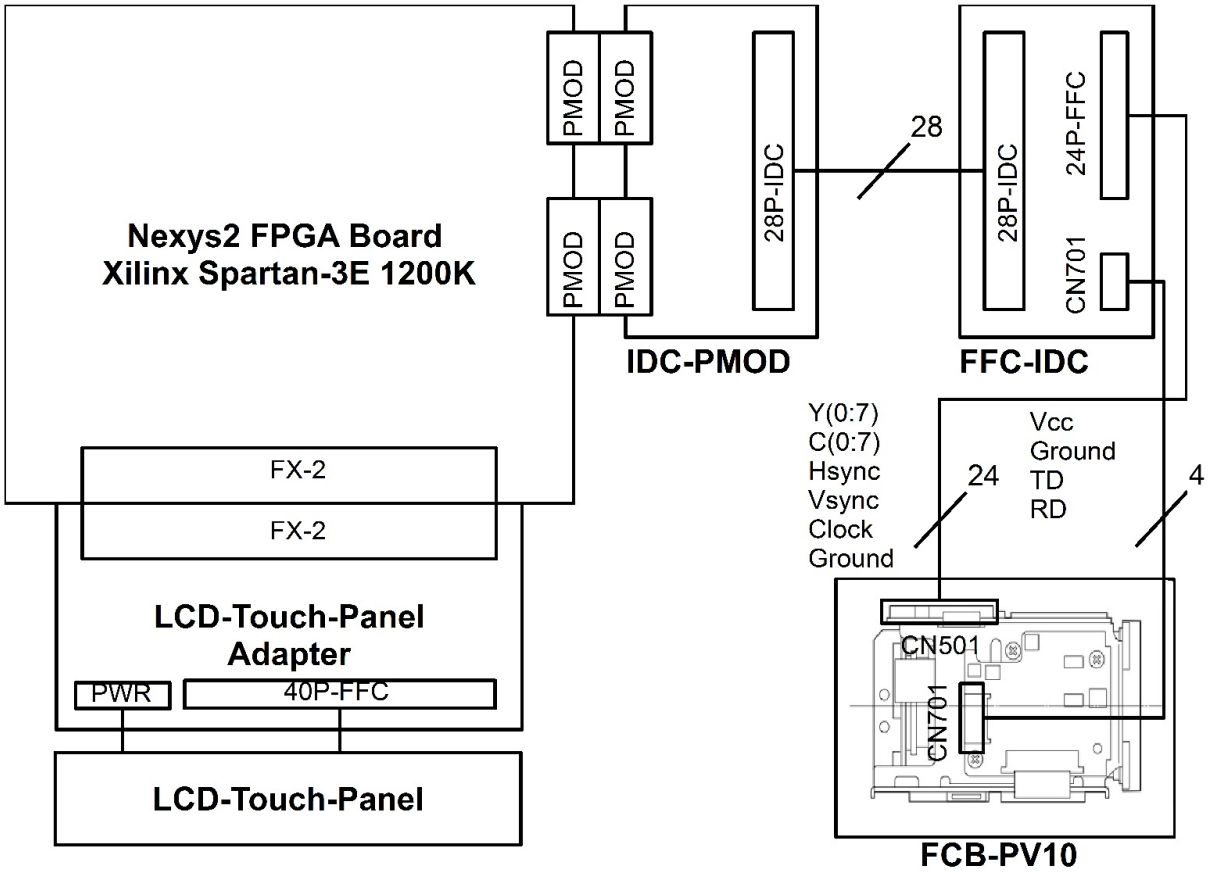

Abbildung 2.1.: SoC-Plattform mit Komponentenkopplung über Erweiterungsadapter zur Visualisierung mit einem LCD.

2.1. CCD-Kamera FCB-PV10

Die Kamera ist mit einem Interline-Transfer CCD-Sensor ausgestattet und liefert je nach Betriebsart 16 Bit, 8 Bit Progressive- oder 8 Bit Interlace-Modus im YCbCr-Farbmodell die digitalen Bilddaten. Diese werden mit einer Bildauflösung von 640x480 Pixeln pro Bild bereitgestellt. Die Bildrate kann zwischen 29,97 und 25 Bildern pro Sekunde eingestellt werden.

Interline-Transfer: Der CCD-Sensor besteht aus lichtempfindlichen Zellen, die das Licht aufnehmen und in Form von elektrischer Ladung weiterreichen. Interline-Transfer ist ein Ladungstransfer, bei dem neben den lichtempfindlichen Zellen auch abgedunkelte

Zellen bereitgestellt werden, die nach der Belichtung die Ladung aufnehmen und in ein Transferregister Zeile für Zeile übergeben (vgl. Abb. 2.2). Der Transferregister wird dann seriell ausgelesen und die elektrischen Ladungen werden über einen ADC digitalisiert [vgl. Erh08, S. 28 - 34].

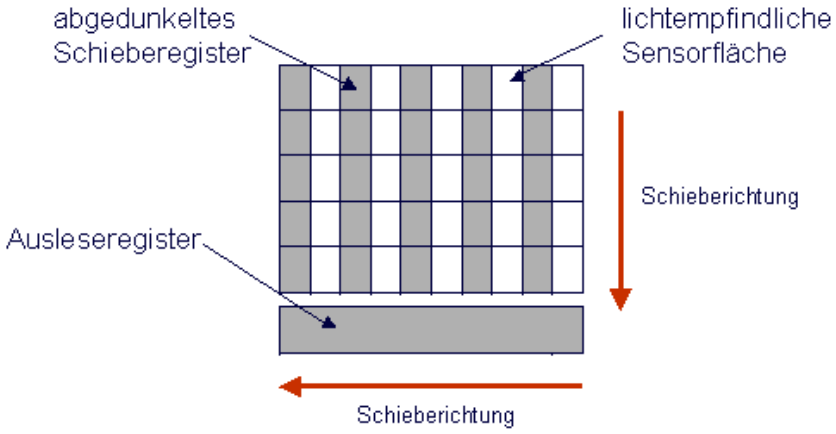

Abbildung 2.2.: Funktionsweise des Interline-Transfer CCD-Sensors über parallele und serielle Schiebetechniken.

YCbCr-Farbmodell: Das YCbCr-Farbmodell ist ein Helligkeits-Farbigkeits-Modell. Die Helligkeit wird mit der Y Komponente bestimmt und die Farbigkeit über die Cb(Chrominanz-Blue) und Cr(Chrominanz-Red) Komponenten. Die Farbunterabtastung ist ein Verfahren, bei dem der Chrominanz-Kanal gegenüber dem Luminanz-Kanal eine geringere Abtastung aufweist um die übertragende Datenmenge zu reduzieren. Dabei ist keine sichtbare Qualitätsverringerung des gesamten Bildes vorhanden. Der FCB-PV10 verwendet eine Farbunterabtastung mit den Werten 4:2:2, die 4 steht für die Abtastrate des Luminanz-Kanals. Die zweite Ziffer steht für die Abtastrate des Chrominanz-Kanals und mit der dritten Ziffer kann die Abtastung der verschiedenen Raumrichtungen unterschieden werden. Somit ist die Abtastung in horizontaler Richtung nur halb so groß wie in vertikaler Richtung (vgl. Abb. 2.3).

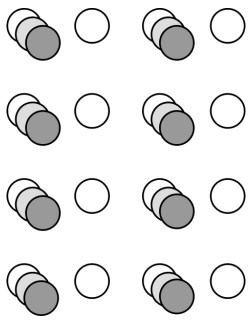

Abbildung 2.3.: Farbunterabtastung über YCbCr 4:2:2 zur Reduzierung der zu übertragenden Datenmenge für Chrominance Blue und Chrominance Red.

Interlace: Interlace ist ein Zeilensprungverfahren, bei denen zwei Halbbilder nacheinander übertragen werden, um die Bildrate zu verdoppeln. Somit wird ein flackern des Bildes verhindert und ein fließendes Bild angezeigt. Die Halbbilder entstehen, in dem nur jede zweite Zeile eingelesen wird (vgl. Abb. 2.4).

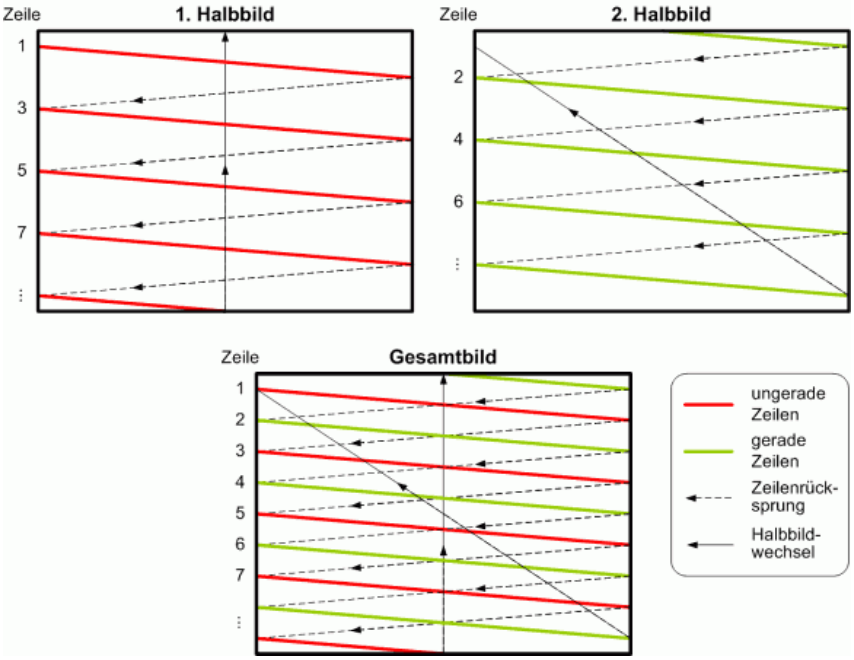

Abbildung 2.4.: Zeilensprungverfahren über gerade und ungerade Halbbildern zur Gesamtbildgenerierung.

2.2. SoC-Plattform mit einem Spartan3E1200 FPGA

Das Nexys2-Evaluationsboard wird als FPGA-Plattform für die Bildverarbeitungspipeline verwendet (vgl. Abb. 2.5). Über die PMOD-Schnittstelle wird die Kamera angeschlossen und der Bildstrom wird über das Nexys2 verarbeitet. An die FX2-Schnittstelle wird ein LCD-Touch-Panel gekoppelt.

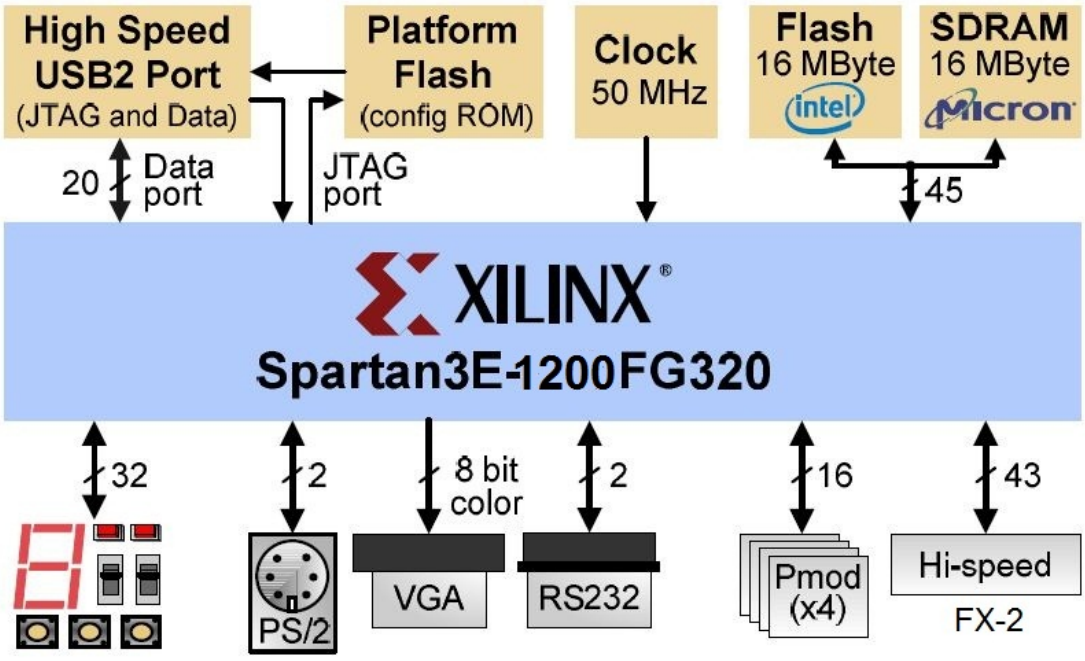

Abbildung 2.5.: Digilent Nexys2-Entwicklungsboard mit dem Spartan 3E-1200K FPGA.

Das Evaluationsboard wird mit einem integrierten MicroBlaze-Softcoreprozessor [Xil10a] als SoC-Plattform genutzt, um eine Steuerungslogik für das Fahrspurführungssystem und als Datenlogger einzusetzen, das über ein Framegrabber-Modul mit dem Desktop-PC

angeschlossen wird.

Das Xilinx Spartan-3E XC3S1200E FPGA bietet die Ressourcen [Xil09]:

- 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 SLICEM und SLICEL (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-Flop (D-FF)

- 17344 LookUp-Table (LUT): Der LookUp-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 Block (IOB): IOBs sind Kontrollblöcke zwischen den I/O Pins und den CLBs. Der Input/Output Block ist eine programmierbare uni- oder bidirektionale Schnittstelle und unterstützen den Tristate-Betrieb. Für eine schnellere Datenübertragung wird ein spezieller Multiplexer eingesetzt, indem zwei D-FFs die synchronisierten Daten einmal über die steigende und die fallende Taktflanke abwechselnd an den Multiplexer weiterleiten.

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

- 28 Block RAM (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-Modi's 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 Manager (DCM): Der Digitale 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 hat die DCM:
  - 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: Die 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 Eingangstaktsignals auf die Ausgangstaktsignale vorgenommen.

2.3. 5.7" LCD-Touch-Panel mit integriertem Timing-Controller

Für Visualisierungszwecke wird ein kompaktes Display ausgewählt, damit während der Fahrt des Modellfahrzeugs die kritischen Bereiche der Fahrspur erkennbar werden. Die Anzeige über ein VGA-Monitor stellt die Region of Interest dar, in der die Pixel eintreffen, die für die Geradenapproximation der Fahrspur zur Datenreduktion über die Hough-Transformation berechnet wird.

Das mit der projektiven-Transformation entzerrte Bild ist ebenfalls für die Anzeige von Bedeutung, da die Effektivität der Schwellwertanpassung für die Graubildbinarisierung in Verbindung mit der morphologischen Erosion ermittelt wird und die Gegenüberstellung des Sobelfilters durch die Visualisierung stattfindet. Weiterhin wird die berechnete Gerade im entzerrten Bild angezeigt und damit die Berechnung verifiziert, indem die Gerade auf der entzerrten Fahrspur liegt.

2.3.1. Gegenüberstellung von Displayparametern

Das TX14D12VM1CAA [Hit06] Display wurde nach folgenden Kriterien ausgewählt, damit eine Darstellung der Bildinformation zustande kommt und eine Kontrolle für die Wirksamkeit der Filterfunktionen erfolgt:

Die Bildübertragungsfrequenz der Kamera beträgt 27 MHz bei einer Auflösung von 640(H) x 240(V) und wird auf 13.5 MHz halbiert, weil nur die Helligkeitskomponente des Bildes für die Bildauswertung genutzt wird. Die Bildwiederholfrequenz ist dabei größer als 25 Hz, da ansonsten bei schnellen Aufzeichnungen das Bild nicht fließend erscheint.

Für die Anzeige wird die Auflösung horizontal halbiert, indem man nur jedes zweite Pixel abtastet, dadurch dass die maximale Auflösung des LCD-Touch-Panels 320(H) x 240(V) beträgt. Durch die Abtastung jedes zweiten Pixels wird die Bildübertragungsfrequenz auf 6.75 MHz verringert und passt somit zu den Anforderungen des Displays. Die in der Tabelle aufgelisteten Displays wurden nach diesem Kriterium ausgewählt und durch weitere Einschränkungen eine Auswahl getroffen (vgl. Tab. 2.1).

Die Stromversorgung wird über die Fahrzeugbatterie bezogen, ein geringer Stromverbrauch führt zu einer längeren Fahrt des Fahrzeugs und die Visualisierung der Bildinformation. Weiterhin ist es von Vorteil, wenn die Betriebsspannung über das Nexys2 bezogen wird, da die Anzeige mit diesem gekoppelt ist.

Die vom Nexys2 ausgehenden Spannungen sind 3.3V und 5V, die zu den Spannungskriterien der ausgewählten Displays passt.

Hersteller	Hitachi	Toshiba	Sharp	Sharp
Her. Bez.	TX14D12VM1CAA	LTA057A344F	LQ050Q5DR01	LQ057Q3DC02
Displaygröße	5.7"(14.4 cm)	5.7"(14.4 cm)	4.96"(12.6 cm)	5.7"(14.4 cm)
Auflösung	320(H) x 240(V)	320(H) x 240(V)	320(H) x 240(V)	320(H) x 240(V)
Farben	262144	262144	262144	262144
Pin Anzahl	40 Pins (CMOS)	33 Pins	40 Pins	33 Pins
Touch-Panel	4 Pins (Resistiv)	4 Pins (Resistiv)	4 Pins (Resistiv)	4 Pins (Resistiv)
Preis	176.00 – 188.85 €	134.62 – 159.33 €	172.45 – 480.00 €	393.84 €

Electric Characteristics				
Versorgungsspannung	3.0 – 3.6 V	3.0 – 3.6 V	2.9 – 3.7 V	3.0 – 3.6 V
Stromverbrauch	65 mA	150 mA	140 – 180 mA	130 – 160 mA

Optical Characteristics				
Lebensdauer	50000 Std.	20000 Std.	20000 Std.	Min: 40000 Std. Typ: 50000 Std.

Timing Characteristics				
Clock Frequenz	Min: 4.85 MHz Typ: 5.85 MHz Max: 16.66 MHz	Min: 6.41 MHz Typ: 6.66MHz Max: 8.33 MHz	Min: 4.5 MHz Typ: 6.3 MHz Max: 6.8 MHz	Min: — Typ: 6.3 MHz Max: 7.0 MHz

Tabelle 2.1.: Tabelle zum Vergleich der untersuchten Touch-Panel-Datenblätter [Hit06], [Tos07], [Sha07], [Sha02]

2.3.2. Ansteuerung des LCD-Touch-Panels

Der TX14D12VM1CAA LCD hat eine Bilddiagonale von 14.4 cm und eine Auflösung von 320(H) x 240(V) Pixel. Die Bildwiederholrate beträgt zwischen 20 und 188 Bildern pro Sekunde und lässt damit einen großen Spielraum für die übertragende Bildfrequenz. Die Anzeige des Bildstromes erfolgt mit der Übertragung eines Datenstroms *(Data)*, einem Datenberechtigungs-Signal *(DTMG)* und dem Taktsignal *(Pixel-Clock)* (vgl. Abb. 2.6).

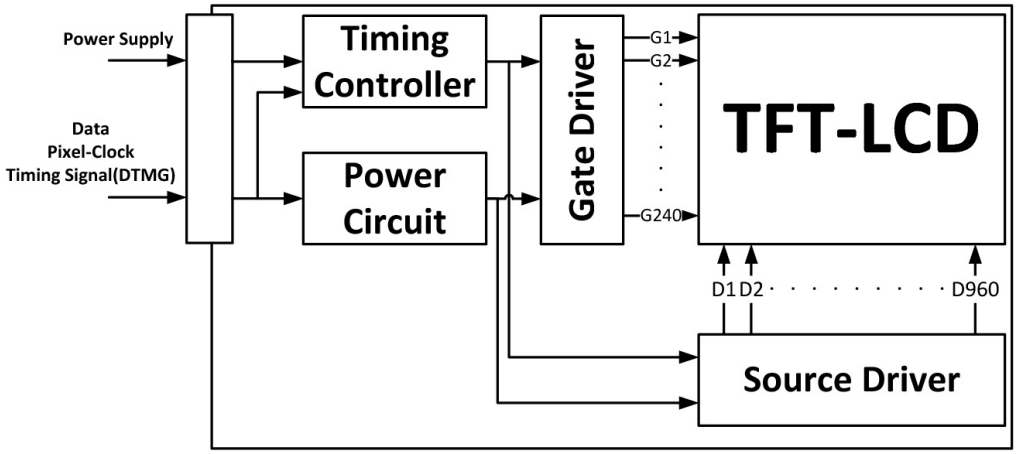

Abbildung 2.6.: LCD Block-Diagramm zur Ansteuerung des LCD-Touch-Panels

Die Synchronisationssignale HSync und VSync, für die horizontale und vertikale Bildsynchronisation, werden über den integrierten Timing Controller des LCD-Touch-Panels anhand eines Datenberechtigungs-Signals und der Pixel-Clock erzeugt. Der LCD-Touch-Panel wird mit einer Betriebsspannung von 3.3V versorgt, die er von der FX2-Schnittstelle empfängt. Mit einem geringen Stromverbrauch von nur 65mA ist das Display für das RC-Fahrzeug geeignet.

2.4. Trenz-Electronic Spartan 3A DSP 3400K

Das TE0320-Board [Tre10] wird mit dem Nexys2-Evaluationsboard gekoppelt, da der Ressourcenfüllstand des Nexys2 bei 98% der Gesamtressourcen des FPGA's liegt (vgl. Anhang B.6). Die Erweiterung liefert zusätzlich zu den Schnittstellen des Nexys2 zwei Stiftleistenblöcke mit jeweils 24- und 32-Pins an, die für weitere Sensorik nutzbar sind. Die als JM5 bezeichnete B2B-Schnittstelle des TE0320-Boards ist an die Adapterplatine gesteckt, die an den FX2-Stecker gekoppelt ist. Somit ist die Verteilung der Ressourcen für die Bildverarbeitungspipeline über die gekoppelte SoC-Plattform ermöglicht.

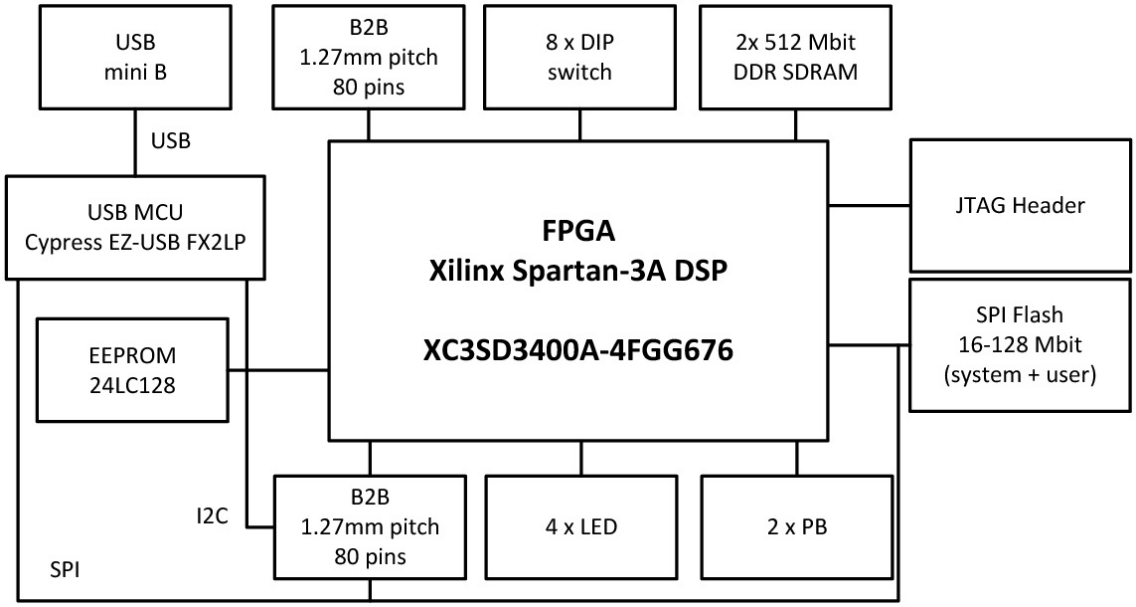

Abbildung 2.7.: Trenz-Electronic TE0320-Board mit dem Spartan-3A DSP 3400A FPGA

Gegenüberstellung des Xilinx Spartan-3A DSP XC3SD3400A [Xil10b] mit dem Spartan-3E XC3S1200E [Xil09]:

Ressource/Typ	XC3SD3400A	XC3S1200E
CLB	5968	2168
Slice(M/L)	23872	8672
Daten Flip-Flop	47744	17344
LookUp-Table	47744	17344
Input/Output	469	250
Dedicated Multiplier	—	28
Block RAM (18kBit)	126	28
DCM	8	8
DSP	126	—

Der SPI-Flashspeicher und der I2C-EEPROM sind als Slave-Komponenten im jeweiligen Datenübertragungsprotokoll eingetragen und werden je nach Konfiguration (vgl. Tab. 2.2 und 2.3) mit dem USB Microcontroller, dem FPGA oder der B2B-Schnittstelle zum Datentransfer berechtigt (vgl. Abb. 2.8). Findet eine Übertragung über SPI vom USB-Microcontroller oder von der B2B-Schnittstelle zum Flashspeicher statt, so wird das PROM-File (FPGA-Bitstream Konfiguration) in das SPI-Flash geschrieben. Wird bei der SPI-Konfiguration das FPGA als Master ausgewählt, so wird die FPGA-Konfigurationsdatei aus dem Flashspeicher ausgelesen.

description	**EZ-USB**	**FPGA**	**B2B JM5**	**serial Flash**
EZ-USB → Flash	**master**	off *(S2=FX2PON, FX2_PS_EN=0)*	deselected	slave
FPGA ← Flash	inactive *(SPI_*=Z)*	**master** *(SPI_/S=1)*	deselected	slave
B2B → Flash	inactive *(SPI_*=Z)*	off *(S2=FX2PON, FX2_PS_EN=0)*	**master** *(SPI_/S=0)*	slave

Tabelle 2.2.: Tabelle zur Konfiguration der Datenübertragungsstrecke des SPI-Busses [Tre10].

Der I2C-Bus wird typischerweise über den USB-Microcontroller genutzt, um das EEPROM mit dem USB-Firmware zu beschreiben. Dabei ist es aber auch möglich über die anderen Komponenten (FPGA, B2B-Schnittstelle) Lese- und Schreibzugriffe auszuführen, wenn diese sich dann im Master-Mode befinden. Lediglich der USB Microcontroller wird inaktiv gesetzt, falls das FPGA oder der B2B-JM5 im Master-Mode konfiguriert ist.

Core	**EZ-USB FX2LP**	**FPGA**	**B2B-JM5**	**serial EEPROM**
default	**master**	slave *(SCL=I)*	slave	slave
custom	inactive *(SCL=SDA=Z)*	**master** *(SCL=O)*	slave	slave
custom	inactive *(SCL=SDA=Z)*	slave *(SCL=I)*	**master**	slave

Tabelle 2.3.: Tabelle zur Konfiguration der Komponenten des I2C-Busses [Tre10].

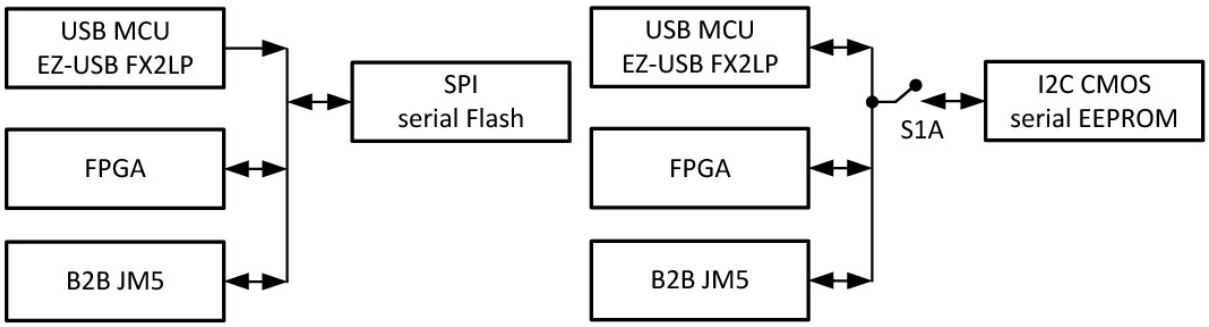

Abbildung 2.8.: Master-Slave-Datenbus für die Datenübertragung an die Flash- und EEPROM-Komponente [Tre10].

Der Spartan 3A DSP stellt zusätzlich zu den umfangreichen FPGA-Ressourcen DSP48-Slices zur Verfügung, die für die Verarbeitung und Analyse digitalisierter analoger Signale verwendet wird, beispielsweise in der Digitalbildverarbeitung sowie die Signalverarbeitung

in der Kommunikationstechnik. Dabei steht die 48 für die am Ausgang zur Verfügung gestellte Bitbreite.

In einer DSP-Funktion sind stets Multiplikationen sowie Addition und Subtraktion enthalten (vgl. Abb. 2.9). Durch den Bit-Wachstum wird die Wortbreite mit den LSB-Funktionen Runden/Abschneiden reduziert. Die Rundung reduziert dabei die Bitbreite, ist aber auch eine Rauschquelle und steuert zu Störsignalen am Ausgang bei und wirkt sich deshalb auf das Signal-Rausch-Verhältnis aus. Durch die Kaskadierung von DSP-Slices sind größere Wortbreiten zu erzeugen, indem die dedizierten Ein-/Ausgänge genutzt werden.

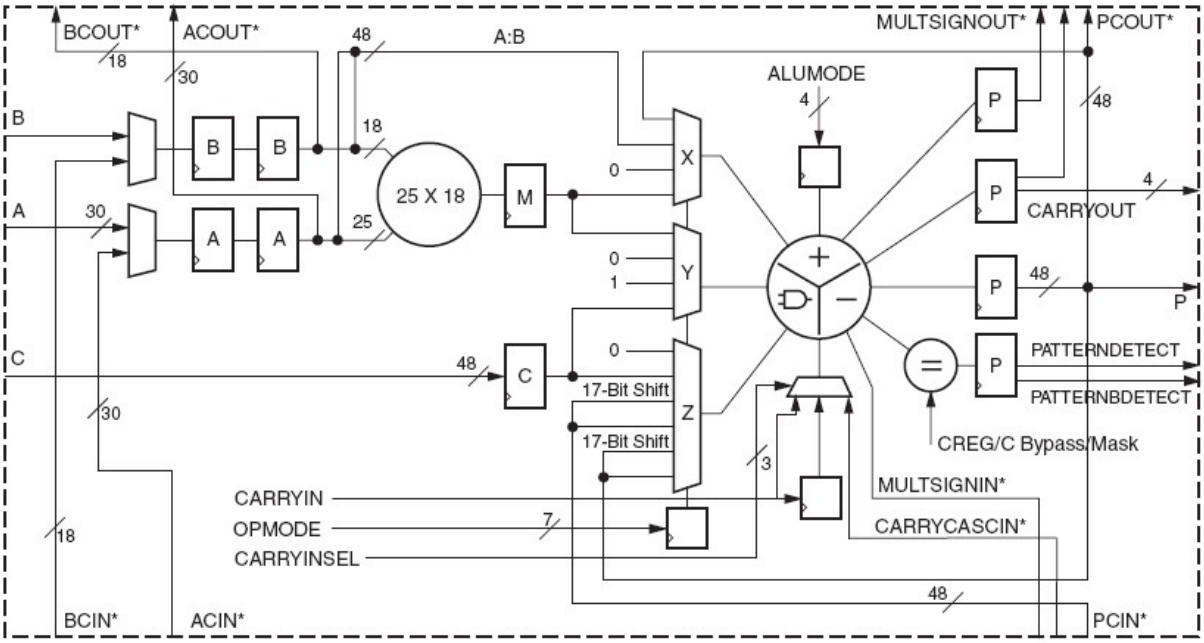

Abbildung 2.9.: Xtreme DSP-Slice mit Carry-Arithmetik, Pre-Adder und dedizierten Ein-/Ausgängen für Kaskadierungen und mit einer Ausgangsbitbreite von 48.

2.5. Gekoppelte SoC-Plattform

Der Nexys2-Evaluationsboard (vgl. Abschnitt 2.2) wird mit dem TE0320-Board (vgl. Abschnitt 2.4) gekoppelt, um für weitere Entwicklungsschritte Freiheitsgrade zur Bildverarbeitungspipeline und zusätzliche Schnittstellen für Sensorik zu schaffen. Die Verbindung der FX2-Schnittstelle mit der B2B-Schnittstelle stellt eine Datenübertragungsstrecke zwischen zwei FPGA's her (vgl. Abb. 2.10).

Die 32P- und 24P-Stiftleisten werden zur Komponentenanbindung für zusätzliche Sensorik bzw. zur Signalkontrolle genutzt. Der TE0320 ist mit der Adapterplatine über die zur Verfügung gestellte JTAG-Schnittstelle beschreibbar. Die Datenübertragung ist taktsynchron, da die Signallaufzeit zwischen den Schnittstellen geringer ist als das Taktsignal und somit immer eine Eintaktung der Daten bei einer HIGH-Flanke erfolgt.

Die Übertragung muss vor Signalausfällen und Umschaltungen der E-/A-Pins geschützt werden, daher werden die Tristate-Ausgänge zusätzlich zu den Zuständen 0 und 1 auf einen Hochohmigen Zustanden 'Z' angelegt.

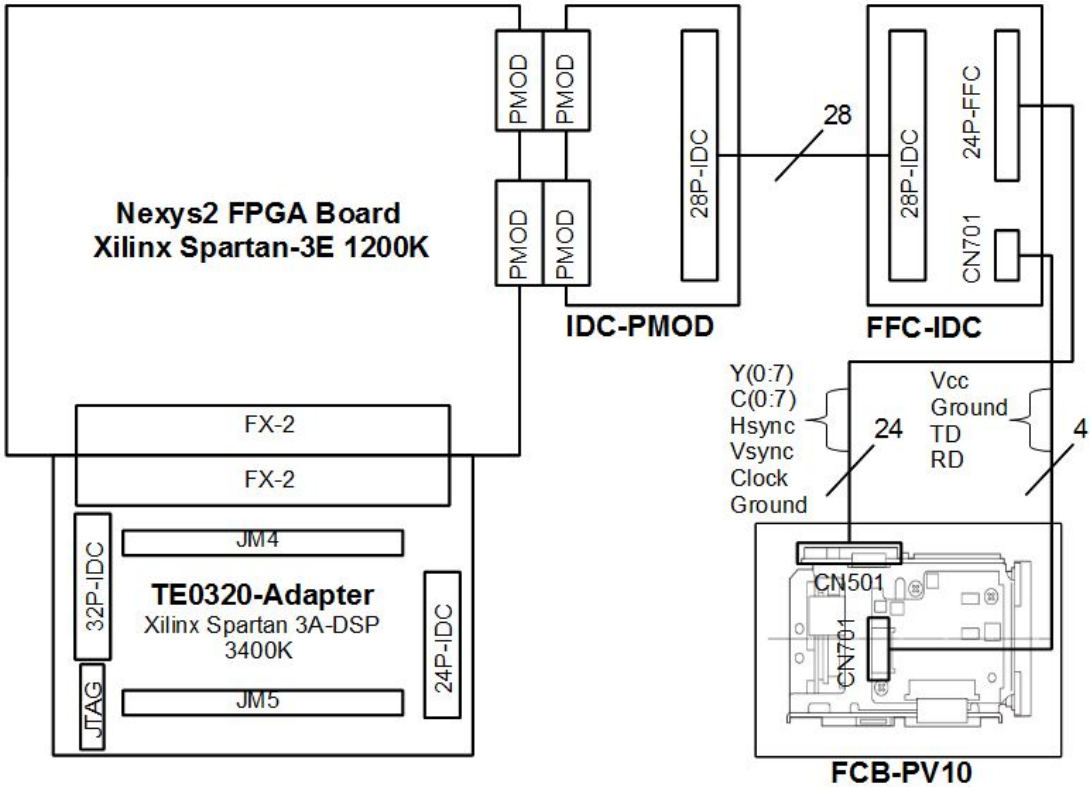

Abbildung 2.10.: Gekoppelte SoC-Plattform mit IDC-PMOD- und FFC-IDC-Adapter für die Verbindung der FCB-PV10 CCD-Kamera mit dem Nexys2 FPGA Board und die Verknüpfung der FPGA's über den TE0320-Adapter

2.6. Systemübersicht zur Vorentwicklungsstufe

Der Videodaten-Strom wird mit 60 Halbbildern pro Sekunde von der Sony FCB-PV10 CCD-Kamera (vgl. Abschnitt 2.1) an die Synchronisationskomponente der Bildverarbeitungspipeline übergeben (vgl. Abb. 2.11). Die Sync.-Komponente generiert aus dem YCbCr-Signal die Signale Line-Valid und Frame-Valid. Das YCbCr-Signal wird über eine Deserialisierung in die Signale Y (Graustufenwert), Cb (Chrominance Blue) und Cr (Chrominance Red) aufgeteilt (vgl. Abschnitt 2.1, Beschreibung: *YCbCr-Farbmodell*). Dabei wird nur der Graustufenwert verwendet. Um die Fahrspur zu erkennen, wird ein 3x3-Sobelfilter angewendet, der die Kanten des Bildes über die Kontrastunterschiede extrahiert.

Eine geringere Datenübertragung und somit weniger FPGA-Ressourcen werden erzielt, indem das Graustufensignal in ein Schwarz/Weiss-Signal über eine Binarisierung umgewandelt wird. Die im Bereich der Region of Interest liegenden Kantenpixel werden an die Hough-Transformation übergeben, die daraus eine Gerade approximiert und abbildet.

Mit dem Microblaze-Softcoreprozessor wurden die Ergebnisse der Hough-Transformation in S/W-Register gespeichert und über zwei ausgewählte Punkte der Geraden eine projektive Entzerrung des Bildes ausgeführt. Der VGA-Monitor hat das Sobel gefilterte Videosignal und die von der Hough-Transformation erkannte Gerade in der Region of Interest dargestellt [Hor11].

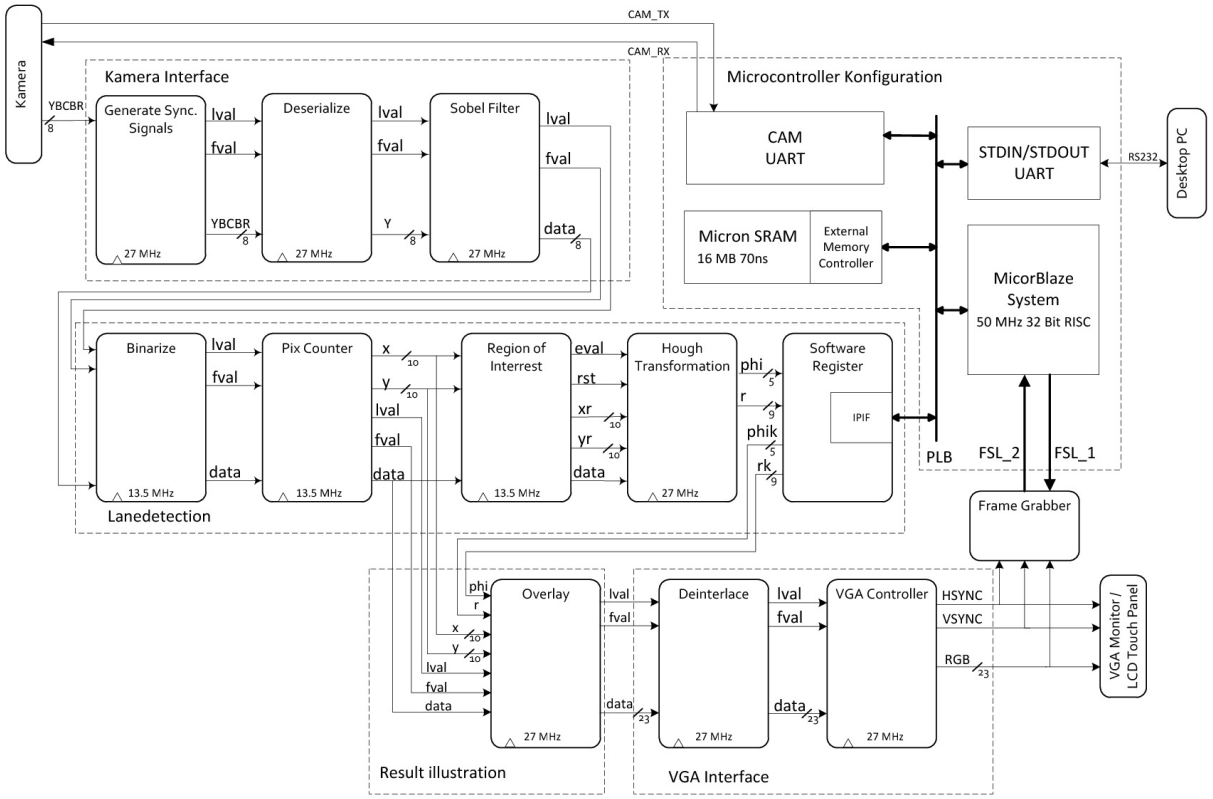

Abbildung 2.11.: Bildverarbeitungspipeline in der vorentwickelten Version mit einem Sobel-Filter zur Kantenerkennung und integriertem Softcoreprozessor, der eine projektive Transformation der approximierten Gerade ausführt [Hor11].

Die einzelnen Komponenten der vorentwickelten Bildverarbeitungspipeline wurden teilweise mit Matlab/Simulink simuliert und zu einer Netzliste synthetisiert. Dabei wurden die Simulationen auf Basis mathematischen Grundlagen der Bildverarbeitung aufgebaut. In der weiterentwickelten Bildverarbeitungspipeline wurde durch die Erkenntnis der Datenausgabe des Videostroms eine Anpassung der Module und die Erweiterung der Bildverarbeitungskette vorgenommen.

3. Konzept und Kohärenz

Den Schwerpunkt dieses Abschnittes bildet eine Übersicht zu den realisierten Modifikationen und Erweiterungen der SysGen-Module für die Fahrspurerkennung und die Visualisierung des entzerrten Bildes. Die Anzeige stellt in Echtzeit das projektiv entzerrte Videobild dar und kann über die Filtervarianten Sobel-Filter, Schwellwertanpassung mit Graustufenbinarisierung und einer morphologischen Erosion, die über User-Switches kombinierbar sind, zu einem Ergebnis der Fahrspurerkennung und -führung Beiträge leisten. Auch die Platinenentwicklung der Erweiterungsadapter für die gekoppelten SoC-Plattformen und die Integration der Bildverarbeitungspipeline ist in dieser Arbeit dokumentiert.

Des Weiteren wird die Systemmodellierung über System-Generator und die Simulation der modifizierten und erweiterten Module dokumentiert.

3.1. Systemübersicht zur weiterentwickelten Bildverarbeitungsplattform

Die Bildverarbeitungsplattform wurde auf ein RC-Fahrzeug installiert und an dessen Spannungsversorgung angeschlossen, um eine Analyse der Eigenschaften der vorentwickelten Bildverarbeitungskette (vgl. Abschnitt 2.6 Abb. 2.11) durchzuführen und anzupassen. Durch die Analyse der Vorentwicklung ergaben sich folgende Eigenschaften:

- Im perspektivisch verzerrtem Bildstrom liegt die Fahrspurmarkierung in einer Kurvenlage nicht innerhalb der fest positionierten Region of Interest. Dies führt zu einer fehlerhaften Geradenapproximation, da die Hough-Transformation nicht die Bildpunkte der Fahrspurmarkierung übertragen bekommt, sondern die außerhalb der Kurve liegenden Pixel.

- Die fehlerhaften Geradenparameter werden perspektivisch entzerrt und führen zu weiteren verfälschten Ergebnissen.

- Ein fester Schwellwert für die Graustufenbinarisierung führt bei unterschiedlichem Lichteinfall zu schlechten Binarisierungsergebnissen.

- Die Kantenextraktion über den Sobel-Filter ist bei einer projektiven-Transformation nicht als Gerade wiederzuerkennen, da im Bild Lücken entstehen und dies zu einer oszillierenden Gerade führt.

Ausgehend von den erkannten Eigenschaften der vorentwickelten Bildverarbeitungskette wurde eine gezielte Anpassung vorgenommen:

- Eine dynamische Schwellwertanpassung für die Graubildbinarisierung führt zu einer sauberen Fahrspur die für die weitere Bildverarbeitungspipeline verwendet wird.

- Die Geradenapproximation erfolgt über einem perspektivisch entzerrtem Bild, damit die Fahrspur zu jedem Zeitpunkt im Bildbereich der ROI zu erfassen ist.

- Die Geradenparameter stehen im metrischen Zusammenhang zur Fahrzeugebene.

- Die ROI wird entsprechend dem Analyseergebnis vertikal fest positioniert und auf der horizontalen Achse verschoben, damit die Fahrspur immer in der Mitte der ROI befindet.

In der weiterentwickelten Bildverarbeitungspipeline sind die Komponenten Upsample und die Konvertierung von YCbCr zu RGB nicht vorhanden, da nur der Grauwert Y für die weitere Bildverarbeitung brauchbar ist (vgl. Abb. 3.1). Die Informationen aus dem Graustufenbild werden über kaskadierte Filtervarianten entnommen, die die störenden Anteile reduzieren und informative Anteile des Musters hervorheben. Die Kaskadierung dient zur Veranschaulichung der Effizienz der einzelnen Filtervarianten und deren Kombination.

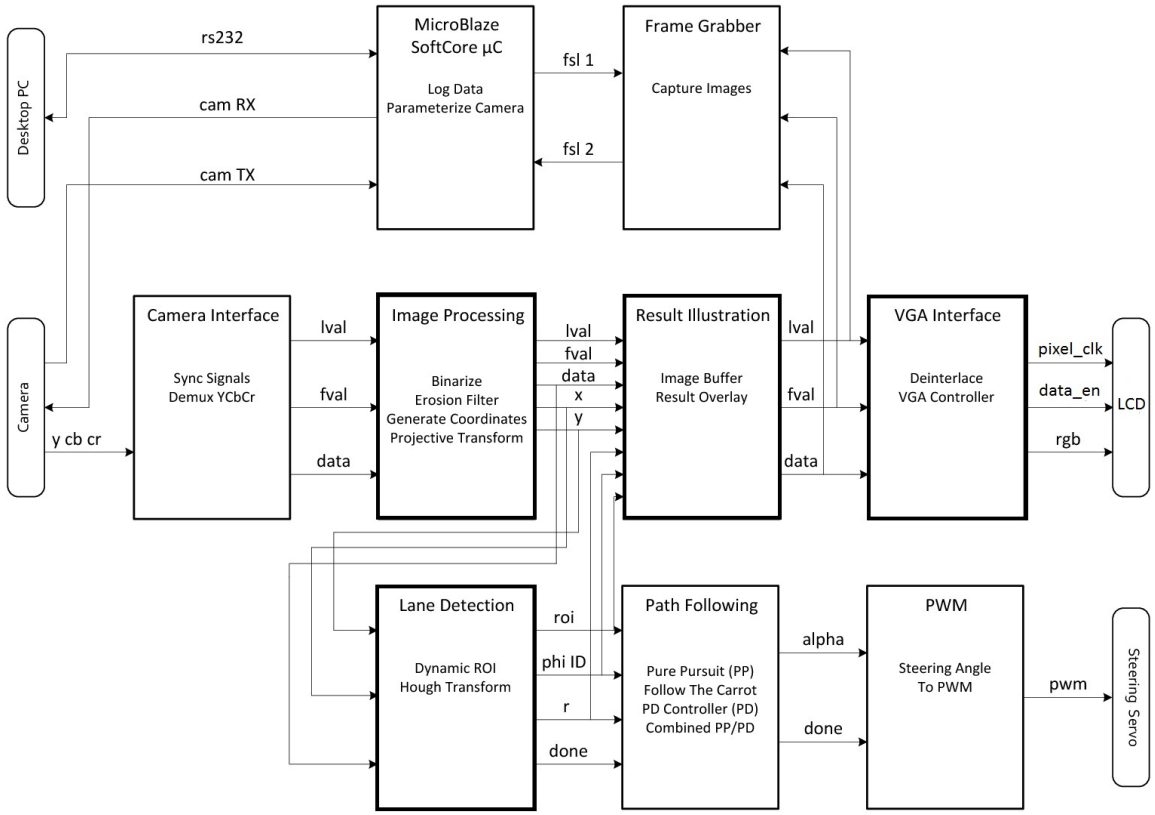

Abbildung 3.1.: In der weiterentwickelten Bildverarbeitungskette wurde im Subsystem *'Image Processing'* die projektive Transformation modifiziert und mit dem *'Adaptive-Binarize'* und dem *'Erosionfilter'* erweitert. Das Subsystem *'Result Illustration'* wurde mit dem *'Image-Buffer'* erweitert um eine perspektivische Entzerrung darzustellen. Im *'Lane Detection'* wurde die Hough-Transformation angepasst und eine dynamische ROI angebunden [Sch11]. Im Anhang B.9 sind die Subsysteme im VHDL-Code an die Bildverarbeitungspipeline als Komponenten gekoppelt.

Die projektive Entzerrung des Bildes wurde für das vorhandene RC-Fahrzeug modifiziert, da die Positionierung der Kamera (vgl. Abschnitt 2.1) und somit die Matrix, die zur mathematischen Lösung der entzerrten Koordinaten genutzt wird, sich verändert hat. Um genauere Koordinaten zu erlangen wurde die Auflösung verdoppelt. Dabei wurde die Bitbreite erhöht und der Koordinatenwert um ein Bit nach rechts verschoben. Über die Visualisierung des entzerrten Bildes wurde die Region of Interest auf ein Bereich fixiert, wo die Fahrspur am besten getroffen wird. Da die Kamera [Son06] eine kurze Blickweite

besitzt, wird für die Berechnung der Fahrspurerkennung über die Hough-Transformation nur eine Fahrstreifen verwendet.

Der maximale Winkelbereich wurde über die Darstellung des entzerrten Bildes erkannt und die Breite der Region of Interest sowie auch die dadurch entstehende Größe des Hough-Raumes verkleinert. Die Visualisierung hat sowohl das verzerrte Bild als Anzeigevariante, als auch das entzerrte Bild. Die Kombination der beiden Bildvarianten ist dadurch entstanden, indem die Region of Interest das entzerrte Bild anzeigt, während das Gesamtbild verzerrt bleibt.

3.2. Zeilenzwischenspeicher-RAM für eine Rekonstruktion von Bildkoordinaten

Der Zeilenzwischenspeicher-RAM wird für die Wiederherstellung der unsystematisch ankommenden korrigierten Koordinaten genutzt, die aus der projektiven Transformation weitergeleitet wurden, indem diese in die richtige Reihenfolge geordnet werden und somit das Verhältnis Pixel zu Koordinate rekonstruiert wird. Dabei ist erstmal festzulegen, wie groß der zu speichernde Bildbereich ist, um den Adressraum der Pixelkoordinaten zu ermitteln.

Die beiden Synchronisationszähler HCount und VCount bestimmen die Reihenfolge der Koordinaten im Zeilenzwischenspeicher-RAM. Der Adressgenerator ist dynamisch aufgebaut, indem über die Größe des Bildbereiches die maximal zu erreichende Adresse ermittelt und passend dazu die Anfangskoordinate des Bereiches zum Ursprung verschoben wird. (vgl. Gl. 3.1). Die verschobene Y-Koordinate y_{origin} wird mit der horizontalen Breite des Bildbereiches multipliziert, um die vertikale Position des Pixels festzustellen und mit der X-Koordinate x_{origin} addiert, damit die endgültige Position bestimmt wird und als Adresse $address_{pt}$ weitergeleitet.

$$\begin{aligned} x_{origin} &= x_{pt} - x_{ptstart} \\ y_{origin} &= y_{pt} - y_{ptstart} \\ y_{area} &= y_{origin} * (x_{ptend} - x_{ptstart}) \\ address_{pt} &= y_{area} + x_{origin} \end{aligned} \tag{3.1}$$

Die systematischen Koordinaten ($address_{pt}$) sind nach einem Bild Verzögerung vorhanden und darstellbar. Bei 60 Bildern pro Sekunde ist ein Bild in 16.6 ms verfügbar und ist daher im Rahmen der Echtzeit-Anforderung, da die Pulsweitenmodulation des Lenkservos eine Periode von 20 ms hat.

Die über den Zeilenzwischenspeicher-RAM geordneten Koordinaten werden an den VGA-Controller übergeben und haben keinen Bezug zu der Berechnungssequenz, die über die Region of Interest bis zur Fahrspurführung abläuft.

3.3. LCD-Controller

Der LCD-Touch-Panel wird über einen LCD-Controller mit der Bildverarbeitungspipeline synchronisiert. Der Controller überträgt ein Digital-RGB Video Signal, das ver-

gleichbar mit dem Bildübertragungsstandard VGA ist. Die Synchronisation erfolgt mit einem DTMG-Signal, der mit dem Line-Valid gleichzusetzen ist. Ein integrierter Timing-Controller sorgt für die Generierung der Synchronisationssignale HSync und VSync, dabei wird das DTMG-Signal (Data Enable) und die Pixel Clock als Referenz genutzt (vgl. Abb. 3.2).

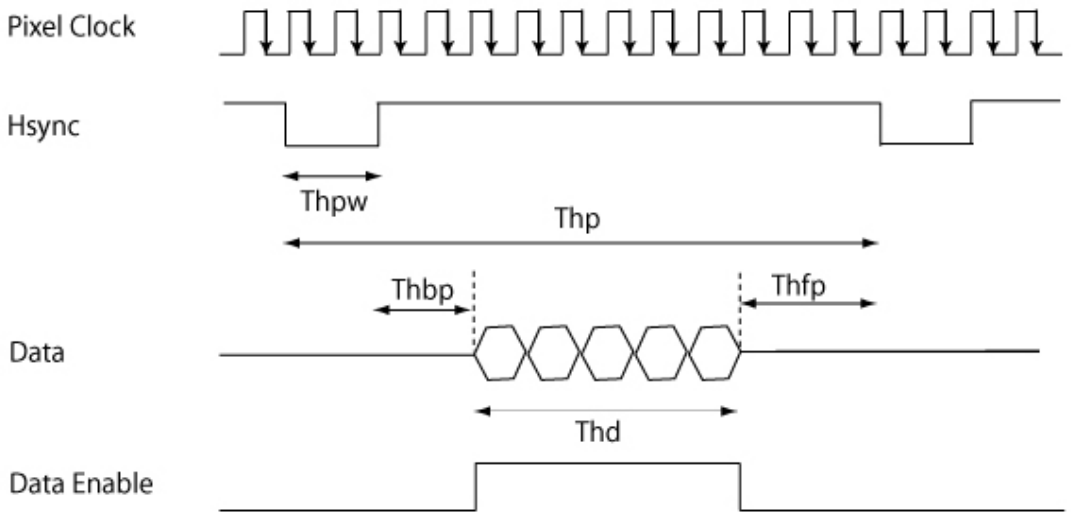

Abbildung 3.2.: Der LCD Timingchart zeigt die Datenberechtigung über das Data Enable. Über die Pixel Clock und dem Berechtigungssignal werden die VGA-Standardsignale HSync, mit dem Front-Porch, Back-Porch und dem Pulse-Width Anteil sowie auch das VSync generiert.[GEMCompiler]

Das Display (vgl. Abschnitt 2.3) wird mit den 6-Bit RGB Daten, dem Datenempfangs-Signal (DTMG), sowie mit dem Pixelclock zur Anzeige angestoßen. Damit eine Anzeige erfolgt, muss die Pixelclock im Rahmen des für das Timing-Controller zugelassenem Clocksignals übergeben werden (vgl. Abschnitt 2.3.2).

Die Bildauflösung der Bildverarbeitungspipeline beträgt 640(H) x 240(V), für den LCD-Touch-Panel wird die Auflösung auf 320(H) x 240(V) reduziert, indem nur jeder zweite Pixel in der horizontalen Achse abgetastet wird, dies führt zu einer Halbierung der Taktfrequenz von 13.5 MHz auf 6.75 MHz. Da nur der Graustufenwert Y für die Datenübertragung genutzt wird, werden für Rot, Grün und Blau die obersten 6-Bit von Y übergeben. Wenn für Rot, Grün und Blau die selben Bitkombinationen verwendet werden, so wird dadurch der Helligkeits-Wert des Bildes angezeigt.

3.4. Erweiterungsadapter für die Bildverarbeitungsplattform

Da der Ressourcenbedarf der Bildverarbeitungsplattform (vgl. Abschnitt 3.1 Abb. 3.1) zu 98% verbraucht ist (vgl. Anhang B.6), wird für die Implementierung weiterer Funktionen der TE0320-Board (vgl. Abschnitt 2.4 Abb. 2.7) mit dem derzeitig verwendeten Nexys2 (vgl. Abschnitt 2.2 Abb. 2.5) durch einen Erweiterungsadapter gekoppelt. Angesichts der Ressourcenerweiterung ist ein Ressource-Sharing der Bildverarbeitungspipeline zwischen den beiden FPGA-Boards sichergestellt. Der Erweiterungsadapter bietet zusätzlich zu den Schnittstellen des Nexys2 Stiftleisten-Verbindungen an, um weitere Sensoren anzuschließen. Die Datenübertragung der CCD-Kamera (vgl. Abschnitt 2.1) an die

Bildverarbeitungsplattform erfolgt über zwei Adapterplatinen die mit einer Flachband-Kabelverbindung miteinander verbunden sind und die an den jeweiligen Komponenten *(CCD-Kamera, Nexys2)* gesteckt sind.

Der LCD-Touch-Panel ist über einen weiteren Erweiterungsadapter an die FX2-Schnittstelle des Nexys2 gekoppelt und wird für die Anzeige der Bildverarbeitungspipeline aufgebaut. Die FX2-Schnittstelle wird für zwei Kopplungsvarianten genutzt, einmal für die Anzeige der Bildverarbeitungsplattform mit dem LCD-Display und die Kopplung des zweiten FPGA-Boards für die Ressourcenerweiterung.

3.4.1. Entwicklung der Adapterplatinen mit dem Altium Designer

Das Leiterplattenlayoutsystem Altium Designer bietet folgende EDA-Funktionen an:

- Entwurf eines Stromlaufplans
- Layoutentwurf (Leiterplatteentwurf)
- Entwurf programmierbarer Logik mittels einer Hardwarebeschreibungssprache bzw. Hochsprache
- Entwurf von Symbolen und Footprints

Das Programmfenster unterteilt sich in ein Entwurfsbereich, eine Navigationsleiste und eine Werkzeugleiste. Über die Navigationsleiste können Dateien bzw. Objekte ausgewählt werden. Die Karteikarten bieten die Unterteilung verschiedener Ansichten wie z.B. Files, Projects und Libraries an. Die Werkzeugleiste hat ein Schematic-Modus für das Zeichnen von Stromlaufplänen und ein PCB-Modus um ein Layoutentwurf für elektronische Leiterplatten zu entwickeln. Das Programm verwendet folgende Dokumenttypen:

- Dateiname.PrjPCB – Projektdatei für das gesamte Leiterplattenprojekt
- Dateiname.SchDoc – Stromlaufplandatei (Schematic)
- Dateiname.PcbDoc – Leiterplattenlayoutdatei (PCB)
- Dateiname.SchLib – Bibliotheksdatei für die Schaltsymbole im Stromlaufplan
- Dateiname.PcbLib – Bibliotheksdatei für das Layout der Footprints

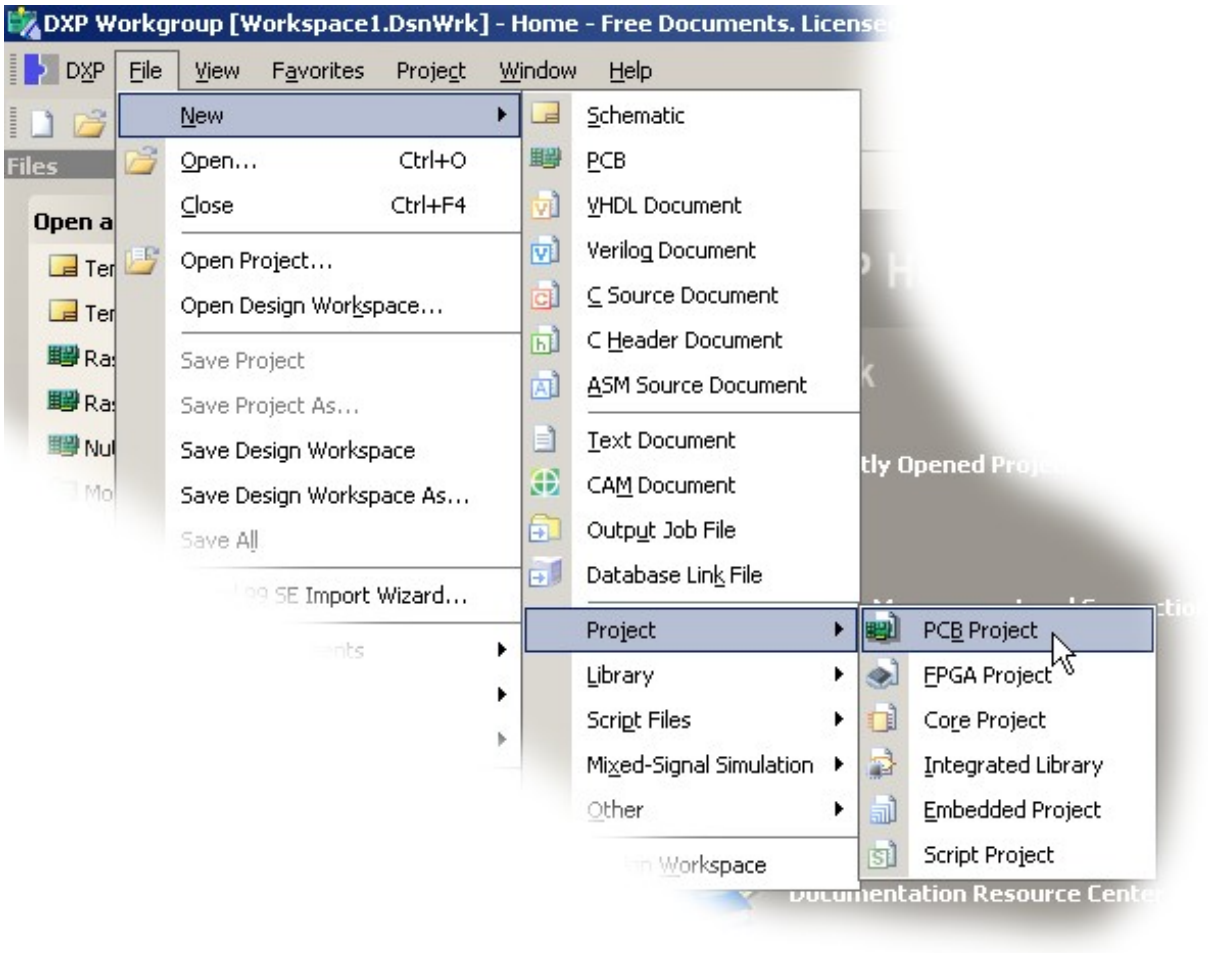

Abbildung 3.3.: Erstellung eines Projektes: $File \rightarrow New \rightarrow Project \rightarrow PCB\ \ Project$

Innerhalb des Projektes muss ein neues Schaltplan angelegt werden, da der PCB nur über eine Referenz auf das Schaltplan entwickelt wird.

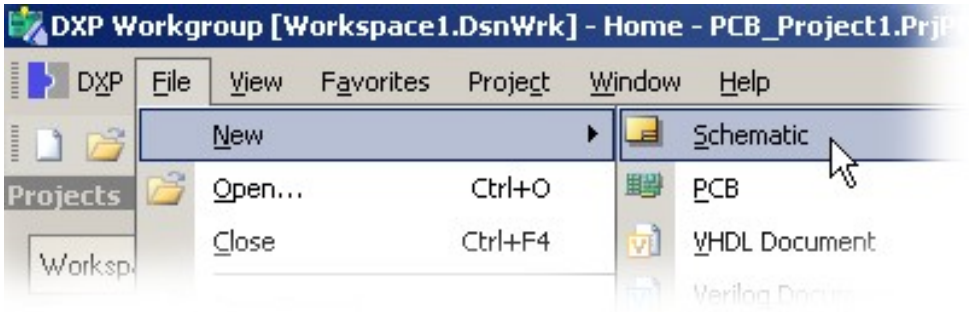

Abbildung 3.4.: Erstellung eines Stromlaufplans zur Generierung eines PCB-Layouts: $File \rightarrow New \rightarrow Schematic$

Auf der rechten Seite des Entwurfbereiches ist ein Karteikarten-Raster aufgeführt. Um Komponenten in das Stromlaufplan zu integrieren, gibt es eine Karteikarte die als Library bezeichnet ist. Dort kann man Anwender-Librarys anbinden bzw. vorhandene Librarys durchsuchen um die benötigte Komponente auszuwählen.

Für die Erstellung einer Anwender Schematic-Library wird eine neue .SchLib Datei erstellt, die über den Schematic-Modus der Werkzeugleiste bearbeitet wird. Damit die eigene Komponente auch im PCB-Layout erscheint, ist ein Footprint über .PCBLib für die Komponente zu erstellen und einzubinden. Dadurch wird eine Referenz auf die Komponente abgebildet und die Kopplung zwischen PCB-Design und Stromlaufplan hergestellt.

Nachdem alle Bauelemente platziert sind, werden die Komponenten miteinander verdrahtet und die Schaltsymbole für Betriebsspannung und Masse hinzugefügt. Der Symbolname bestimmt die Zugehörigkeit zu einem Netz z.B. GND, VCC oder +5V. Die logischen Informationen des Schaltplans werden zu den physikalischen Abmessungen in Relation gesetzt, um das PCB-Layout zu erstellen. Über die logische Struktur des Schaltplans wird eine Netzliste generiert und zu jedem Bauelement ein Footprint zugeordnet.

Komponenten des PCB-Layouts, anhand des LCD-FX2-Adapters (vgl. Abb. 3.5):

Top/Bottom-Layer: Die Ober-/Unterseite der Leiterplatte, die für den Anschluss der Bauelemente genutzt wird.

Top/Bottom-Layer Pad: Anschlussfläche für SMD-Bauteile, die nur auf einer Seite der Leiterplatte angebracht wird.

Multi-Layer Pad: Eine Durchbohrung der Leiterplatte, die mit einem Top- und Bottom-Layer Leiterbahn verbunden werden kann und somit an beiden Seiten verfügbar ist. Wird bei Stiftleisten verwendet.

Top/Bottom-Layer Leiterbahn: Die Verbindungsleitungen zwischen den Anschlussflächen, die über die Ober-/Unterseite der Leiterplatte verlaufen. Es ist auch möglich über weitere hinzugefügte Layerpaare Leiterbahnen zu erstellen, die dann zwischen dem Top-/Bottom Layer liegen.

Via (Layerwechsel): Wird für den Layerwechsel genutzt, falls keine Möglichkeit besteht über die genutzte Seite eine Verbindung mit der Anschlussfläche aufzubauen oder für eine Verkürzung des Laufzeitpfades.

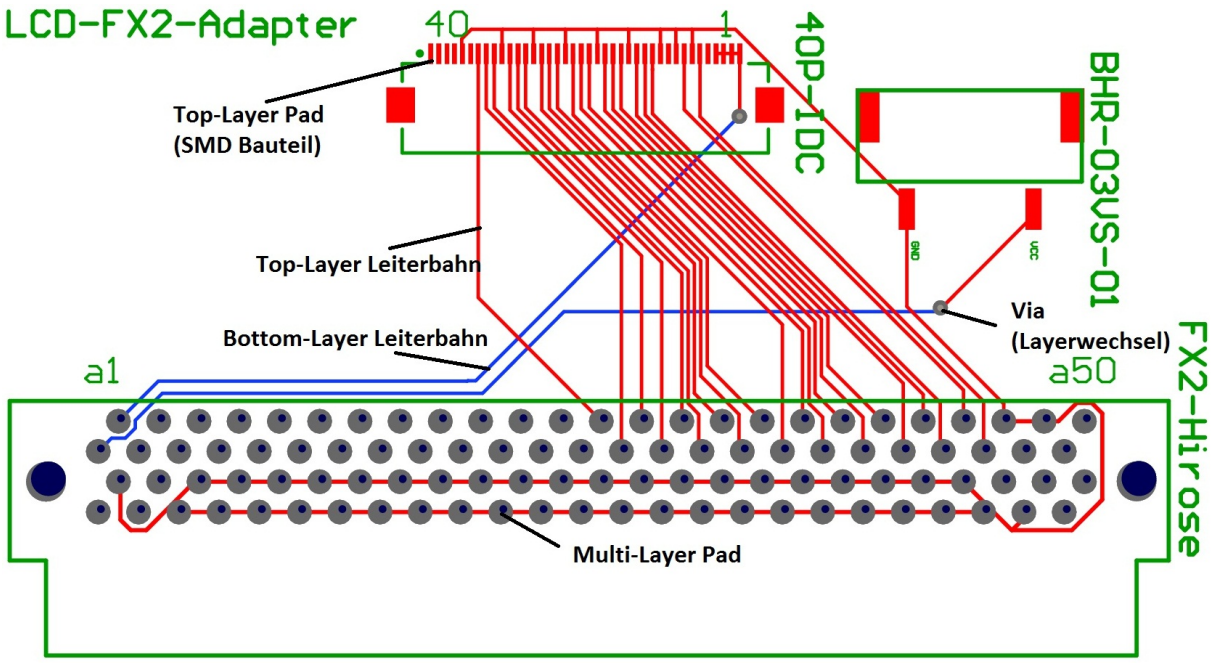

Abbildung 3.5.: PCB-Layout für die Adapterplatine zur Kopplung des LCD-Touch-Panels mit dem Nexys2-Board

3.4.2. Massekonzept und Signallaufzeitpfade der Adapterplatinen

Die Bildverarbeitungsplattform hat mehrere Komponenten die miteinander gekoppelt sind, damit keine Störfelder und somit resultierende Störspannungen während der Signalübertragung zwischen den Komponenten auftauchen, wird ein Massekonzept eingehalten. Der Single-Point-Ground hat eine sternförmige Topologie, der als Massepunkt für alle Komponenten referenziert wird und keine Störfelder erzeugt. Ein einfacheres Konzept ist der Bus-Ground, der den zentralen Massepunkt an die Komponenten weiterleitet, um Rückkopplungen zu vermeiden und den selben Effekt hat wie der Single-Point-Ground (vgl Abb. 3.6).

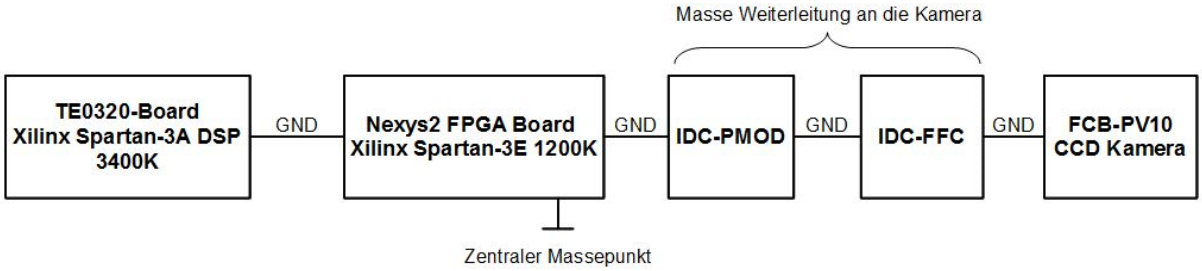

Abbildung 3.6.: Der zentrale Massepunkt wird von dem Nexys2-Board an die Komponenten weitergeleitet um Rückkopplungen der Masse zu vermeiden.

Durch die verschiedenen Signallaufzeitpfade zwischen den Komponenten entstehen Latenzzeiten. Die elektromagnetischen Wellen in einem Printed Circuit Board breiten sich mit einer Geschwindigkeit von $200000\frac{km}{s}$ aus. Nimmt man den Kehrwert und teilt durch 1000 kommt man auf $5\frac{ns}{m}$. Die Signallaufzeit von 1 cm beträgt daher $50ps$, bei einer maximalen Länge einer Leiterbahn von 10 cm wäre das eine Verzögerungszeit von $0.5ns$. Die Taktfrequenz des Nexys2-Boards beträgt 50 MHz und hat daher eine Periodendauer von $20ns$. Die Verzögerungszeit ist 40 mal geringer als die Periodendauer, daher ist kein Synchronisationsproblem während der Datenübertragung vorhanden.

3.5. Simulation, RTL-Codegenerierung und SoC-Konfiguration

Mit der Spezifikation des Nexys2-Evaluationsboards (vgl. Abschnitt 2.2) wird über das Base System Builder Wizard des Xilinx Platform Studios (XPS) das Microcontrollersystem [Xil10a] konfiguriert. Die Bildverarbeitungspipeline wurde als IP-Core über den Create and Import Peripheral Wizard an den PLB-Bus des MicroBlaze-Systems angebunden (vgl. Abb. 3.7).

Die Bildverarbeitung, die Fahrspurerkennung und -führung werden als Beschleunigermodule in die Netzliste der BV-Pipeline integriert und als Komponenten der Bildverarbeitungskette genutzt.

Die Beschleunigermodule sind zusammengesetzt aus den Komponenten: Bildverarbeitung, projektive-Transformation, Hough-Transformation, Zeilenzwischenspeicher-RAM und das Fahrspurführungssystem, die mit dem System-Generator [Xil10c] [Xil10d] entwickelt wurden.

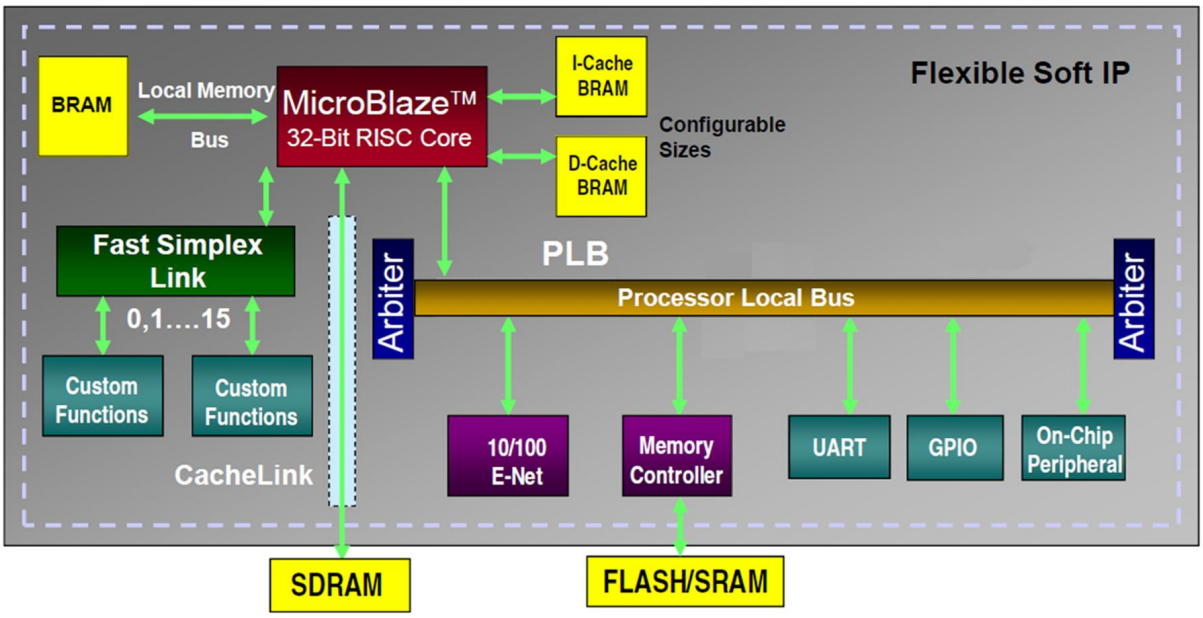

Abbildung 3.7.: Konfiguration eines MicroBlaze-basierten Mikrocontroller-Systems. [Xil10a]

Die modellbasierte RTL-Codegenerierung wird für die Entwicklung der Beschleunigermodule mithilfe der in den Xilinx Blocksets vorhandenen Arithmetik-Blöcken über eine grafische Oberfläche miteinander geschaltet und die Funktionalität des Moduls durch eine Simulink-Simulation getestet.

Der daraus entstandene RTL-Verhaltensmodell kann als Komponente in die *User Functionality* der Bildverarbeitungspipeline angebunden werden bzw. als Netzliste oder als zielplattformspezifisches Xilinx IP-Core erstellt werden (vgl. Abb. 3.8).

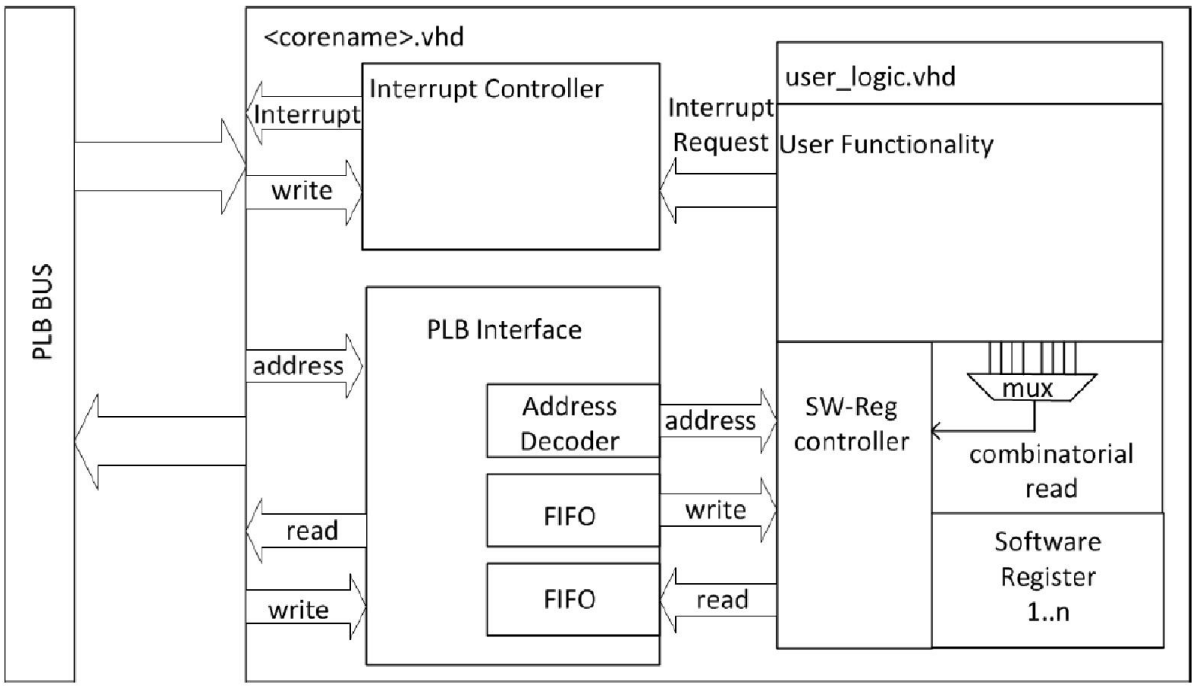

Abbildung 3.8.: Beschleunigermodul mit Softwareregistern zum getakteten Schreiben und kombinatorischen Lesen [Hor11].

Der von Xilinx zur Verfügung gestellte System Generator wird an die Matlab/Simulink Anwendung angebunden, um Funktionsblöcke der Xilinx Blockset-, Xilinx Reference

Blockset- und Xilinx XtremeDSP Kit-Library zu nutzen. Für die Bildverarbeitungspipeline wird nur die Blockset Library verwendet, die Signalverarbeitungsblöcke (z.B FIR, FFT), Speicherblöcke (z.B FIFO, ROM, RAM), Arithmetik-Blöcke (z.B. Mult., AddSub, Div.) und Logikelemente (z.B. Logical, Relational) zur Verfügung stellt.

Die Funktionsblöcke des System-Generators verwenden nur Daten die im Q-Format festgelegt sind, daher werden zur Typkonversion die Gateway In/Out-Blöcke genutzt.

Mit Up- und Downsample-Blöcken lassen sich Systeme mit verschiedenen Taktfrequenzen modellieren, um z.B. vorhandene Ressourcen durch Mehrfachverwendung höher auszulasten.

Eine Verifikation erfolgt mit den "From Workspace" und "To Workspace" Simulinkkomponenten, dabei werden zu den Simulationen Tabellen erstellt (vgl. Gleichung 3.2), die in einer Spalte die Zeit t und eine Spalte mit Eingangsdaten *(z.B. Bildkoordinaten, Pixeldaten oder Synchronisationszähler)* an die In-Blöcke des Xilinx-Blocksets übergeben werden und an den Out-Blöcken die Ausgangsdaten als Structure bzw. Array an eine Matlabvariable zugewiesen (vgl. Abb. 3.9).

Die Tabellen der Eingangswerte werden über ein Matlab-Skript erstellt, die zur Initialisierung der Funktionsblöcke und der Erstellung der Eingangsdaten für die Simulation verwendet wird, während ein weiteres Skript zur Auswertung der Ausgangsdaten führt.

$$Eingangswerte = \begin{array}{|c|c|}\hline t & x \\ \hline 1 & x_1 \\ 2 & x_2 \\ \vdots & \vdots \\ n & x_n \\ \hline \end{array} \qquad Ausgangswerte = \begin{array}{|c|}\hline y \\ \hline y_1 \\ y_2 \\ \vdots \\ y_n \\ \hline \end{array} \tag{3.2}$$

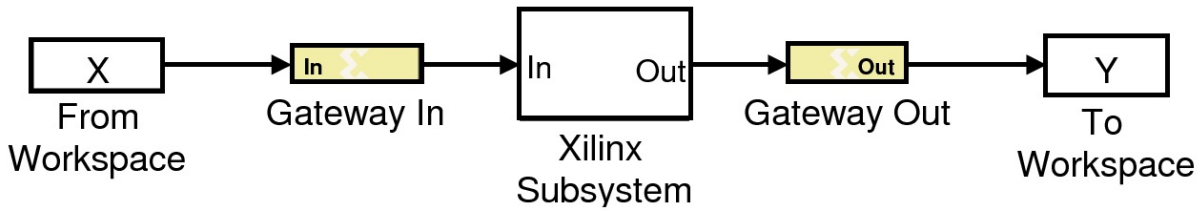

Abbildung 3.9.: Gateway In/Out Blöcke zur Typkonversion der Eingangs- und Ausgangswerte der Verifikationstabellen, um die Funktion der Xilinx Subsysteme zu testen.

4. Modifizierung und Erweiterung der Bildverarbeitungspipeline

Die Analyse der vorentwickelten BV-Kette (vgl. Abschnitt 2.6 und 3.1) führt zu einer Modifikation und Erweiterung der Bildverarbeitungspipeline. Die Sobel-Kantenfilterung mit fester Binarisierungsschwelle wurde ersetzt durch eine Schwellwertanpassung mit Graubildbinarisierung *(Adaptive-Binarize)* in Kombination mit einer morphologischen Erosion *(Erosion Filter)*.

Zur Verbesserung der Fahrspurerkennung wurden die Submodule *'Adaptive-Binarize'* und *'Erosion Filter'* mit den vorhandenen Komponenten *'Coordinate Generator'* und *'Projective Transform'* im Modul *'Image Processing'* (vgl. Abschnitt 3.1 Abb. 3.1) gekoppelt.

Die Schwellwertanpassung mit Graubildbinarisierung dehnt den Grauwertebereich des Pixeldatenstroms auf den gesamten 8-Bit-Bereich auf und passt sich damit den Lichtverhältnissen auf der Fahrbahn an.

Die morphologische Erosion führt zu einer Kontraktion der breiten Pixelbereiche, indem nur das gesetzte Pixel im Eingangsbild überprüft und nach einem Kriterium auf den niedrigsten Rang (0) gesetzt wird.

Die 'Coordinate Generator' Komponente generiert aus dem Line-Valid und Frame-Valid die Synchronisationszähler X und Y, die zur Pixelkoordinatenbestimmung genutzt werden. Die Projektive Transformation korrigiert die verzerrten Pixelkoordinaten und leitet diese an die Region of Interest und an den Pixel-RAM weiter (vgl. Abb. 4.1).

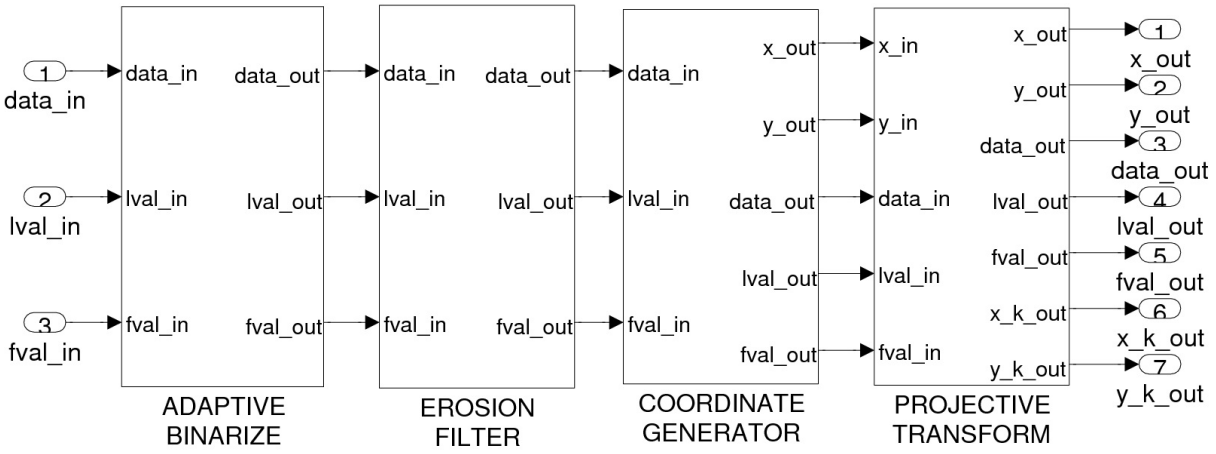

Abbildung 4.1.: Modifizierte Bildverarbeitungskette: Die Binarisierungsschwelle wird pro Bild anhand des bisher höchsten Grauwertes bestimmt. Anschließend wird ein Erosionsfilter verwendet um Bildrauschen zu unterdrücken und Konturen zu verkleinern. Zur Entzerrung des Kameradatenstroms werden mit Hilfe der Synchronisierungssignale Koordinaten generiert, die aus der Kameraebene in die Fahrzeugebene projiziert werden.

4.1. Schwellwertanpassung zur Graubildbinarisierung

Die Grauwertskalierung des Pixeldatenstroms führt zu der Bestimmung der Grauwertschwelle durch homogene Punktoperationen, die unabhängig von den Nachbarpixeln eine Veränderung der Pixel durch das bestimmte Schwellwert durchführt.

Die lineare Skalierung streckt gleichmäßig den Grauwerteumpfang des Bildes, wodurch der Teilbereich $[g_1, g_2]$ mit $0 \leq g_1 \leq g_2 \leq 255$ auf die gesamte Grauwertskala $[0, 255]$ projiziert wird. Die Dehnung des Grauwertebereichs erfolgt durch die Skalierungsfunktion f(g) (vgl. Gl. 4.1 und Abb. 4.2), wobei für den minimalen Grauwert $g_1 = g_{min}$ und den maximal auftretenden Grauwert $g_2 = g_{max}$ gilt:

$$f(g) = \frac{(g - g_1)}{g_2 - g_1} * 255 \tag{4.1}$$

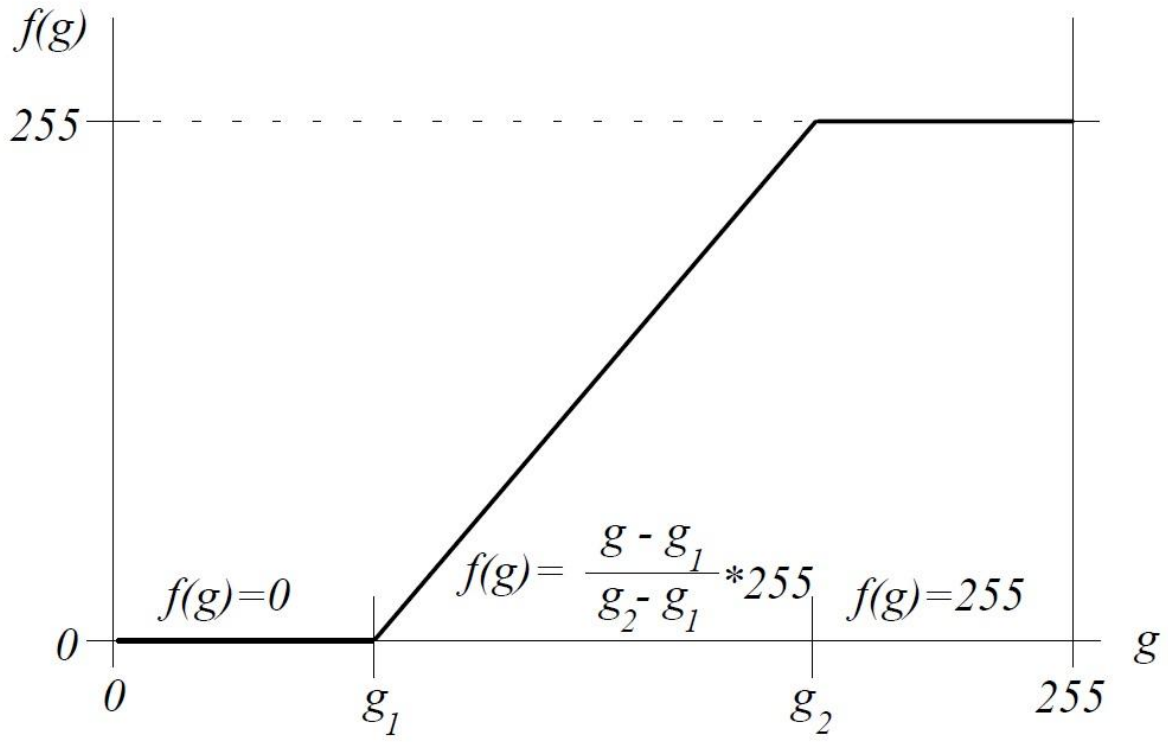

Abbildung 4.2.: Graph der linearen Skalierungsfunktion f(g).

Der minimale und maximale Grauwert wird pro Bild neu bestimmt, durch die gleichmäßige Struktur des Bildstroms sind keinerlei Nachteile bei der Erprobung der Filterfunktion vorgekommen. Durch die Skalierungsfunktion werden die Fahrbahnmarkierungen hervorgehoben und der Grauwertunterschied zwischen Fahrbahn und Fahrspur vergrößert (vgl. Abb 4.3).

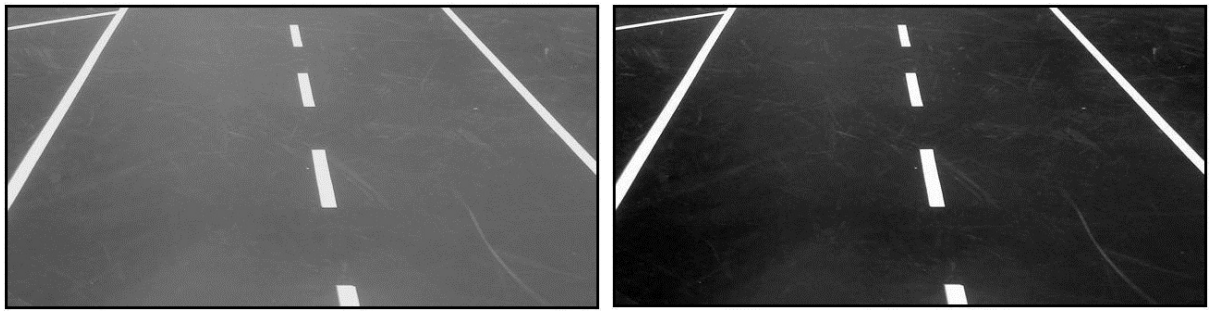

Abbildung 4.3.: Unverändertes (links) und linear skaliertes Graubild der Fahrbahn (rechts) bei höherer Lichtintensität. Die starke Lichtquelle führt zu Reflektionen auf der Fahrbahn und geringeren Grauwertunterschieden zwischen Fahrbahn und Fahrspurmarkierung.

Im Histogramm des unveränderten Bildes liegen die Grauwerte im mittleren Bereich der

Grauwertskala und sind kaum unterscheidbar, während bei der dynamischen Schwellwertgenerierung eine Ausdehnung der Grauwerte erfolgt und deutlicher hervorgehoben sind (vgl. Abb. 4.4).

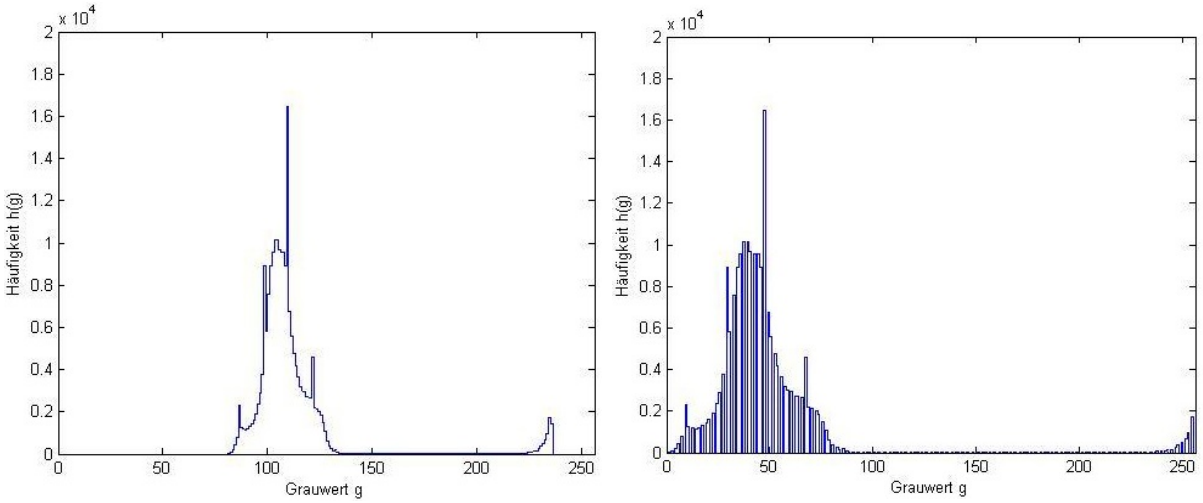

Abbildung 4.4.: Die markanten Grauwerthäufungen im Histogramm des unveränderten Bildes (links) bleiben im linear skalierten Histogramm (rechts) erhalten, werden aber deutlicher hervorgehoben und voneinander besser unterscheidbar.

Die Datenreduktion nach der Skalierung erfolgt über eine Binarisierung, wobei anstelle einer neuen Grauwertzuweisung die Umstellung der Gleichung 4.1 über die Differenz $g = g_{max} - g_{min}$ mit einer prozentualen Grauwertschwelle $p = 0.875$ erfolgt.

Die Schwellwertanpassung mit Graubildbinarisierung wird mit einer Taktfrequenz $f_{13.5} = 13.5$ MHz betrieben, die durch den Digital-Clock-Manager zur Verfügung gestellt wird. Die empfangenen Synchronisationssignale lval und fval werden mit der Taktverzögerung des Moduls weitergeleitet, während fval zusätzlich zur Neugenerierung des Schwellwertes genutzt wird. Die 8-Bit Daten werden im Verlauf des Moduls für eine Datenreduktion binarisiert (vgl. Abb. 4.5).

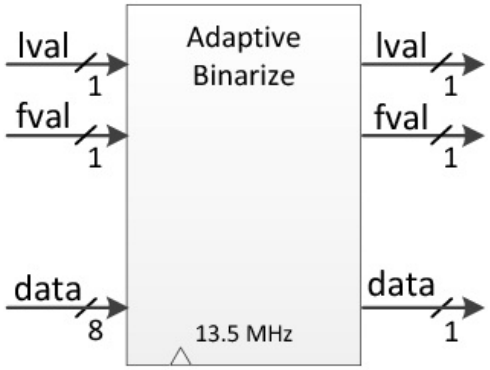

Abbildung 4.5.: Binarisierung mit adaptiver Grauwertschwelle wird mit einer Taktfrequenz von $f_{13.5}$ betrieben. Für eine eingehende Datenreduktion werden die Grauwerte binarisiert.

Die beiden Extremwerte werden permanent über die Relational-Blöcke $>$ und $<$ an ein Flipflop übergeben, während über die Rückkopplung der getakteten Werte der Vergleich zwischen gespeichertem Wert und den ankommenden Daten durchgeführt wird. Falls die neuen Werte größer bzw. kleiner sind als die gespeicherten Werte der jeweiligen Blockfunktion, so wird ein Enable-Signal übergeben, der den Eingangssignal speichert. Die Reinitialisierung der Extremwerte g_{max} und g_{min} erfolgt während der Bildpausen, d.h.

wenn fval eine LOW-Phase hat, dadurch wird im nächsten Bild eine Neuzuweisung der Grauwerte ausgeführt.

Für die Binarisierung wird die Differenz der Extremwerte *(delta_g)* über eine Subtraktion berechnet und durch 3-faches schieben nach rechts mit der Subtraktion der Differenz eine Multiplikation *(delta_g∗p)* mit der festgelegten Grauwertschwelle $p = 0.875$ vermieden. Der Vergleich zwischen dem dynamisch berechneten Schwellwert mit dem aktuellen Eingangswert *(data_in)* führt zu einer Binarisierung. Der verarbeitete Pixel *(data_out)* und die Synchronisationssignale *(lval, fval)* werden mit einer Verzögerung von 1 Takt am Ausgang weitergeleitet (vgl. Abb. 4.6).

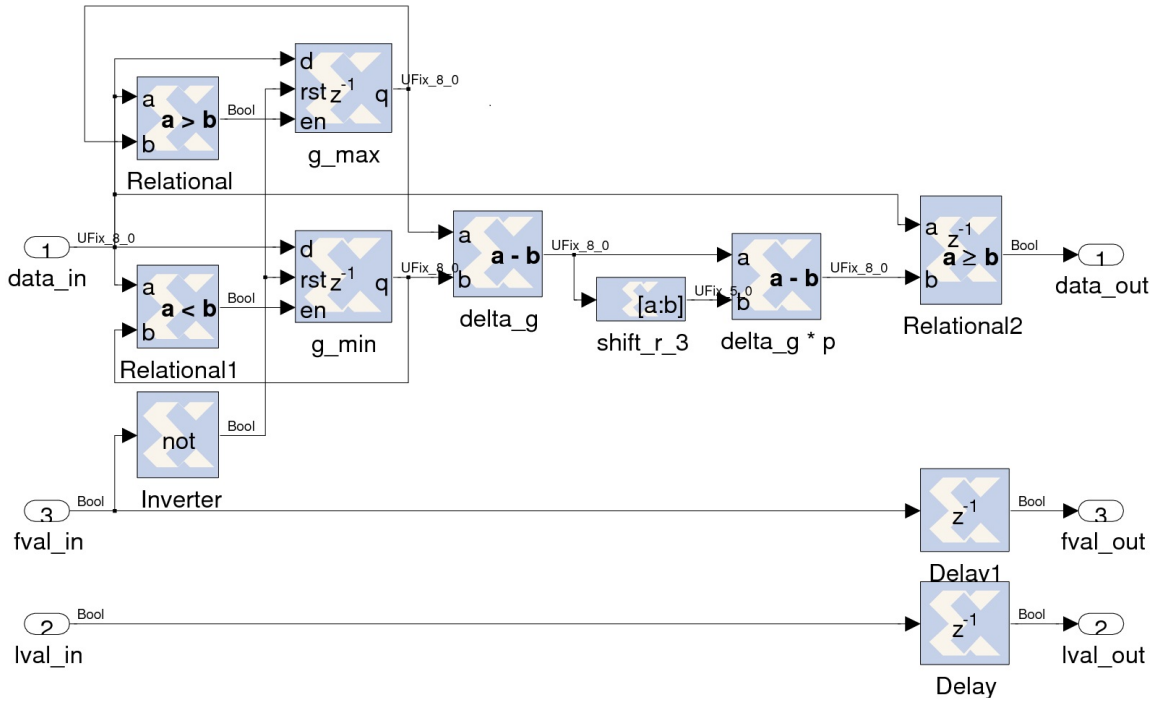

Abbildung 4.6.: Die ermittelten Extremwerte werden mit einer festen Grauwertschwelle verrechnet und mit dem aktuellen Eingangswert verglichen.

Die Binarisierung mit der dynamischen Grauwertschwelle führt zu einer Unempfindlichkeit gegenüber starken Reflektionen der Fahrbahn und die damit verbundene Lichtintensität in der Fahrbahnumgebung. In der Abbildung 4.7 hat die feste Grauwertschwelle *(links)* Störungen im Bild zu verzeichnen, da die Reflektion des Lichtes die Schwelle überschreitet. Während die adaptive Grauwertschwelle keine Störungen im Bild enthält und ein klarer Unterschied zwischen Fahrbahn und Fahrspurmarkierung besteht (vgl. Abb. 4.7).

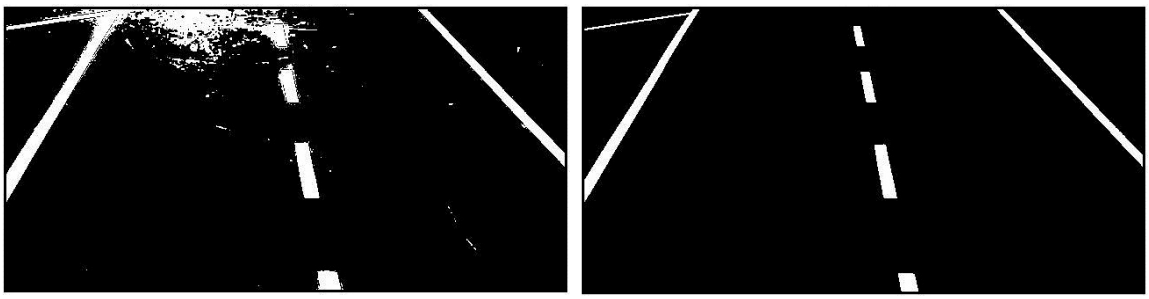

Abbildung 4.7.: Das mit einem festen Schwellwert verarbeitete Bild (links) hat durch die Reflektion des Lichtes im oberen Bildbereich Störungen (weiße Flächen), während im rechten Bild durch die Anpassung des Schwellwertes die Fahrspurmarkierung ohne Störungen hervorgehoben wird.

4.2. Morphologische Erosion zur Kontraktion der Fahrbahnmarkierung

Der Kontrast zwischen Fahrspurmarkierung und Fahrbahn ist deutlich hervorgehoben, reicht aber für eine eindeutige Fahrspurerkennung über die Hough-Transformation nicht aus, da die Geradenapproximation im Hough-Raum zu mehreren Maxima führen kann und zur Folge hat, dass verschiedene r und Φ an die Fahrspurführung übergeben werden (vgl. Abb. 4.8).

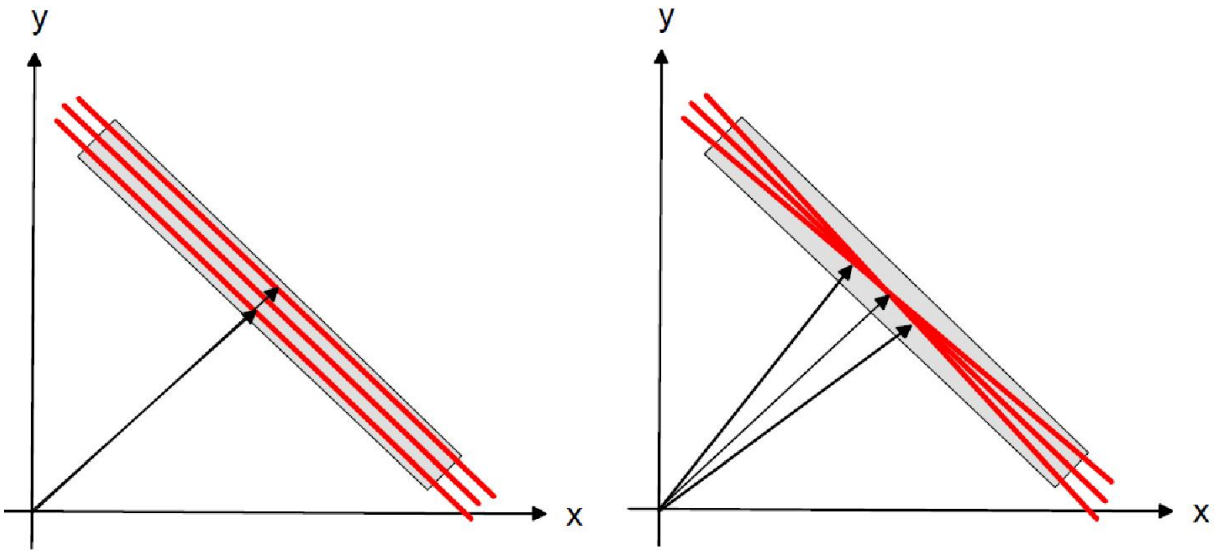

Abbildung 4.8.: Geraden mit unterschiedlichen Abständen r und Φ passen in in die Fahrspurmarkierung (grau).

Die morphologische Filterung nutzt die Umgebungsinformation des zentral zu überprüfenden Pixels, indem die Nachbarpixel überprüft werden und durch das Faltungskern eine Veränderung des Pixels erfolgt (vgl. Abb. 4.9).

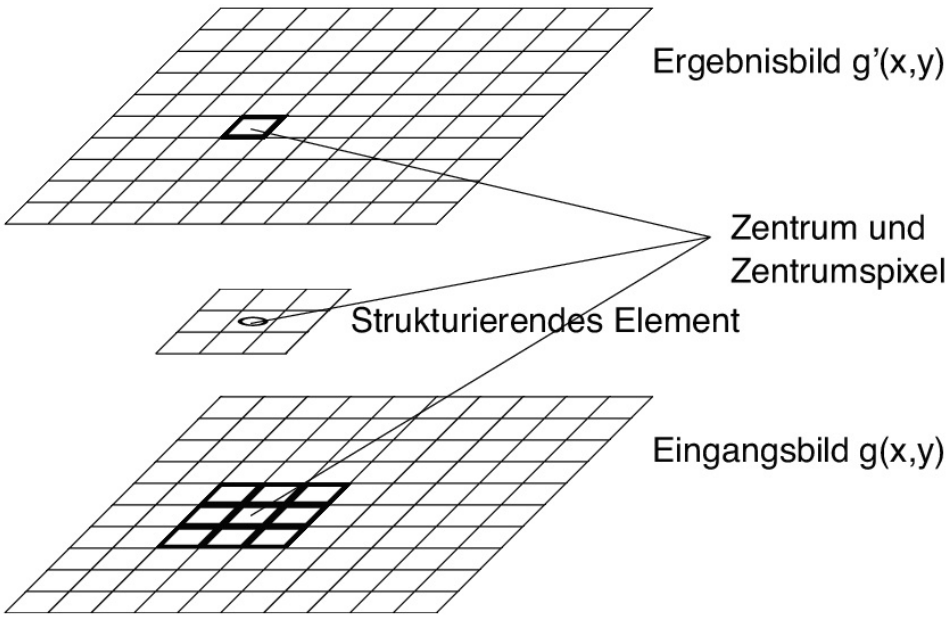

Abbildung 4.9.: Auf der Umgebung des Zentrumpixels im Quellbild *Eingangsbild g(x,y)* wird ein quadratischer Faltungskern (Strukturierendes Element) angewendet und das Ergebnis der Faltung an der Stelle *g'(x,y)* in das Zielbild eingetragen.[Erh08]

Folgende morphologische Operationen werden zur Veränderung der Form eines Objektes genutzt:

Erosions-Operator: Der Erosionsfilter führt eine Kontraktion des Objektes aus, indem die Randpixel entfernt werden, dadurch werden die Löcher innerhalb eines Objektes vergrößert bzw. das Objekt selbst verkleinert.

Sei p die Anzahl der im Eingangsbild g(x,y) gesetzten Pixel und k die Anzahl der Pixel in der Faltungsmatrix, dann gilt:

$$g'(x_0, y_0) = \begin{cases} 1, & \text{für p = k} \\ 0, & \text{für p < k} \end{cases} \tag{4.2}$$

Dilatations-Operator: Ein Rangordnungsfilter der im Binärbild zur Expansion vorhandener Objektstrukturen führt, indem Pixel hinzugefügt werden. Die Dilatation füllt zusätzlich Lücken auf und glättet Ränder von Objekten.

Sei p die Anzahl der im Eingangsbild g(x,y) gesetzten Pixel, dann gilt:

$$g'(x_0, y_0) = \begin{cases} 1, & \text{für p > 0} \\ g(x_0, y_0), & \text{sonst} \end{cases} \tag{4.3}$$

Ouverture-Operator: Der Opening-Operator glättet die Ränder von Objekten und entfernt Brücken zwischen mehreren Objekten, die durch eine Segmentierung übrig geblieben sind. Zusätzlich können gezielt Objekte aus einem Bild entfernt werden, indem die Faltungsmatrix die Form des Elementes enthält, die erhalten bleiben soll. Die Ausführung des Erosion-Operators wird mit anschließender Dilatation abgeschlossen, dabei wird immer die gleiche Anzahl der morphologischen Operationen verwendet.

Fermeture-Operator: Das Closing wird für das schließen kleiner Löcher innerhalb von Objekten genutzt, die während einer Segmentierung entstanden sind. Genauso wie beim Opening sind auch in der Fermeture Objekte entfernbar, indem die gewünschte Struktur als Faltungsmatrix genutzt wird.

Die morphologischen Filter sind in Grauwert- und Binärbildern anzuwenden, wobei die Operationen im Binärbild größere Auswirkungen haben. Beispielsweise wird bei der Objekterkennung eines Grauwertbildes erst eine Binarisierung durchgeführt, dann die morphologischen Operationen und das Ergebnis mit dem Grauwertbild über eine AND-Operation verknüpft.

Eine Effektive Ausführung der morphologischen Erosion ist die Betrachtung der gesetzten Pixel bzw. des Objektes im Eingangsbild, da nur noch das Kriterium besteht ob das Pixel erhalten bleibt oder nicht. Umgekehrt ist es für eine Dilatation effektiver wenn die nicht gesetzten Pixel betrachtet werden bzw. das Hintergrundbild des Eingangsbildes.

Der weitergeleitete Pixeldatenstrom des 'Adaptive-Binarize' wird binarisiert als Dateneingang an das Erosionfilter-Modul übergeben und nach der Berechnung mit der Faltungsmatrix ausgegeben. Die morphologische Filterung wird mit einer Taktfrequenz von $f_{13.5} = 13.5$ MHz betrieben. Der Synchronisationssignal lval wird als Enable-Bit für die Speicherblöcke genutzt um eine Faltungsmatrix aufzubauen und zusätzlich zu fval mit einer Taktverzögerung von 4 Takten an die nächste Komponente weitergeleitet (vgl. Abb. 4.10).

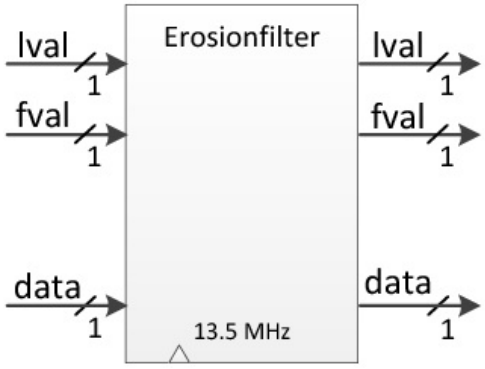

Abbildung 4.10.: Blackbox des Erosionsfilter-Moduls, das den eingehenden Binärpixeldatenstrom mit der Frequenz $f_{13.5}$ mit einer quadratischen 3x3 Faltungsmaske durch UND-Verknüpfung erodiert.

Um eine morphologische Erosion auf einem Videodatenstrom auszuführen, wird der Datenstrom deserialisiert und in eine 3x3-Faltungsmatrix umgeformt. Das aus dem 'Adaptive-Binarize' weitergeleitete 1-Bit-Pixel wird durch die Delay-Blöcke verzögert um das strukturierende Element (vgl. Abb. 4.9) aufzubauen und mit einer UND-Verknüpfung das Zentrum der 3x3-Matrix im Ergebnisbild $g'(x, y)$ einzutragen .

Der Logical-Block liefert nur dann einen gesetzten Pixel, wenn die 3x3-Matrix mit Einsen besetzt ist. Die Kontraktion der Objekte mit Rang 8 führt dazu, dass auf jeder Seite ein Pixel entfernt wird und zusätzlich die Ränder geglättet werden (vgl Abb. 4.11).

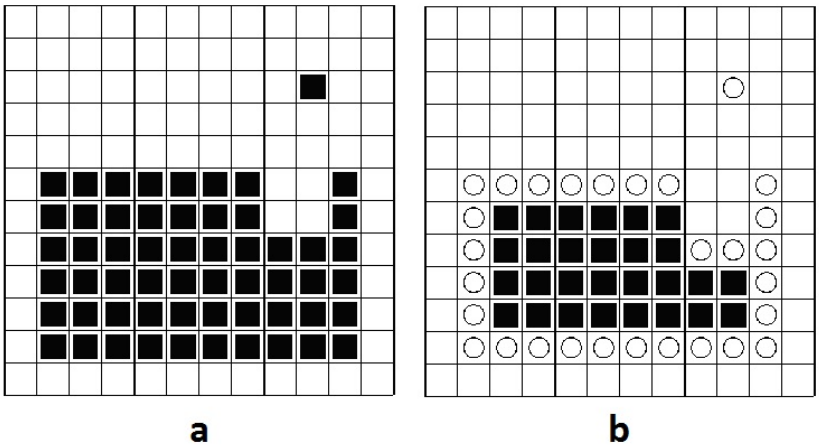

Abbildung 4.11.: Die Kreise markieren die Stellen der erodierten Pixel. a) Eingangsbild, b) Erosionsgrenze = 8, also eine UND-Verknüpfung der 3x3-Faltungsmatrix. [Erh08]

Dazu wird das Synchronisationssignal lval als Enable für die Speicherblöcke genutzt, um die Zeilenpausen zwischen der Datenübertragung zu unterscheiden. Für eine Matrixform werden zwei Delayblöcke mit einer Verzögerung der Zeilenlänge von 638 genutzt, damit der Pixel auf die richtige Position in der Faltungsmatrix überführt wird (vgl. Abb. 4.12).

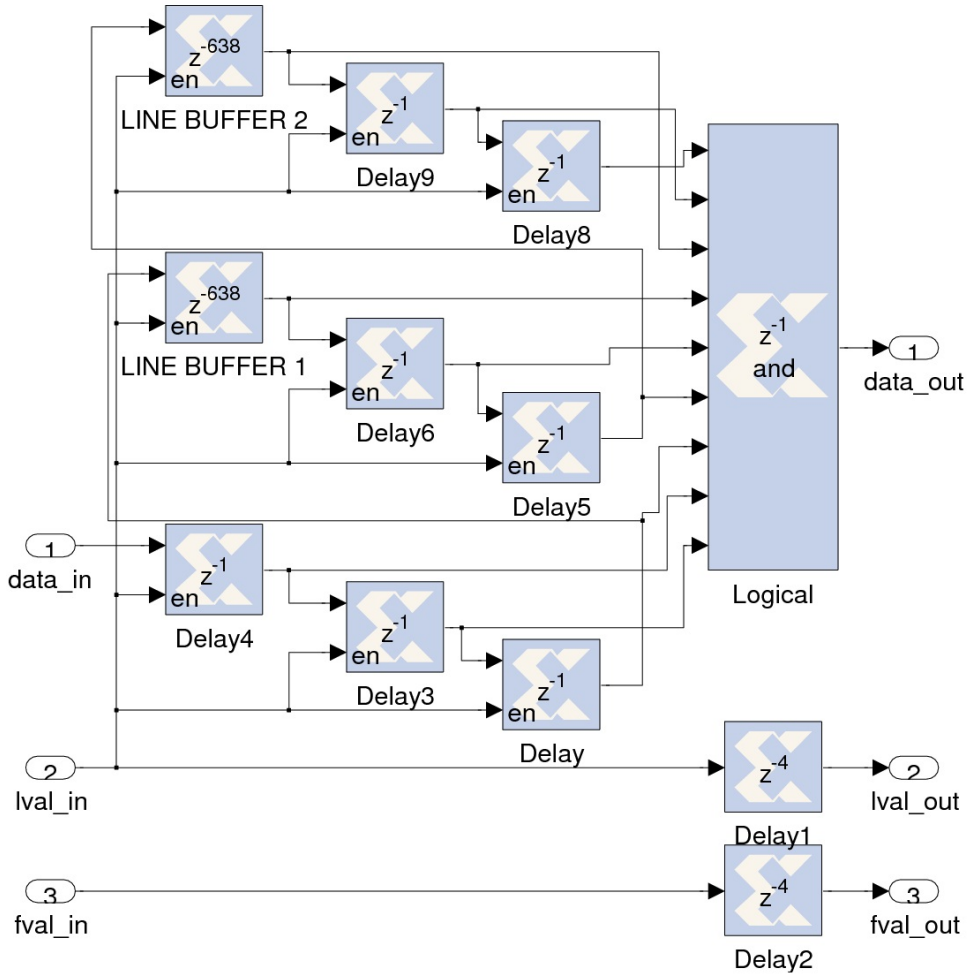

Abbildung 4.12.: Deserialisierung durch Linebuffer-Blöcke mit einem 3x3-Faltungskern durch UND-Gatter als Erosionfilter genutzt.

Die Fahrspurmarkierungen wurden zur Veranschaulichung im grauwertskalierten Bild durch eine repräsentative Anwendung einer quadratischen 5x5 Erosions-Faltungskern kontrahiert. Dadurch ist die Anzahl der Maximabildung in der Hough-Transformation kleiner und führt zu einem eindeutigen Ergebnis von r und Φ (vgl. Abb. 4.13).

In der Bildverarbeitungspipeline wird ein 3x3-Faltungskern verwendet, da ein größerer Faltungskern mehr Speicherblöcke verwendet und die Reduzierung der Seiten um 1 Pixel ausreicht, um ein eindeutiges Ergebnis der HT zu bekommen.

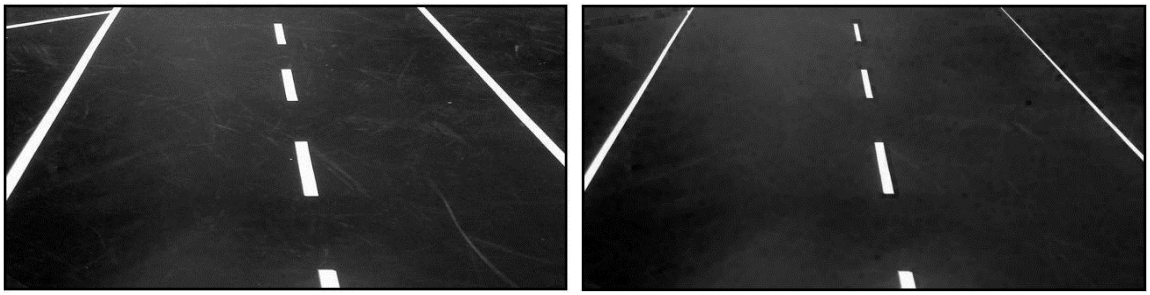

Abbildung 4.13.: Die Fahrspurmarkierungen im grauwertskalierten Bild (links) wurden durch Anwendung eines quadratischen 5x5 Erosions-Faltungskern verschmälert, sodass diese in der Hough-Transformation zu weniger mehrdeutigen Maxima führen.

4.3. Projektive Entzerrung der Kameraperspektive

Die Durchführung einer Fahrspurapproximation erfordert ein perspektivisch korrigiertes Bild, um eine Interpretation der ermittelten Fahrspur zu verwirklichen und eine Spurführung zu realisieren. Bei der perspektivischen Bildtransformation wird die Kamera virtuell in eine senkrechte Position über der Fahrbahn verschoben (vgl. Abb. 4.14). Bei dieser Verschiebung wird der untere Teil des Bildes gestaucht und der obere Teil gestreckt. Durch das Stauchen gehen Bildinformationen verloren, da einige Bildpunkte im Zielbild auf die selbe Stelle referenzieren. Im Gegensatz dazu werden in den zu streckenden Teilen des Bildes mehr Informationen benötigt als im Bild vorhanden sind. Bei der perspektivischen Transformation ist das Fokussieren eines bestimmten Bildbereiches bzw. das Verschieben des entzerrten Bildes möglich, um beispielsweise für eine Anzeige die wichtigsten Merkmale zu visualisieren.

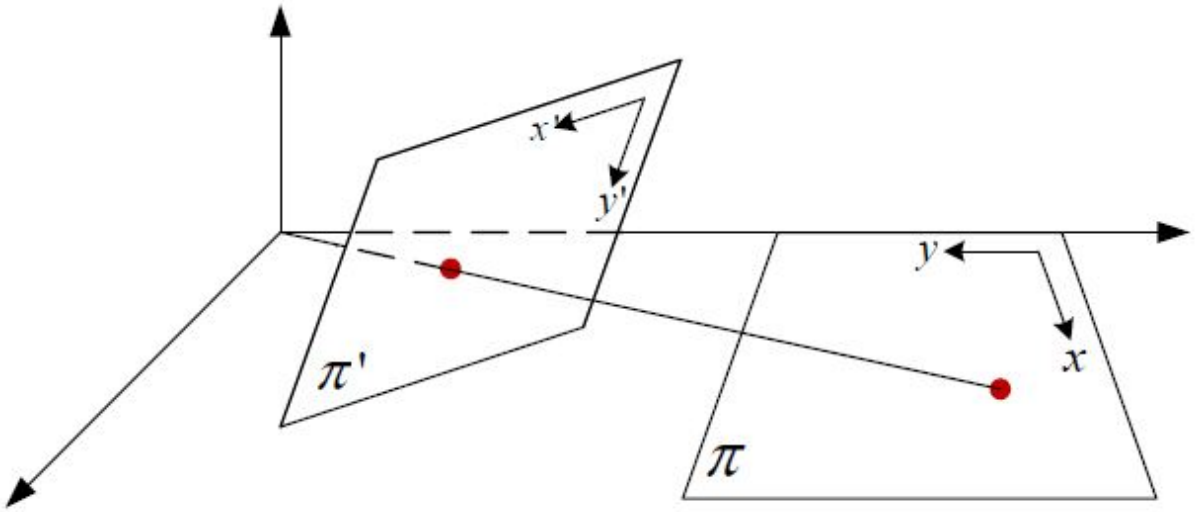

Abbildung 4.14.: Die Referenzierung des Bildpunktes π' von der Kamera-Ebene auf das Pixel π der Fahrbahnebene über die projektive Entzerrung der verzerrten Bildkoordinaten.

Ein lineares Gleichungssystem wird für das Verhältnis der beiden Ebenen über korrespondierender Punkte in Form einer Koeffizienten-Matrix aufgestellt (vgl. Gl. 4.4) [vgl. BB06, S. 367-369].

$$B = \begin{pmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \end{pmatrix} \tag{4.4}$$

Durch die Überführung eines Punktes $\vec{p_k}$ in kartesischen Koordinaten zu einem Punkt $\vec{p_h}$ in homogenen Koordinaten, indem man eine zusätzliche Dimension hinzugefügt, kann der homogene dreidimensionale Vektor $\vec{p_h}$ mit der Transformationsmatrix B (vgl. Gl. 4.4) multipliziert werden.

$$\vec{p_h} = B\vec{p'_h} = \begin{pmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \end{pmatrix} \cdot \begin{pmatrix} x'_h \\ y'_h \\ 1 \end{pmatrix} \tag{4.5}$$

Aus der Gleichung 4.5 folgt:

$$\begin{aligned} x_h &= b_{11}x'_h + b_{12}y'_h + b_{13} \\ y_h &= b_{21}x'_h + b_{22}y'_h + b_{23} \\ 1 &= b_{31}x'_h + b_{32}y'_h + b_{33} \end{aligned} \tag{4.6}$$

Die Rücktransformation eines Punktes in homogenen Koordinaten auf die kartesischen Koordinaten erfolgt über eine Division des Punktes mit dem Skalierungsfaktor $s = 1$ (vgl. Gl. 4.12).

$$\vec{p_h} = \begin{pmatrix} sx_h \\ sy_h \\ s \end{pmatrix} \rightarrow \vec{p_k} = \begin{pmatrix} x_k = \frac{x_h}{s} \\ y_k = \frac{y_h}{s} \end{pmatrix} \tag{4.7}$$

Die Kartesischen Koordinaten (x_k, y_k) ergeben sich mit der aus Gleichung 4.12 aufgestellten Rücktransformation und der Multiplikation aus der Transformationsmatrix und $\vec{p_h}$ (vgl. Gl. 4.5) .

$$x_k = \frac{x_h}{1} = \frac{b_{11}x'_h + b_{12}y'_h + b_{13}}{b_{31}x'_h + b_{32}y'_h + b_{33}} \tag{4.8}$$

$$y_k = \frac{y_h}{1} = \frac{b_{21}x'_h + b_{22}y'_h + b_{23}}{b_{31}x'_h + b_{32}y'_h + b_{33}} \tag{4.9}$$

Mit den Punktkorrespondenzen wird ein lineares Gleichungssystem aufgestellt, deren Grundlage die Gleichungen 4.8 und 4.9 darstellen, die schrittweise umgestellt werden, um alle b_{ij} auf eine Seite zu bringen.

$$\begin{aligned} & x_k = \frac{x_h}{1} = \frac{b_{11}x'_h + b_{12}y'_h + b_{13}}{b_{31}x'_h + b_{32}y'_h + b_{33}} \\ \Leftrightarrow \quad & x_k(b_{31}x'_h + b_{32}y'_h + b_{33}) = b_{11}x'_h + b_{12}y'_h + b_{13} \\ \Leftrightarrow \quad & x_k = b_{11}x'_h + b_{12}y'_h + b_{13} - b_{31}x'_h x_k - b_{32}y'_h x_k - b_{33}x_k \end{aligned} \tag{4.10}$$

$$\begin{aligned} & y_k = \frac{y_h}{1} = \frac{b_{21}x'_h + b_{22}y'_h + b_{23}}{b_{31}x'_h + b_{32}y'_h + b_{33}} \\ \Leftrightarrow \quad & y_k(b_{31}x'_h + b_{32}y'_h + b_{33}) = b_{21}x'_h + b_{22}y'_h + b_{23} \\ \Leftrightarrow \quad & y_k = b_{21}x'_h + b_{22}y'_h + b_{23} - b_{31}x'_h y_k - b_{32}y'_h y_k - b_{33}y_k \end{aligned} \tag{4.11}$$

Aus den Gleichungen 4.10 und 4.11 wird ein lineares Gleichungssystem in einer Matrixdarstellung erstellt, wobei für jedes korrespondierende Punktpaar eine Gleichung sowohl für die x als auch für die y-Komponente entsteht. Um ein Gleichungssystem mit acht

Unbekannten b_{ij} zu lösen, werden mindestens acht Gleichungen benötigt, also mehr als 4 korrespondierende Punktpaare [vgl. BB06, S. 367-369].

$$\begin{pmatrix} x'_{k_1} & y'_{k_1} & 1 & 0 & 0 & 0 & -x'_{k_1}x_{k_1} & -y'_{k_1}x_{k_1} \\ 0 & 0 & 0 & x'_{k_1} & y'_{k_1} & 1 & -x'_{k_1}y_{k_1} & -y'_{k_1}y_{k_1} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x'_{k_n} & y'_{k_n} & 1 & 0 & 0 & 0 & -x'_{k_n}x_{k_n} & -y'_{k_n}x_{k_n} \\ 0 & 0 & 0 & x'_{k_n} & y'_{k_n} & 1 & -x'_{k_n}y_{k_n} & -y'_{k_n}y_{k_n} \end{pmatrix} \cdot \begin{pmatrix} b_{11} \\ b_{12} \\ b_{13} \\ b_{21} \\ b_{22} \\ b_{23} \\ b_{31} \\ b_{32} \\ b_{33} \end{pmatrix} = \begin{pmatrix} x_{k_1} \\ y_{k_1} \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ x_{k_n} \\ y_{k_n} \end{pmatrix} \tag{4.12}$$

Durch die Verwendung von mehr als vier Punktkorrespondenzen wird der Fehler gering gehalten, der durch Kollinearität der Punkte und Ungenauigkeiten bei der Bestimmung von korrespondierenden Koordinaten entsteht. Durch Einsetzen der Punktkorrespondenzen entsteht ein überbestimmtes Gleichungssystem, deren Lösung, unter Anwendung der linearen Ausgleichsrechnung, die Transformationsmatrix B ist und deren Elemente einen minimalen quadratischen Fehler bezüglich der gewählten Punktkorrespondenzen besitzen (vgl Anhang (pt_init.m,B.2)).

4.3.1. Kalibrierung der perspektivischen Transformation

Wird der Neigungswinkel der Kamera verändert, so führt die konstant gehaltene Transformationsmatrix zu fehlerhaften Transformationsergebnissen. Eine Neukalibrierung der Matrix B, die das neue Verhältnis der Bildebene zur Fahrzeugebene berücksichtigt, ermöglicht eine Korrektur der Ergebnisse.

In der Fahrzeugebene wird eine feste Punktanordnung zur Kalibrierung der perspektivischen Verzerrung bestimmt. Über eine festgelegte maximale Sichtweite des projektiv entzerrten Bildes wird die Pixeldifferenz zwischen gleich weit liegenden Punkten in der horizontalen und Vertikalen ermittelt und daraus ein Koordinatenfeld angeordnet.

Das Interlace-Verfahren erzeugt zwei Halbbilder, die auf der vertikalen Bildachse 240 Pixel lang sind (vgl. Abschnitt 2.1). Das Verhältnis zwischen metrischem Maß und Pixellänge entsteht dadurch, indem die festgelegte Sichtweite von 120 cm auf die 240 Pixel abgebildet wird (vgl. Gl. 4.13).

$$\frac{120cm}{240Pixel} = \frac{0.5cm}{1Pixel} \tag{4.13}$$

Durch das Verhältnis kann eine metrische Länge in Pixeln definiert werden. Für die Neukalibrierung der Transformationsmatrix B ist der Abstand von 35 Pixeln zwischen den festen Pixelanordnungen einzuhalten, da im metrischen Maß der Abstand 17.5 cm beträgt und durch die Umrechnung über die Gl. 4.13 der Abstand 35 Pixel ergibt.

4.4. Entwurf eines Zwischenspeicher-RAMs zur Wiederherstellung der Pixelreihenfolge

Für eine Visualisierung des projektiv-transformierten Bildes ist eine Rekonstruktion der Pixelanordnung erforderlich, da die PT die Pixelkoordinaten unkoordiniert weiterleitet (vgl. Abschnitt 4.3). Die Pixelneuordnung wird durch eine Zwischenspeicherung der Pixeldaten und synchrone Ausgabe zu einem Pixelkoordinatenzähler erreicht. Die Datenpixel werden passend zu den Synchronisationszählern der X- und Y-Achse aus dem Zwischenspeicher ausgelesen und für die Darstellung des entzerrten Bildes mit einer Verzögerung von 1 Bild/Sekunde ausgegeben.

4.4.1. Bestimmung des zu speichernden Bildbereiches

Der Bildbereich wird über die korrigierten Pixelkoordinaten der projektiven Transformation in X- und Y-Richtung ermittelt, indem der minimale und maximale Wert der berechneten Koordinaten als Grenzbereich der zu speichernden Pixeldaten gilt (vgl. Abb. 4.15. Dabei ist der reservierte Speicherbereich auf das projektiv entzerrte Bild ausgelegt, da kleinere Bildbereiche wie die Region of Interest auf den größer ausgelegten Zeilenzwischenspeicher-RAM passen (vgl. Gl. 4.14). Die maximale Pixelkoordinate des Grenzbereiches wird von der minimalen Koordinate abgezogen und somit auf den Ursprung verschoben um die Größe des Bildbereiches zu ermitteln:

$$\begin{aligned} \Delta x &= x_{max}2 - x_{min}2 \\ \Delta y &= y_{max}2 - y_{min}2 \\ Memory_{RAM} &= \Delta x \cdot \Delta y \end{aligned} \tag{4.14}$$

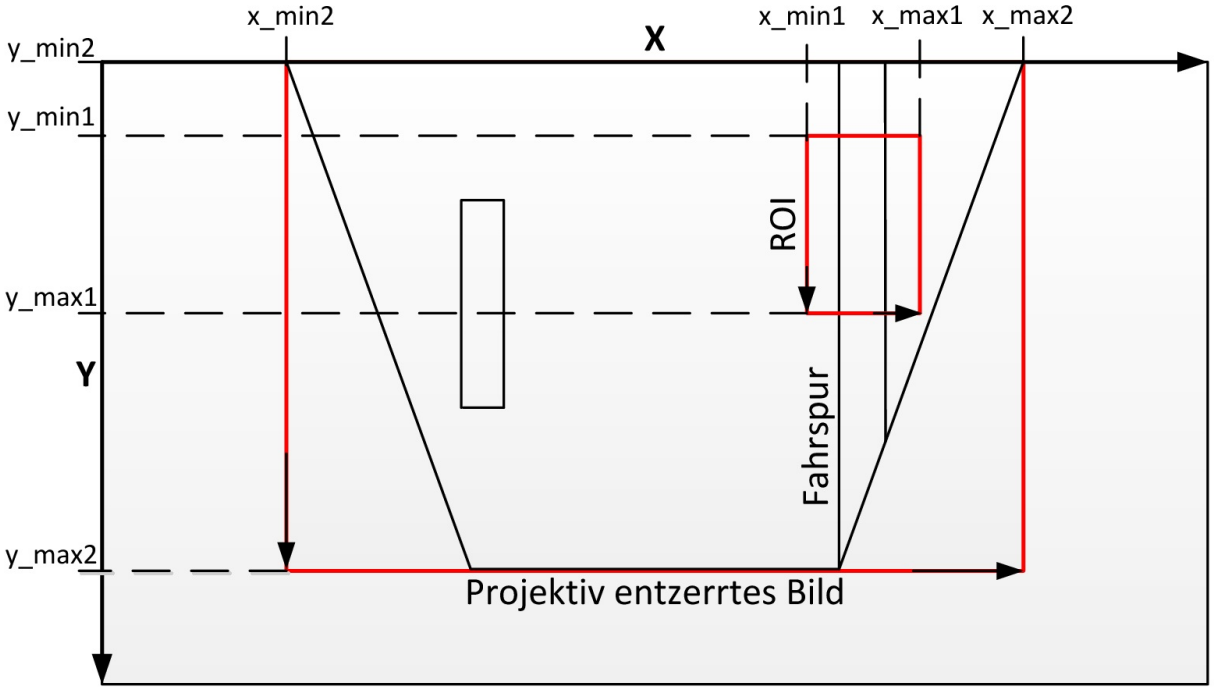

Abbildung 4.15.: Die Rot markierten Zonen zeigen die zu speichernden Speicherbereiche. Die mit dem Index 2 bezeichneten Koordinaten definieren das gesamte entzerrte Bild, während mit Index 1 nur den ROI-Bereich im Zeilenzwischenspeicher sichert.

4.4.2. Entwicklung eines dynamischen Adressgenerators

Befindet sich das zu speichernde Bild nicht im Ursprung und das Zeilenzwischenspeicher-RAM ist der Größe des Bildes angepasst, werden die berechneten Adressen ab einer bestimmten Grenze außerhalb des zu speichernden Adressraumes liegen. Damit dies nicht geschieht, wird für jedes Bild die Zone auf den Ursprung verschoben und die Pixel ab der Adresse 0 bis zur maximal erreichbaren Speicheradresse gespeichert.

Es gibt zwei dynamische Adressgeneratoren, Adressgenerator A berechnet aus den korrigierten Pixelkoordinaten *(x_pt und y_pt)* und dem definierten Speicherbereich die Adresse des zu speichernden Pixels *(xy_addr_out)* und übergibt ein Schreibzugriff *(write_enable)* an den Zeilenzwischenspeicher.

Der Adressgenerator B ermittelt mit den Synchronisationszählern X *(x_hcount)* und Y *(y_vcount)* die derzeitige Position des Bildes und leitet die daraus generierte Adresse *(xy_addr_out)* an das Zeilenzwischenspeicher und übergibt ein Lesezugriff *(roi_enable)* für das Auslesen des korrigierten Pixel, wenn die aktuelle Pixelposition in der ROI liegt.

Zur Synchronisation der Signale werden Delay-Blöcke eingesetzt, die am Ausgang des Modells eine Ausgangssynchronisation der Signale realisiert (vgl. Abb. 4.16). Die

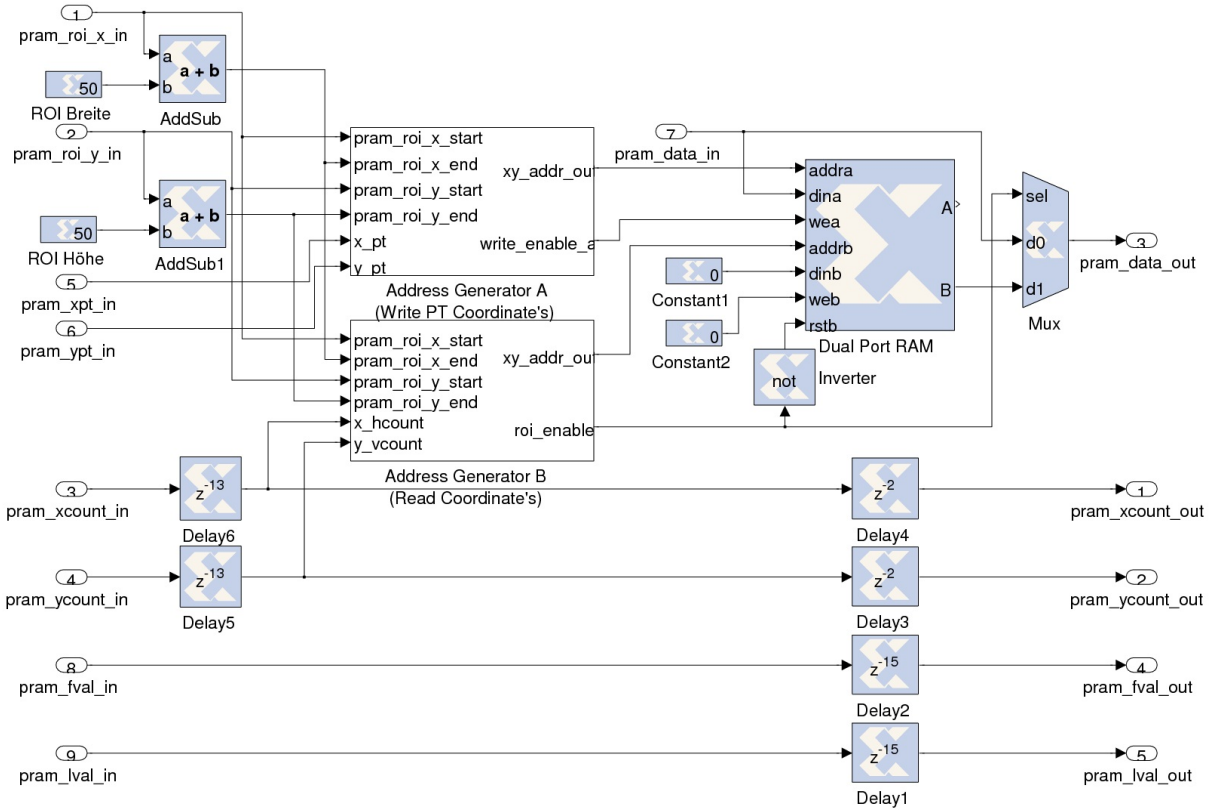

Abbildung 4.16.: Der Zeilenzwischenspeicher-RAM mit zwei dynamischen Adressgeneratoren und einem Dual-Port RAM mit Schreibzugriff auf Port A und Lesezugriff über Port B. Der Multiplexer übergibt im ROI-Bereich die korrigierten Pixeldaten weiter, ansonsten werden die verzerrten Bildpunkte weitergeleitet.

Eingangs- und Ausgangssignale für den Zwischenspeicher (vgl. Abb. 4.17) sind folgende:

Eingang:

roi_x, roi_y: Die Anfangskoordinate der Region of Interest wird auf den Ursprung verschoben und die maximale Adresse wird über die bekannte Größe der ROI ermittelt.

pt_x, pt_y: Die korrigierten Pixelkoordinaten werden auf die P(0,0) Koordinate verschoben und überprüft ob die Koordinate im ROI-Bereich liegt, bei Übereinstimmung der Zone wird der korrigierte Pixel in das Zeilenzwischenspeicher-RAM abgespeichert.

x, y: Die Synchronisationszähler zeigen die derzeitige Position des Pixels an der für die Darstellung verwendet. Ist die X- und Y-Koordinate an der unveränderten Position der ROI angelangt, so wird der abgespeicherte Pixel aus der entsprechenden Adresse geholt und visualisiert.

data: Der korrigierte Pixel wird an den Zeilenzwischenspeicher übergeben, falls dieser in der Bildzone liegt.

lval, fval: Synchronisationssignale die weitergeleitet werden, damit die Signale für weitere Module mit der Verzögerung des Zeilenzwischenspeicher-RAM wiederverwendet werden kann.

Ausgang:

x, y: Weiterleitung der Synchronisationszähler an das Darstellungsmodul mit der Verzögerung des Zeilenzwischenspeicher-RAM Modells.

data: Der Pixel der aus dem RAM ausgelesen wird, falls die Synchronisationszähler im ROI-Bereich liegen.

lval, fval: Synchronisationssignale die weitergeleitet werden, damit die Signale für weitere Module mit der Verzögerung des Zeilenzwischenspeicher-RAM wiederverwendet werden kann.

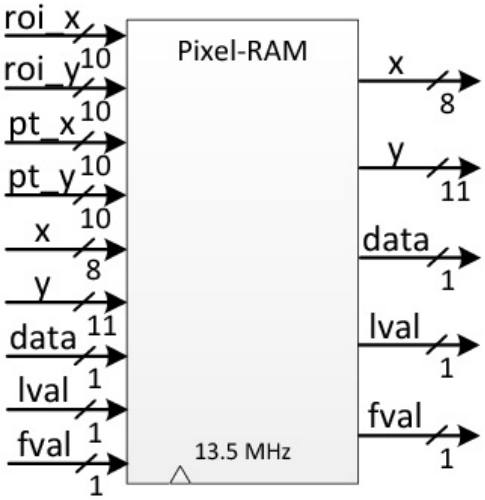

Abbildung 4.17.: Blackbox des Zeilenzwischenspeichers mit Angabe der Taktfrequenz

4.4.3. Visualisierungsergebnisse und Auswertung der Filterfunktion

Durch die Rekonstruktion der Pixelkoordinaten entsteht eine maximale Verzögerung von einem Bildintervall, da am Anfang der Videoübertragung noch keine Pixelinformationen im RAM vorhanden sind und erst beim nächsten Frame die Pixel des vorherigen dargestellt werden. Das verzerrte Bild wird mit einem Sobelfilter und einer Graubildbinarisierung mit Schwellwertanpassung dargestellt (vgl. Abschnitt 4.1), die auf den Bildern zunächst keine Unterschiede für die Geradenapproximation der Hough-Transformation zeigt (vgl. Abb. 4.18).

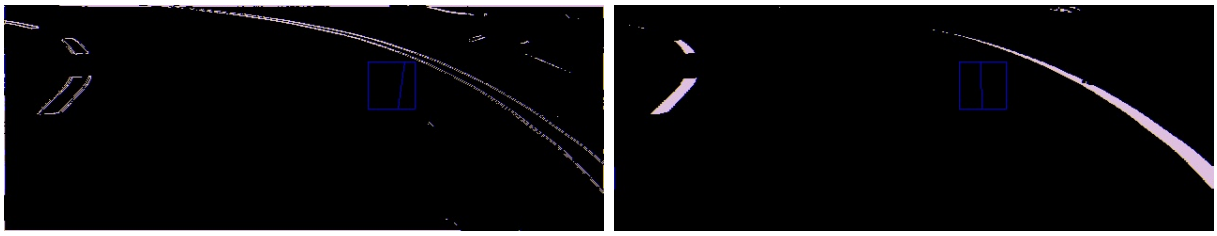

Abbildung 4.18.: Sobelgeflirtetes Bild (links) mit einer eindeutigen Kantenerkennung der Fahrspur. Eine Schwellwertanpassung mit Graubildbinarisierung und einer zusätzlichen morphologischen Erosion, zur geringen Maxima Bestimmung der HT.

Erst die Visualisierung des entzerrten Bildes veranschaulicht die Effektivität des vorher erprobten Sobelfilters gegenüber der Schwellwertanpassung mit Erosionfilter (vgl. die Abschnitte 4.1 und 4.2) und (vgl. Abb. 4.19). Die ROI und die Gerade sind im rechten Bild nicht von Ihrer Position gewichen und zeigen ein eindeutiges Ergebnis der Geradenapproximation. Die Bilder auf der linken Seite haben unterschiedliche ROI Positionen und Geraden, da die Geradenapproximation für ein sobelgefiltertes Bild durch die rauschhafte Anordnung der Pixel mehrere Lösungen enthält und eine feste Gerade nicht bestimmt werden kann.

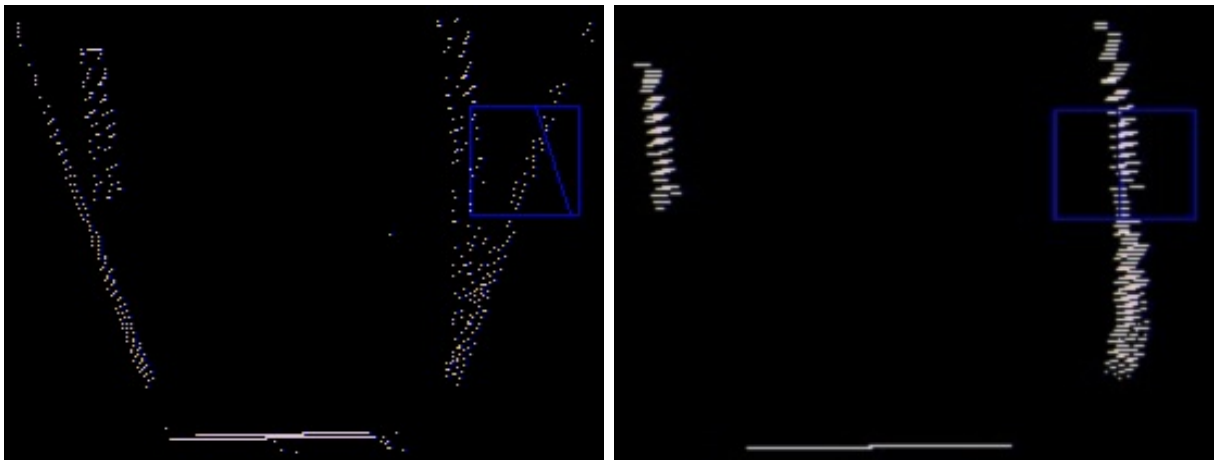

Abbildung 4.19.: Das entzerrte Bild mit Sobelfilter (links) zeigt rauschartige Pixelanordnungen und dadurch keine eindeutige Gerade über dir HT ermitteln kann. Die entzerrte Graubildbinarisierung (rechts) zeigt, dass die Gerade auf der Fahrspur liegt.

5. Adapterplatinen der SoC-Plattformerweiterung

Die Adapterplatinen *FFC-IDC* und *IDC-PMOD* werden zur Kopplung der CCD-Kamera (vgl. Abschnitt 2.1) mit dem Nexys2-Board (vgl. Abschnitt 2.2) verwendet (vgl. Abschnitt 2.5 Abb. 2.10). Der *FFC-FX2*-Adapter verbindet den LCD-Touch-Panel (vgl. Abschnitt 2.3) mit der Bildverarbeitungspipeline und wird für die Übertragung der Visualisierungssignale genutzt, um eine Darstellung der Bildinformationen anzuzeigen. Für das Ressourcen-Sharing zwischen dem TE0320 (vgl. Abschnitt 2.4) und der aktuellen Bildverarbeitungsplattform wird ein Erweiterungsadapter angeschlossen, der zusätzlich Anschlüsse für Sensorik bietet und über das TE0320-Board FPGA-Ressourcen zur Erweiterung des Fahrerassistenzsystems zur Verfügung stehen.

5.1. FFC-IDC-Adapter und IDC-PMOD-Adapter für die Pixelstromübertragung zur Bildverarbeitungspipeline

Das FFC-Kabel der CCD-Kamera wird an die *24P-FFC*-Schnittstelle des Adapters angeschlossen, diese enthält die Synchronisationssignale und die Videodaten. Separat wird über die *CN701*-Schnittstelle der Kamera RX ,TX zur Kamerakonfiguration und Gnd,Vcc für die Betriebnahme der Kamera an die Stiftleisten angesteckt (vgl. Abb. 5.1). Die Top-Layer Leiterbahnen führen zu dem *28P-IDC*, die mit Stiftleisten bestückt ist und über Flachbandkabel eine Verbindung zum *IDC-PMOD*-Adapter herstellt (vgl. Abschnitt 2.5 Abb. 2.10).

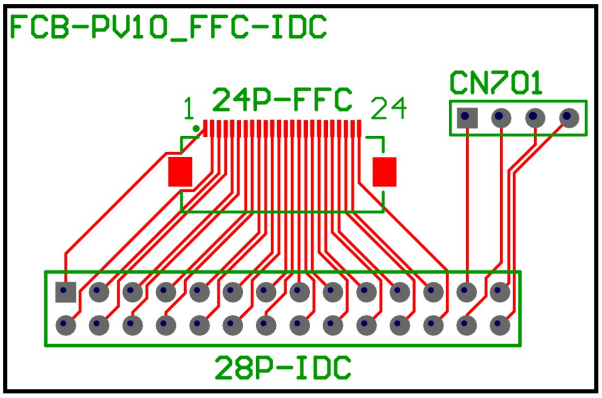

Abbildung 5.1.: Die Synchronisationssignale *(HSync, VSync)*, Pixeldaten *(Helligkeitskomponente, Farbkomponente)*, Parametrisierungsdaten *(RX, TX)* und die Pixelclock werden durch die Kopplung mit der Adapterplatine über einen IDC-Stecker an den IDC-PMOD-Adapter weitergeleitet (vgl. Tab. A.1).

Der PMOD-IDC-Adapter übergibt an die PMOD1-Schnittstelle des Nexys2 die digitalen Bildinformationen und an die PMOD2-Schnittstelle werden die Konfigurationssignale RX,TX zur Parametrisierung der Kamera zugestellt. Die Versorgungsspannung der CCD-Kamera wird aus der PMOD-Schnittstelle, des Nexys2-Boards bezogen (vgl. Abb. 5.2).

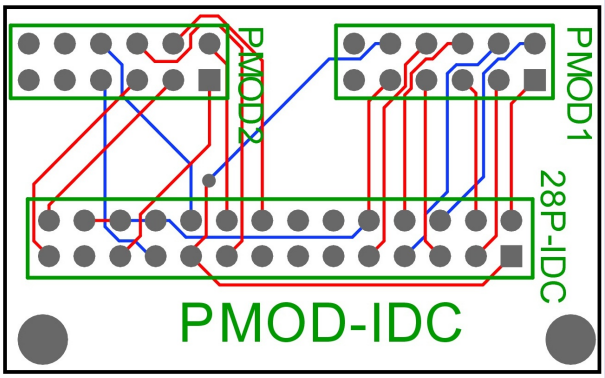

Abbildung 5.2.: Der PMOD-IDC-Adapter wird an die PMOD-Schnittstelle 1 & 2 des Nexys2-Boards gesteckt und leitet die Bildinformationen, die Pixelclock, die Synchronisationssignale und die Konfigurationssignale der Kamera an das FPGA weiter (vgl. Tab. A.2).

5.2. FFC-FX2-Adapter für eine Visualisierung des Echtzeit-Videodatenstroms

Die Bildinformationen *(Fahrspur, ROI, Geradenapproximation)* der Bildverarbeitungspipeline gelangen über die FX2-Schnittstelle an die 40P-FFC-Schnittstelle und werden mit dem FFC-Kabel an den LCD-Touch-Panel weitergeleitet (vgl. Abb. 5.3). Da das LCD einen integrierten Timing-Controller enthält (vgl. Abschnitt 2.3 Abb. 2.6), werden nur die 6Bit-RGB-Daten *(Data)* mit einem Datenenable-Signal *(DTMG)* und der Pixel-Clock für die Anzeige verwendet. Der LCD-Touch-Panel bezieht die Versorgungsspannung und die Masse aus der FX2-Schnittstelle, indem diese an den BHR-03VS-01 übergeben wird.

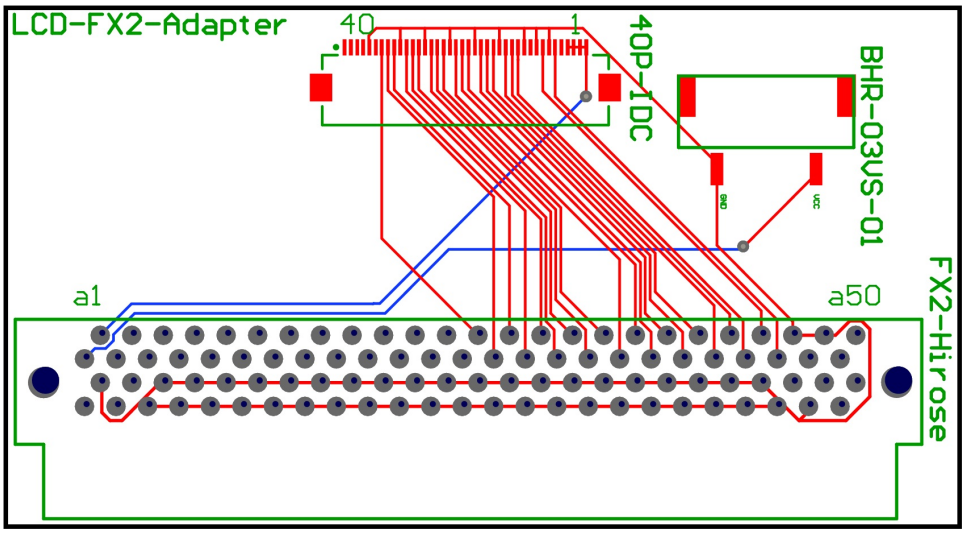

Abbildung 5.3.: Der LCD-Touch-Panel ist über ein FFC-Kabel mit der 40P-FFC-Schnittstelle verbunden, dabei ist der LCD-FX2-Adapter an den Nexys2-Board gekoppelt. Die zu Übertragung Signale sind die Pixel-Clock, die RGB-Daten und der Datenberechtigungssignal (vgl. Tabelle A.3).

Die Signallaufzeitpfade der Leiterbahnen zwischen dem FX2 und dem FFC wurden so gelegt, dass die Längendifferenz keine Auswirkungen auf die synchrone Datenübertragung hat, dabei hat die übertragene Pixel-Clock einen längeren Laufzeitpfad, damit die RGB-Signale gleichzeitig eingetaktet werden.

5.3. Kopplung des Nexys2-Boards mit einem zusätzlichem Spartan3A DSP FPGA

Die Erweiterung der FPGA-Ressourcen ist erforderlich, da der Bedarf der weiterentwickelten Bildverarbeitungsplattform (vgl. Abschnitt 3.1) zu 98% verbraucht ist (vgl. Anhang B.6) und für weitere Funktionen die Ressourcen nicht ausreichen. Die JM5-Schnittstelle des TE0320 (vgl. Abschnitt 2.4) wird für die Datenübertragung zwischen den beiden FPGA's genutzt und ist direkt mit der FX2-Schnittstelle über die Leiterbahnen verknüpft (vgl. Abb. 5.4).

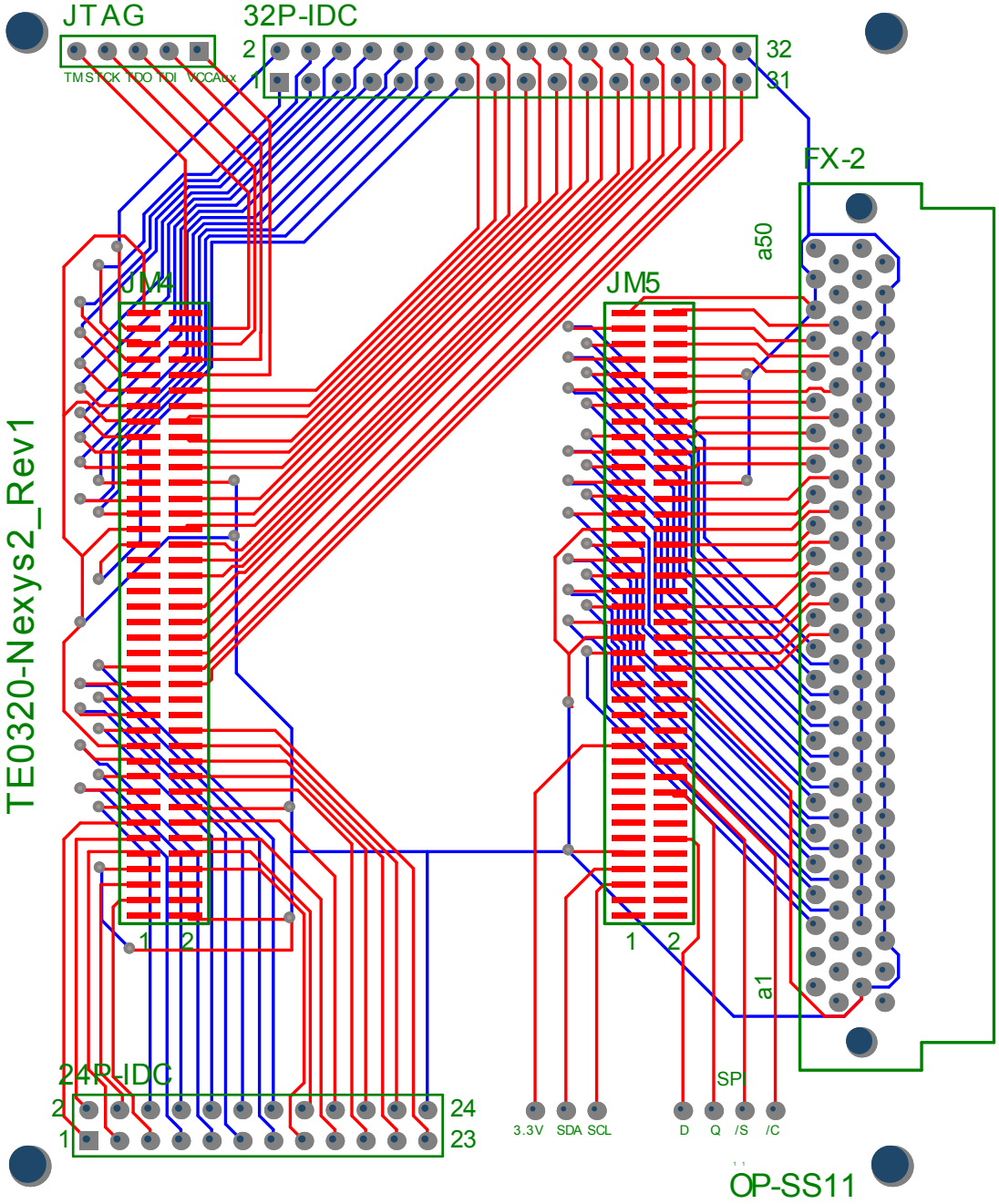

Abbildung 5.4.: Erweiterungsadapter die das TE0320-Board aufnimmt und über die FX2-Schnittstelle des Nexys2 mit dem Spartan 3E-1200K koppelt (vgl. Tabelle A.4).

Die vom TE0320-Board nach außen geführten Stiftleisten:

- Der TE0320 wird über eine externe Stiftleiste *(Vcc-B2B)* in Betrieb genommen.

- Die Konfiguration des FPGA's erfolgt über die JTAG-Schnittstelle.
- Der SPI-Bus wird für das Beschreiben/Auslesen des PROM-Files (FPGA-Bitstream Konfiguration) in/aus das/dem SPI-Flash genutzt (vgl. Abschnitt 2.4) Tab. 2.2).
- Der I2C-Bus wird über den USB-Microcontroller zum beschreiben der USB-Firmware in das EEPROM verwendet (vgl. Abschnitt 2.4) Tab. 2.3).
- Die JM4-Schnittstelle wird für Sensoranschlüsse auf 24 Pin- und 32 Pin-Stifleisten überführt, diese haben unter anderem auch dedizierte Clockeingänge und eine Masseleitung, die an den FX2 angebunden ist, damit ein Masse-Bus entsteht und somit Signalstörungen vermieden werden (vgl. Abschnitt 3.4.2 Abb 3.6).

Die FPGA-IOs des Spartan3A DSPs haben eine direkte Verbindung zu den IOs des Spartan3E. Die Datenübertragung erfolgt synchron, da die Signallaufzeitpfade der Leiterbahnen als eine Pipelinestufe zu betrachten sind (vgl. Abb. 5.5). Anhand eines Beispiels wurden die Laufzeiten der gepipelineten Zwischenstufe überprüft und im Rahmen der gesendeten Clock ermittelt, ob die parallel übertragenen Daten ohne Signalverluste ankommen. Die aus dem Beschleunigermodul weitergeleiteten Daten werden in der IOB über die DCM am Ausgang synchronisiert und mit der DCM_Clk an das Spartan-3A DSP übertragen. Die mitgesendete CLK wird zur Eingangssynchronisation am IOB_In verwendet. Die Signallaufzeit wurde am IOB_Out und am IOB_In der beiden FPGA's gemessen und auf einem Oszilloskop ausgewertet.

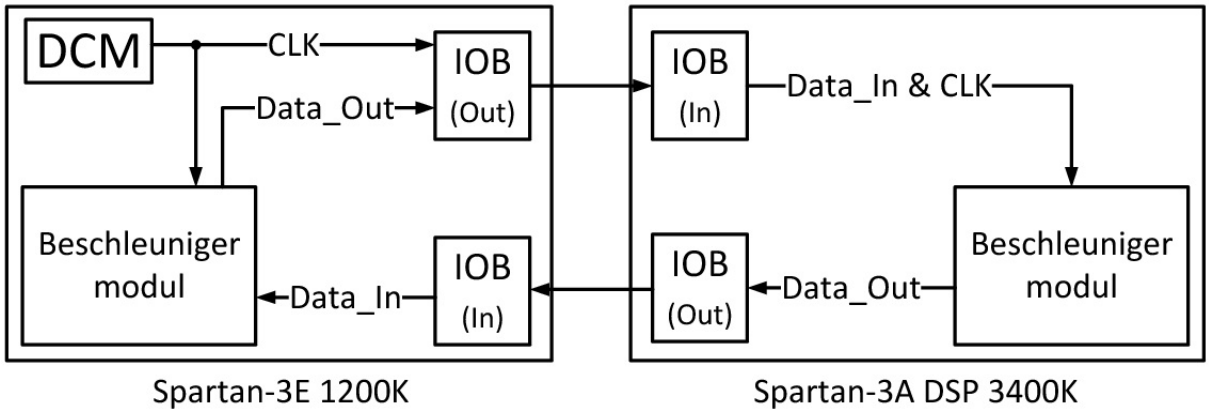

Abbildung 5.5.: Synchrone Datenübertragung zwischen dem Spartan-3E 1200K und dem Spartan-3A DSP 3400K. Die IOBs werden als Pipelinestufen verwendet, während die mit übertragende DCM_Clk auf beiden FPGA's zum Eintakten der Daten genutzt wird.

Eine Verzögerung von 1.6 ns des 18 cm langen Kupferkabels schließt ein Datenverlust aus, da die Periodendauer 40 ns beträgt und die Übertragungsstrecke 25 mal schneller ist als die Eintaktung der Daten (vgl. Abschnitt 3.4.2). Die festen Verzögerungen der IOBs während des Sendens und des Empfangs sind aus den jeweiligen Data Sheets des Spartan-Typs zu entnehmen, um eine Gesamtverzögerung des Signals zu berechnen.

In der Abbildung 5.6 wird Kanal A für die Messung des Ausgangspins *(IOB_Out)* verwendet, während Kanal B den Eingangspin *(IOB_In)* misst und am Oszilloskop die Phasenverschiebung anzeigt.

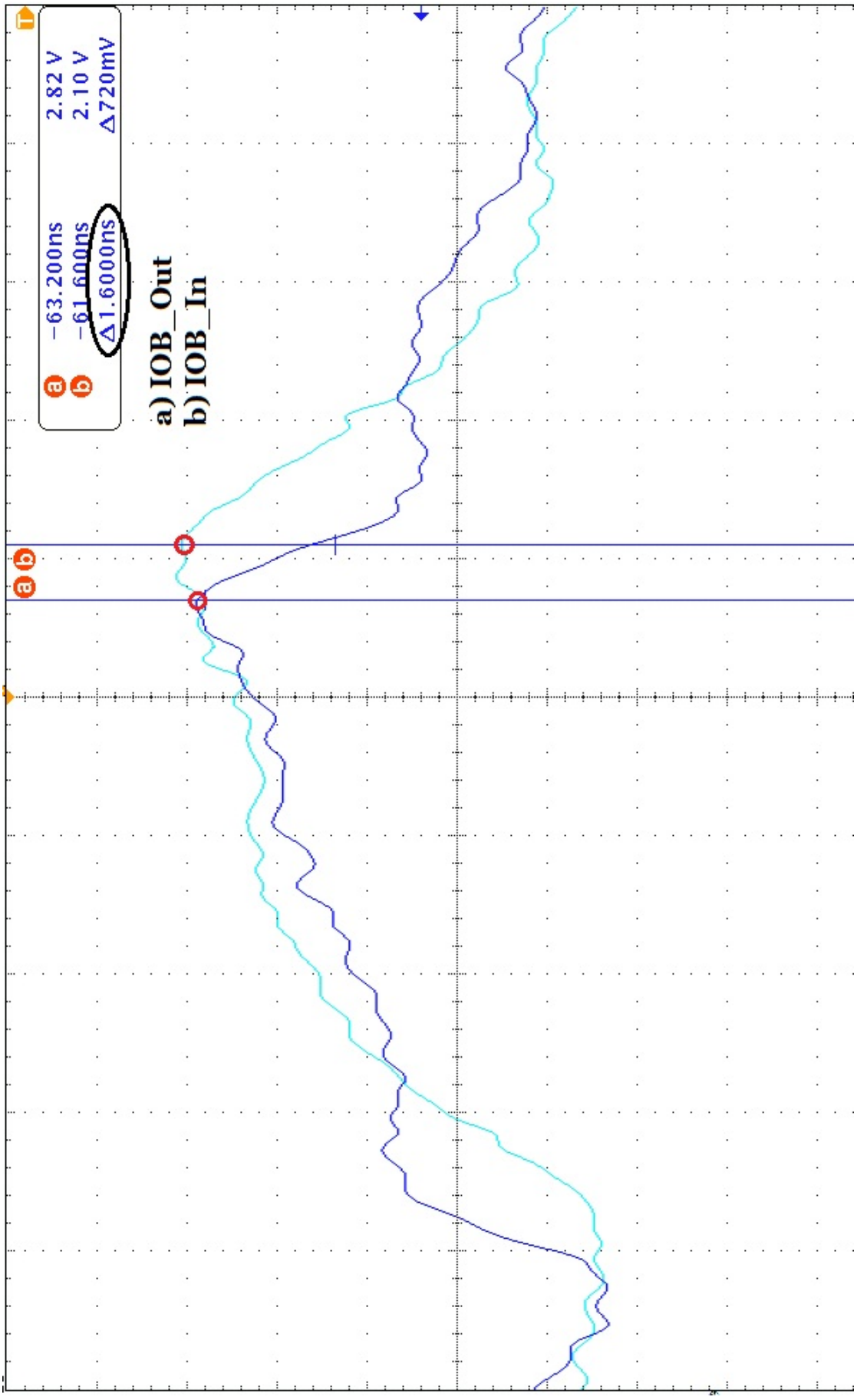

Abbildung 5.6.: Ein Kupferkabel von 18 cm länge wurde zur Bestimmung der Laufzeitpfadverzögerung am Kanal A und B gemessen. Die Verzögerung von 1.6 ns wird anhand der Differenz der Phasenverschiebung am Oszilloskop angezeigt.

6. Verifikation der Bildverarbeitungspipeline-Modelle mit Matlab/Simulink

In diesem Abschnitt werden die Verifikationsschritte der Subsysteme *'Adaptive Binarize'*, *'Erosionfilter'*, *'Projective Transformation'* und *'Image Buffer'* aus der Bildverarbeitungspipeline beschrieben (vgl. Abschnitt 3.1 Abb. 3.1). Die Verarbeitung des Bildstromes erfolgt über *'Adaptive Binarize'* (vgl. Abschnitt 4.1) und *'Erosionfilter'* (vgl. Abschnitt 4.2, während die perspektivische Entzerrung des Bildes mit dem Submodul *'Projective Transformation'* (vgl. Abschnitt 4.3) und für die Rekonstruktion der Pixelreihenfolge die Komponente*'Image Buffer'* (vgl. Abschnitt 4.4) genutzt wird.

6.1. Bildverarbeitungsmodelle zur Aufbereitung des Videodatenstroms

Das Subsystem *Adaptive-Binarize* stellt die Schwelle je nach Helligkeit des Bildes ein und binarisiert die Graustufenwerte anhand des ermittelten Schwellwertes (vgl. Abschnitt 4.1). Hierzu wird mit einem Matlab-Code (system_init.m, vgl. Anhang B.1) eine Tabelle erstellt, die an das Simulinkmodell übergeben wird um ein Ausgangsbild zu erzeugen (vgl. Abb. 6.1).

Die Pixelinformationen sind in der Tabelle *(Data)* enthalten, die über ein im Matlab-Skript gespeichertes Bild in Form von Pixelkoordinaten an die Datenspalte eingetragen wird. Mit den beiden *for*-Schleifen wird über die x- und y-Koordinate des Bildes der Pixel dem Array zugewiesen. Die Eintaktung der Werte ereignet sich zur Zeit t, die bis zum Bildende aufgezählt wird. Die Tabellen *LVAL* und *FVAL* sind mit '1' initialisiert, da während der Simulation keine Pausen zwischen der Bildübertragung besteht.

```
% STIMULI INIT
DATA = zeros(size_x*size_y,2);
LVAL = zeros(size_x*size_y,2);
FVAL = zeros(size_x*size_y,2);

t = 1;

for y = 1:size_y
    for x = 1:size_x
        DATA(t,1) = t;
        DATA(t,2) = I(y,x);      % Pixeldaten des gespeicherten Bildes
        LVAL(t,1) = t;
        LVAL(t,2) = 1;
        FVAL(t,1) = t;
        FVAL(t,2) = 1;
        t = t + 1;
    end
end
```

Abbildung 6.1.: Die Eingangsdaten *DATA*,*LVAL* und *FVAL* werden mit Nullen initialisiert. In der doppelten *for*Schleife wird *DATA* mit dem Pixel der x- und y-Koordinate des Bildes zugewiesen (system_init.m, vgl. Anhang B.1).

Die aus den Variablen übertragenen Werte werden an die In-Blöcke des Xilinx Blocksets übergeben und im Q-Format digitalisiert.

Die Variable *VAL_BINARIZE_OUT* wird im Workspace als Array gespeichert, die zur Auswertung der aufbereiteten Pixelinformationen zur Verfügung steht (vgl. Abb. 6.2). Das binarisierte Bild wird an das *Erosionfilter*-Modul weitergeleitet und die Fahrspur auf beiden Seiten um ein Pixel verschmälert, so dass die Hough-Transformation eine eindeutige Gerade ermitteln kann.

Das Pixelergebnis der kontrahierten Fahrspur wird an die *VAL_EROSION_OUT* übergeben. Die Synchronisationssignale *LVAL*- und *FVAL*-Signale sind konstant mit einer '1' initialisiert und werden mit einer Verzögerung der Subsysteme weitergeleitet.

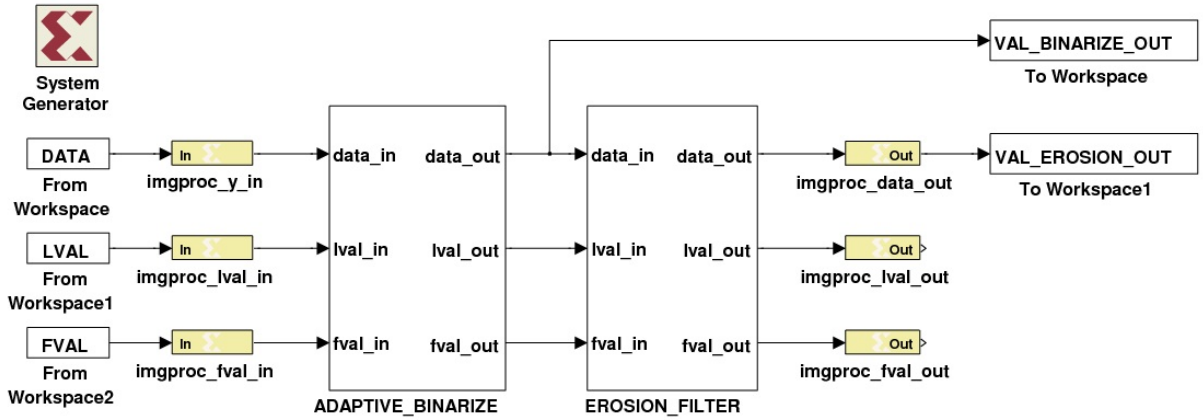

Abbildung 6.2.: Aus den generierten Stimulisignalen bereiten die Subsysteme die Pixeldaten auf und leiten sie an die 'To Workspace'-Komponente weiter. Die Ergebnisse liegen als Array vor (*VAR_BINARIZE_OUT* und *VAR_EROSION_OUT*) und werden in einem Matlab-Skript ausgewertet.

Die Variablen *IP_BIN* und *IP_ERO* werden als 2D-Arrays mit nullen initialisiert. Die Ausgangswerte des Arrays werden über eine doppelte *for*-Schleife zugewiesen, da die Pixelkoordinate des Bildes über die Aufzählung der x- und y-Koordinate bestimmt wird (vgl. Abb. 6.3). Die Funktion *imshow* wird für die Anzeige des aufbereiteten Bildes verwendet, während *figure* die Anzeigen auf verschiedene Fenster teilt.

```
IP_BIN = zeros(size_y,size_x); %2D-Binarize Array
IP_ERO = zeros(size_y,size_x); %2D-Erosionfilter Array
i = 1;
for Y=1:size_y
    for X=1:size_x
            IP_BIN(Y,X) = VAL_BINARIZE_OUT(i); %Zuweisung der Ausgangswerte
            IP_ERO(Y,X) = VAL_EROSION_OUT(i);
            i = i+1;
    end;
end;
figure(1);
imshow(IP_BIN); %Ansicht der zugewiesenen Werte
figure(2);
imshow(IP_ERO);
```

Abbildung 6.3.: Anzeige der aufbereiteten Bilder über eine doppelte for-Schleife und der Zuweisung der Ausgangswerte in das 2D-Array (Image_Processing_TEST.m, vgl. Anhang B.3).

In der Abbildung 6.4 zeigt die adaptive Grauwertschwelle *(links)* einen klaren Unterschied

zwischen der Fahrbahn und der Fahrspurmarkierung. Eine Pixelstörung über die Reflektion des Lichtes auf der Fahrbahn wurde mit der dynamischen Schwelle ausgeschlossen.

Im rechten Bild ist deutlich zu erkennen, dass die Fahrspur durch das 3x3 Erosionfilter auf allen Seiten um ein Pixel verschmälert wurde. Die Geradenapproximation der Hough-Transformation [Mel10],[BB06] führt durch die Kontraktion der Fahrspur zu einem eindeutigen Maxima.

Abbildung 6.4.: Die Auswertungen der beiden Subsysteme 'Adaptive-Binarize' (links) und 'Erosionfilter' (rechts) verifizieren die Funktionalität der System-Generator-Modelle und zeigen eine deutliche Hervorhebung der Fahrspur.

6.2. Projektive Transformation mit Zeilenzwischenspeicher für die Visualisierung des projektiv Entzerrten Bildes

Die Verifikation der projektiven Transformation (vgl. Abschnitt 4.3) erfolgt über die Auswertung der korrigierten Pixelkoordinaten, indem die MATLAB 'plot'-Funktion die berechneten Punkte auf eine 2D-Achse einzeichnet.

Die Pixelkoordinaten X und Y und die Pixeldaten *Data* werden als Stimulisignale an das Subsystem *Projektive Transformation* übergeben (vgl. Abb. 6.5). Für die Verifikation werden nur die entzerrten Bildkoordinaten ausgewertet pt_x_out und pt_y_out.

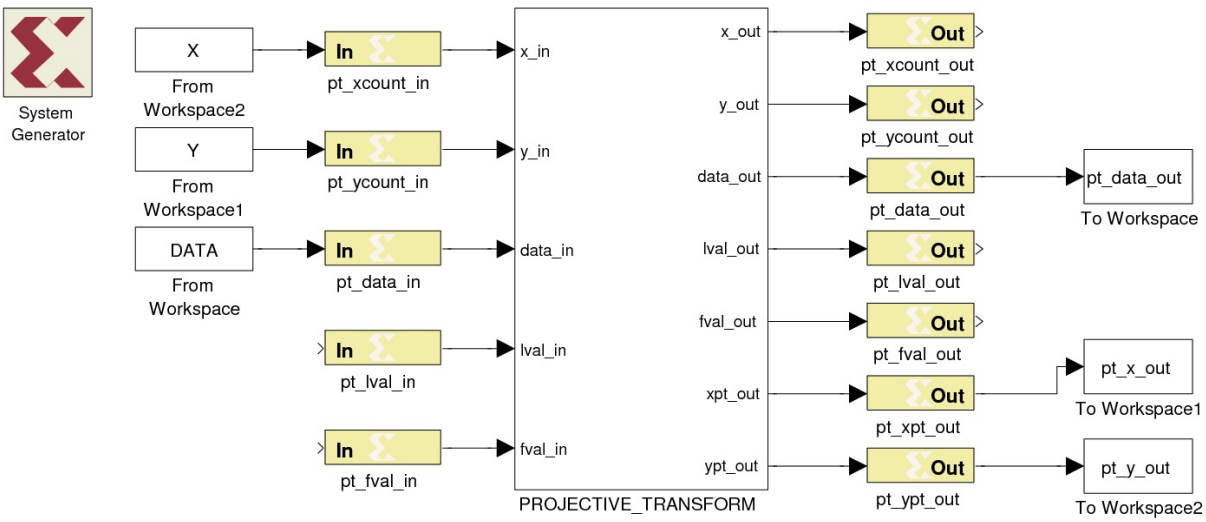

Abbildung 6.5.: Die Stimulisignale *(X)* und *(Y)* werden zur Berechnung der korrigierten Pixelkoordinaten an das Subsysteme *Projektive Transformation* übergeben (system_init.m, vgl. Anhang B.1). Die Ergebnisse (*pt_x_out* und *pt_y_out*) werden über die plot-Funktion ausgewertet.

In der Abbildung 6.6 korrespondieren die x-Koordinaten gegen das Bildende, dies führt zu einer Verschmälerung des Bildes und erzeugt die virtuelle Draufsicht der Fahrspur. Am

Bildanfang sind die Koordinaten ausgedehnt, daher entstehen Lücken in denen die Pixel nicht abgebildet werden. Der negative Bildbereich beschränkt sich auf die Paarbildung mit den negativen y-Koordinaten (vgl. Abb 6.7), die für die ersten 4000 Pixel gilt (vgl. Gl. 6.1).

Auflösung des getesteten Bildes 640(H) x 240(V):

$$\frac{4000(Pixel)}{640(H)} = 6.25(Zeilen) \tag{6.1}$$

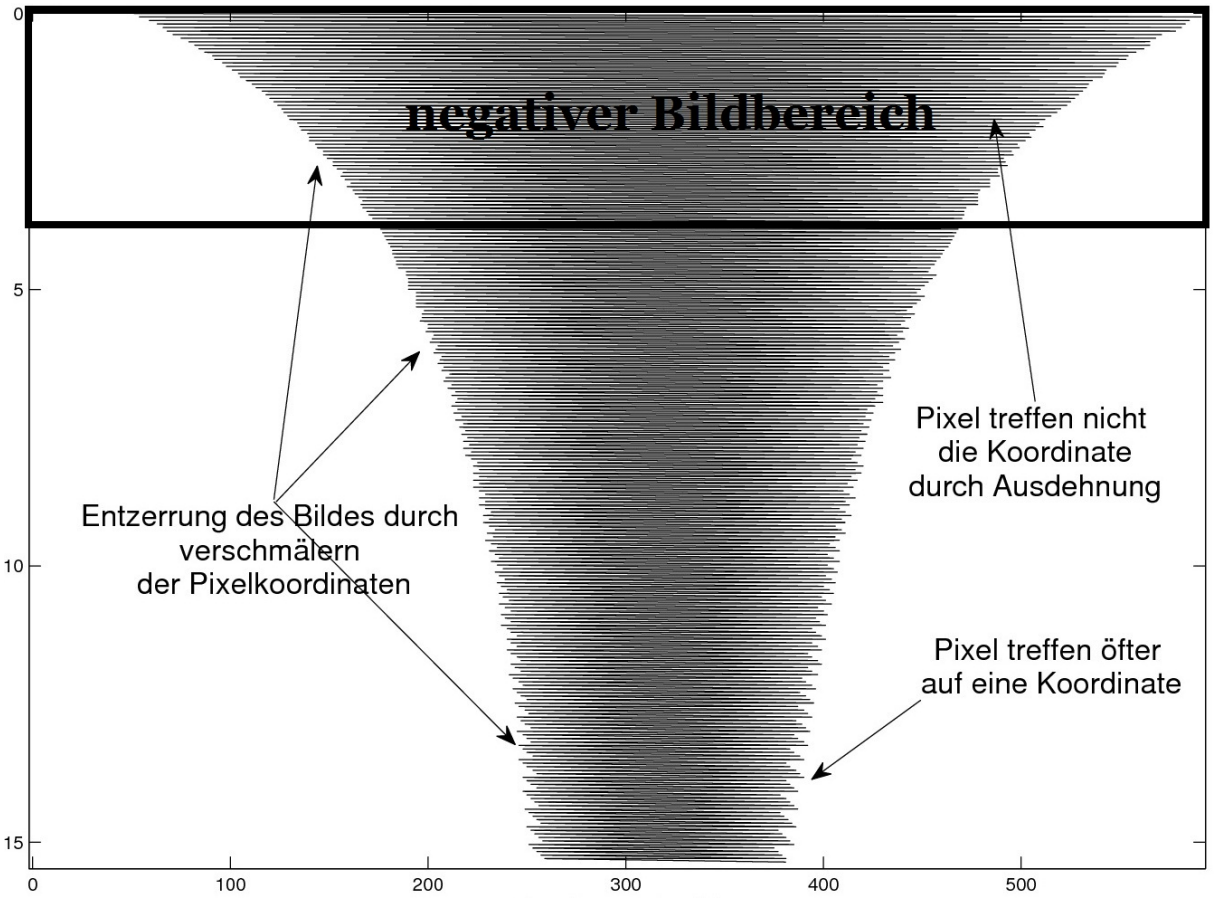

Abbildung 6.6.: Die y-Achse zeigt den Zählstand der Synchronisationszähler xcount und ycount, während die x-Achse den Bereich der x-Koordinate darstellt. Der negative Bildbereich ist für die Darstellung der projektiven Transformation nicht geeignet.

Durch die Transformation erfolgt im unteren Bereich des Bildes eine Verdichtung und im oberen Bildbereich eine Ausdehnung der Pixelkoordinaten. Die Verdichtung führt dazu, dass mehrere Pixel die selbe Koordinate treffen, während die Ausdehnung die Pixel voneinander trennt und dadurch Lücken entstehen.

Im eingerahmten Bereich der Abbildung 6.7 ist zu erkennen, dass sich bei der y-Koordinate ein Rücksprung ereignet, die zu einer Unordnung der Pixelkoordinaten führt. Weiterhin sind die in der PT weitergeleiteten Pixel nicht mit den korrigierten Koordinaten in Zusammenhang gestellt, daher wird für die Darstellung der projektiven Transformation ein Zeilenzwischenspeicher verwendet, der die Pixelkoordinaten rekonstruiert und die Verbindung zwischen Pixel und den entzerrten Bildpunkten wiederherstellt.

Die korrigierten y-Koordinaten liegen im Bereich [-90;214], die Entstehung der negativen Pixelkoordinaten hängt mit der Verdichtung der Bildpunkte im unteren Bildbereich zusammen, dadurch weichen die Bildpunkte im oberen Bereich aus und gelangen in die

negativen Bildzeilen.

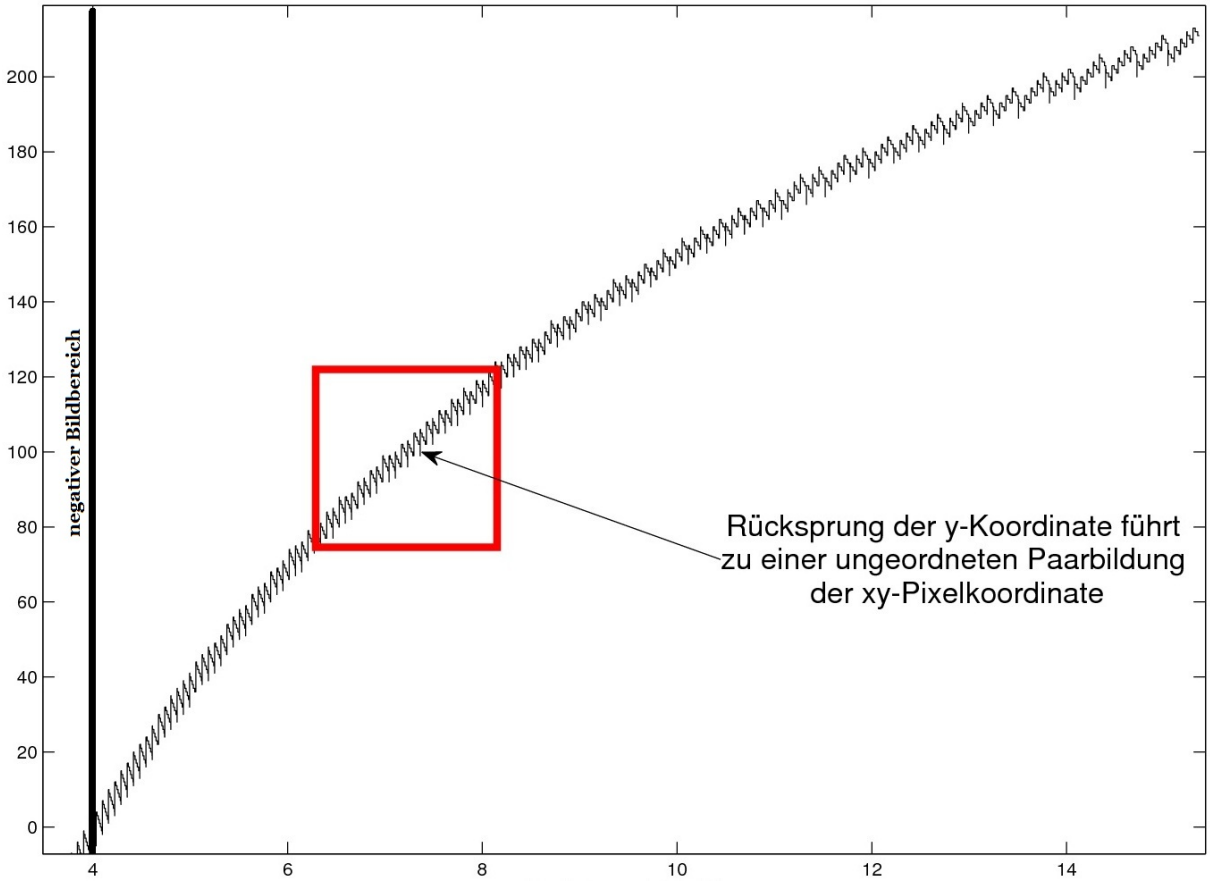

Abbildung 6.7.: Durch die projektive Transformation erfolgt ein kontinuierlicher Rücksprung der y-Koordinate in eine vorherige Ordinate. Die negativen Pixelkoordinaten werden für die Darstellung nicht genutzt.

Das entzerrte Bild wird angezeigt, indem ein Zeilenzwischenspeicher (vgl. Abschnitt 4.4 Abb. 4.16) die ungeordneten Pixelkoordinaten rekonstruiert. Der Adressgenerator (vgl. Abschnitt 4.4.2) der die Adresse des korrigierten Pixels ermittelt, passt die maximale Adressgröße dem entzerrten Bild an.

Die Verifikation des Adressgenerators erfolgt über zwei Eingangswerte, ein Zähler für die x-Achse und für die y-Achse (vgl. Abb 6.8. Der Bereich des zu speichernden Bildes wird über die minimale und maximale x- und y-Koordinate eingegrenzt. Die Relational-Blöcke ermitteln dann, ob die Pixelkoordinate zwischen diesen Grenzwerten liegt. Das Expression-Block gibt eine eins aus, wenn alle Relational-Blöcke zutreffen und übermittelt die aus den Koordinaten berechnete Adresse an den Zeilenzwischenspeicher (vgl. Gl. 6.2).

$$\begin{aligned} x_{origin} &= x_{pt} - x_{ptstart} \\ y_{origin} &= y_{pt} - y_{ptstart} \\ y_{area} &= y_{origin} * (x_{ptend} - x_{ptstart}) \\ address_{pt} &= y_{area} + x_{origin} \end{aligned} \tag{6.2}$$

Der Adressgenerator wird einmal für das beschreiben der korrigierten Pixel genutzt und für das Auslesen der Pixel über die Adressgenerierung der Synchronisationszähler, die für die Anzeige genutzt werden.

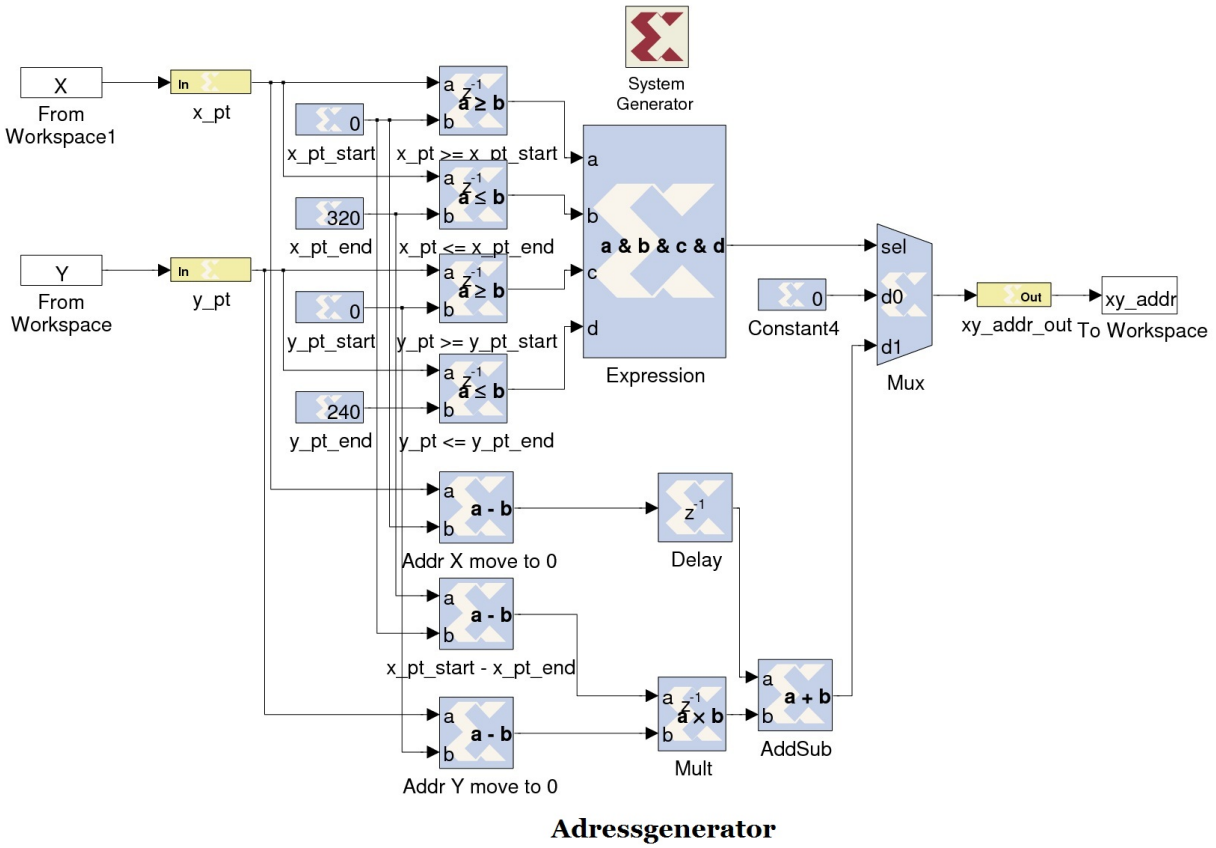

Abbildung 6.8.: Die Eingangswerte des Adressgenerators werden über Relational-Blöcke auf das Zutreffen im Bildbereich liegenden Pixelkoordinaten überprüft (vgl. Abschnitt 3.2). Die Berechnung der Pixeladresse wird an das Zeilenzwischenspeicher weitergeleitet, wenn das Expression-Block eine eins ausgibt (system_init, vgl. Anhang B.1).

In der Abbildung 6.9 sind die Ausgangswerte der beiden Adressgeneratoren eingetragen, während die lineare Funktion *(Blau)* die Pixeladresse der Synchronisationszähler ausgibt und die logarithmische Funktion *(Grün)* die Pixeladresse der korrigierten Pixel anzeigt. Die maximal zulässige Pixeladresse wird mit den maximalen x- und y-Werten der korrigierten Pixelkoordinaten berechnet.

Durch den Rücksprung der korrigierten Pixeladressen werden die weitergeleiteten Pixel an den Zeilenzwischenspeicher überschrieben.

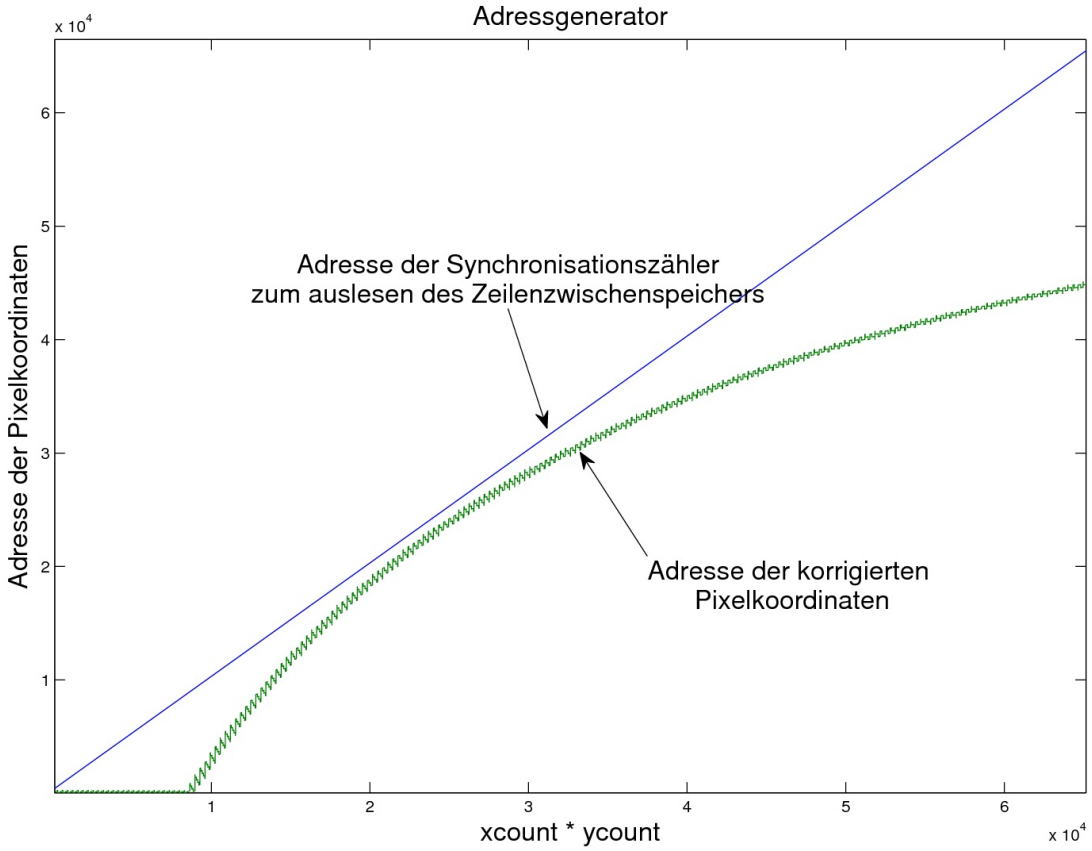

Abbildung 6.9.: Pixeladresse der Synchronisationszähler und der korrigierten Pixelkoordinaten.

Die Verifikation des Zeilenzwischenspeichers erfolgt über die Auswertung der korrigierten Pixelkoordinaten, die einmal über die Software berechnet wurden (system_init.m, vgl. Anhang B.1) und mit dem System Generator als Ausgangswerte in einem Array gespeichert. Die Pixel des verzerrten Bildes werden an die korrigierten Bildpunkte übergeben, dadurch entstehen die in der Abbildung 6.10 dargestellten entzerrten Bilder. Die mit dem Matlab-Skript (system_init.m, vgl. Anhang B.1) berechneten entzerrten Pixelkoordinaten *(links)* bestätigen die im oberen Bildbereich entstandenen Lücken, während die Auswertung des SysGen-Moduls *(rechts)* durch die Einschränkung der Fixed-Point-Arithmetik Näherungsweise das selbe Ergebnis vorweist. Durch das Runden und Abschneiden verschieben sich die Positionen der Koordinaten, dadurch liegen die Punkte der Fahrspur nicht immer Senkrecht aufeinander.

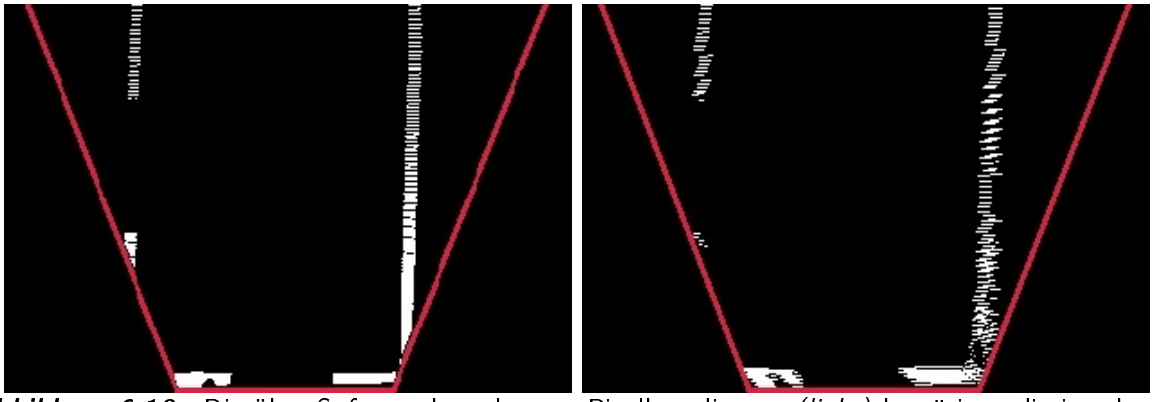

Abbildung 6.10.: Die über Software berechneten Pixelkoordinaten *(links)* bestätigen die im oberen Bildbereich entstandenen Lücken. Die Ausgangswerte des SysGen-Moduls *(rechts)* verifizieren die Funktionalität der generierten Netzliste.

7. Ergebnisanalyse der Fahrspurerkennung

Für die Fahrspurerkennung wird das Ergebnis des aufbereiteten Bildes analysiert und erläutert. Hierbei wird auf die Schwellwertanpassung mit Graubildbinarisierung (vgl. Abschnitt 4.1), die morphologische Erosion (vgl. Abschnitt 4.2),die perspektivische Entzerrung des Bildes (vgl. Abschnitt 4.3) und für die Rekonstruktion der Pixelkoordinaten den Zeilenzwischenspeicher-RAM eingegangen (vgl. Abschnitt 4.4).

Schwellwertanpassung mit Graubildbinarisierung: Die feste Grauwertschwelle führt zu einer Störung im oberen Bildbereich (vgl. Abb. 7.1 *(links)*), die durch die Reflektion des Lichtes auf der Fahrspur erscheint, da die Helligkeit des Lichtes die Schwelle überschreitet. Die adaptive Grauwertschwelle *(rechts)* hat eine klare Unterscheidung der Fahrbahn mit der Fahrspurmarkierung und beinhaltet keine Störungen im Bild, da sich der Schwellwert der Helligkeit anpasst.

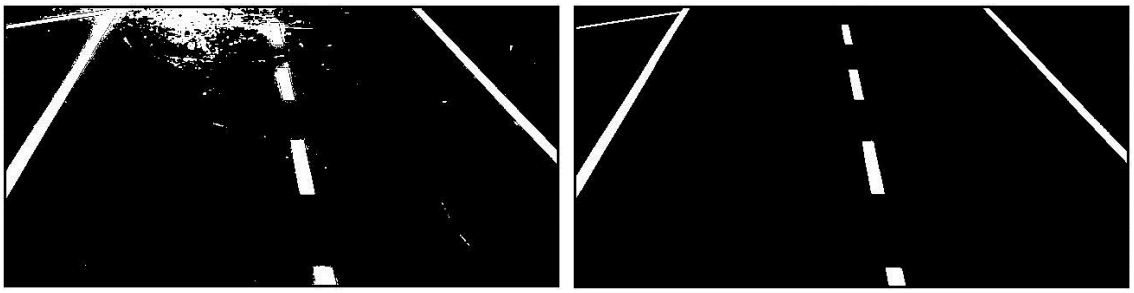

Abbildung 7.1.: Im unveränderten Bild (links) sind durch die Reflektion des Lichtes Störungen im oberen Bildbereich zu erkennen, während die adaptive Grauwertschwelle (rechts) zu einem sauberen Bild führt und die Fahrspurmarkierung hervorhebt.

Morphologische Erosion: Das grauwertskalierte Bild wird mit einem quadratischen 3x3 Erosions-Faltungskern zur Kontraktion der Fahrspurmarkierung verarbeitet. Die Fahrspurmarkierung wird jeweils an jeder Seite um 2 Pixel verkleinert und führt zu einer Verminderung der Maximabildung in der Hough-Transformation (vgl. Abb. 7.2).

Die Verschmälerung der Fahrspur führt zu einer eindeutigen Geraden, die mit den Parametern r (Abstand) und Φ (Winkel) bestimmt wird. Die Erprobung des Erosions-Filters am RC-Fahrzeug hat gezeigt, dass im Stillstand der Lenkservo sich nicht bewegt und zusätzlich durch die Darstellung der Geradenapproximation erkennbar ist, dass die in der ROI eingezeichnete Gerade unverändert bleibt (vgl. Abb. 4.19 (rechtes Bild, ROI-Gerade)).

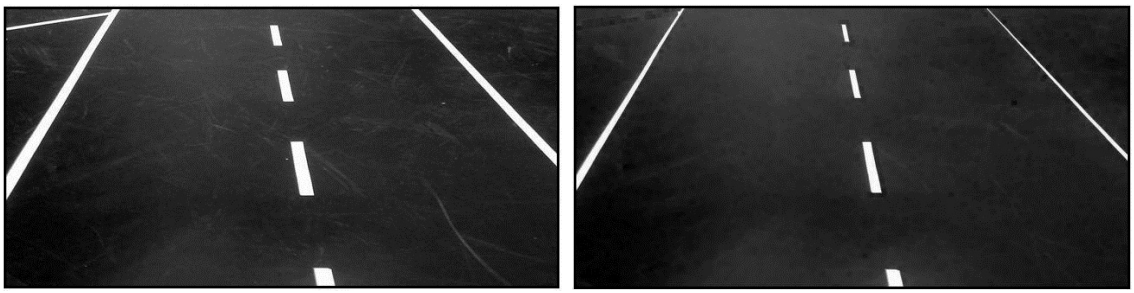

Abbildung 7.2.: Die Fahrspurmarkierungen im grauwertskalierten Bild (links) wurden durch Anwendung eines quadratischen 3x3 Erosions-Faltungskern verschmälert (rechts).

Projektive Transformation der Kameraperspektive: Durch die Simulation der projektiven Transformation werden die berechneten perspektivisch korrigierten Bildkoordinaten über die 'plot'-Funktion ausgewertet. In der linken Abbildung 7.3 ist zu erkennen, dass der untere Bereich des Bildes verdichtet wird, während im oberen Bildbereich das Bild sich ausdehnt und dadurch Lücken entstehen, da die gesetzten Pixel sich voneinander entfernen. Durch den Rücksprung der y-Koordinate (vgl. Abb. 7.3 *(rechts)*), entsteht eine Unordnung während der Pixelzuweisung zu den korrigierten Bildkoordinaten. Die Rekonstruktion erfolgt über einen Zeilenzwischenspeicher-RAM (vgl. Abschnitt 4.4).

Des Weiteren wurde für die Visualisierung die Auflösung der korrigierten Bildkoordinaten verdoppelt und in die Mitte des Bildes verschoben, damit mehrere Pixel der Fahrspur erfasst werden und das entzerrte Bild mit einer höheren Auflösung dargestellt wird.

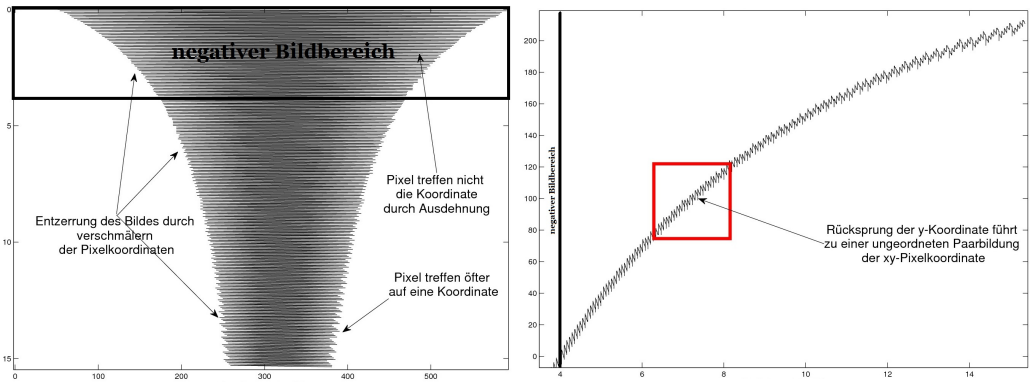

Abbildung 7.3.: Die Trichterform (rechts) zeigt die perspektivisch korrigierten x-Koordinaten. Im linken Bild erfolgt ein kontinuierlicher Rücksprung der y-Koordinate.

Visualisierung und Gegenüberstellung der Filterfunktion: Die kaskadierten Filteroperationen *(Adaptive Binarize und Erosionfilter)* (vgl. Abschnitt 4.1 und 4.2) wurden mit dem Sobelfilter gegenübergestellt (vgl. Abb. 7.4), indem über User-Switches die Filtervarianten gewechselt und zusätzlich eine Anzeige des entzerrten Bildes mit der houghtransformierten Gerade dargestellt wurde. Im linken Bild führt die pixelhafte Anordnung der sobelgefilterten Fahrspurmarkierung zu wechselhaften Ergebnissen, während im rechten Bild eine unveränderte Gerade im Stillstand des Fahrzeugs ermittelt wurde.

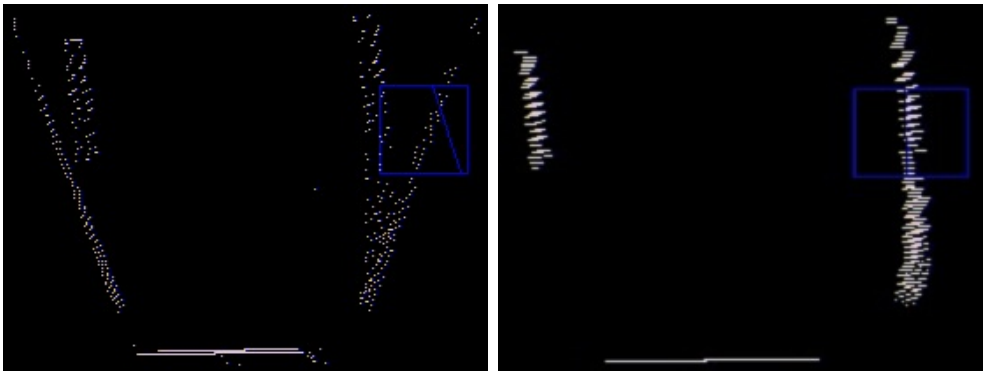

Abbildung 7.4.: Sobelfilter mit pixelhafter Anordnung und wechselnden Geradenergebnissen (links). Unveränderte Gerade mit dynamischen Schwellwert und Erosionfilter (rechts).

8. Weiterentwicklung

Durch die Erweiterung der vorentwickelten Bildverarbeitungspipeline mit der Schwellwertanpassung mit Graubildbinarisierung (vgl. Abschnitt 4.1) und der morphologischen Erosion (vgl. Abschnitt 4.2) in Verbindung mit der Fahrspurführung [Sch11] wurde das RC-Fahrzeug auf der miniaturisierten Fahrbahn erfolgreich in Betrieb genommen. Aufgrund der Systemerweiterung mit der Kamera und dem TE0320-Board (vgl. Abschnitt 2.4) ist im Laufe der Bearbeitung eine Verschiebung der Schwerpunkte eingetreten, sodass weitere Schritte zur Inbetriebnahme des LCD-Touch-Panels erforderlich sind.

LCD-Touch-Panel FSM: Die FSM steuert die Anzeigemodi über die Wahl der perspektivischen Betrachtung und der Filtervariationen für die Fahrspurerkennung. Mit dem Touch-Panel soll eine Parametrisierung der Kamera erfolgen können und zusätzlich die Darstellung der Fahrzeugparameter angezeigt werden.

Font-ROM Generator: Damit eine Darstellung der Parameter erfolgen kann, ist der entwickelte Font-ROM Generator an die Darstellungskomponente der Bildverarbeitungspipeline anzubinden. Dieser holt sich die Buchstaben aus dem Block-RAM und bildet Schritt für Schritt die Buchstaben über die Sychronisationszähler.

Cordic Algorithmus für Hough-Transformation: Der Cordic Algorithmus nutzt für die Berechnung nur Addier- und Shift-Blöcke, die für die Geradenapproximation in der Gesamtbetrachtung weniger FPGA-Ressourcen verbraucht als die derzeitig verwendete Hough-Transformation. Mit dem Cordic Algorithmus wird weniger Block-RAM benötigt und somit kann eine höhere Genauigkeit erzielt werden, indem der Winkelabstand erniedrigt wird.

HW/SW Partitionierung des Fahrspurführungsmoduls: Das derzeitige Gesamtsystem (vgl. Anhang B.5 und B.9) nutzt den MicroBlaze nur für die Erzeugung von Bildern, die eine aktuelle Sicht der Kamera anzeigt. Da mit einem zusätzlichen FPGA die Ressourcen erweitert wurden, kann nun eine HW/SW Partitionierung für die Bildverarbeitungspipeline ermittelt und ausgeführt werden.

Einparkassistenzsystem: Das Einparkassistenzsystem verwendet die Infrarotsensoren um ein Einparken vorzunehmen. Hierbei wird mit den festen Parametern des Fahrzeugs und einer S-förmigen Einlenkung das Fahrzeug zwischen zwei Objekte gesteuert.

Hinderniserkennung und Ausweichsystem: Die Hinderniserkennung wird in Kooperation der Infrarotsensoren mit dem vorne montierten Laserscanner ausgeführt und über ein Ausweichsystem Vorkehrungen für die Manövrierung vorgenommen.

Entscheidungslogik der Fahrspurführungsalgorithmen: Die Fahrspurführung [Sch11] wurde mit 3 Algorithmen erprobt, dabei wirkt sich das Fahrverhalten je nach Geschwindigkeit unterschiedlich aus. Eine Entscheidungslogik würde je nach Geschwindigkeit und Fahrsituation das beste Fahrspurführungsalgorithmus bestimmen und sich der Situation anpassen.

9. Zusammenfassung

Diese Arbeit dokumentiert die Entwicklungsergebnisse einer Fahrspurerkennung und -visualisierung eines Fahrspurführungssystems, die über eine FPGA-basierte SoC-Plattform ausgeführt wird und zusätzlich die Kopplung der Bildverarbeitungsplattform mit einer CCD-Kamera, einem LCD-Touch-Panel und einem weiteren FPGA-Board darstellt.

Die Visualisierung der Fahrspur erfolgte über einen VGA-Monitor, die zur Gegenüberstellung der kaskadierten Filtervarianten genutzt wurde. Die Ergebnisbilder wurden mit dem Framegrabber IP-Core abgespeichert und ausgewertet.

Filtervarianten: Für die Fahrspurerkennung wurde eine Schwellwertanpassung mit einer Graustufenbinasierung vorgenommen, diese ermittelt die höchste Graustufe im Bild und setzt die Schwelle auf 87,5% der ermittelten Graustufe, um die Fahrspur aus dem Bild zu extrahieren und hervorzuheben. Die morphologische Erosion führt zu einer kontrahierten Fahrspur, damit bei der Geradenapproximation die Maxima Bildung verringert wird, um eine eindeutige Gerade zu erkennen.

Die beiden kaskadierten Filter werden mit dem Sobel-Operator verglichen, die zur Kantendetektion genutzt wird, dabei wurde durch die Erprobung der Filter am RC-Fahrzeug festgestellt, dass die dynamische Schwellwertanpassung mit der morphologischen Erosion sich gegenüber dem Sobel-Operator bei der Geradenerkennung hervorhebt.

Projektive Transformation: Die Montierung der CCD-Kamera auf das RC-Fahrzeug führt zu einer Neukalibrierung der Transformationsmatrix über ein Matlab-Skript, die für die Korrektur der verzerrten Bildpunkte verwendet wird. Für die Visualisierung wurde die Auflösung der korrigierten Bildkoordinaten verdoppelt und in die Mitte des Bildes verschoben, damit mehrere Pixel der Fahrspur erfasst werden und das entzerrte Bild dargestellt wird.

Zeilenzwischenspeicher-RAM: Für die Darstellung der projektiven Transformation auf einem VGA-Monitor wurde der Pixel-RAM mit einer Latenz von einem Bildintervall zur Rekonstruktion der korrigierten Bildkoordinaten genutzt. Der Zeilenzwischenspeicher-RAM kann dynamisch den Bereich des entzerrten Bildes einstellen, der über die korrigierten x- und y-Koordinaten ermittelt wird. Der Speicherbereich ist auf das projektiv entzerrte Bild ausgelegt, damit eine gesamte perspektivisch entzerrte Darstellung erfolgt und zusätzlich dazu die Mischung aus einem verzerrten Bild mit dem korrigierten Bild in der ROI angezeigt wird.

LCD-Touch-Panel: Für die Handheld-Visualisierung der Bildverarbeitungspipeline wurde eine Gegenüberstellung von vier LCD-Touch-Panels und die daraus folgende Auswahl getroffen. Die Kopplung erfolgte über eine Adapterplatine, die an die FX2-Schnittstelle des Nexys2-Boards angeschlossen wurde. Aufgrund der Systemerweiterung mit einer Kamera und eines weiteren FPGA-Boards, wurde der Schwerpunkt der Visualisierung mit dem LCD-Touch-Panel auf eine Weiterentwicklung verschoben.

Erweiterungsadapter für die Bildverarbeitungsplattform: Da der Ressourcenbedarf 98% der Bildverarbeitungsplattform abdeckt (vgl. Anhang B.6), wird ein weiteres FPGA-Board mit der Plattform über eine Adapterplatine gekoppelt, um weitere Funktionen zu implementieren. Der Erweiterungsadapter führt die freien IOs des TE0320-Boards auf Stiftleisten mit 24-Pins und 34-Pins, damit weitere Sensoren für die Fahrspurführung eingesetzt werden. Die SPI- und I2C-Busse des Spartan-3A DSPs sind auf einzelne Stiftleisten überführt, die zum Beschreiben und Auslesen des SPI-Flashspeichers und des EEPROMs genutzt werden.

Die Kopplung weiterer Adapterplatinen mit der Bildverarbeitungsplattform wird für die Datenübertragung der Kamera bis zur Bildverarbeitungspipeline und für die Anzeige der Echtzeit-Videodaten genutzt.

Matlab/Simulink Simulation: Zur Erzeugung der Stimulisignale für die Simulink-Modelle Adaptive-Binarize, Erosionfilter, Projective-Transformation und Pixel-RAM wurde ein Initialisierungs-Matlab-Skript geschrieben, das die Pixeldaten und die Synchronisationszähler und zusätzlich für den Pixel-RAM die korrigierten Pixelkoordinaten generiert. Durch die Stimulisignale werden während der Simulation Ausgangswerte erzeugt, die in einer Matlabvariable als Array gespeichert werden.

Anhand der Simulationsergebnisse erfolgt eine Auswertung mit der plot-Funktion oder über ein Matlab-Skript und die darauf folgende Verifikation des Simulink-Models. Nach der Verifikation wird das Simulink-Model in die Bildverarbeitungspipeline integriert.

10. Literaturverzeichnis

[BB06] Burge, Mark J. ; Burger, Wilhelm: *Digitale Bildverarbeitung - Eine Einführung mit Java und ImageJ.* Berlin : Springer-Verlag, 2006 http://www.springerlink.com/content/m00486/. – ISBN 978–3–540–30940–6

[Dig08] Digilent, Inc.: *Digilent Nexys2 Board Reference Manual.* Pullman : Digilent, 2008 http://www.digilentinc.com/Data/Products/NEXYS2/Nexys2_rm.pdf

[DLR10] DLR: *Allgemeine Hinweise zur BMBF-Förderung von Embedded Systems.* Bundesministerium für Bildung und Forschung, 2010

[Erh08] Erhardt, Angelika: *Einführung in die Digitale Bildverarbeitung - Grundlagen, Systeme und Anwendungen.* Wiesbaden : Vieweg+Teubner | GWV Fachverlage GmbH, 2008. – ISBN 978–3–519–00478–3

[Hit06] Hitachi: *LCD-Touch-Panel - Customers Acceptance Specifications - TX14D12VM1CAA.* Hitachi, 2006

[Hor11] Horsinka, Sven A.: *Integration einer Hough-Transformation zur Fahrspurerkennung in eine SoC-basierte Videoverarbeitungspipeline.* Hamburg : Hochschule für Angewandte Wissenschaften Hamburg, Bachelorarbeit, 2011

[Mel10] Mellert, Dennis: *Modellierung einer Video-basierten Fahrspurerkennung mit dem Xilinx System Generator für eine SoC-Plattform.* Hamburg : Hochschule für Angewandte Wissenschaften Hamburg, Bachelorarbeit, 2010

[Sch11] Schneider, Christian: *Ein System-on-Chip-basiertes Fahrspurführungssystem.* Hamburg : Hochschule für Angewandte Wissenschaften Hamburg, Bachelorarbeit, 2011

[Sha02] Sharp: *Device Specification for TFT-LCD module Model No. LQ050Q5DR01.* Sharp, 2002

[Sha07] Sharp: *Color TFT-LCD Module LQ057Q3DC02 Specifications.* Sharp, 2007

[Son06] Sony: *FCB-PV10 Color Camera Module - Technical Manual.* Sony, 2006 http://pro.sony.com/bbsccms/assets/files/mkt/indauto/manuals/FCB-PV10_Technical_Manual.pdf

[Tos07] Toshiba: *14 cm Colour TFT-LCD Module (5.7 Type) LTA057A344F - Product Information.* Toshiba, 2007

[Tre10] Trenz, Electronic: *TE0320 Series User Manual - Xilinx Spartan-3A DSP Industrial-Grade FPGA Micromodule.* Trenz-Electronic, 2010 http://docs.trenz-electronic.de/Trenz_Electronic/TE0320/UM-TE0320.pdf

[Xil09] XILINX: *Spartan-3E FPGA Family: Data Sheet.* Xilinx, 2009 `http://www.xilinx.com/support/documentation/data_sheets/ds312.pdf`

[Xil10a] XILINX: *MicroBlaze Processor Reference Guide - Embedded Development Kit EDK 12.4.* Xilinx, 2010 `http://www.xilinx.com/support/documentation/sw_manuals/xilinx12_4/mb_ref_guide.pdf`

[Xil10b] XILINX: *Spartan-3A DSP FPGA Family: Data Sheet.* Xilinx, 2010 `http://www.xilinx.com/support/documentation/data_sheets/ds610.pdf`

[Xil10c] XILINX: *System Generator for DSP - Reference Guide.* Xilinx, 2010 `http://www.xilinx.com/support/documentation/sw_manuals/xilinx12_4/sysgen_ref.pdf`

[Xil10d] XILINX: *System Generator for DSP - User Guide.* Xilinx, 2010 `http://www.xilinx.com/support/documentation/sw_manuals/xilinx12_4/sysgen_user.pdf`

11. Abbildungsverzeichnis

1.1. SoC-Plattform für die Entwicklung eines Fahrspurführungssystems für ein autonomes Modellfahrzeug mit der Visualisierung der Bildverarbeitungsergebnisse über einen VGA-Monitor. Der Softcore-Prozessor *(μBlaze)* wird als Daten-Logger des Bildstromes mit einem Desktop-PC über die RS232-Schnittstelle verbunden (vgl. Anhang B.7. 5

2.1. SoC-Plattform mit Komponentenkopplung über Erweiterungsadapter zur Visualisierung mit einem LCD. 8
2.2. Funktionsweise des Interline-Transfer CCD-Sensors über parallele und serielle Schiebetechniken. 9
2.3. Farbunterabtastung über YCbCr 4:2:2 zur Reduzierung der zu übertragenden Datenmenge für Chrominance Blue und Chrominance Red. 9
2.4. Zeilensprungverfahren über gerade und ungerade Halbbildern zur Gesamtbildgenerierung. 10
2.5. Digilent Nexys2-Entwicklungsboard mit dem Spartan 3E-1200K FPGA. . . 10
2.6. LCD Block-Diagramm zur Ansteuerung des LCD-Touch-Panels 13
2.7. Trenz-Electronic TE0320-Board mit dem Spartan-3A DSP 3400A FPGA . 14
2.8. Master-Slave-Datenbus für die Datenübertragung an die Flash- und EEPROM-Komponente [Tre10]. 15
2.9. Xtreme DSP-Slice mit Carry-Arithmetik, Pre-Adder und dedizierten Ein-/Ausgängen für Kaskadierungen und mit einer Ausgangsbitbreite von 48. . 16
2.10. Gekoppelte SoC-Plattform mit IDC-PMOD- und FFC-IDC-Adapter für die Verbindung der FCB-PV10 CCD-Kamera mit dem Nexys2 FPGA Board und die Verknüpfung der FPGA's über den TE0320-Adapter 17
2.11. Bildverarbeitungspipeline in der vorentwickelten Version mit einem Sobel-Filter zur Kantenerkennung und integriertem Softcoreprozessor, der eine projektive Transformation der approximierten Gerade ausführt [Hor11]. . . 18

3.1. In der weiterentwickelten Bildverarbeitungskette wurde im Subsystem *'Image Processing'* die projektive Transformation modifiziert und mit dem *'Adaptive-Binarize'* und dem *'Erosionfilter'* erweitert. Das Subsystem *'Result Illustration'* wurde mit dem *'Image-Buffer'* erweitert um eine perspektivische Entzerrung darzustellen. Im *'Lane Detection'* wurde die Hough-Transformation angepasst und eine dynamische ROI angebunden [Sch11]. Im Anhang B.9 sind die Subsysteme im VHDL-Code an die Bildverarbeitungspipeline als Komponenten gekoppelt. 20
3.2. Der LCD Timingchart zeigt die Datenberechtigung über das Data Enable. Über die Pixel Clock und dem Berechtigungssignal werden die VGA-Standardsignale HSync, mit dem Front-Porch, Back-Porch und dem Pulse-Width Anteil sowie auch das VSync generiert.[GEMCompiler] 22
3.3. Erstellung eines Projektes: $File \rightarrow New \rightarrow Project \rightarrow PCB\ Project$ . . . 24

3.4. Erstellung eines Stromlaufplans zur Generierung eines PCB-Layouts: *File* → *New* → *Schematic* 24
3.5. PCB-Layout für die Adapterplatine zur Kopplung des LCD-Touch-Panels mit dem Nexys2-Board 25
3.6. Der zentrale Massepunkt wird von dem Nexys2-Board an die Komponenten weitergeleitet um Rückkopplungen der Masse zu vermeiden. 26
3.7. Konfiguration eines MicroBlaze-basierten Mikrocontroller-Systems. [Xil10a] 27
3.8. Beschleunigermodul mit Softwareregistern zum getakteten Schreiben und kombinatorischen Lesen [Hor11]. 27
3.9. Gateway In/Out Blöcke zur Typkonversion der Eingangs- und Ausgangswerte der Verifikationstabellen, um die Funktion der Xilinx Subsysteme zu testen. 28

4.1. Modifizierte Bildverarbeitungskette: Die Binarisierungsschwelle wird pro Bild anhand des bisher höchsten Grauwertes bestimmt. Anschließend wird ein Erosionsfilter verwendet um Bildrauschen zu unterdrücken und Konturen zu verkleinern. Zur Entzerrung des Kameradatenstroms werden mit Hilfe der Synchronisierungssignale Koordinaten generiert, die aus der Kameraebene in die Fahrzeugebene projiziert werden. 29
4.2. Graph der linearen Skalierungsfunktion f(g). 30
4.3. Unverändertes (links) und linear skaliertes Graubild der Fahrbahn (rechts) bei höherer Lichtintensität. Die starke Lichtquelle führt zu Reflektionen auf der Fahrbahn und geringeren Grauwertunterschieden zwischen Fahrbahn und Fahrspurmarkierung. 30
4.4. Die markanten Grauwerthäufungen im Histogramm des unveränderten Bildes (links) bleiben im linear skalierten Histogramm (rechts) erhalten, werden aber deutlicher hervorgehoben und voneinander besser unterscheidbar. 31
4.5. Binarisierung mit adaptiver Grauwertschwelle wird mit einer Taktfrequenz von $f_{13.5}$ betrieben. Für eine eingehende Datenreduktion werden die Grauwerte binarisiert. 31
4.6. Die ermittelten Extremwerte werden mit einer festen Grauwertschwelle verrechnet und mit dem aktuellen Eingangswert verglichen. 32
4.7. Das mit einem festen Schwellwert verarbeitete Bild (links) hat durch die Reflektion des Lichtes im oberen Bildbereich Störungen (weiße Flächen), während im rechten Bild durch die Anpassung des Schwellwertes die Fahrspurmarkierung ohne Störungen hervorgehoben wird. 33
4.8. Geraden mit unterschiedlichen Abständen r und Φ passen in in die Fahrspurmarkierung (grau). 33
4.9. Auf der Umgebung des Zentrumpixels im Quellbild *Eingangsbild g(x,y)* wird ein quadratischer Faltungskern (Strukturierendes Element) angewendet und das Ergebnis der Faltung an der Stelle *g'(x,y)* in das Zielbild eingetragen.[Erh08] 34
4.10. Blackbox des Erosionsfilter-Moduls, das den eingehenden Binärpixeldatenstrom mit der Frequenz $f_{13.5}$ mit einer quadratischen 3x3 Faltungsmaske durch UND-Verknüpfung erodiert. 36

4.11. Die Kreise markieren die Stellen der erodierten Pixel. a) Eingangsbild, b) Erosionsgrenze = 8, also eine UND-Verknüpfung der 3x3-Faltungsmatrix. [Erh08] 36
4.12. Deserialisierung durch Linebuffer-Blöcke mit einem 3x3-Faltungskern durch UND-Gatter als Erosionfilter genutzt. 37
4.13. Die Fahrspurmarkierungen im grauwertskalierten Bild (links) wurden durch Anwendung eines quadratischen 5x5 Erosions-Faltungskern verschmälert, sodass diese in der Hough-Transformation zu weniger mehrdeutigen Maxima führen. 37
4.14. Die Referenzierung des Bildpunktes π' von der Kamera-Ebene auf das Pixel π der Fahrbahnebene über die projektive Entzerrung der verzerrten Bildkoordinaten. 38
4.15. Die Rot markierten Zonen zeigen die zu speichernden Speicherbereiche. Die mit dem Index 2 bezeichneten Koordinaten definieren das gesamte entzerrte Bild, während mit Index 1 nur den ROI-Bereich im Zeilenzwischenspeicher sichert. 41
4.16. Der Zeilenzwischenspeicher-RAM mit zwei dynamischen Adressgeneratoren und einem Dual-Port RAM mit Schreibzugriff auf Port A und Lesezugriff über Port B. Der Multiplexer übergibt im ROI-Bereich die korrigierten Pixeldaten weiter, ansonsten werden die verzerrten Bildpunkte weitergeleitet. 42
4.17. Blackbox des Zeilenzwischenspeichers mit Angabe der Taktfrequenz 43
4.18. Sobelgeflirtetes Bild (links) mit einer eindeutigen Kantenerkennung der Fahrspur. Eine Schwellwertanpassung mit Graubildbinarisierung und einer zusätzlichen morphologischen Erosion, zur geringen Maxima Bestimmung der HT. 44
4.19. Das entzerrte Bild mit Sobelfilter (links) zeigt rauschartige Pixelanordnungen und dadurch keine eindeutige Gerade über dir HT ermitteln kann. Die entzerrte Graubildbinarisierung (rechts) zeigt, dass die Gerade auf der Fahrspur liegt. 44

5.1. Die Synchronisationssignale *(HSync, VSync)*, Pixeldaten *(Helligkeitskomponente, Farbkomponente)*, Parametrisierungsdaten *(RX, TX)* und die Pixelclock werden durch die Kopplung mit der Adapterplatine über einen IDC-Stecker an den IDC-PMOD-Adapter weitergeleitet (vgl. Tab. A.1). . . 45
5.2. Der PMOD-IDC-Adapter wird an die PMOD-Schnittstelle 1 & 2 des Nexys2-Boards gesteckt und leitet die Bildinformationen, die Pixelclock, die Synchronisationssignale und die Konfigurationssignale der Kamera an das FPGA weiter (vgl. Tab. A.2). 46
5.3. Der LCD-Touch-Panel ist über ein FFC-Kabel mit der 40P-FFC-Schnittstelle verbunden, dabei ist der LCD-FX2-Adapter an den Nexys2-Board gekoppelt. Die zu Übertragung Signale sind die Pixel-Clock, die RGB-Daten und der Datenberechtigungssignal (vgl. Tabelle A.3). 46
5.4. Erweiterungsadapter die das TE0320-Board aufnimmt und über die FX2-Schnittstelle des Nexys2 mit dem Spartan 3E-1200K koppelt (vgl. Tabelle A.4). 47

5.5. Synchrone Datenübertragung zwischen dem Spartan-3E 1200K und dem Spartan-3A DSP 3400K. Die IOBs werden als Pipelinestufen verwendet, während die mit übertragende DCM_Clk auf beiden FPGA's zum Eintakten der Daten genutzt wird. 48
5.6. Ein Kupferkabel von 18 cm länge wurde zur Bestimmung der Laufzeitpfadverzögerung am Kanal A und B gemessen. Die Verzögerung von 1.6 ns wird anhand der Differenz der Phasenverschiebung am Oszilloskop angezeigt. 49

6.1. Die Eingangsdaten *DATA*,*LVAL* und *FVAL* werden mit Nullen initialisiert. In der doppelten *for*Schleife wird *DATA* mit dem Pixel der x- und y-Koordinate des Bildes zugewiesen (system_init.m, vgl. Anhang B.1). . . 50
6.2. Aus den generierten Stimulisignalen bereiten die Subsysteme die Pixeldaten auf und leiten sie an die 'To Workspace'-Komponente weiter. Die Ergebnisse liegen als Array vor (*VAR_BINARIZE_OUT* und *VAR_EROSION_OUT*) und werden in einem Matlab-Skript ausgewertet. 51
6.3. Anzeige der aufbereiteten Bilder über eine doppelte for-Schleife und der Zuweisung der Ausgangswerte in das 2D-Array (Image_Processing_TEST.m, vgl. Anhang B.3). 51
6.4. Die Auswertungen der beiden Subsysteme 'Adaptive-Binarize' (links) und 'Erosionfilter' (rechts) verifizieren die Funktionalität der System-Generator-Modelle und zeigen eine deutliche Hervorhebung der Fahrspur. . 52
6.5. Die Stimulisignale *(X)* und *(Y)* werden zur Berechnung der korrigierten Pixelkoordinaten an das Subsysteme *Projektive Transformation* übergeben (system_init.m, vgl. Anhang B.1). Die Ergebnisse (*pt_x_out* und *pt_y_out*) werden über die plot-Funktion ausgewertet. 52
6.6. Die y-Achse zeigt den Zählstand der Synchronisationszähler xcount und ycount, während die x-Achse den Bereich der x-Koordinate darstellt. Der negative Bildbereich ist für die Darstellung der projektiven Transformation nicht geeignet. 53
6.7. Durch die projektive Transformation erfolgt ein kontinuierlicher Rücksprung der y-Koordinate in eine vorherige Ordinate. Die negativen Pixelkoordinaten werden für die Darstellung nicht genutzt. 54
6.8. Die Eingangswerte des Adressgenerators werden über Relational-Blöcke auf das Zutreffen im Bildbereich liegenden Pixelkoordinaten überprüft (vgl. Abschnitt 3.2). Die Berechnung der Pixeladresse wird an das Zeilenzwischenspeicher weitergeleitet, wenn das Expression-Block eine eins ausgibt (system_init, vgl. Anhang B.1). 55
6.9. Pixeladresse der Synchronisationszähler und der korrigierten Pixelkoordinaten. 56
6.10. Die über Software berechneten Pixelkoordinaten *(links)* bestätigen die im oberen Bildbereich entstandenen Lücken. Die Ausgangswerte des SysGen-Moduls *(rechts)* verifizieren die Funktionalität der generierten Netzliste. . . 56

7.1. Im unveränderten Bild (links) sind durch die Reflektion des Lichtes Störungen im oberen Bildbereich zu erkennen, während die adaptive Grauwertschwelle (rechts) zu einem sauberen Bild führt und die Fahrspurmarkierung hervorhebt. 57
7.2. Die Fahrspurmarkierungen im grauwertskalierten Bild (links) wurden durch Anwendung eines quadratischen 3x3 Erosions-Faltungskern verschmälert (rechts). 57
7.3. Die Trichterform (rechts) zeigt die perspektivisch korrigierten x-Koordinaten. Im linken Bild erfolgt ein kontinuierlicher Rücksprung der y-Koordinate. 58
7.4. Sobelfilter mit pixelhafter Anordnung und wechselnden Geradenergebnissen (links). Unveränderte Gerade mit dynamischen Schwellwert und Erosionfilter (rechts). 58

B.1. Matlab-Skript zur Initialisierung der gesamten Simulink-Modelle und zur Generierung der Stimulisignale für die Verifikation der Subysteme. 17
B.2. Generierung der Transformationsmatrix B zur Neukalibrierung für die Projektiven Transformation. 18
B.3. Anzeige der aufbereiteten Bilder über eine doppelte for-Schleife und der Zuweisung der Ausgangswerte in das 2D-Array. 19
B.4. Die Zeit t wird für die plot-Funktion genutzt, damit die Adressen der korrigierten Pixelkoordinaten mit der aktuellen Adresse der Synchronzähle ausgewertet werden. 19
B.5. Der Dateibaum zeigt die wichtigsten Dateipfade des XPS-Projektes an und die darin enthaltenen Daten, die zur Erstellung der Bildverarbeitungspipeline genutzt werden. 20
B.6. Der Design Summary zeigt die verbrauchten FPGA-Ressourcen an. Der prozentuale Anteil der verbrauchten Slices liegt bei 98%, daher auch die Kopplung mit einem weiteren FPGA. 21
B.7. 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. 21
B.8. Die UCF verbindet die äußeren Schnittstellen mit der inneren Logik des RTL-Modells. Die lanedetect_DATA_IN_pin ist das Videosignal der CCD-Kamera, während die Signale lanedetect_TBRED_pin, lanedetect_TBGRN_pin und lanedetect_TBBLU_pin an die VGA-Schnittstelle weitergeleitet werden. Die video_0-Pins sind Switches, um verschiedene Eintstellungen der Fahrspurführung und der Filter auszuwählen. 22
B.9. Die RTL-Beschreibung des Anwender-IP-Cores ist in der user_logic enthalten. Die Netzlisten und VHDL-Module werden als Komponenten der user_logic eingebunden und in der Port-Map miteinander verknüpft. . . . 33

12. Tabellenverzeichnis

2.1. Tabelle zum Vergleich der untersuchten Touch-Panel-Datenblätter [Hit06], [Tos07], [Sha07], [Sha02] 13
2.2. Tabelle zur Konfiguration der Datenübertragungsstrecke des SPI-Busses [Tre10]. 15
2.3. Tabelle zur Konfiguration der Komponenten des I2C-Busses [Tre10]. 15

A.1. Pin Verbindung zwischen dem 24P-FFC und 28P-IDC 11
A.2. Pin Verbindung zwischen dem 28P-IDC und PMOD1 & 2 12
A.3. Pin Verbindung zwischen dem 40P-FFC und der FX2-Schnittstelle zur Kopplung des LCD-Touch-Panels 14
A.4. Pin Verbindung zwischen dem Spartan-3A DSP 3400 FPGA-IOs und dem Spartan-3E 1200K. Die Verbindungen vom FPGA-Ball des TE0320 zum FPGA-Ball des Nexys2 gehen über die äußeren Schnittstellen der Evaluationsplattformen. 14

Anhang

A. Tabelle zu den Pin-Connections zwischen den Adapterplatinen und der Hardwareplattform

24P-FFC	28P-IDC	Signalname	Distanz in mm
FFC-IO01	IDC-IO01	Ground	17.2155
FFC-IO02	IDC-IO02	Y0	19.8640
FFC-IO03	IDC-IO03	Y1	15.9871
FFC-IO04	IDC-IO04	Y2	18.6720
FFC-IO05	IDC-IO05	Y3	15.3436
FFC-IO06	IDC-IO06	Y4	18.0507
FFC-IO07	IDC-IO07	Y5	14.7222
FFC-IO08	IDC-IO08	Y6	17.4293
FFC-IO09	IDC-IO09	Y7	14.1009
FFC-IO10	IDC-IO10	Ground	16.8080
FFC-IO11	IDC-IO11	C0	13.4796
FFC-IO12	IDC-IO12	C1	16.1867
FFC-IO13	IDC-IO13	C2	12.8583
FFC-IO14	IDC-IO14	C3	15.5654
FFC-IO15	IDC-IO15	C4	12.2751
FFC-IO16	IDC-IO16	C5	15.4477
FFC-IO17	IDC-IO17	C6	12.8964
FFC-IO18	IDC-IO18	C7	16.0691
FFC-IO19	IDC-IO19	Ground	13.5177
FFC-IO20	IDC-IO20	VSync	16.6904
FFC-IO21	IDC-IO21	HSync	14.1390
FFC-IO22	IDC-IO22	Ground	17.3117
FFC-IO23	IDC-IO23	Pixel Clock	14.7603
FFC-IO24	IDC-IO24	Ground	17.9330

CN701	28P-IDC	Signalname	Distanz in mm
CN701-IO01	IDC-IO25	Power Input	13.0651
CN701-IO02	IDC-IO26	Ground	16.6172
CN701-IO03	IDC-IO27	TX	14.1338
CN701-IO04	IDC-IO28	RX	17.6859

Durchschnittliche Distanz in mm	maximaler Laufzeitpfad in mm	minimaler Laufzeitpfad in mm
15.6723	19.8640	12.8583

Tabelle A.1.: Pin Verbindung zwischen dem 24P-FFC und 28P-IDC

IDC	**PMOD**	**Signalname**
IDC-IO01	PMOD1-IO10	Ground
IDC-IO02	PMOD1-IO01	Y0
IDC-IO03	PMOD1-IO03	Y1
IDC-IO04	PMOD1-IO05	Y2
IDC-IO05	PMOD1-IO02	Y3
IDC-IO06	PMOD1-IO07	Y4
IDC-IO07	PMOD1-IO04	Y5
IDC-IO08	PMOD1-IO06	Y6
IDC-IO09	PMOD1-IO08	Y7
IDC-IO10	PMOD1-IO09	Ground
IDC-IO16	PMOD2-IO04	C5
IDC-IO17	PMOD2-IO06	C6
IDC-IO18	PMOD2-IO02	C7
IDC-IO19	PMOD1-IO10	Ground
IDC-IO20	PMOD2-IO08	VSync
IDC-IO21	PMOD2-IO07	HSync
IDC-IO22	PMOD1-IO09	Ground
IDC-IO23	PMOD2-IO01	Pixel Clock
IDC-IO24	PMOD1-IO09	Ground
IDC-IO25	-	Power Input (RC-Fahrzeug)
IDC-IO26	PMOD1-IO09	Ground
IDC-IO27	PMOD2-IO05	TX
IDC-IO28	PMOD2-IO03	RX

Tabelle A.2.: Pin Verbindung zwischen dem 28P-IDC und PMOD1 & 2

40P-FFC	**FX2-Connector**	**Signalname**	**Distanz in mm**
40P-FFC-IO01	VCC3V3	Power Supply	52.7841
40P-FFC-IO02	VCC3V3	Power Supply	52.7841
40P-FFC-IO03	VCC3V3	Power Supply	52.7841
40P-FFC-IO04	VCC3V3	Power Supply	52.7841
40P-FFC-IO05	-	No Connection	-
40P-FFC-IO06	FX2-IO40	DTMG (Timing Signal)	32.6945
40P-FFC-IO07	FX2-GND	Ground	195.0051
40P-FFC-IO08	FX2-IO39	DCLK	30.4221
40P-FFC-IO09	FX2-GND	Ground	195.0051
40P-FFC-IO10	-	No Connection	-
40P-FFC-IO11	FX2-GND	Ground	195.0051
40P-FFC-IO12	FX2-IO38	B5	32.7445
40P-FFC-IO13	FX2-IO37	B4	30.4055
40P-FFC-IO14	FX2-IO36	B3	31.9916
40P-FFC-IO15	FX2-GND	Ground	195.0051
40P-FFC-IO16	FX2-IO35	B2	29.9747
40P-FFC-IO17	FX2-IO34	B1	31.5608
40P-FFC-IO18	FX2-IO33	B0	29.3368
40P-FFC-IO19	FX2-GND	Ground	195.0051
40P-FFC-IO20	FX2-IO32	G5	31.1316
40P-FFC-IO21	FX2-IO31	G4	28.9061
40P-FFC-IO22	FX2-IO30	G3	30.4921
40P-FFC-IO23	FX2-GND	Ground	195.0051
40P-FFC-IO24	FX2-IO29	G2	28.4753
40P-FFC-IO25	FX2-IO28	G1	30.0630
40P-FFC-IO26	FX2-IO27	G0	27.8374
40P-FFC-IO27	FX2-GND	Ground	195.0051
40P-FFC-IO28	FX2-IO26	R5	29.6322
40P-FFC-IO29	FX2-IO25	R4	28.1942
40P-FFC-IO30	FX2-IO24	R3	28.9943
40P-FFC-IO31	FX2-GND	Ground	195.0051
40P-FFC-IO32	FX2-IO23	R2	26.9758
40P-FFC-IO33	FX2-IO22	R1	28.5635
40P-FFC-IO34	FX2-IO21	R0	26.3396
40P-FFC-IO35	-	TEST	-
40P-FFC-IO36	FX2-GND	GND	-
40P-FFC-IO37	-	XT (Analog Top)	-
40P-FFC-IO38	-	YL (Analog Left)	-
40P-FFC-IO39	-	XB (Analog Bottom)	-
40P-FFC-IO40	-	YR (Analog Right)	-

BHR-03VS-1	FX2-Connector	Signalname	Distanz in mm
BHR-01	VCC3V3	Power Supply (CFL)	65.9539
BHR-02	-	No Connection	-
BHR-03	GND	Ground	195.0051

Tabelle A.3.: Pin Verbindung zwischen dem 40P-FFC und der FX2-Schnittstelle zur Kopplung des LCD-Touch-Panels

Tabelle A.4.: Pin Verbindung zwischen dem Spartan-3A DSP 3400 FPGA-IOs und dem Spartan-3E 1200K. Die Verbindungen vom FPGA-Ball des TE0320 zum FPGA-Ball des Nexys2 gehen über die äußeren Schnittstellen der Evaluationsplattformen.

	B2B(TE0320)				FX2(Nexys2)		
Pin Nr.	FPGA-Pin	FPGA-Ball	JM5-Signal	FX2-Signal	FPGA-Ball	FPGA-Pin	Pin Nr.
35	IO_L27P_2,GCLK0	Y14	J5_IO01	FX2-IO1	B4	IO_L24N_0	6
39	IO_L28P_2,GCLK2	AF14	J5_IO02	FX2-IO2	A4	IO_L24P_0	7
41	IO_L29N_2	AC14	J5_IO03	FX2-IO3	C3	IO_L25P_0	8
43	IO_L39N_2	AE20	J5_IO04	FX2-IO4	C4	IO	9
45	IO_L39P_2	AF20	J5_IO05	FX2-IO5	B6	IO_L20P_0	10
47	IO_L40N_2	AC19	J5_IO06	FX2-IO6	D5	IO_L23N_0/VREF_0	11
49	IO_L40P_2	AD19	J5_IO07	FX2-IO7	C5	IO_L23P_0	12
53	IO_L41N_2	AC20	J5_IO08	FX2-IO8	F7	IO_L19P_0	13
55	IO_L41P_2	AD20	J5_IO09	FX2-IO9	E7	IO_L19N_0/VREF_0	14
57	IO_L42N_2	U16	J5_IO10	FX2-IO10	A6	IO_L20N_0	15
59	IO_L42P_2	V16	J5_IO11	FX2-IO11	C7	IO_L18P_0	16
61	IO_L43N_2	Y17	J5_IO12	FX2-IO12	F8	IO_L17N_0	17
63	IO_L43P_2	AA17	J5_IO13	FX2-IO13	D7	IO_L18N_0/VREF_0	18
67	IO_L44N_2	AD21	J5_IO14	FX2-IO14	E8	IO_L17P_0	19
69	IO_L44P_2	AE21	J5_IO15	FX2-IO15	E9	IO_L15P_0	20
71	IO_L45N_2	AC21	J5_IO16	FX2-IO16	C9	IO_L14P_0/GCLK10	21
73	IO_L45P_2	AD22	J5_IO17	FX2-IO17	A8	IO	22
75	IO_L46N_2	V17	J5_IO18	FX2-IO18	G9	IO	23
77	IO_L46P_2	W17	J5_IO19	FX2-IO19	F9	IO_L15N_0	24
34	IO_L47N_2	AA18	J5_IO20	FX2-IO20	D10	IO_L11P_0/GCLK4	25
36	IO_L47P_2	AB18	J5_IO21	FX2-IO21	A10	IO_L12N_0/GCLK7	26
38	IO_L48N_2	AE23	J5_IO22	FX2-IO22	B10	IO_L12P_0/GCLK6	27
40	IO_L48P_2	AF23	J5_IO23	FX2-IO23	A11	IO	28
42	IO_L51N_2	AE25	J5_IO24	FX2-IO24	D11	IO_L09N_0	29
46	IO_L51P_2	AF25	J5_IO25	FX2-IO25	E10	IO_L11N_0/GCLK5	30
48	IO_L05N_2	Y9	J5_IO26	FX2-IO26	B11	IO/VREF_0	31
50	IO_L05P_2	W9	J5_IO27	FX2-IO27	C11	IO_L09P_0	32
52	IO_L06N_2	AF3	J5_IO28	FX2-IO28	E11	IO_L08P_0	33
54	IO_L06P_2	AE3	J5_IO29	FX2-IO29	F11	IO_L08N_0	34
56	IO_L07N_2	AF4	J5_IO30	FX2-IO30	E12	IO_L06N_0	35
60	IO_L07P_2	AE4	J5_IO31	FX2-IO31	F12	IO_L06P_0	36
62	IO_L08N_2	AD6	J5_IO32	FX2-IO32	A13	IO_L05P_0	37
64	IO_L08P_2	AC6	J5_IO33	FX2-IO33	B13	IO_L05N_0/VREF_0	38
66	IO_L09N_2	W10	J5_IO34	FX2-IO34	E13	IO	39
68	IO_L09P_2	V10	J5_IO35	FX2-IO35	A14	IO_L04N_0	40
70	IO_L25N_2,GCLK13	Y13	J5_IO36	FX2-IO36	C14	IO_L03N_0/VREF_0	41
74	IO_L25P_2,GCLK12	AA13	J5_IO37	FX2-IO37	D14	IO_L03P_0	42
76	IO_L26N_2,GCLK15	AE13	J5_IO38	FX2-IO38	B14	IO_L04P_0	43
78	IO_L26P_2,GCLK14	AF13	J5_IO39	FX2-IO39	A16	IO_L01N_0	44
80	IO_L20N_2	W13	J5_IO40	FX2-IO40	B16	IO_L01P_0	45

Links

Rechts

B. EDK-Projekt, VHDL-Code und Matlab-Skript der Bildverarbeitungskomponenten

B.1. Matlab-Skript zur Initialisierung der gesamten Simulink-Modelle

```
clear all;
clc;

RGB = imread('./Beispiel/Fahrspur.jpg');

I = .2989*RGB(:,:,1)...
    +.5870*RGB(:,:,2)...
    +.1140*RGB(:,:,3);

imshow(I);

%GENERICS
ts = 1;
size_y = 240;
size_x = 640;
roi_size_x = 50;
roi_size_y = 50;

%PT INIT
B = [ %640 * 240
    0.4296    4.5595  183.9132;
   -0.0082    3.6293  -96.7422;
   -0.0000    0.0141    1.0000
];

x_area = zeros(size_x*size_y, 1);
y_area = zeros(size_x*size_y, 1);
divisor_area = zeros(size_x*size_y, 1);
div_erg_matlab = zeros(size_x*size_y, 1);
x_erg = zeros(size_x*size_y, 1);
y_erg = zeros(size_x*size_y, 1);
k = 1;

for y=1:size_y
    for x=1:size_x
        x_area(k) = B(1,1)*x+B(1,2)*y+B(1,3);
        y_area(k) = B(2,1)*x+B(2,2)*y +B(2,3);
        divisor_area(k) = B(3,1)*x+B(3,2)*y + B(3,3);
        div_erg_matlab(k) = 1./divisor_area(k);
        x_erg(k) = x_area(k).*div_erg_matlab(k);
        y_erg(k) = (y_area(k).*div_erg_matlab(k));
        k = k + 1;
    end;
end;
```

```
max_dividend = max(max(x_area), max(y_area));
latency = 13;
sim_length = (latency+(size_y*size_x))*ts;
fractionals = 12;
divider_fractionals = 8;

b11 = 1+ ceil(log2(B(1,1) + 1))+ fractionals;
b12 = 1+ ceil(log2(B(1,2) + 1))+ fractionals;
b13 = 1+ ceil(log2(B(1,3) + 1))+ fractionals;
b21 = 1+ ceil(log2(B(2,1) + 1))+ fractionals;
b22 = 1+ ceil(log2(B(2,2) + 1))+ fractionals;
b23 = 1+ ceil(log2(abs(B(2,3)) + 1))+ fractionals;
b32 = 1+ ceil(log2(B(3,2) + 1))+ fractionals;

b11x = 1+ ceil(log2(B(1,1)*size_x + 1))+ fractionals;
b12y = 1+ ceil(log2(B(1,2)*size_y + 1))+ fractionals;
b11x_b12y = 1+ ceil(log2(B(1,1)*size_x + B(1,2)*size_y + 1))+↙
fractionals;
b11x_b12y_b13 = 1+ ceil(log2(B(1,1)*size_x + B(1,2)*size_y + B(1,3) +↙
1))+ fractionals;

b21x = 1+ ceil(log2(abs(B(2,1)*size_x) + 1))+ fractionals;
b22y = 1+ ceil(log2(abs(B(2,2)*size_y) + 1))+ fractionals;
b21x_b22y = 1+ ceil(log2(abs(B(2,1)*size_x + B(2,2)*size_y) + 1))+↙
fractionals;
b21x_b22y_b23 = 1+ ceil(log2(abs(B(2,1)*size_x + B(2,2)*size_y + B↙
(2,3)) + 1))+ fractionals;

b32y = 1+ ceil(log2(B(3,2)*size_y + 1))+ fractionals;
b32y_b33 = 1+ ceil(log2(B(3,2)*size_y + B(3,3)+ 1))+ fractionals;

%HT INIT
phi = -18:2:18;

% -- Ermitteln der Tiefe des RAMs -- %
rmax = ceil(cosd(max(phi)) * roi_size_x + sind(max(phi)) * roi_size_y);
rmin = floor(cosd(min(phi))* 0 + sind(min(phi)) * roi_size_y);
rmin = - rmin;

% -- Adressdekoder Initialisierungen -- %
addr_offset = 1; % wenn keine Zweierpotenz, wird nächstgrößere↙
verwendet
```

```
%Pathfollowing init

phi_table = phi * pi / 180;

L = 25.7;
LAD = 80;

dt = 1/60;
P = 0.0708815295807401*6;
D = 0.228314514646616/2;
N = 112.632956895447;
a = (1-N*dt/2);
b = 1/(1+N*dt/2);

C2 = (P + D*N*b);
C1 = (P*a + D*N)*b;
C3 = a*b;
C4 = 1/100;
% STIMULI INIT
X = zeros(size_x*size_y,2);
Y = zeros(size_x*size_y,2);
DATA = zeros(size_x*size_y,2);

t = 1;

for y = 1:size_y
    for x = 1:size_x
        X(t,1) = t;
        X(t,2) = x;
        Y(t,1) = t;
        Y(t,2) = y;
        DATA(t,1) = t;
        DATA(t,2) = I(y,x);
        t = t + 1;
    end
end

sim_length = t;
```

Abbildung B.1.: Matlab-Skript zur Initialisierung der gesamten Simulink-Modelle und zur Generierung der Stimulisignale für die Verifikation der Subysteme.

B.2. Matlab-Skript zur Neukalibrierung der Transformationsmatrix B und die Auswertung der projektiven Transformation

```
clear all;

X_STERN = [90  90  125 125 125 125 125 160 160 160 160 160 195 195 195↙
195 195 230 230 230];
Y_STERN = [10  45  10  45  80  115 150 10  45  80  115 150 10  45  80↙
115 150 10  45  80 ];
X       = [40  20  96  87  70  48  11  155 154 152 149 146 214 223 235↙
253 281 271 288 314];
Y       = [33  47  31  47  71  106 162 30  47  70  106 164 33  49  72↙
107 164 30  50  74 ];


L_X = [X(:) Y(:) ones(length(X),1) zeros(length(X),3) -X_STERN(:).*X(:)↙
-X_STERN(:).*Y(:)];
L_Y = [zeros(length(X),3) X(:) Y(:) ones(length(X),1) -Y_STERN(:).*X(:)↙
-Y_STERN(:).*Y(:)];
L_GES = [L_X; L_Y];
RES_GES = [X_STERN'; Y_STERN'];

b = L_GES\RES_GES;

B = zeros(3,3);
B(1,1) = b(1);
B(1,2) = b(2);
B(1,3) = b(3);
B(2,1) = b(4);
B(2,2) = b(5);
B(2,3) = b(6);
B(3,1) = b(7);
B(3,2) = b(8);
B(3,3) = 1


X_CTRL = B(1,1).*X+B(1,2).*Y+B(1,3)-B(3,1).*X_STERN.*X-B(3,2).*X_STERN.↙
*Y;
Y_CTRL = B(2,1).*X+B(2,2).*Y+B(2,3)-B(3,1).*Y_STERN.*X-B(3,2).*Y_STERN.↙
*Y;

X_CTRL2 = (B(1,1).*X+B(1,2).*Y+B(1,3))./(B(3,1).*X+B(3,2).*Y+B(3,3));
Y_CTRL2 = (B(2,1).*X+B(2,2).*Y+B(2,3))./(B(3,1).*X+B(3,2).*Y+B(3,3));
```

Abbildung B.2.: Generierung der Transformationsmatrix B zur Neukalibrierung für die Projektiven Transformation.

```
IP_BIN = zeros(size_y,size_x); %2D-Binarize Array
IP_ERO = zeros(size_y,size_x); %2D-Erosionfilter Array
i = 1;
for Y=1:size_y
    for X=1:size_x
            IP_BIN(Y,X) = VAL_BINARIZE_OUT(i); %Zuweisung der Ausgangswerte
            IP_ERO(Y,X) = VAL_EROSION_OUT(i);
            i = i+1;
    end;
end;
figure(1);
imshow(IP_BIN); %Ansicht der zugewiesenen Werte
figure(2);
imshow(IP_ERO);
```

Abbildung B.3.: Anzeige der aufbereiteten Bilder über eine doppelte for-Schleife und der Zuweisung der Ausgangswerte in das 2D-Array.

B.3. Initialisierung der Zeit t zur Auswertung der generierten Adressen des Adressgenerators

```
ts = zeros(1, size_x*size_y)';

t = 1;
m = 1;

for y=1:size_y
    for x=1:size_x
        ts(t) = t;
        t = t + 1;
    end;
end;
```

Abbildung B.4.: Die Zeit t wird für die plot-Funktion genutzt, damit die Adressen der korrigierten Pixelkoordinaten mit der aktuellen Adresse der Synchronzähle ausgewertet werden.

B.4. Das System EDK Projekt der Bildverarbeitungspipeline mit integriertem Framegrabber

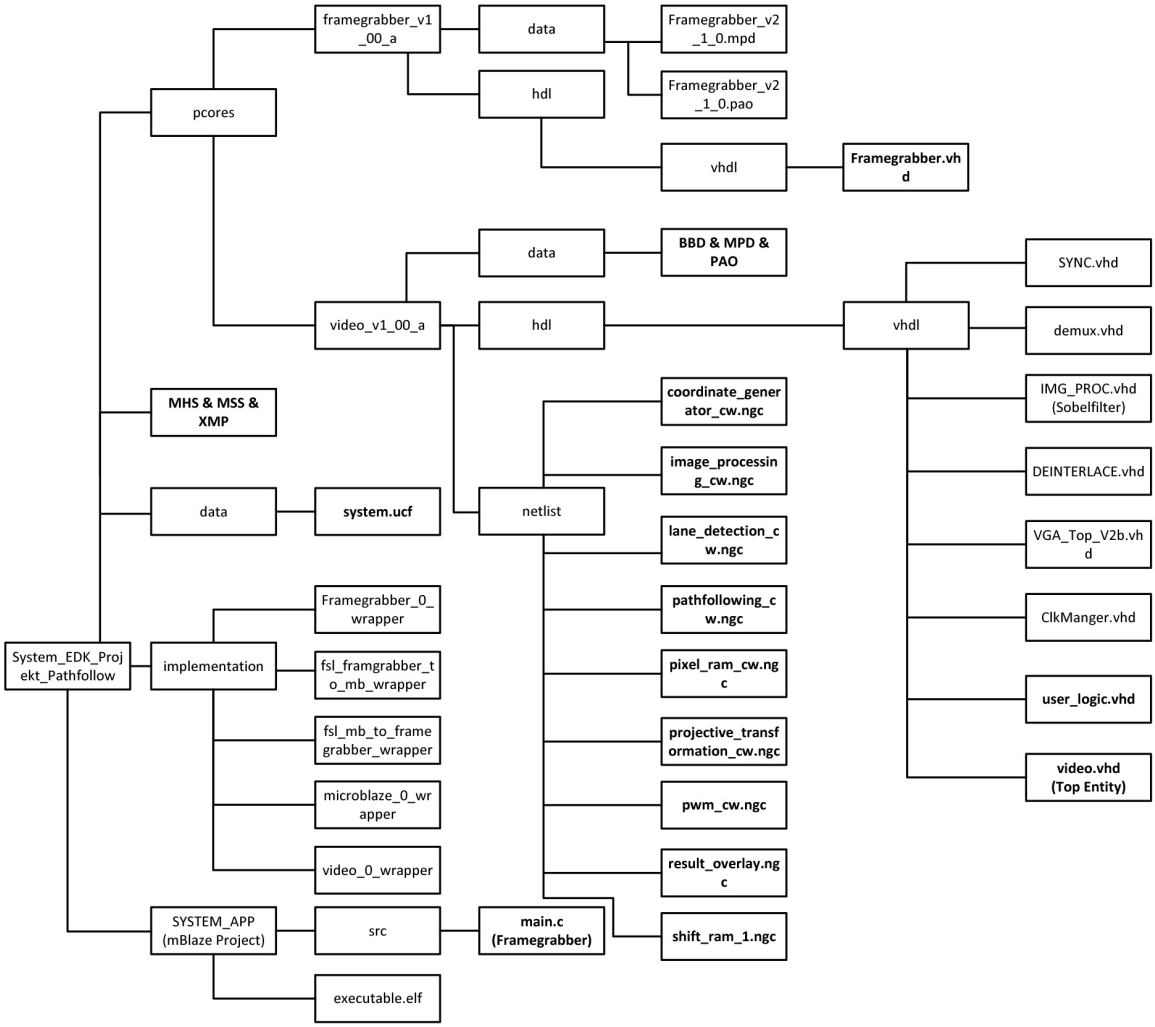

Abbildung B.5.: Der Dateibaum zeigt die wichtigsten Dateipfade des XPS-Projektes an und die darin enthaltenen Daten, die zur Erstellung der Bildverarbeitungspipeline genutzt werden.

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			

Abbildung B.6.: Der Design Summary zeigt die verbrauchten FPGA-Ressourcen an. Der prozentuale Anteil der verbrauchten Slices liegt bei 98%, daher auch die Kopplung mit einem weiteren FPGA.

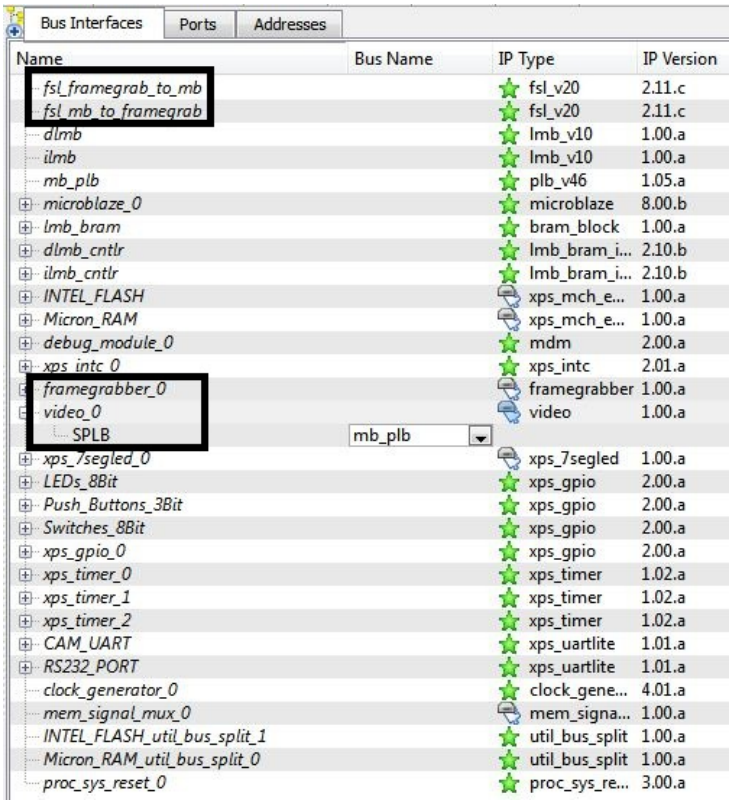

Abbildung B.7.: Die IP-Cores des EDK-Projektes werden im System Assembly View angezeigt und deren Verknüpfungen über den mb_plb-Bus, dabei sind Framegrabber und Video Anwender-IP-Cores. Der System Assembly View ist an die MHS-Datei angebunden.

```
28   Net CAM_UART_TX_pin LOC=M18;# | IOSTANDARD=LVTTL;
29   Net CAM_UART_RX_pin LOC=N18;# | IOSTANDARD=LVTTL;
30
31   NET "lanedetect_CLK_pin" LOC= "J13" | CLOCK_DEDICATED_ROUTE=FALSE;
32
33   NET "lanedetect_DATA_IN_pin<0>" LOC = "G15";
34   NET "lanedetect_DATA_IN_pin<1>" LOC = "H15";
35   NET "lanedetect_DATA_IN_pin<2>" LOC = "J16";
36   NET "lanedetect_DATA_IN_pin<3>" LOC = "F14";
37   NET "lanedetect_DATA_IN_pin<4>" LOC = "G13";
38   NET "lanedetect_DATA_IN_pin<5>" LOC = "G16";
39   NET "lanedetect_DATA_IN_pin<6>" LOC = "H16";
40   NET "lanedetect_DATA_IN_pin<7>" LOC = "J12";
41
42   # Pin assignment for VGA demo
43   # Connected to Nexys 2
44    NET "lanedetect_TBRED_pin<0>" LOC= "R9";
45    NET "lanedetect_TBRED_pin<1>" LOC= "T8";
46    NET "lanedetect_TBRED_pin<2>" LOC= "R8";
47    NET "lanedetect_TBGRN_pin<0>" LOC= "N8";
48    NET "lanedetect_TBGRN_pin<1>" LOC= "P8";
49    NET "lanedetect_TBGRN_pin<2>" LOC= "P6";
50    NET "lanedetect_TBBLU_pin<0>" LOC= "U5";
51    NET "lanedetect_TBBLU_pin<1>" LOC= "U4";
52
53    NET "lanedetect_TBHSYNC_pin" LOC= "T4";
54    NET "lanedetect_TBVSYNC_pin" LOC= "U3";
55
56    #### Module Switches_8Bit constraints
57    NET "lanedetect_OUTPUT_SWITCH_pin" LOC = "N17";
58    NET "video_0_SWITCH_ROI_pin" LOC = "L13";
59    NET "video_0_PD_SWITCH_pin" LOC = "G18";
60    NET "video_0_PT_SWITCH_pin" LOC = "L14";
61    NET "video_0_FTC_PP_SWITCH_pin" LOC = "H18";
62    NET "video_0_FILTER_SWITCH_pin" LOC = "K17";
63    NET "video_0_PATHFOLLOWING_SWITCH_pin" LOC = "K18";
64
65    #### PMOD Pin
66    Net "video_0_PWM_SIGNAL_pin" LOC = "L15";
```

Abbildung B.8.: Die UCF verbindet die äußeren Schnittstellen mit der inneren Logik des RTL-Modells. Die lanedetect_DATA_IN_pin ist das Videosignal der CCD-Kamera, während die Signale lanedetect_TBRED_pin, lanedetect_TBGRN_pin und lanedetect_TBBLU_pin an die VGA-Schnittstelle weitergeleitet werden. Die video_0-Pins sind Switches, um verschiedene Eintstellungen der Fahrspurführung und der Filter auszuwählen.

B.5. VHDL Code der User-Logic des aktuellen EDK-Projektes

```
86   entity user_logic is
87     generic
88     (
89       -- ADD USER GENERICS BELOW THIS LINE ---------------
90       --USER generics added here
91       -- ADD USER GENERICS ABOVE THIS LINE ---------------
92
93       -- DO NOT EDIT BELOW THIS LINE ----------------------
94       -- Bus protocol parameters, do not add to or delete
95       C_SLV_DWIDTH                    : integer              := 32;
96       C_NUM_REG                       : integer              := 16;
97       C_NUM_INTR                      : integer              := 1
98       -- DO NOT EDIT ABOVE THIS LINE ----------------------
99     );
100    port
101    (
102      -- ADD USER PORTS BELOW THIS LINE ------------------
103      --USER ports added here
104     CLK: in std_logic;
105     t_rst: in std_logic;
106     DATA_IN : in std_logic_vector(7 downto 0);
107     OUTPUT_SWITCH : in std_logic;
108
109     ROI_SWITCH: in std_logic;
110     PT_SWITCH: in std_logic;
111     FILTER_SWITCH: in std_logic;
112
113     FTC_PP_SWITCH: in std_logic;
114     PD_SWITCH: in std_logic;
115     PATHFOLLOWING_SWITCH: in std_logic;
116     PWM_SIGNAL: out std_logic;
117
118     TBHSYNC: out std_logic;
119     TBVSYNC: out std_logic;
120
121     FG_LVAL : out std_logic;
```

```
122       FG_FVAL : out std_logic;
123       FG_CLK  : out std_logic;
124       FG_DATA : out std_logic_vector(7 downto 0);
125
126       TBRED: out std_logic_vector(2 downto 0);
127       TBGRN: out std_logic_vector(2 downto 0);
128       TBBLU: out std_logic_vector(1 downto 0);
129
130
131        -- ADD USER PORTS ABOVE THIS LINE ------------------
132
133        -- DO NOT EDIT BELOW THIS LINE ---------------------
134        -- Bus protocol ports, do not add to or delete
135        Bus2IP_Clk                     : in  std_logic;
136        Bus2IP_Reset                   : in  std_logic;
137        Bus2IP_Data                    : in  std_logic_vector(0 to C_SLV_DWIDTH-1);
138        Bus2IP_BE                      : in  std_logic_vector(0 to C_SLV_DWIDTH/8-1);
139        Bus2IP_RdCE                    : in  std_logic_vector(0 to C_NUM_REG-1);
140        Bus2IP_WrCE                    : in  std_logic_vector(0 to C_NUM_REG-1);
141        IP2Bus_Data                    : out std_logic_vector(0 to C_SLV_DWIDTH-1);
142        IP2Bus_RdAck                   : out std_logic;
143        IP2Bus_WrAck                   : out std_logic;
144        IP2Bus_Error                   : out std_logic;
145        IP2Bus_IntrEvent               : out std_logic_vector(0 to C_NUM_INTR-1)
146        -- DO NOT EDIT ABOVE THIS LINE ---------------------
147      );
148
149      attribute SIGIS : string;
150      attribute SIGIS of Bus2IP_Clk    : signal is "CLK";
151      attribute SIGIS of Bus2IP_Reset  : signal is "RST";
152
153    end entity user_logic;
154
155    ------------------------------------------------------------------------------
156    -- Architecture section
157    ------------------------------------------------------------------------------
158
159    architecture IMP of user_logic is
160
161       --USER signal declarations added here, as needed for user logic
162       signal t_clk:  std_logic;
163       signal t_clk2: std_logic;
164       signal t_clk3: std_logic;
165
166       signal sync_lval: std_logic;
167       signal sync_fval: std_logic;
168       signal sync_data: std_logic_vector (7 downto 0);
169
170       signal demux_lval:   std_logic;
171       signal demux_fval:   std_logic;
172       signal demux_data:      std_logic_vector (7 downto 0);
173
174       signal sobel_lval:   std_logic;
175       signal sobel_fval:   std_logic;
176       signal sobel_data:   std_logic;
177
178       signal imgproc_data:    std_logic;
179       signal imgproc_fval:    std_logic;
180       signal imgproc_lval:    std_logic;
181
182       signal imgproc_mux_data:   std_logic;
```

```
183     signal imgproc_mux_fval:   std_logic;
184     signal imgproc_mux_lval:   std_logic;
185
186     signal coordgen_data:   std_logic;
187     signal coordgen_fval:   std_logic;
188     signal coordgen_lval:   std_logic;
189     signal coordgen_xcount:    std_logic_vector (9 downto 0);
190     signal coordgen_ycount:    std_logic_vector (7 downto 0);
191
192     signal pt_data:   std_logic;
193     signal pt_fval:   std_logic;
194     signal pt_lval:   std_logic;
195     signal pt_xcount:    std_logic_vector (9 downto 0);
196     signal pt_xpt:    std_logic_vector (10 downto 0);
197     signal pt_ycount:    std_logic_vector (7 downto 0);
198     signal pt_ypt:    std_logic_vector (10 downto 0);
199
200     signal pram_data:    std_logic;
201     signal pram_fval:    std_logic;
202     signal pram_lval:    std_logic;
203     signal pram_xcount:  std_logic_vector (9 downto 0);
204     signal pram_ycount:  std_logic_vector (7 downto 0);
205
206     signal pt_mux_data:  std_logic;
207     signal pt_mux_fval:  std_logic;
208     signal pt_mux_lval:  std_logic;
209     signal pt_mux_xcount:   std_logic_vector (9 downto 0);
210     signal pt_mux_ycount:   std_logic_vector (7 downto 0);
211
212     signal disp_data:    std_logic_vector (23 downto 0);
213     signal disp_fval:    std_logic;
214     signal disp_lval:    std_logic;
215
216     signal lanedet_done:    std_logic;
217     signal lanedet_max_val:    std_logic_vector (9 downto 0);
218     signal lanedet_mroi_x:  std_logic_vector (11 downto 0);
219     signal lanedet_mroi_y:  std_logic_vector (11 downto 0);
220     signal lanedet_phi_id:  std_logic_vector (4 downto 0);
221     signal lanedet_r:    std_logic_vector (7 downto 0);
222     signal lanedet_roi_x:   std_logic_vector (9 downto 0);
223     signal lanedet_roi_y:   std_logic_vector (7 downto 0);
224
225     signal pathfollow_alpha:   std_logic_vector (17 downto 0);
226     signal pathfollow_done:    std_logic;
227
228     signal pram_lanedet_roi_x: std_logic_vector(11 downto 0);
229     signal pram_lanedet_roi_y: std_logic_vector(11 downto 0);
230
231     signal switch_out_lval: std_logic;
232     signal switch_out_fval: std_logic;
233     signal switch_out_r:    std_logic_vector (7 downto 0);
234     signal switch_out_g:    std_logic_vector (7 downto 0);
235     signal switch_out_b:    std_logic_vector (7 downto 0);
236
237     signal t_lval_deint:    std_logic;
238     signal t_fval_deint:    std_logic;
239     signal t_r_deint:       std_logic_vector (7 downto 0);
240     signal t_g_deint:       std_logic_vector (7 downto 0);
241     signal t_b_deint:       std_logic_vector (7 downto 0);
242
243     signal HSYNC, VSYNC, BLANK:                 std_logic;
```

```
244     signal RED_OUT_TB, GRN_OUT_TB, BLU_OUT_TB :  std_logic_vector(7 downto 0);
245     signal DATA_IN_BUF:                          std_logic;
246
247     COMPONENT image_processing_cw is
248       port (
249         ce: in std_logic := '1';
250         clk: in std_logic; -- clock period = 74.0740740740741 ns (13.499999999999995 Mhz)
251         imgproc_fval_in: in std_logic;
252         imgproc_lval_in: in std_logic;
253         imgproc_y_in: in std_logic_vector(7 downto 0);
254         imgproc_data_out: out std_logic;
255         imgproc_fval_out: out std_logic;
256         imgproc_lval_out: out std_logic
257       );
258     end COMPONENT;
259
260     COMPONENT coordinate_generator_cw is
261     port (
262       ce: in std_logic := '1';
263       clk: in std_logic; -- clock period = 74.0740740740741 ns (13.499999999999995 Mhz)
264       coordgen_fval_in: in std_logic;
265       coordgen_lval_in: in std_logic;
266       coordgen_y_in: in std_logic;
267       coordgen_data_out: out std_logic;
268       coordgen_fval_out: out std_logic;
269       coordgen_lval_out: out std_logic;
270       coordgen_xcount_out: out std_logic_vector(9 downto 0);
271       coordgen_ycount_out: out std_logic_vector(7 downto 0)
272     );
273   end COMPONENT;
274
275   COMPONENT projective_transformation_cw is
276     port (
277       ce: in std_logic := '1';
278       clk: in std_logic; -- clock period = 18.5185185185185 ns (54.00000000000005 Mhz)
279       pt_data_in: in std_logic;
280       pt_fval_in: in std_logic;
281       pt_lval_in: in std_logic;
282       pt_xcount_in: in std_logic_vector(9 downto 0);
283       pt_ycount_in: in std_logic_vector(7 downto 0);
284       pt_data_out: out std_logic;
285       pt_fval_out: out std_logic;
286       pt_lval_out: out std_logic;
287       pt_xcount_out: out std_logic_vector(9 downto 0);
288       pt_xpt_out: out std_logic_vector(10 downto 0);
289       pt_ycount_out: out std_logic_vector(7 downto 0);
290       pt_ypt_out: out std_logic_vector(10 downto 0)
291     );
292   end COMPONENT;
293
294   --COMPONENT pixel_ram_cw is
295   --  port (
296   --    ce: in std_logic := '1';
297   --    clk: in std_logic; -- clock period = 74.0740740740741 ns (13.499999999999995 Mhz)
298   --    pram_data_in: in std_logic;
299   --    pram_fval_in: in std_logic;
300   --    pram_lval_in: in std_logic;
301   --    pram_xcount_in: in std_logic_vector(9 downto 0);
302   --    pram_xpt_in: in std_logic_vector(10 downto 0);
303   --    pram_ycount_in: in std_logic_vector(7 downto 0);
304   --    pram_ypt_in: in std_logic_vector(10 downto 0);
```

```
305   --    pram_data_out: out std_logic;
306   --    pram_fval_out: out std_logic;
307   --    pram_lval_out: out std_logic;
308   --    pram_xcount_out: out std_logic_vector(9 downto 0);
309   --    pram_ycount_out: out std_logic_vector(7 downto 0)
310   --  );
311   --end COMPONENT;
312
313   COMPONENT pixel_ram_cw is
314     port (
315       ce: in std_logic := '1';
316       clk: in std_logic; -- clock period = 74.0740740740741 ns (13.499999999999995 Mhz)
317       pram_data_in: in std_logic;
318       pram_fval_in: in std_logic;
319       pram_lval_in: in std_logic;
320       pram_roi_x_in: in std_logic_vector(11 downto 0);
321       pram_roi_y_in: in std_logic_vector(11 downto 0);
322       pram_xcount_in: in std_logic_vector(9 downto 0);
323       pram_xpt_in: in std_logic_vector(10 downto 0);
324       pram_ycount_in: in std_logic_vector(7 downto 0);
325       pram_ypt_in: in std_logic_vector(10 downto 0);
326       pram_data_out: out std_logic;
327       pram_fval_out: out std_logic;
328       pram_lval_out: out std_logic;
329       pram_xcount_out: out std_logic_vector(9 downto 0);
330       pram_ycount_out: out std_logic_vector(7 downto 0)
331     );
332   end COMPONENT;
333
334      COMPONENT result_overlay_cw is
335     port (
336       ce: in std_logic := '1';
337       clk: in std_logic; -- clock period = 37.037037037037 ns (27.000000000000025 Mhz)
338       disp_data_in: in std_logic;
339       disp_fval_in: in std_logic;
340       disp_lval_in: in std_logic;
341       disp_phi_id_in: in std_logic_vector(4 downto 0);
342       disp_r_in: in std_logic_vector(7 downto 0);
343       disp_roi_switch_in: in std_logic;
344       disp_roi_x_in: in std_logic_vector(9 downto 0);
345       disp_roi_y_in: in std_logic_vector(7 downto 0);
346       disp_x_in: in std_logic_vector(9 downto 0);
347       disp_y_in: in std_logic_vector(7 downto 0);
348       disp_data_out: out std_logic_vector(23 downto 0);
349       disp_fval_out: out std_logic;
350       disp_lval_out: out std_logic
351     );
352   end COMPONENT;
353
354   COMPONENT lane_detection_cw is
355     port (
356       ce: in std_logic := '1';
357       clk: in std_logic; -- clock period = 37.037037037037 ns (27.000000000000025 Mhz)
358       lanedet_data_in: in std_logic;
359       lanedet_xpt_in: in std_logic_vector(10 downto 0);
360       lanedet_ypt_in: in std_logic_vector(10 downto 0);
361       lanedet_done_out: out std_logic;
362       lanedet_max_val_out: out std_logic_vector(9 downto 0);
363       lanedet_mroi_x_out: out std_logic_vector(11 downto 0);
364       lanedet_mroi_y_out: out std_logic_vector(11 downto 0);
365       lanedet_phi_id_out: out std_logic_vector(4 downto 0);
```

```
366       lanedet_r_out: out std_logic_vector(7 downto 0);
367       lanedet_roi_x_out: out std_logic_vector(9 downto 0);
368       lanedet_roi_y_out: out std_logic_vector(7 downto 0)
369     );
370   end COMPONENT;
371
372   COMPONENT pathfollowing_cw is
373     port (
374       ce: in std_logic := '1';
375       clk: in std_logic; -- clock period = 18.5185185185185 ns (54.00000000000005 Mhz)
376       pathfollow_d_in: in std_logic_vector(9 downto 0);
377       pathfollow_max_val_in: in std_logic_vector(9 downto 0);
378       pathfollow_pd_switch_in: in std_logic;
379       pathfollow_phi_id_in: in std_logic_vector(4 downto 0);
380       pathfollow_pp_switch_in: in std_logic;
381       pathfollow_r_in: in std_logic_vector(7 downto 0);
382       pathfollow_roi_x_in: in std_logic_vector(11 downto 0);
383       pathfollow_roi_y_in: in std_logic_vector(11 downto 0);
384       pathfollow_start_in: in std_logic;
385       pathfollow_alpha_out: out std_logic_vector(17 downto 0);
386       pathfollow_done_out: out std_logic
387     );
388   end COMPONENT;
389
390   COMPONENT pwm_cw is
391     port (
392       ce: in std_logic := '1';
393       clk: in std_logic; -- clock period = 74.0740740740741 ns (13.499999999999995 Mhz)
394       pwm_alpha_in: in std_logic_vector(17 downto 0);
395       pwm_update_in: in std_logic;
396       pwm_signal_out: out std_logic
397     );
398   end COMPONENT;
399
400     ---------------------------------------
401     -- Signals for user logic slave model s/w accessible register example
402     ---------------------------------------
403     signal slv_reg0                       : std_logic_vector(0 to C_SLV_DWIDTH-1);
404     signal slv_reg1                       : std_logic_vector(0 to C_SLV_DWIDTH-1);
405     signal slv_reg2                       : std_logic_vector(0 to C_SLV_DWIDTH-1);
406     signal slv_reg3                       : std_logic_vector(0 to C_SLV_DWIDTH-1);
407     signal slv_reg4                       : std_logic_vector(0 to C_SLV_DWIDTH-1);
408     signal slv_reg5                       : std_logic_vector(0 to C_SLV_DWIDTH-1);
409     signal slv_reg6                       : std_logic_vector(0 to C_SLV_DWIDTH-1);
410     signal slv_reg7                       : std_logic_vector(0 to C_SLV_DWIDTH-1);
411     signal slv_reg8                       : std_logic_vector(0 to C_SLV_DWIDTH-1);
412     signal slv_reg9                       : std_logic_vector(0 to C_SLV_DWIDTH-1);
413     signal slv_reg10                      : std_logic_vector(0 to C_SLV_DWIDTH-1);
414     signal slv_reg11                      : std_logic_vector(0 to C_SLV_DWIDTH-1);
415     signal slv_reg12                      : std_logic_vector(0 to C_SLV_DWIDTH-1);
416     signal slv_reg13                      : std_logic_vector(0 to C_SLV_DWIDTH-1);
417     signal slv_reg14                      : std_logic_vector(0 to C_SLV_DWIDTH-1);
418     signal slv_reg15                      : std_logic_vector(0 to C_SLV_DWIDTH-1);
419     signal slv_reg_write_sel              : std_logic_vector(0 to 7);
420     signal slv_reg_read_sel               : std_logic_vector(0 to 7);
421     signal slv_ip2bus_data                : std_logic_vector(0 to C_SLV_DWIDTH-1);
422     signal slv_read_ack                   : std_logic;
423     signal slv_write_ack                  : std_logic;
424
425     ---------------------------------------
426     -- Signals for user logic interrupt example
```

```
427     ---------------------------------------
428     signal intr_counter                   : std_logic_vector(0 to C_NUM_INTR-1);
429
430   begin
431
432     --Synchronisieren auf die Kameradaten -> LVAL FVAL generieren
433     syncer : entity SYNC
434        port map(
435           CLK         => t_clk,
436           RST         => t_rst,
437           DATA_IN     => DATA_IN,
438           LVAL_OUT    => sync_lval,
439           FVAL_OUT    => sync_fval,
440           DATA_OUT    => sync_data
441        );
442
443     -- Serielle Kameradaten parallelisieren
444     dmuxer: entity demux
445        PORT MAP (
446           clk      => t_clk,
447           rst      => t_rst,
448           lval_in  => sync_lval,
449           fval_in  => sync_fval,
450           data_in  => sync_data,
451           lval_out => demux_lval,
452           fval_out => demux_fval,
453           y_out    => demux_data,
454           cb_out   => open,
455           cr_out   => open
456        );
457
458     sobel: entity IMG_PROC
459        PORT MAP (
460           clk      => t_clk2,
461           rst      => t_rst,
462           lval_in  => demux_lval,
463           fval_in  => demux_fval,
464           data_in  => demux_data,
465           lval_out => sobel_lval,
466           fval_out => sobel_fval,
467           data_out => sobel_data
468        );
469
470     imgproc: image_processing_cw
471       port map (
472         clk => t_clk2,
473         imgproc_fval_in => demux_fval,
474         imgproc_lval_in => demux_lval,
475         imgproc_y_in => demux_data,
476         imgproc_data_out => imgproc_data,
477         imgproc_fval_out => imgproc_fval,
478         imgproc_lval_out => imgproc_lval
479       );
480
481     imgproc_mux: process(FILTER_SWITCH)
482        begin
483           if(FILTER_SWITCH = '0') then
484              imgproc_mux_data <= sobel_data;  --Binarize
485              imgproc_mux_fval <= sobel_fval;
486              imgproc_mux_lval <= sobel_lval;
487           else
```

```
488               imgproc_mux_data <= imgproc_data;
489               imgproc_mux_fval <= imgproc_fval;
490               imgproc_mux_lval <= imgproc_lval;
491             end if;
492          end process;
493
494       coordgen: coordinate_generator_cw
495         port map (
496
497           clk => t_clk2,
498           coordgen_fval_in => imgproc_mux_fval,
499           coordgen_lval_in => imgproc_mux_lval,
500           coordgen_y_in => imgproc_mux_data,
501           coordgen_data_out => coordgen_data,
502           coordgen_fval_out => coordgen_fval,
503           coordgen_lval_out => coordgen_lval,
504           coordgen_xcount_out => coordgen_xcount,
505           coordgen_ycount_out => coordgen_ycount
506         );
507
508       pt: projective_transformation_cw
509         port map (
510
511           clk => t_clk3,
512           pt_data_in => coordgen_data,
513           pt_fval_in => coordgen_fval,
514           pt_lval_in => coordgen_lval,
515           pt_xcount_in => coordgen_xcount,
516           pt_ycount_in => coordgen_ycount,
517           pt_data_out => pt_data,
518           pt_fval_out => pt_fval,
519           pt_lval_out => pt_lval,
520           pt_xcount_out => pt_xcount,
521           pt_xpt_out => pt_xpt,
522           pt_ycount_out => pt_ycount,
523           pt_ypt_out => pt_ypt
524         );
525
526       pram_lanedet_roi_x <= lanedet_roi_x(9) & lanedet_roi_x(9) & lanedet_roi_x;
527       pram_lanedet_roi_y <= lanedet_roi_y(7) & lanedet_roi_y(7) & lanedet_roi_y(7) &
      lanedet_roi_y(7) & lanedet_roi_y;
528
529       pram: pixel_ram_cw
530         port map (
531           clk => t_clk2,
532           pram_data_in => pt_data,
533           pram_fval_in => pt_fval,
534           pram_lval_in => pt_lval,
535           pram_roi_x_in => pram_lanedet_roi_x,
536           pram_roi_y_in => pram_lanedet_roi_y,
537           pram_xcount_in => pt_xcount,
538           pram_xpt_in => pt_xpt,
539           pram_ycount_in => pt_ycount,
540           pram_ypt_in => pt_ypt,
541           pram_data_out => pram_data,
542           pram_fval_out => pram_fval,
543           pram_lval_out => pram_lval,
544           pram_xcount_out => pram_xcount,
545           pram_ycount_out => pram_ycount
546         );
547
```

```
548         pt_mux: process(PT_SWITCH)
549         begin
550            if(PT_SWITCH = '0') then
551               pt_mux_data <= pt_data;
552               pt_mux_fval <= pt_fval;
553               pt_mux_lval <= pt_lval;
554               pt_mux_xcount <= pt_xcount;
555               pt_mux_ycount <= pt_ycount;
556            else
557               pt_mux_data <= pram_data;
558               pt_mux_fval <= pram_fval;
559               pt_mux_lval <= pram_lval;
560               pt_mux_xcount <= pram_xcount;
561               pt_mux_ycount <= pram_ycount;
562            end if;
563         end process;
564
565        disp: result_overlay_cw
566        port map (
567          clk => t_clk,
568          disp_data_in => pt_mux_data,
569          disp_fval_in => pt_mux_fval,
570          disp_lval_in => pt_mux_lval,
571          disp_phi_id_in => lanedet_phi_id,
572          disp_r_in => lanedet_r,
573          disp_roi_switch_in => ROI_SWITCH,
574          disp_roi_x_in => lanedet_roi_x,
575          disp_roi_y_in => lanedet_roi_y,
576          disp_x_in => pt_mux_xcount,
577          disp_y_in => pt_mux_ycount,
578          disp_data_out => disp_data,
579          disp_fval_out => disp_fval,
580          disp_lval_out => disp_lval
581        );
582
583      lanedet: lane_detection_cw
584        port map (
585          clk => t_clk,
586          lanedet_data_in => pt_data,
587          lanedet_xpt_in => pt_xpt,
588          lanedet_ypt_in => pt_ypt,
589          lanedet_done_out => lanedet_done,
590          lanedet_max_val_out => lanedet_max_val,
591          lanedet_mroi_x_out => lanedet_mroi_x,
592          lanedet_mroi_y_out => lanedet_mroi_y,
593          lanedet_phi_id_out => lanedet_phi_id,
594          lanedet_r_out => lanedet_r,
595          lanedet_roi_x_out => lanedet_roi_x,
596          lanedet_roi_y_out => lanedet_roi_y
597        );
598
599      pathfollow: pathfollowing_cw
600        port map (
601          clk => t_clk3,
602          pathfollow_d_in => "0000000000", --Distance
603          pathfollow_max_val_in => lanedet_max_val,
604          pathfollow_pd_switch_in => PD_SWITCH,
605          pathfollow_phi_id_in => lanedet_phi_id,
606          pathfollow_pp_switch_in => FTC_PP_SWITCH,
```

```
609           pathfollow_roi_y_in => lanedet_mroi_y,
610           pathfollow_start_in => lanedet_done,
611           pathfollow_alpha_out => pathfollow_alpha,
612           pathfollow_done_out => pathfollow_done
613         );
614
615       pwm: pwm_cw
616         port map (
617           clk => t_clk2,
618           pwm_alpha_in => pathfollow_alpha,
619           pwm_update_in => pathfollow_done,
620           pwm_signal_out => PWM_SIGNAL
621         );
622
623
624       --Zeilenverdopplung
625       deinterer: entity DEINTERLACE
626          port map (
627             CLK          => t_clk,
628             RST          => t_rst,
629             R_IN         => switch_out_r,
630             G_IN         => switch_out_g,
631             B_IN         => switch_out_b,
632             LVAL_IN      => switch_out_lval,
633             FVAL_IN      => switch_out_fval,
634             LVAL_OUT     => t_lval_deint,
635             FVAL_OUT     => t_fval_deint,
636             R_OUT        => t_r_deint,
637             G_OUT        => t_g_deint,
638             B_OUT        => t_b_deint
639          );
640
641       --VGA-Controller
642       VGA: entity VGA_Top_V2b
643          port map(
644             CLK25_IN     => t_clk,
645             RST_IN       => OUTPUT_SWITCH,
646
647             RED_IN       => t_r_deint,
648             BLUE_IN      => t_b_deint,
649             GREEN_IN     => t_g_deint,
650             LVAL_IN      => t_lval_deint,
651             FVAL_IN      => t_fval_deint,
652
653             HSYNC        => HSYNC,
654             VSYNC        => VSYNC,
655             BLANK_INV    => BLANK,
656
657             RED_OUT      => RED_OUT_TB,
658             BLUE_OUT     => BLU_OUT_TB,
659             GREEN_OUT    => GRN_OUT_TB
660          );
661
662       --CLK Manager fuer 13,5(t_clk2) 27(t_clk) und 54(t_clk3) MHz
663       CLKMGR: entity ClkManager
664          PORT MAP(
665             CLKIN_IN           => CLK,
666             RST_IN             => t_rst,
667             CLKDV_OUT          => t_clk2,
668             CLKIN_IBUFG_OUT    => open,
669             CLK0_OUT           => t_clk,
```

```
670             CLKX2_OUT         => t_clk3,
671             LOCKED_OUT        => open
672         );
673
674     --Beschalten des VGA-Controller, hier andere Quellen Muxen
675
676     process (t_clk, t_rst, OUTPUT_SWITCH)
677        variable fval,lval : std_logic;
678        variable r,g,b :std_logic_vector(7 downto 0);
679        begin
680           if t_rst = '1' then
681              DATA_IN_BUF <= '0';
682           elsif rising_edge(t_clk) then
683              DATA_IN_BUF <= DATA_IN(0);
684           end if;
685
686           fval := disp_fval;
687           lval := disp_lval;
688           r := disp_data(23 downto 16);
689           g := disp_data(15 downto 8);
690           b := disp_data(7 downto 0);
691
692           switch_out_fval <= fval;
693           switch_out_lval <= lval;
694           switch_out_r <= r;
695           switch_out_g <= g;
696           switch_out_b <= b;
697           FG_LVAL <= lval;
698           FG_FVAL <= fval;
699           FG_DATA <= r(7 downto 5) & g(7 downto 5) & b(7 downto 6);
700
701     end process;
702
703     --abridge RGB-Channels
704     TBRED <= RED_OUT_TB(7 downto 5) AND (t_lval_deint&t_lval_deint&t_lval_deint);
705     TBGRN <= GRN_OUT_TB(7 downto 5) AND (t_lval_deint&t_lval_deint&t_lval_deint);
706     TBBLU <=  BLU_OUT_TB(7 downto 6)AND  (t_lval_deint&t_lval_deint);
707
708     TBHSYNC <= HSYNC;
709     TBVSYNC <= VSYNC;
710
711     FG_CLK <= t_clk2;
```

Anmerkung: Rest der user_logic ist ein Beispielcode zum Beschreiben der S/W-Register, daher nicht für die eigentliche Anwendung notwendig.

Abbildung B.9.: Die RTL-Beschreibung des Anwender-IP-Cores ist in der user_logic enthalten. Die Netzlisten und VHDL-Module werden als Komponenten der user_logic eingebunden und in der Port-Map miteinander verknüpft.