Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Masterarbeit

Johann-Nikolaus Andreae

Automatische VHDL-Codegenerierung für Beschleunigermodule einer SoC-Plattform am Beispiel der CCD-basierten Fahrspurerkennung

Fakultät Technik und Informatik
Department Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Johann-Nikolaus Andreae

Automatische VHDL-Codegenerierung für Beschleunigermodule einer SoC-Plattform am Beispiel der CCD-basierten Fahrspurerkennung

Masterarbeit eingereicht im Rahmen der Masterprüfung
im Studiengang 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. Thomas Canzler

Abgegeben am 16. Januar 2011

Johann-Nikolaus Andreae

Thema der Bachelorarbeit

Automatische VHDL-Codegenerierung für Beschleunigermodule einer SoC-Plattform am Beispiel der CCD-basierten Fahrspurerkennung

Stichworte

VHDL, Xilinx System Generator, projektive Bildtransformation, Bildatenstrom regeneration, Fahrspurerkennung, FPGA, VHDL-Codegenerator, System on Chip, Bildverarbeitung, Matlab, Simulink, Multiratesystem

Kurzzusammenfassung

Implementierung einer Bildverarbeitungspipeline zur Darstellung des projektiv transformierten Kamerabildes auf einem VGA-Monitor unter Verwendung des modellbasierten Entwicklungskonzeptes des „System Generators“ von Xilinx zur Entwicklung von FPGA-Echtzeitsystemen. Zur Regenerierung des Bilddatenstromes hinter der projektiven Bildtranformation wird ein Zwischenspeicher mit einem minimalen Verbrauch von FPGA-Hardwareresourcen eingesetzt. Die Mindestgröße des Zwischenspeichers wird durch die Analysen des Abstandes zwischen Quell- und Zielkoordinate der Bildpunkte in der projektiven Transformation ermittelt.

Johann-Nikolaus Andreae

Title of the paper

Automatic vhdl code generation of accelerator modules for System on Chip platform, using the example of the CCD based lane detection

Keywords

VHDL, Xilinx System Generator, projective transformation, homography, collineation, Framebuffer, lane detection, FPGA, VHDL code generation, System on Chip, image processing, Matlab, Simulink, multi rate system

Abstract

Implementation of an image processing pipeline to display of the projective transformed camera image on a VGA monitor using the model-based development concept from the "system generator" to develop Xilinx FPGA real-time systems. To regenerate the image data stream behind the projective transformation a frame buffer is used with a minimum consumption of FPGA hardware resources. The minimum cache size is determined by the analysis of the distance between source and target coordinate of the pixels in the projective transformation.

Inhaltsverzeichnis

1. Einführung

Die ersten Steuergeräte im Automobil dienten der Verbesserung des Fahrverhaltens. Durch sie wird das Verhalten des Fahrzeuges in einer Weise verändert, wie es durch die rein mechanische Auslegung nicht zu realisieren wäre. Steuergeräte machen das Automobil auch in schwierigen Situationen noch für den Fahrer beherrschbar. Als eine der ersten Sicherheitsfunktionen wurde das Antiblockiersystem im Automobil eingeführt (vgl. Isermann, 2006). Das Antiblockiersystem verhindert, dass die Räder beim Bremsen vollständig blockieren und somit von der Haftreibung in die Gleitreibung übergehen deshalb bleibt das Fahrzeug auch bei einer Vollbremsung noch lenkbar (vgl. Burckhardt, 1993).

Neben den Sicherheitssystemen wurden auch Komfortsysteme entwickelt. Diese Systeme machen das Fahren des Fahrzeuges angenehmer, indem sie dem Fahrer Aufgaben abnehmen (vgl. Santarini, 2008). Eines der ersten Komfortsysteme war der Tempomat (vgl. Isermann, 2006). Der Tempomat hält für den Fahrer das Automobil auf einer konstanten Geschwindigkeit.

Die Straßen sind für eine visuelle Wahrnehmung durch den Menschen ausgelegt. Um den Fahrer bei der Wahrnehmung seiner Umwelt zu unterstützen, werden in Fahrerassistenzsystemen Kameras eingesetzt, um aus den Bildern Informationen zu extrahieren. Zu den kamerabasierten Fahrerassistenzsystemen gehören:

Rückfahrkamera Die Kamera befindet sich im Heck des Fahrzeuges. Durch die Kamera wird das Blickfeld des Fahrers erweitert und der tote Winkel hinter dem Fahrzeug eliminiert. Durch Einblenden der Fahrspur bei aktuellem Lenkeinschlag wird das Rückwertseinparken erleichtert.

Verkehrszeichenerkennung Die Anzeige der aktuell gültigen Beschränkungen ermöglicht es dem Fahrer, sich auch nach dem Passieren des Verkehrszeichens, z. B. über die aktuell gültige Höchstgeschwindigkeit zu informieren.

Fußgänger- und Wilderkennung Unterstützt den Fahrer bei der frühzeitigen Erkennung von Gefahrensituationen durch Wild oder Fußgänger auf der Straße an unübersichtlichen Stellen oder nachts.

Abbildung 1.1.: Autobahnabschnitt aus der Kamera eines Google-Streeview-Fahrzeuges. Die Kamera befindet sich ca. 4 m über dem Boden. Die Fahrbahnmarkierungen laufen durch die perspektivische Verzerrung am Horizont zusammen, wobei die Abstände der gestrichelten Mittellinien abnehmen.

Fahrspuerassistent Verhindert das unbeabsichtigte Wechseln der Fahrspur auf der Autobahn. Dies geschieht entweder durch Warnung oder durch das Vergrößern des für den Fahrspurwechsel nötigen Lenkausschlag.

Weitere Fahrassistenzsysteme sind in der Entwicklung. So gibt es erste Fahrzeuge, die selbstständig in eine Parklücke einparken können.

Ein Ziel der Forschung ist es, ein komplett autonom fahrendes Fahrzeug für den Straßenverkehr zu entwickeln. Erste Fahrzeuge befinden sich bereits auf der Straße. Diese Fahrzeuge sind in eng abgesteckten Gebieten unterwegs, welche teilweise extra für das autonome Fahren präpariert wurden. Bis das autonome Fahren im Serienfahrzeug verfügbar ist, werden noch etliche Jahre vergehen.

An alle Systeme, welche in die Fahrzeugsteuerung eingreifen, werden hohe Echtzeitanforderungen gestellt. Ein Auto legt bei einer Geschwindigkeit von 200 km/h auf der Autobahn 55,6 m/s zurück. Der Abstand zu den Fahrbahnmarkierungen an den Seiten ist meist kleiner als 1m. Bei einer Unaufmerksamkeit des Fahrers muss der Fahrspurassisten also innerhalb kürzester Zeit das Annähern an die Leitlinie erkennen und seine Maßnahmen ergreifen.

An der „Hochschule für Angewandte Wissenschaften Hamburg“ wird im „Faust“-Projekt an autonomen Fahrzeugen geforscht. Eine der eingesetzten Versuchsplattformen ist ein Modellauto im Maßstab 1:10. Neben der Erprobung von Softwarealgorithmen wird auch an der Verlagerung von Funktionen in die Hardware geforscht.

Die Verlagerung von Aufgaben aus der Software in die Hardware hat den Vorteil, dass sie nebenläufig verarbeitet werden, ohne sich gegenseitig zu stören. Dies ist bei softwarebasierten Steuerungsmodulen nicht der Fall. Softwarebasierte Steuerungsmodule werden meist auf Mikrokontrollern eingesetzt, welche nur einen Rechenkern besitzen. Alle Funktionen des Steuerungsmodules müssen sich diesen Rechenkern teilen.

Ein weiterer Vorteil der Verlagerung von Aufgaben in die Hardware ist, dass der Hardwareaufbau auf die Aufgabe optimiert ist. Dies ermöglicht es, Rechenaufgaben mit niedrigeren Systemtaktfrequenzen in der geforderten Zeit zu lösen. Das Absenken der Taktfrequenz hat einen geringeren Stromverbrauch zu folge, welcher für alle mobilen Systeme wichtig ist.

1.1. Ziele der Arbeit

Basierend auf dem Gesamtkonzept zur Fahrspurerkennung (vgl. Abb. 1.2) werden in dieser Arbeit die Module entwickelt, welche zur Darstellung des perspektivisch transformierten Kamerabildes auf einem VGA-Monitor notwendig sind. Die Entwicklung dieser Module wird modellbasiert mit dem „System Generator“ von Xilinx durchgeführt.

Schwerpunkte dieser Arbeit sind:

- Modellbasierte Implementierung der zum Darstellen des Kamerabildes auf einem VGA-Monitor notwendigen Module im „System Generator“ von Xilinx (vgl. Kap. 5, Seite 43).
- Integration der projektiven Transformation in die Bildverarbeitungspipeline (vgl. Kap. 2, Seite 15 und Kap. 5.5, Seite 55).
- Entwicklung eines Zwischenspeichers für das Videobild zum Wiederherstellen des Bilddatenstroms hinter der projektiven Transformation (vgl. Kap. 4, Seite 30 und Kap. 5.6, Seite 59).
- Erprobung des modellbasierten Konzeptes des „System Generators“ für die Entwicklung von echtzeitbasierten FPGA-Systemen (vgl. Kap. 3, Seite 21).
- Vergleich des vom „System Generator“ generierten VHDL-Codes mit direkt in VHDL implementierten Modulen (vgl. Anhang C, Seite 81).

1.2. Übersicht über das Bildverarbeitungssystem

Das Bildverarbeitungssystem besteht aus Kamera-Interface, projektive Transformation und VGA-Interface (vgl. Abb. 1.2 und 1.3). Die Module zur Erkennung des Verlaufs der Fahrbahnmarkierung mit der Hough-Transoformation werden hier nicht beschrieben.

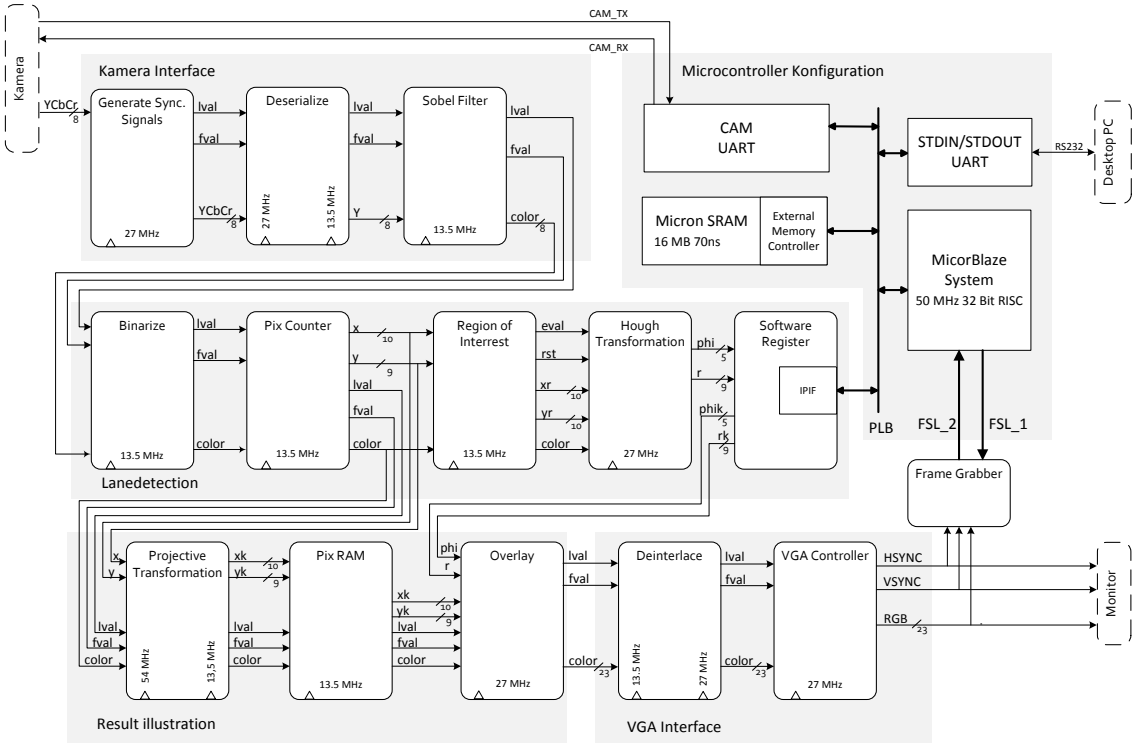


Abbildung 1.2.: Gesamtübersicht der Bildverarbeitungspipeline zur Fahrspurerkennung mit MicroBlaze. In dieser Arbeit werden die Module für die Darstellung des projektiv transformierten Kamerabildes auf einem VGA-Monitor implementiert (vgl. Abb. 1.3).

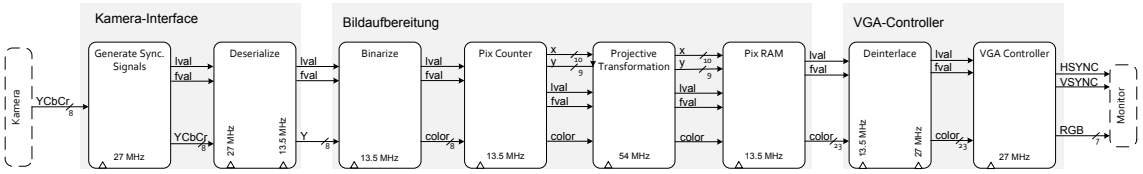


Abbildung 1.3.: Bildverarbeitungspipeline zur Darstellung des projektiv transformierten Kamerabildes auf einem VGA-Monitor.

Die Bildverarbeitungspipeline wird mit dem Takt der Videokamera von 27 MHz betrieben. Mit einem externen Clockmanager werden die von den einzelnen Modulen verwendeten Taktfrequenzen 13,5 MHz, 27 MHz und 54 MHz erzeugt.

Logic	Used	Available	Utilization
Number of Slice Flip Flops	1.074	17.344	6 %
Number of 4 input LUTs	2.389	17.344	13 %
Number of occupied Slices	1.790	8.672	20 %
Total Number of 4 input LUTs	2.872	17.344	16 %
Number of IOBs	21	250	8 %
Number of RAMB16s	6	28	21 %
Number of BUFGMUXs	3	24	12 %
Number of DCMs	1	8	12 %
Number of MULT18X18SIOs	1	28	3 %
Avarage Fanout of Non-Clock Nets	2,10		

Tabelle 1.1.: Hardwareverbrauch der Bildverarbeitungspipeline. Die Übersetzung des Codes fand mit Xilinx ISE 12.3 statt.

	Verzögerung		
	Takte		Zeit
Synchron. Gen.	2	27 MHz	74 ns
Deserialiser	3	27 MHz	111 ns
Binarisierung	0	13,5 MHz	0 ns
Pix Counter	0	13,5 MHz	0 ns
Projektive Trans.	11	13,5 MHz	814 ns
Zwischenspeicher	4285	13,5 MHz	317 μs
Deinterlacer	857	27 MHz	32 μs
VGA-Controler	1	27 MHz	37 ns

Tabelle 1.2.: Verzögerungszeiten der einzelnen Bildverbeitungsmodule

Synchronisationsgenerator Für Markierung von Bild- und Zeilenanfang bzw. Ende sind im ITU Recommendation BT.656 Datenstrom der Sony FCB-PV10 CCD-Kamera Synchronisationsmarkierungen enthalten, die vom Synchronisationsgenerator extrahiert und als Linevalid- (LVAL) und Framevalid-Signal (FVAL) den nachfolgenden Modulen zur Verfügung gestellt werden. Die Signale LVAL und FVAL zeigen den nachfolgenden Modulen ob ein gültiges Bild (FVAL) und eine gültige Zeile (LVAL) im Bilddatenstrom anliegt. (Kap. 5.2, Seite 45)

Deserialisierer Von der Kamera kommt das Videosignal als serieller Datenstrom. Dieses Modul spaltet den Datenstrom in die einzelnen Farbkomponenten Y, Cb und Cr auf. (Kap. 5.2.2, Seite 51)

Binarisierung Aus den Helligkeitsinformationen Y des YC_BC_R-Farbmodells wird anhand eines festen Schwellwertes ein Schwarz-Weiß-Bild erzeugt. (Kap. 5.3, Seite 53)

Projektive Transformation Durch den schrägen Blickwinkel der Kamera auf die Straße ist das Kamerabild perspektivisch verzerrt (vgl. Abb. 1.1 und 1.4). Dies lässt keine lineare Transformation der Bildpunktkoordinaten in eine metrische Maßeinheit zu. Zum Ausgleichen der perspektivischen Verzerrung wird das Bild mit der projektiven Bildtransformation in eine entzerrte Ansicht überführt, welche dem senkrechten Blick aus der Vogelperspektive entspricht. Durch die projektive Bildtranformation ändert sich die Position der Bildpunkte im Bild. (Kap. 2, Seite 15 und Kap. 5.5, Seite 55)

Bildstromrekonstruktion Ein Zwischenspeicher lässt aus der ungeordneten Bildpunktemenge, welche die perspektivische Transformation verlässt, wieder einen geordneten Bildstrom entstehen. Zur Ermittlung der minimalen Speichergröße werden in Kap. 4, Seite 30 die Abstände zwischen Bildkoordinaten von jedem Bildpunkt im Quell- und Zielbild analysiert. Anhand der Analyseergebnisse wird die passende Speichertechnik ausgewählt. (Kap. 5.6, Seite 59)

Deinterlacer Durch Zeilenverdoppelung wird im Deinterlacer aus einem Halbbild ein Vollbild erzeugt und somit die Bildwiederholfrequenz verdoppelt (Kap. 5.7.1, Seite 67).

VGA-Controler Damit das Bild auf einem PC-Monitor dargestellt werden kann, werden im VGA-Controller die Synchronisationssignale HSync und VSync erzeugt. (Kap. 5.7.2, Seite 68)

1.3. Hardwareaufbau des Versuchsfahrzeugs

Die Kamera von Fahrassistenzsystemen befindet sich im PKW meist zwischen dem mittleren Rückspiegel und der Windschutzscheibe. Von dort hat sie einen ähnlichen Überblick wie der Fahrer und durch ihre hohe Lage eine geringere perspektivische Verzerrung der Straße.

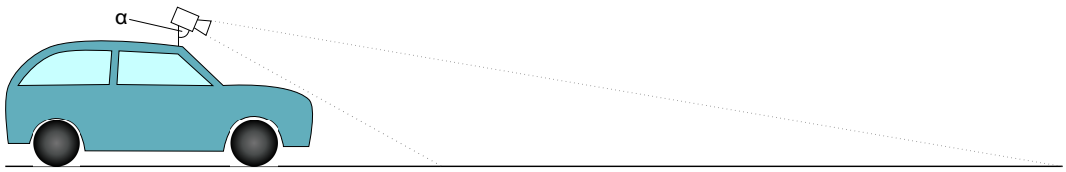


Abbildung 1.4.: Die Kamera des Fahrzeuges blickt um den Winkel α geneigt auf die Straße.

Als Versuchsfahrzeug kommt ein Modellauto im Maßstab 1:10 zum Einsatz. In diesem Fahrzeug ist kein Platz, um dort eine Kamera hinter der Windschutzscheibe unterzubringen. Weitere Kriterien, die gegen eine Montage im Wageninneren sprechen, sind das unterschiedliche Maßstabsverhältnis von Kamera und Modellauto, sowie die meist nicht durchsichtigen Scheiben die in Modellautos eingesetzt werden.

Um für die Kamera ähnliche Sichtverhältnisse wie im PKW zu schaffen, wird diese auf dem Dach montiert. Die Kamera ist in der Längsachse des Fahrzeugs nach unten auf die Straße geneigt (vgl. Abb. 1.4). Die Kamera ist so ausgerichtet das Sie die Straße möglichst bildfüllend erfasst.

Neben der Kamera verfügt das Fahrzeug über Infrarot und Ultraschallsensoren zur Wahrnehmung seiner Umgebung. Auf diese Sensoren wird in dieser Ausarbeitung nicht weiter eingegangen.

CCD-Kamera Als Kamera kommt die CCD-Kamera FCB-PV10 von Sony zum Einsatz (vgl. Abb. 1.5). Die Kamera verfügt über einen Autofokus, eine automatische Anpassung der Bildhelligkeit und einen automatischen Weißabgleich.

Die Konfiguration findet über eine EIA232 (RS232) Schnittstelle statt. Diese muss nur verwendet werden, wenn die Konfiguration geändert werden soll. Die eingestellten Parameter bleiben über eine Unterbrechung der Stromversorgung hinweg erhalten.

Das Kamerabild wird über eine digitale 8/16-Bit Schnittstelle nach dem ITU Recommendation BT.601 und BT.656 Standard zur Verfügung gestellt.

Nexus2 Das Nexus 2 ist ein Entwicklungsbord von Digilent (vgl. Abb. 1.6). Es zeichnet sich durch seine geringen Abmaße und seine Erweiterbarkeit über Pfostenleisten aus. Für die Erweiterung steht eine große Auswahl an Platinen bereit, welche die Hauptplatine um A/D-Wandler, Temperaturfühler, Netzwerkschnittstelle und vieles mehr erweitert.

Das Nexus 2 ist mit einem Spartan 3E-1200 FPGA von Xilinx bestückt. Der FPGA wird von der Platine mit einem 50-MHz-Taktsignal versorgt und kann über USB oder JTAG programmiert werden.

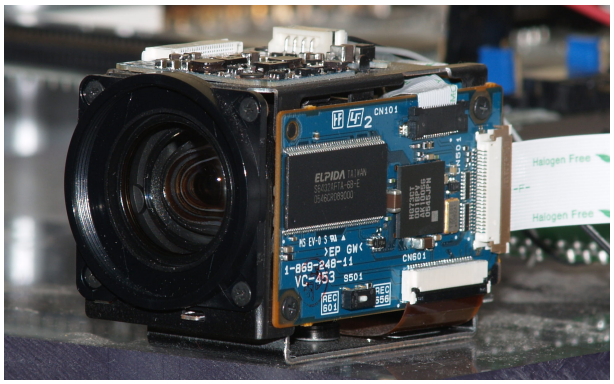


Abbildung 1.5.: CCD-Kamera FCB-PV10 von Sony

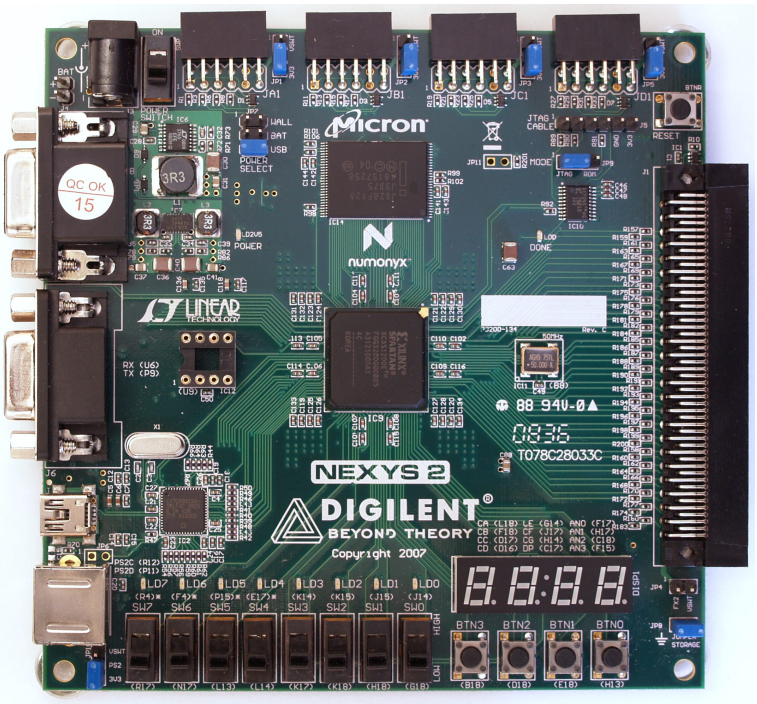


Abbildung 1.6.: Entwicklungsbord Nexus 2 von Digilent

Auflösung	640x480 Pixel
Brennweite	4,2 mm bis 42 mm
maximale Blendenöffnung	1:1,8 (Weitwinkel) / 1:2,9 (Tele)
Minimale Aufnahmeentfernung	10 mm (Weitwinkel) / 1000 mm (Tele)
Pixelfrequenz	27 MHz
Konfigurationsschnittstelle	EIA232, 5 V
Videoausgang	YUV 4:2:2, 3,2 V
Versorgungsspannung	6-12 V DC
Leistungsaufnahme	1,5 W (2,7 W)
Gewicht	84 g
Maße	37,3 x 43,8 x 61 mm

Tabelle 1.3.: Technische Daten der Kamera Sony FCB-PV10 [Sony]

FPGA	Spartan 3E XC3S1200E-4FG320
Anschlüsse	USB2, VGA, PS/2, 4 x 12-Pin PMOD (32 I/O), 100-Pin FX2 (43 I/O)
Input	4 Taster, 8 Schiebeschalter
Output	8 LEDs, 4-Stellige 7-Segmentanzeige
RAM	16MB Micron PSDRAM
Flash	16MB Intel StrataFlash ROM
Clock	FPGA: 50MHz / ext., USB: 48MHz
Programmierschnittstelle	JTAG, USB
Versorgungsspannung	5 - 15V DC
Stromaufnahme	Typ. 200mA

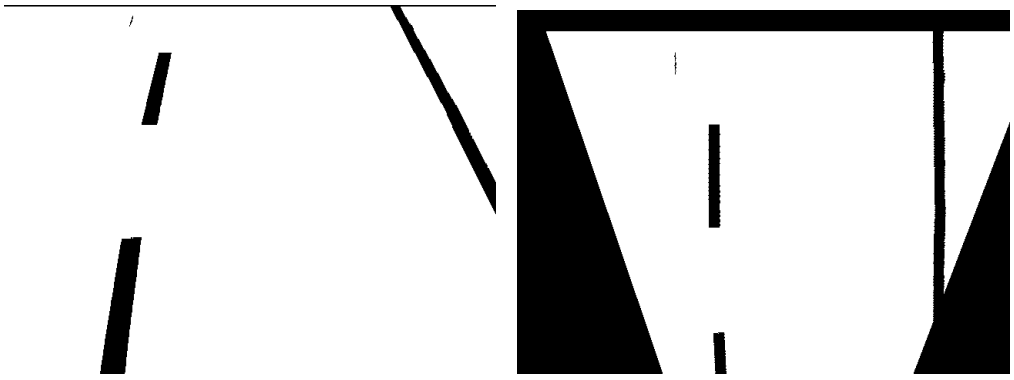
Tabelle 1.4.: Technische Daten des Nexus2 [Digilent (2008)]

System Gates	1200K
Slices	8.672
Look-Up Tables (LUTs)	17.344
Configurable Logic Blocks (CLBs)	2.168
Input/Output Blocks (IOBs)	250
Digital Clock Manager (DCMs)	8
Block RAM	28 18KBit Blöcke
Dedicated Multipliers	28

Tabelle 1.5.: Technische Daten des Sparten 3E (XC3S1200E-4FG320) [Xilinx (2009b)]

2. Grundlagen zur projektiven Bildtransformation

Mit der projektiven Bildtransformation lassen sich Gegenstände aus einer Perspektive betrachten, die die Kamera nicht einnehmen kann. Für die Fahrspurerkennung ist ein Bild aus der Vogelperspektive von Vorteil, in ihm lassen sich die Abstände zur Fahrspur leichter messen. Blickt man aus der Vogelperspektive auf die Straße verlaufen die Seitenmarkierungen parallel, eine gerade Straße vorausgesetzt. Schaut man aber aus der Perspektive des Fahrers auf die Straße laufen die geraden Fahrspurmarkierungen am Horizont zusammen. Aus der Fahrerperspektive ist es schwieriger Abstände zu messen, da kein einheitlicher Maßstab, z. B. 1 Bildpunkt = 1 mm, im Bild vorhanden ist. Mit zunehmendem Abstand zum Fahrzeug wird die Straße im Bild schmaler, wenn z. B. direkt vor dem Fahrzeug am unteren Bildrand ein Bildpunkt 5 mm entspricht, sind es in der Mitte des Bildes vielleicht nur noch 1 mm (vgl. Abb. 2.1 und 1.1, Seite 7). Durch die projektive Transformation bekommt das Bild einen einheitlichen Maßstab.



(a) Strasse aus sicht des Fahrers (b) Perspektivisch Transformierte Strasse

Abbildung 2.1.: Straße aus der Vogelperspektive und Fahrerperspektive aufgenommen. Bei dem Bild aus der Vogelperspektive ist der Verlauf der Straße besser zu erkennen und abstände leichter zu messen.

Für einen guten Überblick ist eine hohe Position der Kamera von Vorteil. Um so höher die Kameraposition ist um so geringer ist die perspektivische Verzerrung. Optimal ist ein senkrechter

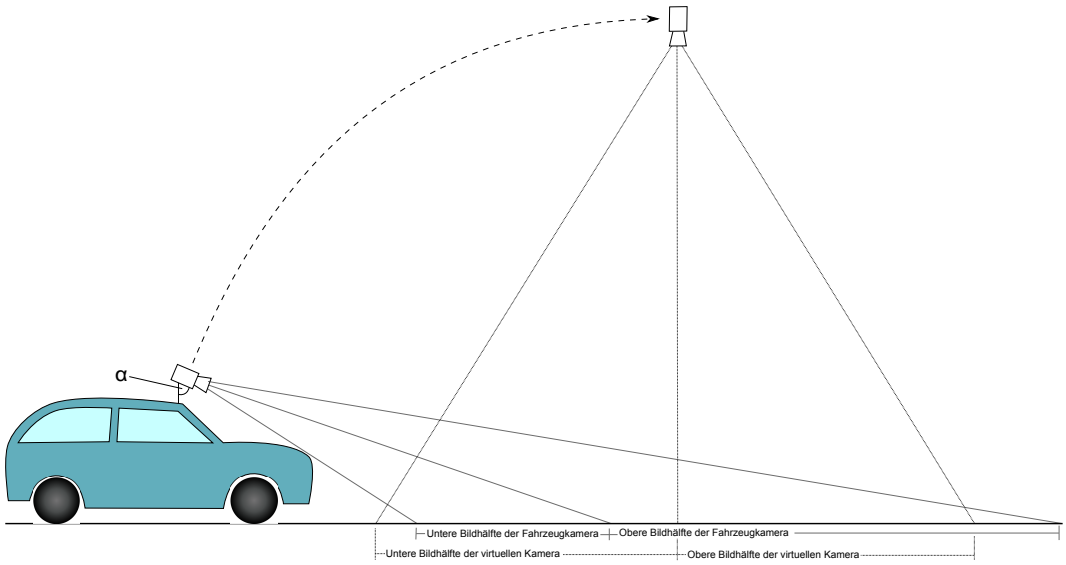
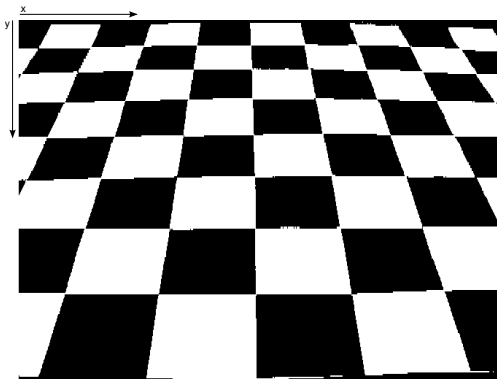


Abbildung 2.2.: Durch die projektive Bildtransformation wird die Kamera virtuell in die Vogelperspektive geschwenkt. Für diesen Kameraschwenk wird der untere Teil des Bildes gestreckt und der Obere gestaucht und somit die perspektivische Verzerrung ausgeglichen.

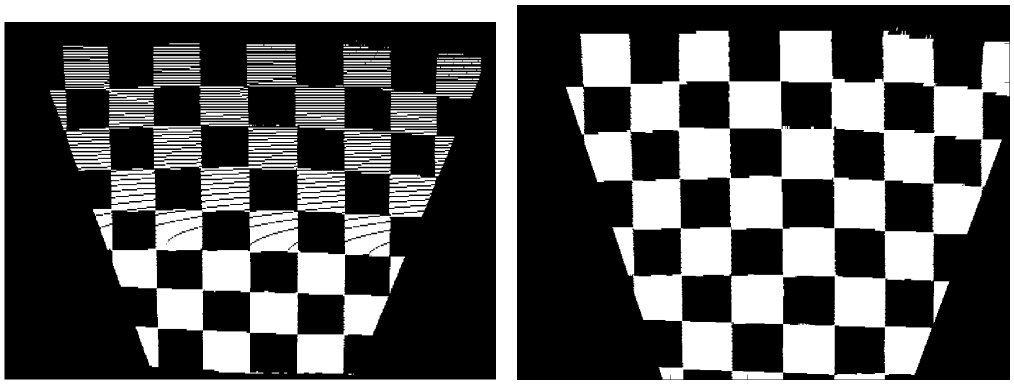
Blick aus der Vogelperspektive auf die Straße. Da diese Position technisch nicht zu Realisieren ist, wird die Verzerrung aus dem Bild herausgerechnet.

Bei der projektiven Bildtransformation wird die Kamera virtuell in eine senkrechte Position über der Straße verschoben (vgl. Abb. 2.2). Bei dieser Verschiebung müssen einige Teile des Bildes gestaucht und andere gestreckt werden. Durch das Stauchen gehen Bildinformationen verloren, da nicht alle Bildpunkte im Zielbild verwendet werden. Im Gegensatz dazu werden in den zu streckenden Teilen des Bildes mehr Informationen benötigt als im Bild vorhanden sind. Die Verschiebung der Bildmitte in der Vertikalachse des Bildes ist in Abb. 2.2 dargestellt. Bei der projektiven Transformation kann auch der Bildausschnitt vergrößert oder verschoben werden.

Bei der projektiven Bildtransformation wird zwischen der Vorwärts- und der Rückwärtstranformation unterschieden. Aus Sicht der PC-gestützten Bildverarbeitung ist die Rückwärtstransformation der Vorwärtstranformation überlegen. Bei ihr ist sichergestellt, dass im Zielbild außer an den Rändern keine undefinierten Bildpunkte vorkommen, welche durch das Strecken des Bildes entstehen (vgl. Abb. 2.3(b) und 2.3(c)). Dies wird entweder durch Doppelverwendung von Bildpunkten des Quellbildes erreicht oder durch Interpolation des Farbwertes aus den benachbarten Bildpunkten, z. B. mit der bilinearen Interpolation, im Quellbild (vgl. Meisel, 2008).



(a) Perspektivisches Kamerabild (Quellbild)



(b) Vorwärtstransformiert mit der Matrix HV (Gl. 2.2) (c) Rückwärtstranformiert mit der Matrix HR (Gl. 2.5)

Abbildung 2.3.: Projektive Entzerrung der Abb. 2.3(a). Größe und Bildausschnitt werden durch die verwendeten Parameter (Matrix) bestimmt.

2.1. Vorwärtstransformation

Bei der Vorwärtstranformation wird für jeden Bildpunkt (x_q, y_q) aus dem Quellbild (Kameraebene) eine Koordinate (x_z, y_z) im Zielbild (Fahrzeugebene) berechnet. Die Abbildung ist weder injektiv noch surjektive. Es kommt also vor, dass Bildpunkte im Zielbild gar nicht oder doppelt getroffen werden (vgl. Abb. 2.4). Vor allem die Stellen, die keinen Wert erhalten, fallen im Zielbild dem menschlichen Betrachter besonders auf, da sie den Wert ihrer Vorinitialisierung behalten (vgl. schwarze Streifen in Abb. 2.3(b)).

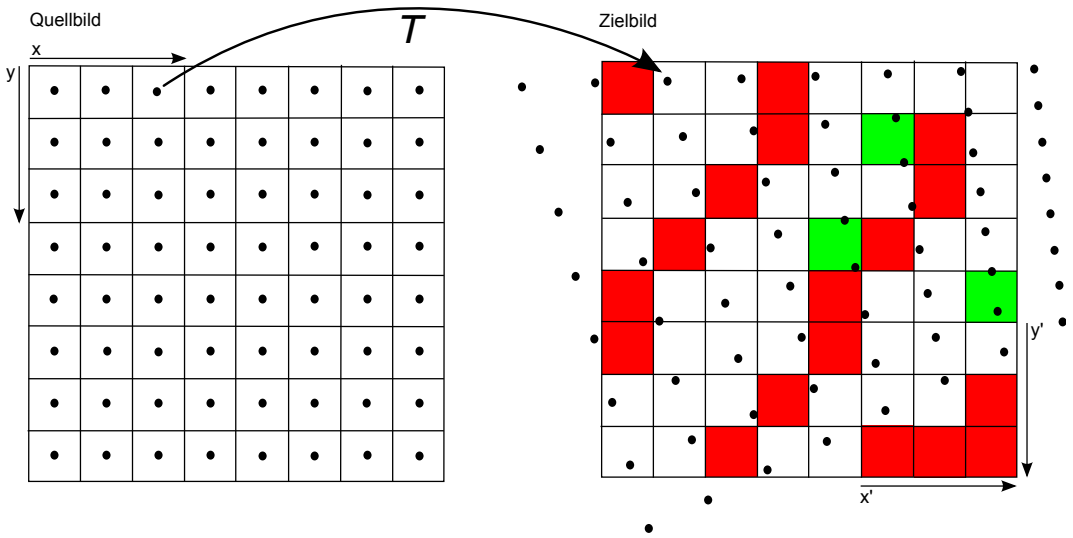


Abbildung 2.4.: Vorwärtstransformation: Für jeden Bildpunkt aus dem Quellbild wird mit der Berechnungsvorschrift T eine Koordinate im Zielbild berechnet. Hierbei werden einige Zielbildpunkte gar nicht (rot) und mache doppelt (grün) adressiert. (vgl. Meisel, 2008)

Für die Brechung der Zieladresse wird eine Projektionsmatrix eingesetzt.

$$H = \begin{pmatrix} x_1' \\ x_2' \\ x_3' \end{pmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} \tag{2.1}$$

Die Transformationsmatrix kann entweder durch Messen des Winkels α (vgl. Abb. 2.2) und der Berechnung der Parameter (vgl. Harley und Zisserman, 2003, Kap. 2.4) oder durch Ausmessen von Referenzpunkten in einem Kamerabild (vgl. Jenning, 2008, Kap. 3.3.2). Die hier verwendete Matrix wurde durch Ausmessen der Referenzpunkte ermittelt (vgl. Mellert, 2010, Kap. 5.4.2).

$$HV = \begin{bmatrix} 0{,}9579 & 0{,}8932 & 35{,}6182 \\ -0{,}0173 & 2{,}4353 & 22{,}4984 \\ -0{,}0001 & 0{,}0026 & 1{,}0 \end{bmatrix} \tag{2.2}$$

Für die Messung war die Kamera auf einem Stativ montiert. Der Neigungswinkel α wurde frei

gewählt. Nach der Montage auf dem Fahrzeug ist die Messung mit einem festen Neigungswinkel α zu wiederholen.

Die Parameter der Matrix HV fließen wie folgt in die Brechung der Koordinaten ein:

$$x' = \frac{h_{11}x + h_{12}y + h_{13}}{h_{31}x + h_{32}y + h_{33}} \qquad y' = \frac{h_{21}x + h_{22}y + h_{23}}{h_{31}x + h_{32}y + h_{33}} \tag{2.3}$$

Für die Berechnung der Koordinaten $x_z = x'$ und $y_z = y'$ des Bildpunktes im Zielbild werden die Koordinaten $x_q = x$ und $y_q = y$ des Bildpunktes im Quellbild in Gl. 2.3 eingesetzt.

2.2. Rückwärtstransformation

Bei der Rückwärtstransformation wird für jeden Bildpunkt (x_z, y_z) im Zielbild die Quelladresse (x_q, y_q) berechnet.

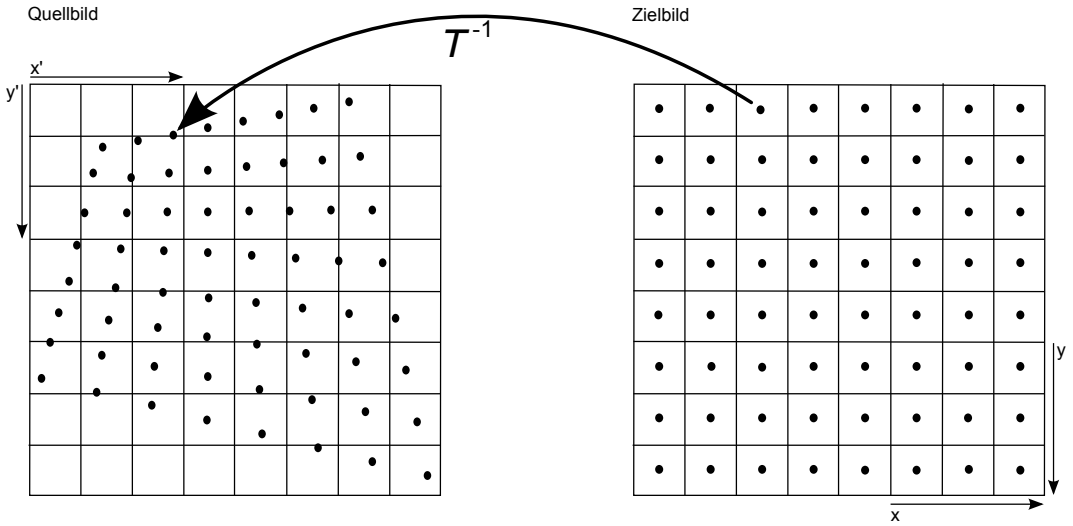


Abbildung 2.5.: Rückwärtstransformation: Für jeden Bildpunkt des Zielbildes wird mit der Berechnungsvorschrift T^{-1} eine Koordinate im Quellbild berechnet. (vgl. Meisel, 2008)

Die Gleichung 2.3 wird nach den Quellkoordinaten x und y umgestellt (vgl. Horsinka, 2010, Anhang A). Daraus ergibt sich die Gleichung 2.4 mit der dazugehörigen Transformationsmatrix 2.5.

$$x = \frac{h_{11}x' + h_{12}y' + h_{13}}{h_{31}x' + h_{32}y' + h_{33}} \qquad y = \frac{h_{21}x' + h_{22}y' + h_{23}}{h_{31}x' + h_{32}y' + h_{33}} \tag{2.4}$$

$$HR = \begin{bmatrix} -1{,}0121 & 0{,}3409 & 28{,}3811 \\ -0{,}0064 & -0{,}4094 & 9{,}44 \\ -0{,}00008 & 0{,}001098 & -1{,}0 \end{bmatrix} \tag{2.5}$$

3. Einführung zum System Generator

Der „System Generator“ von Xilinx ist eine Toolbox für das modellbasierte Entwickeln von digitalen Systemen für FPGA-Plattformen in der Matlab/Simulink Entwicklungsumgebung. Hierzu bietet der „System Generator“ Funktionsblöcke an, welche die FPGA-Logic repräsentieren. Aus dem „System Generator“-Teil des Simulink-Modells wird eine Implementierung, z. B. VHDL-Code im RTL-Stil, für verschiedene FPGA-Typen von Xilinx generiert.

Bei der Simulation von Regelkreisen ist der Einsatz von Matlab/Simulink für die Auslegung weit verbreitet. Durch das direkte Einbinden des Hardwaremodells in ein bestehendes Modell eines Regelkreises werden die Auswirkungen, welche durch den Übergang von einem kontinuierlichen in ein zeitdiskretes System entstehen, direkt sichtbar. Somit lässt sich der Regler direkt auf der Regelstrecke simulieren.

Alternativen zum System Generator sind der „Altera DSP Builder“ und „Simulink HDL Coder“ von MathWorks. Auf diese Alternativen soll hier aber nicht weiter eingegangen werden.

Für den Übergang, zwischen den Simulink-Funktionsblöcken zur Stimulierzeugung und zur Darstellung von Simulationsergebnissen und denen des „System Generators“, gibt es spezielle In- und Output-Funktionsblöcke. Diese Funktionsblöcke bilden die I/O-Leitungen des FPGAs ab. Welcher Pin des FPGAs bei der Implementierung verwendet werden soll, lässt sich im Konfigurationsmenü jedes I/O-Funktionsblockes festlegen. Bei den Input-Funktionsblöcken lässt sich zusätzlich noch festlegen, in welchen Datentyp die Eingangssignale umgewandelt werden sollen und wie hoch die Abtastrate für die Simulation ist. Als Datentypen stehen Boolean, signed und unsigned Vektoren zur Verfügung. Für die signed und unsigned Datentypen muss die Anzahl der Bits vor und nach dem Komma im Q-Format angegeben werden.

Jedes Modell enthält einen „System Generator“-Funktionsblock, über den das Modell konfiguriert wird (vgl. Abb. 3.1).

Übersetzungsart

- HDL Netzlisten
- NGC Netzlisten
- Bitstream
- EDK Exportwerkzeug

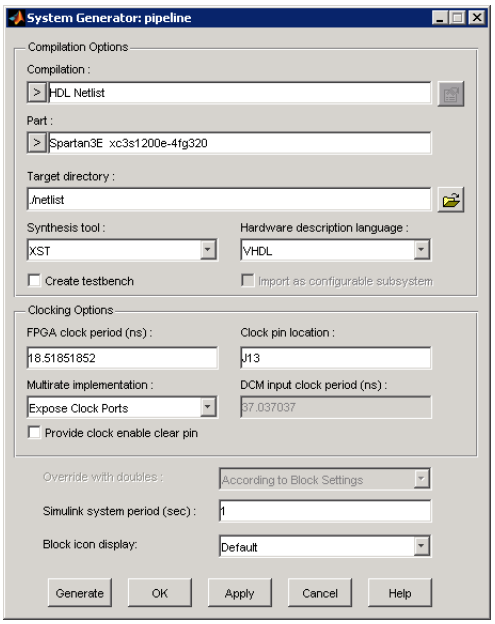


Abbildung 3.1.: Konfigurationsdialog des System Generators. Die Einstellungen beziehen sich auf die Konfiguration der Bildverarbeitungspipeline.

Zielhardware Typ des FPGAs auf dem das Hardwaremodell implementiert werden soll.

Anlegen einer Testbench Zur Ermittlung der Simulationsparameter, für eine externe Simulation z. B. eine Timingsimulation mit ModellSim, werden die Eingangssignale an den Input-Funktionsblöcken verwendet. Das Aufzeichnen der Eingangssignale dauert dieselbe Zeit wie ein Simulationsdurchlauf des Modells in Simulink. Die Simulation von projektiver Bildtranformation mit Zwischenspeicher dauert 1,5 Stunden. Die Codegenerierung ist mit Testbench, je nach Größe des Modells, deutlich langsamer.

Periodendauer des Taktes Die Periodendauer ist für die Hardwaresynthese eine Zwangsbedingung. Sie wird als „Timespec constraint“ in die UCF-Datei eingefügt. Auf die Simulation hat sie keine Auswirkungen.

Pin des Taktgebers Pin, über den der FPGA sein Taktsignal bezieht.

3.1. Taktsignale mit unterschiedlicher Frequenz

In der Digitaltechnik wird mit unterschiedlichen Taktfrequenzen gearbeitet um die gleiche Datenmenge über schmalere Datenbusse zu Transportieren oder zum Beschleunigen von auf-

wendigen Rechnungen in einer Pipeline. Die CCD-Kamera Sony FCB-PV10 überträgt in dem 16-Bit-Bilddatenstrom im 8-Bit-Modus mit der doppelten Taktfrequenz. Beim deserialisieren des Datenstroms halbiert sich die Taktfrequenz wieder (vgl. Kap. 5.2, Seite 45). Das Erhöhen der Taktfrequenz zur Beschleunigung einer Berechnung wird für die Division der projektiven Transformation eingesetzt (vgl. Kap. 5.5, Seite 55).

Im „System Generator“-Modell wird die Taktleitung nicht mitmodelliert. Der Takt jedes Funktionsblockes wird vom Takt der Eingangsdatenleitung abgeleitet. Bei den meisten Funktionsblöcken müssen alle Eingangssignale den gleichen Takt haben. Eine Ausnahme ist z. B. der „Dual Port RAM“. Bei ihm können beide Ports mit unterschiedlichem Takt betrieben werden. Dies kommt beim Deinterlacer zur Anwendung (vgl. Kap. 5.7.1, Seite 67).

Der „System Generator“ unterstützt unterschiedlichen Taktfrequenzen in einem Hardwaremodell. Der Übergang zwischen den verschiedenen Taktfrequenzen wird durch Upsample- und Downsample-Funktionsblöcke realisiert. Auch in anderen Funktionsblöcken ist ein Wechsel der Frequenz von den Ein- und Ausgangssignale möglich. Dieser findet z. B. bei der Bildverarbeitungspipeline im VHDL-Modul „Demultiplexer“ statt (vgl. Kap. 5.2.2, Seite 51).

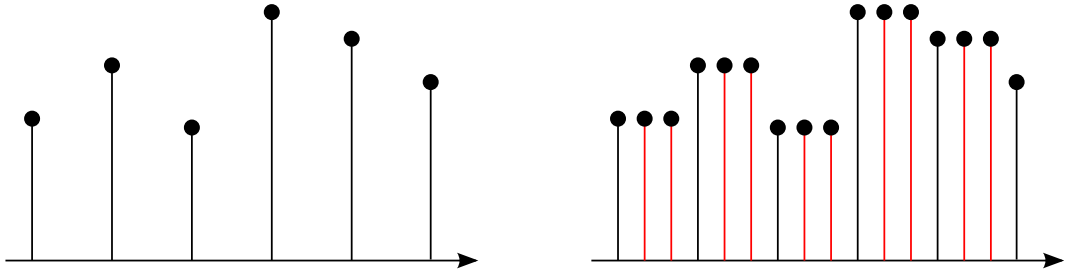
Zur Erhöhung der Taktfrequenz im Upsample-Funktionsblock werden die Eingangsdaten vervielfacht. Für die Reduzierung der Taktfrequenz werden Eingangsdaten weggelassen (vgl. Abb. 3.2).

Für die Erzeugung der Taktsignale mit unterschiedlicher Frequenz kennt der „System Generator“ drei verschiedene Methoden:

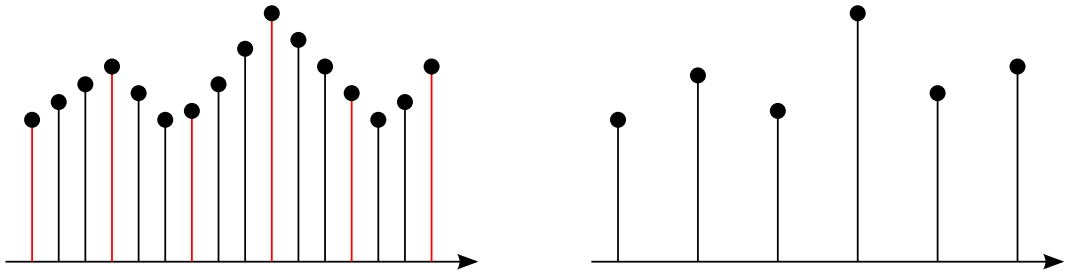
Clock Enable Alle „System Generator“-Funktionsblöcke verfügen über einen „Clock Enable“-Eingang. Durch Abschalten des Taktsignals für einzelne Funktionsblöcke wird die Taktfrequenz diese reduziert. Eine Erhöhung der Taktfrequenz über die Systemtaktfrequenz wird nicht unterstützt.

Hybrid zwischen DCM und Clock Enable Viele FPGA-Typen haben extra „digitale Clock Manager“ (DCM) eingebaut. Der „System Generator“ unterstützt nur die DCM von den Virtex 4, 5 und Spartan 3a DSP FPGAs. Bei diesen FPGAs wird für eine vom FPGA-Typ abhängige Anzahl von Taktfrequenzen die integrierte DCM verwendet. Für alle weiteren Taktfrequenzen wird die „Clock Enable“-Methode verwendet.

Externer Clockmanager Mit den obigen beiden Konfigurationen lassen sich nicht alle Systemkonfigurationen abdecken. Ein Fall ist die nicht vom „System Generator“ unterstützte DCM des Spatan 3E. In der Bildverarbeitungspipeline wird z. B. die DCM des Spatan 3E zur Taktfrequenzverdoppelung von 27 MHz auf 54 MHz eingesetzt, weshalb hier ein externer Clockmanager verwendet wird (vgl. Kap. 5.1, Seite 43). Ein anderer Grund für die Wahl dieser Option ist die externe Takterzeugung mit unterschiedlichen Quarzen.



(a) Im Upsample-Funktionsblock werden die Eingangssignale durch duplikation auf die Ausgangsfrequen gebracht.



(b) Im Downsample-Funktionsblock wird nur jedes n-te Eingangssignal am Ausgang übernommen.

Abbildung 3.2.: Ändert sich die Taktfrequenz zwischen den Funktionsblöcken, ist diese mit Upsample- und Downsample-Blocken anzupassen.

Welche dieser Methoden zur Takterzeugung für das eigene Modell die günstigsten ist, hängt vom verwendeten FPGA-Typ und den zur Verfügung stehenden Hardwareressourcen ab (siehe Kap. C). Als Standard ist beim „System Generator“ die „Clock Enable“ Methode eingestellt. Sie läuft auf allen FPGAs und erfordert keinen zusätzliche Arbeit.

Das Festlegen der Taktfrequenz bei eingebettetem VHDL-Code wird im nächsten Abschnitt beschrieben.

3.2. Einbetten von VHDL-Code

Neben den vom „System Generator“ zur Verfügung gestellten Funktionsblöcken lassen sich auch eigene VHDL-Module über den „Black Box“-Funktionsblock in das Modell einbinden. Beim Einbinden einer VHDL-Datei über den „Black Box“-Funktionsblock wird für diese Datei eine Konfigurationsfunktion als MCode automatisch erzeugt (vgl. Xilinx, 2010, Kap. 4). In der Konfigurationsfunktion wird das Modul mit seinen Schnittstellen beschrieben. Aus der VHDL-Datei können nicht alle Informationen für das „System Generator“-Modell abgeleitet werden, so fehlt z. B. die Information über die Taktfrequenz der Ein- und Ausgangssignale. In diesen Fällen werden vom „System Generator“ Defaultwerte eingesetzt welche anzupassen sind.

Eingangssignale Für alle Input-Ports wird die Prüfung, ob der Datentyp der eingehenden Verbindung zu dem Datentyp des VHDL-Models passt, durchgeführt. 1-Bit-Signale werden vom „System Generator“ bei der automatischen Generation der Konfigurationsfunktion als UFix_1 angelegt.

```
1 this_block.addSimulinkInport('LVAL_IN');
2 if (this_block.port('LVAL_IN').isBool ~= true);
3   this_block.setError('Input data type for port "LVAL_IN" must
        be boolean.');
4 end
5
6 this_block.port('LVAL_IN').useHDLVector(false); % std_logic
```

Listing 3.1: Beispiel für die Konfiguration eines boolean Eingangssignals in der Konfigurationsfunktion einer VHDL-Datei. Die Fehlermeldung in der if-Bedingung wird angezeigt, wenn das Eingangssignal nicht vom Typ boolean ist.

Ausgangssignale Der „System Generator“ kennt im Gegensatz zu VHDL nur drei Datentypen. Für jede ausgehende Verbindung muss angegeben werden, welchen Datentyp diese in Simulink bekommen soll.

```
this_block.addSimulinkOutport('LVAL_OUT');
LVAL_OUT_port = this_block.port('LVAL_OUT');
LVAL_OUT_port.setType('Bool');
LVAL_OUT_port.useHDLVector(false); % std_logic
```

Listing 3.2: Beispiel für die Konfiguration eines boolean Ausgangssignals in der Konfigurationsfunktion einer VHDL-Datei. Alternativ ließe sich für LVAL_OUT mit useHDLVector(true) auch als 1-Bit-Vektor (std_logic_vector(0 downto 0)) angeben.

Taktfrequenz Am Namen eines Signals kann man nur bedingt auf dessen Funktion schließen.Für jeden „Black Box“-Funktionsblock ist mindestens eine Gruppe aus Clock- und ein „Clock Enable“-Signal anzugeben. Jede Datenleitung hat ihre eigene Taktfrequenz. Diese Information ist in der VHDL-Datei nicht vorhanden. Unterscheiden sich die Taktfrequenzen von Eingangs- und Ausgangssignalen, muss dies in der Konfigurationsfunktion geändert werden. Die Angabe der Taktfrequenz bezieht sich in der Konfiguration auf die Taktfrequenz der Eingangssignale.

```
% Konfiguration eines Blocks mit einer Taktfrequenz
  inputRates = block.inputRates;
  theInputRate = inputRates(1);

  % Ausgangssignale haben die halbe Taktfrequenz des
     Eingangssignales
  for i = 1:block.numSimulinkOutports
    block.outport(i).setRate(theInputRate*2);
  end

  % Festlegen der Namen des Clock- und "Clock Enable"-Signals
  block.addClkCEPair('CLK','CE',theInputRate);
```

Listing 3.3: Beispiel für die Konfiguration der Taktfrequenz in der Konfigurationsfunktion einer VHDL-Datei. In diesem Beispiel wurde die Prüfung, ob alle Eingangssignale dieselbe Taktfrequenz haben, entfernt. Es wird die Taktfrequenz des ersten Eingangssignals verwendet.

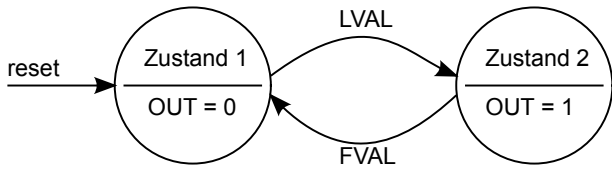


Abbildung 3.3.: Beispiel für einen Moor-Zustandsautomaten mit zwei Eingängen (FVAL und LVAL) und einem Ausgang (out). Wenn FVAL bzw. LVAL logisch 1 ist, springt der Zustandsautomat taktsynchron zwischen den beiden Zuständen hin und her. Der Zustandsautomat ist in Listing 3.5 implementiert.

Subkomponenten Besteht den „Black Box“-Funktionsblock nicht nur aus einer VHDL-Datei, sondern aus mehreren, müssen diese genauso wie die erste VHDL-Datei eingebunden werden.

```
1 this_block.addFile('DEMUX.vhd');
```

Listing 3.4: Einbinden von VHDL-Dateien. Für jede einzubindende Datei wird die Anweisung wiederholt.

3.3. Erstellen von Zustandsautomaten

Mit Zustandsautomaten werden zyklische Funktionsabläufe realisiert, deren Zustandsübergänge taktsynchron meist in Abhängigkeit von den Eingangssignalen Stadtfinden. Die Ausgangssignale können vom Zustandsübergang (Mearly), vom Zustand (Moore) oder auch beidem abhängen. Zustandsautomaten werden in der Bildverarbeitungspipeline (vgl. Abb. 1.3, Seite 9) z. B. zum Generieren der Synchronisationssignale FVAL und LVAL in den Modulen „Generate Sync.-Signal“ (vgl. Kap. 5.2.1, Seite 50), „Pix RAM“ (vgl. Kap. 5.6.3, Seite 63) und Deinterlace (vgl. Kap. 5.7.1, Seite 67).

Zustandsautomaten mit nur jeweils einem Ein- und Ausgang lassen sich mit den im „Xilinx Reference Blockset“ enthaltenen „Mealy“ und „Moore State Mashine“-Funktionsblöcken erstellen. Hat der Automat mehre Ein- oder Ausgänge muss der Automat mit dem MCode-Funktionsblock implementiert werden.

Der MCode-Funktionsblock bindet eine in einer Matlab M-Datei enthaltene Funktion ein. Die Funktion wird vom „System Generator“ nach VHDL übersetzt. In der M-Datei dürfen nur eine Auswahl von logischen (Zuweisungen, if, switch, größer kleiner, gleich, und, oder, nicht) und mathematischen ($+$, $-$, $*$, $/$ nur ^{2},) Ausdrücken verwendet werden (vgl. Xilinx, 2010). Der MCode-Funktionsblock ist nicht mit dem Funktionsumfang von „Xilinx AccelDSP“ zu vergleichen.

Alle Funktionsparameter der, in der M-Datei enthaltenen Funktion, werden in Simulink als Eingangssignale angezeigt und lassen sich mit anderen Funktionsblöcken verbinden. Soll eine Funktion im Matlab mehre Ausgangsvariablen haben, müssen diese als eindimensionale Matrix `[var1, var2, var3]` definiert werden. Alle Parameter der Matrix werden in Simulink als Ausgangssignal angezeigt.

Für die Datentypen hat Xilinx eigene Festkommatypen eingeführt. Diese müssen in den meisten Fällen nicht explizit angegeben werden. Der „System Generator“ erkennt aus den typenlosen Ausgangssignalen den passenden Datentyp. Die Zuweisung `dout=true;` wird zu Boolean, `dout=1;` zu UFix. Wird mit `if din` abgefragt geht der „System Generator“ von einem Boolean Eingangssignal aus. Ein UFix Eingangssignal ist dann für din nicht zulässig. Anders herum ist es bei `if din==1`. Hier darf das Eingangssignal nur UFix oder Fix sein.

Über den Konfigurationsdialog des MCode-Funktionsblocks lassen sich für die Eingangsvariablen feste Werte setzen. Variablen, denen ein fester Wert zugewiesen wurde, werden in Simulink nicht mehr als Eingänge angezeigt. Nicht verwendete Ausgangssignale lassen sich über den Konfigurationsdialog ausblenden.

Für die Fehlersuche in den MCode-Funktionsböcken lässt sich im Konfigurationsdialog des Funktionsblocks der Debugmodus aktivieren. Der Debugmodus ermöglicht es: Haltepunkte zu setzen und den Matlab-Code schrittweise auszuführen. Im Debugmodus wird der Code in den Funktionsblöcken von Matlab-Interpreter ausgeführt. Für die Verwendung des Debugmodus müssen alle Xilinx Datentypen aus dem MCode entfernt werden, da der Code nun von Matlab und nicht vom „System Generator verarbeitet wird. Das Aktivieren des Debugmodus verlangsamt die Simulation spürbar.

```
1  function out = select(fval, lval, reset)
2
3    % Variable zum Speichern des aktuellen Zustandes
4    persistent state,
5    state = xl_state(0, {xlUnsigned, 1, 0});
6
7    % Nach dem Reset immer Zustand 1
8    if reset
9      state=0;
10   end
11
12   % Zustandsautomat
13   switch state
14     case 0 % Zustand 1
15       % Ausgangssignal
16       out = false;
```

```
17        % Bedingung fuer den Zustandsuebergang
18        if lval
19          state = 1;
20        else
21          state = 0;
22        end
23
24      case 1 % Zustand 2
25        % Ausgangssignal
26        out = true;
27        % Bedingung fuer den Zustandsuebergang
28        if fval
29          state = 0;
30        else
31          state = 1;
32        end
33
34      % Wird vom "System Generator" benoetig. Wird nie erreicht.
35      otherwise
36        out = false;
37        state = 0;
38    end;
39  end
```

Listing 3.5: Beispiel für eine Implementierung eines Moor-Zustandsautomaten im „System Generator“ mit zwei Eingängen (FVAL und LVAL) und einem Ausgang (out). Wenn FVAL bzw. LVAL logisch 1 ist, springt der Zustandsautomat taktsynchron zwischen den beiden Zuständen hin und her (vgl. Abb. 3.3).

4. Dimensionierung des Zwischenspeichers zur Regenerierung des Bildstroms

In einem FPGA werden kontinuierliche Datenströme verarbeitet. D. h., für jedes Eingangssignal wird, meist taktsynchron, eine Ausgabe berechnet. Bei der PC-basierten Bildverarbeitung werden die Bildpunkte in einer Struktur mit wahlfreiem Zugriff im Arbeitsspeicher gehalten. Für die Verarbeitung im FPGA ist dieser Ansatz unerwünscht, da das Kopieren in den Speicher Zeit kostet.

Bei der perspektivischen Bildtranformation wird für jeden Bildpunkt eine neue Position berechnet. Wenn in der weiteren Verarbeitung des Bildes es eine Rolle spielt, in welcher Reihenfolge die Bildpixel zur Verfügung stehen, ist eine Zwischenspeicherung unumgänglich.

Für die folgende Weiterverarbeitung der Bilddaten ist die zeilenweise sortierte Reihenfolge der Pixel erforderlich:

- Zeilenweises Darstellen des Bildes auf einem Monitor (VGA)
- Filter mit Nachbarschaftsoperationen: Rechteck, Biominal, Laplace, Sobel ...
- weitere Bildtransformationen

Hingegen kann die Zwischenspeicherung bei diesen Anwendungen entfallen, da diese Verfahren nicht auf die Reihenfolge der Bildpunkte angewiesen sind:

- Hough-Transformation
- Histogramm

Zur Kontrolle der Bildverarbeitungsergebnisse ist eine Ausgabe auf dem Monitor vorgesehen, was eine Zwischenspeicherung erfordert (vgl. Abb. 1.3 auf Seite 9).

Ein komplettes Bild belegt einen Speicherplatz von $Breite * Hoehe * Farbtiefe\ bit$. Für ein binarriesiertes Bild mit der Auflösung 640x480 sind es $640 * 480 * 1 = 307.200bit = 38,4kByte$.

Der Speicher muss auf Bitebene adressierbar sein, damit jeder Bildpunkt einzelnd abgelegt werden kann. Für die Speicherung von 307.200 bit werden 19 Adressleitungen benötigt, was $2^{19} = 524.288$ Speicherzellen entspricht.

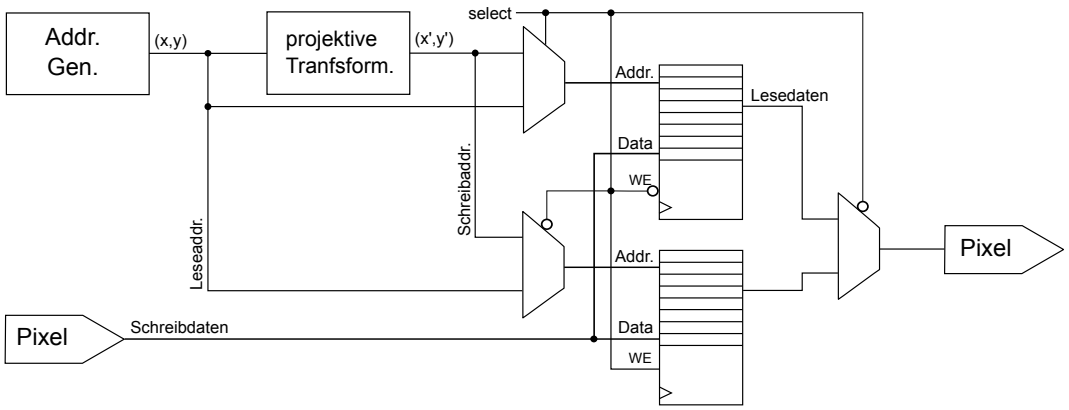


Abbildung 4.1.: Schematischer Aufbau der Vorwärtstransformation mit 2 Zwischenspeichern. Der Adressgenerator erzeugt zu dem Pixelstrom passende Adressen. Die beiden Zwischenspeicher werden abwechselnd mit den perspektivisch transformierten Bilddaten gefüllt. Das „Select“-Signal schaltet sowohl die Multiplexer um als auch das „Write-Enable“-Signal (WE) der beiden Speicher.

Bei der Zwischenspeicherung muss sichergestellt werden, dass sich das Bild komplett im Speicher befindet, bevor es gelesen wird. Ist dies nicht sichergestellt, kann es passieren, dass die Leseoperation auf eine Speicheradresse ausgeführt wird, die noch nicht geschrieben wurde. An diesen Adressen befinden sich noch alte Daten, die anstelle des erwarteten neuen Wertes ausgegeben werden.

Zur Vermeidung von Lesefehlern wird der Zwischenspeicher doppelt ausgelegt. Während des Schreibens in den ersten Zwischenspeicher wird der andere Zwischenspeicher gelesen. Am Ende des Bildes werden die Speicher getauscht (vgl. Abb. 4.1 und 4.2). Durch die Zwischenspeicherung wird die Ausgabe des Bildes verzögert.

Die Vorwärts- und die Rückwärtstranformation unterscheiden sich durch die Verwendung der Adressen. Bei der Vorwärtstranformation wird die von der Transformation errechnete Adresse für das Schreiben in den Speicher genutzt. Die Rückwärtstranformation nutzt die berechneten Adressen der Transformation für das Lesen aus dem Speicher. Für das Lesen werden bei der Vorwärtstranformation die fortlaufenden Adressen aus dem Adressgenerator genutzt, bei der Rückwärtstranformation werden die fortlaufenden Adressen für das Schreiben verwendet.

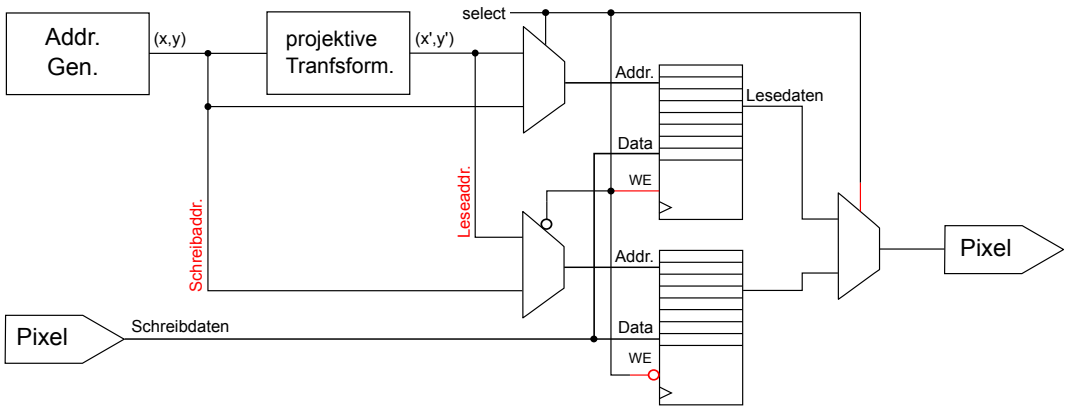


Abbildung 4.2.: Schematischer Aufbau der Rückwärtstranformation mit 2 Zwischenspeichern. Die Rückwärtstranformation unterscheidet sich von der Vorwärtstranformation durch die für das Lesen und Schreiben verwendeten Adressen. Die Änderungen zu Abb. 4.1 sind in Rot hervorgehoben. Die „Write-Enable“-Signale der Speicher und der Ausgangsmultiplexer (rot) sind genau invertiert zur Vorwärtstranformation.

4.1. Reduzierung des Zwischenspeichers durch Einsatz eines Dual-Chanel-RAMs

Bei der perspektivischen Bildtranformation ändert sich die Reihenfolge der Bildpunkte nicht komplett, wie es z. B. bei einer Drehung des Bildes um $180°$ der Fall ist. Durch den schrägen Blickwinkel der Kamera auf die Straße (vgl. Abb. 2.2, Seite 16), im Vergleich zum Blick aus der Vogelperspektive werden die weiter entfernten Teile der Straße im oberen Teil des Bildes gestaucht und die näheren Teile gestreckt dargestellt (vgl. Abb. 2.3 auf Seite 17). Die Verschiebung der Bildpunkte im Bild findet also nur in einem lokalen Teil des Bildes statt. Hieraus leitet sich die Annahme ab, dass der Abstand zwischen Quell und Zielkoordinate kleiner ist als der Abstand zwischen der ersten und der letzten Bildzeile. In einer ersten Abschätzung gehen wir davon aus, dass die Sprungweite kleiner als die Hälfte des Bildes ist.

Unter diese Annahme lässt sich Speicher sparen und der zeitliche Versatz minimieren. Die Reduzierung basiert darauf, dass es Speicherbereiche gibt, die bereits komplett aufgebaut sind, es also keinen Konflikt zwischen dem Leseprozess und dem Schreibprozess gibt, wenn beide auf dem gleichen Speicher arbeiten. Der Lese- und Schreibzugriff findet auf denselben Speicher über getrennte Adressleitungen zeitversetzt statt. Wenn der Schreibprozess die erste Hälfte des Bildes in den Speicher geschrieben hat und mit der ersten Zeile der zweiten Bildhälfte beginnt, kann der Leseprozess die erste Bildzeile der ersten Bildhälfte gefahrlos auslesen, da diese komplett in den Speicher geschrieben wurde (vgl. Abb. 4.3). Im weiteren Verlauf

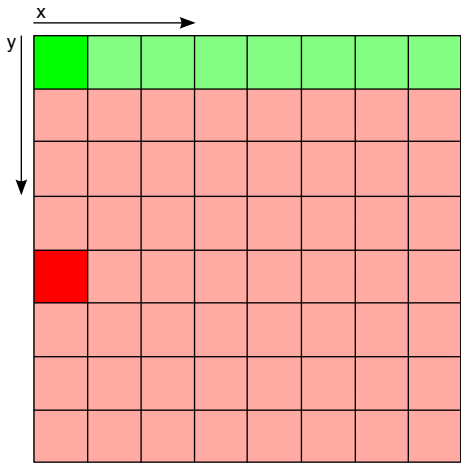


Abbildung 4.3.: Der Schreibprozess berechnet die Speicheradresse (im hellroten Bereich) für den ersten Bildpunkt der zweiten Bildhälfte (dunkelrot). Die berechnete Adresse kann nicht mehr in der ersten Zeile der ersten Speicherhälfte (hellgrün) liegen. Diese kann synchron zum Schreibprozess gelesen werden (dunkelgrün).

laufen beide Prozesse taktsynchron nebeneinander her. Wenn einer am Ende des Speichers angelangt ist, fängt er vorne wieder an. Diese Beschreibung bezieht sich auf die Vorwärtstranformation ist aber auch auf die Rückwärtstranformation übertragbar.

Durch diese Maßnahme fällt der zweite Zwischenspeicher weg. Des Weiteren verkürzt sich die Latenz auf ein halbes Frame, da vor dem Start der Ausgabe nicht mehr gewartet wird, bis ein Bild komplett im Speicher aufgebaut ist.

Der zeitliche Versatz zwischen Schreiben und Lesen muss nur so groß sein wie die maximale Sprungweite zwischen Quell- und Zielkoordinate. Dies stellt sicher das in dem zu lesenden Bereich keine ungeschriebenen Bildpunkte mehr vorhanden sind. Ziel ist die Minimierung des Bildbereiches, der im Zwischenspeicher gehalten werden muss. Dies hat die Senkung der Latenz der perspektivischen Bildtranformation zur Folge, welches die Reaktionsgeschwindigkeit des Gesamtsystems erhöht, da die Daten schneller zur Verfügung stehen.

4.2. Analyse der Abstände zwischen Quell- und Zielkoordinate der Bildpunkte bei der Transformation

Um die minimale Speichergröße zu finden, wird der Abstand von Quell- und Zielkoordinaten untersucht. In Abb. 4.4 wird die Distanz zwischen Quell- und Zielpixel in Zeilen dargestellt. Die

Darstellung ist abhängig von der verwendeten Matrix, nicht aber vom Bildinhalt (vgl. Anhang B). Bei der Auswertung beschränken wir uns auf die Bildzeilen (Y-Achse) da diese mit je 640 Pixeln deutlich mehr ins Gewicht fallen als die Verschiebungen innerhalb einer Zeile.

Der Verlauf der Sprungweitenverteilung (vgl. Abb. 4.4) sowie das Histogramm der Sprungweitenhäufigkeit (vgl. Abb. 4.5) ist bei der Vorwärtstranformation und der Rückwärtstranformation ähnlich.

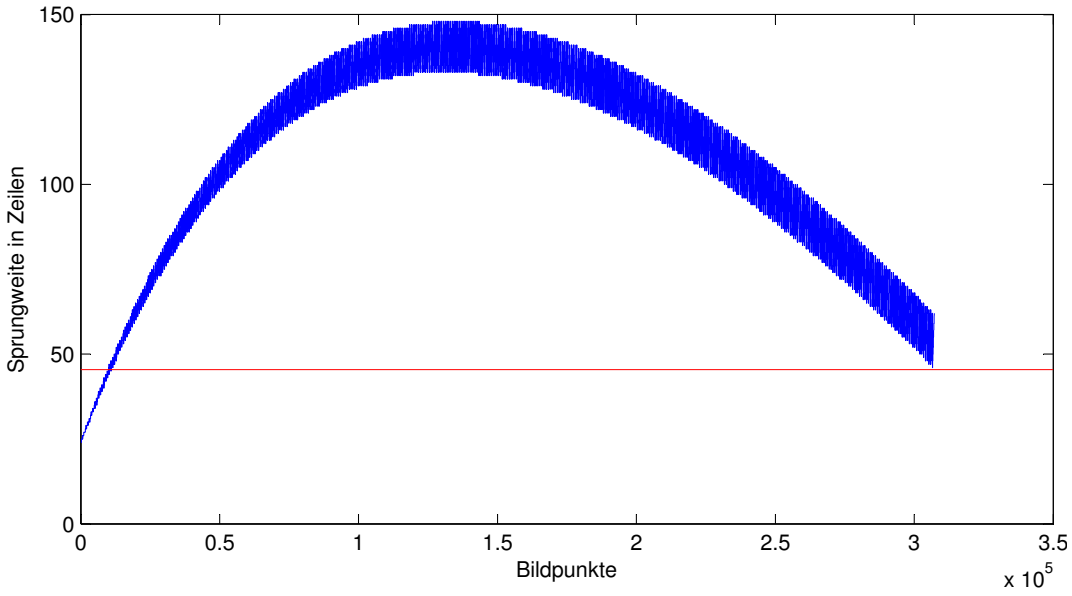
Die Kurve der Vorwärtstranformation macht ihren Bogen in den positiven Bereich, während der Bogen der Rückwärtstranformation in den negativen Bereich geht. Dies liegt an der umgekehrten Blickrichtung der Rückwärtstranformation (vgl. Kap. 2.2, Seite 19).

Bei der Vorwärtstranformation liegt die maximale Sprungweite bei 148 Zeilen (vgl. Abb. 4.4(a) und 4.5(a)) d. h., dass ein Bildpunkt im Extremfall 148 Bildpunkte weiter unten im Bild landet als sein Ursprung ist. Der Leseprozess kann somit mit einem zeitlichen Versatz von $148 * 640 = 94720$ Takte, was $\sim 7ms$ entspricht, den Zwischenspeicher auslesen ohne das es zu einem Konflikt mit dem Schreibprozess kommt.

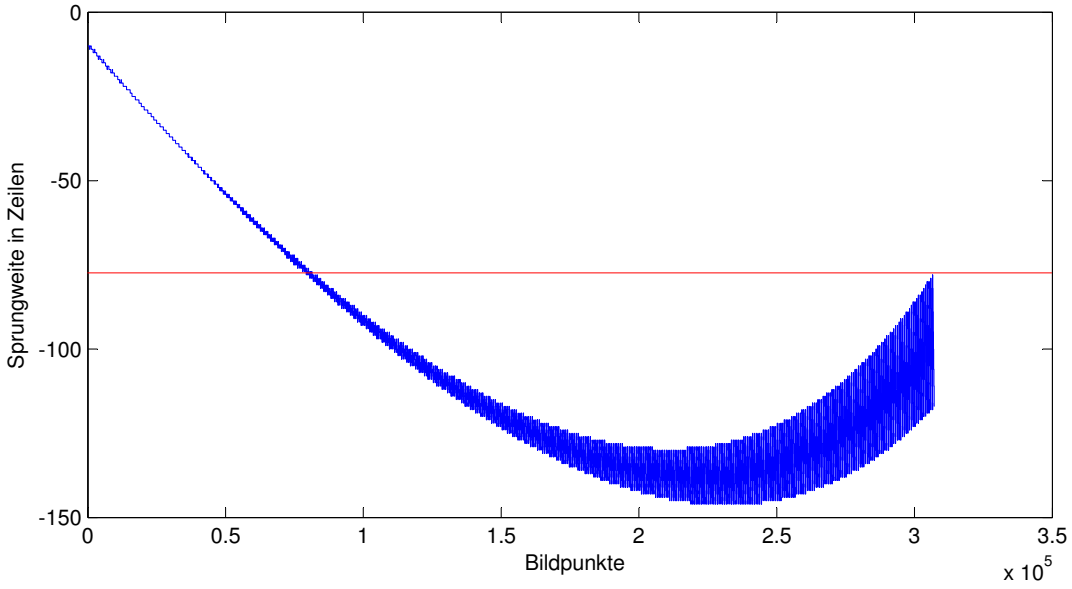
Die Sprungweite der Rückwärtstranformation fällt bei der verwendeten Matrix geringer aus (vgl. Abb. 4.4(b) und 4.5(b)). Ihr Maximum beträgt 146 Bildzeilen, was $146 * 640 = 93440$ Takten oder $\sim 6,9ms$ entspricht.

Bei beiden Transformationen sind die weiten Sprünge die häufigsten, während die kurzen Sprünge seltener vorkommen. Gerade die Sprünge mit einer Entfernung von unter 23 Bildzeilen bei der Vorwärtstranformation bzw. 9 bei der Rückwärtstranformation kommen besonders selten vor (vgl. Abb. 4.5).

Die Sprünge bis 48 Zeilen bei der Vorwärtstransformation und bis 78 Zeilen bei der Rückwärtstranformation enthalten nur Teile vom Bildanfang (vgl. rote Linie in Abb. 4.4 und 4.5). Die größeren Sprünge liegen sowohl im oberen Bildbereich als auch im unteren. Dies macht sich auch durch einen stärkeren Anstieg im Verlauf der Histogrammkurve bemerkbar (vgl. Abb. 4.5).

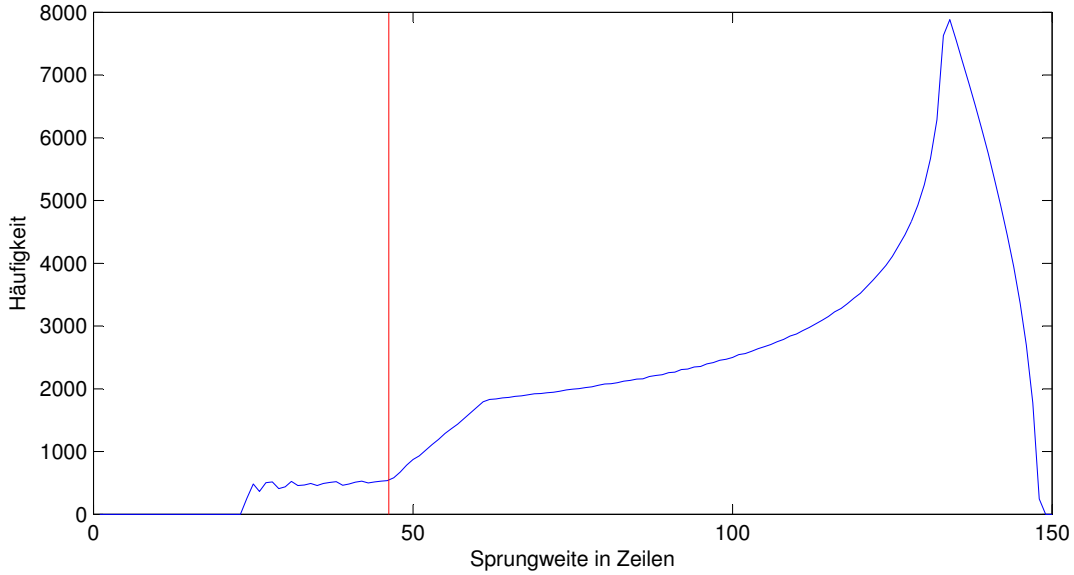


(a) Vorwärtstranformation mit der Transformationsmatrix HV (Gl. 2.2)

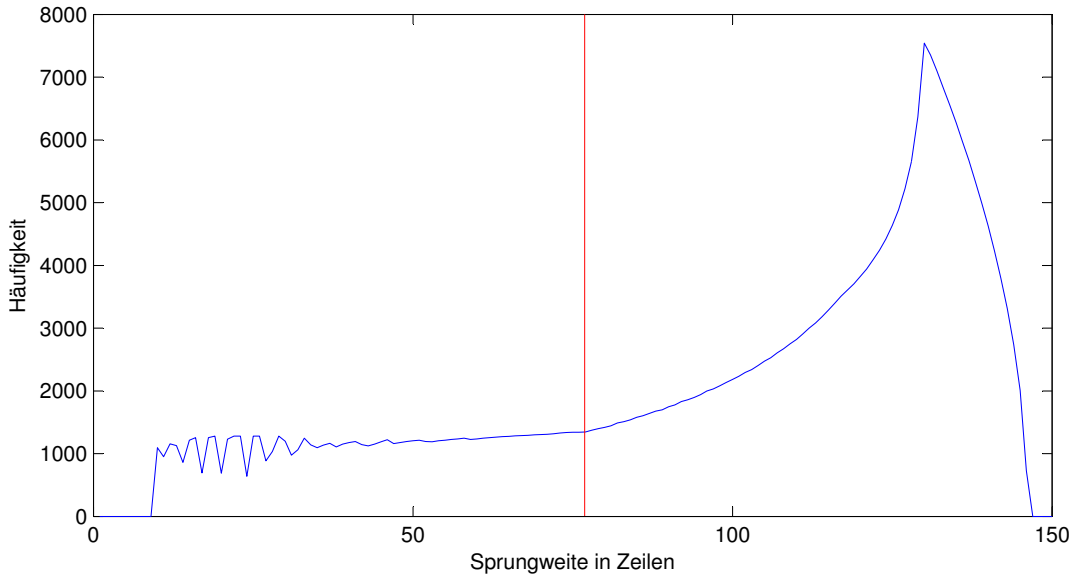


(b) Rückwärtstranformation mit der Transformationsmatrix HR (Gl. 2.5)

Abbildung 4.4.: Verteilung der Sprungweite in Zeilen über die Bildpunkte. Die rote Linie markiert die Grenze, ab der in der oberen und in der unteren Bildhälfte Bildpunkte mit gleicher Sprungweite vorkommen.



(a) Vorwärtstranformation mit der Transformationsmatrix HV (Gl. 2.2)



(b) Rückwärtstranformation mit der Transformationsmatrix HR (Gl. 2.5)

Abbildung 4.5.: Histogramm: Häufigkeit der Zeilensprünge in den unterschiedlichen Weiten. Die rote Linie markiert die Stelle, ab der sich die Zeilensprünge sowohl in vorderen als auch im hinteren Bildteil befinden (vgl. Abb. 4.4).

4.3. Reduzierung der Latenz des Zwischenspeichers durch Verkürzen des Abstandes zwischen Lese- und Schreibprozess

Basierend auf der vorangegangenen Analyse der Sprungweiten, soll die Zeit, die der Bilddatenstrom durch die Regenerierung im Zwischenspeicher verzögert wird, weiter verkürzt werden.

Durch das Ergebnis der Analyse kann der zeitliche Abstand zwischen Lese- und Schreibprozess auf 148 Zeilen, für die Vorwärtstransformation bzw. 146 Zeilen, für die Rückwärtstransformation, verkürzt werden. Durch die Verkürzung des Abstandes verkürzt sich die Latenz wie folgt:

Halbes Bild Abstand	153600 Takte, 11ms
148 Zeilen Abstand	94720 Takte, 7ms
146 Zeilen Abstand	93440 Takte, 6,9ms

Der Speicherverbrauch bleibt unverändert, da sich nur der zeitliche Abstand zwischen dem Lese- und dem Schreibprozess verringert nicht aber die Größe des Dual-Port-RAMs.

Reduziert man den Abstand über diese Grenze hinaus, gehen Informationen verloren, da sich nun zum Zeitpunkt des Lesens noch nicht alle benötigten Bildpunkte im Speicher befinden. Die Sprünge mit maximaler Weite, welche bei diesem Ansatz betroffen sind, liegen etwas oberhalb der Bildmitte (vgl. Abb. 4.4). Dies sind aber gleichzeitig auch die Sprünge, die am häufigsten vorkommen (vgl. Abb. 4.5). Schon bei der Reduzierung, der für den Leseprozess zur Verfügung stehenden Informationen, um ein paar Zeilen geht ein Großteil des Bildes in einem zentralen Teil verloren (vgl. Abb. D.1, Anhang D). Wie wichtig dieser Bereich für die Fahrspurerkennung ist, muss bei der Ermittlung der optimalen ROI (Region of Interest) festgestellt werden. Für eine nennenswerte Speicherreduzierung reicht es nicht ein paar Zeilen zu sparen, hier sind min. 30 Zeilen und mehr notwendig, daher ist eine Reduzierung des zeitlichen Abstandes zwischen Schreib- und Leseprozess über den ermittelten Mindestabstand hinaus nicht erfolgsversprechend.

4.4. Reduzierung des Speichergröße durch Einsatz eines Schieberegisters mit wahlfreiem Zugriff

Zur Reduzierung des Speicherbedarfs wird der Speicher durch ein Schieberegister mit wahlfreiem Zugriff ersetzt. Dieses Schieberegister hat die Größe der maximalen Sprungweite, also $640 * 148 = 94720$ Takte für die Vorwärtstransformation und $640 * 146 = 93440$ Takte für

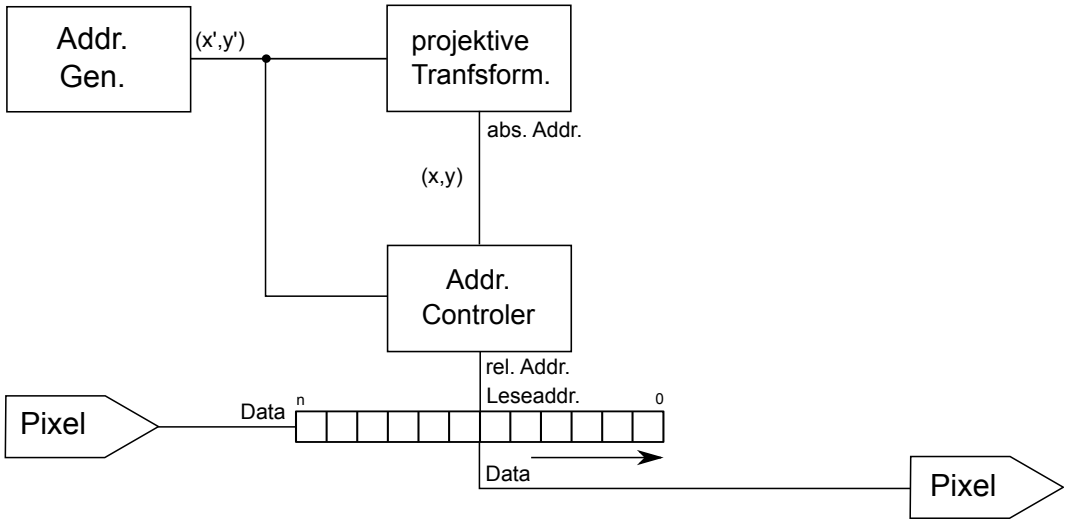


Abbildung 4.6.: Rückwärtstranformation des Bildes mit einem Schieberegister mit wahlfreiem Zugriff als Zwischenspeicher.

die Rückwärtstransformation. Durch das Schieben wird nur der aktuell benötigte Adressraum im Speicher gehalten, hierdurch ändert sich die Adressierung der Bildpunkte von absoluten in relative Adressen.

Das Schieberegister wird bei der Rückwärtstranformation kontinuierlich mit Daten gefüllt (vgl. Abb. 4.6). Die Daten werden über den wahlfreien Zugriff von der Transformation abgegriffen. Für die Berechnung der zu den absoluten Adressen, welche die Transformation liefert, passenden relativen Speicheradressen ist ein Adress-Controller zuständig. Die alten Daten, welche nicht mehr benötigt werden, verfallen durch das Schieben automatisch.

Die Vorwärtstranformation läuft genau umgekehrt ab (vgl. Abb. 4.7). Über den wahlfreien Zugriff werden die Farbwerte in das Schieberegister geschrieben. Durch das Schieben kommen am Ende die Bildpunkte in der richtigen Reihenfolge wieder heraus.

Der minimale Zeitversatz zwischen Schreib- und Lesezugriff beträgt für die Vorwärtstransformation 5 Bildzeilen (vgl. Kap. 5.6.1, Seite 59). Für die Rückwärtstransformation ist aufgrund des Verlaufs der Sprungweiten (vgl. Abb. 4.4) ein ähnlicher Zeitversatz zu erwarten.

Ein andere Ansatz für die Implementierung dieses Verfahrens ist ein Ringbuffer. Die Daten verfallen beim Ringbuffer nicht automatisch wie beim Schieberegister, sondern sind nach dem Lesen zu löschen. Der Ringbuffer wird beim Zwischenspeicher eingesetzt, da er sich mit „System Generator“-Funktionsblöcken zu implementieren ist (vgl. Kap. 5.6.1, Seite 59).

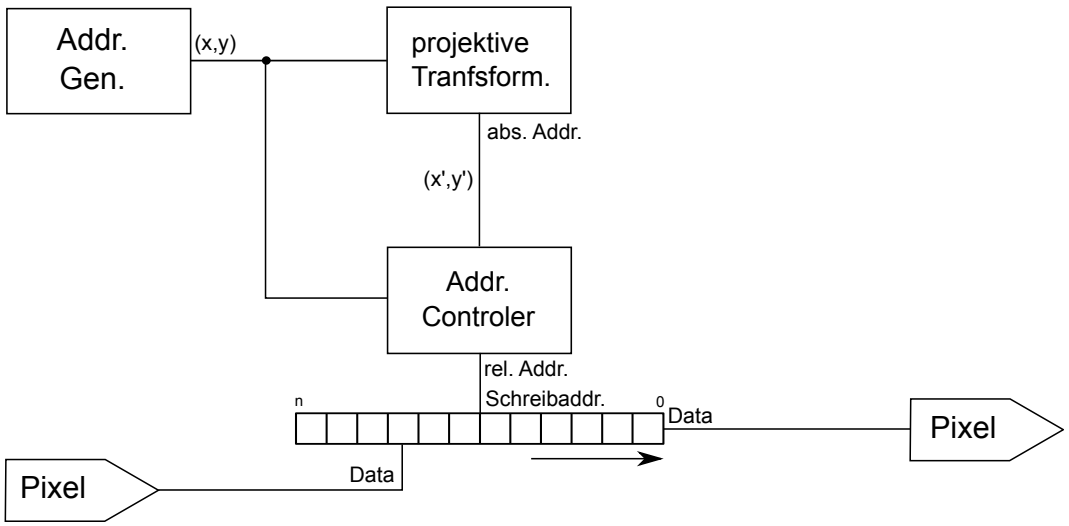


Abbildung 4.7.: Vorwärtstranformation des Bildes mit einem Schieberegister mit wahlfreiem Zugriff als Zwischenspeicher.

4.5. Alternativen zur Regeneration des Bilddatenstroms im Zwischenspeicher

Ohne regenerierten Bilddatenstrom lässt sich das projektive transformierte Bild nicht auf einem Monitor darstellen. Im Folgenden werden drei Konzepte vorgestellt, welche ohne den Zwischenspeicher auskommen oder sich nur auf einen Bildausschnitt beschränken. Diese Konzepte haben Einschränkungen in der Darstellung auf dem Monitor zugunsten eines geringeren Verbrauchs von Hardwareressourcen im FPGA.

4.5.1. Konzept 1

In diesem Ansatz wird die projektive Bildtranformation nur für die Fahrspurerkennung mit der Hough-Transformation verwendet. Die erkannten Geraden werden über das untransformierte Kamerabild gelegt (vgl. Abb. 4.8). Es bleibt dem Betrachter überlassen zu prüfen, ob die Fahrspur richtig erkannt und transformiert wurde.

Durch die getrennten Verarbeitungswege von original Kamerabild und Fahrspurerkennung und deren unterschiedliche Länge (vgl. Abb. 4.9), sind die Bilder bei der Monitorausgabe nicht mehr synchron. Das Ergebnis der Hough-Transformation steht erst zur Verfügung, wenn alle

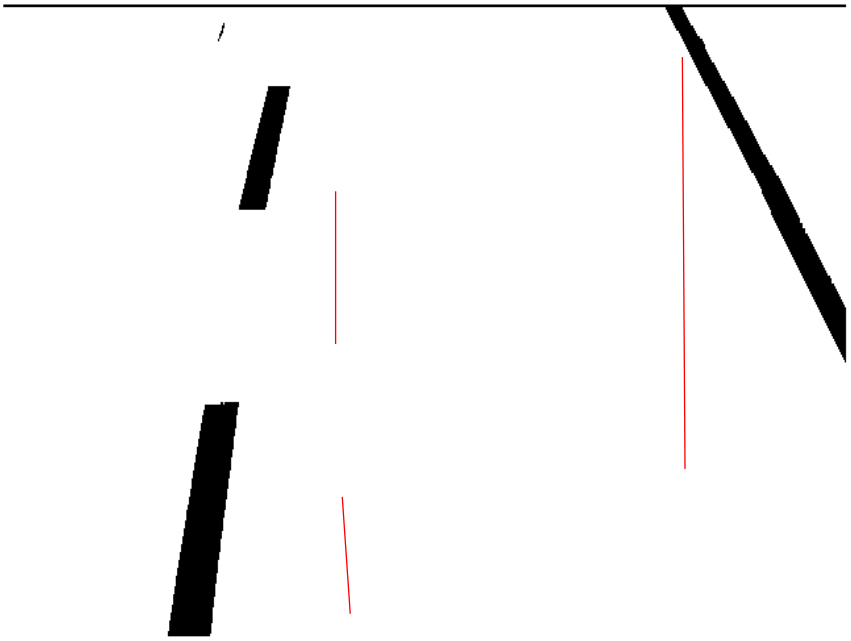


Abbildung 4.8.: Binarisiertes Kamerabild der perspektivisch verzerrten Straße mit perspektivisch korrigierten von der Houg-Transformation berechneten Linien.

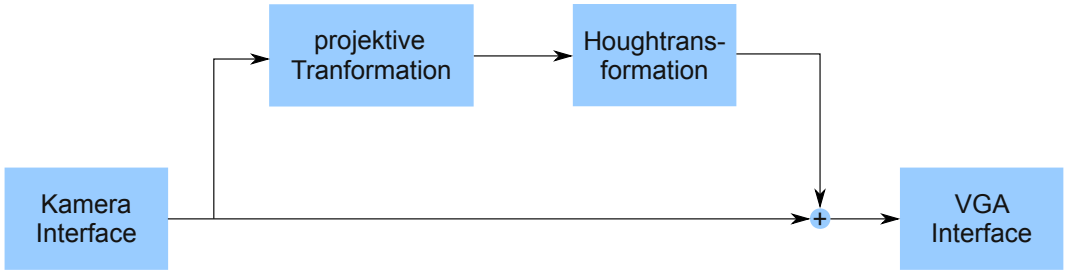


Abbildung 4.9.: Die Berechnung der projektiven Transformation und der Hough-Transformation findet parallel zur Ausgabe des Bildes auf dem Monitor statt. Das Ergebnis der Berechnung wird erst ein Bild später eingeblendet.

Bildpunkte des zu untersuchenden Bereiches ausgewertet sind. Die Synchronisation der Laufzeitunterschiede erfordert das Zwischenspeichern des Bildes. Zur Vermeidung des Zwischenspeichers wird das Ergebnis der der Hough-Transformation erst in das nächste Kamerabild eingeblendet, ist also um einen Frame zeitversetzt. Dieser zeitliche Versatz ist bei 30 Bildern/s vom Menschen nicht wahrzunehmen.

4.5.2. Konzept 2

Wendet man die projektive Bildtranformation nur auf einen kleinen Ausschnitt des Bildes, der Region of Interest (ROI) an, so verringert sich die Zwischenspeichergröße und die benötigte Rechenzeit. Der Nachteil dieses Verfahrens ist, dass sich durch die Bildtransformation die Position des Ausschnittes im Gesamtbild verschiebt (vgl. Abb. 4.10).

4.5.3. Konzept 3

Ein anderes Verfahren zum Ausgleich der perspektivischen Verzerrung des Bildes ist die geometrische Transformation der Geradengleichung. Die Geradengleichung wird von der Hough-Transformation aus dem verzerrten Bild berechnet (vgl. Abb. 4.11). Für die geometrische Transformation wird der umgekehrte Weg eingeschlagen, welcher bei der 3D-Bildverarbeitung für die Darstellung einer Straße auf einem 2D-Monitor verwendet wird. Für dieses Konzept ist der Neigungswinkel der Kamera auf dem Fahrzeug genau zu messen. Der Neigungswinkel geht, anders als bei der Berechnung der Transformationsmatrix über bekannte Bildpunkte (vgl. Kap. 2.1, Seite 17) direkt in die Berechnung ein.

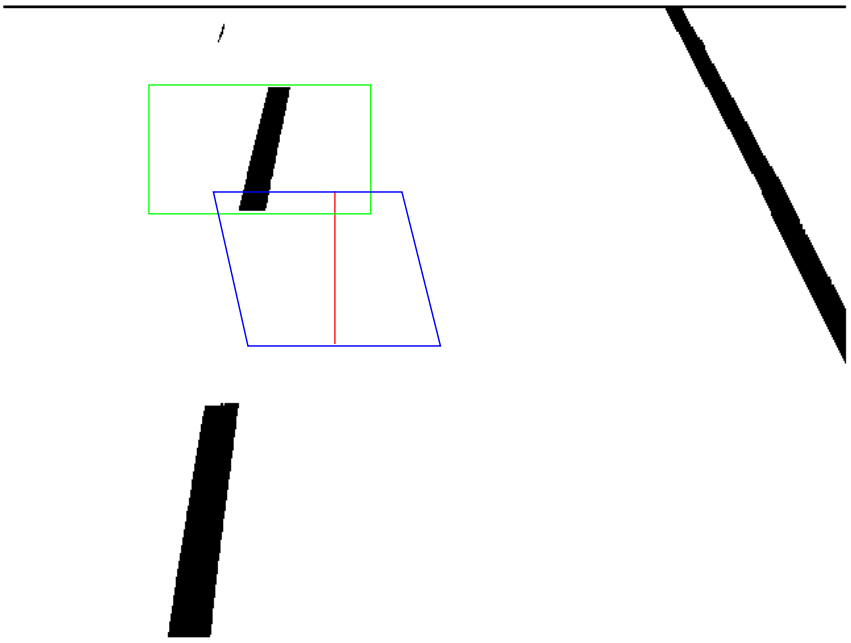


Abbildung 4.10.: Durch die projektive Transformation eines Bildausschnittes (grün) ändert sich dessen Form und die Lage im Gesamtbild (blau).

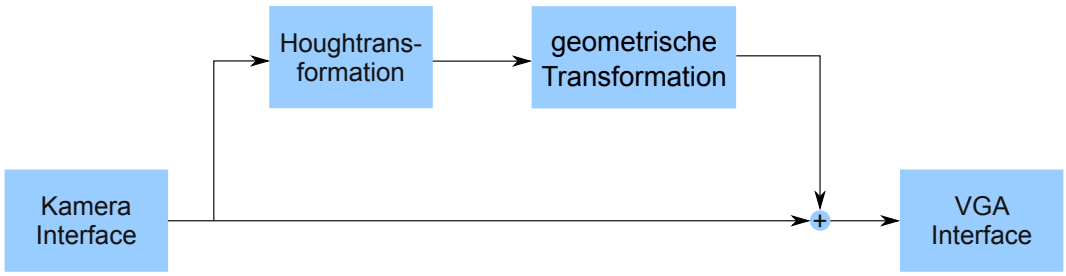


Abbildung 4.11.: Die Hough-Transformation wird auf das verzerrte Kamerabild angewendet. Die berechnete Geradengleichung wird durch geometrische Berechnungen perspektivisch korrigiert. Wie bei Alternative 1 wird die Gerade erst ein Bild später eingeblendet.

5. Realisierung der Bildverarbeitungspipeline

In der Bildverarbeitungspipeline wird der von der Videokamera kommende Bilddatenstrom projektiv transformiert und für die Darstellung auf einem VGA-Monitor aufbereitet. Die Bildverarbeitungspipeline gliedert sich in drei Teile (vgl. Abb. 5.1 und 1.3, Seite 9):

- Kamera-Interface zum Extrahieren von Synchronisationssignalen und Farbinformationen aus dem Kameradatenstrom
- Bildaufbereitung für die Fahrspurerkennung
  - Konvertierung des Graustufenbildes in ein Schwarz-Weiß-Bild
  - Generierung der X/Y-Koordinaten zu den Bildpunkten
  - Projektive Bildtranformation
  - Bildzwischenspeicher zum Wiederherstellen des Bilddatenstroms
- VGA-Interface mit Deinterlacer zur Zeilenverdoppelung und VGA-Controller für die Monitoransteuerung

Die gesamte Bildverarbeitungspipeline ist mit Ausnahme des Clockmangers im „System Generator“ modelliert. Die VHDL-Module „Synchronisations Generator“, Deserialisierer und VGA-Controller sind als Blackbox-Block in das „System Generator“-Modell eingefügt (vgl. Kirschke (2009), Peters (2009)).

Für die 1 Bit breiten Steuer- und Statusleitungen wird innerhalb des Models der Datentyp Boolean verwendet. Der Bilddatenstrom des Binärbildes ist als 1-Bit-Vektor modelliert, um in den übrigen Modulen Bilddaten mit größerer Farbtiefe zu unterstützen (vgl. Kap. 5.3, Seite 53).

5.1. Clockmanager für die Erzeugung der Taktfrequenzen

Die Bildverarbeitungspipeline wird mit dem 27 MHz Kamerataktsignal betrieben. Im Deserialisierer wird die Taktfrequenz des Bilddatenstroms durch Parallelisieren der Farbsignale auf 13,5 MHz abgesenkt, mit welcher die Bildverarbeitungspipeline bis zum Deinterlacer weiterarbeitet. In der projektiven Bildtransformation wird zur Beschleunigung der Division, die Taktfrequenz auf 54 MHz vervierfacht.

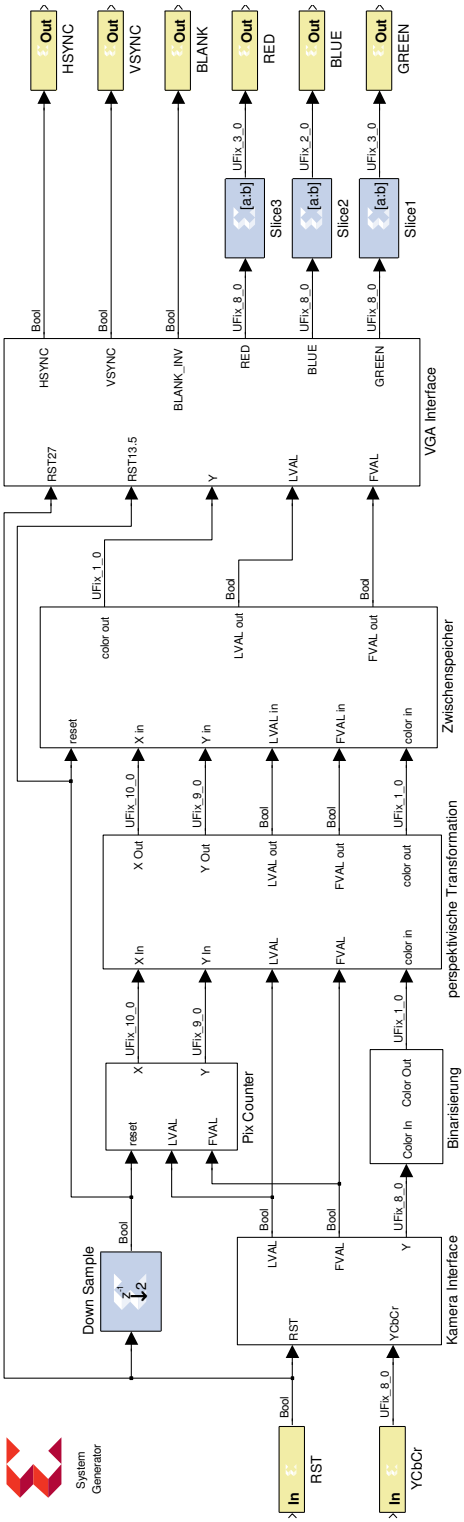


Abbildung 5.1.: Die Bildverarbeitungspipeline besteht aus drei Teilen: dem Kamera-Interface, den bildaufbereitenden Blöcken für die Fahrspurerkennung und dem VGA-Interface. (vgl. Abb. 1.3, Seite 9)

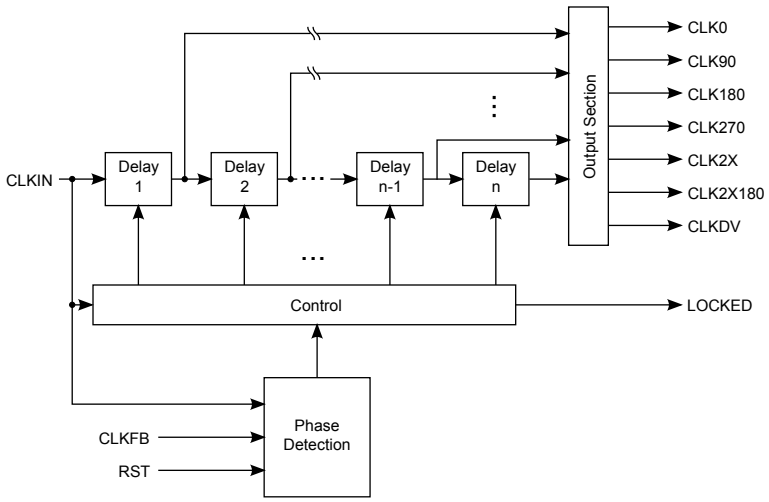


Abbildung 5.2.: Vereinfachtes Funktionsdiagram der „Delay-Looked Loop“ [Xilinx (2009b)]. Durch das Hintereinanderschalten von „Delay-Blöcken“ wird das Taktsignal verzögert. Über den Ausgangsmultiplexer (Output Selection) werden phasenverschobene Taktsignale aber auch verdoppelte (CLK2X und CLK2X180) oder reduzierte (CLKDV) Taktfrequenzen erzeugt. Das Reduzieren der Taktfrequenz lässt sich über einen Divisionsfaktor konfigurieren.

Für die Erzeugung von Taktfrequenzen, wie z. B. der Frequenzverdoppelung, welche nicht mit RTL-Logic zu implementieren sind (vgl. Xilinx, 2009b), enthalten die FPGAs „digitale Clock Manager“ (DCM), welche bei Xilinx mit „Delay-Locked Loop“ realisiert werden. In den „Delay-Locked Loop“ werden Logikelemente hintereinander geschaltet welche das Taktsignal verzögern (vgl. Abb. 5.2). Über einen Multiplexer werden aus den verzögerten Taktsignalen die unterschiedlichen Frequenzen erzeugt.

Die DCM des Spartan 3E wird nicht vom „System Generator“ unterstützt, weshalb eine externe Takterzeugung eingesetzt wird (vgl. Kap. 3.1). Das DCM-Modul wird mit dem „Core Generator“ von Xilinx erzeugt und über eine Toplevel-VHDL-Datei mit der Bildverarbeitungspipeline verbunden (vgl. Abb. 5.3).

5.2. Kamera-Interface

Die Kamera liefert das Video als seriellen Bilddatenstroms gemäß ITU Recommendation BT.601 und BT.656 (vgl. Sony). Der Bilddatenstrom nach BT.601 ist 16 Bit breit und wird über die Signale VSync und HSync synchronisiert. Der BT.656 Bilddatenstrom ist nur 8 Bit breit,

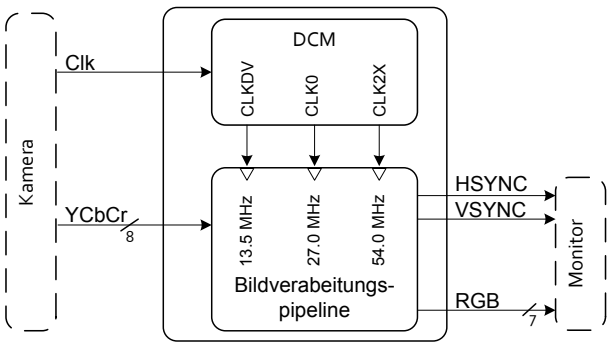


Abbildung 5.3.: Der Digital Clock Manager generiert aus der Kameratakfrequenz die drei Taktfrequenzen für die Bildverarbeitungspipeline. Die mit dem Core Generator erzeugten DCM wird mit einer TopLevel-VHDL-Datei an die Bildverarbeitungspipeline angeschlossen.

in ihm werden neben den Synchronisationssignalen in den Bilddatenstrom eingebettete Synchronisationsmarkierung übertragen. Um die gleiche Datenmenge, wie beim 16 Bit breiten Datenstrom zu übertragen, ist die Taktfrequenz beim 8 Bit breiten Datenstrom von 13,5 MHz auf 27 MHz verdoppelt. Der 8 Bit breite Datenstrom steht neben dem progressiven Format (Vollbild) auch als interlaced (Halbbild) zur Verfügung (vgl. Tab. 5.1).

Modus	Busbreite	Syncsignale	ITU Rec.	Clock
16bit Progressive	16 bit	HSYNC/VSYNC	601	13,5 MHz
8bit Progressive	8 bit	HSYNC/VSYNC, SAV/EAV	656	27 MHz
8bit Interlace	8 bit	HSYNC/VSYNC, SAV/EAV	656	27 MHz

Tabelle 5.1.: Datenübertragungsmodi der Kamera Sony FCB-PV10 [Sony]

Die Kamera liefert 29,5 Bilder/sec. mit einer Auflösung von 640 x 480 Bildpunkten. Für die Darstellung von Bildern auf einem VGA-Monitor schreibt der VESA-Standard eine Bildwiederholfrequenz von mindestens 60 Hz vor (vgl. Kap. 5.7, Seite 66). Zur Verdoppelung der Bildwiederholfrequenz werden die Bildzeilen des interlaced Datenstroms verdoppelt und mit geringem Aufwand aus einem Halbbild ein Vollbild gemacht, was die Bildwiederholfrequenz auf 59 Hz erhöht (vgl. Kirschke (2009) / Peters (2009)). Um die Bildwiederholfrequenz eines Vollbildes des progressiv Datenstroms zu erhöhen ist dieser zwischenzuspeichern und zweimal zum Monitor zu senden. Für den Zwischenspeicher werden so viele Logikelemente und BRAM-Zellen verbraucht, dass das Modul nicht mehr in den verwendeten FPGA passt.

Ein Bild des 480i (640 x 480 Bildpunkte interlaced) Datenstroms besteht aus zwei Halbbildern, welche durch Blankabschnitte voneinander getrennt sind (vgl. Abb. 5.4). Anfang und Ende eines Halbbildes, ohne die Blankabschnitte, wird innerhalb des FPGA durch das Framevalid-

Abbildung 5.4.: Bildaufbau eines Kamerabildes im 480i-Datenstrom. Die Halbbilder und die Bildeilen sind durch Blankabschnitte voneinander getrennt. (Jack, 2005, Abb. 4.8, Seite 42)

Signal FVAL markiert (vgl. Abb. 5.5). Gleiches gilt auch für die einzelnen Bildzeilen. Auch sie sind durch Blankabschnitte voneinander getrennt und ihr Anfang und Ende wird durch das Linevalid-Signal LVAL markiert.

Das Kamera-Interface besteht aus den Funktionsblöcken „Synchronisations Generator“ und Deserialisierer (vgl. Abb. 5.6). Der „Synchronisations Generator“ erzeugt aus dem Synchronisationsmarkierung im Bilddatenstrom die Synchronisationssignal FVAL und LVAL. Im Deserialisierer werden die seriell übertragenen Farbinformationen in parallele Signale aufgeteilt.

Der Videodatenstrom verwendet das YC_BC_R-Farbmodell, dieses ist mit dem YUV-Farbmodell verwand, welches im analogen PAL-Fernsehen eingesetzt wird. Das YC_BC_R-Farbmodell wird im digitalen PAL-Fernsehen eingesetzt und zur Datenkompression in JPEG-Bildern oder MPEG-Videos verwendet. Das YC_BC_R-Farbmodell spaltet das Farbbild in drei Bildinformationsteile, welche nicht wie beim RGB-Farbmodell die drei Grundfarben Rot, Gelb und Blau sind, sondern die Helligkeit (Y), was dem Graustufen-Bild entspricht, sowie den Farbdifferenzen Blau-Gelb (C_B) und Rot-Grün (C_R) (vgl. Abb. 5.7).

Nach dem ITU BT.565 Standard werden im seriellen 8 Bit Datenstrom abwechselnd ein Farbdifferenzwert und ein Helligkeitswert übertragen (vgl. Abb. 5.8). Die beiden Farbdifferenzen C_B und C_R wechseln sich ab. Es steht also zu jedem Helligkeitswert nur einer der beiden Farbdifferezwerte zur Verfügung.

Für die Weiterverarbeitung des Bildes zu einem Binärbild wird nur die Helligkeitsinformation

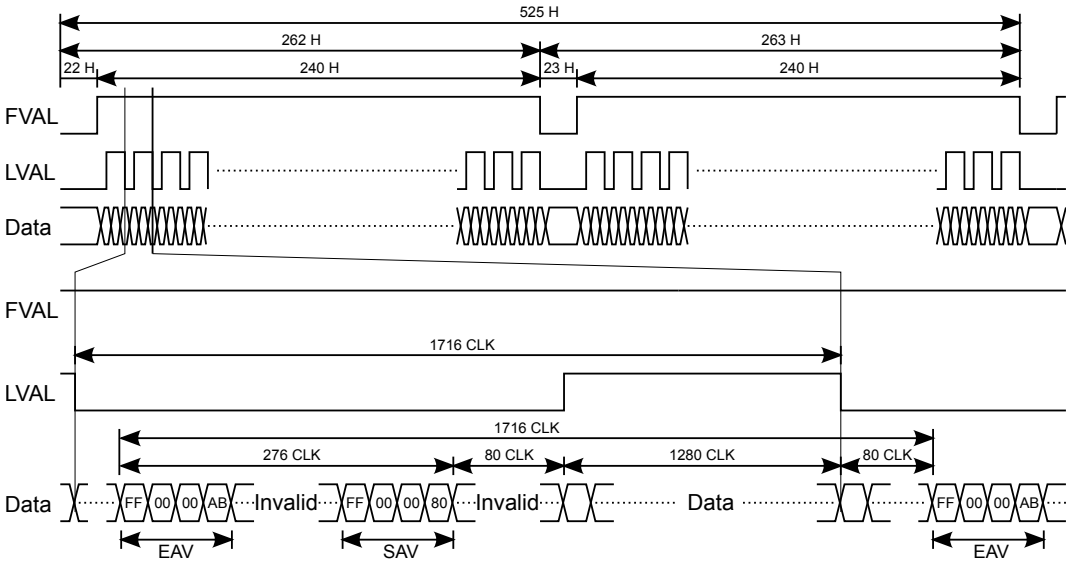


Abbildung 5.5.: Timingdiagram für die Kamera Sony FCB-PV10 und die FVAL- und LVAL-Signale.

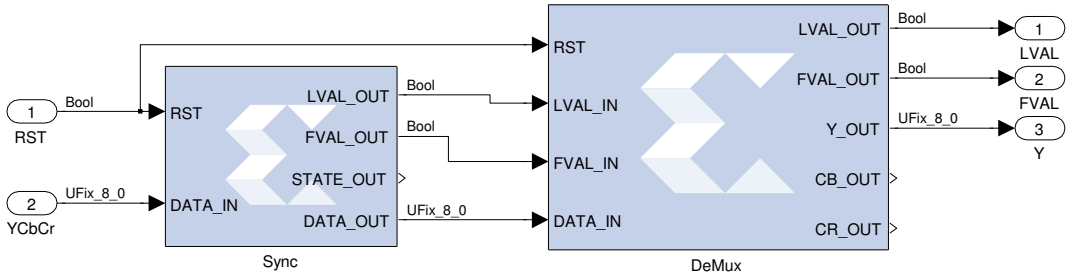


Abbildung 5.6.: Das Kamera-Interface besteht aus zwei Funktionsblöcken. Der Sync-Block gewinnt aus dem Bilddatenstrom die Synchronisationssignale FVAL und LVAL. Der DeMux-Block deserialisiert den Bilddatenstrom in ein paralleles YC_BC_R-Signal.

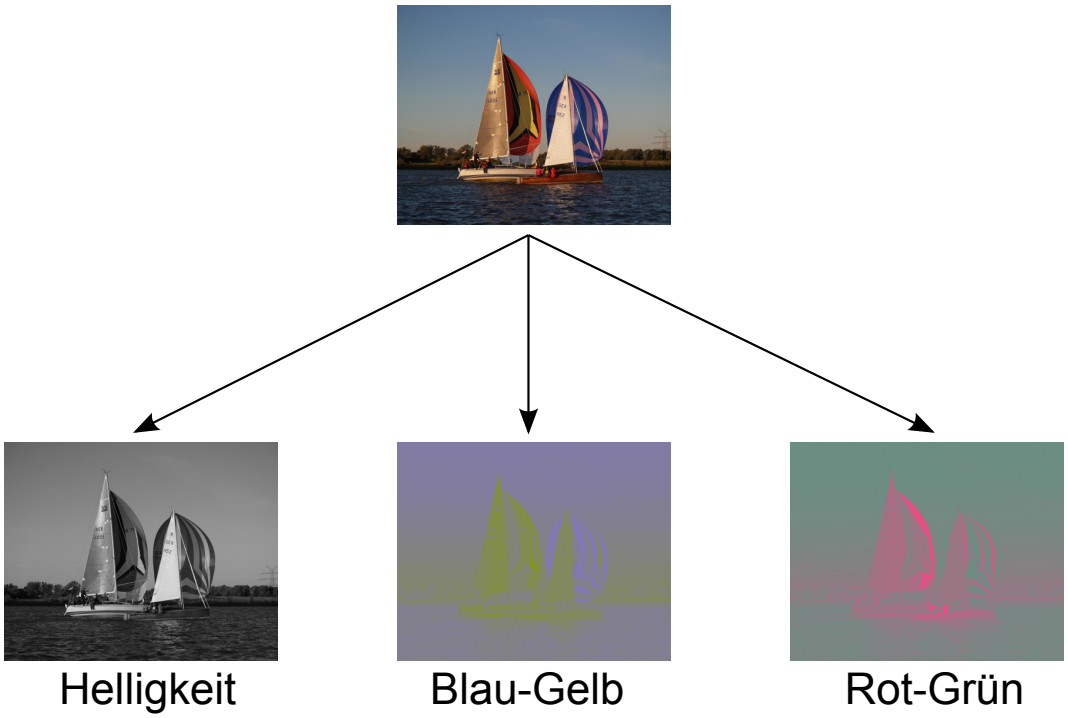


Abbildung 5.7.: Aufspaltung eines Bildes in die Helligkeit (Liminanz) Y und Farbdifferenzen (Chrominanz) C_B und C_R.

Abbildung 5.8.: Im seriellen 8 Bit Videodatenstrom werden zu jedem Helligkeitswert abwechselnd einer der beiden Farbdifferenzwerte übertragen (vgl. ITU, 2007, Kap 2.2).

Bit Nr.		Name	Wert	Funktion
7	MSB		1	
6		F	0	erstes Halbbild
			1	zeites Halbbild
5		V	0	sichtbarer Bildberech
			1	Blankingbereich
4		H	0	SAV
			1	EAV
3		P3	V XOR H	Schutzbits
2		P2	F XOR H	
1		P1	F XOR V	
0	LSB	P0	F XOR V XOR H	

Tabelle 5.2.: Funktion der Bits des letzten Bytes der Bildsynchronisationsmarkierung im Bilddatenstrom (vgl. ITU, 2007, Kap. 2.4).

verwendet, weshalb kein Upsampling-Funktionsblock, der das Abtastverhältnis von 4:2:2 nach 4:4:4 erhöht, im Kamera-Interface enthalten ist (vgl. Peters, 2009, Kap. 3.5).

5.2.1. Erzeugung der Synchronisationssignale aus den Synchronisationsmarkierungen im Kameradatenstrom

Wie bereits oben erwähnt stehen für die Bildsynchronisation sowohl die Signale VSync und HSync sowie die in den Bilddatenstrom eingebetteten SAV und EAV Markierungen zur Verfügung. Um für zukünftige Erweiterungen Pins am FPGA frei zu halten, werden die in den Bilddatenstrom eingebetteten Synchronisationsmarkierungen genutzt.

Die Synchronisationsmarkierungen SAV (start of active video) und EAV (end of active video) bestehen aus einer Folge von vier Bytes, die aufgrund des Wertebereiches der Bilddaten, niemals im Bilddatenstrom vorkommen. Jede Synchronisationsmarkierung beginnt mit 0xFF 0x00 0x00, worauf ein Byte folgt, welches die Funktion der Markierung angibt (vgl. Tab. 5.2). Des weiteren ist in diesem Byte die Information, ob es sich um das erste oder das zweite Halbbild bei einer interlaced Übertragung handelt, und eine Prüfsumme enthalten.

Jede Bildzeile beginnt mit der Synchronisationsmarkierung SAV und endet mit EAV. Befindet sich die Bildzeile in der vertikalen Austastlücke, ist das Blankbit gesetzt. Zum Finden der ersten Bildzeile wird auf ein Blank-SAV folgendes SAV gewartet (vgl. Abb. 5.10). Alternativ kann statt dem Blank-SAV auch das Blank-EAV verwendet werden. Dies hätte den Vorteil das in manchen Fällen beim Einschalten sich die Bildverarbeitungspipeline ein Bild früher auf das Kamerabild synchronisieren kann. Dieser Fall tritt ein, wenn sich die Kamera beim Einschalten schon in

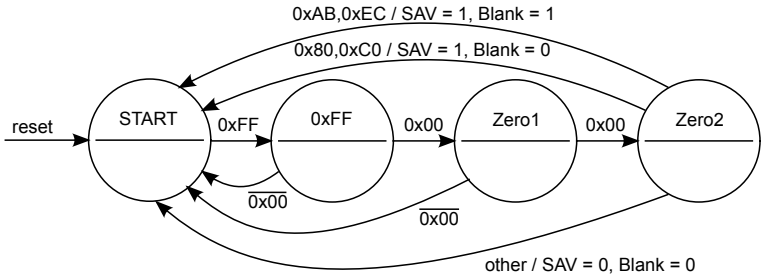


Abbildung 5.9.: Mealy-Zustandsautomat zum Erkennen der SAV und Blank-SAV im Bilddatenstrom (vgl. Kirschke, 2009, Abb. 15). Die Zustandsübergange erfolgen anhand des letzten Bytes der Synchronisationsmarkierung (vgl. Tab. 5.2).

der letzten Blankzeile befindet, also das Blank-SAV schon übertragen wurde. Das erkennen des Blank-EAV benötigt im Zustandsautomaten (vgl. Abb. 5.9) einen zusätzlichen Zustandsübergang. Die Einschaltgeschwindigkeit ist nicht kritisch, weshalb auf die für das Erkennen von Blank-EAV zusätzliche Logik verzichtet wurde.

Anders als im Standard ITU BT.656 beschrieben folgt auf die Startmarkierung nicht unmittelbar der Bilddatenstrom. Dieser fängt erst mit einer Verzögerung von 80 Takten an (vgl. Abb. 5.5). Auch am Ende des Bilddatenstroms jeder Zeile gibt es den Abstand von 80 Takten zur Endmarkierung.

Aufgrund der Verzögerung von EAV werden für den Zustandsübergang von „Line“ nach „Line Blank“ die Datenbytes mitgezählt. Der Zählerstand für den Zustandsübergang ist doppelt so hoch wie die Anzahl der Bildpunkte in einer Zeile. Dies liegt am verwendeten Dateiformat, welches nur in jedem zweiten Byte einen Grauwert enthält. Ohne das Mitzählen würde der Zustandsübergang 84 Takte zu spät stattfinden.

5.2.2. Demultiplexer

Im Demultiplexer werden die seriell übertragenen Bilddaten YC_BC_R in einen parallelen Bildstrom Y, C_B und C_R zerlegt. Die Farbdifferenzen C_B und C_R werden abwechselnd mit dem Helligkeitswert übertragen. Der Datenstrom hat das folgende Format: $C_{B_0}, Y_0, C_{R_0}, Y1, C_{B_2}, Y_2, C_{R_2}\ldots$ (vgl. Abb. 5.8). Es steht nur für jeden zweiten Helligkeitswert die beiden Farbdifferenzen zur Verfügung. Fehlende Farbdifferenzen werden mit dem Wert 0 ergänzt (vgl. Kirschke, 2009, Kap. 3.1.3).

Das Demultiplexen des Datenstroms erfolgt mit einem Zustandsautomaten (vgl. Abb. 5.11), welcher aus 5 Zuständen besteht. Ein Startzustand, zwei Zuständen für den Helligkeitswert und je ein Zustand für die Farbdifferenz.

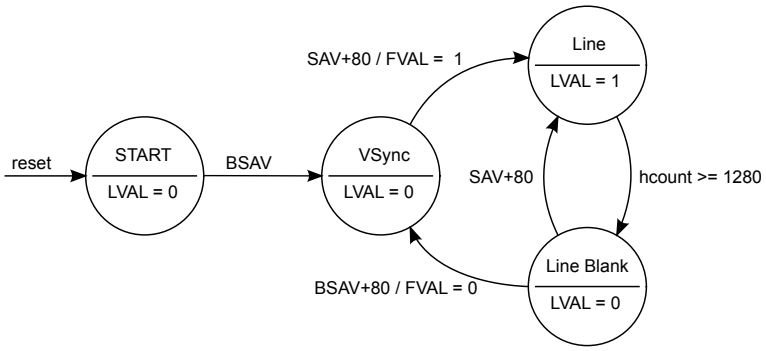


Abbildung 5.10.: Mit den in den Bilddatenstrom eingebetteten Zeilenanfangsmarkierungen SAV werden die Bildsynchronisationssignale FVAL und LVAL generiert. Der Anfang des Bildes wird durch das Wegfallen des Blankflags in der Zeilenanfangsmarkierung erkannt. Das Bildende wird durch das gesetzte Blankflag erkannt.

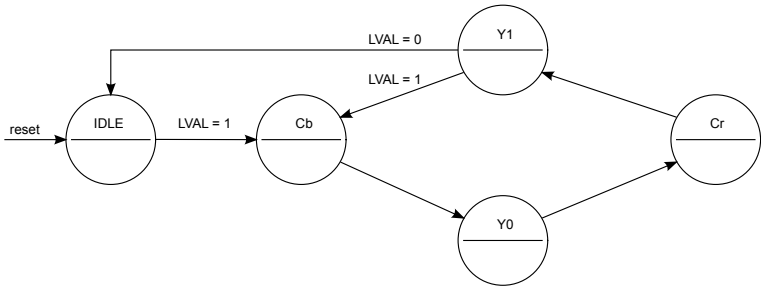


Abbildung 5.11.: Zustandsautomat zum demultiplexen des YC_BC_R-Datenstroms. Der Automat startet sobald $LVAL = 1$ gültige Bilddaten signalisiert und läuft solange bis mit $LVAL = 0$ das Ende der Bildzeile erreicht ist. (vgl. Kirschke, 2009, Abb. 17)

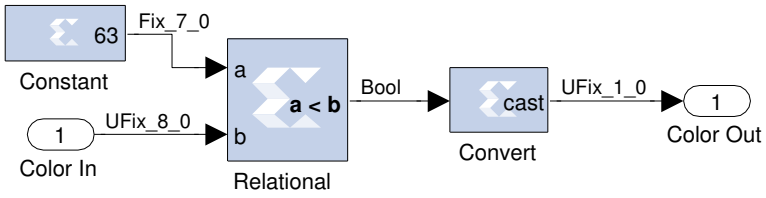


Abbildung 5.12.: Das Graustufenbild wird mit einem festen Schwellwert in ein Binärbild verwandelt.

Die YC_BC_R-Werte werden in Registern vor der Weitergabe zwischengespeichert. Durch das deserialisieren des Kameradatenstroms verringert sich die Ausgangstaktfrequenz von 27 MHz auf 13,5 MHz.

5.3. Binarisierung

Für das Binärbild werden die Helligkeitswerte Y genutzt. Die Farbdifferenzen werden nicht verwendet, weshalb die fehlenden Farbdifferenzen nicht rekonstruiert werden. Die Binarisierung findet mit einem festen Schwellwert 63 statt, welcher experimentell ermittelt wurde und auf kein Einsatzgebiet optimiert ist, dies kann erst nach der Montage der Kamera auf dem Fahrzug erfolgen, da erst dann die Parameter Kameraneigungswinkel, Bildausschnitt und Versuchsstrecke bekannt sind.

Für den Einsatz mit wechselnden oder unbekannten Lichtverhältnissen ist ein fester Schwellwert nicht geeignet. Für dieses Einsatzgebiet ist der Binarisierungs-Block um einen dynamischen Schwellwert, welcher z. B. mittels Histogrammausgleich für einen hohen Kontrast zwischen Straße und Fahrbahnmarkierung sorgt, zu ergänzen. Für die Ermittlung des Schwellwertes ist zu beachten, dass der Wertebereich des Helligkeitswertes Y zwischen 16 und 235 liegt (vgl. Sony).

Der Bilddatenstrom wird als 1-Bit-Vektor weitergereicht, um die weiterverarbeitenden Funktionsblöcke unabhängig von der Vektorbreite entwickeln zu können. So ist es jederzeit möglich den Binarisierungs-Block zu entfernen, ohne alle Funktionsblöcke anpassen zu müssen.

5.4. Bildpunktkoordinatengenerator

Die projektive Transformation des Bildes, aber auch die Hough-Transformation, arbeiten mit den Koordinaten der Bildpunkte im Bild (vgl. Abb. 1.2, Seite 9). Das Koordinatensystem zählt

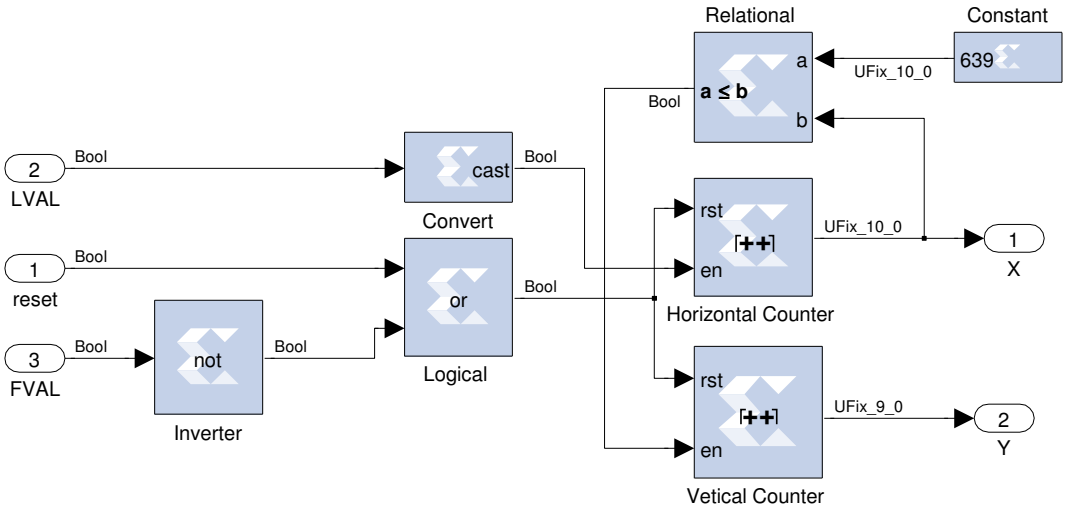


Abbildung 5.13.: Für die perspektivische Transformation werden die X/Y-Koordinaten der Bildpunkte benötigt. Für jede der beiden Koordinaten ist ein Zähler zuständig. Sobald mit LVAL=1 die Bildzeile anfängt, beginnt der „Horizontal Counter“ die X-Koordinate zu zählen. Erreicht der „Horizontal Counter“ mit seinem maximalen Zählerstand von 639 das Ende der Bildzeile, wird der „Vertikal Counter“ aktiviert und zählt die Zeile um 2 weiter. Der maximale Zählerstand des „Vertikal Counters“ ist 478.

die Bildpunkte von links oben nach rechts unten. Die x-Koordinate gibt die Position des Bildpunktes in der Zeile an, die y-Koordinate die Zeile. Im Koordinatensystem ist 0 die niedrigste Koordinate. In der Horizontalen ist 639 die höchste Koordinate, in der Vertikalen ist es 478.

Für jede der beiden Koordinaten (x,y) wird ein Zähler verwendet, welcher vom Begin des Bildes bei 0 anfängt zu zählen. Der Zahler für die horizontale Koordinate X wird vom LVAL-Signal aktiviert. Liegen gültige Bilddaten an wird mit jedem Takt eine Position weitergezählt. Erreicht der horizontale Zähler den Stand 639, wird der vertikale Zähler aktiviert und zählt mit dem folgenden Takt die vertikale Position um zwei hoch. Im selben Takt läuft der horizontale Zähler über, d. h., er wird auf 0 zurückgesetzt.

Der vertikale Zähler zählt in Zweierschritten, da die Transformationsmatrix auf ein Bild mit von 640 x 480 Bildpunkten berechnet ist, wir aber nur ein Halbbild mit 640 x 240 Bildpunkten haben.

Das FVAL-Signal setzt invertiert am Ende von jedem Bild die Zähler wieder auf 0 zurück. Dies stellt sicher, dass auch bei unvollständigen Bildern das Folgebild korrekt gezählt wird.

5.5. Projektive Bildtransformation

Durch die projektive Bildtransformation wird die perspektivische Verzerrung des Bildes korrigiert (vgl. Kap. 2, Seite 15).

In dem Funktionsblock der projektiven Bildtransformation ist die Gleichung 5.1 unter Verwendung der Transformationsmatrix (Gl. 2.2) für die Vorwärtstransformation implementiert (vgl. Kap. 2.1, Seite 17).

$$x' = \frac{h_{11}x + h_{12}y + h_{13}}{h_{31}x + h_{32}y + h_{33}} \qquad y' = \frac{h_{21}x + h_{22}y + h_{23}}{h_{31}x + h_{32}y + h_{33}} \tag{5.1}$$

Im ersten Schritt werden die beiden Dividenden $h_{11}x + h_{12}y + h_{13}$ und $h_{21}x + h_{22}y + h_{23}$ sowie der Divisor $h_{31}x + h_{32}y + h_{33}$, welcher für beide Teile der Gleichung gleich ist, berechnet. Die Berechnung dieser drei Teile unterscheidet sich nur durch die Elemente h_n der Maxtrix. Die Implementierung besteht jeweils aus zwei Multiplizierern sowie zwei Additionen (vgl. Abb. 5.16).

Im nächsten Schritt wird im Divider für den Divisor der Kehrwert $\frac{1}{h_{31}x+h_{32}y+h_{33}}$ berechnet. Durch die Berechnung Kehrwertes kann das x' und y' im nächsten Schritt mit jeweils einer Multiplikation ausgerechnet werden. Die Division ist im FPGA am aufwendigsten zu implementieren und wurde aus diesem Grund in eine Einzelrechnung ausgelagert.

Für die Division wird der „Divider Generator“-Funktionsblock vom „System Generator“ benutzt (vgl. Abb. 5.17). Auf FPGAs ohne DSP48-Hardwareblock ist die Division nach dem Radix-2 Algorithmus implementiert (vgl. Xilinx, 2009a). Der Radix-2 Algorithmus löst mit Additionen und Subtraktionen ein Bit des Quotienten pro Takt (vgl. Mellert, 2010). Durch Einsatz des Pipelineverfahrens wird ein Durchsatz von einer Division pro Takt erreicht. Das Divisionsergebnis besteht aus dem Integeranteil des Quotienten sowie dem Integer-Rest der Division oder dem Fractional-Anteil.

Zur Beschleunigung der Division wird sie mit der vierfachen Taktfrequenz durchgeführt. Dazu wird die Taktfrequenz mit einem Upsample-Funktionsblock vor dem „Divider Generator“ angehoben und nach der Division mit den Downsample-Funktionsblöcken wieder abgesenkt (vgl. Abb. 5.17(b)).

Der Divider Generator akzeptiert nur integer Divisoren. Mit dem Reinterpret-Funktionsblock wird die Festkommazahl als integer interpretiert. Nach der Division wird das Ergebnis aus Quotienten und Fractional-Anteil zusammengesetzt, und wieder als Festkommazahl interpretiert.

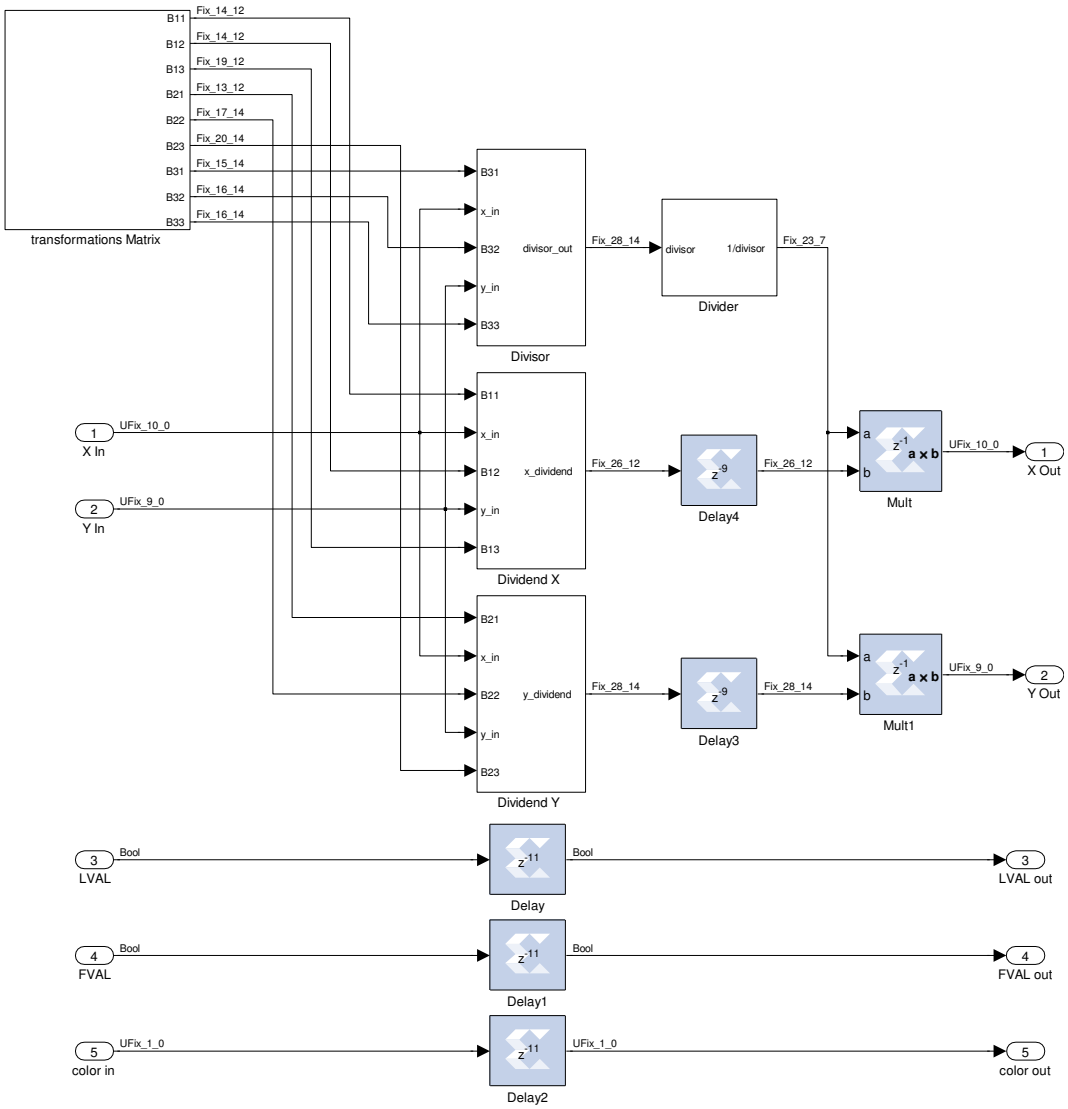


Abbildung 5.14.: Die projektive Transformation mit Transformationsmatrix H, Divisor- und Dividendenberechnung sowie Multiplizieren für die Kehrwertmultiplikation (vgl. Mellert, 2010, Abb. 27).

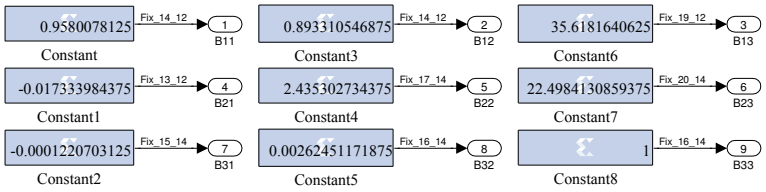


Abbildung 5.15.: Transformationsmatrix H (vgl. Gl. 2.2, Seite 18) als Festkommazahlen (vgl. Mellert, 2010, Abb. 28)

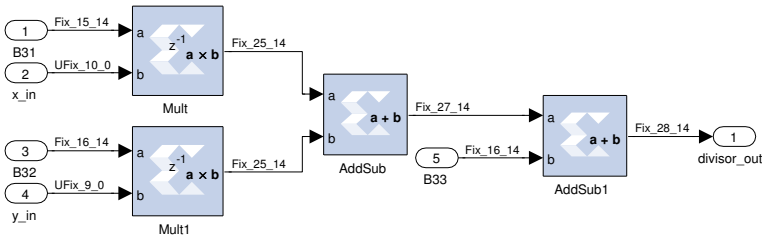
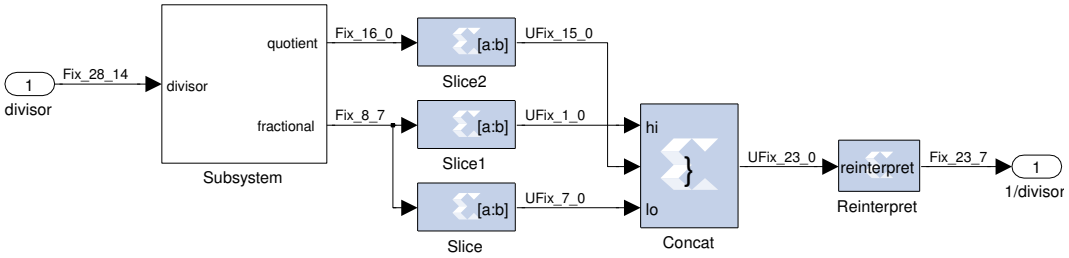
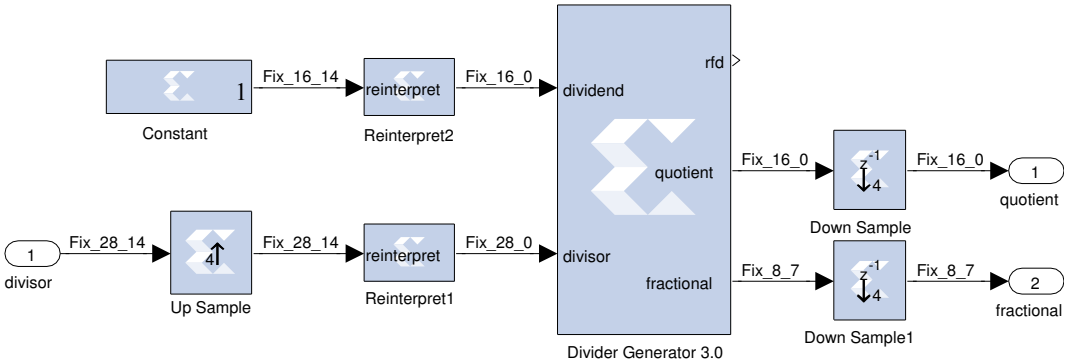


Abbildung 5.16.: Die Subsystemfunktionsblöcke Divider, Dividend x und Dividend y (vgl. gesamt) sind identisch aufgebaut. In ihnen werden die X/Y-Koordinaten mit jeweils den ersten beiden Elementen einer Zeile der Matrix multipliziert. Das Ergebnis wird mit dem letzten Element der Matrixzeile summiert. (vgl. Mellert, 2010, Abb. 28)



(a) Nach der Division wird das Ergebnis als Festkommazahl aus dem ganzzahligen Quotienten und dem gebrochenen Anteil zusammengesetzt. Vorzeichen des Bruchs, ganzzahliger Quotient, Nachkommastellen.



(b) Der Dividend muss die gleiche Anzahl von Nachkommastellen haben wie der Divisor. Nur so ist eine Verschiebung bei Festkommazahlen im Q-Format für die Division möglich. Die Division wird im „Divider Generator"-Funktionsblock in 29 Takten durchgeführt. Durch die Erhöhung der Taktfrequenz auf 54MHz entspricht dies 8 Takten im Bilddatenstrom.

Abbildung 5.17.: Berechnung des Kehrwertes des Divisors für die Multiplikation mit den Dividenden (vgl. Mellert, 2010, Abb. 29).

5.6. Bildstromrekonstruktion

Aus der projektiven Bildtransformation kommen die Bildpunkte in unsortierter Reihenfolge (vgl. Kap. 2, Seite 15). Für Darstellung des projektiv transformierten Bildes auf einem VGA-Monitor stellt der Zwischenspeicher den Bilddatenstrom wieder her (vgl. Kap. 4, Seite 30).

Die Bildstromrekonstruktion besteht aus drei Teilen der Speicheradressberechnung aus den X/Y-Zielkoordinaten der projektiven Bildtransformation zum Schreiben in den Zwischenspeicher, der Generierung der Synchronisationssignale für den ausgehenden Bilddatenstrom und dem Zwischenspeicher (vgl. Abb. 5.18).

5.6.1. Bildzwischenspeicher

Die Rekonstruktion des Bildstromes basiert auf den Ergebnissen der Analyse aus Kap. 4 (ab Seite 30). Anders als in der Analyse vorgeschlagen kommt ein Ringbuffer, statt dem Schieberegister, zum Einsatz. Der Ringbuffer wurde gewählt weil das Schieberegister, welches im „System Generator“ als Funktionsblock mitgeliefert wird, mit einer maximalen Tiefe von 1024 nicht groß genug für die zwischenzuspeichernde Datenmenge ist.

Der Ringbuffer wird mit dem Dual-Port-RAM-Funktionsblock implementiert, dessen Größe sich an den Zweierpotenzen orientiert. In der Analyse (vgl. Kap. 4.2 Seite 33) wurde für die Vorwärtstransformation eine maximale Sprungweite von 148 Zeilen ermittelt. Da wir mit Halbbildern arbeiten, ergibt sich eine Sprungweite von $148/2 = 74$ Zeilen.

Der zeitliche Versatz zwischen Lese- und Schreibprozess stellt sicher, dass keine Bildpunkte in einen Bereich geschrieben werden, der bereits ausgelesen wurde. Als minimaler zeitlicher Versatz zwischen Schreiben und Lesen wurde experimentell ein Abstand von 5 Bildzeilen ermittelt. Ist der Versatz kleiner als 5 Bildzeilen, entstehen Lücken in den obersten Bildzeilen (vgl. Abb. 5.19).

Die 5 Zeilen Versatz zwischen Lese- und Schreibprozess addieren sich zu den 74 Zeilen für die Sprungweite. Es sind also insgesamt $74 + 5 = 79$ Zeilen zwischenzuspeichern.

Von den aus den x und y-Koordinaten errechneten Speicheradressen werden die vordersten beiden Bits (MSB) abgeschnitten. Durch das Abschneiden der vordersten Bits gibt es für jeden physikalischen Speicherbereich 4 mögliche Adressen. Für die Adressierung ist die Grenze zwischen physikalischem Speicherende und Anfang nicht sichtbar. Ein Ring entsteht. Damit sich die physikalische Speichergröße an der Anzahl der Adressleitung orientiert, muss sie sich nach den Zweierpotenzen richten.

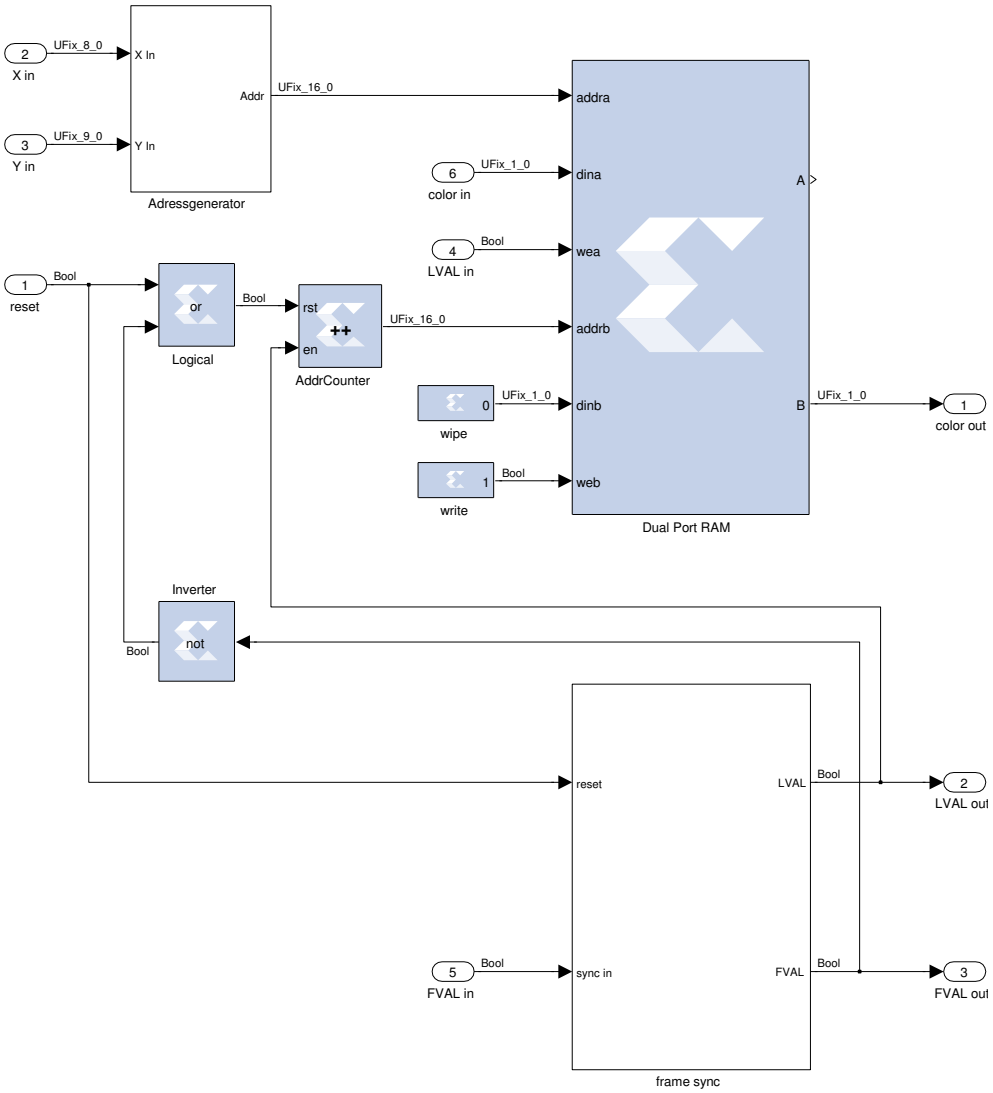
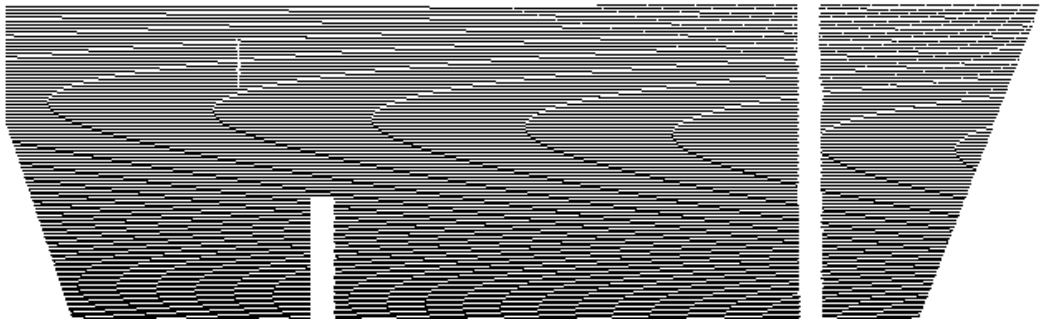
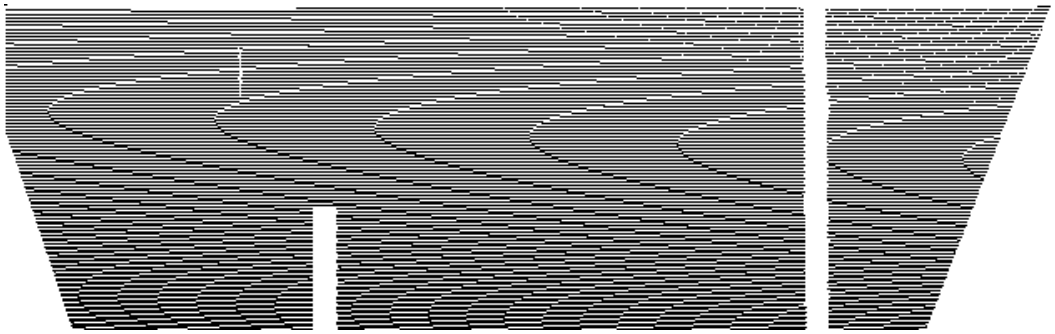


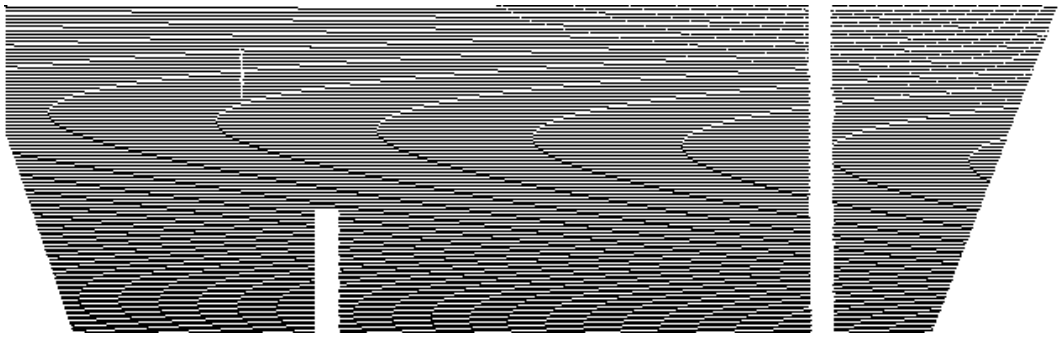
Abbildung 5.18.: Die Bildstromrekonstruktion besteht aus drei Teilen: der Berechnung der Speicheradresse aus den X/Y-Koordinaten, dem „Dual Port RAM für die Zwischenspeicherung der Bildpunkte und der Regeneration der Bildsynchronisationssignale FVAL und LVAL. Port A schreiben; Port B lesen. Die Bildpunkte aus der projektiven Transformation werden an die vom Adressgenerator berechnete Speicheradresse geschrieben wenn LVAL=1 gültige Bilddaten signalisiert. Der Adresszähler für das Lesen der Bildpunkte aus dem Speicher ist ein freilaufender 16-Bit-Zähler. Der Adresszähler wird über die FVAL- und LVAL-Signale für den ausgehenden Bilddatenstrom synchronisiert.



(a) Lesen ohne zeitlichen Versatz zum Schreiben

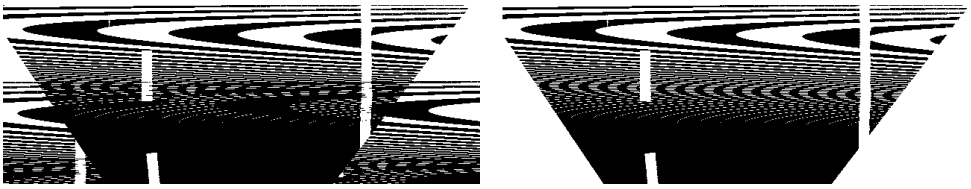


(b) Lesen mit einem Versatz von 2 Bildzeilen zum Schreiben



(c) Lesen mit einem Versatz von 5 Bildzeilen zum Schreiben

Abbildung 5.19.: Ist der Abstand zwischen Schreibe- und Leseprozess zu gering, gehen Bildinformationen verloren (vgl. erste Bildzeilen). Bilder wurden in einer ModelSim Simulation des Zwischenspeichers mit Eingangdaten aus einer „System Generator“ Simulation der projektiven Transformation erzeugt.



(a) ohne Löschen von bereits gelesenen Speicherbereichen

(b) mit Löschen von bereits gelesenen Speicherbereichen

Abbildung 5.20.: Vorwärtstransformiertes Bild einer Straße. Bilder wurden in einer ModelSim Simulation des Zwischenspeichers mit Eingangdaten aus einer „System Generator“ Simulation der projektiven Transformation erzeugt.

Zum Zwischenspeichern von 79 Zeile werden 79 Zeilen * 640 Bildpunkt * 1 Bit Farbtiefe = 50560 Speicherplätze für die einzelnen Bildpunkte benötigt. Die nächste Zweierpotenzgrenze ist $2^{16} = 65536$, sie passen in 102,4 Bildzeilen.

In der projektiven Vorwärtstransformation wird nicht jeder Bildpunkt von der Transformationsmatrix erreicht (vgl. Kap. 2.1). Es entstehen Lücken im Zielbild, damit sie nicht nicht Daten von vorranggegangenen Teilbildern enthalten (vgl. Abb. 5.20(a)), werden die Daten nach jedem Lesevorgang überschrieben (vgl. Abb. 5.20(b)). Um dieses zu erreichen, ist der Port B des Dual-Port-RAMs auf „Write after Read“ konfiguriert. Als Bilddaten liegt am Port B eine konstante 0 an. Der „Write Enable“-Eingang des Port B ist konstant 1.

5.6.2. Speicheradressgenerator

Die Adressierung des Zwischenspeichers beginnt für jedes Bild des Videodatenstroms bei der Adresse 0. Die Adresse eines Bildpunktes im Speicher errechnet sich wie folgt aus den X/Y-Koordinaten: $(y - 31) * 640 + (x - 65)$.

Die Matrix der projektiven Transformation ist für ein Bild mit einer Auflösung von 640 x 480 Bildpunkten berechnet. Für die Zurückgewinnung des Halbbildes wird Anzahl der Bildzeilen durch Schieben der Bits um eine Position nach rechts halbiert.

Durch den Streckungsfaktor der Transformationsmatrix hat das Bild eine Größe von 692 x 271 Bildpunkten. Zur Reduzierung des Bildausschnittes auf eine Auflösung von 640 x 240 Bildpunkten wird in die x-Koordinaten ein Offset von -65 Bildpunkten und in die y-Koordinaten ein Offset von -31 Bildzeilen eingerechnet (vgl. Abb. 5.22). Die ersten 31 Zeilen eines Bildes bzw. die ersten 65 Bildpunkte einer Zeile werden durch die Adresse 0 ersetzt. Die Bildpunkte, die unten aus dem Bild hinauslaufen, werden zwar in den Speicher geschrieben aber nicht wieder gelesen. Dadurch, dass Sie nicht gelesen werden, werden sie

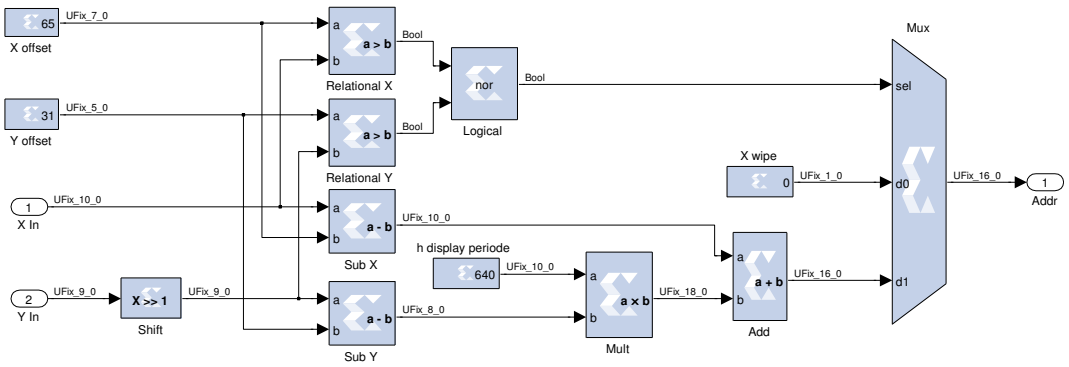


Abbildung 5.21.: Im Adressgenerator wird aus der X/Y-Koordinate des Bildpunktes eine Speicheradresse berechnet. Die Speicheradressen werden nach der Gleichung: $(y - 31) * 640 + (x - 65)$ berrechnet. Koordinaten, deren X-Anteil kleiner ist als 65 oder deren Y-Anteil kleiner als 31 werden auf die Speicheradresse 0 abgebildet.

nicht gelöscht und können als Schattenbilder das Bild verunreinigen. Bildpunkte, die über den rechten Rand hinausgehen, werden in den meisten fällen von einem nachfolgenden Bildpunkt überschrieben, es können aber auch Bildfehler auftreten. In der Simulation konnten keine Bildfehler entdeckt werden, die das Ergebnis beeinträchtigen.

5.6.3. Bildsyncornisation

Durch die Verzögerung des Bilddatenstroms im Zwischenspeicher sind auch die Synchronisationssignale FVAL und LVAL entsprechend um 5 Bildzeilen zu verzögern. Die beiden Synchronisationssignale um $5 * 857 = 4285$ Takte mit Registern zu verzögern hatte einen doppelt so großen Hardwareaufwand zur Folge wie das Speichern der 1-Bit-Bildinformation. Aus diesem Grund werden die Synchronisationssignale für den ausgehenden Bilddatenstrom des Zwischenspeichers mit Zeilen- und Spaltenzählern neu generiert.

Das Generieren der Bildsynchronisationssignale geschieht ähnlich wie beim Bildkoordinatengenerator vor der projektiven Transformation mit zwei Zählern (vgl. Kap. 5.4 Seite 53). Jeweils einen Zähler für die Horizontale und einen für die Vertikale (vgl. Abb. 5.23). Die Wertebereiche der Zähler sind größer als beim Bildkoordinatengenerator für die projektive Transformation, da nicht nur die sichtbaren Bildpunkte zählen, sondern auch die unsichtbaren zwischen den Bildzeilen und Bildern (vgl. Abb. 5.4, Seite 47).

Die Signale FVAL und LVAL werden von jeweils einem Zustandsautomaten generiert. Für das FVAL-Signal ist die „Vertikal FSM“ zuständig. Sie synchronisiert sich auf den Anfang des ersten

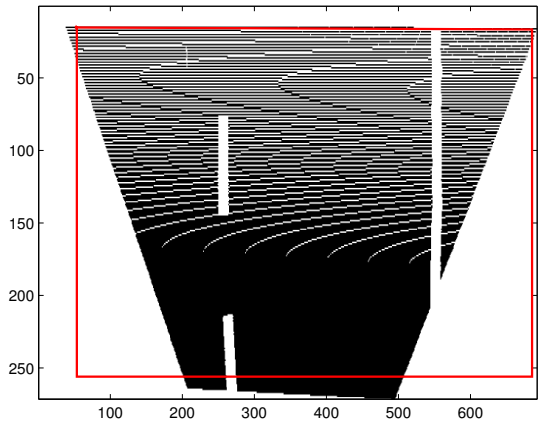


Abbildung 5.22.: Nach der perspektivischen Transformation ist das Bild durch den Streckungsfaktor der Matrix größer. Durch Abschneiden der Bereiche außerhalb des roten Rahmens wird es wider auf seine ursprüngliche Auflösung von 640x480 Bildpunkten gebracht.

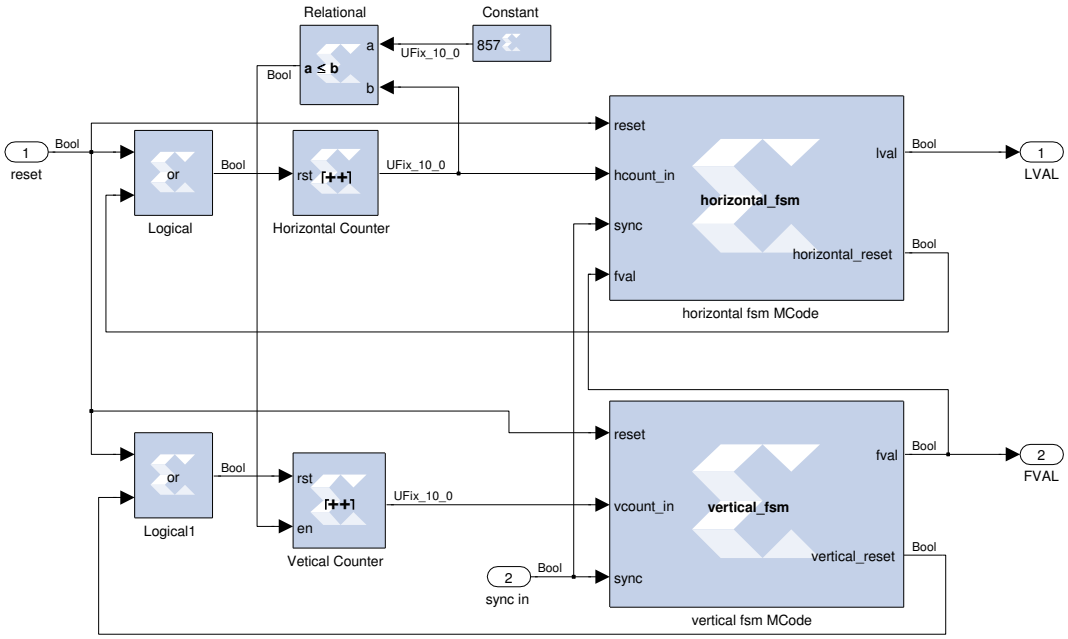


Abbildung 5.23.: Für die Erzeugung der Bildsynchronisationssignale FVAL und LVAL werden wie bei beim Bildkoordinatengenerator zwei Zähler verwendet. Der „Horizontal Counter“ zählt bis 857 und der „Vertical Counter“ bis 524. Die Funktion der Zustandsautomaten „Horizontal FSM“ und „Vertical FSM“ sind in Abb. 5.24 und 5.25 dargestellt.

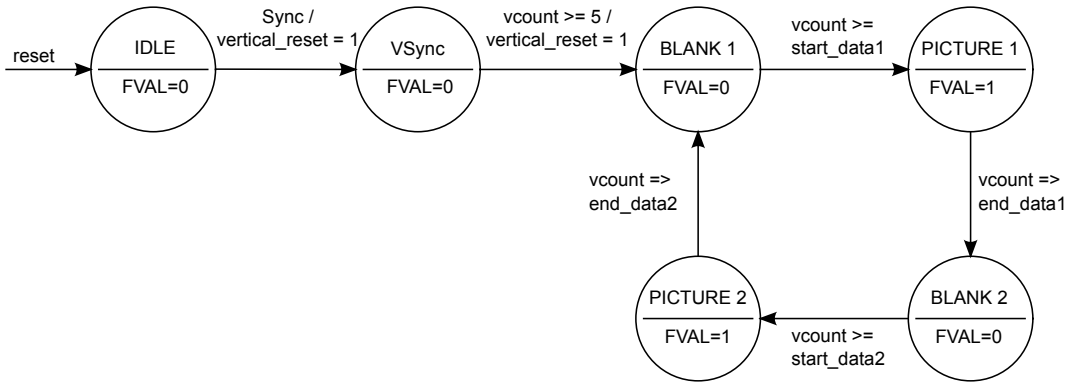


Abbildung 5.24.: Zustandsautomat für das FVAL-Signal (vgl. Abb. 5.23). Damit der Bildzeilenzähler für unterschiedliche Aufgaben eingesetzt werden kann, wird er zwischendurch auf 0 zurückgesetzt (vertical_reset=1). Variablen in der aktuellen Konfiguration: start_data1=21, end_data1=261, start_data2=284, end_data2=524 (vgl. Abb. 5.5, Seite 48 und Zeile 6-8 im Anhang E.1, Seite 85 mit Tab. 5.4).

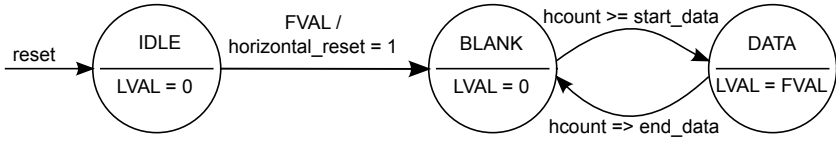


Abbildung 5.25.: Zustandsautomat für das LVAL-Signal (vgl. Abb. 5.23). Zu Begin des ersten Bildes wird der Zähler auf 0 zurückgesetzt (horizontal_reset=1). Konstanten in der aktuellen Konfiguration end_data=640, end_line=857 (vgl. Zeile 6-9 im Angang E.2, Seite 87 mit Tab. 5.4).

Bildes, welches aus der projektiven Transformation kommt (vgl. Abb. 5.18). Von dem Begin des eingehenden Bildes werden 5 Bildzeilen für den zeitlichen Versatz gewartet, bevor das ausgehende Bild gelesen wird (vgl. Abb. 5.24). Danach folgen abwechselnd die Blank- und Bildabschnitte des Bildes (vgl. Abb. 5.4, Seite 47). Da das Vollbild aus einer ungeraden Anzahl von Bildzeilen besteht, sind die Blankabschitte der beiden Halbbilder unterschiedlich lang (vgl. Abb. 5.5, Seite 48).

Der Zustandsautomat für das LVAL-Signal ist die „Horizontal FSM“. Sie synchronisiert sich auf das FVAL-Signal also den Anfang des ausgehenden Bildes (vgl. Abb. 5.23). In den folgenden Zuständen wechseln sich Blank- und Bildabschnitte der Bildzeilen ab.

Das Zeitverhalten wird über Konstanten voreingestellt. Für die Zählerstände der Zustandsüber-

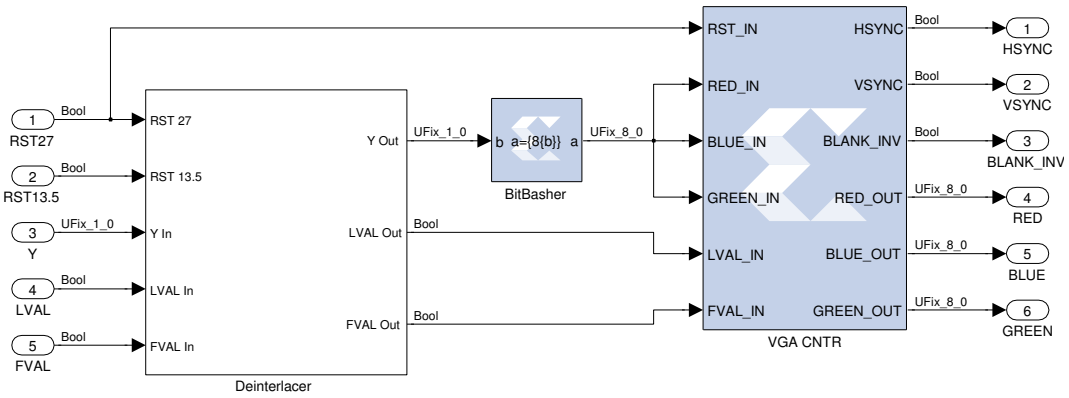


Abbildung 5.26.: Im VGA-Interface werden vom Deinterlacer die Bildzeilen des Halbbildes verdoppelt. So entstehen aus dem 29,5 Hz interlaced Bilddatenstrom ein progressiver 60 Hz Bilddatenstrom. Der BitBaher weitet das 1-Bit-Schwarz-Weiß-Signal auf 8 Bit auf. Aus $0 \Rightarrow 0x00$ und aus $1 \Rightarrow 0xFF$. Der VGA-Controller erzeugt die Synchronisationssignal VSync und HSync für die VGA-Schnittstelle.

gänge werden die Konstanten intern zusammengezählt. Die Konstanten einzeln zu konfigurieren macht die Abstimmung mit dem VGA-Controller einfacher.

5.7. VGA-Interface

Im VGA-Interface wird der Bilddatenstrom für die Darstellung auf dem VGA-Monitor aufbereitet. Der VGA-Standard schreibt eine mindest Bildwiederhohlfrequenz von 60 Hz vor (vgl. Brock). Um von den 29,5 Hz Bildwiederhohlfrequenz der Kamera auf die 60 Hz des Monitors zu kommen, werden im Deinterlacer die Bildzeilen verdoppelt. Die Kamera wird im Interlace-Modus betrieben, liefert also pro Bild zwei Halbbilder (vgl. Kap. 5.2, Seite 45). Ein Halbbild hat die Auflösung von 640 x 240 Bildpunkten. Durch Verdoppeln der Bildzeilen wird es wieder auf seine ursprüngliche Auflösung von 640 x 480 Bildpunkten erweitert, mit dem Unterschied, dass nun pro Kamerabild zwei Bilder zur Verfügung stehen.

Zur Darstellung des 1-Bit-Schwarz-Weiß-Bildes auf einem RGB-Farbmonitor wird das Signal im „BitBasher“ auf 8 Bit vervielfacht und an die RGB-Eingänge des VGA-Controllers geleichmäßig verteilt (vgl. Abb. 5.26). Der „BitBasher“ macht aus dem Eingangssignal 0 das Ausgangssignal 00000000 und aus dem Eingangssignal 1 wird 1111111111. Durch das gleichmäßige Ansteuern aller drei Farben wird auf dem Monitor nur Schwarz oder Weiß dargestellt.

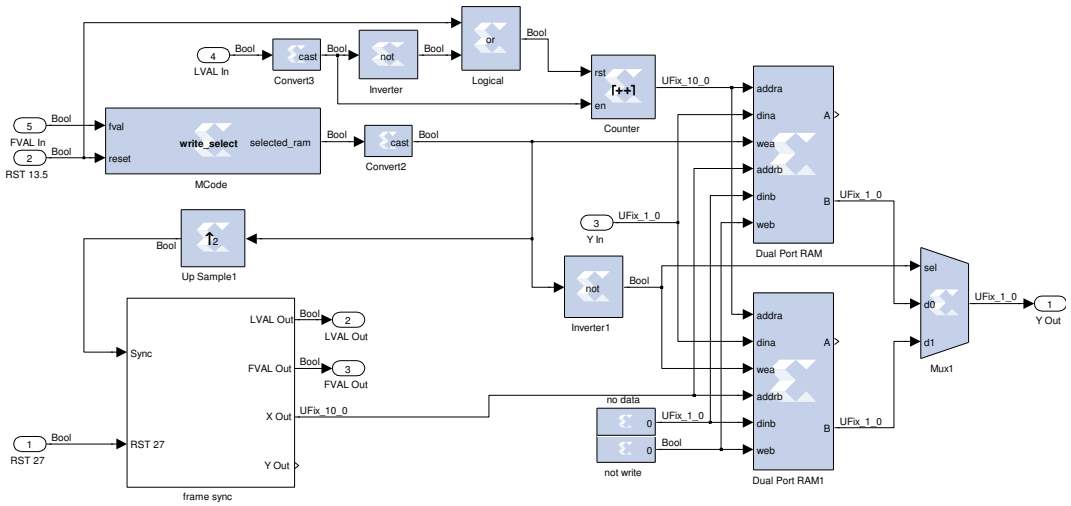


Abbildung 5.27.: Zur Verdoppelung der Bildwiederholfrequenz werden im Deinterlacer die Bildzeilen verdoppelt und so aus zwei Halbbildern zwei Vollbilder.

Der VGA-Controller erzeugt für die VGA-Schnittstelle die Synchronisationssignale VSync und HSync. VSync und HSync unterscheiden sich von FVAL und LVAL durch ein abweichendes Zeitverhalten während der Blankphase.

5.7.1. Deinterlacer

Der Deinterlacer besteht aus zwei „Dual Port RAM“-Blöcken, in welche abwechselnd ankommende Daten geschrieben werden. Der RAM-Block, in welchen gerade nicht geschrieben wird, wird mit der doppelten Geschwindigkeit zwei Mal ausgelesen.

Für den Schreibprozess steuert das eingehende LVAL-Signal den Adresszähler (vgl. Abb. 5.27). Ist LVAL high zählt der Adresszähler von 0 bis 639. Fällt LVAL, am Ende der Bildzeile, auf low, wird der Adresszähler auf 0 zurückgesetzt.

Die Auswahl, welcher der beiden RAM-Blöcke für das Schreiben und Lesen verwendet wird, erfolgt anhand des eingehenden FVAL-Signals. Im „Write Select“-MCode-Block ist ein Flipflop als Moor-Zustandsautomat implementiert der seinen Zustand auf die fallende Flanke von FVAL ändert (vgl. Abb. 5.28).

Für den Leseprozess werden die Synchronisationssignale FVAL und LVAL mit doppelter Taktfrequenz neu erzeugt. Die Erzeugung des Synchronisationssignales funktioniert genauso wie für den Zwischenspeicher der projektiven Transformation (vgl. Kap. 5.6.3, Seite 63). Der einzige Unterschied ist, dass kein zeitlicher Versatz von 5 Bildzeilen beim Starten des Automaten

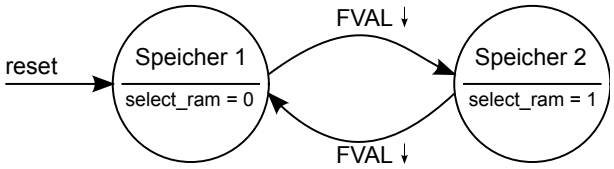


Abbildung 5.28.: Mit der fallenden Flanke von FVAL schalltet der Flipflop zwischen den beiden Speichern um.

vorhanden ist. Die Startsynchronisation des vertikalen Zustandsautomaten findet über den ersten Speicherwechsel, d. h. dem Wechsel von „select_ram“ von low auf high statt. Damit das „select_ram“-Signal aus dem langsameren Schreibprozess im doppelt so schnellen Leseprozess genutzt werden kann, muss es mit einem Upsample-Block an die Taktfrequenz angepasst werden.

Neben dem LVAL-Signal wird aus Zählern auch die X/Y-Koordinate des Bildpunktes gewonnen (vgl. Abb. 5.29). Die Koordinaten errechnen sich aus dem Zählerstand wie folgt:

$$\begin{aligned} x &= horizontal\_counter - h\_pulse - h\_back\_porch - h\_front\_porch \quad (5.2) \\ y &= vertical\_counter - v\_pulse - v\_back\_porch - v\_front\_porch \end{aligned}$$

Die X-Koordinate wird als Adresse zum Lesen des Zeilenspeichers genutzt. Über einen Multiplexer hinter den beiden RAM-Blöcken wird sichergestellt das die Daten des Speichers ausgegeben werden, der gerade gelesen wird.

5.7.2. VGA-Controller

Die VGA-Schnittstelle wurde ursprünglich für Kathodenstrahlröhren-Monitore entwickelt, welche das Bild zeilenweise von oben links nach unten rechts aufbauen. Das Zurücksetzen des Kathodenstrahls am Zeilenende zum Zeilenanfang und am Bildende zum Bildanfang erfordert Zeit. Das Zeitverhalten der Monitore wurde von der „Video Electronics Standards Association“ (VESA) in der „Discrete Monitor Timing“ (DMT) Tabelle standardisiert. Später wurden die Monitore flexibler, sodass Sie nicht mehr auf die starren Tabellen angewiesen sind. Dies spiegelt sich in der „General Timing Formula“ (GTF) wieder. Bei TFT-Bildschirmen ist kein Kathodenstrahl mehr vorhanden, sodass die Zeiten in den „Coordinated Video Timings“ (CVT) wegfallen. Zusätzlich wurden in den „Coordinated Video Timings“ neue Bildschirmauflösungen ergänzt.

Die Synchronisationssignal VSync und HSync sind in jeweils 4 Zeitabschnitte unterteilt (vgl. Abb. 5.30). VSync und HSync sind nur währen der „Pulse“-Phase low (vgl. Abb. 5.31). Den

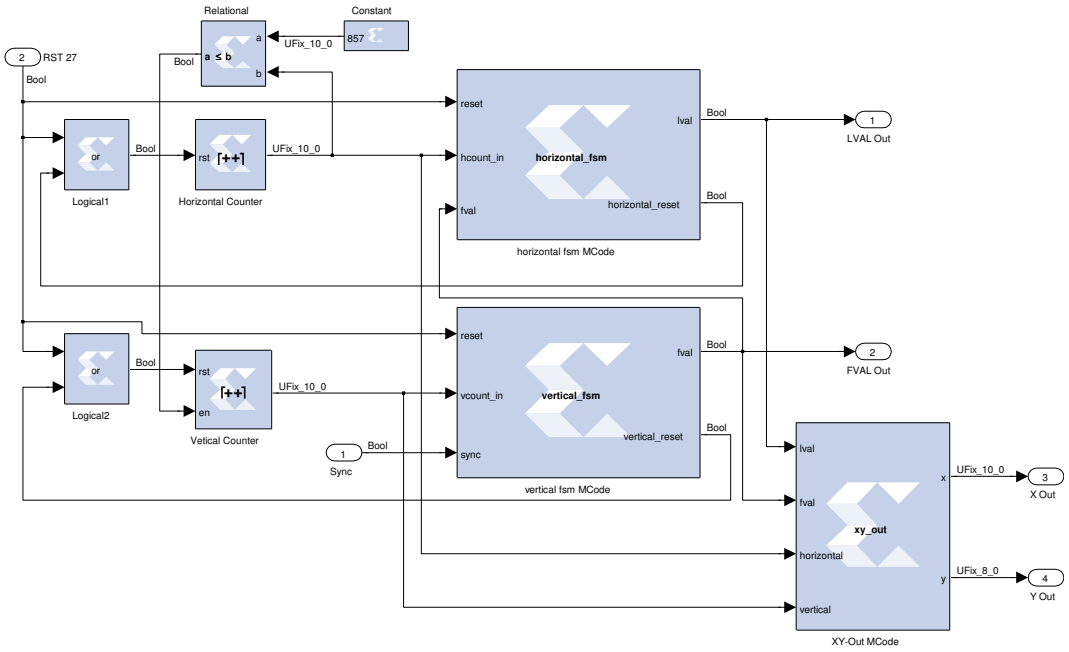


Abbildung 5.29.: Für die Erzeugung der Bildsynchronisationssignale FVAL und LVAL werden wie bei beim Bildkoordinatengenerator zwei Zähler verwendet. Der „Horizontal Counter“ zählt bis 857 und der „Vertical Counter“ bis 524. Die Funktion der Zustandsautomaten „Horizontal FSM“ und „Vertical FSM“ sind in Abb. 5.24 und 5.25 dargestellt.

Abbildung 5.30.: Aufteilung des VGA-Signals in die einzelnen Zeitabschnitte. Nur der graue Bereich enthalt das auf dem Monitor dargestellt Bild. VP = vertical pulse, VBP = vertical back porch, VFP = vertical front porch, VDP = vertical display period, HP = horizontal puls, HBP = horizontal back porch, HFP = horizontal front porch, HDP = horizontal display period

Rest der Zeit sind sie High. Die High-Phase ist damit deutlich länger als die von FVAL und LVAL da sie zusätzlich auch einen Teil des Blankabschnitts beinhaltet. Ein weiterer unterschied zu FVAL und LVAL ist das HSync auch in den vertikalen Blankzeiten die Zeilen synchronisiert.

Ein Eintrag mit einer Taktfrequenz von 27 MHz mit welcher die Bildpunkte von der Kamera kommen ist in der Monitor Timing Tabelle nicht vorhanden (vgl. Tab. 5.3). Der Modus 0 hat die gleiche Bildwiederholfrequenz wie das verdoppelte Kamerabildsignal und wird als Grundlage für die Ansteuerung verwendet.

Modus	f_h	f_v	f_p	**HFP**	**HS**	**HBP**	**VFP**	**VS**	**VBP**
0	31,5 KHz	60 Hz	25,179 MHz	16	96	48	10	2	33
3	37,9 KHz	73 Hz	31,5 MHz	24	40	128	9	3	28
14	35 KHz	67 Hz	30,253 MHz	64	64	96	3	3	39

Tabelle 5.3.: VGA Monitor Timing Tabelle für die Auflösung 640 x 480 Bildpunkte [Brock]

Für die Ansteuerung des Monitors wurde das Zeitverhalten von der Kamera übernommen (vgl. Tab. 5.4). Die Abweihung der Taktfrequenz der Kamera zum Modus 0 wird nur über die Zeilenlänge ausgeglichen. Die Anzahl der Zeilen entspricht dem Vesa-Standard.

Die Erzeugung, der Synchronisationssignal VSync und HSync verläuft, ähnlich wie die von

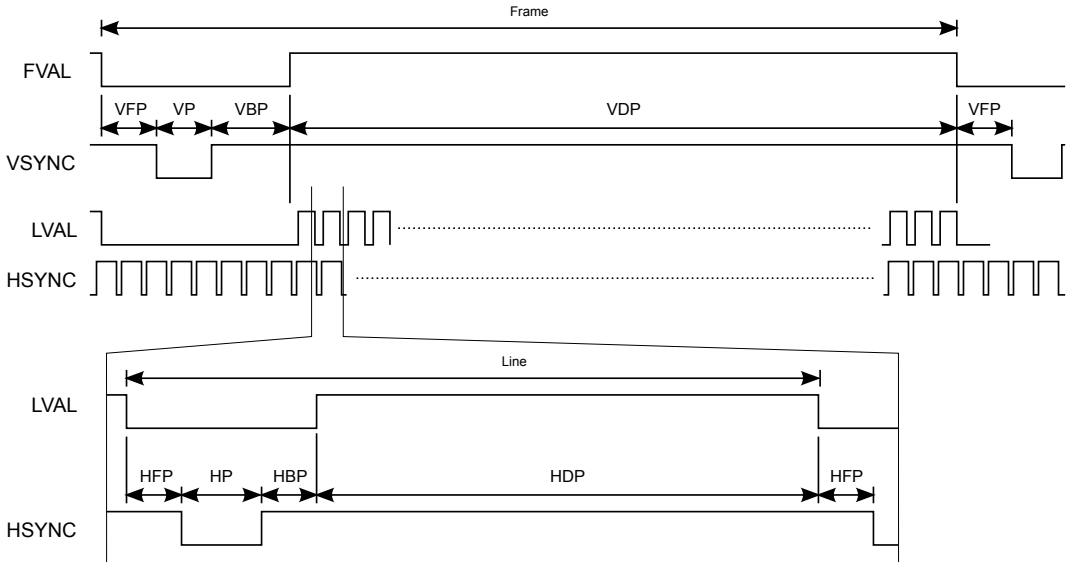


Abbildung 5.31.: Timingdiagramm mit LVAL/FVAL und HSYNC VSYNC

FVAL und LVAL (vgl. Kap. 5.6.3, Seite 63). Zu Synchronisation der beiden Zähler wird auf eine fallende Flanke von FVAL gewartet. Die Synchronisation findet als auf dem Ende des ersten Bildes statt, da die High-Phase von VSync bereits vor der High-Phase von FVAL beginnt (vgl. Abb. 5.31).

Name	Kürzel	Kamera		VGA-Standard		VGA-Controller	
		Takte	Zeit	Takte	Zeit	Takte	Zeit
vertical pulse	VP			2	$63\mu s$	2	$63\mu s$
vertical back porch	VBP	$22,5$	$714\mu s$	33	$1ms$	33	$1ms$
vertical front porch	VFP			10	$317\mu s$	10	$317\mu s$
vertical display periode	VDP	240	$7,6ms$	480	$15ms$	480	$15ms$
vertical summe		$262,5$	$8,3ms$	525	$17ms$	525	$17ms$
horizontal pulse	HP			96	$4\mu s$	137	$5\mu s$
horizontal back porch	HBP	217	$8\mu s$	48	$2\mu s$	40	$1,5\mu s$
horizontal front porch	HFP			16	$635ns$	40	$1,5\mu s$
horizontal display periode	HDP	640	$24\mu s$	640	$25\mu s$	640	$24\mu s$
horizontal summe		857	$32\mu s$	800	$32\mu s$	857	$32\mu s$

Tabelle 5.4.: Konfigurationsparameter für den VGA-Controller zur Anpassung des Zeitverhaltens der CCD-Kamera Sony FCB-PV10 an den VGA-Standard.

6. Zusammenfassung

Durch die Integration, der projektive Bildtranformation wurde, die Fahrspurerkennung des Faust SoC-Fahrzeugs weiter vervollständigt. Zur Rekonstruktion des Bilddatenstroms hinter der projektiven Transformation wurde ein Zwischenspeicher entwickelt, welcher nur ein Drittel eines Bildes zwischenspeichert und somit im FPGA Logikelemente und BRAM-Zellen für andere Module freilässt. Auch die Latenz des Zwischenspeichers konnte auf 5 Bildzeilen = $237\mu s$ gesenkt werden.

Die Größe des Zwischenspeichers ist abhängig von dem maximalen Abstand zwischen Quell- und Zielkoordinate der Bildpunkte in der projektiven Transformation. Bei einer Änderung der Transformationsmatrix ist der Abstand neu zu messen und der Zwischenspeicher entsprechend anzupassen.

Die gesamte Bildverarbeitungspipeline wurde in ein modellbasiertes Konzept überführt und in den „System Generator“ von Xilinx integriert. Die Integration in den „System Generator“ erhöht die Übersicht über das Gesamtsystem und die Funktion der einzelnen Funktionsblöcke. Auch lassen sich die Verarbeitungswege leichter anpassen. So sind für die Umstellung von vom Schwarz-Weiß-Bild zu einem Graustufenbild nur an zwei stellen Änderungen durchzuführen. Neben der Entfernung des Binarisierungsfunktionsblockes (vgl. Kap. 5.3, Seite 53) ist zwischen dem Deinterlacer und dem VGA-Controller die Erweiterung des 1-Bit-Signals auf 8 Bit zu entfernen. Alle Funktionsblöcke die dazwischenliegen passen sich an den breiteren Bilddatenstrom an.

Ein weiterer Vorteil des „System Generators“ ist die Integration in die Matlab/Simulink Simulationsumgebung. Hierdurch lassen sich einzelne Funktionsblöcke des Simulink-Models in eine FPGA-Implementierung auslagern und zusammen Simulieren. Für die Berechnung der Simulation verwendet der „System Generator“ nur einen Rechenkern des Prozessors, wie an der Systemauslastung zu Beobachten war. Die Simulation von projektiver Transformation und Zwischenspeicher dauert ungefähr 1,5 Stunden.

Mit dem WaveScope-Funktionblock lässt sich, ähnlich wie in ModelSim, der Verlauf von Signalen analysieren, welche vor der Simulation ausgewählt wurden. Das Vergrößern und Verschieben des Signalverlaufs im Scope-Fenster dauert öfter mehre Minuten. Das Generieren einer TestBench für eine ModelSim-Simulation dauert genauso lange wie die Simulation des Modells, da die Daten für die Testbench in der Simulation gewonnen werden. Bei jeder Änderung

im Modell wird die Testbench neu generiert, da dem „System Generator“ nicht bekannt ist, ob sich die Eingangsdaten geändert haben.

Im Vergleich von reinen VHDL-Implementierung und vom „System Generator generierten VHDL-Code lag der Verbrauch der Hardwaresourcen des FPGAs bei der „System Generator“-Implementierung meist über der reinen VHDL-Implementierung. Im Verhältnis zu den im FPGA zur verfügungstehenden Logicelemente ist liegt der unterschied unter 0,5

Ein Bruch bei der modellbasierten Entwicklung liegt bei der Implementierung von Zustandsautomaten vor, welche im „System Generator“ als MCode zu implementieren sind.

Die nächsten Entwicklungsschritte für die Bildverarbeitungspipeline sind:

- Montage der Kamera mit einem festen Neigungswinkel auf einem Modellfahrzeug.
- Ausmessen der Transformationsmatrix und konvertieren in eine Festkommazahl mit minimaler Bitbreite.
- Analyse des maximalen Abstandes zwischen Quell- und Zielkoordinate der projektiven Transformation und anpassen des Zwischenspeichers.
- Entwickeln einer dynamischen Schwellwertberechnung für die Konvertierung des Graustufenbildes in ein Schwarz-Weiß-Bild zum Erreichen eines maximalen Kontrastes bei wechselnden Lichtverhältnissen.
- Umstellung des VGA-Controllers auf eine frei konfigurierbare Version (vgl. Jeyrani-Mameghani, 2009)
- Einbinden der Hough-Transformation in die Bildverarbeitungspipeline (vgl. Abb. 1.2, Seite 9)

A. Quellcode zur Analyse der Sprungweite

```
RGB = imread('../projective_transform/pt_schachbrett.bmp');

% Konvertieren in Graustufenbild
% Grauwert = 0.299*R+0.587*G+0.114*B

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

% Transformationsmatrix Vorwaertstransformation
BV = [ 0.9579    0.8932   35.6182;
     -0.0173    2.4353   22.4984;
     -0.0001    0.0026    1.0000
    ];

% Transformationsmatrix Rueckwaertstransformation
BR = [ -1.0121 0.3409 28.3811;
     -0.0064 -0.4094 9.44;
     -0.00008 0.001098 -1];

[size_y size_x] = size(I);

% Initialisierung mit Schwarz
DST = ones(size_y, size_x);

% Binarisierung mit Schwellwert 65
for y=1:size_y
    for x=1:size_x
        if(I(y,x)<65)
            I(y,x) = 0;
        else
            I(y,x) = 255;
        end;
```

```
34      end;
35  end;
36
37  % Anzeigen des Quellbildes
38  figure(1);
39  imshow(I);
40
41  % ------------------------------------------------------------
42  % Vorwaertstransformation
43  % ------------------------------------------------------------
44
45  DIVYF = zeros(1,size_x*size_y);
46  HISTYF = zeros(1,170);
47
48  k = 1;
49  for y=1:size_y
50    for x=1:size_x
51
52      % Berechnung X/Y Zielbild
53      X = (BV(1,1)*x+BV(1,2)*y+BV(1,3))/(BV(3,1)*x+BV(3,2)*y+BV
            (3,3));
54      Y = (BV(2,1)*y+BV(2,2)*y+BV(2,3))/(BV(3,1)*x+BV(3,2)*y+BV
            (3,3));
55      % Runden auf ganze Zahlen
56      X = round(X);
57      Y = round(Y);
58
59      % Schreiben des Quellbildpunktes in Zielbild
60      DST(Y,X) = I(y,x);
61
62      % Verkleinerung des Speichers f r Anhang B
63      if (Y - y) > 142
64        DST2(Y,X) = I(y,x);
65      else
66        DST2(Y,X) = 50;
67      end;
68
69      DIVYF(k) = Y - y;
70      HISTYF(abs(Y - y)+1)=HISTYF(abs(Y - y)+1)+1;
71      k = k + 1;
```

```
72   end;
73 end;
74
75 % Anzeigen Zielbild
76 figure(2);
77 colormap(gray(255));
78 image(DST2);
79
80 % Anzeigen der Sprungweite
81 figure(3);
82 plot(DIVYF);
83
84 % Anzeigen des Sprungweitenhistogramms
85 figure(4);
86 plot(HISTYF);
87
88 % ------------------------------------------------------------
89 % Rueckwaertstransformation
90 % ------------------------------------------------------------
91
92 DIVYR = zeros(1,size_x*size_y);
93 HISTYR = zeros(1,150);
94
95 k = 1;
96
97 for y=1:size_y
98   for x=1:size_x
99     % Berechnung X/Y Zielbild
100    X = (BR(1,1)*x+BR(1,2)*y+BR(1,3))/(BR(3,1)*x+BR(3,2)*y+BR
         (3,3));
101    Y = (BR(2,1)*y+BR(2,2)*y+BR(2,3))/(BR(3,1)*x+BR(3,2)*y+BR
         (3,3));
102    % Runden auf ganze Zahlen
103    X = round(X);
104    Y = round(Y);
105
106    % Quellpunkte ausserhalb des Bildes mit 0 ersetzen
107    if(X > 0 && Y > 0 && X <= size_x && Y <= size_y)
108      DST2R(y,x) = I(Y,X);
109    else
```

```
110       DST2R(y,x) = 0;
111     end;
112
113     DIVYR(k) = Y - y;
114     HISTYR(abs(Y - y))=HISTYR(abs(Y - y))+1;
115
116     k = k + 1;
117   end;
118 end;
119
120 % Anzeigen Zielbild
121 figure(10);
122 colormap(gray(255));
123 image(DST2R);
124
125 % Anzeigen der Sprungweite
126 figure(11);
127 plot(DIVYR);
128
129 % Anzeigen des Sprungweitenhistogramms
130 figure(12);
131 plot(HISTYR);
```

B. Auswirkungen der Matrix auf den Abstand zwischen Quell- und Zielkoordinate

Die Transformationsmatrix legt den Bildausschnitt des Zielbildes fest. Bei der hier vorgestellten Matrix B.1 für die Rückwärtstranformation fallen die Ränder mit undefinierten Bildpunkten gegen über der Matrix aus Kap. 2.2, Seite 19 deutlich geringer aus (vgl. schwarze Dreiecke in Abb. 2.3(c), Seite 17 mit Abb. B.1).

$$HR_2 = \begin{bmatrix} 0{,}63543 & -0{,}32908 & 107 \\ 0{,}00483 & 0{,}43387 & 4 \\ -0{,}00002 & -0{,}00111 & 1 \end{bmatrix} \tag{B.1}$$

Die Transformationsmatrix hat gegenüber der Matrix Gl. 2.5, Seite 20 den Nachteil das die X- und Y-Koordinaten nicht den gleichen Maßstab haben. Dies ist an der Steckung der Quadrate in Abb. B.1 zu erkennen. Somit erfüllt sie nicht die Gewünschten Kriterien (vgl. Kap. 2.1, Seite 17). Die Matrix wurde mit dem Bildbearbeitungsprogramm Gimp[1] generiert.

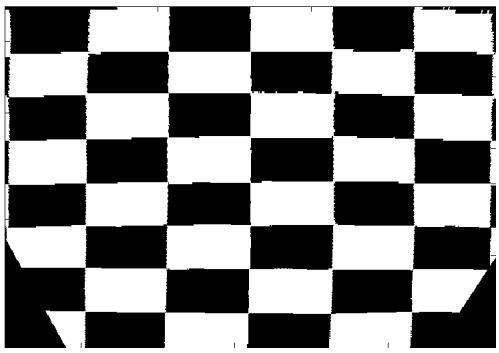


Abbildung B.1.: Bildausschnitt des perspektivisch transformierten Kamerabildes mit einer alternativen Transformationsmatrix Gl. B.1 (vgl. Abb. 2.3 auf Seite 17)

Die Matrix B.1 hat mit 102 Bildzeilen eine geringere Sprungweite als die Matrix Gl. 2.5, Seite 20 (vgl. Abb. 4.4(b), Seite 35 mit Abb. B.2).

[1] http://www.gimp.org

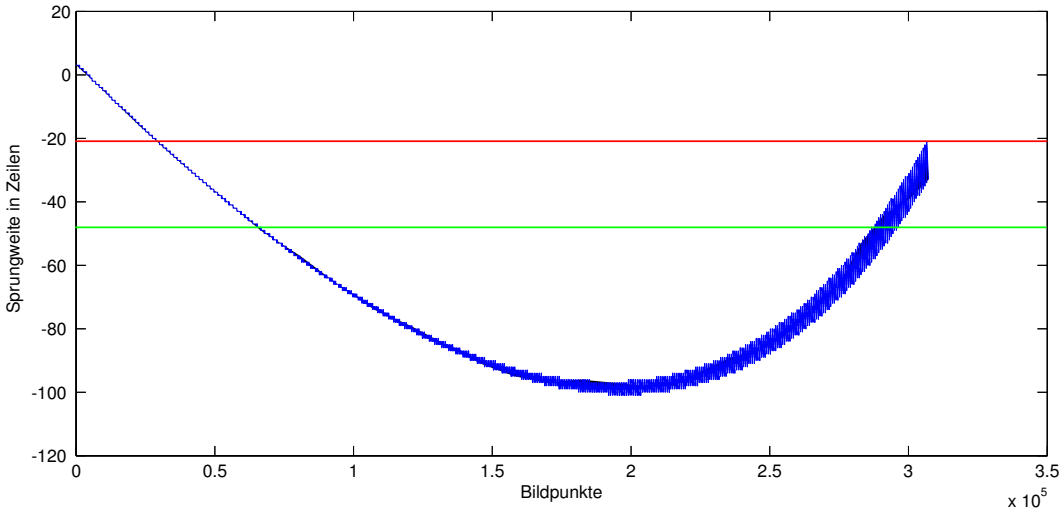


Abbildung B.2.: Verteilung der Sprungweite in Zeilen über die Bildpunkte mit einer alternativen Transformationsmatrix Gl. B.1

C. Vergleich von VHDL und System Generator

In diesem Kapitel wird der vom „System Generator erzeugte VHDL-Code mit einer direkten VHDL-Implementierung vergleichen. Ziel ist es herauszufinden welche Auswirkungen der Einsatz des „System Generator“ auf den Verbrauch von FPGA-Hardwareresourcen hat. Als Vergleichsobjekt wird der Zwischenspeicher zur Regenerierung des Bilddatenstroms verwendet. In den Zwischenspeicher werden die, aus der projektiven Bildtransformation kommenden, Bildpunkte geschrieben und zeilenorientierte Bilddatenstrom wieder ausgegeben (vgl. Kap. 5.6, Seite 59). Der Zwischenspeicher besteht aus der Berechnung der Speicheradresse für das Schreiben mit einem Multiplizierer, einem Dual-Port-RAM mit 65536 Bit Speicher sowie einem Zustandsautomaten zur Regenerierung der Synchronisationssignale. Der Zwischenspeicher wurde zuerst mit VHDL implementiert und im Anschluss die gleiche Funktionalität mit dem „System Generator“ umgesetzt.

Der Ressourcenverbrauch ist bei dieser Implementierung in der „System Generator“-Variante höher (vgl. Tab. C.1). Lediglich die Anzahl der Flip Flops ist beim „System Generator“ niedriger.

Logic	VHDL	System Generator
Number of Slice Flip Flops	46	41
Number of 4 input LUTs	164	179
Number of occupied Slices	104	119
Total Number of 4 input LUTs	198	227
Number of bonded IOBs	46	46
Number of RAMB16s	4	4
Number of BUFGMUXs	1	1
Number of MULT18X18SIOs	1	1
Avarage Fanout of Non-Clock Nets	2,82	2,76

Tabelle C.1.: Gegenüberstellung der VHDL und der System Generator Implementierung des Zwischenspeichers. Die Übersetzung des Codes fand mit ISE 12.2 statt.

Um herauszufinden, wo der zusätzliche Ressourcenbedarf entsteht, wurde die VHDL-Implementierung der Bildverarbeitungspipeline aus Kirschke (2009) / Peters (2009) als Blackbox-Module in ein „System Generator“ Projekt eingefügt (vgl. Abb. C.1). In diesem Projekt

werden nur die Verbindungen zwischen den einzelnen VHDL-Modulen im „System Generator“ erzeugt. Des Weiteren ist der „System Generator“ für die Taktfrequenzerzeugung zuständig. Die gleichen VHDL-Module wurden in VHDL zu einer Pipeline verbunden.

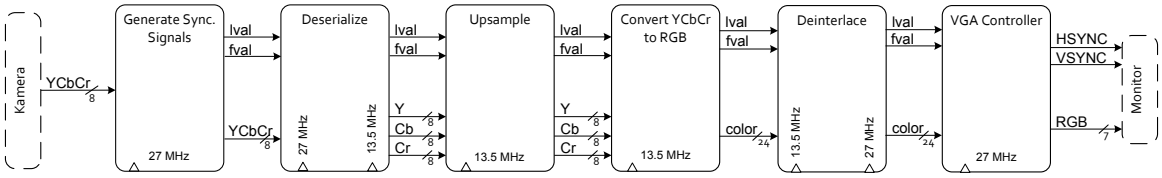


Abbildung C.1.: Die Bildverarbeitungspipeline bereitet die Bilddaten der Kamera für die Darstellung auf einem VGA-Monitor auf. [Kirschke (2009), Peters (2009)]

Der Unterschied des Hardwareverbrauchs der beiden kompilierten Implementierungen ist im Vergleich zu der gestiegenen Komplexität des Systems gering (vgl. Tab. C.2). Bei den Implementierungen wird der Takt auf unterschiedliche Art erzeugt. Die VHDL-Implementierung verwendet eine vom Xilinx „Core Generator“ erzeugte DCM. Die „System Generator“-Implementierung arbeitet mit „Clock Enable“. Um diesen Unterschied auszugleichen, wurde die „System Generator“-Implementierung mit der gleichen DCM als externes Modul kompiliert. Mit der externen DCM ist der Unterschied noch geringer. Nur die „occupied Slices“ sind weiter angestiegen. Eine Ursache hierfür wurde nicht ermittelt.

Logic	VHDL	System Generator	System Generator DCM
Number of Slice Flip Flops	960	988	962
Number of 4 input LUTs	1.173	1.193	1.179
Number of occupied Slices	785	930	958
Total Number of 4 input LUTs	1.401	1.452	1.407
Number of bonded IOBs	21	21	21
Number of RAMB16s	0	0	0
Number of BUFGMUXs	2	1	2
Number of DCMs	1	0	1
Number of MULT18X18SIOs	5	5	5
Avarage Fanout of Non-Clock Nets	2,70	2,67	2,70

Tabelle C.2.: Gegenüberstellung der VHDL und der System Generator Implementierung der Videopipeline. Die Übersetzung des Codes fand mit ISE 12.3 statt.

Aus diesem Vergleich sind folgende Schlüsse zu ziehen:

- Bei der Verknüpfung von Funktionsblöcken im „System Generator“ ist der Ressourcenbedarf nur geringfügig (unter 0,5%) höher, als bei einer VHDL-Verknüpfung.
- Die vorgefertigten Funktionsblöcke des „System Generators“ sind allgemeiner gehalten und können nicht so gut wie in VHDL auf ihre Aufgabe optimiert werden. Dadurch entsteht ein höherer Ressourcenbedarf.
- Bei der Takterzeugung ist es situationsabhängig, ob „Clock Enable“ oder DCM die bessere Wahl ist.

D. Darstellung der wegfallenden Bildinformationen bei der Verkleinerung des Speichers unter die max. Sprungweite

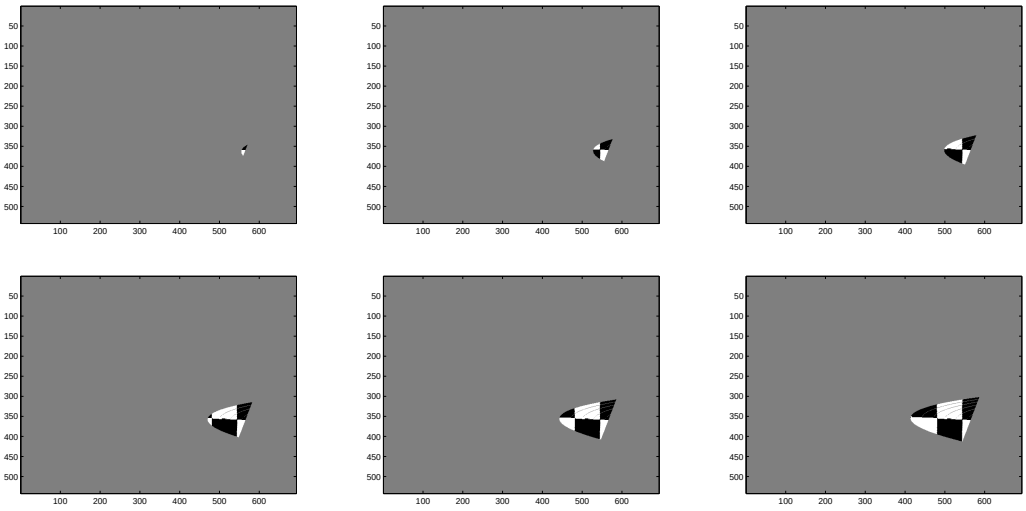


Abbildung D.1.: Bei der Verwendung eines Zwischenspeichers welcher kleiner ist als die max. Sprungweite gehen Bildinformationen verloren. In dieser Abbildung wurde gewählt, welcher 1-6 Zeilen kleiner ist als die max. Sprungweite. Schon bei kleinen Reduzierungen gehen Informationen in der Bildmitte verloren.

E. Zustandsautomaten als MCode

E.1. Vertical FSM

```
function [ fval,  vertical_reset] = vertical_fsm(reset,
    vcount_in, sync, v_blank_period1, v_back_porch,
    v_display_period, v_blank_period2)

  persistent state,
  state = xl_state(0, {xlUnsigned, 2, 0});

  start_data1 = v_blank_period1;
  end_data1 = start_data1 + v_display_period;
  start_data2 = end_data1 + v_blank_period2;
  end_data2 = start_data2 + v_display_period;


  if reset
    state=0;
  end

  switch state
    case 0 % IDLE
      fval = false;
      if sync
        state = 1; % TOPBLANK
        vertical_reset = true;
      else
        state = 0; % IDLE
        vertical_reset = false;
      end
    case 1 % DELAY
      fval = false;
      if vcount_in >= 5
```

```
29          state = 2;
30          vertical_reset = true;
31        else
32          state = 1;
33          vertical_reset = false;
34        end
35      case 2 % BLANK
36        fval = false;
37        vertical_reset = false;
38        if vcount_in >= start_data1
39          state = 3; % PICTURE
40        else
41          state = 2; % BLANK
42        end
43
44      case 3 % PICTURE
45        fval = true;
46        vertical_reset = false;
47        if vcount_in >= end_data1
48          state = 4; % BLANK
49        else
50          state = 3; % PICTURE
51        end
52      case 4 % BLANK
53        fval = false;
54        vertical_reset = false;
55        if vcount_in >= start_data2
56          state = 5; % PICTURE
57        else
58          state = 4; % BLANK
59        end
60
61      case 5 % PICTURE
62        fval = true;
63        vertical_reset = false;
64        if vcount_in >= end_data2
65          state = 2; % BLANK
66        else
67          state = 5; % PICTURE
68        end
```

```
69
70      otherwise
71        fval = false;
72        vertical_reset = false;
73        state = 0; % IDLE
74    end;
75  end
```

E.2. Horizontal FSM

```
1  function [ lval,  horizontal_reset] = horizontal_fsm(reset,
       hcount_in, fval, h_pulse_width, h_back_porch,
       h_display_period, h_front_porch)
2
3    persistent state,
4    state = xl_state(0, {xlUnsigned, 2, 0});
5
6    start_line = 0;
7    start_data = start_line + h_front_porch + h_pulse_width +
        h_back_porch;
8    end_data = start_data + h_display_period;
9    end_line = end_data;
10
11   if reset
12     state=0;
13   end
14
15   switch state
16     case 0 % IDLE
17       lval = false;
18       if fval
19         state = 1; % DATA
20         horizontal_reset = true;
21       else
22         state = 0; % IDLE
23         horizontal_reset = false;
24       end
25     case 1 % BLANK
26       lval = false;
27       horizontal_reset = false;
```

```
28        if hcount_in >= start_data
29          state = 2; % DATA
30        else
31          state = 1; % BLANK
32        end
33
34      case 2 % DATA
35        lval = fval;
36        horizontal_reset = false;
37        if hcount_in >= end_line || hcount_in == start_line
38          state = 1; % BLANK
39        else
40          state = 2; % DATA
41        end
42
43      otherwise
44        lval = false;
45        horizontal_reset = false;
46        state = 0; % IDLE
47    end;
48  end
```

E.3. Flipflop zur Auswahl des Speichers des Deinterlacers

```
1  function selected_ram = write_select(fval, reset)
2
3    persistent fval_last,
4    fval_last = xl_state(0, {xlBoolean});
5    persistent state,
6    state = xl_state(0, {xlUnsigned, 1, 0});
7
8    if reset
9      state=0;
10   end
11
12   switch double(state)
13     case 0 % IDLE
14       selected_ram = false;
15       if fval_last && fval
16         state = 1;
```

```
17        else
18          state = 0;
19        end
20      case 1
21        selected_ram = true;
22        if fval_last && fval
23          state = 0;
24        else
25          state = 1;
26        end
27
28      otherwise
29        selected_ram = false;
30        state = 0;
31    end;
32
33    fval_last = fval;
34  end
```

F. Hardwareverbrauch der VHDL-Module

Logic	Used
Number of Slice Flip Flops	123
Number of 4 input LUTs	86
Number of occupied Slices	98
Total Number of 4 input LUTs	183
Number of RAMB16s	0
Number of BUFGMUXs	1
Number of MULT18X18SIOs	0
Avarage Fanout of Non-Clock Nets	2,99

Tabelle F.1.: Hardwareverbrauch des Synchronisationsgenerator-Moduls

Logic	Used
Number of Slice Flip Flops	72
Number of 4 input LUTs	26
Number of occupied Slices	41
Total Number of 4 input LUTs	26
Number of RAMB16s	0
Number of BUFGMUXs	1
Number of MULT18X18SIOs	0
Avarage Fanout of Non-Clock Nets	3,13

Tabelle F.2.: Hardwareverbrauch des Demultiplexer-Moduls

Logic	Used
Number of Slice Flip Flops	23
Number of 4 input LUTs	85
Number of occupied Slices	53
Total Number of 4 input LUTs	103
Number of RAMB16s	0
Number of BUFGMUXs	1
Number of MULT18X18SIOs	0
Avarage Fanout of Non-Clock Nets	2,78

Tabelle F.3.: Hardwareverbrauch des VGA-Controllers

G. Pinbelegung Nexus2 für die Bildverarbeitungspipeline

G.1. VGA-Schnittstelle

Nexus 2		FPGA		
Pin Nr.	**Funktion**	**Pin Nr**	**Name**	**Funktion**
		R9	red(0)	VGA Rot 0
1	red	T8	red(1)	VGA Rot 1
		R8	red(2)	VGA Rot 2
		N8	green(0)	VGA Grün 0
2	grn	P8	green(1)	VGA Grün 1
		P6	green(2)	VGA Grün 2
3	blu	U5	blue(0)	VGA Blau 0
		U4	blue(1)	VGA Blau 1
13	HS	T4	hsync	VGA Horizontal Sync
14	VS	U3	vsync	VGA Vertikal Sync

G.2. CCD-Kamera Sony FCB-PV10

Der Pin J13, an dem das Taktsignal der Kamera angeschlossen ist, ist nicht an das Clock-Netzwerk des FPGAs angeschlossen. Aus diesem Grund ist eine Ausnahme in die UCF-Datei aufzunehmen.

```
1 NET "clk" CLOCK_DEDICATED_ROUTE = FALSE;
```

	CCD-Kamera		**Nexus 2**		**FPGA**	
Pin Nr.	**Name**	**Funktion**	**Pin Nr.**	**Pin Nr.**	**Name**	**Funktion**
1	GND	Masse	-	-	-	-
2	Y0	Digital Out 0	JC1	G15	ycbcr(0)	Video In 0
3	Y1	Digital Out 1	JC7	H15	ycbcr(1)	Video In 1
4	Y2	Digital Out 2	JC2	J16	ycbcr(2)	Video In 2
5	Y3	Digital Out 3	JC8	F14	ycbcr(3)	Video In 3
6	Y4	Digital Out 4	JC3	G13	ycbcr(4)	Video In 4
7	Y5	Digital Out 5	JC9	G16	ycbcr(5)	Video In 5
8	Y6	Digital Out 6	JC4	H16	ycbcr(6)	Video In 6
9	Y7	Digital Out 7	JC10	J12	ycbcr(7)	Video In 7
10	GND	Masse	JC5	-	-	-
11	C0	-	-	-	-	-
12	C1	-	-	-	-	-
13	C2	-	-	-	-	-
14	C3	-	-	-	-	-
15	C4	-	-	-	-	-
16	C5	-	-	-	-	-
17	C6	-	-	-	-	-
18	C7	-	-	-	-	-
19	GND	Masse	-	-	-	-
20	VSYNC	Vertikal Sync	-	-	-	-
21	HSYNC	Horizontal Sync	-	-	-	-
22	GND	Masse	-	-	-	-
23	CLOCK	Takt	JD1	J13	clk	-
24	GND	Masse	-	-	-	-

Glossar

injektiv Eine Zielmenge ist linkseindeutig, wenn ihr Funktionswert nur höchstens einmal angenommen wird. Es gibt also für jeden Funktionswert nur eine Eingabe um ihn zu erreichen.

Interlace Das Zeilensprungverfahren wurde zur Verringerung des Bildflimmerns in der Fernsehtechnik entwickelt. Bei diesem Verfahren besteht das Bild aus zwei Halbbildern. Das erste Halbbild enthält die geraden und das zweite Halbbild die ungeraden Bildzeilen. Durch das abwechselnde Neuzeichnen der Halbbilder auf Röhrenfernsehern wird das Bild als weniger flimmernd empfunden.

Mealy-Automat Beim Mealy-Automat hängt die Ausgabe vom Zustand und vom Eingangssignal ab. Jedem Zustandsübergang ist nicht nur ein Eingangsignal zugeordnet sonder auch ein Ausgangssignal.

Moore-Automat Das Ausgangssignal eines Moor-Automaten hängt nur von seinem Zustand nicht aber vom Eingangssignal ab.

Progressive Das Vollbildverfahren bezeichnet den Bildaufbau von Monitoren, wo anders als beim Zeilensprungverfahren keine Halbbilder gesendet werden, sondern Vollbilder.

Q-Format Das Q-Format ist eine Darstellung von Festkommazahlen, welche sich aus Quotienten und Bruchanteil zusammensetzt. Z. B. bedeutet Q10.3, dass der gesamte Vektor 10 Bit breit ist. Davon werden 3 Bits für den Bruchanteil verwendet, es bleiben 7 Bits für den Integeranteil mit Vorzeichen.

surjektiv Eine Zielmenge ist rechtstotal, wenn jedes Element mindestens einmal als Funktionswert vorkommt.

Zustandsautomaten ‘Mit Zustandsautomaten werden zyklische Funktionsabläufe realisiert, sie steuern andere Logikschaltungen und in komplexen digitalen Systemen werden sie zur Syncronisation mehrerer Komponenten eingesetzt. Zustandsautomaten sind sequentiell arbeitende Logikschaltungen, die gesteuert durch ein periodisches Taktsignal eine Abfolge von Zuständen zyklisch durchlaufen.’ Reicherdt und Schwarz (2009)

Literaturverzeichnis

[Bagni u. a. 2008] BAGNI, Daniele ; MARZOTTO, Roberto ; ZORATTI, Paul: Building Automotive Driver Assistance System Algorithms with Xilinx FPGA Platforms. In: *Xcell Journal* (2008), Nr. 66, S. 20–26

[Brock] BROCK, Baldur: *VGA Standard - Timing - Modes*. – URL http://info.electronicwerkstatt.de/bereiche/monitortechnik/vga/Standard-Timing/index.html. – Zugriff 2010-11-15

[Burckhardt 1993] BURCKHARDT, Manfred ; REIMPELL, Jörnsen (Hrsg.): *Fahrwerktechnik: Radschlupf-Regelsysteme*. 1. Vogel, 1993. – ISBN 3-8023-0477-2

[Digilent 2008] Digilent: *Digilent Nexys2 Board Reference Manual*. Juni 2008. – URL http://www.digilentinc.com/Data/Products/NEXYS2/Nexys2_rm.pdf

[Fischer 2010] FISCHER, Walter: *Digitale Fernseh- und Hörfunktechnik in Theorie und Praxis*. 3. Springer, September 2010. – ISBN 978-3-642-15047-0

[Harley und Zisserman 2003] HARLEY, Richard ; ZISSERMAN, Andrew: *Multiple View Geometry in computer vision*. 2. Cambrige University Press, 2003. – ISBN 978-0-521-54051-3

[Horsinka 2010] HORSINKA, Sven A.: *Integration einer Hough-Transformation zur Fahrspurerkennung in eine SoC-basierte Videoverarbeitungspipeline*, Hochschule für Angewandte Wissenschaften Hamburg, Bachelorarbeit, November 2010

[Isermann 2006] ISERMANN, Rolf: Das mechatronische Kraftfahrzeug. In: ISERMANN, Rolf (Hrsg.): *Fahrdynamik-Regelung*. Vieweg, 2006, S. 1–26. – ISBN 978-3-8348-9049-8

[ITU 1994] ITU: *RECOMMENDATION ITU-R BT.601-4*. 1994

[ITU 2007] ITU: *RECOMMENDATION ITU-R BT.656-5*. 2007

[Jack 2005] JACK, Keith: *Video Demystified*. 4. Elsevier, 2005. – ISBN 0-7506-7822-4

[Jenning 2008] JENNING, Eike: *Systemidentifikation eines autonomen Fahrzeuges mit einer robusten, kammerabasierten Fahrspurerkennung in Echtzeit*, Hochschule für Angewandte Wissenschaften Hamburg, Masterarbeit, Dezember 2008. – URL http://opus.haw-hamburg.de/volltexte/2009/742/

[Jeyrani-Mameghani 2009] JEYRANI-MAMEGHANI, Ramin: *SoC-basierte Erprobungsplattform für CCD-Kameras mit Channel Link Interface*, Hochschule für Angewandte Wissenschaften Hamburg, Diplomarbeit, September 2009. – URL http://opus.haw-hamburg.de/volltexte/2009/860/

[Kirschke 2009] KIRSCHKE, Marco: *Echtzeitbildverarbeitung mit einer FPGA-basierten Prozessorelementkette in einem Fahrsurerkennungssystem*, Hochschule für Angewandte Wissenschaften Hamburg, Bachelorarbeit, Mai 2009. – URL http://opus.haw-hamburg.de/volltexte/2009/827/

[Meisel 2008] MEISEL, Andereas: *Robot Vision*. Vorlesungfolien. März 2008

[Mellert 2010] MELLERT, Dennis: *Modellierung einer Video-basierten Fahrspurerkennung mit dem Xilinx System Generator für einen SoC-Plattform*, Hochschule für Angewandte Wissenschaften Hamburg, Bachelorarbeit, April 2010. – URL http://opus.haw-hamburg.de/volltexte/2010/1054/

[Peters 2009] PETERS, Falko: *FPGA-basierte Bildverarbeitungspipeline zur Fahrspurerkennung eines autonomen Fahrzeuges*, Hochschule für Angewandte Wissenschaften Hamburg, Bachelorarbeit, April 2009. – URL http://opus.haw-hamburg.de/volltexte/2009/833/

[Reicherdt und Schwarz 2009] REICHERDT, Jürgen ; SCHWARZ, Bernd: *VHDL-Synthese*. Oldenbourg Verlag, 2009

[Santarini 2008] SANTARINI, Mike: Driver Assistance Revs Up On Xilinx FPGA Platforms. In: *Xcell Journal* (2008), Nr. 66, S. 8–15

[Sony] Sony: *Technical Manual FCB-PV10*

[Xilinx 2009a] Xilinx: *Divider Generator 3.0*. Juni 2009

[Xilinx 2009b] Xilinx: *Spartan-3E FPGA Family Data Sheet*. Version 3.8. August 2009

[Xilinx 2010] Xilinx: *System Generator for DSP User Guide*. 12.3. September 2010

Versicherung über Selbstständigkeit

Hiermit versichere ich, dass ich die vorliegende Arbeit im Sinne der Prüfungsordnung nach §16(5) ohne fremde Hilfe selbstständig verfasst und nur die angegebenen Hilfsmittel benutzt habe.

Hamburg, 16. Januar 2011	
Ort, Datum	Unterschrift