Bachelorarbeit

Paulin Pekezou Fouopi

Extrahierung von stereoskopischer
Tiefeninformation modelliert für eine SoC-Plattform

Paulin Pekezou Fouopi

Extrahierung von stereoskopischer Tiefeninformation modelliert für eine SoC-Plattform

Bachelorarbeit eingereicht im Rahmen der Bachelorprüfung
im Studiengang Angewandte 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. Jürgen Reichardt

Abgegeben am 01. September 2009

Paulin Pekezou Fouopi

Thema der Bachelorarbeit

Extrahierung von stereoskopischer Tiefeninformation modelliert für eine SoC-Plattform

Stichworte

FPGA, autonome Fahrzeuge, Bildverarbeitungsplattform, System-on-Chip, IIC-Businterface, ADV7183B-Controller, Antikollisionssystem, Bildtiefenauswertung, stereoskopisch, IIR-Filter, TFT-Monitor, Matlab.

Kurzzusammenfassung

Diese Arbeit behandelt die Entwicklung einer FPGA-basierten System-on-Chip Bildverarbeitungsplattform, die sukzessive zu einem Antikollisionssystem für autonome Fahrzeuge weiterentwickelt wird. Das System besteht aus einem Softwaremodul, das zur Parametrierung des ADV7183B-Controllers über ein IIC-Businterface dient. Eine Kette von Hardwaremodulen dient zur Verarbeitung des Videostroms aus dem ADV7183B-Controller und zur Anzeige des verarbeiteten Videostroms auf einen TFT-Monitor. Das Matlab-Modell einer Methode zur Bildtiefenauswertung als Phasenverschiebung in stereoskopisch gewonnenen Bildern durch Einsatz von IIR-Filtern wird implementiert.

Paulin Pekezou Fouopi

Title of the paper

Extraction of stereoscopic depth information modelled for SoC platform

Keywords

FPGA, autonomous vehicles, video signal processing, system-on-chip, IIC bus interface, ADV7183B controller, anti-collision system, stereoscopic, depth, IIR-filters, TFT monitor, Matlab.

Abstract

This work deals with the development of an FPGA-based system-on-chip image processing platform, which is gradually developing into an anti-collision system for autonomous vehicles. The system consists of a software module that is used to parameterize the ADV7183B controller via an IIC bus interface. A chain of hardware modules used for processing the video stream from the ADV7183B controller and for displaying the processed video stream to a TFT monitor. The Matlab model of a phase based method for stereoscopic depth analysis which utilizes IIR-filters is implemented.

Inhaltsverzeichnis

1 Einführung

Im Rahmen des FAUST-Projekts (Fahrerlose Autonome Transportsysteme) im Department Informatik der Hochschule für angewandte Wissenschaft (HAW) Hamburg, werden Software im Bereich des Fahrerassistenzsystems und autonomer Fahrzeuge entwickelt. Das autonome Fahren wird zur Teilnahme an dem Carolo-Cup in Modellfahrzeugen implementiert. Mit Sensorik, Bildverarbeitung, Modellierung und FPGA basierter Signalverarbeitung werden Brems- und Ausweichsystemen entwickelt. Die Entwicklung von Sensorik- und Telemetrielösungen für den HAWKS Racing Rennwagen, welche in Echtzeit die Fahrdaten misst, überträgt und bewertet ist Teil dieses Projektes [Vgl. HAW-FAUST 2009].
Bei der Entwicklung von Fahrerassistenzsystemen in der Kraftfahrzeugindustrie stehen die Anforderungen an eine hohe Rechengeschwindigkeit der Komponenten für die digitale Signalverarbeitung (DSP), sowie geringe Stückkosten für die Massenproduktion im Mittelpunkt. FPGA basierte System on Chip (SoC) Plattformen können diese gegensätzlichen Anforderungen mit parallelen DSP-Funktionselementen und integrierten Prozessoren bei niedrigen Taktfrequenz erfüllen. Dabei werden zu Simulationszwecken PC basierte Modelle mit beispielweise Matlab Simulink entwickelt, um eine schnelle Übersicht zu erhalten und verschiedene Varianten zu testen. Mit Xilinx Tools, wie dem *System Generator* oder *AccelDSP Codegenerator*, werden diese Modelle zur Echtzeit FPGA basierten Hardwareimplementierung transformiert, was eine Schnittstelle zwischen PC und HW basierte Entwicklung liefert [Vgl. Bagni 2008].
Im Rahmen des *Carolo-Cup* Projekts soll diese Bachelorarbeit zum Aufbau einer SoC-Bildverarbeitungsplattform beitragen, die sukzessive zu einem Antikollisionssystem für das Modellfahrzeug weiterentwickelt wird. Es werden Vorstudien zu einer Bildtiefenauswertung mit einem speziellen Filteransatz vorgestellt, der Objektentfernungen als Phasenverschiebungen in stereoskopisch gewonnenen Bildern identifiziert [Vgl. Porr 2002]. Folgende Komponenten wurden im Rahmen dieser Arbeit entwickelt und werden sukzessiv dargestellt [Vgl. Abb. 1.1]:

- Die Kopplung einer PAL Kamera FCB-IX11AP mit analogen Y/C Videosignal über das *Video Decoder Board* (VDEC1) an die SoC-Plattform Nexys2. Dafür wird auf einem Xilinx Spartan3E FPGA ein Modul, zur Konfiguration eines µControllersystems mit dem Softcore-Prozessor MicroBlaze über das IIC-Businterface, implementiert.
- Die Ausgabe des digitalen ITU-R BT.656 kompatiblen *Interlaced*-Videostroms des VDEC1-Boards mit einem VGA-Controller auf einem TFT-Bildschirm.
- Erprobung einer parallelen Filterstruktur zur Messung der Phasenverschiebung in stereoskopischen Bildern mit Matlab.

Die Arbeit ist wie folgt aufgebaut:
Im Kapitel 2 wird das System dargestellt. Eine Beschreibung der Hardwarekomponenten, sowie die Inhalte und Aufgaben aller Hardware- und Softwarefunktionsblöcke, werden erläutert. Der auf einem MicroBlaze µProzessor basierende eingebettete Entwurf wird behandelt und das Xilinx Entwicklungstool *Embedded Development Kit* (EDK) wird kurz dargestellt.
Die Hardware- (HW) und Software (SW) -Module zur Parametrierung des Video-Decoders werden im Kapitel 3 erläutert.

Die HW-Module zur Extrahierung und Bearbeitung des Pixelstroms werden im Kapitel 4 vorgestellt, wobei eines dieser Module mit dem Pixelstrom aus einem Testbildgenerator und aus dem VDEC1 entsprechend implementiert wird. Das SW-Modul zur Parametrierung des VGA-Controllers und zur Ausgabe auf dem TFT-Bildschirm wird ebenfalls hier beschrieben.
Im Kapitel 5 werden Begriffe zur Disparität in stereoskopischen Bildern und Konzepte zur Bestimmung der Disparität dargestellt. Die HW basierte Methode wird vorgestellt und eine Simulation mit Matlab Simulink wird durchgeführt.

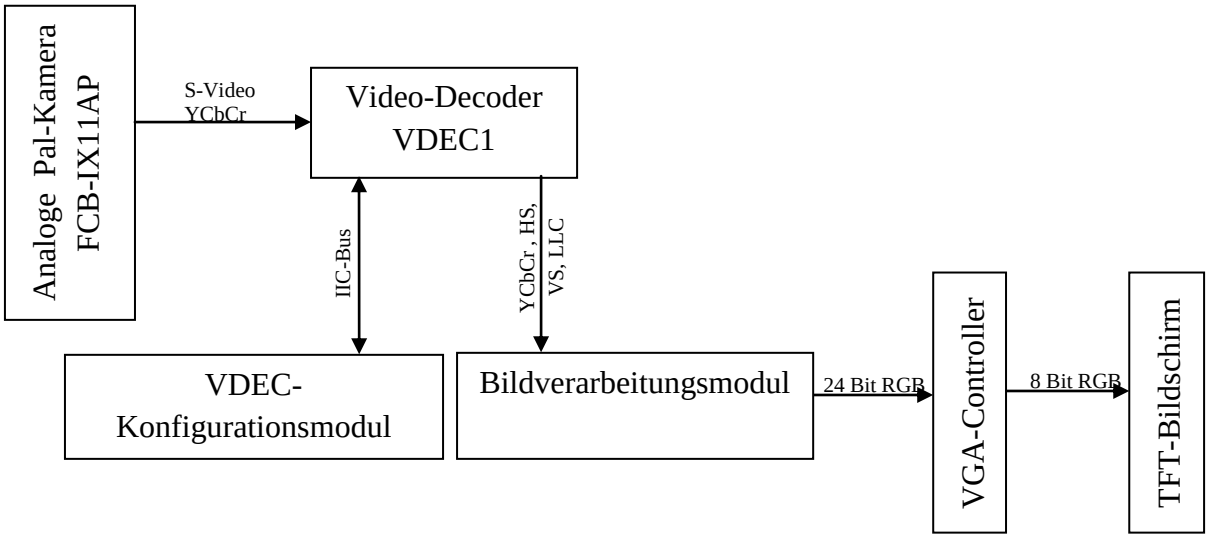

Abbildung 1.1: Systemübersicht zur Parametrierung des VDEC1-Boards und Verarbeitung von Videosignalen

2 Übersicht des Systems zur Parametrierung des VDEC1-Boards und Verarbeitung von Videosignalen

In diesem Kapitel wird eine gesamte Übersicht des Systems dargestellt. Im ersten Abschnitt werden die HW-Komponenten, deren Eigenschaften und Funktionalitäten beschrieben. Der zweite Abschnitt stellt den auf einem MicroBlaze Prozessor basierten eingebetteten Entwurf dar. Die HW und SW-Module werden im dritten Abschnitt behandelt und das Xilinx Entwicklungstool *Embedded Development Kit* (EDK) wird im letzen Abschnitt kurz zusammengefasst.

2.1 HW-Komponenten

In diesem Abschnitt werden alle HW-Komponenten des Systems [Vgl. Abb. 2.1, sowie Abb. 2.2] dargestellt, und zwar das Nexys2 Board [Vgl. Abb. 2.8], das VDEC1-Board [Vgl. Abb. 2.6] und die analoge Kamera FCB-IX11AP [Vgl. Abb. 2.4].

Abbildung 2.1: Hardwareaufbau des Systems.

2.1.1 FCB-IX11AP Kamera und ihr Circuitboard

Das Circuitboard der Kamera ist die Schnittstelle zwischen der Kamera und den externen Komponenten, die mit ihr verbunden sein müssen. Es stellt folgende Schnittstellen zur Verfügung, die in 2 Gruppen geteilt werden [Vgl. Sony 2005b, sowie Abb. 2.3]:

Gruppe 1: Schnittstellen zur Kommunikation mit externen Komponenten:
1. Visca Socket-8Pol. Mini DIN zur Kommunikation mit dem PC über die serielle Schnittstelle.

2. DIP- Switch RS / TTL Level zur Umstellung zwischen RS232 und TTL.
3. Video Output- BNC Socket zur Verbindung mit dem BNC-Kabel.
4. S-Video Output 4 Pins Mini-DIN zur Verbindung mit dem S-Video-Kabel.
5. Key Lock für das Keypanel.
6. DC- Socket (5,5mm / 2,5mm) für die Stromversorgung.

Gruppe 2: Schnittstellen zur Kommunikation mit der Kamera:
7. 12 Pins FFC Anschlüsse für die Keypanelfunktionen Zoom sowie Focus Matrix.
8. 9 Pins FFC Anschlüsse zur Stromversorgung und Videosignalübertragung.
9. Visca Connection OFF (Switch UP) / ON (Switch Down).
10. 10 Pins Anschlüsse zur Verbindung der Kamera mit der RS232, sowie TTL.
11. 10 Pins FFC Anschlüsse für das Keypanel.

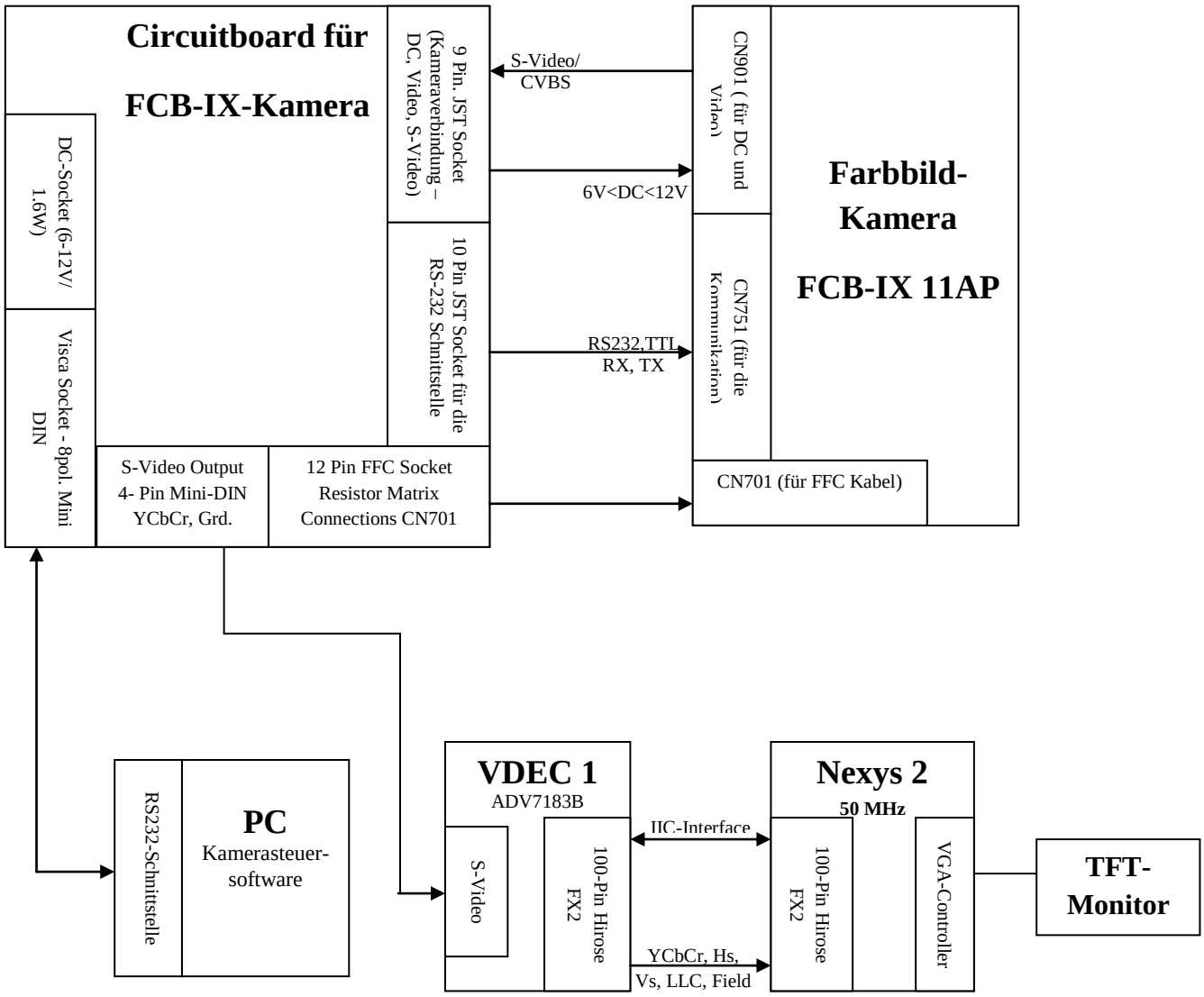

Abbildung 2.2: Schematischer Aufbau des Systems.

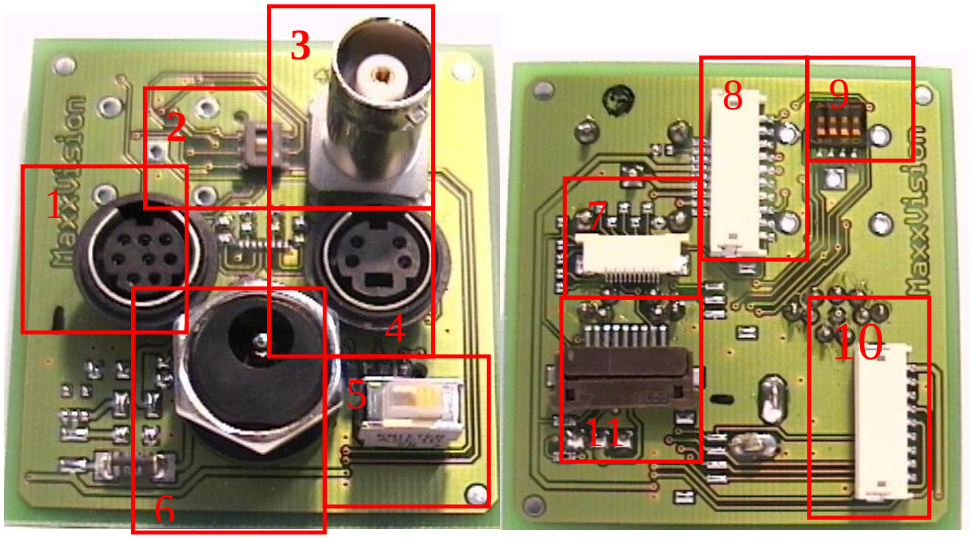

Abbildung 2.3: Circuitboard der FCB-IX11AP-Kamera zur Verbindung mit externen Komponenten [Sony 2005b].

Die FCB-IX11AP ist eine analoge PAL Farbbild-Kamera der Firma Sony, die mit ihrer On-Board digitalen Signalverarbeitungsfeature Bilder mit höherer Qualität erzeugen kann. Ihre geringere Größe, ihr kleines Gewicht, ihre gute Empfindlichkeit gegenüber Bewegungen und Lichtstärke sowie ihr niedriger Stromverbrauch bilden eine ideale Auswahl für eingebettete Systeme [Vgl. Sony 2005a, sowie Abb. 2.4]. Folgende Komponenten sind auf der Kamera zu finden [Vgl. Abb. 2.4]:

1. Die Linse.
2. Der CN901 Konnektor zur Spannungsversorgung und Videosignalübertragung.
3. Der Wide-Knopf.
4. Der TELE-Knopf.
5. Der CN701 Konnektor für die Keypanelfunktionen.
6. Der CN751 Konnektor für die RS232-, sowie die TTL-Schnittstelle.
7. die Stativschraubenlöcher.

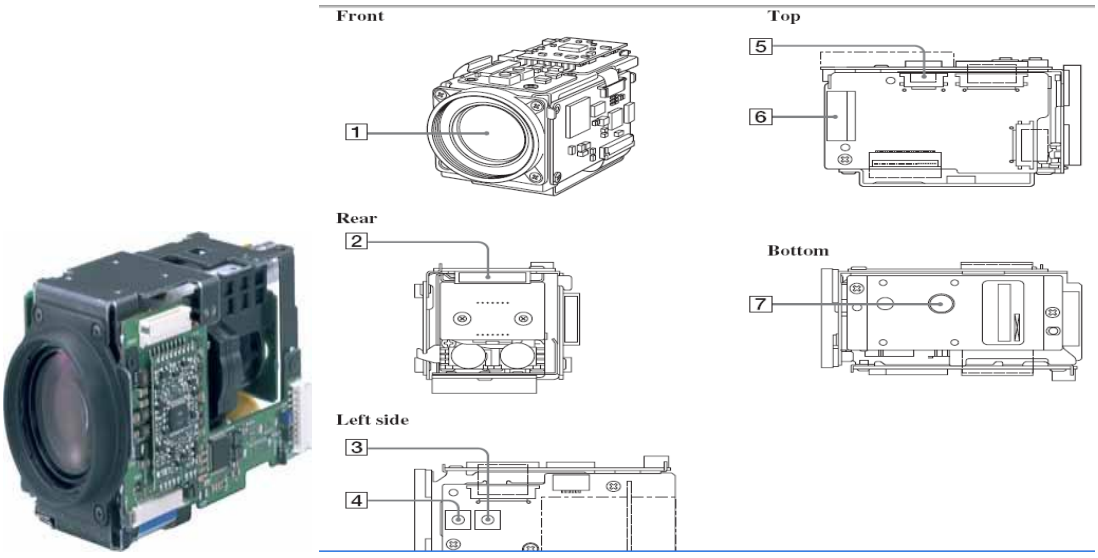

Abbildung 2.4: Seitenansichten der FCB-IX11AP-Kamera [Sony 2005a].

Die wichtigen Eigenschaften dieser Kamera sind in der Tabelle 2.1 beschrieben [Vgl. Sony 2005a, sowie Sony 2004].

Tabelle 2.1: Eigenschaften der FCB-IX11AP Kamera

Sensor	1/4-Typ EX-view HAD CCD
Auflösung	Ca. 768(H) x 582(V)
Horizontale Auflösung	460 TV Zeile
Linse	10 x Zoom, Fokuslänge: 4.2 mm (WIDE) und 42 mm(TELE)
Minimale Lichtstärke	1.5 Lux
Digitaler Zoom	4x bzw. 40x mit dem optischen Zoom
Sichtwinkel	40° (WIDE end) und 4.6° (TELE end)
Minimale Entfernung	10mm(WIDE end) und 1000mm (TELE end)
Fokus	Auto (AF), On-Push AF, Manuel sowie Unendlich
Video Output	(C)VBS (Composite) sowie Y/C(S-Video) Interlaced
Serielle Schnittstelle	TTL und RS232 9600 Baud
Versorgungsspannung	Zwischen 6 und 12 V DC/ 2W (2.0 W beim laufenden Motor)
Gewicht/Größe	95g/39.3 x 44.8 x 65 mm (Höhe x Breite x Tiefe)

Die Kamera arbeitet im sogenannten *Interlaced*-Verfahren, was bedeutet, dass die Bildzeilen nicht sukzessiv übertragen werden, sondern erst die ungeraden und anschließend die geraden Zeilen. Der Vorteil ist hier, dass für ein Bild nur die Hälfte der Informationen übertragen wird. Hierbei gehen Informationen verloren, so dass bei schnellen Bewegungen das Bild verschwommen wirkt. Deshalb werden heute sowohl das *Interlaced*- als auch das *Progressive*-Videosignal eingesetzt [Vgl. Jack 2005].
Zur Parametrierung der Kamera stehen zwei Möglichkeiten zur Verfügung. Zum einen die mitgelieferte Software und zum anderen die Entwicklung eigener Applikation [Vgl. Peters 2009]. In beiden Fällen basieren die Applikation auf dem VISCA-Protokoll, das im Folgenden kurz zusammengefasst wird, da für diese Arbeit die erste Möglichkeit ausgewählt wurde.
Das VISCA-Protokoll ist ein Kommunikationsprotokoll, das das Verfahren zur Steuerung der Kamera mit einem entfernten Rechner spezifiziert. Der Sender (in diesem Fall der Desktop PC) heißt Controller und der Empfänger (hier die Kamera) ist das Peripheriegerät, wobei bis zu sieben Empfänger in einer *Daisy Chain* aufgebaut werden können. Zur Verbindung wird die seriellen Schnittstelle RS232 mit folgenden Eigenschaften benutzt [Vgl. Abb. 2.5]: 9600 Baud, 8 Bit Daten, 1 Start Bit, ½ Stop Bit und keine Parität [Vgl. Sony 2004].
Ein VISCA-Paket besteht aus 1 Byte Header (Sender- und Empfängeradresse), 1 bis 14 Bytes Daten und 1 Byte für das Packetende, wobei der Sender immer die Adresse 0 hat und das erste Bit des Headers immer 1 ist. Am Anfang der Kommunikation bekommen die Empfänger ihre Adresse per Broadcast, danach werden die Puffer des Empfängers gelöscht. Der Sender kann dann die Kameraparameter wie zum Beispiel das Power On/Off, die Fokuslänge oder den Fokusmodus ändern, sowie deren Zustand abfragen und entsprechend die Antwort des Empfängers bekommen [Vgl. Sony 2004].

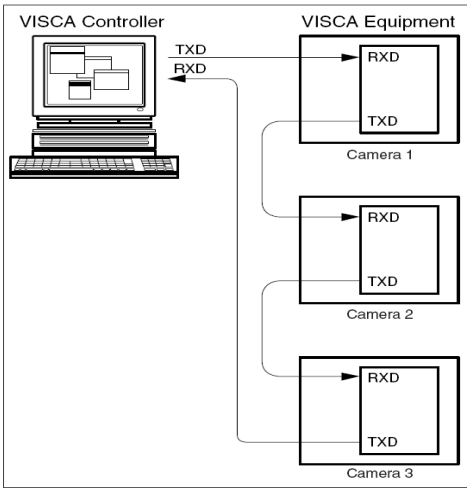

Abbildung 2.5: Aufbau eines VISCA-Netzwerks zur Parametrierung der Kamera [Sony 2004].

2.1.2 Der Video-Decoder VDEC1

Das VDEC1-Board ist ein auf dem Controller ADV7183B basierter Analog-Digitalumsetzer, der analoge Videosignale vom Typ PAL, SECAM, sowie NTSC mit drei 50/54 MHz ADC in einen 8 oder 16 Bit ITU-R BT.656 YCbCr 4:2:2 *Interlaced*-Videosignal digitalisiert. Dabei werden die Synchronisationssignale HS, VS, Field, sowie die Pixelclock LLC am Ausgang zur Verfügung gestellt [Vgl. Digilent 2005a]. Folgende Komponenten sind Bestandteil des Boards[Vgl. Abb. 2.6]:

1. CVBS: *Composite*-Eingang.
2. YPrPb: *Component*-Eingang.
3. Y/C: S-Video-Eingang.
4. µController ADV7183B.
5. ALSB-Jumper zur Auswahl der Adresse für die IIC-Bus-Kommunikation.
6. 27 MHz Clock LLC.
7. Der FX-2 Konnektor.

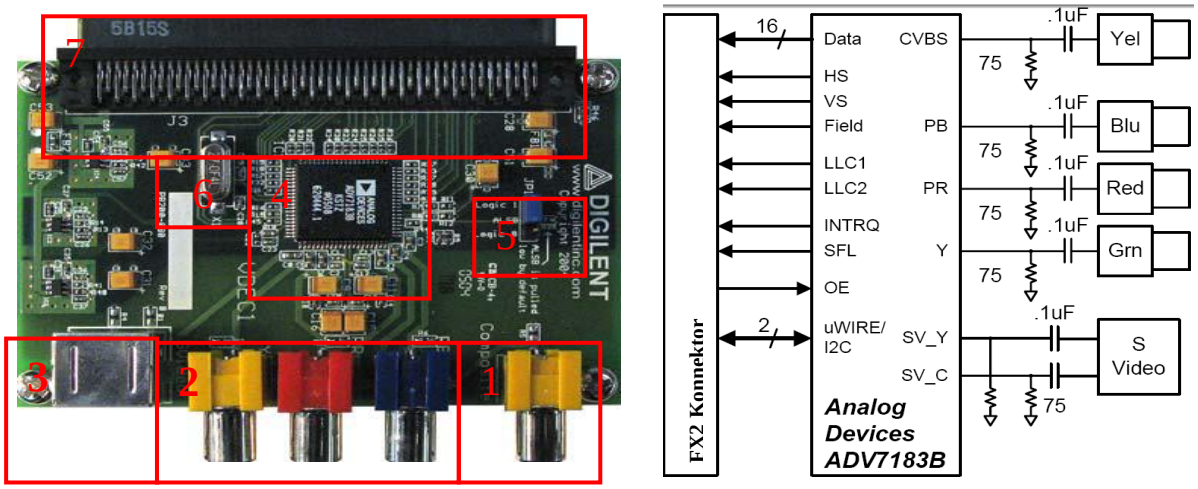

Abbildung 2.6: HW-Aufbau (links) und funktioneller Aufbau (rechts) des VDEC1– Boards [Digilent 2005a].

Damit der ADV7183B-Controller in dem richtigen Modus arbeiten kann, muss er, nachdem das VDEC1-Board mit dem FX2-Konnektor an das Nexys2 gekoppelt wird, über das in dem FX2-Konnektor eingebetteten IIC-Businterface konfiguriert werden [Vgl. Digilent 2005a, sowie Abb. 2.6]. Dabei ist bei der Ankopplung des VDEC1-Boards und des Nexys2 zu beachten, dass das Nexys2-Board nicht die Pins des FX2-Konnektor als Ausgang benutzt, die gleichzeitig auch dem VDEC1-Board als Ausgang dienen, damit die beiden Boards nicht beschädigt werden [Vgl. Digilent 2005a]. Darüber hinaus muss der *Jumper* neben dem FX2-Konnektor des Nexys2-Boards entsprechend gesetzt werden, damit das VDEC1-Board ausreichend mit Spannung versorgt wird.
Die Parametrierung des ADV7183B-Controllers erfolgt über zwei serielle bidirektionale Leitungen, die mit dem IIC-Bus kompatible ist. Dadurch werden die *Global Control* Registers des Controllers gesetzt bzw. gelöscht. Der Zustand des Controllers kann über die *Global Status* Register abgefragt werden. Vor dieser Parametrierung wird eine Resetsequenz durchgeführt, damit die Register des Controllers initialisiert werden können. Dafür wird der Resetpin (*low-activ*) für mindesten 2 ms auf 0 und danach auf 1 gesetzt. Während dieser Sequenz sind der Power-Downpin(*low-activ*) auf 0 und optional der OE-Pin(*low-activ*) auf 1 zu setzen, und es kann keine IIC-Buskommunikation stattfinden. Es ist empfohlen noch 5 ms nach dieser Sequenz zu warten, bevor die IIC-Buskommunikation gestartet wird [Vgl. ANALOG-DEVICES 2005].
Wenn die Resetsequenz abgeschlossen ist, wird die Parametrierung gestartet, indem das Nexys2-Board (*Master*) eine IIC-Buskommunikation mit der ADV7183B (*Slave)* initiiert. Je nach der Position des ALSB-Jumpers [Vgl. Abb. 2.6] hat der ADV7183B entweder die Adresse 0x40 für das Lesen bzw. 0x41 für das Schreiben oder 0x42 für das Lesen bzw. 0x43 für das Schreiben. Die Tabelle der zusendenden Kommandos für verschiedene Eingangsvideosignale ist in der Dokumentation zu finden und wird später genauer beschrieben [Vgl. ANALOG-DEVICES 2005).

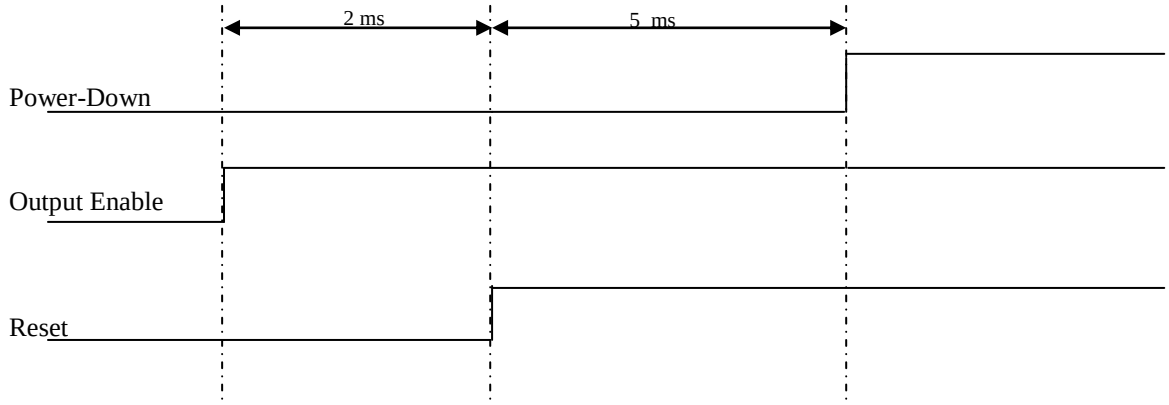

Abbildung 2.7: Timing der Resetsequenz

2.1.3 Nexys2-Board

Das Nexys2-Board ist ein Entwicklungsboard der Firma Digilent, das auf einem Xilinx Spartan 3E FPGA basiert ist. Es ist mit allen Xilinx ISE Tools kompatibel. Folgende Komponenten stellt dieses Board zur Verfügung [Vgl. Digilent 2008, sowie Abb. 2.8]:
1. Xilinx Spartan3E XC3S1200 FPGA.
2. 16 MB Mikron SRAM.
3. RS232-Schnittstelle.

4. Pmod Schnittstellen (jeweils 8 Daten-, 2 Masse- sowie 2 Power-Pins).
5. VGA-Schnittstelle
6. USB2-Schnittstelle
7. PS2-Schnittstelle
8. FX-2 Konnektor
9. 8 LEDs, 8 Schiebeschalter, 4 Drucktaster und eine vierstellige Siebensegmentanzeige
10. 50 MHz Oszillator

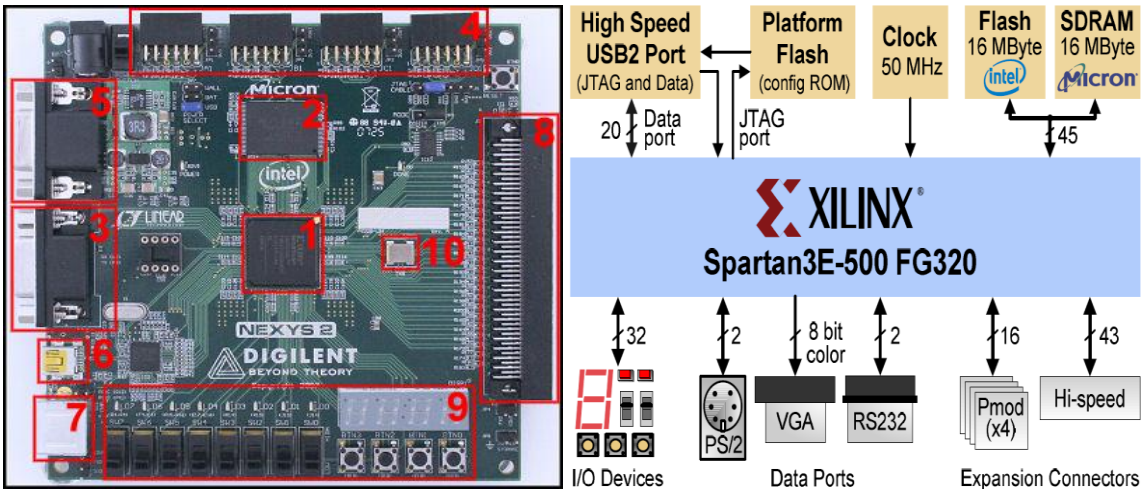

Abbildung 2.8: HW-Aufbau (links) und funktioneller Aufbau (rechts) des Nexys2-Boards [Digilent 2008].

Wenn die Programmierung des Boards über den USB2-Port geführt wird, muss zuerst die Software *Adept* von Digilent installiert werden. Dabei werden auch die Treiber für das Board installiert, sodass es vom PC erkannt wird. Der *Configuration Mode* Jumper muss vor der Programmierung auf *JTAG* gestellt werden. Für die Programmierung über den *JTAG Header* kann die Software *iMPACT* von Xilinx benutzt werden [Vgl. Digilent 2008]. Das Spartan 3E FPGA, was auf dem Nexys2-Board zu finden ist, ist auf das vorherige Spartan 3 FPGA gebaut, und erfüllt die Anforderungen an den Aufwand bei niedrigen Kosten heutiger elektronischer Anwendungen, indem die Anzahl der Logik per I/O und daher die Kosten pro logischen Zellen reduziert wird. Darüber hinaus verbessern neue Features die Systemleistung und reduzieren die Konfigurationskosten. Diese Verbesserungen, kombiniert mit der erweiterten 90-Nanometer-Prozesstechnologie bietet mehr Funktionalität sowie Bandbreite pro Kosten als je zuvor an [Vgl. Xilinx 2008a].
Funktional betrachtet, sieht das Spartan 3E wie folgt aus [Vgl. Xilinx 2008a sowie Abb. 2.9]:

- Die *Configurable Logic Blocks* (CLBs) bestehen aus *Look-Up* Tabelle (LUTs), die die Logik und die Speicherelemente wie *Flip-Flops* oder *Latches* implementieren. Sie bieten also eine vielfältige Anzahl an logischen Funktionen sowie Datenspeicherelemente an.
- Die *Input/Output Blocks* (IOBs) verwalten den Datenfluss zwischen den I/O-Pins und der internen Logik. Dabei unterstützen sie sowohl bidirektionale Datenflüsse als auch *Three-State* Treiber, sowie vielfältige Signalstandards. Sie sind als Ring gebaut und umkreisen den CLBs-Array.
- Der Block RAM bietet Datenspeicherung in der Form von 18 KBit *dual-port* Blocks an. Der ist in zwei Spalten geteilt und jede Spalte ist mit einem dedizierten *Multiplier* verbunden.

- Die *Multiplier Blocks* multiplizieren zwei 18 Bit binär Zahlen.
- Die *Digital Clock Manager (*DCM*) Blocks* bieten eine voll digitale Lösung zur Verteilung, Verzögerung, Multiplikation, Division sowie Phasenverschiebung des Clocksignal an. Auf dem FPGA sind zwei in der Miete oben und zwei andere unten verteilt.

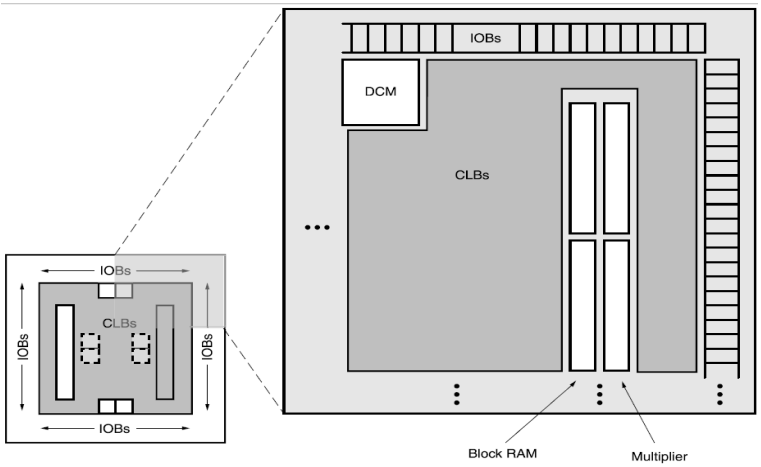

Abbildung 2.9: Funktionaler Aufbau des Spartan 3E FPGA [Xilinx 2008a].

2.2. MicroBlaze µProzessor

Der MicroBlaze µProzessor ist ein Softcore µProzessor. Mit einer RISC-Architektur ist er für die Implementierung in Xilinx FPGA optimiert. Eine seiner Eigenschaften ist, dass er hoch parametrierbar ist und die Möglichkeit bietet, andere Features auszuwählen und zu konfigurieren, wobei folgende Features unumgänglich sind und nicht geändert werden können [Vgl. Xilinx 2008b, sowie Abb. 2.11]:

- zwei und dreißig 32 Bit breite *General Purpose Registers*(GPR).
- 32 Bit Instruktionswort.
- 32 Bit Adressbus sowie eine *Single issue* drei bis fünft stufige Pipeline.

Im Gegenteil dazu sind folgende Komponenten konfigurierbar [Vgl. Xilinx 2008b sowie Abb. 2.10 und Abb. 2.11):

- Die Anzahl der Pipelinestufen des Prozessors kann zwischen drei und fünf gewählt werden.
- Eine Instruktionscache und eine Datencache, sowie deren Größe kann gewählt werden. Auf dem FPGA wird der *block* RAM (BRAM) dafür verwendet.
- *Der Local Memory Bus* (LMB) ist ein 32Bit Bus zum Zugriff auf dem On-Chip BRAM.
- *Off-Chip Memory:* der MicroBlaze Prozessor verfügt über einen doppelt gekoppelte *Off-Chip* Flash/SRAM Speichercontroller für eine verzögerungslose Übertragung von Daten mit höherer Datenrate.

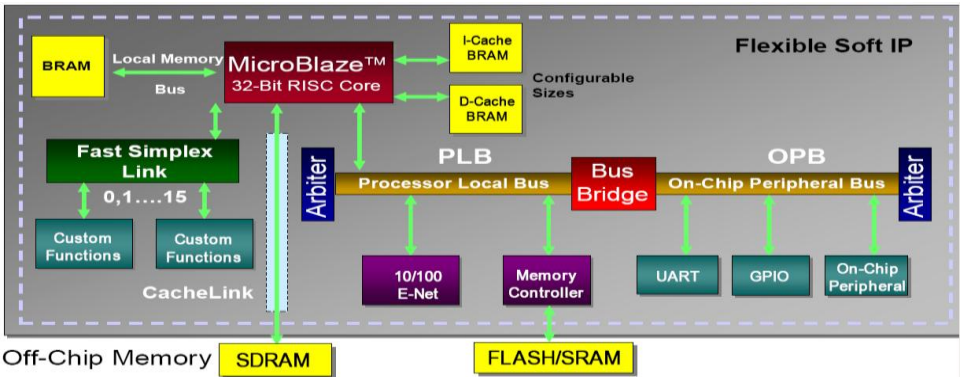

Abbildung 2.10: Aufbau eines MicroBlaze Prozessor basierten eingebetteten Entwurfs

- *Fast Simplex Link* (FSL) Chanels sind unidirektionale 32 Bit breite Punkt zu Punkt Datenübertragungsinterface, die jeweils von 0 bis 16 als Input als auch Output auf dem Prozessor verfügbar sind. Der FSL wird also zur direkt Verbindung zwischen zwei IPs benutzt(PLB wird entlastet).
- *Processor Local Bus (PLB)*Interface besteht aus einem 32 Bit breiten Adressbus und drei 64 Bit breiten Datenbussen. Zwei von den Datenbussen sind an die Datencacheeinheit für das Lesen beziehungsweise Schreiben gebunden und einer ist mit der Instruktionscacheeinheit verbunden. Der PLB-Bus bietet also eine hohe Bandbreite für eine verzögerungslose Verbindung zwischen schnellen Peripheriegeräten, sowie dem Speicher und dem Prozessor.

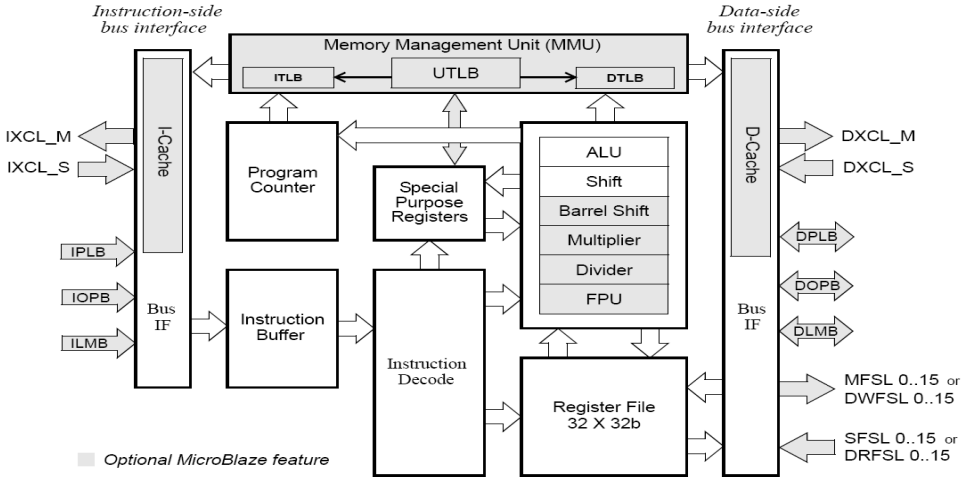

Abbildung 2.11: Aufbau des MicroBlaze Prozessors [Xilinx 2008b].

2.2 Hardware- und Softwaremodule

In diesem Abschnitt werden Hardware und Softwaremodule, die in dieser Arbeit entwickelt wurden dargestellt. Zuerst wird eine Beschreibung des VDEC1-Board-Parametrierungsmoduls dargestellt. Danach werden die Softwaremodule zur Extrahierung Verarbeitung und Ausgabe des Videosignals kurz zusammengefasst.

2.2.1 IP reset_vdec

Der IP *reset_vdec* ist ein Hardwaremodul mit Software-Treiber, der zur Parametrierung des VDEC1-Boards beiträgt. Da das VDEC1-Board vor der Parametrierung erst initialisiert sein soll, erzeugt dieser IP eine Initialisierungssequenz [Vgl. ANALOG-DEVICES 2005]. Dieses Hardwaremodul wird über ein Softwareregister von dem Softwaremodul *IIC-Schnittstelle* angefordert, um eine Initialisierungssequenz zu starten. Der IP generiert dann ein Resetsignal auf dem Resetpin des VDEC1-Boards und informiert das Softwaremodul *IIC-Schnittstelle* über das Ende der Sequenz, damit die Parametrierung gestartet wird [Vgl. Abb. 2.12].

2.2.2 SW-Modul IIC-Schnittstelle

Dieses Modul dient zur Parametrierung des Controllers ADV7183B auf dem VDEC1-Board. Es fordert eine Resetsequenz an, indem ein Startbit für den IP *reset_vdec* in einem SW-Register gesetzt wird [Vgl. Abb. 2.12]. Nachdem diese Sequenz abgeschlossen ist, wird der Controller durch das Senden bestimmter Kommandos über dem IIC-Bus parametriert. Für die IIC-Buskommunikation wird der von Xilinx mitgelieferte XPS IIC-Businterface IP-Core verwendet. Nach der Parametrierung wird die Digitalisierung gestartet und gültige digitale Videosignale, sowie deren Pixelclock stehen am Ausgang des VDEC1 zur Verfügung.

2.2.3 Digitale Clock Manager (DCM)

Der DCM ist ein HW-Modul, das zur Verarbeitung von Clocksignale auf Spartan 3 FPGAs dient. Er eliminiert das Clock-Skew und verbessert dabei die Systemleistungen. Optional kann die Clock-Phase verschoben, die Frequenz multipliziert oder dividiert werden. Die daraus resultierende Clock wird direkt in den auf dem FPGA für die Clock reservierten Wegen gemapt. In dieser Arbeit wird der vom Xilinx *Core Generator* generierte DCM-Modul zur Verdoppelung der Pixelclock (von 13.5 MHz auf 27MHz) aus dem VDEC1 eingesetzt [Vgl. Abb. 2.12 sowie Xilinx 2008d].

2.2.4 Sync-Modul

Dieses Modul hat als Input das 16 Bit ITU-R BT.656 YCbCr 4:2:2 *Interlaced*-Videosignal, sowie die HS-, VS-, OE- und FIELD-Signale, die dazu dienen, die Synchronisationssignale LVAL und FVAL zu erzeugen. Die Videosignale Y und CbCr werden gemäß dem ITU-R BT.656 Protokoll aus dem Datenstrom extrahiert und mit den beiden erzeugten Synchronisationssignalen LVAL und FVAL weitergeleitet [Vgl. Jack 2005].

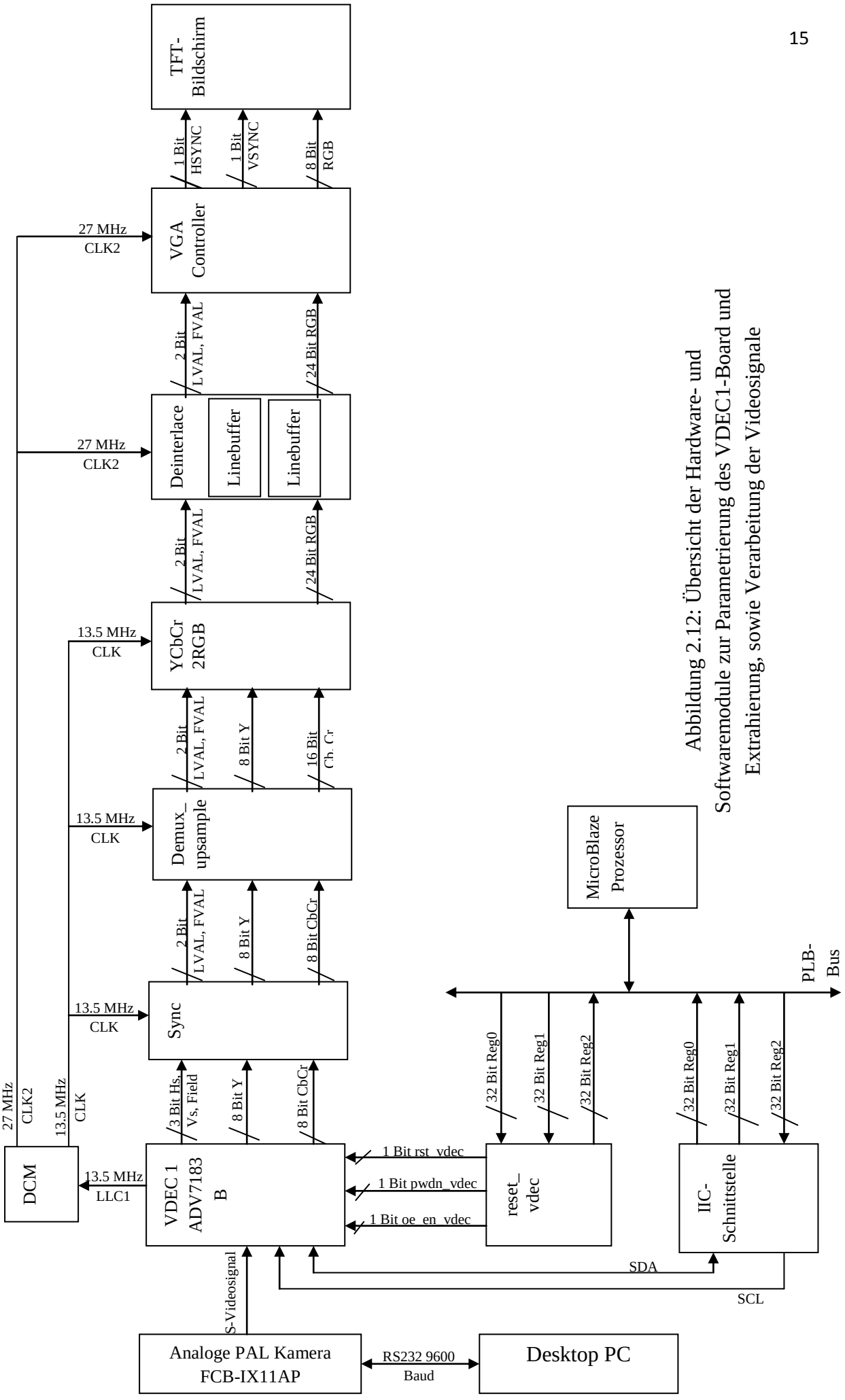

Abbildung 2.12: Übersicht der Hardware- und Softwaremodule zur Parametrierung des VDEC1-Board und Extrahierung, sowie Verarbeitung der Videosignale

2.2.5 Demux_upsample-Modul

Hier wird das 16 Bit YCbCr 4:2:2 *Interlaced*-Videosignal in 24 Bit YCbCr 4:4:4 umgewandelt. Dafür werden die fehlenden Cb- und Cr-Werte durch eine Pixelreplikation berechnet [Vgl. Poynton 2005]. Das resultierende *Interlaced*-Videosignal ist 24 Bit YCbCr 4:4:4. LVAL und FVAL werden ohne Änderungen weitergeleitet.

2.2.6 YCbCr2RGB-Modul

Dieses Modul fasst die Umwandlung von 24 Bit YCbCr *Interlaced*-Videosignal in 24 Bit RGB *Interlaced*-Videosignal. Da der Wert von Y, zwischen 16 und 235 und bei Cb und Cr zwischen 16 und 240 liegt wird darauf geachtet, dass die RGB-Werte zwischen 0 und 255 liegen [Vgl. Jack 2005]. LVAL und FVAL werden ohne Änderungen weitergeleitet.

2.2.7 Deinterlace-Modul

Zur Anzeige auf einem 60 Hz TFT-Monitor wird das *Interlaced*-Videosignal zeilenweise verdoppelt, sodass eine Zeile zwei Mal angezeigt wird [Vgl. Jack2005]. Dabei werden zwei Schiebenregister zur Zwischenspeicherung der Pixelzeile benutzt und die entsprechenden LVAL und FVAL-Signale werden nebenbei erzeugt.

2.2.8 VGA-Controller-Modul

Damit der auf dem Nexys2-Board verfügbare VGA-Controller das 24 Bit RGB *Deinterlaced*-Videosignale auf einem TFT-Monitor ausgeben kann, werden die dafür nötige Synchronisationssignale HS und VS Erzeugt. Hier wird das 24 Bit RGB Videosignal auf 8 Bit reduziert, da der VGA-Controller nur zehn Leitungen zur Verfügung stellt wobei, zwei für HS und VS reserviert sind und acht für die Daten.

2.3 Xilinx Embedded Development Kit (EDK)

EDK ist eine Software der Firma Xilinx, die die Tools, die Dokumentation sowie die IPs für die gesamte Entwicklung eingebetteter programmierbarer Systeme mit dem Hardcore Prozessor PowerPC von IBM und/oder dem Softcore MicroBlaze Prozessor von Xilinx zur Verfügung stellt. Dabei werden die Hardware- und Softwareteile eines SoC-Systems separat implementiert und zusammen integriert. EDK enthält folgende Tools [Vgl. Xilinx 2009]:

- *Xilinx Plattform Studio* (XPS) bietet eine Entwicklungsumgebung für den Entwurf eingebetteter Systeme, die auf dem MicroBlaze oder PowerPC µProzessor basieren. Darin enthalten ist sowohl ein Interface zur Projektverwaltung und Source Codebearbeitung, als auch eine graphische Oberfläche für die Verbindung von Bussen, Prozessoren und Peripheriegeräten [Vgl. Abb. 2.13].
- *Base System Builder (*BSB*) Wizard* erzeugt ein Standardprojekt. Für ein gewähltes Board hilft der Wizard bei der Wahl und Parametrierung von Systemelementen wie dem Prozessor, dem Speicher, der Cache sowie den Peripheriegeräten. Es ist empfohlen

immer mit dem Wizard ein Projekt zu erzeugen. Danach können je nach Bedarf Komponenten geändert oder hinzugefügt werden.

- *Create and Import Peripheral* (CIP) *Wizard* unterstützt bei dem Entwurf und der Inbetriebnahme selbst entwickelter Peripherie. Dafür erzeugt der Wizard ein Interface, das die XPS-Anforderungen, wie das Kommunikationsprotokoll oder die Namenskonvention, überdeckt. Gleichzeitig werden zwei für die Peripherie wichtige Dateien erzeugt und zwar die *Microprocessor Peripheral Definition* (MPD)-Datei (definiert das Peripherieinterface) und die *Peripheral Analyze Order* (PAO)-Datei (besagt welche Dateien für die Peripherie bei der Simulation beziehungsweise der Synthese benötigt werden).
- *Platform Generator* (Platgen) kompiliert die *high-level* Beschreibung des eingebetteten System in einer HDL-*Netlist*, die in FPGA implementiert werden kann.
- *Software Development Kit* (SDK): basiert auf *Eclipse*, dient SDK zur Entwicklung des Softwareteils des Systems.
- *Library Generator* (Libgen) erzeugt eine Softwareplattform, indem er die Bibliotheken, die Gerätetreiber, sowie das Dateisystem des eingebetteten Systems konfiguriert.
- *GNU Compiler Tools* (GCC) wird beim Kompilieren und Linken des C-Codes jedes Prozessors aufgerufen.
- Eingebettete IPs mit Prozessoren und Peripheriegeräten.

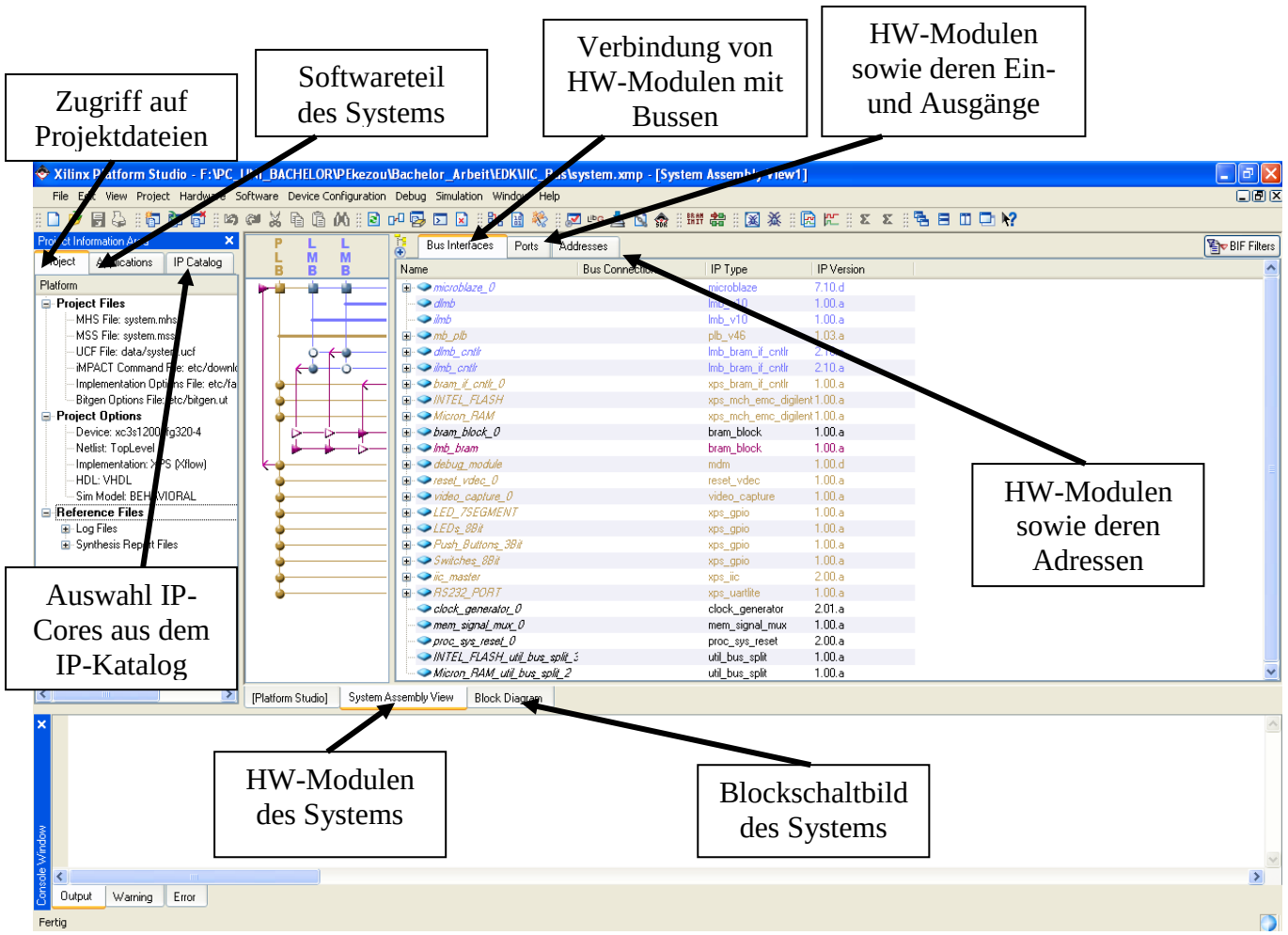

Abbildung 2.13: XPS graphischer Oberfläche

3 Parametrierung des VDEC1-Boards

Basierend auf dem Controller ADV7183B digitalisiert das VDEC1-Board analoge *Interlaced*-Videosignale aus der PAL-Kamera. Damit der Controller die Signale korrekt bearbeitet, muss er parametriert werden [Vgl. ANALOG-DEVICES 2005], was das Ziel dieses Kapitels ist. Im ersten Abschnitt wird eine konzeptionelle Übersicht der hier genutzten Methode dargestellt. Die Implementierung dieser Methode wird das Thema des zweiten Abschnitts.

3.1 Die konzeptionelle Übersicht

In diesem Abschnitt werden Konzepte zur Parametrierung der ADV7183B Controllers dargestellt. Zuerst wird die Resetsequenz erklärt. Anschließend wird die eigentliche Parametrierung erläutert.

3.1.1 Resetsequenz

Vor der Parametrierung des Controllers muss eine Resetsequenz durchgeführt werden, damit alle Register des Controllers initialisiert werden können. Dafür bietet der ADV7183B Controller zwei Möglichkeiten an [Vgl. ANALOG-DEVICES 2005]:

- Das *Reset Control* Register (RES): Es gehört zu den *Global Control* Registern und dient zur Initialisierung des Controllers. Per Default ist dieses Register auf 0 gesetzt. Das bedeutet, dass der Controller im Betriebszustand ist. Wenn dieses Register aber über den IIC-Bus auf 1 gesetzt wird, startet eine Resetsequenz.
- Der Pin *not*-RESET (*low-activ*) wirkt sich auf dem Controller genauso wie das Register RES aus. Per Default ist dieser Pin auch auf 1 gesetzt, sodass der Controller im Betriebszustand ist. Der Unterschied hier ist, dass dieser Pin auf 0 über die Hardware gesetzt werden kann, damit die Resetsequenz gestartet wird.

Da während dieser Sequenz keine IIC-Buskommunikation erfolgreich abgeschlossen werden kann, wird sie von einem Hardwaremodul durchgeführt. Dabei wird jedes Register entweder auf ihren letzen oder auf ihren Default-Wert gesetzt. Nach dieser Sequenz ist der Controller im Betriebszustand, wobei es empfohlen ist, dass diese Sequenz mindesten zwei ms dauern muss. Nach der Sequenz muss mindesten fünf ms gewartet werden, bevor die IIC-Buskommunikation gestartet wird [Vgl. ANALOG-DEVICES 2005 sowie Abb. 2.7].

3.1.2 Parametrierung des ADV7183B-Controllers

Das Ziel dieses Moduls ist die Parametrierung des Controllers über das IIC-Businterface. Der Controller stellt dafür sämtliche Register zur Verfügung die über das IIC-Businterface angesprochen werden können.
Der IIC(*Inter-Integrated Circuit*)-Bus ist ein serieller Bus der Firma Philips Semiconductors, der meistens für eine effiziente Kommunikation, zwischen

Schaltungsteilen mit geringerer Übertragungsgeschwindigkeit, eingesetzt wird. Folgende Features sind verfügbar [Vgl. Philips 2000 sowie Abb. 3.1, Abb. 3.2, Abb. 3.3]:

- Er verfügt über eine Datenleitung (SDA: *Serial Data Line*) und eine Clockleitung (SCL: *Serielle Clock Line*).
- In der Kommunikation ist der Sender *Master* genannt und der Empfänger *Slave*. Es können mehrere *Masters* und *Slaves* an dem Bus gebunden sein, wobei ihre Anzahl mit einer Buskapazität von 400 pF begrenzt ist. Dabei muss jeder Teilnehmer an der Kommunikation mit einer 7- bzw. 10 Bit Adresse eindeutig identifizierbar sein. Der Buszugriff wird durch einen *Arbiter* festgelegt.
- Die 8 Bit breite Daten können mit einer Datenrate von 100 kbit/s im *Standard-mode*, 400 kbit/s im *Fast-mode*, sowie 3.4 Mbit/s im *High-speed-mode* übertragen werden.

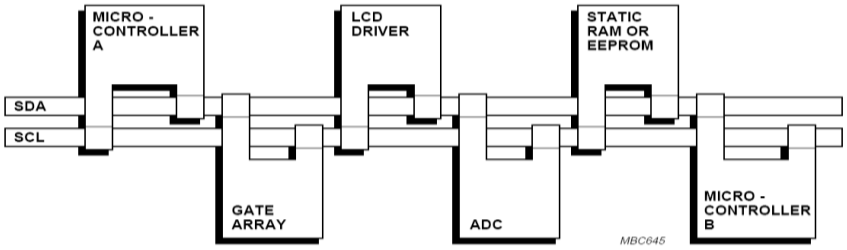

Abbildung 3.1: IIC-Bus-Aufbau mit zwei *Masters* und vier *Slaves* [Philips 2000].

Nach der Resetsequenz wird die IIC-Buskommunikation gestartet, indem das Nexys2-Board (der *Master)* den ADV7183B Controller (der *Slave)* je nach der Position des ALSB-Jumpers (Vgl. Abb. 2.6) entweder über die Adresse 0x40 für das Lesen beziehungsweise 0x41 für das Schreiben oder 0x42 für das Lesen beziehungsweise 0x43 anspricht. Dabei muss das folgende Protokoll genau befolgt werden [Vgl. ANALOG-DEVICES 2005 sowie Abb. 3.2 und 3.3]:

- Der *Master* initiiert eine Datenübertagung, indem er eine Startbedingung setzt: die Datenleitung (SDA/SDATA) geht von 1 auf 0 während die Clockleitung (SCL/SCLOCK) auf 1 bleibt. Das hat die Bedeutung, dass eine Datenübertragung gleich folgen wird.
- Der *Slave* antwortet auf diese Anfrage und der *Master* schreibt die 7 Bit-Adresse des *Slaves* in die SDA-Leitung gefolgt von 1 für Lesen oder 0 für das Schreiben.
- Wenn der *Slave* die Adresse als seine erkennt, antwortet er mit einem *Acknowledge* (ACK), indem er die SDA-Leitung in der neunten SCL-Periode auf 0 setzt. Wenn er diese nicht erkennt, geht er in den *Idle*-Zustand. Das heißt, er überwacht die SDA- und die SCL-Leitung, bis eine neue Startbedingung initiiert wird.
- Das nächste Byte enthält die Adresse des Registers (SUBADRESS), der beim *Slave* gelesen bzw. geschrieben wird. Wenn diese Adresse gültig ist antwortet der *Slave* mit ACK und wartet auf die Daten, sonst mit *not*-ACK und geht in den *Idle*-Zustand.
- Für das Schreiben, schreibt der *Master* byteweise die Daten in die SDA-Leitung. Der *Slave* antwortet nach jedem Byte mit ACK und die SUBADRESS wird inkrementiert. Wenn die maximale SUBADRESS (hier 249) erreicht wird, antwortet der *Slave* mit not-ACK und geht in den *Idle*-Zustand
- Für das Lesen auf einer bestimmten Adresse, adressiert der *Master* den Slave zuerst im Schreibmodus mit der SUBADRESS, wo der Master lesen will. Wenn der *Slave* mit ACK antwortet, schreibt der *Master* die Slaveadresse in die SDA-Leitung gefolgt von 1 für das Lesen. Der Slave antwortet mit ACK und schreibt die Daten byteweise in die SDA-Leitung. Nach jedem Byte antwortet der *Master* mit ACK, damit die

Übertragung fortgesetzt werden kann und die SUBADRESS inkrementiert werden kann. Der *Master* stoppt die Übertragung mit *not*-ACK (die SDA-Leitung wird in der neunten SCL-Periode auf 1 gesetzt). Wenn die maximale SUBADRESS (hier 249) erreicht wird, sendet der *Slave* immer das Byte dieser Adresse, bis der *Master* die Übertragung stoppt.

- Am Ende der Übertragung initiiert der *Master* eine Stoppbedingung, indem er die SDA-Leitung von 0 auf 1 setzt, während die SCL-Leitung auf 1 bleibt. Die Übertragung ist erfolgreich abgeschlossen und der *Slave* geht in den *Idle*-Zustand, bis eine neue Startbedingung initiiert wird.
- Außer für die Start- und die Stoppbedingung, dürfen keine Daten in der SDA-Leitung übertragen werden, wenn die SCL-Leitung auf 1 gesetzt ist. Sollte so eine Situation auftreten, geht das gesamte System in den *Idle*-Zustand und der *Master* muss eine neue Startbedingung initiieren.

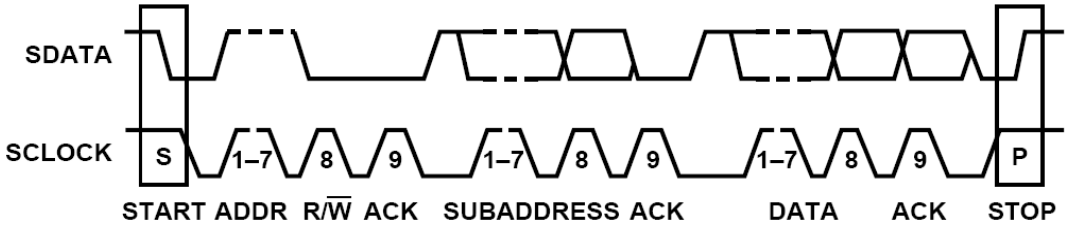

Abbildung 3.2: Datenübertragung beim IIC-Businterface [ANALOG-DEVICES 2005].

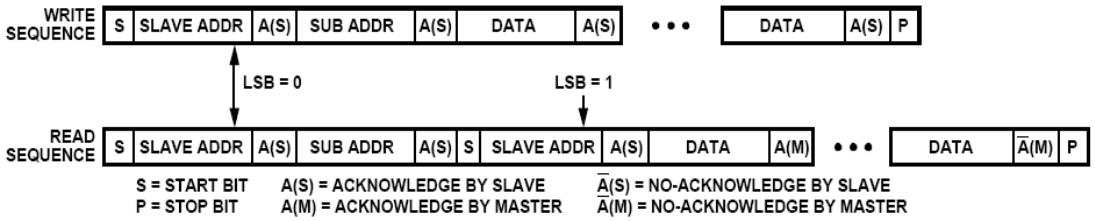

Abbildung 3.3: Lesen- bzw. Schreibpacket in einer IIC-Buskommunikation [ANALOG-DEVICES 2005].

Da einige Parameter des ADV7183B mehr als 8 Bit breit sind und somit zwei nacheinander folgenden IIC-Busadressen (SUBADRESS) besitzen, muss beim Schreiben mehr als eine IIC-Bus-Schreiboperation durchgeführt werden, um den gesamten Parameter zu ändern. Dafür ist ein IIC-*Sequencer* bei dem IIC-Businterface des ADV7183B Controllers implementiert. Dieser *Sequencer* sorgt dafür, dass der Parameterwert im Register nur übernommen wird, wenn die gesamten Schreiboperationen abgeschlossen sind, indem er die Zwischenwerte in einem Puffer speichert, wobei es von niedrigeren zu höheren Adressen geschrieben werden sollte, ohne dass eine nicht zu dem Parameter gehörende Adresse dazwischen geschrieben wird [Vgl. ANALOG-DEVICES 2005].
Da nicht alle Register des Controllers beschrieben sind, und um den Aufwand zu reduzieren, stehen Tabellen der zusendenden Kommandos für verschiedene Eingangsvideosignale in dem Datenblatt des Controllers zur Verfügung und müssen genau verfolgt werden, um eine korrekte Parametrierung sicherzustellen. Deshalb werden im Rahmen dieser Arbeit nur zwei Register beschrieben [Vgl. Tab. 3.2]. Für die analoge PAL-Kamera mit dem S-Video-Ausgang, was der Fall in dieser Arbeit ist, sieht die Tabelle für eine 27 MHz Pixelclock wie folgt aus [Vgl. ANALOG-DEVICES 2005, sowie Tab. 3.1].

Tabelle 3.1 Empfohlene Parametrierung für S-Videosignal mit 27 MHz Pixelclock [ANALOG-DEVICES 2005].

Nummer	Registeradresse	Registerwert	Beschreibung
1	0x00	0x06	Y1 = AIN1, C1 = AIN4.
2	0x15	0x00	Slow down digital clamps.
3	0x3A	0x12	Power down ADC 2.
4	0x50	0x04	Set DNR threshold to 4 for flat response.
7	0x0E	0x80	ADI recommended programming sequence. This sequence must be followed exactly when setting up the decoder.
8	0x50	0x20	Recommended setting.
9	0x52	0x18	Recommended setting.
10	0x58	0xED	Recommended setting.
11	0x77	0xC5	Recommended setting.
12	0x7C	0x93	Recommended setting.
13	0x7D	0x00	Recommended setting.
14	0xD0	0x48	Recommended setting.
15	0xD5	0xA0	Recommended setting.
16	0xD7	0xEA	Recommended setting.
17	0xE4	0x3E	Recommended setting.
18	0xE9	0x3E	Recommended setting.
19	0xEA	0x0F	Recommended setting.
20	0x0E	0x00	Recommended setting.

Da das VDEC1-Board nur eine 13.5 MHz Pixelclock liefert, müssen noch folgende Kommandos zu der oberen Tabelle hinzugefügt werden [Vgl. Digilent 2005b, ANALOG-DEVICES 2005, sowie Tab. 3.2].

Tabelle 3.2 Zusätzliche Parametrierungen für S-Videosignal mit 13.5 MHz Pixelclock [ANALOG-DEVICES 2005].

Nummer	Registeradresse	Registerwert	Beschreibung
5	0x03	0x09	16-bit @ LLC1 4:2:2., AV codes duplicated (for16-bit interfaces)
6	0x8F	0x50	LLC2 (nominally 13.5 MHz) selected out on LLC1 pin

3.2 Implementierung

In diesem Abschnitt werden die Implementierungen der Module zur Parametrierung des ADV7183B-Controllers sowie deren Zusammenspiel dargestellt und zwar das Reset-Modul und das IIC-Bus-Modul.

3.2.1 IP reset_vdec

Dieser IP-Core ist für die Initialisierung des ADV7183B-Controllers zuständig. Er besteht aus einer *TopEntity* namens *reset_vdec*, die die Schnittstelle des IPs darstellt und einer

user_logic zur Implementierung der Funktionalitäten dieses IPs [Vgl. Anhang 9.1]. Wie ein IP mit XPS erzeugt und benutzt werden kann ist in der Dokumentation zu finden [Vgl. Schwarz 2008]. Der Grund der Implementierung dieses Moduls in Hardware ist, dass während der Resetsequenz keine IIC-Buskommunikation möglich ist [Vgl. ANALOG-DEVICES 2005, sowie Abschnitt 3.1.1.].

3.2.1.1 TopEntity reset_vdec

Damit die Ein- und Ausgänge des IPs für andere Komponenten des System sichtbar sind, erzeugt XPS bei der Generierung des IPs die *TopEntity*. In diesem VHDL-Modul wird die *user_logic* instanziiert. Während der Generierung werden drei Softwareregister gewählt, die zur Kommunikation mit dem Softwaremodul über den PLB-Bus dienen.

3.2.1.2 user_logic

Die *user_logic* enthält die eigentliche Implementierung der Funktionalitäten dieses IPs. Mit der Generierung des IPs, ist der VHDL-Modul zum Zugriff auf den Registern mitgeliefert. Es muss nur die Resetsequenz implementiert werden.
basierend auf einem getakteten Prozess verfügt das Resetsequenz-Modul über folgende *low-activ* Signale:

- *rst_vdec* ist ein Ausgangssignal und wird verwendet, um das Resetpin des ADV7183B-Controllers zu betreiben.
- *pwdn_vdec* wird als Ausgangssignal verwendet, um den ADV7183B-Controller im Power-Down- bzw. Betriebsmodus zu setzen. Dafür wird er an dem *not-PWRDN*-Pin des Controllers gebunden.
- *oe_en_vdec* besagt, ob die Videodaten am Ausgang des ADV7183B-Controllers gültig sind oder nicht. Dieses Signal wird als Ausgang in diesem Modul mit dem *not-OE*-Pin des Controllers verbunden.

Der Ablauf dieses Moduls ist wie folgt beschrieben [Vgl. Abb.3.3, Listing. 3.3 sowie Abschnitt 3.1.1 und Abschnitt 3.2.2.2]:

- Der Zähler *cnt_1_s* fängt an zu zählen, wenn das Softwaremodul eine Resetsequenz angefordert hat, dabei wird auch mitgeteilt, wie lange diese Sequenz dauern muss. Wenn *cnt_1_s* seinen maximalen Wert noch nicht erreicht hat, werden die Signale *cnt_2_en_s*, *pwdn_vdec*, *oe_en_vdec* sowie *rst_vdec* auf 0 gesetzt, während *oe_en_vdec* auf 1 gesetzt wird, sonst werden die Signale *cnt_2_en_s* sowie *rst_vdec* auf 1 gesetzt, während *pwdn_vdec* auf *0* bleibt.
- Wenn *cnt_2_en_s* auf 1 gesetzt ist, startet der Zähler *cnt_2_s*. *cnt_1_s* wird gestoppt, indem *cnt_1_en_s* auf 0 gesetzt wird. Wenn *cnt_1_s* seinen maximalen Wert erreicht hat, werden die Signale *cnt_2_en_s*, *oe_en_vdec* sowie *rst_vdec* auf 0 gesetzt, während *pwdn_vdec* auf 1 gesetzt wird. Das Softwaremodul wird darüber informiert, informiert die Parametrierung zu starten.
- Wenn die Software das Hardwaremodul über das Ende der Parametrierung informiert, werden die Signale *oe_en_vdec* sowie *pwdn_vdec* auf 0 gesetzt. *rst_vdec* bleibt dabei auf 0 und gültige Videodaten stehen am Ausgang des VDEC1-Boards zu Verfügung.

Folgende Funktionalitäten werden von diesem Modul weiterhin erfüllt:

- Das *oe_en_vdec* Signal wird als Startbedingung bei den SW-Modulen zur Verarbeitung der Videosignale aus dem VDEC-Board benutzt, so dass sie nur starten,

wenn die Parametrierung erfolgreich abgeschlossen ist. Natürlich sind noch andere Bedingungen zu erfüllen, was in dem folgenden Kapitel behandelt wird.

- Das *pwdn_vdec* Signal setzt der ADV7183B-Controller während der Parametrierung schon im Betriebsmodus. Der Grund dafür ist, dass während der Implementierung nach mehreren Proben festgestellt wird, dass die IIC-Buskommunikation nur erfolgreich abgeschlossen ist, wenn der Controller im Betriebsmodus ist.

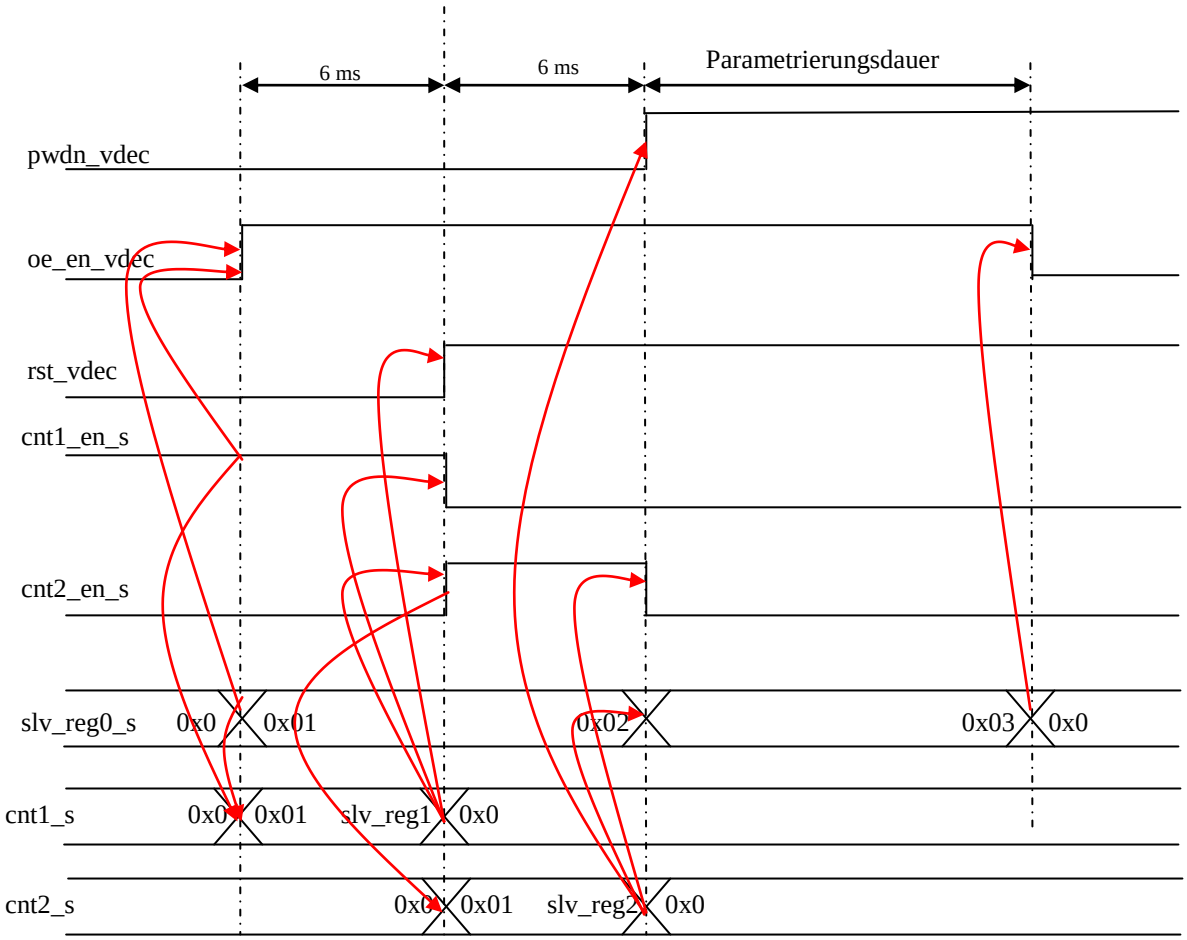

Abbildung 3.4: Timingsdiagramm der Resetsequenz

3.2.2 **Parametrierungsmodul *IIC-Schnittstelle***

In diesem Modul wird die eigentliche Parametrierung des ADV7183B-Controllers implementiert. EDK stellt der IP-Core *XPS-IIC* zur Verfügung, so dass die Parametrierung per Software erfolgt. Die IIC-Buskommunikation wird also durch Funktionsaufrufe durchgeführt.

3.2.2.1 IP-Core XPS-IIC

Das XPS-IIC-Modul ist ein vom Xilinx freigegebener IP-Core, der ein parametrierbares Interface für die IIC-Buskommunikation anbietet, wobei alle Modi außer dem *high speed*-

Modus unterstützt werden [Vgl. Xilinx 2008c]. Folgende Features stellt dieser Core zur Verfügung:

- *Master* oder *Slave*-Operation sowie *Multi-Master*-Operation.
- Generierung und Erkennung der *Start- bzw. Stop*-Bedingung sowie des *Acknowledge*-Bit.
- 7 oder 10 Bit Adressierung sowie 16 Bit Sende- bzw. Empfangspuffer.
- Unterstützung 100 KHz (*standard mode*) sowie 400 KHz (*fast mode*) Übertragungsgeschwindigkeit.
- Filterung der SCL bzw. SDA-Leitung sowie Implementierung eines *Arbiters*.

Auf dem FPGA ist der Ressourcenverbrauch wie in der folgenden Tabelle beschrieben [Vgl. Xilinx 2008c sowie Tab. 3.3].

Tabelle 3.3: Ressourcenverbrauch des XPS-IIC-Moduls auf dem FPGA.

	Minimal	Maximal
Slice	213	262
LUTs	391	405
FFs	343	353
Block RAMs	0	0

Bei der Implementierung dieses Moduls sind die SDA- und die SCL-Leitung als IO bereit gestellt. Diese beiden Leitungen werden in dieser Arbeit an dem SCL- und SDA-Pin des VDEC1-Boards gebunden. Dieses Modul stellt 32 Bit Register zur Verfügung, die über ihre Adresse angesprochen werden können. Diese Register werden benutzt, um die IIC-Buskommunikation durchzuführen. Die für diese Arbeit relevanten Register sind in der folgenden Tabelle beschrieben [Vgl. Xilinx 2008c sowie Tab. 3.4].

Tabelle 3.4: Register des XPS-IIC-Moduls.

Register	Offset	Beschreibung
Global Interrupt Enable (GIE)	0x01C	Es wird benutzt, um den Interrupt-Modus des XPS-IIC-Moduls zu aktivieren oder zu deaktivieren.
Interrupt Status Register (ISR)	0x020	Es besagt welches Interrupt beim XPS-IIC-Modul aufgetreten ist.
Interrupt Enable Register (IER)	0x028	Es wird benutzt, um jede Interruptsquelle des XPS-IIC-Moduls zu aktivieren oder zu deaktivieren.
SW-Reset Register (SOFTR)	0x040	Er wird benutzt, um das XPS-IIC-Modul zu initialisieren.
Control Register (CR)	0x100	Es wird benutzt, um das Übertragungsmodus wie zum Beispiel Master-Sender bzw. Empfänger des XPS-IIC-Moduls zu konfigurieren
Status Register (SR)	0x104	Es wird benutzt, um den Zustand des XPS-IIC-Moduls abzufragen.
Transmit FIFO (Tx_FIFO)	0x108	Es wird für die Sendeoperation benutzt und enthält die zu sendenden Daten.
Receive FIFO (Rc_FIFO)	0x10C	Es wird für die Empfangsoperation benutzt und enthält die zu empfangenen Daten.

Für die Nutzung dieser Register stellt Xilinx C-Funktionen zur Verfügung, wobei die für diese Arbeit relevanten Dateien die *xiic_l.h* und die *xiic_l.c* sind [Vgl. Anhang 9.1].

In der *xiic_l.h*-Datei sind die Low-Level Funktionen zum Zugriff auf den Registern des XPS-IIC-Moduls, die die IIC-Buskommunikation implementieren. Diese Funktionen sind als Makro implementiert, so dass deren Aufruf nicht viel CPU-Zeit kostet [Vgl. Listing 3.1].

Listing 3.1: Implementierung der Funktion *XIic_mDynSend7BitAddress* in der *xiic_l.h*-Datei als Makro

```
482 * This macro sends the address for a 7 bit address during both read and write
483 * operations. It takes care of the details to format the address correctly.
484 * This macro is designed to be called internally to the drivers.
485 *
486 * @param   BaseAddress contains the base address of the IIC Device.
487 * @param   SlaveAddress contains the address of the slave to send to.
488 * @param   Operation indicates XIIC_READ_OPERATION or XIIC_WRITE_OPERATION.
489 *
490 * @return  None.
491 *
492 * @note            C-Style signature:
493 *          void XIic_mDynSend7BitAddress(u32 BaseAddress, u8 SlaveAddress,
494 *                                        u8 Operation);
495 *
496 ****************************************************************************/
497 #define XIic_mDynSend7BitAddress(BaseAddress, SlaveAddress, Operation)   \
498 {                                                                        \
499     u8 LocalAddr = (u8)(SlaveAddress << 1);                              \
500     LocalAddr = (LocalAddr & 0xFE) | (Operation);                        \
501     XIo_Out16(BaseAddress + XIIC_DTR_REG_OFFSET - 1,                     \
502         XIIC_TX_DYN_START_MASK | LocalAddr);                             \
503 }
```

Die *xiic_l.c*-Datei fasst mehrere Funktionen aus der *xiic_l.h*-Datei zusammen, um eine komplette IIC-Buskommunikation zu implementieren, sodass für die Sendeoperation zum Beispiel die Funktion *XIic_Send()* aufgerufen wird. Diese Funktion wird dann die Sendeoperation gemäß dem IIC-Busprotokoll durchführen, indem sie die Register des XPS-IIC-Moduls entsprechend setzt beziehungsweise löscht sowie deren Zustände abfragt [Vgl. Listing 3.2].

Listing 3.2: Implementierung der *XIic_Send* Funktion in der *xiic_l.c*-Datei als Aufruf von Funktionen aus der *xiic_l.h*-Datei

```
391 *****************************************************************************/
392 unsigned XIic_Send(u32 BaseAddress, u8 Address,
393                    u8 *BufferPtr, unsigned ByteCount, u8 Option)
394 {
395     unsigned RemainingByteCount;
396     u8 ControlReg;
397     volatile u8 StatusReg;
398     u32 cr = 0;
399   u32 sr = 0;
400   u32 isr = 0;
401
402     /* Check to see if already Master on the Bus.
403      * If Repeated Start bit is not set send Start bit by setting MSMS bit else
404      * Send the address.
405      */
406     ControlReg = XIo_In8(BaseAddress + XIIC_CR_REG_OFFSET);
407     if ((ControlReg & XIIC_CR_REPEATED_START_MASK) == 0) {
408         /* Put the address into the FIFO to be sent and indicate that the operation
409          * to be performed on the bus is a write operation
410          */
411         XIic_mSend7BitAddress(BaseAddress, Address,
412                               XIIC_WRITE_OPERATION);
413         /* Clear the latched interrupt status so that it will be updated with the
414          * new state when it changes, this must be done after the address is put
415          * in the FIFO
416          */
417         XIic_mClearIisr(BaseAddress, XIIC_INTR_TX_EMPTY_MASK |
418                 XIIC_INTR_TX_ERROR_MASK |
419                 XIIC_INTR_ARB_LOST_MASK);
420
421         /* MSMS must be set after putting data into transmit FIFO, indicate the
422          * direction is transmit, this device is master and enable the IIC device
423          */
424         XIo_Out8(BaseAddress + XIIC_CR_REG_OFFSET,
425                 XIIC_CR_MSMS_MASK | XIIC_CR_DIR_IS_TX_MASK |
426                 XIIC_CR_ENABLE_DEVICE_MASK);
427
428         /* Clear the latched interrupt
429          * status for the bus not busy bit which must be done while the bus is busy
430          */
431
432         StatusReg = XIo_In8(BaseAddress + XIIC_SR_REG_OFFSET);
433         xil_printf("\r\n StatusReg: 0X%x\r\n", StatusReg);
434         while ((StatusReg & XIIC_SR_BUS_BUSY_MASK) == 0) {
435
436             StatusReg = XIo_In8(BaseAddress + XIIC_SR_REG_OFFSET);
437         }
438         xil_printf("2\r\n");
439
440         XIic_mClearIisr(BaseAddress, XIIC_INTR_BNB_MASK);
441
442     }
443     else {
444         /* Already owns the Bus indicating that its a Repeated Start call.
445          * 7 bit slave address, send the address for a write operation
446          * and set the state to indicate the address has been sent
447          */
448          xil_printf("3\r\n");
449         XIic_mSend7BitAddress(BaseAddress, Address,
450                               XIIC_WRITE_OPERATION);
451     }
452
453     /* Send the specified data to the device on the IIC bus specified by the
454      * the address
455      */
466     RemainingByteCount =
467         SendData(BaseAddress, BufferPtr, ByteCount, Option);
468     xil_printf("after send\r\n");
469     ControlReg = XIo_In8(BaseAddress + XIIC_CR_REG_OFFSET);
470     if ((ControlReg & XIIC_CR_REPEATED_START_MASK) == 0) {
471         /* The Transmission is completed, disable the IIC device if the Option
472          * is to release the Bus after transmission of data and return the number
473          * of bytes that was received. Only wait if master, if addressed as slave
474          * just reset to release the bus.
475          */
476         if ((ControlReg & XIIC_CR_MSMS_MASK) != 0) {
477             XIo_Out8(BaseAddress + XIIC_CR_REG_OFFSET,
478                     (ControlReg & ~XIIC_CR_MSMS_MASK));
479             StatusReg = XIo_In8(BaseAddress + XIIC_SR_REG_OFFSET);
480             xil_printf(" before the end of send\r\n");
481             while ((StatusReg & XIIC_SR_BUS_BUSY_MASK) != 0) {
482                 StatusReg =
483                     XIo_In8(BaseAddress +
484                         XIIC_SR_REG_OFFSET);
485             }
486         }
487
488         XIo_Out8(BaseAddress + XIIC_CR_REG_OFFSET, 0);
489     }
490     xil_printf("the end of send\r\n");
491     return ByteCount - RemainingByteCount;
492 }
```

3.2.2.2 Parametrierung des ADV7183B mit dem IP-Core XPS-IIC

Für die Parametrierung des ADV7183B-Controllers, wird zuerst das XPS-IIC-Modul generiert, parametrisiert und zu dem Projekt hinzugefügt. In der von EDK generierten *xparameters.h-Datei* ist der Parameter *XPAR_IIC_0_BASEADDR* zu finden, welcher der Basisadresse des XPS-IIC-Moduls im Adressraum des gesamten Systems entspricht. Über diese Adresse wird das XPS-IIC-Modul angesprochen.
Die Parametrierung wird gestartet, in dem eine Resetsequenz angefordert wird. Da das Modul zur Generierung der Resetsequenz in Hardware implementiert wird, stellt dieses Modul drei Softwareregister (SW-Register) zur Kommunikation zur Verfügung. Das *Hand-Shake* zwischen dem HW-Modul zur Generierung der Resetsequenz (*reset_vdec*) und dem SW-Modul (*IIC-Schnittstelle*) zur Parametrierung des ADV7183B-Controllers sieht wie folgt aus [Vgl. Listing 3.3 sowie Abb. 3.3]:

- Das SW-Modul *IIC-Schnittstelle* schreibt den Wert 0X493E0 in dem dritten SW-Register des HW-Moduls *reset_vdec*, damit das HW-Modul *reset_vdec* den Resetpin für 6 ms (0X493E0 / 50Mhz) auf 0 setzt, was die Dauer der Resetsequenz entspricht.
- Der Wert 0X493E0 wird auch von *IIC-Schnittstelle* in dem zweiten SW-Register von *reset_vdec* geschrieben, damit das HW-Modul das SW-Modul erst 6 ms (0X493E0 / 50Mhz), nachdem der Resetpin auf 1 gesetzt wurde, über das Ende der Resetsequenz informiert. So wird erfordert, dass die IIC-Buskommunikation frühesten 6 ms nach der Resetsequenz gestartet wird [Vgl. ANALOG-DEVICES 2005].
- Der Wert 0x01 wird von *IIC-Schnittstelle* in dem ersten SW-Register des HW-Moduls *reset_vdec* geschrieben und das HW-Modul startet die Resetsequenz. Während dieser Sequenz fragt das SW-Modul den Wert des ersten Registers des HW-Moduls ab, was in dieser Arbeit nicht kritisch ist, da der MicroBlaze Prozessor keine andere Task außer der Parametrierung des ADV7183B-Controllers durchführen muss. Wenn dieser Wert 2 ist, heißt es, dass die Resetsequenz erfolgreich abgeschlossen wurde. Die IIC-Buskommunikation kann gestartet werden.

Listing 3.3: SW-Modul zur Kommunikation mit dem HW-Modul zur Generierung der Resetsequenz

```
368 void reset_videodec(void){
369     Xuint32 done;
370     Xuint32 i;
371     Xuint32 timer1_top      = 0X493E0;//f = 50 Mhrz, T =  10 nanoS ====> top = 0X493E0 ==> 6 ms
372     Xuint32 timer2_top      = 0X493E0;//f = 50 Mhrz, T =  10 nanoS ====> top = 0X3D090 ==> 6 ms
373     Xuint32 timer1_start    = 0X1;
374     xil_printf("\r\nin reset 1 \r\n");
375     Xuint32 *reset_vdec_ptr = (Xuint32 *)XPAR_RESET_VDEC_0_BASEADDR;
376     i=2;
377     *(reset_vdec_ptr + i) = timer2_top;//top für timer 1 in SW Reg 1 schreiben
378     i=1;
379     *(reset_vdec_ptr + i) = timer1_top;//top für timer 2 in SW Reg 2 schreiben
380     *reset_vdec_ptr = timer1_start;    //start für timer 1 in SW Reg 0 schreiben
381     xil_printf("\r\nin reset 2 \r\n");
382     while ((done = *(reset_vdec_ptr)) != 2){xil_printf("\r\nin reset done: 0X%x\r\n",
383     done);}//reg 0 lesen und Warten bis done = 1 ===> Reset sequence wurde anbeschloßen
384     xil_printf("\r\nin reset 3 done: 0X%x\r\n", done);
385     return;
386 }
```

Wenn die Resetsequenz erfolgreich abgeschlossen wurde, wird die Parametrierung wie folgt von dem SW-Moduldurch *IIC-Schnittstelle* geführt [Vgl. Listing 3.4].

- Die Tabelle *decoder_svid* enthält die 20 Kommandos, die für die Parametrierung des ADV7183B-Controllers mit S-Video als Inputsignal und eine 13.5 MHz Pixelclock notwendig sind.
- Die Makro *XIIC_RESET()* initialisiert das IIC-Businterface, in dem der Wert 0x0A in SOFTR geschrieben wird [Vgl. Anhang 9.1].
- In einer For-Schleife sendet die Funktion *XIic_Send()* die Daten aus der Tabelle *decoder_svid* zeilenweise. Da immer die SUBADRESS und der Wert des Parameters gesendet werden (2 Bytes), liefert die Funktion XIic_Send() 2 zurück, wenn das Senden erfolgreich war.
- Wenn das Senden erfolgreich war, wird das HW-Modul *reset_vdec* über das Ende der Parametrierung dadurch informiert, dass das SW-Modul *IIC-Schnittstelle* den Wert 0x03 in dem ersten SW-Register dieses HW-Moduls schreibt. Das HW-Modul *reset_vdec* setzt den *OE*-Pin des VDEC1-Boards auf 0, damit die HW-Module zur Videosignalverarbeitung starten können [Vgl. Abb. 3.4].

Listing 3.4: SW-Modul zur Parametrierung des ADV7183B-Controllers für das S-Video Inputsignal und 13.5MHz Pixelclock

```
114  #define DECODER_SVID_CONFIG_CNT 20
115  //#define DECODER_SVID_CONFIG_CNT 19
116 struct VideoModule decoder_svid[] = {
117     { 0x00, 0x06, 0 },//Y on AIN1, C on AIN4.
118     { 0x15, 0x00, 0 },//Digital Clamp Control 1
119     { 0x3a, 0x12, 0 },//Power down ADC2.
120     { 0x50, 0x04, 0 },//CTI DNR Control 4
121     { 0x0e, 0x80, 0 },//ADI Control
122       //Selbst geändert
123     { 0x04, 0x44, 0 },//ITU R BT 656 - 3
124     { 0x03, 0x09, 0 },//16-bit @ LLC1 4:2:2.
125     //AV codes duplicated (for16-bit interfaces)
126     { 0x50, 0x20, 0 },
127     { 0x52, 0x18, 0 },//reserved
128     { 0x58, 0xed, 0 },//reserved
129     { 0x77, 0xc5, 0 },//reserved
130     { 0x7c, 0x93, 0 },//reserved
131     { 0x7d, 0x00, 0 },//reserved
132     { 0xd0, 0x48, 0 },//reserved
133     { 0xd5, 0xa0, 0 },//reserved
134     { 0xd7, 0xea, 0 },//reserved
135     { 0xe4, 0x3e, 0 },//SD Saturation Cr
136     { 0xE9, 0x3e, 0 },
137     { 0xea, 0x0f, 0 },
138     { 0x0e, 0x00, 0 } };
300   for( i = 0; i < config_cnt; i++ )
301    {
302     XIIC_RESET(XPAR_I2C_BASEADDR);
303     send_data[0] = decoder[i].addr;
304     send_data[1] = decoder[i].config_val;
305     send_cnt = XIic_Send(XPAR_I2C_BASEADDR, DECODER_ADDR, send_data, 2, XIIC_STOP);
306     if( send_cnt != 2 )
307      {
308       xil_printf("Error writing to address 0X%02x\r\n", decoder[i].addr);
309       success = 0;
310       break;
311      }else{
312         xil_printf("\r\n writing to address 0X%02x \t data: 0X%02x \r sucess\n",
313         send_data[0], send_data[1]);
314      }
315    }
316
317   if( success ){
318     print("SUCCESS!\r\n");
319     //HW kann jetzt die digitalisierten Daten lesen
320     *vdec_ptr = 0X03;
321     }
```

4 Hardwaremodule zur Pixelextrahierung und Darstellung

In diesem Kapitel werden die HW-Module zur Verarbeitung des Pixelstroms aus dem VDEC1-Board und Ausgabe auf dem TFT-Monitor dargestellt, wobei diese Kette HW-Module zuerst mit einem Testbildgenerator, der ähnliche Videosignale, wie diese aus dem VDEC1-Board generiert, getestet wird.

4.1 16 Bit ITU-R BT.656 YCbCr 4:2:2 *Interlaced*-Videosignal

Nachdem der ADV7183B-Controller korrekt parametriert wurde, liefert er am Ausgang des VDEC1-Boards 16 Bit ITU-R BT.656 YCbCr 4:2:2 *Interlaced*-Videosignale mit einer Auflösung von 720x576.
Entwickelt für die digitale PAL-Fernsehnorm beruht das YCbCr-Farbmodell auf der Fähigkeit des Auges, geringe Helligkeitsunterschiede besser zu erkennen als kleine Farbtonunterschiede, und diese wiederum besser als kleine Farbsättigungsunterschiede. Als Helligkeit-Farbigkeits-Modell steht Y für die Grundhelligkeit. Cb ist ein Maß für die Abweichung von Grau in Richtung Blau. Cr ist die entsprechende Maßzahl für Abweichung in Richtung Rot bzw. Türkis (Komplementärfarbe von Rot). Diese Darstellung nutzt die Eigenheit des Sehsinnes, für grünes Licht besonders empfindlich zu sein. Die Information über den Grünanteil ist in der Grundhelligkeit Y eingerechnet. Nur die farblichen Abweichungen beim Rot/Türkis- und Blau/Gelb-Anteil sind deshalb anzugeben [Vgl. Wikipedia 2009a, sowie Abb. 4.1].

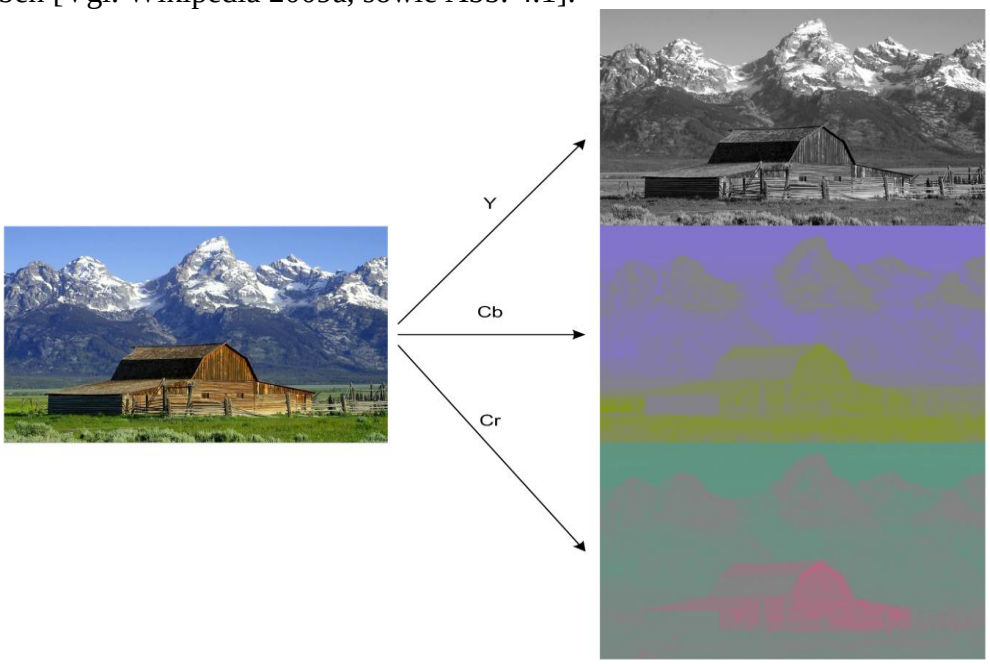

Abbildung 4.1 Originalbild (Mitte links) und seine Aufspaltung in die Komponenten Y(oben rechts), Cb (Mitte rechts) und Cr (unten rechts) [Wikipedia 2009a].

Das 16 Bit YCbCr 4:2:2 Format basiert auf dieser Fähigkeit des Auges, Helligkeitsunterschiede besser als Farbenunterschiede zu erkennen, um die Datenübertragung des YCbCr-Videosignals zu reduzieren. Dabei wird pro Pixel immer einen Y-Wert übertragen, während die Cb- und Cr-Werte abwechselnd übertragen

werden. Die fehlenden Cb- beziehungsweise Cr-Werte werden durch eine Pixelreplikation oder eine Interpolation bestimmt [Vgl. Jack 2005, Poynton 2005 sowie Tab. 4.1].

SAMPLE 0	SAMPLE 1	SAMPLE 2	SAMPLE 3	SAMPLE 4	SAMPLE 5	
Y7 - 0 Y6 - 0 Y5 - 0 Y4 - 0 Y3 - 0 Y2 - 0 Y1 - 0 Y0 - 0	Y7 - 1 Y6 - 1 Y5 - 1 Y4 - 1 Y3 - 1 Y2 - 1 Y1 - 1 Y0 - 1	Y7 - 2 Y6 - 2 Y5 - 2 Y4 - 2 Y3 - 2 Y2 - 2 Y1 - 2 Y0 - 2	Y7 - 3 Y6 - 3 Y5 - 3 Y4 - 3 Y3 - 3 Y2 - 3 Y1 - 3 Y0 - 3	Y7 - 4 Y6 - 4 Y5 - 4 Y4 - 4 Y3 - 4 Y2 - 4 Y1 - 4 Y0 - 4	Y7 - 5 Y6 - 5 Y5 - 5 Y4 - 5 Y3 - 5 Y2 - 5 Y1 - 5 Y0 - 5	16 BITS PER SAMPLE
CB7 - 0 CB6 - 0 CB5 - 0 CB4 - 0 CB3 - 0 CB2 - 0 CB1 - 0 CB0 - 0	CR7 - 0 CR6 - 0 CR5 - 0 CR4 - 0 CR3 - 0 CR2 - 0 CR1 - 0 CR0 - 0	CB7 - 2 CB6 - 2 CB5 - 2 CB4 - 2 CB3 - 2 CB2 - 2 CB1 - 2 CB0 - 2	CR7 - 2 CR6 - 2 CR5 - 2 CR4 - 2 CR3 - 2 CR2 - 2 CR1 - 2 CR0 - 2	CB7 - 4 CB6 - 4 CB5 - 4 CB4 - 4 CB3 - 4 CB2 - 4 CB1 - 4 CB0 - 4	CR7 - 4 CR6 - 4 CR5 - 4 CR4 - 4 CR3 - 4 CR2 - 4 CR1 - 4 CR0 - 4	

- 0 = SAMPLE 0 DATA
- 1 = SAMPLE 1 DATA
- 2 = SAMPLE 2 DATA
- 3 = SAMPLE 3 DATA
- 4 = SAMPLE 4 DATA

Tabelle 4.1: 16 Bit YCbCr 4:2:2 Format [Jack 2005].

Das ITU-R BT.656 Format beschreibt den Inhalt des Pixelstrom, sowie die Synchronisationssignale, die mit den Synchronisationspulsen *Horizontal Synchronization* (HS) und *Vertical Synchronization* (VS) geliefert sind. Der Pixelstrom besteht aus dem eigentlichen Videosignal und den Synchronisationssignalen, die benutzt werden können, um FVAL und LVAL zu generieren [Vgl. ANALOG-DEVICES 2005 sowie Abb. 4.2 und Abb. 4.3].

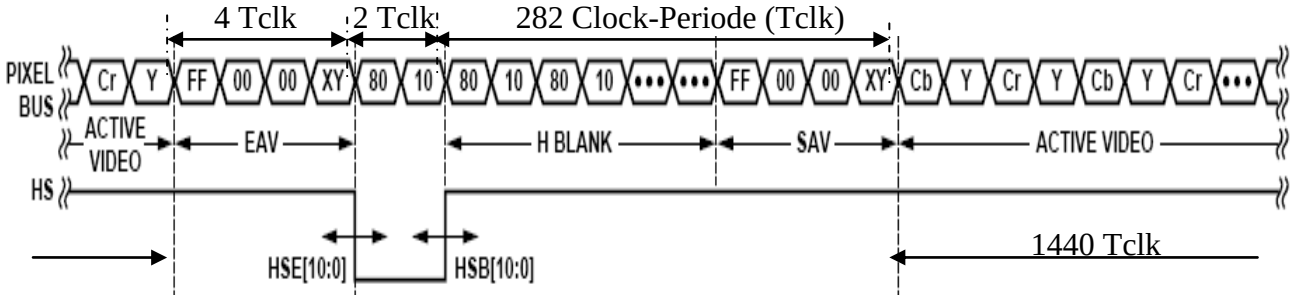

Abbildung 4.2: Pixelstrom sowie HS für das 8 Bit ITU-R BT.656 YCbCr 4:2:2 Format mit einer 27 MHz Pixelclock [ANALOG-DEVICES 2005].

Da in dieser Arbeit ein 16 Bit ITU-R BT.656 YCbCr 4:2:2 Videosignal mit einer 13.5 MHz Clock vorliegt, sehen die in der Abbildung 4.2 beschriebenen Werte wie folgt aus [Vgl. Tab. 4.2].

Tabelle 4.2: Dauer aller Signale aus dem Pixelstrom sowie HS für das 16 Bit ITU-R BT.656 YCbCr 4:2:2 Format mit einer 13.5 MHz Pixelclock.

Signale	**Dauer in Tclk**	**Beschreibung**
EAV-Signal	2	Es signalisiert das Ende einer Pixelzeile
HS-Puls	1	Er signalisiert den Anfang einer neuen Pixelzeile
HBLANK- und SAV-Signal	141	Sie beschreiben das Blanking und den Anfang von gültigen Pixelwerten
Aktive Video	720	Gültige Pixelwerte einer Pixelzeile
Dauer einer Pixelzeile	864	Gesamte Dauer einer Pixelzeile

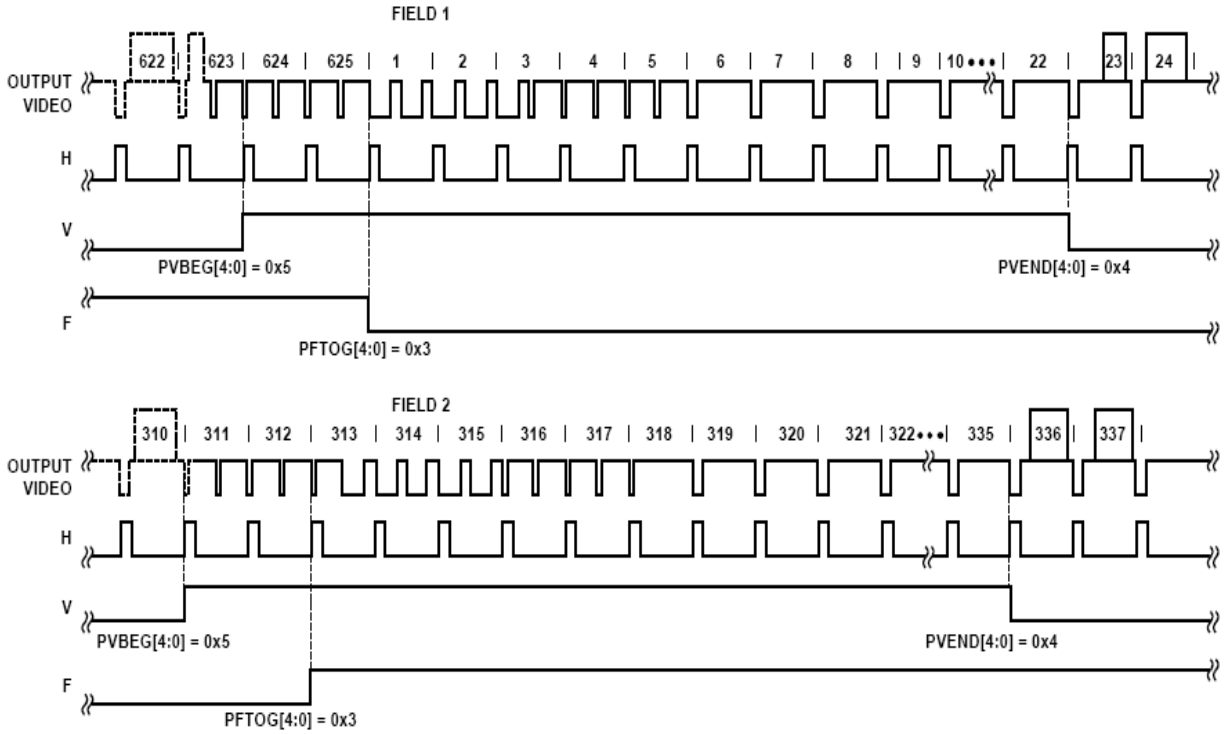

Abbildung 4.3: Synchronisationspulse H (HS), V (VS), F (FIELD) sowie der Pixelstrom aus dem VDEC1-Board im ITU-R BT.656 *Interlaced*-Format [ANALOG-DEVICES 2005].

4.2 Testbildgenerator

Zu Simulationszwecken erzeugt dieses HW-Modul ein 16 Bit ITU-R BT.656 YCbCr 4:2:2 *Interlaced*-Videosignal sowie die Synchronisationspulse HS und VS gemäß der Beschreibung im Abschnitt 4.1 [Vgl. Anhang 9.1].

- Für die Erzeugung einer Pixelzeile wird der Counter *HPOS* benutzt [Vgl. Abb. 4.2, sowie Tab. 4.2]. Dieser Counter zählt von 0 bis 864(Dauer einer Pixelzeile). Zwischen 0 und 139 werden die Pixelwerte auf schwarz gesetzt (H Blank), wobei zwischen 0 und 1 der HS-Puls generiert wird. Zwischen 140 und 141 wird das *Start of Active Video* (SAV)-Signal gemäß dem ITU-R BT.656 Format generiert. Von 142 bis 861 werden Y, Cb und Cr so gesetzt werden, dass am Ende rote, grüne, blaue, schwarze und weiße Farbstreifen mit einer Intensität von 75% auf dem Bildschirm dargestellt werden [Vgl.

Jack 2005, S. 18 sowie Abb. 4.4]. Zwischen 862 und 863 wird das *End of Active Video* (EAV)-Signal erzeugt. Bei 864 wird dieser Counter auf 0 zurückgesetzt und der Counter VPOS wird inkrementiert.

- Der Counter VPOS zählt von 0 bis 624 und erzeugt somit die für ein Bild benötigten Pixelzeilen [Vgl. Abb. 4.3]. Zwischen 0 und 21, 310 und 334, sowie 623 und 624 wird der Synchronisationspuls VS generiert und die Pixelwerte werden auf schwarz gesetzt. Dabei werden auch wie in dem oberen Abschnitt beschrieben die SAV- und EAV-Signale genegiert und in den Pixelstrom eingebettet. Bei 624 wird dieser Counter auf 0 zurückgesetzt. Zwischen 22 und 309 sowie 335 und 622 werden die Pixelzeilen wie im oberen Abschnitt beschrieben generiert.

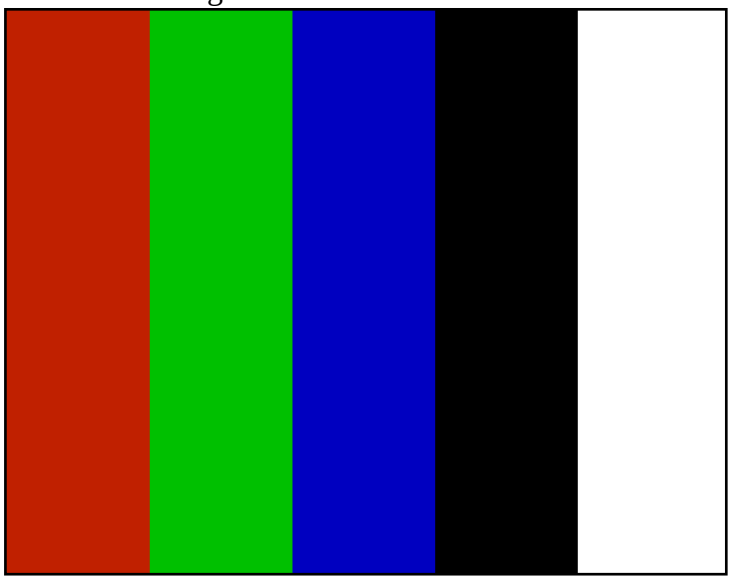

Abbildung 4.4: Farbstreifen aus dem Testbildgenerator

4.3 Erzeugung der Synchronisationssignale Frame Valid (FVAL) und Line Valid (LVAL)

Die Synchronisationssignale FVAL und LVAL besagen, wann die Pixelwerte gültig sind oder nicht. Zur Erzeugung dieser beiden Signale werden die Synchronisationspulse HS und VS benutzt. Da nach Messung der HS- und VS-Pulse aus dem VDEC1-Board festgestellt wurde, dass die beiden Pulse nicht immer synchron zueinander auftreten [Vgl. Abb. 4.5], was auf der Abbildung 4.4 ohne weitere Hinweise durch den Knick bei den Zeilen 622, 623, 23, 24, 310, 311, 336 sowie 337 beschrieben ist, wird das HW-Modul *Sync* zur Erzeugung von FVAL und LVAL zweimal implementiert und zwar für den Fall, dass der Testbildgenerator als Pixelquelle benutzt wird und den Fall, dass das VDEC1-Board als Pixelquelle benutzt wird.

4.3.1 *Sync*-Modul mit HS und VS aus dem Testbildgenerator

Die Pulse HS und VS aus dem Testbildgenerator haben die Eigenschaft, dass sie immer Synchron zu einander verändert werden. Der Anfang der ersten Pixelzeile eines Bildes

wird daran erkannt, dass VS auf 1 und gleichzeitig HS auf 0 gesetzt werden. Das Ende der letzten Pixelzeile dieses Bildes tritt dann auf, wenn VS auf 0 und gleichzeitig HS auf 0 gesetzt werden. Diese Eigenschaft wird von einer *Final State Machine* (FSM) benutzt, um mit zwei Counter LVAL und FVAL wie folgt beschrieben zu erzeugen, wobei VS aus dem VDEC1-Board hier invertiert wird [Vgl. Abb. 4.6]. Wenn in einem Zustand keine Zuweisung für die getakteten Signale *H_CNT*, *H_CNT1*, *Y_OUT*, *CBCR_OUT*, *LVAL*, *FVAL*, *F_FRAME* oder *NEXT_STATE* erfolgt, werden sie durch ihre *Default*-Werte ersetzt wie in dem Listing 4.4 beschrieben, wobei die Änderung *„_S“* für die Signale stehen, die getaktet sein werden [Vgl. Listing 4.4].

- Der Startzustand der FSM ist *IDLE*: in diesem Zustand bleibt die FSM bis VS auf 0 gesetzt wird. Dabei wird das Signal *F_FRAME* auf 1 gesetzt, was die Bedeutung hat, dass die erste Pixelzeile des neuen Frames noch nicht eingetroffen ist. Wenn VS auf 0 gesetzt wird, wird das Ende des letzen Frames beziehungsweise den Anfang eines neuen Frames signalisiert. Die FSM geht in den Zustand *W_FRAME*.

- Im Zustand *W_FRAME* bleibt die FSM bis VS auf 1 geht, was den Anfang eines neuen Frames signalisiert. Dabei bleibt das Signal *F_FRAME* weiterhin auf 1. Wenn VS und HS auf 1 gehen, wird der Counter *H_CNT* hochgezählt und die FSM geht in den Zustand *LINE_SAV*. Wenn aber HS auf 0 bleibt während VS auf 1 gesetzt wird, wird der Counter *H_CNT* hochgezählt. Die FSM geht in den Zustand *W_LINE*.

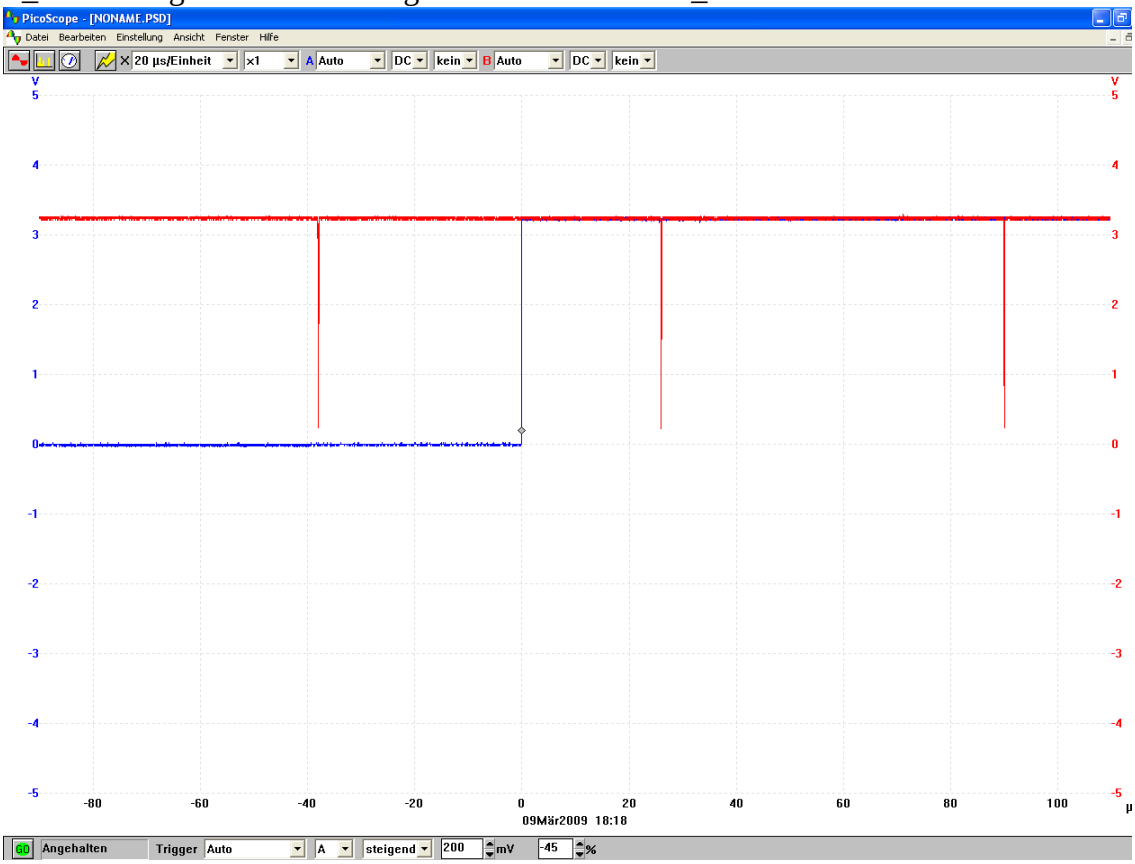

Abbildung 4.5: gemessene Pulse HS (rot) und VS (blau) aus dem VDEC1-Board.

- In dem Zustand *W_LINE* heißt es, dass der Anfang eines neuen Frames schon eingetroffen ist und der Synchronisationspuls HS für die erste Zeile gerade startet. Der Counter *H_CNT* wird hochgezählt. Es wird hier gewartet bis HS auf 1 gesetzt wird und die FSM geht in den Zustand *LINE_SAV*. Wenn das Signal *F_FRAME* auf 0 gesetzt ist, heißt es, dass für dieses Frame schon mehr als eine Pixelzeile angetroffen sind. FVAL

wird deshalb weiterhin auf 1 gesetzt. Wenn aber *F_FRAME* auf 1 gesetzt wird, geht es um die erste Pixelzeile für dieses Frame. FVAL wird dann erst auf 1 gesetzt, wenn die für die erste Zeile gültigen Pixelwerte in dem Pixelstrom vorliegen. Wenn es in diesem Zustand vorkäme, dass VS 0 gesetzt ist also, dass das Frame nicht gültig ist, wird der Counter *H_CNT1* benutzt, um das Synchronisationssignal FVAL noch für 142 Tclk auf 1 zu setzen, danach geht die FSM in den Zustand *W_FRAME*.

- Der Zustand *LINE_SAV* besagt, dass der Synchronisationspuls für eine Pixelzeile HS schon eingetroffen ist. Hier wird gemäß dem ITU-R BT.656 Format 141 Tclk gewartet, bis die in dem Pixelstrom eingebetteten Synchronisationssignale HBLANK und SAV eintreffen [Vgl. Abb. 4.2, sowie Tab. 4.2]. Wie im Zustand *W_LINE*, hat hier das Signal *F_FRAME* dieselbe Bedeutung. Deshalb wird FVAL genau so wie in dem Zustand *W_LINE* behandelt für den Fall, dass *F_FRAME* auf 0 oder 1 gesetzt wird. Der Counter H_CNT wird bis 142 hochgezählt, was den Anfang von gültigen Daten im Pixelstrom signalisiert. FVAL und LVAL werden auf 1 gesetzt, die zwischengespeicherten Y- und Cb/Cr-Werte aus dem VDEC1-Board werden als Ausgang verfügbar gestellt und die FSM geht in den Zustand *LINE_A_VIDEO_EAV*. Wenn es ist in diesem Zustand auch vorkäme, dass VS auf 0 gesetzt wird, geht die FSM in den Zustand *W_FRAME*.
- Der Zustand *LINE_A_VIDEO_EAV* behandelt eine Pixelzeile mit gültigen Werten und mit dem eingebetteten Synchronisationssignal EAV. In diesem Zustand ist logischerweise VS auf 1 gesetzt. *H_CNT* wird hochinkrementiert und die FSM bleibt in diesem Zustand, bis *H_CNT* größer als 863 ist. Wenn *H_CNT* zwischen 142 und 861 liegt (720 Tclk), geht es um die gültigen Pixelwerte einer Zeile. FVAL und LVAL werden auf 1 gesetzt, die zwischengespeicherten Y- und Cb/Cr-Werte aus dem VDEC1-Board werden als Ausgang zur Verfügung gestellt. Wenn aber *H_CNT* zwischen 862 und 863 liegt (2 Tclk), geht es um das eingebettete Synchronisationssignal EAV und gleichzeitig das Ende der Pixelzeile. LVAL wird auf 0 gesetzt, während FVAL weiterhin auf 1 bleibt. die Y- und Cb/Cr-Werte werden so gesetzt, dass sie die schwarze Farbe bilden. Wenn H_CNT den Wert 864 erreicht, ist die Zeile am Ende und die FSM geht in den Zustand *W_LINE*, um auf die nächste Pixelzeile zu warten. Wenn es in diesem Zustand auch vorkäme, dass VS auf 0 gesetzt wird, geht die FSM in den Zustand *W_FRAME*.

Listing 4.4: Default-Werte für die Signale in der FSM zur Erzeugung von FVAL und LVAL mit HS und VS aus dem Testbildgenerator.

```
92      begin    --default wert
93      H_CNT_S       <= (others=>'0');
94      H_CNT1_S          <= (others=>'0');
95      --black
96      Y_OUT_S           <= "00010000" ;
97      CBCR_OUT_S        <= "10000000";
98      LVAL_S        <= '0';
99      FVAL_S        <= '0';
100     F_FRAME_S     <= '0';
101     NEXT_STATE    <= IDLE;
```

4.3.2 Sync-Modul mit HS und VS aus dem VDEC1-Board

Dieses Modul unterscheidet sich von dem in Abschnitt 4.3.1. beschriebenen Modul dadurch, dass die Synchronisationspulse HS und VS aus dem VDEC1-Board nicht immer synchron zueinander treffen, was eigentlich nicht erwartet wird aber tatsächlich durch Messungen festgestellt worden ist[Vgl. Abb. 4.5]. In diesem Modul wird eine FSM und

zwei Counter benötigt, um LVAL und FVAL wie folgt zu generieren [Vgl. Abb. 4.7]. Dabei gilt genauso wie in dem Abschnitt 4.3.1, dass wenn in einem Zustand keine Zuweisung für die getakteten Signale *H_CNT*, *V_CNT*, *Y_OUT*, *CBCR_OUT*, *LVAL*, *FVAL*, *F_FRAME* oder *NEXT_STATE* erfolgt, werden sie durch ihre *Default*-Werte ersetzt, wie in dem Listing 4.5 beschrieben, wobei die Endung *„_S“* für die Signale stehen, die getaktet werden [Vgl. Listing 4.5].

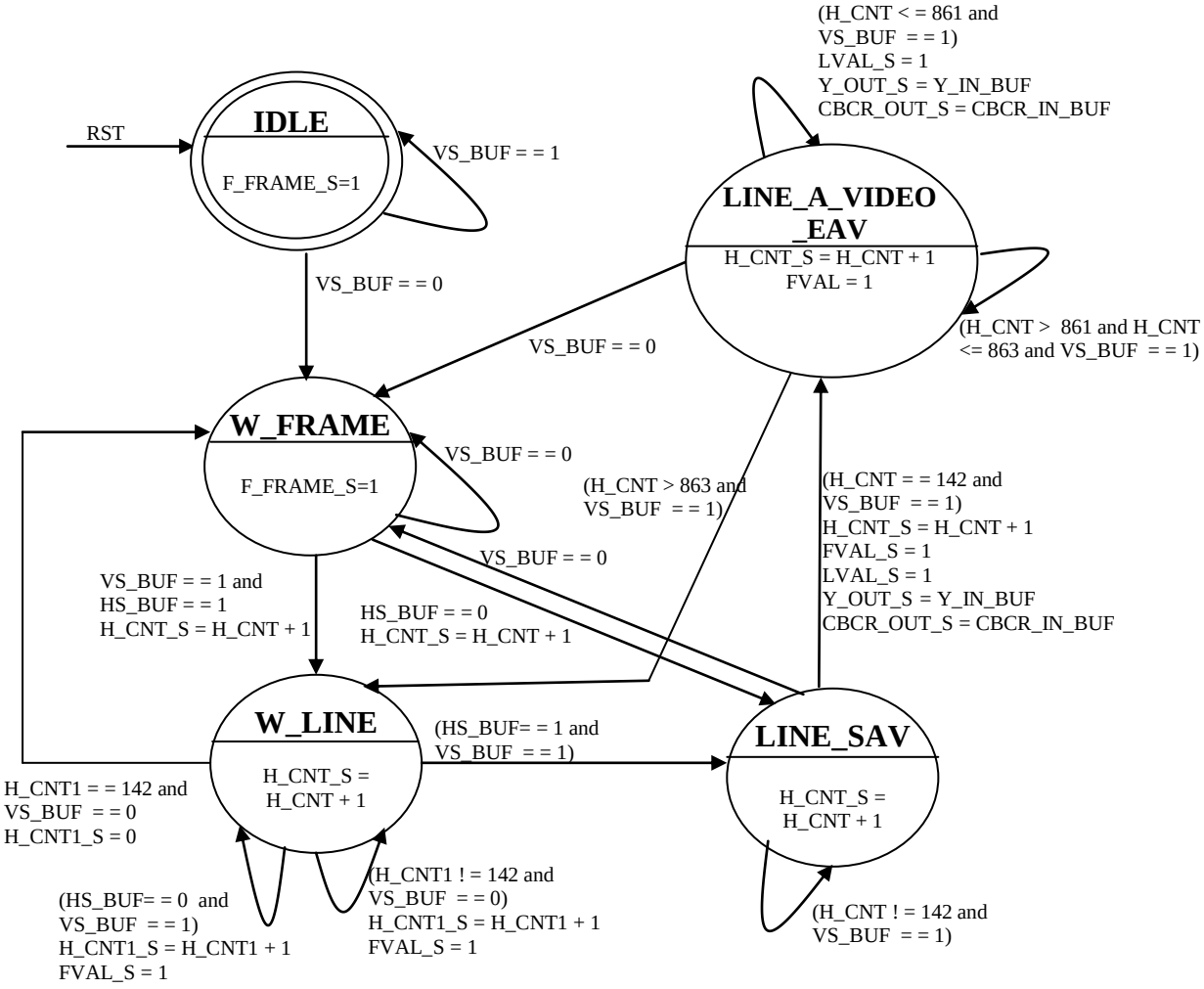

Abbildung 4.6: FSM zur Erzeugung von FVAL und LVAL mit HS und VS aus dem Testbildgerator

- Wie in der in Abschnitt 4.3.1 beschriebenen FSM ist der Startzustand dieser FSM *IDLE*: Bei der Initialisierung des Systems, landet die FSM in diesen Zustand, bis VS auf 0 gesetzt wird. Dabei wird das Signal *F_FRAME* auf 1 gesetzt und hat dieselbe Bedeutung wie in dem Abschnitt 4.3.1. Wenn VS auf 0 gesetzt wird, wird das Ende eines Frames signalisiert. Die FSM geht in den Zustand *W_F_FRAME*.
- Der Zustand *W_F_FRAME* besagt, dass der Synchronisationspuls VS für ein neues Frame eingetroffen ist, indem VS auf 0 gesetzt wird. Dabei bleibt das Signal *F_FRAME* weiterhin auf 1. Die FSM bleibt in diesem Zustand bis VS auf 1 gesetzt wird, wobei zwei Fälle zu unterscheiden sind. Wenn VS auf 1 gesetzt wird während HS auf 1 gesetzt wird, heißt es, dass der Anfang des neuen Frames innerhalb einer gültigen Pixelzeile eingetroffen ist. Die FSM geht in den Zustand IDLE, das ganze Frame wird nicht betrachtet und es wird auf ein neues Frame gewartet. Wenn aber VS auf 1 und

gleichzeitig HS auf 0 gesetzt werden, wird der Anfang der ersten Pixelzeile dieses neuen Frames signalisiert. Die FSM geht in den Zustand *W_LINE* und der Counter *H_CNT* wird hochgezählt.

- Die FSM befindet sich in dem Zustand *W_LINE*, wenn der Synchronisationspuls HS innerhalb eines Frames auf 0 gesetzt wird. Das heißt, dass eine neue Pixelzeile gerade anfängt. Da ein Frame aus 288 gültigen Pixelzeilen besteht, wird dieser Zustand jedes Mal angetroffen, wenn eine neue Pixelzeile in dem Frame anfängt. Das Signal *F_FRAME* wird benutzt damit das FVAL nicht innerhalb eines Frames auf 0 gesetzt wird, obwohl das Frame noch nicht zu Ende ist. Wenn also *F_FRAME* auf 0 gesetzt wird, heißt es, dass für das aktuelle Frame schon mehr als eine Pixelzeile angetroffen wurde. FVAL wird weiterhin auf 1 gesetzt. Wenn aber *F_FRAME* auf 1 gesetzt wird, geht es um die erste Pixelzeile dieses Frames. FVAL wird erst auf 1 gesetzt, wenn die gültigen Werte in dem Pixelstrom vorhanden sind. Die FSM verlässt diesen Zustand wenn HS auf 1 gesetzt wird und geht in den Zustand *LINE_SAV*.
- Der Zustand *LINE_SAV* besagt, dass der Synchronisationspuls HS schon den Anfang eine neue Pixelzeile signalisiert hat. Der Counter H_CNT wird bist 141 hochgezählt, was gemäß dem ITU-R BT.656 Format der Dauer der Signale HBLANK und SAV entspricht. Das *F_FRAME* wird genauso wie im Zustand *W_LINE* behandelt. Wenn H_CNT den Wert 141 erreicht hat, wird der Counter V_CNT, der die Anzahl der Pixelzeilen eines Frames zählt, hochgezählt. Wenn V_CNT zischen 1 und 286 (286 Pixelzeilen) oder 314 und 599 (286 Pixelzeilen) liegt, heißt es, dass für dieses Frame die Pixelzeilen gültig sind. LVAL und FVAL werden auf 1 gesetzt, und die zwischengespeicherten Y- und Cb/Cr-Werte aus dem VDEC1-Board werden als Ausgang zur Verfügung gestellt. Die FSM geht in den Zustand *LINE_A_VIDEO_EAV*. Wenn aber V_CNT zischen 287 und 313 (27 Pixelzeilen) oder 600 und 625 liegt (26 Pixelzeilen), heißt es, dass für dieses Frame die Pixelzeilen ungültig sind. H_CNT wird auf 1 gesetzt. LVAL und FVAL werden auf 0 gesetzt. Die Y- und Cb/Cr-Werte werden so gesetzt, dass sie die schwarze Farbe bilden und die FSM geht in den Zustand *W_FRAME*. Um sicherzustellen, dass keine Pixelzeile übernommen wird, wo HS und VS nicht synchron zu einander auftreten, wird die Anzahl der gültigen Pixelzeilen pro Frame auf 286 reduziert und somit die Anzahl der ungültigen Pixelzeilen auf 26 beziehungsweise 27 erhöht.
- Da diese FSM FVAL nicht mehr bezüglich VS, wie im Abschnitt 4.3.1 generiert, sondern mit dem Counter V_CNT, wird der Zustand *W_FRAME* benutzt um das korrekte FVAL zu generieren. Der Counter H_CNT wird von 0 bis 864 gezählt. Wenn H_CNT 864 erreicht hat, wird V_CNT hochgezählt. Solange V_CNT zischen 287 und 313 (27 Pixelzeilen) oder 600 und 625 liegt (26 Pixelzeilen), heißt es, dass für dieses Frame die Pixelzeilen ungültig sind. LVAL und FVAL werden auf 0 gesetzt. Die Y- und Cb/Cr-Werte werden so gesetzt, dass sie die schwarze Farbe bilden und die FSM bleibt in diesem Zustand. Wenn aber V_CNT zischen 1 und 286 (286 Pixelzeilen) oder 314 und 599 (286 Pixelzeilen) liegt, dann geht die FSM in den Zustand *LINE_A_VIDEO_EAV*. LVAL und FVAL werden auf 1 gesetzt. H_CNT bekommt den Wert 142.
- In dem Zustand *LINE_A_VIDEO_EAV* liegen in dem Pixelstrom die 720 Pixelwerte einer Pixelzeile und EAV. FVAL wird immer auf 1 gesetzt. Der Counter H_CNT wird hochgezählt bis er den Wert 863 erreicht, dann wird er auf 0 gesetzt. Wenn H_CNT zwischen 142 und 860 liegt, sind die gültigen Pixelwerte für die aktuelle Zeile vorhanden. LVAL wird auf 1 gesetzt und die zwischengespeicherten Y- und Cb/Cr-Werte aus dem VDEC1-Board werden als Ausgang zur Verfügung gestellt. Die FSM

bleibt in diesem Zustand. Wenn aber H_CNT zwischen 861 und 862 liegt, geht es um das EAV-Signal. Die Y- und Cb/Cr-Werte werden so gesetzt, dass sie die schwarze Farbe bilden und die FSM bleibt in diesem Zustand. Wenn H_CNT den Wert 863 erreicht, wird er auf 0 gesetzt, was dem Ende der Pixelzeile entspricht. Die FSM geht in den Zustand *W_LINE*.

Listing 4.5: Default-Werte für die Signale in der FSM zur Erzeugung von FVAL und LVAL mit HS und VS aus dem VDEC1-Board

```
101     begin    --default wert
102     H_CNT_S      <= (others=>'0');
103     V_CNT_S      <= V_CNT;
104     Y_OUT_S      <= "00010000";
105     CBCR_OUT_S   <= "10000000";
106     LVAL_S       <= '0';
107     FVAL_S       <= '0';
108     F_FRAME_S    <= '0';
109     NEXT_STATE   <= IDLE;
```

4.4 Umsetzung von 16 Bit ITU-R BT.656 YCbCr 4:2:2 in 24 Bit YCbCr 4:4:4

Da für das 16 Bit ITU-R BT.656 YCbCr 4:2:2 pro Takt einen 8 Bit Y-Wert und entweder einen 8 Bit Cb-wert oder einen 8 Bit Cr-Wert abwechselnd übertragen werden, ist das Ziel des HW-Moduls *DEMUX_UPSAMPLE* die fehlende Cb- bzw. Cr-Werte so zu ersetzen, dass pro Pixel 8 Bit Y- sowie Cb- und Cr- Werte verfügbar sind. Das Upsampling wird mit Pixelreplikation, Interpolation oder FIR-Filterung durchgeführt [Vgl. Poynton 2005], wobei für diese Arbeit die erste Methode benutzt wird.
Bei der Pixelreplikation geht es einfach darum den fehlenden Cb- beziehungsweise Cr-Wert durch den vom vorhergehenden Pixel zu ersetzen [Vgl. Abb. 4.8].

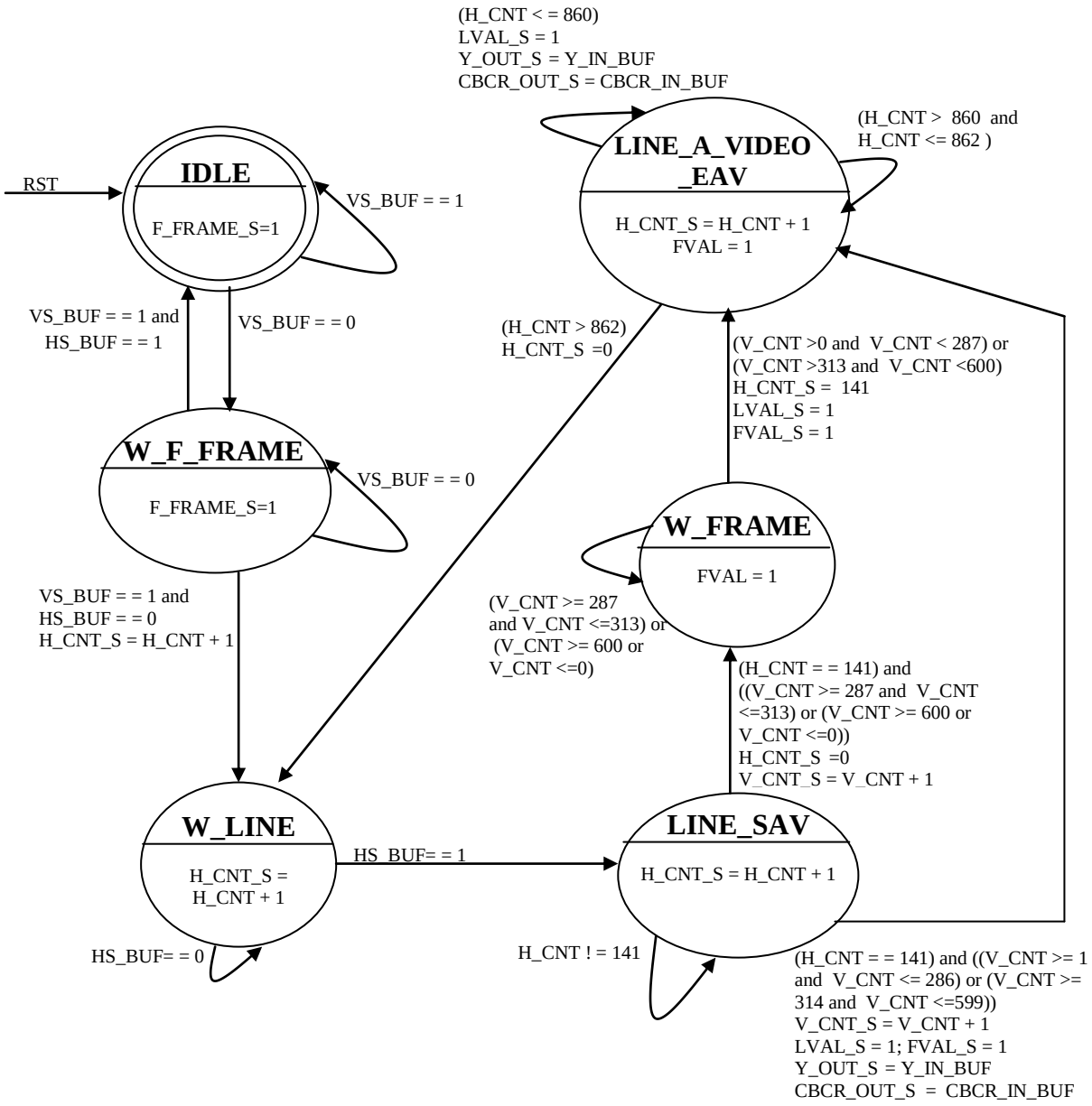

Abbildung 4.7: FSM zur Erzeugung von FVAL und LVAL mit HS und VS aus dem VDEC1-Board

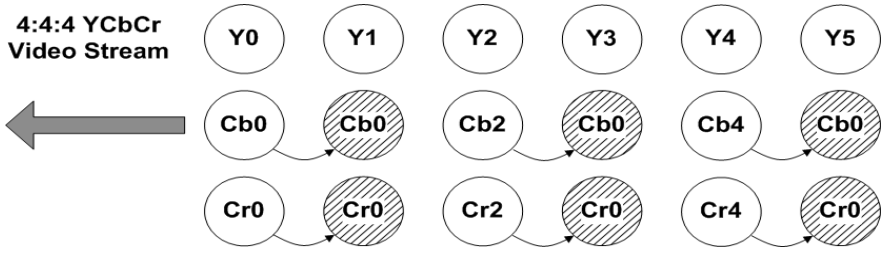

Abbildung 4.8: Upsampling von 16 Bit YCbCr 4:2:2 zu 24 Bit YCbCr 4:4:4 durch Pixelreplikation [Peters 2009].

Zur Implementierung der Pixelreplikation werden eine FSM und ein paar Schiebenregister benötigt. Wie die FSM und die Register zusammenarbeiten ist in folgenden beschrieben [Vgl. Abb. 4.8 sowie Abb. 4.9]:

- In dem Zustand *IDLE* befindet sich die FSM beim Reset und bleibt in diesem Zustand bis das Signal *FVAL_IN* auf 0 gesetzt wird. Das bedeutet, dass das aktuelle Frame gerade zu Ende ist. Die FSM geht in den Zustand *F_FRAME*. Die Ausgangsignale werden auf ihre Default-Werte gesetzt und zwar 16 für *Y_OUT*, 128 für *CB_OUT* und *CR_OUT* sowie 0 für *FVAL_OUT* und *LVAL_OUT*.
- Die FSM bleibt in dem Zustand *F_FRAME* bis *FVAL_IN* und *LVAL_IN* auf 1 gesetzt werden. Das bedeutet, dass eine neue Pixelzeile für das neue Frame gerade eingetroffen ist. Die FSM geht in den Zustand *Y0CB0* und die Ausgangssignale werden wie im *IDLE*-Zustand auf ihre Default-Werte gesetzt.
- Der Zustand *Y0CB0* besagt, dass für die aktuelle Pixelzeile die Pixelwerte Y0 und Cb0 eingetroffen sind. Die FSM geht unmittelbar in den Zustand *Y1CR0*, wenn die Pixelzeile gültig ist, indem LVAL_IN und FVAL_IN auf 1 gesetzt sind. Dabei werden Y0, Cb0 sowie die entsprechenden Signale LVAL und FVAL mitgespeichert. Wenn aber LVAL_IN oder FVAL_IN auf 0 gesetzt wird, geht die FSM in den Zustand *F_FRAME* und wartet bis eine gültige Pixelzeile eintrifft. Wie im *F_FRAME*-Zustand werden auch die Ausgangsignale auf ihre Default-Werte gesetzt.

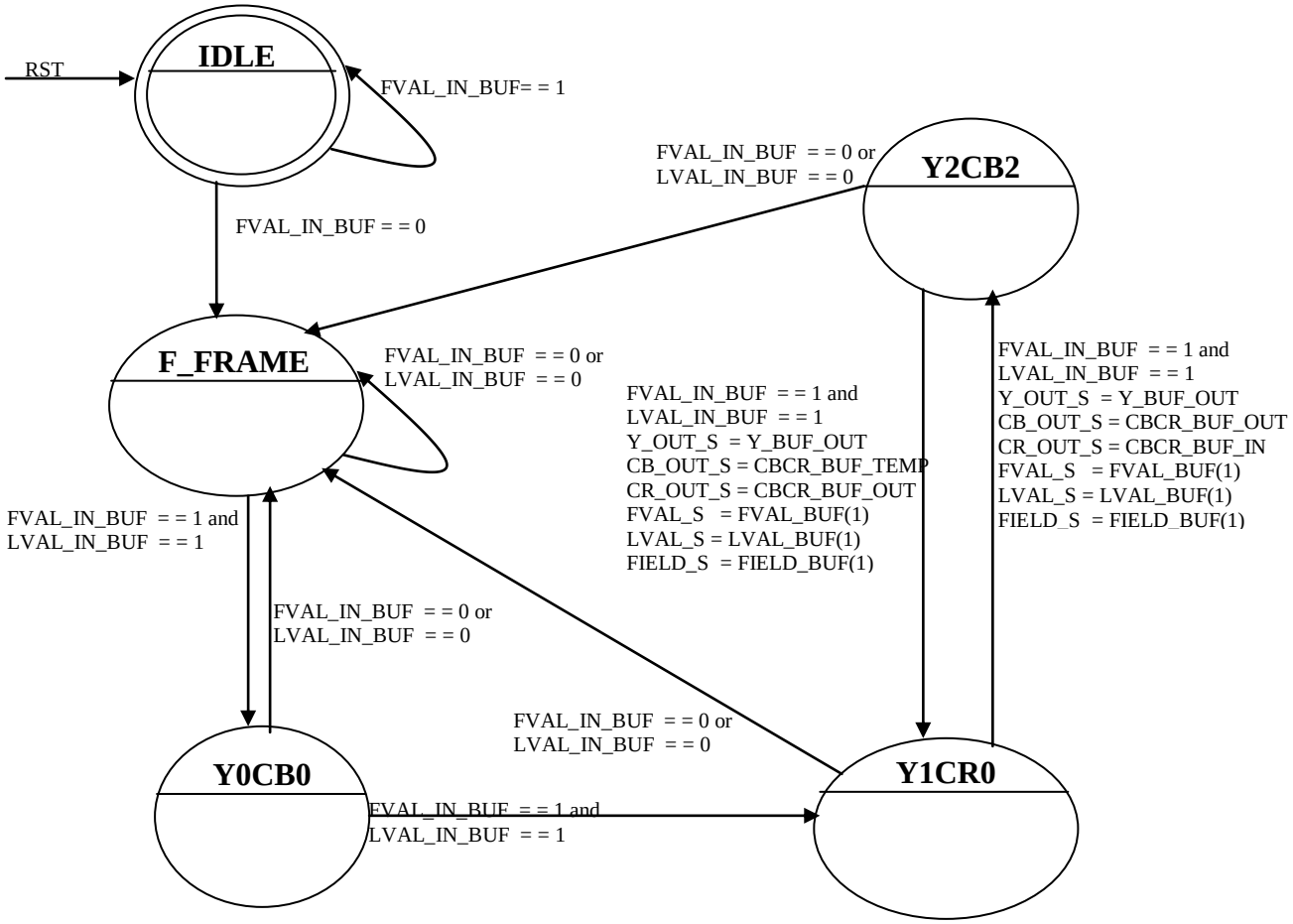

Abbildung 4.8: FSM zur Umwandlung von 16 Bit YCbCr 4:2:2 zu 24 Bit YCbCr 4:4:4 durch Pixelreplikation

- In dem Zustand *Y1CR0* sind zu den Pixelwerten Y0 und CB0 auch Y1 und CR0, sowie deren Synchronisationssignale LVAL und FVAL eingetroffen. Die Ausgangsignale werden dann berechnet, indem *Y_OUT Y_BUF_OUT* zugewiesen wird, was dem zwischengespeicherten Y0-Wert entspricht. *CB_OUT* bekommt *CBCR_BUF_OUT*, was *CB0* entspricht. *CR_OUT* bekommt *CBCR_BUF_IN*, was *CR0* entspricht. Die mitgespeicherte LVAL und FVAL werden an LVAL_OUT und FVAL_OUT weitergeleitet. Dabei müssen LVAL_IN und FVAL_IN auf 1 gesetzt werden und die FSM geht unmittelbar in den Zustand *Y2CB2*. Wenn aber eines der beiden Synchronisationssignalen auf 0 gesetzt wird, ist die Pixel Zeile nicht gültig. Die FSM geht wie im vorherigen Zustand in den Zustand *F_FRAME* und wartet bis eine gültige Pixelzeile eintrifft. Die Ausgangssignale werden auch auf ihre Default-Werte gesetzt.
- Wenn dir Pixelwerten Y2 und Cb2 eintreffen ist die FSM im Zustand *Y2CB2*. Die Ausgangsignale werden wie im Zustand *Y1CR0* berechnet, indem *Y_OUT Y_BUF_OUT* zugewiesen wird, was dem zwischengespeicherten Y1-Wert entspricht. *CB_OUT* bekommt aber *CBCR_BUF_TEMP*, was weiterhin *CB0* entspricht. *CR_OUT* bekommt aber auch *CBCR_BUF_OUT*, was weiterhin *CR0* entspricht. Die mitgespeicherte LVAL und FVAL werden an LVAL_OUT und FVAL_OUT weitergeleitet. Dabei müssen auch hier LVAL_IN und FVAL_IN auf 1 gesetzt werden und die FSM geht unmittelbar in den Zustand *Y1CR0*, was die Bedeutung hat, dass auf die Y3-und Cr2-Werte gewartet werden. Wenn aber ein von den beiden Signalen auf 0 gesetzt wird, ist die Pixelzeile nicht gültig. Die FSM geht wie im vorherigen Zustand in Zustand *F_FRAME* und wartet bis eine gültige Pixelzeile eintrifft. Die Ausgangsignale werden auch auf ihre Default-Werte gesetzt.

4.5 Umrechnung von 24 Bit YCbCr 4:4:4 in 24 Bit RGB

Das Ziel des HW-Moduls *YCBCR2RGB* ist die Umrechnung von den 24 Bit YCbCr-Werten jedes Pixels in 24 Bit RGB. Das 8 Bit Y-Signal liegt üblicherweise zwischen 16 und 235 während das 8 Bit Cb-Signal sowie Cr-Signal zwischen 16 und 240 liegen. In der Berechnung wird beachtet, dass die 8 Bit R-, G- sowie B-Signale zwischen 0 und 255 liegen. Für die Berechnung werden die drei folgenden Gleichungen benutzt [Vgl. Jack 2005].

$$R = 1.164(Y - 16) + 1.596(Cr - 128) \quad (4.1)$$
$$G = 1.164(Y - 16) - 0.813(Cr - 128) - 0.391(Cb - 128) \quad (4.2)$$
$$B = 1.164(Y - 16) + 2.018(Cb - 128) \quad (4.3)$$

In diesem HW-Modul werden die Gleichungen (4.1), (4.2), sowie (4.3) in Integer-Arithmetik implementiert. Das heiß sie werden so multipliziert, dass die Nachkommateile bei den Faktoren ausgelöscht werden und anschließend werden sie entsprechend so geteilt, dass das Ergebnis korrekt ist. Die drei Gleichungen sehen so aus [Vgl. Peters 2009]:

$$R = (298Y + 409Cr - 57065) / 256 \quad (4.4)$$
$$G = (298Y - 100Cb - 208Cr + 34658) / 256 \quad (4.5)$$
$$B = (298Y + 516Cb - 70894) / 256 \quad (4.6)$$

Die Gleichung (4.4) wurde hergeleitet, indem die Gleichung (4.1) mit 256 multipliziert, umgestellt und anschließend mit 256 geteilt wurde, wobei die Faktoren abgerundet sind. Genau so wurden auch die Gleichung (4.5) beziehungsweise (4.6) von den Gleichungen

(4.2) beziehungsweise (4.3) hergeleitet. Die Auswahl von 256, was eine Potenz von 2 ist, lässt sich dadurch erklären, dass die Division mit einer Potenz von 2 in HW einfach durch Schieben realisiert wird. Darüber hinaus ist 256 genug groß damit die Abrundung in den Faktoren minimiert wird [Vgl. Abb. 4.10].
In der VHDL-Implementierung werden die Gleichungen (4.4), (4.5), sowie (4.6) sukzessiv gerechnet [Vgl. Abb. 4.10]. Die Vektorbreiten für die Berechnung der Zwischenergebnisse werden auf 20 gesetzt, um Overflow bzw. Underflow zu vermeiden. Der Wertbereich von R sowie G und B wird durch die Division mit 256 zwischen 0 und 255 begrenzt [Vgl. Anhang 9.1]. Die zwischengespeicherten Synchronisationssignale werden mit den entsprechenden R-, G-, B-Werte als Output zur Verfügung gestellt.

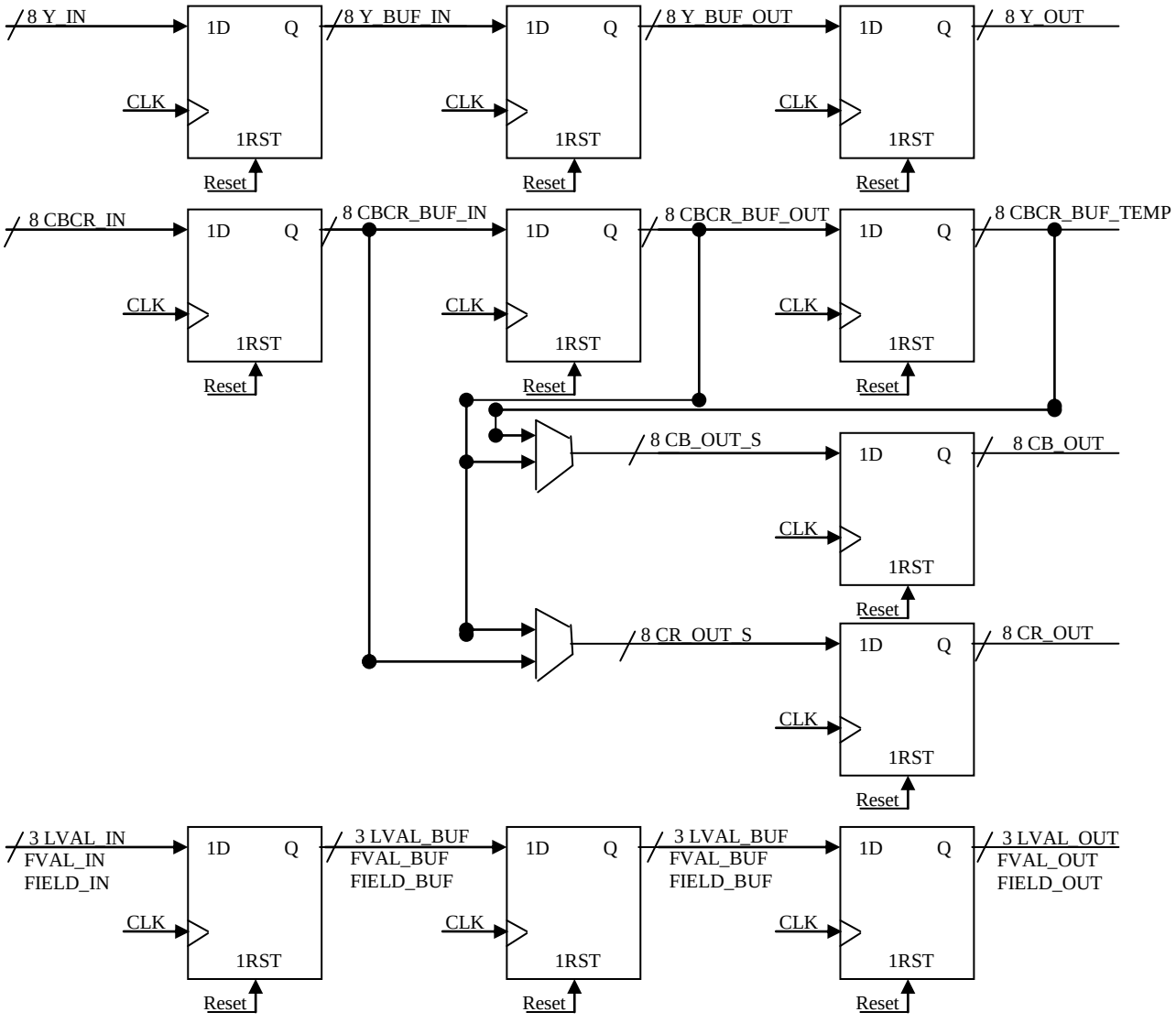

Abbildung 4.9: Datenpfad des HW-Moduls *DEMUX_UPSAMPLE* zur Umwandlung von 16 Bit YCbCr 4:2:2 zu 24 Bit YCbCr 4:4:4 durch Pixelreplikation.

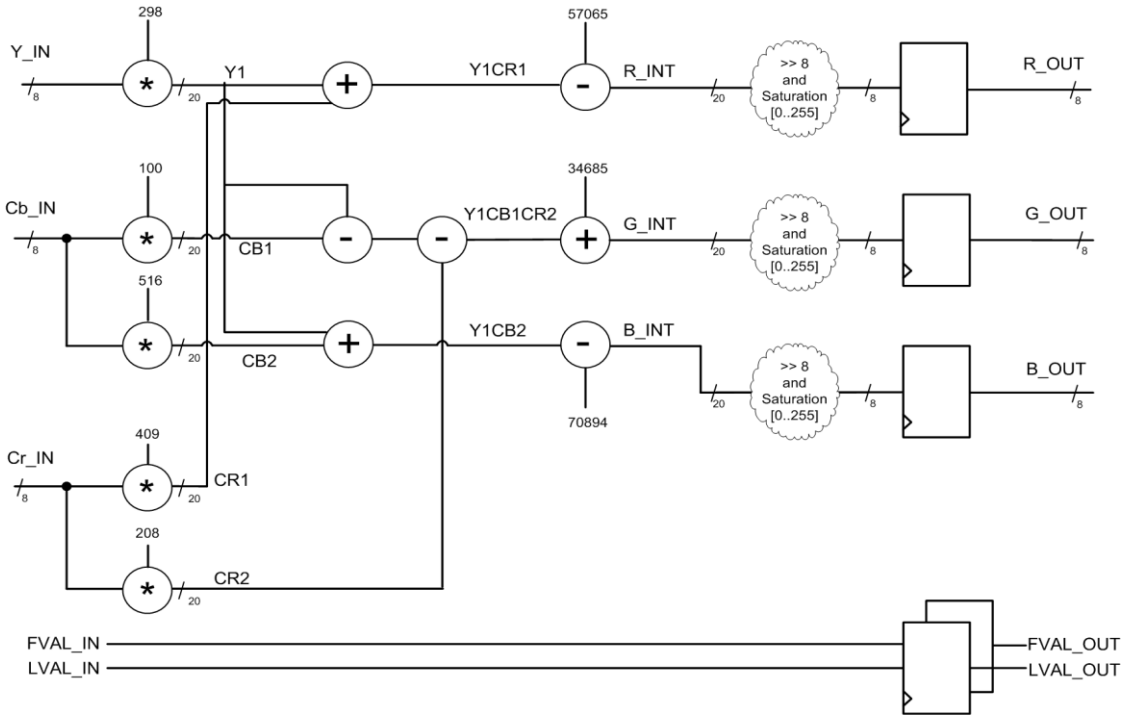

Abbildung 4.10: Datenpfad des HW-Moduls *YCBCR2RGB* zur Umrechnung von 24 Bit YCbCr 4:4:4 in 24 Bit RGB [Peters 2009]

4.6 Deinterlacing des 24 Bit RGB *Interlaced*-Videosignals

Da der TFT-Monitor nur einen *Progressive* Videostrom ausgeben kann, wird in diesem Modul das 24 Bit RGB *Interlaced*-Videosignal in einem 24 Bit RGB *Progressive* Videosignal umgewandelt. Dabei ist die Auflösung 720x572 mit 60 Hz. Da im *Interlaced*-Videosignal nur die Hälfte der Pixelzeile verfügbar ist, ist das Deinterlacing eingesetzt, um die fehlende Pixelzeile zu generieren. Dafür sind folgende Methoden verfügbar [Vgl. Jack 2005], wobei für die Arbeit wegen ihrer einfachen Implementierung die erste Methode gewählt wird:

- Die Pixelzeilenverdopplung: In dieser Methode wird jede Pixelzeile verdoppelt, so dass die vertikale Auflösung dadurch verbessert wird [Vgl. Abb. 4.11].

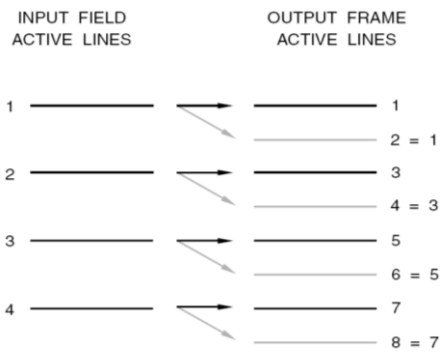

Abbildung 4.11: Deinterlacing durch Verdopplung von Pixelzeilen

- Die Pixelzeile Interpolation: Bei dieser Methode wird eine fehlende Pixelzeile durch den Mittelwert aus der vorherigen und der nächsten Pixelzeile berechnet. Obwohl die Pixelzeilen verdoppelt werden, wird aber die vertikale Auflösung nicht verbessert.
- Eine bessere aber aufwändigere Methode ist die Filterung mit einem FIR-Filter. Der Aufwand liegt darin, dass für die Berechnung der fehlenden Pixelzeile zwölf Pixelzeilen benötigt werden.

Die HW-Implementierung des Deinterlacing mit einer doppelten Pixeltaktrate (27 MHz) ist wie folgt geteilt[Vgl. Abb. 2.12 sowie Abb. 4.12]:

- Die *Control*-FSM steuert die beiden Schiebenregister zur Zwischenspeicherung der Pixelzeilen und erzeugt dabei das Synchronisationssignal FVAL [Vgl. Abb. 4.13]. Mit dem Reset des Systems ist die *Control*-FSM im Zustand *NONE* und die beiden Scheibenregister sind deaktiviert. Dieser Zustand wird verlassen, wenn eine steigende Flanke beim LVAL auftritt, was dem Anfang einer gültigen Pixelzeile entspricht. Die FSM geht in den Zustand *FST*. In dem Zustand *FST* wird das erste Scheibenregister aktiviert und die aktuelle Pixelzeile wird gespeichert. Dabei bliebt das zweite Schiebenregister inaktiv. Wenn eine steigende Flanke beim LVAL auftritt geht die FSM in den Zustand *SND*. Das zweite Schiebenregister wird aktiviert und das erste deaktiviert. Die FSM bleib also entweder in dem Zustand *FST* oder *SND* bis die Anzahl der Pixelzeilen für das Frame erreicht wird. Die FSM geht in den Zustand *NONE* und wartet, bis die neue gültige Pixelzeile auftritt.

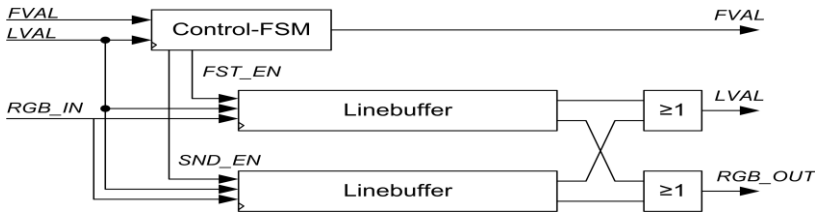

Abbildung 4.12 Blockdiagramm des Deinterlacing Moduls [Peters 2005]

- Das *Linebuffer*-Modul besteht aus einem Schiebenregister und einer FSM [Vgl. Abb. 4.14] zur Zwischenspeicherung und Ausgabe der Pixelzeilen. Dieses Modul wird zweimal instanziiert, sodass während in einer Instanz die aktuelle Pixelzeile gespeichert wird, die andere Instanz die alte Pixelzeile zweimal ausgibt. Wenn eine Instanz das *Enable*-Signal der *Control*-FSM bekommt, geht seiner FSM vom Zustand *IDLE* in Zustand *RUNNING*. Im Zustand *RUNNING* wird der Counter *CNT* gestartet. Je nach dem *CNT*-Stand wird die Pixelzeile mit der Pixeltaktrate (13.5 MHZ) gespeichert und dann zweimal mit dem entsprechenden LVAL-Signal ausgegeben und die FSM kehrt in den *IDLE*-Zustand zurück. Das im *Linebuffer*-Modul instanziierte Schiebenregister wird mit dem Xilinx Tool *Core Generator* generiert und so parametriert, dass es genau 720 Takten (Pixelanzahl pro Zeile) benötigt, um ein Signal, was an seinem Eingang vorliegt, an seinem Ausgang zur Verfügung stellt.
- Das 24 Bit Output-Signal *RGB_OUT sowie das LVAL_OUT*-Signal werden durch eine OR-Verknüpfung der Ausgänge von den beiden *Linebuffer*-Instanzen generiert [Vgl. Abb. 4.12]. *LVAL_OUT* wird auch benutzt, um *FVAL_OUT* zu generieren und, um die *Control*-FSM auf dem *NONE*-Zustand zu setzen, in dem bei jeder steigenden Flanke von LVAL_OUT die Pixelzeilen gezählt werden.

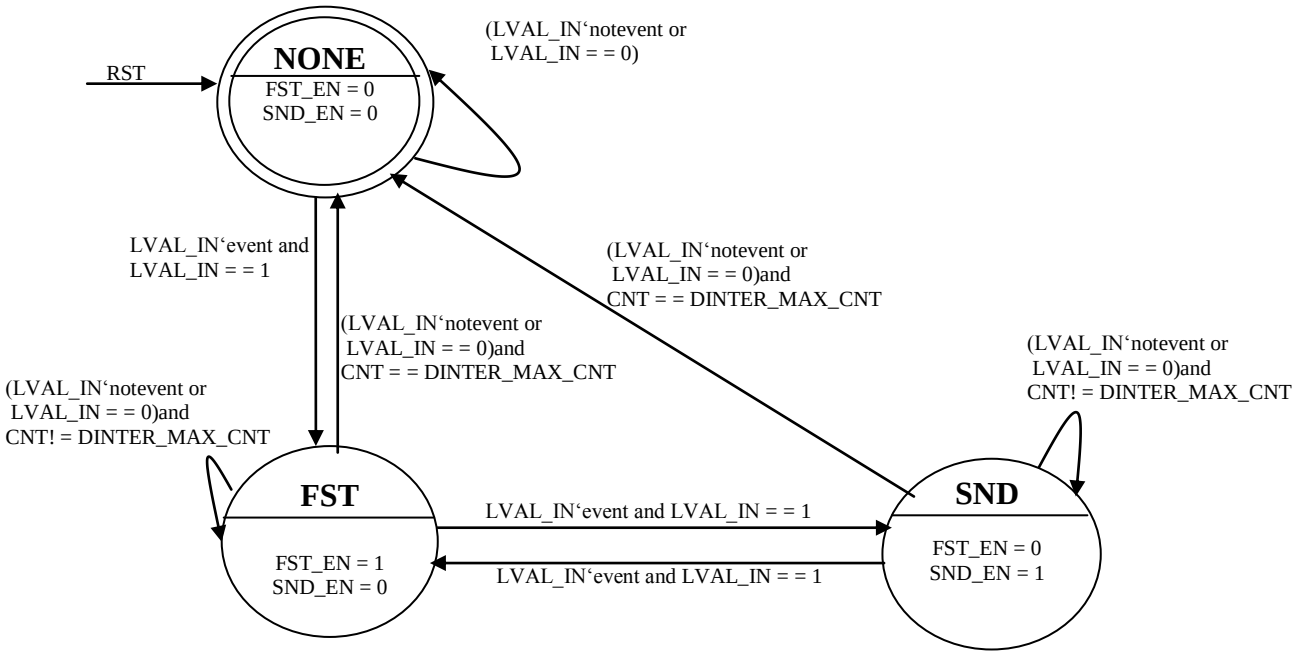

Abbildung 4.13: Control-FSM zur Steuerung der beiden *Linebuffer*-Instanzen (Schiebenregister).

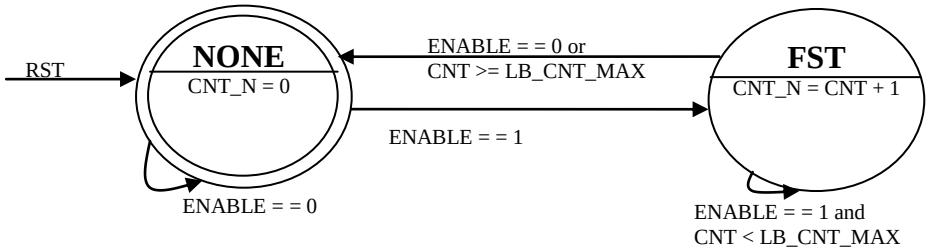

Abbildung 4.14: FSM zur Steuerung des Schiebenregisters im *Linebuffer*-Modul.

4.7 Parametrierung des VGA-Controllers und Ausgabe auf dem TFT-Bildschirm

Der VGA-Controller dient dazu, die zur Steuerung des Monitors benötigten Synchronisationspulse HSYNC (*Horizontal Synchronization*) und VSNYC (*Vertical Synchronization*), zu generieren. Diese beiden Pulse sind low-activ und werden während ihrer jeweiligen Blankingsphasen auf 0 gesetzt [Vgl. Digilent 2008, sowie Abb. 4.15]. Da Pixelzeilen pixelweise auf dem Monitor in einer Richtung (von links nach rechts und von oben nach unten) dargestellt werden, signalisieren diese beiden Synchronisationspulse das Ende einer Pixelzeile beziehungsweise eines Frame. Dabei werden die Pixelwerte geblänkt, damit der Bildschirm die Zeit hat, um die Darstellung korrekt zu positionieren. Die Tabelle 4.3 stellt die Timingparameter dar [Vgl. Tab. 4.3].

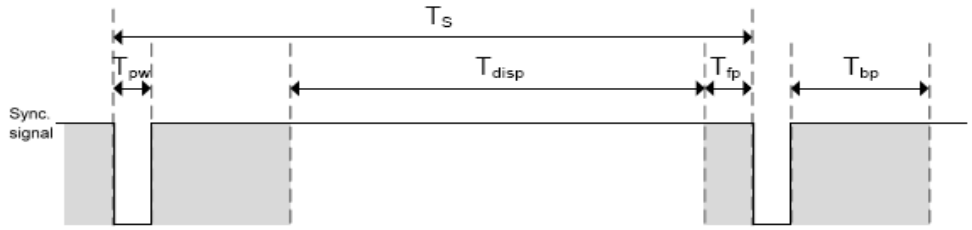

Abbildung 4.15: Timingbezeichner der VGA-Synchronisationspulse HSYNC und VSYNC [Digilent 2008].

Das HW-Modul zur Erzeugung dieser beiden Pulse besteht aus einer FSM und zwei Zähler mit Komparatoren:

- Die FSM dient dazu, die Zähler zu starten wenn eine gültige Pixelzeile vorliegt [Vgl. Abb. 4.16]. Beim Reset ist die FSM im *PWRON*. Die FSM bleibt in diesem Zustand, bis *FVAL_IN* auf 0 gesetzt wird. Wenn *FVAL_IN* auf 0 gesetzt ist, endet das aktuelle Frame. Die FSM geht dann in den Zustand *WAIT_FRAME1*. Dabei werden die Zähler nicht gestartet, da *EN_COUNT* auf 0 gesetzt wird. In dem Zustand *WAIT_FRAME1* wird gewartet, bis das neue Frame anfängt. Das geschieht, wenn *FVAL_IN* auf 1 gesetzt ist. Die FSM geht in den Zustand *FRAME1*. *EN_COUNT* bleibt weiterhin auf 0. Der Zustand *FRAME1* wird verlassen, wenn das neue Frame zu Ende ist, indem *FVAL_IN* auf 0 gesetzt ist. Die FSM geht in den Zustand *ACTIVED*. Die Zähler werden gestartet, da *EN_COUNT* auf 1 gesetzt wird.

Tabelle 4.3: Timing der VGA-Synchronisationspulse

Parameter	**Symbol**	**Vertikal Lines**	**Horizontal CLKs**
Periodendauer	Ts	625	864
Display Time	T_{disp}	572	720
Pulse Width	T_{pw}	3	86
Front Porch	$T_{f\,p}$	14	16
Back Porch	T_{bp}	36	42

- Wenn EN_COUNT auf 1 gesetzt ist, wird der Counter zur Erzeugung von HSYNC gestartet (H_CNT) und bei jeder steigenden Flanke hochgezählt. Das Signal HBLANK wird auf 0 gesetzt, wenn der Stand dieses Counters zwischen HT_BP_END+1 und HT_DP_END liegt. Das heißt, dass die Pixelwerte gültig sind. Sonst wird HBLANK auf 1 gesetzt. HSYNC wird auf 0 gesetzt, wenn der H_CNT-Stand zwischen HT_FP_END+1 und HT_PULSE_END liegt. Sonst wird es auf 1 gesetzt. Wenn H_CNT den Wert HT_MAX erreicht hat, wird er auf 0 gesetzt und der Counter zur Generierung von VSYNC (V_CNT) wird einmal hochgezählt. H_CNT wird weiterhin jeder steigenden Flanke hochgezählt und sein Stand wird zur Generierung von HBLANK und HSYNC abgefragt.
- Jedes Mal, wenn V_CNT hochgezählt wird, wird sein Stand abgefragt. Das VBLANK-Signal wird auf 0 gesetzt, wenn der V_CNT-Stand zwischen VT_BP_END+1 und VT_DP_END liegt. Sonst wird VBLANK auf 1 gesetzt. VSYNC wird auf 0 gesetzt, wenn der V_CNT-Stand zwischen VT_FP_END+1 und VT_PULSE_END liegt. Sonst

wird es auf 1 gesetzt. Wenn V_CNT den Wert HT_MAX erreicht hat, wird es auf 0 gesetzt und hochgezählt wenn H_CNT den Wert HT_MAX erreicht.

- Wenn HBLANK und VBLANK auf 1 gesetzt sind, sind die Ausgänge gültig und die RGB-Pixelwerte werden dargestellt. Sonst werden die Pixelwerten auf 0 gesetzt (Schwarz).

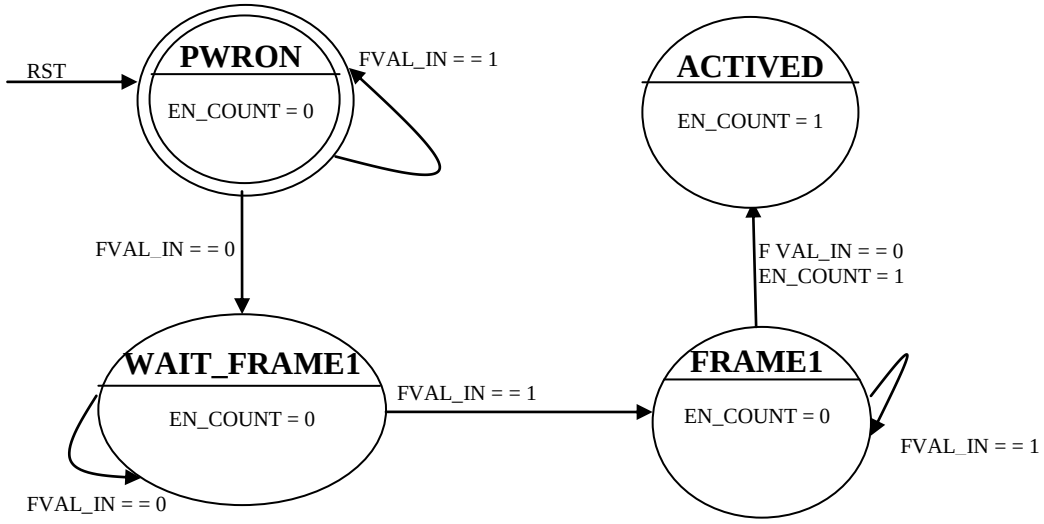

Abbildung 4.15: FSM zur Steuerung der Module zur Erzeugung von HSYNC und VSYNC

Bevor die 24 Bit RGB-Werte, sowie die Synchronisationspulse HSYNC und VSYNC mit den VGA-Pins des Nexys2-Boards in der UCF-Datei angekoppelt werden, müssen die 24 Bit RGB auf 8 Bit reduziert werden, da der VGA-Controller nur 10 Pins zur Verfügung stellt, wobei 8 Pins für die Pixeldaten und 2 Pins für HSYNC und VSYNC gedacht sind. Dafür werden 3 Pins für den R-Wert, 3 Pins für den G-Wert und 2 Pins für B-Wert reserviert. Mit dem TestBildgerator als Pixelquelle sind die Farbenstreifen korrekt auf dem TFT-Monitor angezeigt [Vgl. Abb. 4.16]. Mit dem Pixelstrom aus dem VDEC1-Board wird bis zum Abschluss dieser Arbeit kein korrektes Bild dargestellt. Die Konturen des Bildes sind zwar auf dem TFT-Bildschirm zu erkennen aber die Farben sind falsch, so dass Rauchen entstanden. Um dieses Problem zu Lösen, wäre eine mögliche Lösung, die Prüfung der Pixelwerte auf Korrektheit am Eingang und Ausgang jedes HW-Modules. So könnte festgestellt werden, woher der Fehler kommt.

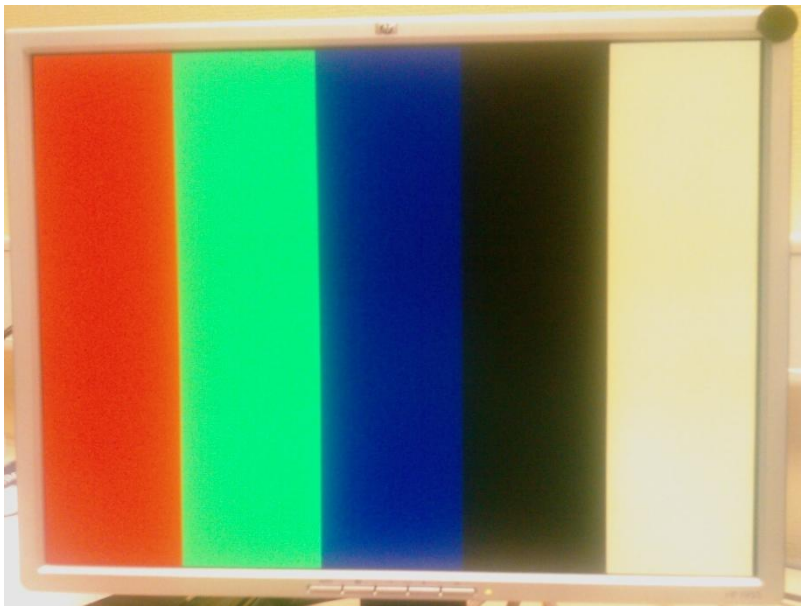

Abbildung 4.16: Ausgabe des Videostroms aus dem Testbildgenerator auf einem TFT-Monitor

5 Bildtiefenauswertung mit stereoskopischen Signalen

Im Bereich der digitalen Videosignalverarbeitung ist die Abschätzung der Disparität mit stereoskopischen Videosignalen ein sehr relevantes Thema. Die mit der Disparität gewonnenen Tiefendaten, werden sowohl in Navigationssystemen, als auch in der Robotik, sowie in Überwachungssystemen eingesetzt. Da die Disparitätsabschätzung sehr komplex ist, wurden dafür viele Methoden entwickelt sowohl für PC-Anwendungen als auch für HW-basierende Systeme entwickelt. Der Vorteil bei den HW-basierten Systemen ist, dass die Berechnung parallel ausgeführt werden kann, was auch in vielen PC-Anwendungen implementiert ist [Vgl. Masrani 2006]. In diesem Kapitel werden im ersten Abschnitt Begriffe und Konzepte zur Bestimmung der Disparität dargestellt. Eine Phasenverschiebung basierte HW-Methode wird das Thema des zweiten Abschnitts. Anschließend wird ein Matlab-Modell dieser Methode erläutert.

5.1 Konzept zur Bestimmung von Objektentfernungen

Stereoskopisches Tiefensehen beruht auf den durch die Parallaxe bedingten Unterschieden zwischen den Bildern der beiden Augen [Vgl. Mallot 2000]. Anders gesagt, durch den Abstand der Augen, wird jedes Objekt aus 2 verschiedenen Winkel gesehen, woraus sich ergibt, dass die beiden Bilder (von dem linken und rechten Auge), auch Halbbilder genannt, feine Unterschiede aufweisen. Wenn man ein Objekt fixiert, konvergieren die Sehachsen, wobei diese Konvergenz von der Objektentfernung abhängig ist. Wenn das Objekt sehr weit ist, sind die Sehachsen eher parallel zueinander. Je näher das zu betrachtende Objekt an die Augen kommt, desto weiter konvergieren die Achsen. Die Parallaxe ist definiert, als der Winkel, den die Sehachsen am gesehenen Objekt bilden. Der örtliche Unterschied des gleichen Teilobjektes zwischen den zwei Halbbildern heißt parallaktische Verschiebung [Vgl. Zang 2009, sowie Abb. 5.1].

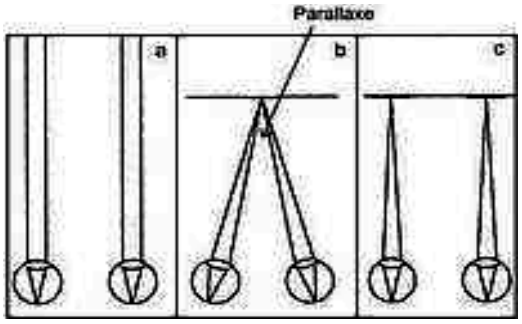

Abbildung 5.1: Konvergenz der Sehachsen bei einem weitentfernten Objekt (a), sowie einem nahliegenden Objekt (b und c) [Zang 2009] .

Aus den Bildern des linken und rechten Auges baut das Gehirn ein 3D Raumbild. Dabei wird der Abstand zwischen dem betrachteten Objekt und den Augen aus der parallaktischen Verschiebung hergeleitet [Vgl. Zang 2009].

Dieser Zusammenhang zwischen stereoskopischen Tiefeneindruck und Bildunterschieden wurde von Wheatstone, im Jahr 1838 mit Hilfe eines Spiegelstereoskops, bewiesen. Das Spiegelstereoskop trennt die Bilder mit Hilfe zweier gegeneinander gekippter Spiegel in zwei Halbbilder, die an den Bildhaltern angebracht und über die beiden Spiegel betrachtet werden. Es entsteht ein fusioniertes Scheinbild mit einem starken, stereoskopischen Tiefeneindruck [Vgl. Mallot 2000, sowie Abb. 5.2].

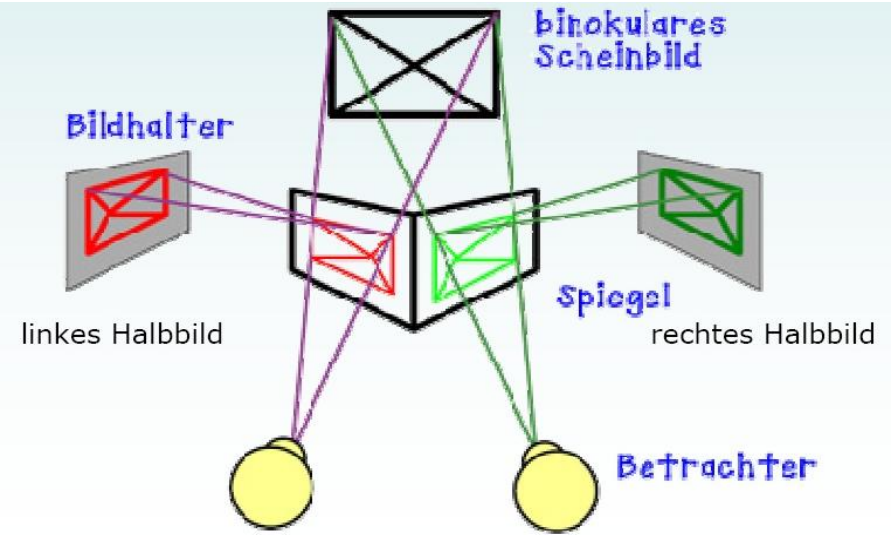

Abbildung 5.2: Schema des von Wheatstone entwickelten Spiegelstereoskops zum Beweis des Zusammenhangs zwischen stereoskopischen Tiefeneindruck und Bildunterschieden [Mallot 2000].

Die parallaktische Bildunterschiede, auch Disparationen genannt, kommen in verschiedenen Typen Vor [Vgl. Mallot 2000, sowie Hardieß 2007]:

- Querdisparität: Ein isolierter Punkt in der 3D Welt wird in den beiden Halbbildern an leicht verschiedenen Stellen abgebildet. Die horizontal gemessene Differenz der Abstände zum Bildmittelpunkt oder Winkel zur Sehachse heißt Querdisparität.
- Die vertikale Disparität ist die vertikale Positionsdifferenz, die bei vergierenden Blickachsen auftritt.
- Die Orientierungsdisparität bezeichnet die Steigungsdifferenz, die bei einer schiefen Gerade in den beiden Halbbildern auftritt.
- Schattierungsdisparität: Glatte Oberflächen reflektieren Licht in Abhängigkeit ihrer Orientierung in Bezug zur Lichtquelle (Einfallrichtung) und Betrachter (Ausfallrichtung). Betrachtet man eine glatt schattierte Kugel mit beiden Augen, so sind die Farbwertverläufe für beide Augen (für beide Bilder) verschieden. Die Gründe dafür sind dass, einerseits die Oberflächenpunkte, welche auf den korrespondierenden Netzhautpunkten abgebildet werden, verschiedene lokale Orientierungen haben. Man spricht von Grauwertdisparität. Anderseits, strahlt derselbe Punkt des Körpers bei glänzenden Oberflächen unterschiedlich viel Licht in die beiden Augen. Dies heißt photometrische Disparität.

In der Praxis werden die beiden Augen durch zwei Kameras abbildet. Dabei ist sehr wichtig diese beiden Kameras so zu kalibrieren, dass ihre inneren Parameter (Brennweite und Position des Targets in Bezug auf den Knotenpunkt) gleich sind [Vgl. Mallot 2000].

Obwohl diese Kalibrierung manchmal schwer zu realisieren ist, existiert in der Literatur unter dem Namen Kamerakalibrierung (*camera calibration*) mehrere Methoden, um sie zu korrigieren [Vgl. Klinker 2004, Sowie Hartley 2000], was aber in dieser Arbeit nicht behandelt wird.
Die Stereogeometrie, auch binokular Perspektive genannt, erläutert den Zusammenhang zwischen den inneren Parametern der beiden Kameras, der gerechneten Disparität und der Tiefenauswertung. Dabei können die Kameraachsen parallel oder konvergent sein. Für diese Arbeit werden die Kameras mit parallelen Achsen bevorzugt, da die hier zur Disparitätsabschätzung eingesetzte Methode dies voraussetzt.
Wenn die Kameraachsen parallel zueinander ausgerichtet sind, gibt es nur die Querdisparität, da für einen Punkt, der dem Punkt $P_l = (x_l,y_l)$ auf dem linken Bild, sowie $P_r = (x_r,y_r)$ auf dem rechten Bild entspricht, sind y_l und y_r gleich, so dass die vertikale Disparität immer 0 ist. Sei b die Basis des Stereokamerasystem, wobei b für unseren Fall, wo, die Kameraachsen parallel zu einander sind, senkrecht zu der Verbindunglinie der Kameraknotenpunkte steht, f die Brennweite der Kamera und z der vertikale Abstand zwischen dem Punkt P und der Basislinie [Vgl. Abb. 5.3], dann ist die Querdisparität d (hier immer negativ) umgekehrt proportional zu z und wächst mit f und b, wie in der folgenden Gleichung beschrieben [Vgl. Mallot 2000].

$$d = x_l - x_r = -f\frac{b}{z} \qquad (5.1)$$

Um eine bessere Tiefenauflösung zu bekommen werden in technischen Systemen größere Werte für b bevorzugt[Vgl. Mallot 2000].

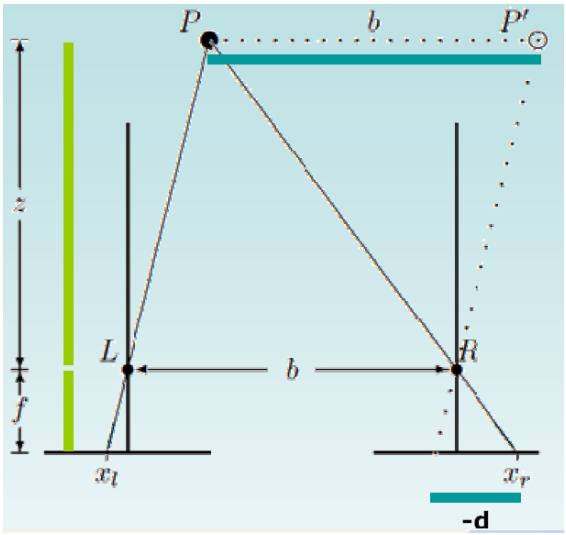

Abbildung 5.3: Querdisparität bei paralleler Kameraausrichtungen [Mallot 2000].

In der Gleichung 5.1 wird nur gesagt, wie aus P_l und P_r d gerechnet wird. Davor müssen aber die korrespondierenden Punkte P_l und P_r bestimmt werden. Das Problem ist also die eindeutige Zuordnung dieser korrespondierenden Punkte, was auch als Korrespondenzproblem bekannt ist. Um dieses Problem zu minimieren, wird die Epipolargeometrie verwendet. Die Epipolargeometrie ist ein mathematisches Modell aus der Geometrie, das die geometrischen Beziehungen zwischen verschiedenen (zwei in

dieser Arbeit) Kamerabildern des gleichen Objekts darstellt [Vgl. Wikipedia 2007]. Die Epipolarbedingung besagt, dass für einen Punkt X_l auf dem linken liegen seine möglichen korrespondierenden Punkte X_r auf der Epipolarlinie. Die Epipolarlinie entsteht dadurch, dass die Epipolarebene die beiden Bilder in jeweils einer Gerade schneidet, wobei die Epipolarebene, die von den beiden Projektionszentren der Kameras und der aufgenommene Objektpunkt P aufgespannte Ebene ist. Alle die Epipolarlinien eines Bildes laufen durch seinen Epipol, der der Punkt ist, wo die Gerade, die die beiden Projektionszentren der Kameras verbindet, die Bildebene durchstößt [Vgl. Wikipedia 2007, sowie Abb. 5.4].

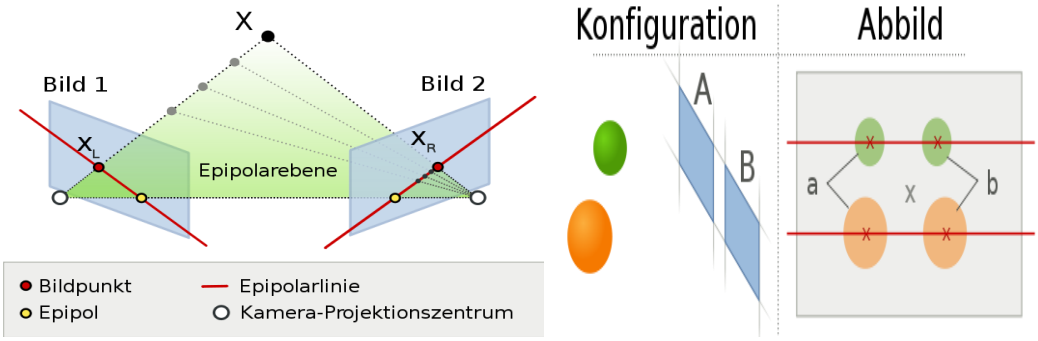

Abbildung 5.4: Schematische Darstellung der Epipolargeometrie (links) und für den Fall, dass die beiden Kameraachsen parallel zueinander ausgerichtet sind [Wikipedia 2007].

Da für diese Arbeit die beiden Kameraachsen exakt parallel zueinander ausgerichtet sind, verschieben sich die Epipole auf dem linken und rechten Bild ins Unendlich. In diesem Fall werden die Epipolarlinien exakt horizontal. Deshalb wir die Korrespondenzsuche lediglich entlang einer Pixelzeile durchgeführt [Vgl. Wikipedia 2007, sowie Abb. 5.4]. Verfahren zur Bestimmung der Disparität aus stereoskopischen Bildern befassen sich also in der ersten Linie mit der Lösung des Korrespondenzproblems, wobei diese Verfahren in zwei verschiedenen Typen geteilt werden [Vgl. Mallot 2000]:

- Korrespondenz lokalisierter Bildelemente: In dieser Methode werden zuerst Bildmerkmale wie Kanten und Linienenden extrahiert. Im zweiten Schritt wird das Korrespondenzproblem gelöst, in dem es mit Einschränkungen, Vorwissen, sowie Plausibilitätsbetrachtungen eine plausible Zuordnung der Bildelemente hergeleitet wird. Der Suchraum wird auf die Epipolarlinie begrenzt. Die Merkmaleigenschaften werden während der Zuordnung betrachtet. Die Ordnungsbedingungen, wie die Eindeutigkeit und die Vollständigkeit (eindeutige Zuordnung jedes Bildelements), sowie die die Reihenfolge werden auch während der Zuordnung betrachtet. Eine Grob-zu-fein Strategie wird ebenfalls angewendet. Im dritten Schritt wird mit einer Triangulation die mit der Lösung des Korrespondenzproblems gewonnene Disparitätkarte verwendet, um die Tiefekarte zu bestimmen. Im letzen Schritt wird die Oberflächerekonstruktion mit einer Interpolation durchgeführt.
- Intensitätsbasierte Verfahren: In solchen Verfahren wird eine stetige Disparitätkarte gerechnet, indem der Korrelations- oder der Phasenverschiebungssatz auf den beiden Halbbildern angewendet wird, wobei solche Verfahren das Korrespondenzproblem nicht lösen. Bei dem Korrelationssatz wird die Disparität als die relative Verschiebung der Grauwertverläufe beider Halbbilder definiert, wobei die Ähnlichkeit (quadratische Abweichung) der beiden Grauwertverläufe maximal ist. Bei dem

Phasenverschiebungssatz wird die Disparität, als die Phasendifferenz der Grauwertverläufe beider Halbbilder in Frequenzbereich berechnet.

Globale Verfahren werden eingesetzt, wo hochauflösende Tiefenkarte nicht erforderlich ist. Beim Ergreifen eines Objekts zum Beispiel, reicht für die Armbewegung die Wahrnehmung eines mittleren Tiefenwertes für das ganze Objekt aus. Tiefenvariationen der Oberfläche dieses Objekt sind eher für den eigentlichen Griff relevant. Der Korrelationssatz wird hier beispielweise mit einem sehr großen Fenster eingesetzt [Vgl. Mallot 2000].

5.2 Disparitätsbestimmung durch Auswertung von Phasenverschiebung in parallelen Pixelzeilen

Die phasenbasierende Methode bestimmt die Disparität als Phasendifferenz der im Frequenzbereich gefilterten Grauwertverläufe (Xl und Xr) der beiden Halbbildern. Für die Filterung im Frequenzbereich werden jeweils ein symmetrischer und ein antisymmetrischer Filterkern eingesetzt. Die Disparität ist dann bestimmt als der Phasendifferenz zwischen Xl und Xr geteilt durch die Ortsfrequenz ω [Vgl. Mallot 2000]. Die in dieser Arbeit verwendete Methode, stellt eine HW-kompatible Implementierung der phasenbasierenden Methode dar, indem Die Filterung im Frequenzbereich durch IIR-Filter realisiert wird [Vgl. Porr 2002, sowie Abb. 5.5].

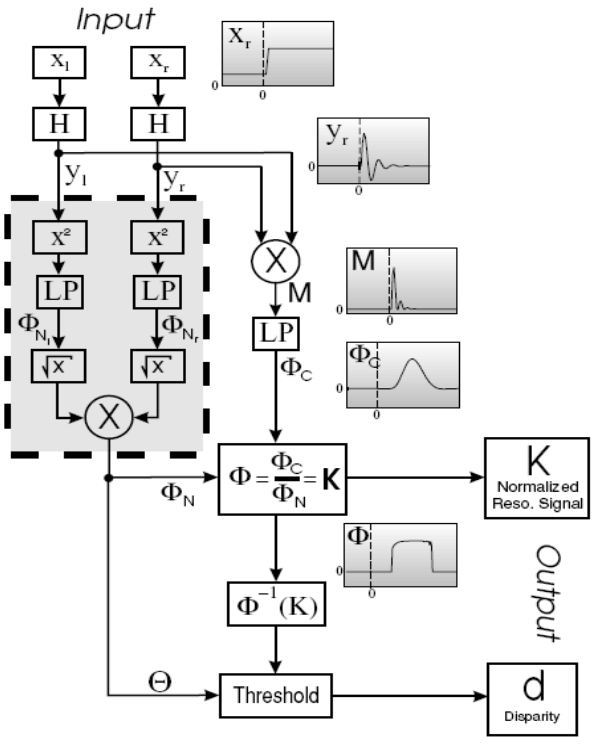

Abbildung 5.5: Schema der Methode zur Bestimmung der Disparität als Phasendifferenz [Porr 2002].

Seien x_l die Positionen eines Punktes in der Pixelzeile des linken Bildes und x_r die Position des korrespondieren Punktes in dem rechten Bild. Wenn t_l die Zeit ist, wo x_l von der linken Kamera verfügbar gestellt ist und t_r die Zeit, wo x_r von der rechten Kamera verfügbar gestellt, wobei die beiden Kameraachsen parallel zueinander ausgerichtet sind,

dann ist die Disparität d sowohl die Differenz zwischen x_l und x_r (Ortbereich), als auch zwischen t_l und t_l (Frequenzbereich) [Vgl. Porr 2002]. Der Algorithmus Bestimmung der Disparität im Frequenzbereich ist wie folgt geteilt [Vgl. Porr 2002, sowie Abb. 5.5]:

- Der Hauptpfad: Die zu x_l und x_r korrespondierenden Grauwerten Xl und Xr werden mit dem Bandpassfilter (H auf der Abb. 5.5) gefiltert. die gefilterten Werte Yl und Yr repräsentieren Sprünge bei Xl und Xr. Dann werden Yl und Yr multipliziert (M) und mit einem Tiefpassfilter gefiltert (LP auf der Abb. 5.5), was die Schwingungen in Yl und Yr eliminiert. Der resultierende Wert Φc muss bezüglich dem Grauwertlevel auf dem Bild normalisiert werden, um K zu bestimmen, was der umgekehrte Wert der Disparität ist. Die Disparität ist dann gültig
- Der Normalisierungspfad: Hier werden Yl und Yr quadriert, was M aus dem Hauptpfad überlappt. Dann werden die quadrierten Werte mit demselben Tiefpassfilter wie im Hauptpfad gefiltert, damit Schwingungen eliminiert werden. Nach der Filterung stehen zwei Normalisierungsfaktoren Φ_{N_l} und Φ_{N_r} für das linke Bild und das rechte Bild. Der gesamte Normalisierungsfaktor Φ_N ist dann der Mittelwert von den Wurzeln aus Φ_{N_l} und Φ_{N_r}. Die Disparität gilt als gültig genau dann, wenn Φ_N größer als die Konstante θ ist.

Die Beschreibungen der im Algorithmus verwendeten Tief- und Bandpassfilter sehen wie folgt aus, wobei diese für analoge Filter gilt [Vgl. Porr 2002]:

- Das Bandpassfilter ist ein Resonator mit der Übertragungsfunktion H(s) [Vgl. Gleichungen (5.1), (5.2), (5.3), (5.3), (5.5)], wobei $f_0$die Resonanzfrequenz und Q der Gütefaktor des Resonators sind.

$$H(s) = \frac{s}{(s-s_\infty)(s-s_\infty^*)} \tag{5.1}$$

$$s_\infty = Re(s_\infty) + jIm(s_\infty) \tag{5.2}$$

$$s_\infty^* = Re(s_\infty) - jIm(s_\infty) \tag{5.3}$$

$$Re(s_\infty) = {-\pi f_0}/{Q} \tag{5.4}$$

$$Im(s_\infty) = \pi f_0 \sqrt{1 - \frac{1}{Q^2}} \tag{5.5}$$

Die Beziehung zwischenf_0, Q und die Disparität d ist bei der Gleichung (5.6) gegeben.

$$0 \leq f_0 \leq \frac{1}{d\sqrt{4 - \frac{1}{Q^2}}} \tag{5.6}$$

- Das Tiefpassfilter muss eine genug größere Ordnung (N) haben, um die nicht konstanten Faktoren zu eliminieren. Die Phasenverschiebung zwischen den Signalen darf nicht vom Filter verändert werden sonst wird die abgeschätzte Disparität verfälscht. Deshalb ist empfohlen *Bessel*-Filter zu verwenden, das als Grenzfrequenz die Resonanzfrequenz des Tiefpassfilters hat. Darüber hinaus darf die Impulsantwort des Filters nicht negativ sein. Deshalb wird das digitale Filter als *Infinite Impulse Response* (IIR)-Filter implementiert. Das IIR-Filter berechnet den Eigangwert y(n) eines Signals

x(n) zu dem Zeitpunkt n mit der folgenden Gleichung, wobei N die Ordnung, a und b die Koeffizienten des Filters sind [Vgl. Schwarz 2007]:

$$y(n) = \sum_{k=1}^{N} b_k * x(n-k) - \sum_{k=1}^{N} a_k * y(n-k) \quad (5.7)$$

Ein Vorteil des IIR-Filters ist, dass es einen vorgegebenen Frequenzgang im Vergleich zu einem *Finite Impulse Response* (FIR)-Filter im HW mit weniger Multiplizieren, Addieren und Verzögerungselementen realisieren [Vgl. Schwarz 2007].

5.3 Matlab-Modell

Das Ziel dieses Modells ist es, mit Hilfe von Matlab-Funktionen aus den Übertragungsfunktionen der Bandpass- und Tiefpassfilter, die entsprechenden IIR-Filterkoeffizienten zu berechnen. Dafür werden zunächst die im Abschnitt 5.2 verwendeten Faktoren wie folgt gewählt [Vgl. Porr 2002]: $f_0 = 0.1\ pixel^{-1}$, Q = 1, θ = 128 und n = 4, wobei f_0 in Hz umgerechnet wird [Vgl. Gleichung (5.9)]. Angenommen wir haben eine Bildauflösung von 640x480 Pixel mit einer Framefrequenz von 25 Frame/s, dann ist $f_a$die Pixelfrequenz (Abtastfrequenz) und f_n die *Nyquist*-Frequenz [Vgl. Gleichung (5.8)].

$$f_a = 2 * f_n = 640 * 480 * 25 = 7.68MPixel * s^{-1} \quad (5.8)$$
$$f_0 = f_a * 0.1 = 768KHz \quad (5.9)$$

Die Übertragungsfunktion des Bandpassfilters H(s) wird ebenfalls geändert [Vgl. Wikipedia 2009b, sowie Gleichung (5.10), bis (5.14)], wobei B die Bandbreite, D der Dämpfungsgrad, ω_0 die Grenzkreisfrequenz, f_0 die Grenzfrequenz und Q der Gütefaktor des Filters ist.

$$H(s) = \frac{s * \frac{2D}{\omega_0}}{1 + s * \frac{2D}{\omega_0} + s^2 * \frac{1}{\omega_0^2}} \quad (5.10)$$

$$B = 2D\frac{\omega_0}{2\pi} \quad (5.11)$$

$$f_0 = \frac{\omega_0}{2\pi} \quad (5.12)$$

$$Q = \frac{f_0}{B} \quad (5.13)$$

$$D = \frac{1}{2Q} \quad (5.14)$$

Für das Tiefpassfilter wird die Übertragungsfunktion H(s) verwendet [Vgl. Schwarz 2007, sowie Gleichung (5.15)], wobei ω_0 die Grenzkreisfrequenz des Filters und D der Dämpfungsfaktor ist.

$$H(s) = \frac{\omega_0^2}{s^2 + 2 * D * s + \omega_0^2} \quad (5.15)$$

Die Faktoren a und b des digitalen Bandpassfilters (IIR-Filter) werden wie folgt berechnet [Vgl. Matworks 2008, sowie Anhang 9.3]: Mit der Matlab-Funktion *freqs* wird aus der Übertragungsfunktion des analogen Bandpassfilters [Vgl. Gleichung (5.10), bis (5.14)] die komplexe Frequenzantwort (Magnitude und Kreisfrequenz) dieses analogen

Bandpassfilters berechnet. Für eine Magnitude von 0.707 ($\sqrt{1/2}$), werden die zwei entsprechenden *Cutoff*-Kreisfrequenzen gelesen und mit der Teilung durch $\pi * f_a$ normalisiert. Die zwei normalisierten *Cutoff*-Kreisfrequenzen werden als Parameter in die Matlab-Funktion *butter* eingegeben, die damit die Koeffizienten a und b des korrespondierenden IIR-Filters [Vgl. Gleichung (5.7)] bestimmt.

Für die Berechnung der Faktoren a und b des digitalen Tiefpassfilters als IIR-Filter wird anders als bei dem Tiefpassfilter vorgegangen. Der Grund dafür ist, dass Matlab keine Funktion zur Verfügung stellt, die aus der Übertragungsfunktion eines analogen *Bessel*-Filters ein digitales IIR-Filter erzeugt. Deshalb werden die Matlab-Funktionen *besselap*, *zp2ss* sowie *lp2lp* verwendet, um das analoge *Bessel*-Filter zu generieren. Einschließend werden die Funktionen *bilinear* und *ss2tf* verwendet, um die Faktoren a und b zu berechnen [Vgl. Angermann 2007].

Der Im Abschnitt 5.2 beschriebene Algorithmus wird dann implementiert, indem die Berechnung sukzessiv durchgeführt wird. Die beiden Halbbilder werden Zeilenweise gefiltert, wobei für Filterung mit der Matlab-Funktion *filtfilt* und den entsprechenden Koeffizienten des IIR-Filters durchgeführt wird. Die Abbildung 5.6 zeigt das linke Halbbild des Pentagon, sowie die abgeschätzte Disparitätkarte für den Fall, dass die Bandpass- und Tiefpassfilter die gleiche Grenzfrequenz haben. Auf der Disparitätskarte entspricht die blaue Farbe kleinen, grüne Farbe mittleren und rote Farbe größeren Disparitäten. Die Tiefenkarte wird aus der Disparitätkarte durch Triangulation hergeleitet [Vgl. Gleichung (5.1)].

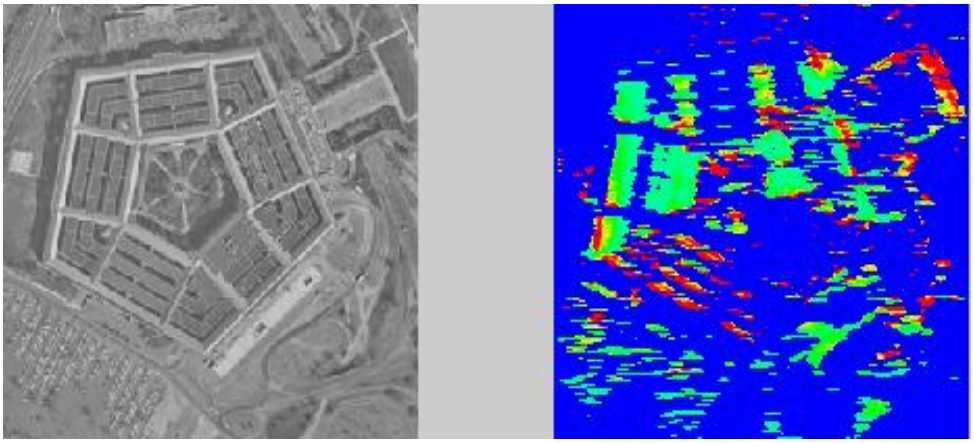

Abbildung 5.6: linkes Halbbild des Pentagon (links), sowie seine abgeschätzte Disparitätskarte (rechts).

6 Zusammenfassung

Das Ziel dieser Arbeit war es, eine FPGA-basierende SoC-Bildverarbeitungsplattform zu entwickeln, die sukzessive zu einem Antikollisionssystem durch Bildtiefenauswertung für Modellfahrzeuge im Rahmen des Projekts Fahrerlose Autonome Transportsystem (FAUST) weiterentwickelt wird.

Auf dem in das FPGA eingebetteten MicroBlaze µProzessorsystem wurde ein SW-Modul zur Parametrierung des ADV7183B Controllers implementiert. Dieses Softwaremodul kommuniziert mit dem HW-Modul, das zur Initialisierung des Controllers implementiert wurde. Nachdem die Initialisierung des Controllers abgeschlossen ist, wird der Controller parametriert, indem das SW-Modul dem Controller über ein IIC-Interface die Kommandos zur korrekte Digitalisierung des analogen S-Video Signals aus der Kamera sendet.

Der 16 Bit ITU-R BT.656 YCbCr 4:2:2 *Interlaced*-Videostrom aus dem VDEC1-Board wird mit einer Kette HW-Module in 24 Bit RGB-Format umgewandelt. Die Synchronisationssignalen FVAL und LVAL werden mit Hilfe der vom VDEC1-Board zu dem Videostrom mitgelieferten Synchronisationspulse VS und HS generiert. Mit dem Chroma Upsampling wird das 16 Bit YCbCr 4:2:2-Format in 24 Bit YCbCr 4:4:4-Format, welches anschließend in das 24 Bit RGB-Format umgewandelt wird.

Zur Visualisierung des Videostroms auf einem TFT-Monitor wurde das HW-Modul zur Konfigurierung des VGA-Controllers entwickelt. Dafür wird zunächst ein *Deinterlaced*-Verfahren implementiert, um den *Interlaced*-Videostrom in *Progressive*-Videostrom umzuwandeln. Dabei werden die Pixelzeilen verdoppelt und die vertikale Auflösung des Bildes wird verbessert. Anschließend wird der VGA-Controller entsprechend parametriert und das 24 Bit RGB Videosignal wird auf 8 Bit RGB reduziert. Die Konturen des auf dem TFT-Monitor angezeigten Bildes lassen sich deutlich erkennen, wobei die Farbenmischung verfälscht wird, sodass Rauschen auf dem Bild entstehen. Mit dem Testbildgenartor sind die Farbestreifen auf dem Bildschirm korrekt dargestellt.
Die Analyse einer auf Phasenverschiebung basierten Methode zur Tiefenabschätzung in stereoskopisch gewonnen Bildern wurde gemacht. Diese Methode berechnet die Disparität als Phasendifferenz in den beiden Halbbildern, indem die beiden Halbbilder in Frequenzbereich sukzessiv gefiltert werden. Der Einsatz von IIR-Filtern ist ein Vorteil für HW-Implementierung dieser Methode. Um Diese Methode zu testen wurde ein Matlab-Modell entwickelt und das Ergebnis wurde dargestellt.

7 Literaturverzeichnis

[HAW-FAUST 2009] FAUST-Projekt an der HAW-Hamburg: *Homepage des FAUST-Projekts an der HAW Hamburg*, 2009, URL: http://www.informatik.haw-hamburg.de/faust.html.

[Bagni and al. 2008] BAGNI, Daniel, MARZOTTO, Roberto, ZORATTI, Paul: *Building Automotive Driver Assistance System Algorithms with Xilinx FPGA Platforms*, Xilinx Xcell Journal No. 66 4th quarter 08.

[Porr and al. 2002] PORR, Bernd, NÜRENBERG, Bernd, WÖRGÖTTER, Florentin: *A VLSI-Compatible Computer Vision Algorithm for Stereoscopic Depth Analysis in Real-Time***.** International Journal of Computer Vision, July 13, 2002.

[Sony 2005b] SONY: *Circuitboard for FCB-IX Cameras***,** Datenblatt, 2005

[Sony 2004] SONY: *Colour Camera Module: Technical Manual*, Datenblatt, 2004.

[Sony 2005a] SONY: *Colour Block Camera*, Datenblatt, 2005.

[Jack 2005] JACK, Keith*: Video Demystified: A Handbook for the Digital Engineer*, Fourth Ed. Elsevier Science & Technology Books, 2005.

[Peters 2009] PETERS, Falko: *FPGA-basierte Bildverarbeitungspipeline zur Fahrspurerkennung eines autonomen Fahrzeugs*, HAW-Hamburg, Bachelorarbeit, 2009.

[Digilent 2005a] DIGILENT: *Digilent Video Decoder Board (VDEC1) Reference Manual*, Datenblatt , 2008.

[ANALOG-DEVICES 2005] ANALOG DEVICES: *Multiformat SDTV Video Decoder: ADV7183B,* Datenblatt, 2005.

[Digilent 2008] DIGILENT: *Digilent Nexys2 Board Reference Manual,* Datenblatt, 2008.

[Xilinx 2008a] XILINX: *Spartan-3E FPGA Family: Complete Data Sheet,* Datenblatt, 2008.

[Xilinx 2008b] XILINX: *MicroBlaze Processor Reference Guide*, Datenblatt, 2008.

[Poynton 2005] [Poynton 2005] POYNTON, Charles: *Converting YCbCr to RGB*, April 2005, – URL http://www.poynton.com/notes/short_subjects/video/YCbCr_to_RGB.

[Xilinx 2009] XILINX: *EDK Concepts, Tools, and Techniques: A Hands-On Guide to Effective Embedded System Design,* Datenblatt, 2009.

[Philips 2000] Philips Semiconductors**:** *THE I2C-BUS SPECIFICATION*, Datenblatt,

VERSION 2.1, JANUARY 2000.

[Digilent 2005b] DIGILENT: *Digilent Video Decoder Board: Schema*, Schema, 2005.

[Schwarz 2008] SCHWARZ, Bernd, REICHARDT, Jürgen: *WP Hardware-, Software-Codesign*, HAW-Hamburg, WS08/09.

[Xilinx 2008c] XILINX: *XPS IIC Bus Interface (v2.00a): Product Specification*, Datenblatt, July 22, 2008.

[Wikipedia 2009a]: WIKIPEDIA: *YCbCr-Farbmodell*, 2009, -URL: http://de.wikipedia.org/wiki/YCbCr-Farbmodell.

[Masrani 2006] MASRANI, Divyang K., MACLEAN, W. James: *Expanding Disparity Range in an FPGA Stereo System While Keeping Resource Utilization Low*, 2006.

[Zang 2009] ZANG, Christoph: Facharbeit: dreidimensionales Sehen, 2009, - URL: http://www.christoph-zang.de/Facharbeit.32.0.html.

[Mallot 2000] MALLOT, Hanspeter A.: *Sehen und die Verarbeitung visueller Information (Taschenbuch)*, Vieweg Verlagsgesellschaft; Auflage: 2, 2000.

[Hardieß 2007**]** HARDIEß, Gregor: *STEREOPSIS Beim Menschen*, Universität Tübingen, WS2007/08, - URL: http://www.uni-tuebingen.de/cog/teaching/ws2007_08/Cog.Neuro/Chap_4_Stereopsis.pdf.

[Klinker 2004]: KLINKER, Gudrun: Augmented Reality II- Camera Calibration-, May 11, 2004, - URL: http://campar.in.tum.de/twiki/pub/Far/AugmentedRealityIISoSe2004/L3-CamCalib.pdf

[Hartley 2000] HARTLEY, Richard, ZISSEMAN, Andrew: *Multiple View Geometry in Computer Vision*, Cambridge University Press, (Section 5, pp. 139-161), 2000.

[Wikipedia 2007] WIKIPEDIA: *Epipolargeometrie*, August 2007, - URL: http://de.wikipedia.org/wiki/Epipolargeometrie.

[Schwarz 2007] SCHWARZ, Bernd, REICHARDT, Jürgen: *VHDL-Synthese: Entwurf digitaler Schaltungen und Systeme*, Oldenburg Wissenschaftsverlag GmbH, 4. Auflage, 2007.

[Wikipedia 2009b] WIKIPEDIA: *Bandpass*, Juli 2009, - URL: http://de.wikipedia.org/wiki/Bandpass.

[Matworks 2008] THE MATWORKS: Matworks Help, 2009, - URL: http://www.mathworks.com/access/helpdesk/help/toolbox/signal/index.html?/access/helpdesk/help/toolbox.

[Angermann 2007] ANGERMANN, Anne, BEUSCHEL, Michael, RAU, Martin: *MATLAB – Simulink – Stateflow: Grundlagen, Toolboxen, Beispiele*, Oldenburg, 5. Auflage, 2007.

8 Glossar

TFT*: Thin-film transistor.*
FPGA*: Field Programmable Gate Array.*
DSP*: digital signal processor.*
PC*: Personal Computer.*
TTL*: Transistor-Transistor-Logic.*
BNC*: Transistor-Transistor-Logic.*
DIP*: direct current.*
FFC*: Flat Flexible Cable.*
CVBS*: Colour Video Baseband Signal.*
PAL*: Phase Alternating Line.*
SECAM*: Séquentiel Couleur a Mémoire.*
NTSC*: National Television Systems Committee.*
SRAM*: Static Random Access Memory.*
USB*: Universal Serial Bus.*
LED*: Light Emitting Diode.*
JTAG*: Joint Test Action Group.*
RAM*: Random Access Memory.*
RISC*: Reduced Instruction Set Computing.*
VHDL*: Very High Speed Integrated Circuit Hardware Description Language.*

9 Anhang

9.1 CD: Implementierung in VHDL mit ISE und EDK.

9.2 CD: Bericht der Arbeit als PDF-Datei.

9.3 CD: Matlab-Modell.

Versicherung über Selbstständigkeit

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

Hildesheim, 01.09.09