Power and productivity for a better world™

DHBW Duale Hochschule Baden-Württemberg

Inbetriebnahme einer Camera Link Schnittstelle für das ZedBoard

STUDIENARBEIT

des Studienganges Elektrotechnik

an der Dualen Hochschule Baden-Württemberg Mannheim

von

Nils Lindenthal

20.03.2015

Bearbeitungszeitraum	10 Wochen
Matrikelnummer, Kurs	4093957, TEL12AAT
Ausbildungsfirma	ABB Automation GmbH, Mannheim
Betreuer der Dualen Hochschule	Prof. Dr. Rüdiger Heintz

Ehrenwörtliche Erklärung

gemäß § 5 (3) der „Studien- und Prüfungsordnung DHBW Technik“ vom 22. September 2011.

Ich habe die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet.

---------- ----------

Ort Datum Unterschrift

Überblick Tätigkeiten der Studienarbeit

Diese Studienarbeit entstand während des 6. Semesters an der DHBW-Mannheim. Ziel der Arbeit ist es, eine funktionierende Verbindung zwischen einem ZedBoard und einer Kamera mit Camera Link Schnittstelle herzustellen. In vorrangegangenen Studienarbeiten wurde dafür bereits eine Adapterplatine zwischen FMC und Camera Link entwickelt. Diese war zu Beginn noch nicht vollständig funktionsfähig. Daher sind zunächst die Fehlerquellen durch Messen und Analysieren der Signale gesucht und dann behoben wurden. Dadurch existiert nun ein funktionsfähiger Adapter zwischen FMC und Camera Link. Als nächstes ist das FPGA und die ARM CPU auf dem ZedBoard so konfiguriert und programmiert worden, dass es möglich ist, Signale an die Kamera zu senden bzw. von ihr zu empfangen. Dazu ist zum einen untersucht worden, wie CPU und FPGA Daten austauschen können und wie diese kodiert sein müssen, damit das gewünschte Verhalten erreicht wird. Für die Kommunikation zwischen CPU und FPGA ist eine Schnittstelle über DMA bzw. AXI4 hergestellt worden. Die Kodierung der Daten ergibt sich aus dem Datenblatt der Kamera.
Daher ist es mit Ende dieser Arbeit möglich, auf Befehl Daten von der Kamera zu empfangen und diese sich ausgeben zu lassen.

Inhaltsverzeichnis

Abkürzungsverzeichnis

Abkürzung	Bedeutung
AIA	Automated Imaging Association
AMBA	Advanced Microcontroller Bus Architecture
AXI	Advanced eXtensible Interface
CL	Camera Link
EMV	Elektromagnetische Verträglichkeit
FMC	FPGA Mezzanine Card
FPGA	Field programmable gate array
IC	Integrated circuit
LSB	Least signifcant bit
LUT	Lookup Table / Tabelle
LVDS	Low Voltage Diffentail Signaling
MSB	Most significant bit
SoC	System on a Chip
UIODevice	Universal Input Output Device

Abbildungsverzeichnis

Tabellenverzeichnis

Formelgrößen und Einheiten

Größe	Einheit	Bedeutung
I	Ampere	Stromstärke
U	Volt	Spannung
W	Watt	Leistung
t	Sekunde	Zeit
R	Ohm	Widerstand
D	Bit/s	Datenrate

1 Freifall-Sortiereinheit

Eine Freifall-Sortiereinheit ist eine Anlage, die Granulat oder Materialien mit ähnlichem Volumen und Fallverhalten sortiert. Meist wird sie eingesetzt, um Nutz- von Störgut zu trennen. Ein Einsatzgebiet ist z.B. bei der Getreideernte. Hier müssen Getreidekörner von Fremdkörpern, wie Steinchen getrennt werden. Auch in der Kunststoffaufbereitung werden sie eingesetzt, um verschiedenfarbige Kunststoffstücke voneinander zu trennen. Abb. 1-1 zeigt den prinzipiellen Aufbau einer solchen Sortiereinheit. Über ein Fördersystem werden die zu sortierenden Partikel in das System eingeführt. Sie fallen an einer Kamera vorbei, die die Farbe und Position der Partikel erfasst. Optional kann die Präzision durch ein zusätzliches Lasersystem erhöht werden. Ein (hier nicht dargestellter) Controller erfasst den Input und berechnet die Ausgangswerte der Ventile, die über Druckluft die Partikel ausblasen. Um eine hohe Präzision zu erreichen, muss eine möglichst kurze Zeit zwischen Aufnahme mit der Kamera und Reaktion der Ventile, sowie eine hohe Framerate der Kamera sichergestellt werden.

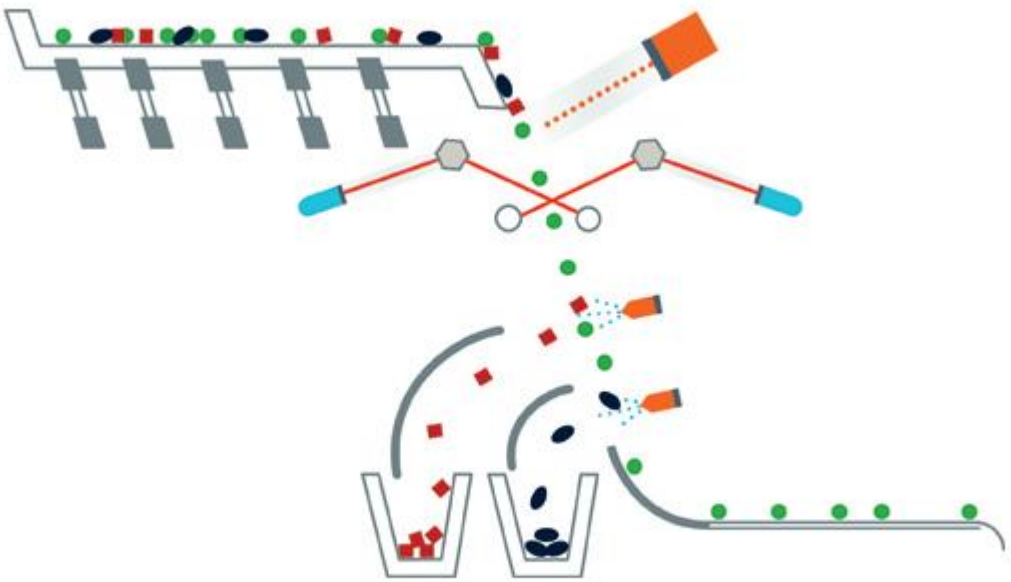


Abb. 1-1: Allgemeiner Aufbau einer Freifall-Sortiereinheit für Kunststoffpartikel
http://www.bestsorting.com/wp-content/uploads/2012/09/nimbus-wp-large.jpg

Durch eine Reihe von Studienarbeiten soll eine solche Anlage an der DHBW-Mannheim entstehen. Es gibt Arbeiten zu der Anbindung der Ventile an das Steuersystem, zu der Programmierung jenes und zur der Anbindung der Kamera. Diese Arbeit dreht sich um die Anbindung der Kamera. Als Kamera kommt eine Zeilenkamera vom Typ AViiVA SC2 CL von e2v [1] zum Einsatz. Sie besitzt eine Camera Link (kurz CL) Schnittstelle. Als Steuersystem kommt ein ZedBoard von Avnet und Digilent [2][3]. Es besitzt ein FPGA[1] und einen ARM CPU. Durch das FPGA kann das Board parallele Ausgaben sehr schnell bewältigen, [4] während die ARM CPU es ermöglicht, auf Programme und Bibliotheken in den verbreiteten Hochsprachen wie C, C++ oder Java zurückzugreifen. Auch kann ein (Linux-) Betriebssystem installiert werden. Für die Anbindung von Peripherie steht eine Reihe von Schnittstellen zur Verfügung. Ziel dieser Arbeit ist eine Verbindung zwischen Kamera und der Software auf der ARM CPU. Dazu muss zunächst eine Verbindung zwischen der Camera Link Schnittstelle der Kamera und dem FMC-Stecker des ZedBoardes hergestellt werden. Danach muss das FPGA und die Software so konfiguriert und programmiert werden, dass die Kommunikation hergestellt werden kann. Bereits fertige Adapterplatinen zwischen FMC und Camera Link der Firmen Toyon mit dem *BOCCACCIO FMC CAMERA LINK* [5], Alpha Data mit der *FMC-CAMERALINK* [6] Platine sowie Integre Technologies mit dem FMC-200 [7] Adapter wurden aufgrund des hohen Preises (500 $ (Toyon) bis 1.400 $ (Integre Technologies)) nicht weiter in Betracht gezogen. Stattdessen wird ein fast fertiger Adapter aus vergangen Arbeiten weiterentwickelt. Jener hat noch Funktionsstörungen, die zunächst behoben werden müssen.

[1] Field Programmable Gate Array.

2 Schnelleinstieg und Bedienung

Dieses Kapitel dient zum Schnelleinstieg in den aktuellen Status des Projektes. Es erklärt wie die Hardware verbunden werden muss, sowie wie sie bedient wird ohne auf die Funktionsweise einzugehen.

2.1 Hardwareaufbau

Die Kamera ist aktuell so eingestellt, das Aufnahmezeitpunkt und Belichtungszeit extern gesteuert werden. Die Paramater lauten wie folgt:

- H=6 (60 MHz Modus)
- S=0 (Paralleler RGB Modus)
- T=0 (Rohbild)
- M=3 (externer Trigger und Belichtungszeitvorgabe über eine Leitung)

Um diese Parameter zu setzen, ist die Kamera mit dem bereitgestellten PC zu verbinden und die o.g. Parameter über die clshell.exe einzugeben. Anschließend sind die Parameter über den Befehl „–C=1" zu speichern. Näheres zur Einrichtung ist unter [8][2] zu finden.

Zur Benutzung mit dem ZedBoard ist die Kamera mit dem Eingang des Adapters zu verbinden. Ist der Kühlkörper oben und zeigt die Platine in Richtung Benutzer, so ist dies der rechte Stecker. Zusätzlich ist das Board mit dem Netzteil mit Spannung zu versorgen. Auch ist die aktuelle *boot.bin* auf die Boot-Partition der SD-Karte zu kopieren.

[2] PDF auf DVD

2.2 Ohne PC

Nun lässt sich die Kamera in Betrieb nehmen. Dazu wird das ZedBoard eingeschaltet und die beiden Schalter 0 und 1 (wenn die Schalter zum Benutzer zeigen sind dies die beiden rechten) auf EIN geschaltet. Schalter 0 schaltet die Anzeige über die LEDs ein. Diese geben die Helligkeit jeder RGB-Gruppe aus. Dazu werden die drei Farbkanäle addiert und durch drei geteilt. Die 8 Bit breiten Daten werden auf sieben der acht LEDs gegeben. Jede LED repräsentiert so ein Bit; das höchste Bit wird nicht ausgegeben. Diese LED wird für andere Zwecke genutzt. Da die Ausgabe für jede RGB-Gruppe ohne Mittelwertbildung geschieht, ist ein Blinkmuster zu erkennen. Je heller das Bild ist desto, heller leuchten die LEDs, die die hochwertigen Bits repräsentieren.
Hat eine RGB-Gruppe z.B. den Wert 48, so leuchten LED 4 und 5. Hat die nächste Gruppe den Wert 32, so leuchtet nur LED 5. Aufgrund der hohen Geschwindigkeit erscheint für den Betrachter LED 5 daher heller als LED 4.
Schalter 1 schaltet zwischen Software und Hardware gesteuertem Trigger-Signal um. Ist der Schalter auf EIN, wird ein Signal ausgegeben, das für 18.000 Takte (bei 60 MHz sind dies 300µs) die Kamera belichten lässt und dann für 2^{16} Takte wartet. In dieser Zeit sendet die Kamera das Bild. Ist das Board in diesem Modus, leuchtet die LED 7.
Das Camera Link Kabel hat im Stecker zur Kamera einen Wackelkontakt bzw. Kabelbruch. Sollte das angezeigte Ergebnis nicht zu den aktuellen Bedingungen passen, so kann durch eine Änderung des Neigungswinkels wieder ein sauberer Kontakt hergestellt werden. Eine Neigung von rund 40° bis 50° noch oben zeigt meist gute Ergebnisse.
Der Aufbau des ZedBoard ist in Abb. 2-2 zu sehen.

2.3 Mit PC

Soll das System über die Software gesteuert werden, so ist das Board über die Ethernet Schnittstelle mit einem PC zu verbinden. Das ZedBoard hat die feste IP 192.168.137.80 mit der Subnetzmaske 255.255.255.0. Der Netzwerkadapter im PC muss sich im gleichen Subnetz befinden. Soll die VM, mit der gearbeitet wird, zusätzlich ins Internet, so ist eine zweite Netzwerksschnittstelle mit Internetzugang nötig. In dieser muss eingerichtet sein, dass der Internetzugang mit der zweiten Netzwerkschnittstelle geteilt wird. Dies wird in den Eigenschaften des Netzwerkadapters unter Freigabe eingerichtet.
Zum Zugriff auf das ZedBoard über Ethernet muss das Linux auf dem Board gestartet werden. Geschieht dies nicht automatisch, ist das Board über die UART-Verbindung über ein USB zu Mini-USB Kabel mit dem PC zu verbinden. Über diese Schnittstelle kann mit dem Tool „SerialConnect“ Befehle an das Board gesendet werden. Der Befehl „boot“ startet das Betriebssystem. Ggf. muss dies öfters geschehen, da die Kommunikation über die UART Schnittstelle nicht einwandfrei funktioniert.
Nun kann Eclipse gestartet werden. Über den Punkt „DataTransfer Debug“ (vgl Abb. 2-1) wird das Projekt kompiliert, auf das ZedBoard geladen und anschließend ausgeführt. Das verlange Passwort lautet „root“.

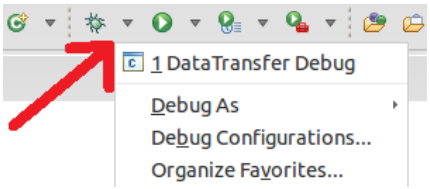


Abb. 2-1: Start des DataTRansfer Debug in Eclipse

In der letzten Zeile (return 0) ist ein Breakpoint zu setzen, um die Ausgabe in der Konsole lesen zu können. Läuft das Programm erfolgreich durch, so wird jede 16. RGB-Gruppe in Hex Darstellung ausgegeben.
Der Aufbau des ZedBoardes ist in Abb. 2-3 zu sehen.

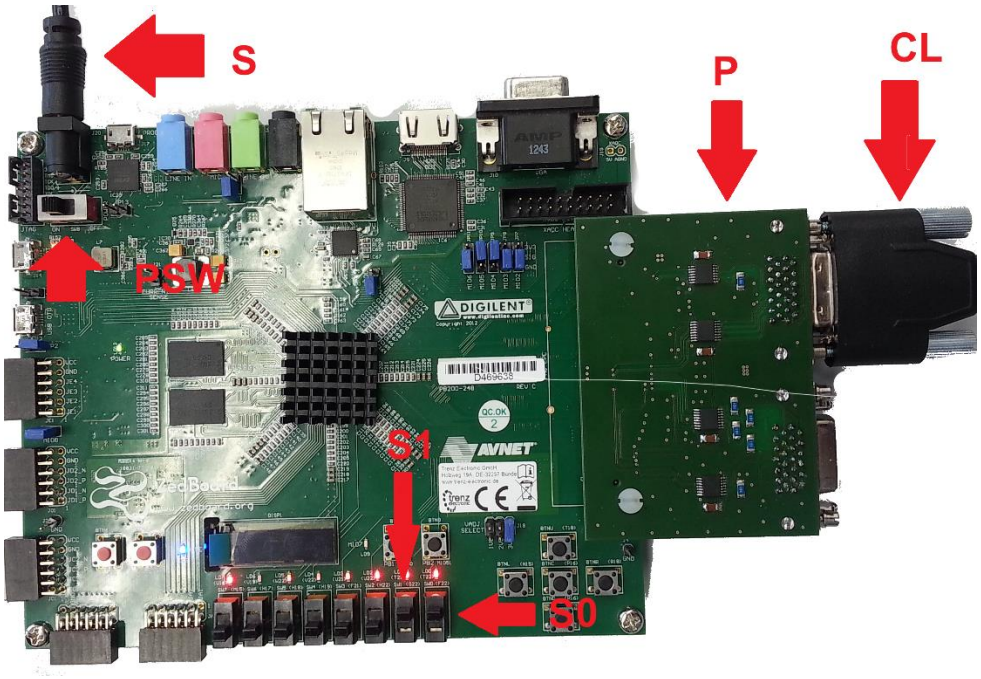


Abb. 2-2: ZedBoard im Betrieb ohne PC. Angeschlossen sind Strom (S) und Camera Link (CL) über die Adapterplatine (P). Zusätzlich sind gezeigt, der PowerSwitch (PSW) und die Schalter S0 und S1. Zu erkennen sind die unterschiedlich hellen LEDs

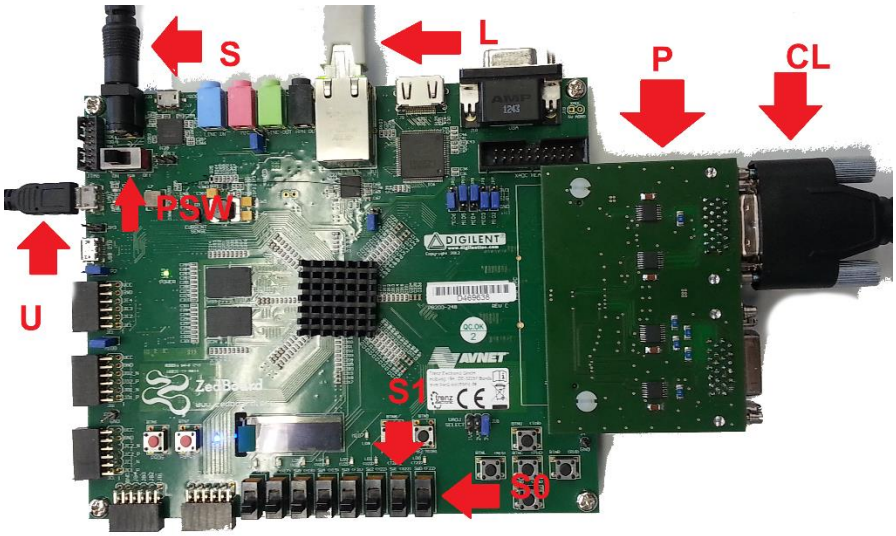


Abb. 2-3: ZedBoard im Betrieb mit PC. Angeschlossen sind Strom (S) und Camera Link CL) über die Adapterplatine P, sowie UART (U) und Ethernet (L). Zusätzlich sind gezeigt, der PowerSwitch (PSW) und die Schalter S0 und S1

3 Camera Link und Low Voltage Differential Signaling

Die in dem Projekt verwendete Kamera verfügt über eine Camera Link Schnittstelle. Camera Link ist ein von der Automated Imaging Association (AIA) verwaltete synchrone Schnittstelle zur Übertragung von Bilddaten. Sie ist dabei auf hohe Übertragungsraten ausgelegt. [9][10] Durch eine immer weitere Verbreitung von der optischen Erkennung in der Automation stiegen auch die Anforderungen an die Datenrate und Geschwindigkeit der Kamera. Nach einer Reihe von Schnittstellen verschiedener Hersteller sollte Camera Link eine herstellerübergreifende Möglichkeit bieten, Kameras mit dem System zu verbinden. Zur Übertragung wird auf mehrere parallele Adernpaare gesetzt, die über Low Voltage Diffential Signaling (kurz LVDS) Daten senden und empfangen. Camera link basiert auf dem Channel-Link Interface von National Semiconductor. Deshalb kann für Camera Link auch auf Quellen über Channel-Link zurückgegriffen werden um die Funktionsweise von Camera Link zu belegen [11].

3.1 Low Voltage Differential Signaling

LVDS ist eine nach ANSI/TIA/EIA-644-1995 standarisierte Schnittstelle zur Übertragung serieller Daten. Sie definiert die Übertragung auf Schicht eins im ISO/OSI Modell und geht daher nicht auf Aspekte wie die Datensicherung ein. Wie der Name schon andeutet, setzt LVDS anstatt der üblich 3,3 bzw. 5 Volt [12] auf Spannungen um 1,2 Volt. Dadurch können sich Leitungskapazitäten nicht so stark aufladen, wodurch höhere Frequenzen ermöglicht werden. Das Signal wird als Differenz zwischen den beiden Leiterpaaren übertragen. Abb. 3-1 zeigt den Aufbau eines LVDS Leiterpaares. Zur Übertragung wird ein 35 mA großer Strom vom Sender über die Leitung geschickt. Im Empfänger sitzt ein 100 Ohm Widerstand. Nach dem Ohm’schen Gesetz fällt über ihn eine Spannung von 350 mV ab, die vom Empfänger gemessen wird.

Abb. 3-1: Aufbau einer LVDS Verbindung mit Sender, Empfänger und 100 Ohm Widerstand „Lvds interface de“ von Lueggy - Eigenes Werk. Lizenziert unter CC0 über Wikimedia Commons - https://commons.wikimedia.org/wiki/File:Lvds_interface_de.svg#mediaviewer/File:Lvds_interface_de.svg

Die Polung der Spannung und somit die Richtung des Stromes bestimmen den logischen Zustand. Zwischen zwei logischen Zuständen befinden sich also 700 mV. Da für diesen Wert nur die Differenz zwischen den Leitern von Bedeutung ist, wirken sich Gleichtaktstörungen, die beide Leiter gleichermaßen beeinflussen, nicht auf das Signal aus, sofern sie unter 1 V liegen[12]. Durch eine Verdrillung und/oder Schirmung des Leiternpaares kann die Störunempfindlichkeit noch weiter gesteigert werden. [9] So sind Datenraten von mehreren hundert Mbit/s bis zu GBit/s möglich. Die vergleichbare RS-422 Schnittstelle, die mit ±6V arbeitet, leistet hier nur bis zu 10 Mbit/s. Durch die niedrige Spannung sowie die Verwendung eines festen Stroms liegt die Leistungsaufnahme bei rund 8,75 mW während RS-422 hier 90 mW benötigt [12].
Durch die niedrigen Spannungen ist ein LVDS System empfindlich gegenüber Störungen. Daher ist bei der Auslegung und Entwicklung einer solchen Schaltung auf die elektromagnetische Verträglichkeit (EMV) zu achten.

3.2 Aufbau Camera Link

Camera Link nutzt mehrere solcher LVDS Verbindungen zur Übertragung von Daten. Auf der Senderseite erfasst ein Framepusher die parallel eingehenden Bilddaten und sendet sie. Auf der Empfängerseite dekodiert der Framegrabber das eingehende Signal und gibt die Daten parallel wieder aus [9][10][13].

Camera Link unterscheidet drei Varianten**:**

- Base: pro Takt werden 24 Bit Bilddaten übertragen
- Medium: pro Takt werden 48 Bit Bilddaten übertragen
- Full: pro Takt werden 72 Bit Bilddaten übertragen

Die Daten werden dabei 7:1 serialisiert. Dies bedeutet, dass pro Eingangstakt sieben Eingangsbits über eine LVDS Leitung geschickt werden. Abb. 3-2 zeigt den Ablauf der Übertragung auf einer Leitung.

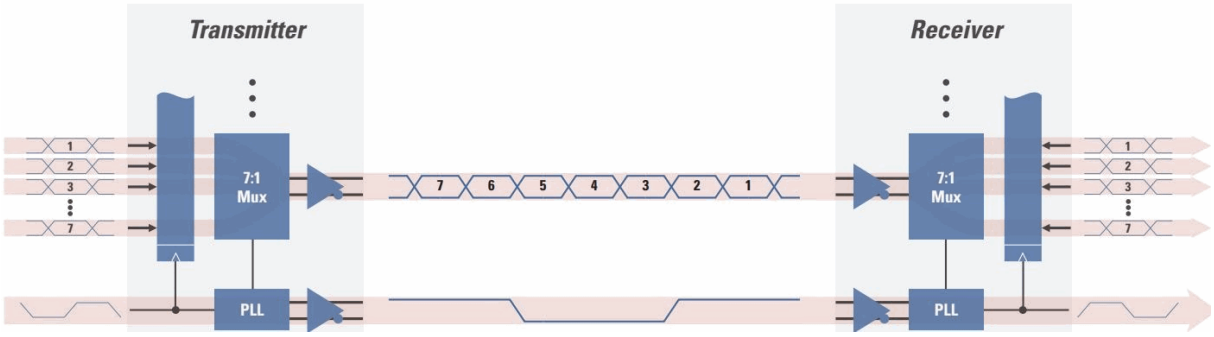


Abb. 3-2: Serielle Übertragung von 7 Bits über eine LVDS Leitung. Dargestellt ist eine LVDS Leitung sowie der Takt
National Semiconductor Corporation (2006): Channel Link Design Guide

Die sieben Eingangsbits werden über ein Zeitmultiplexer mit dem MSB zuerst übertragen. Parallel wird der Takt mit übertragen, der nach Bit 2 abfällt und bei Bit 6 wieder ansteigt. Auf der Empfängerseite wird der Bitstrom von einem Demultiplexer wieder zu sieben parallelen Signalen umgesetzt. Da die Base-Variante 24 Bit Nutzdaten überträgt, werden also vier LVDS Leitungen benötigt. Die vier verbleibenden vier Bits sind:

- Line-Sync
- Frame-Sync
- Data-Valid
- Spare Bit – nicht spezifiziert

Ihr Verlauf ist in Abb. 3-3 dargestellt.

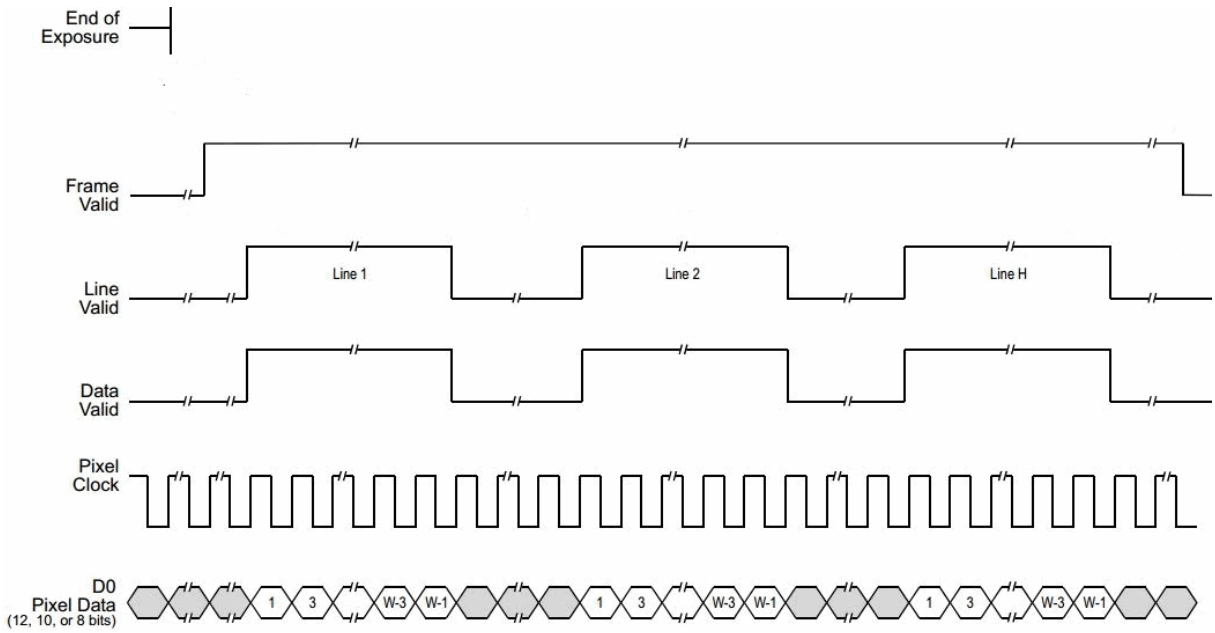


Abb. 3-3: Verlauf und Bedeutung der drei Valid-Steuerleitungen.
Basler (2011): Camera Link information for framegrabber designers, Document Number AW000990, Version 01

Sie werden genutzt, um Nutzdaten aus dem eingehenden Datenstrom herauszufiltern. Nur wenn alle drei Valid-Bits auf HIGH sind, werden die übertragenen Daten als Nutzdaten interpretiert. Wenn der Frame fertig aufgenommen wurde, wird das Frame-Valid-Bit auf HIGH gesetzt und bleibt in diesem Zustand bis der Frame fertig übertragen wurde. Das Bit bleibt nun solange auf LOW, bis ein neuer Frame bereit steht. Es zeigt also an, das aktuell ein Frame übertragen wird. Der Empfänger kann so die Grenze zwischen zwei Frames erkennen. Bei speziellen Anwendungen, wie z.B. bei einer Zeilenkamera kann auch auf das Framevalid Bit verzichtet werden. Das Line-Valid-Bit ist dann HIGH, wenn eine Bildzeile übertragen wird. Zwischen jeder Zeile wird eine Pause von mehreren Takten eingeführt, damit der Empfänger zwischen zwei Zeilen unterscheiden kann. Das Data-Valid-Bit entscheidet nun für jeden Takt, ob Nutzdaten vorliegen. In Abb. 3-3 ist das Dat-Valid-Bit gleich dem Line-Valid-Bit, da hier eine Zeile ohne Unterbrechung übertragen wird. Die Verwendung des Spare-Bit ist nicht näher festgelegt. Die Kamera kann darüber z.B. signalisieren, dass sie gerade bereit ist

oder gerade ein Frame aufnimmt. Es kann auch nicht genutzt werden. Abb. 3-4 zeigt alle LVDS Verbindungen bei einer Base Camera link Verbindung.

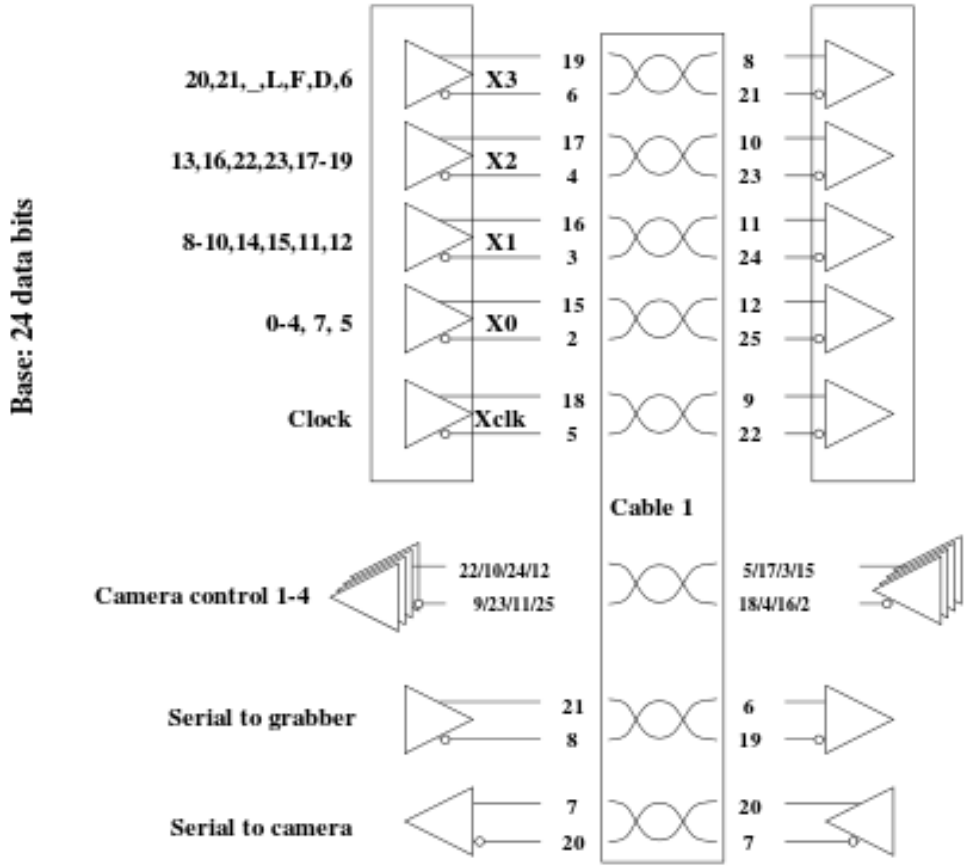


Abb. 3-4: Alle LVDS Verbindungen einer base Camera link Verbindung mit vier Daten, einer Takt-, vier steuer- sowie zwei seriellen Verbindungen

"CamerLink Bits Wires" by Vswitchs - Own work. Licensed under CC BY-SA 3.0 via Wikimedia Commons - http://commons.wikimedia.org/wiki/File:CamerLink_Bits_Wires.svg#mediaviewer/File:CamerLink_Bits_Wires.svg

Neben den vier Datenleitungen, wird der Takt über eine fünfte LVDS Verbindung übertragen.

Die Medium- bzw. Full-Variante besitzt zwei bzw. drei dieser 28 Bit breiten Verbindungen, verdoppelt bzw. verdreifachen also die Übertragungsbandbreite.

Gemeinsam haben alle Camera Link Varianten zwei LVDS Verbindungen für eine asynchrone bidirektionale serielle Kommunikation sowie vier unidirektionale Steuerleitungen. Über die asynchrone bidirektionale serielle Schnittstelle können Kamera und System Daten austauschen. Die Kamera kann über diesen Kanal z.B. ihren aktuellen Status melden oder vom System aus parametriert werden. Zusätzlich

stehen vier LVDS Verbindungen vom Empfänger zur Kamera zu Verfügung. Ihre Aufgabe ist nicht näher spezifiziert. Ihre Verwendung hängt von Kamera und Anwendungsgebiet ab. Meist werden sie verwendet, um schnelle Signale an die Kamera zu schicken. So kann man eine Lösung realisieren, in der die Kamera nur auf Befehl anstatt zyklisch Bilder aufnimmt.

Der Pixeltakt ist mit bis zu 85 MHz festgelegt. Bei 24 Bit pro Takt ergibt dies eine maximale Datenrate D:

$$D = 85\ MHz * 24\ Pixel = 2040 \frac{Mbit}{s} = 255 \frac{MByte}{s}$$

Bezieht man die vier extra Bits mehr ein, so erhöht sich die maximale Datenrate auf:

$$D = 85\ MHz * 28\ Pixel = 2380 \frac{Mbit}{s} = 297{,}5 \frac{MByte}{s}$$

Da nicht jeder Takt für die Übertragung von Nutzdaten genutzt wird, ist die effektive Datenrate niedriger als die o.g. 255 MByte/s.

Bei der Verwendung einer Medium- bzw. Full-Verbindung verdoppeln bzw. verdreifachen sich diese Werte, sodass eine Datenrate von bis zu

$$D = 85\ MHz * 72\ Pixel = 6120 \frac{Mbit}{s} = 765 \frac{MByte}{s}$$

möglich ist. Durch die 7:1 Serialisierung muss eine LVDS Verbindung mit bis zu

$$85\ MHZ * 7 = 595\ MHz$$

angesteuert werden.

4 Das ZedBoard

Das ZedBoard ist ein von Avnet und Digilent entwickeltes Entwicklungsboard. Kern ist ein Zynq®-7020 All Programmable SoC von Xilinx. Das System on a Chip (SoC) enthält ein FPGA mit 85.000 Gattern und 560 MB BlockRAM sowie ein Dual ARM® Cortex™-A9 Prozessor [14]. Das ZedBoard wurde entwickelt um Systementwicklern eine Entwicklungs- und Testplattform für den Zynq®-7000 zu bieten. Dafür stellt das Board eine Reihe von Schnittstellen bereit, über die das Board an Peripherie angeschlossen und programmiert werden kann. Im Folgenden werden nur die Schnittstellen vorgestellt, die für die Programmierung und diese Arbeit relevant sind [2][3].

4.1 Schnittstellen

- SD-Karte
  Beim Start des Boardes wird die Konfigurationsdatei des FPGA sowie das Betriebssystem des ARM Prozessors von der SD-Karte gelesen. Eine Änderung des FPGS Designs oder am Betriebssystems muss über die entsprechende Datei auf der SD-Karte erfolgen.

- UART über USB
  Über diese Schnittstelle lässt sich eine serielle Kommunikation zwischen PC und ZedBoard herstellen.

- FMC
  Der FMC-Stecker stellte eine Verbindung zwischen FPGA und externer Logik her. FMC steht für FPGA Mezzanine Card und ist ein nach ANSI/VITA 57.1 spezifizierte Schnittstelle, die speziell auf FPGAs zugeschnitten ist. Auf dem ZedBoard ist die Low Pin Count Variante mit 160 Pins verbaut. Sie wird genutzt, um die Adapterplatine mit dem ZedBoard zu verbinden.

- Ethernet
  Über die Ethernetschnittstelle kann das System bzw. das auf dem Prozessor laufende Betriebssystem eine Verbindung mit dem Netzwerk und dem Internet herstellen. Sie kann auch verwendet werden, um das System von außen zu programmieren und zu debuggen.

- Schalter und LEDs
  Auf dem Board befinden sich mehrere LEDs, Schalter und Knöpfe, mit denen sich Signale auch ohne externe Hardware erzeugen und ausgeben lassen. Dies erleichtert das Debuggen und Konfigurieren des Systems.

4.2 Das Fieldprogrammable gate array – FPGA

FPGAs sind frei programmierbare Logikbausteine. Im Gegensatz zu Mikroprozessoren erhalten diese ihre Dynamik nicht durch das sequentielle Ausführen von Schritten, sondern durch das Verschalten der Logikelementen [4][15]. Kern dieser Logikelemente ist die Lookup Tabelle (LUT). Sie ordnet jeder Kombination von Eingängen einen Ausgangswert zu. Durch die Kombination mehrerer Tabellen über Flipflops, Multiplexer und andere Logikelemente lassen sich so beliebige Schaltungen realisieren. Der Vorteil gegenüber einem Mikroprozessor ist, dass die Logikelemente parallel laufen können. Aufgaben, die sich gut parallelisieren lassen, können so deutlich schneller ausgeführt werden. Ein Beispiel dafür ist die Anwendung eines Filters auf Bilddaten. Soll ein Filter mit einem 3x3 Kern auf das Bild angewendet werden, so muss in einem Mikroprozessor jeder Pixel über eine Schleife einzeln betrachtet werden. Pro Schleifendurchlauf müsste der Pixel geladen, der Filter angewendet und das Ergebnis verrechnet werden. Bei einem 3x3 Kern würde diese Schleife neun Mal durchlaufen. Es dauert daher viele Taktzyklen bis der Filter angewendet ist. Ein FPGA lässt sich so programmieren, dass alle neun Operationen gleichzeitig ausgeführt werden. Je komplexer die Aufgabe, desto mehr Schaltungen werden benötig. Die Komplexität einer Aufgabe wird daher von der Anzahl der Logikgatter beschränkt. Bei sehr komplexen Aufgaben oder bei Aufgaben, die sich schlecht parallelisieren lassen, ist es daher oft günstiger,

dennoch auf einen Mikroprozessor mit fester Hardware zu setzten. Für die gleiche Logikoperation wird in einem festverdrahteten Mikrokontroller nur die Hälfte bis ein Drittel der Fläche benötigt wie auf einem FPGA [4].
Programmiert wird ein FPGA in einer Hardwarebeschreibungssprache wie VHDL oder Verilog. Ein Synthetisierer übersetzt die Beschreibung dann in eine Konfigurierungsdatei, die das FPGA beim Start liest und darauf die Logikgatter und die LUT dementsprechend schaltet. Einige Entwicklungsumgebungen, wie das hier verwendete Vivado von Xilinx, ermöglicht es einzelne Anwendungen in IP-Cores zu verpacken. Diese können über vordefinierte Schnittstellen kommunizieren. Dies ermöglicht, den Entwicklungsprozess zu unterteilen. Auch können Hersteller so vorgefertigte Schaltungen bereitstellen, die der Anwender nur noch mit seinen IP-Cores verbinden muss.

4.1 Die ARM CPU

Um die Vorteile beider Systeme zu verbinden, ist auf dem Zynq®-7020 ein Mikroprozessor in Form eines dualcore ARM Cortex-A9 MPcore mit 512 MB DDR3 RAM und 32 Bit. Der Aufbau des Prozessors ist in Abb. 4-1 dargestellt. Für den weiteren Verlauf sind der DMA-Controller sowie das AMBA Interconnect von Bedeutung. Der DMA wird in Kapitel 4.3 und das AMBA Interface in Kapitel 4.2 näher beleuchtet. Die CPU ermöglicht es ein Betriebssystem wie ein Linux auf dem ZedBoard zu verwenden. Auch können Programme und Bibliotheken die in einen der Hochsprachen geschrieben wurden für Aufgaben wie die Bildverarbeitung verwendet werden.

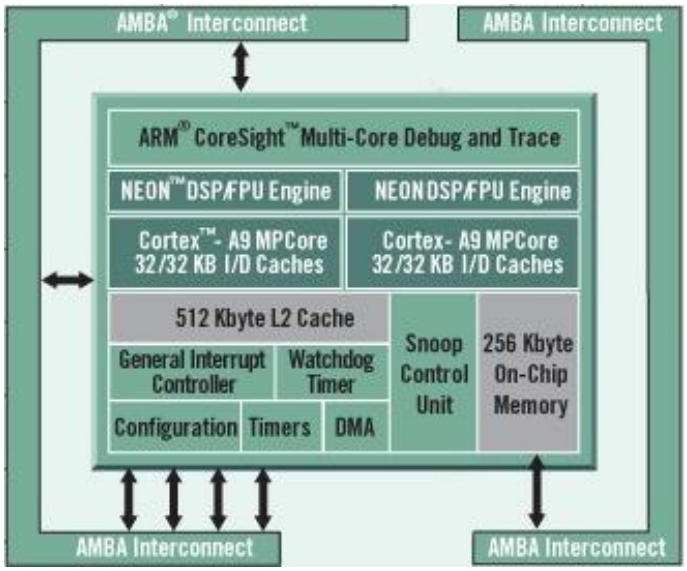


Abb. 4-1: Komponenten des ARM Prozessors mit Anbindung über AMBA Interconect
http://www.xilinx.com/products/silicon-devices/soc/zynq-7000.html

4.2 Das Advanced eXtensible Interface - AXI

Das Advanced eXtensible Interface (AXI) ist eine von ARM eingeführte Schnittstelle zur Verbindung von funktionalen Einheiten auf einem SoC. Es ist Teil der Advanced Microcontroller Bus Architecture (AMBA) [16]. AMBA beschreibt einige Schnittstellen zur Kommunikation von Komponenten auf einem SoC; eine davon ist AXI. AXI wurde das erste Mal in der Version 3 von AMBA eingeführt. Im Folgenden wird allerdings nur auf die aktuelle Version 4 eingegangen. Auch wenn das Dokument von Xilinx [16] nicht das Original von ARM ersetzten will, so wird es hier als Quelle benutzt, da es die Verwendung von AXI in der Xilinx und Vivado Umgebung besser, sowie die Spezifikationen ausreichend genau beschreibt.
AXI beschreibt eine Verbindung zwischen einem AXI-Master und einem AXI-Slave. In der AMBA 4 wird zwischen drei Varianten unterschieden.

- AXI4
- AXI4-Light
- AXI4-Stream

Bei AXI4 und AXI-Light besitzt der Slave und der Master mehrere Speicherbereiche, die adressiert werden können. Will der Master Daten an den Slave senden, schickt er zuerst die Adresse über den *Write Adresse Channel*. Danach werden die Daten über den *Write Data Channel* geschrieben. Sind alle Daten beim Slave angekommen, antwortet dieser über den *Write Response Channel.* Dies ist in Abb. 4-2 dargestellt. Abb. 4-3 zeigt den Lesevorgang. Will der Master lesen, so sendet er zunächst die gewünschte Adresse über den *Read Adress Channel*. Der Slave antwortet darauf mit den Daten über den *Read Data Channel*.

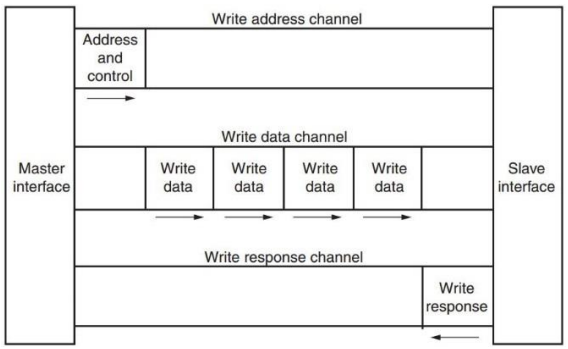


Abb. 4-2: AXI Sendevorgang von Master zu Slave

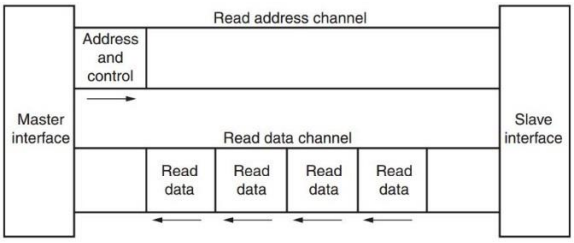


Abb. 4-3: AXI Sendevorgang von Slave zu Master

Da Sende- und Empfangskanal getrennt sind, kann in beide Richtungen gleichzeitig gesendet werden. AXI4 und AXI4-Light unterscheidet sich dahingehend, dass bei AXI4 pro Adressphase bis zu 256 Datenblöcke gesendet werden können. Bei AXI-4 Light ist es nur ein Block. Daher ist AXI4-Light langsamer, benötigt aber auch weniger Logik bei der Umsetzung. Er wird daher benutzt, wenn nur einzelne kleine Dateneinheiten ausgetauscht werden sollen. Jeder Channel besitzt Valid-Bits, die anzeigen, ob die Daten gültig sind. Mehrere Slaves und Master können über ein AXI-Interconnect zu einem Netzwerk verbunden werden.

AXI4-Stream verzichtet auf Adressen sowie einen Rückkanal. So kann ununterbrochen gesendet werden. Dies führt zu einem sehr hohen Datendurchsatz und wird verwendet, wenn große Mengen an Daten in Kilo- und Megabyte- Bereich gesendet werden sollen.

AXI4 und AXI4-Light können pro Adresse 32 bis 1024 Bit ansprechen. Die Breite von AXI4-Stream geht von 8 bis 1024 Bit.

4.3 Direct Memory Access - DMA

AXI4 und AXI4-Light sind Memory Maped, dies bedeutet, dass die Speicherbereiche können auch im RAM des Systems liegen können [16]. So kann direkt von einem Programm, das auf dem Prozessor läuft, Daten in die FPGA Logik geschrieben bzw. von dort empfangen werden. AXI4-Stream bietet dies nicht. Um die hohe Geschwindigkeit von AXI4-Stream zu nutzen, ist eine Schnittstelle nötig. Diese nennt sich *Direct Memory Access*, kurz DMA [17]. Für den DMA wird ein Speicherbereich des Hauptspeichers reserviert. In diesen kann ohne Verwendung der CPU direkt gelesen und geschrieben werden. Daher ist ein DAM-Controller nötig [24] (vgl. Abb. 4-1). Von dem Speicher werden Daten von einem AXI4 Master gelesen und dann über eine AXI4-Stream Master ausgegeben. Beide zusammen werden als *Memory Maped to Stream* (MM2S) bezeichnet. In der anderen Richtung werden die Daten von einem AXI4-Stream Slave empfangen und in den Speicher geschrieben. Dieser Prozess nennt sich *Stream to Memory Mapped* (S2MM). Beide Richtungen lassen sich simultan nutzen. Durch Verwendung eines Offsets lässt sich jedes Byte im reservierten Speicherbereich direkt ansprechen. Neben dem hier vorgestellten *Memory Maped Modus* gibt es noch einen *Gather-Scatter Modus*, auf den nicht näher eingegangen wird.
Gesteuert wird der DMA durch Register. Sowohl S2MM als auch MM2S besitzen vier Register, die die gleichen Aufgaben erfüllen.

- Control Register
- Status Register
- Source-/Destination-Register
- Transfer/Buffer lenght Register

Über das Control Register lässt sich der entsprechende Kanal einrichten. So lässt sich der Interrupt konfigurieren, der ausgelöst wird, wenn sich an den Daten etwas ändert. Die genaue Bedeutung jedes Bits lässt sich den Spezifikationen [17] entnehmen.

Über das Statusregister lässt sich u.a. entnehmen, ob ein Interrupt ausgelöst wurde. Auch hier lässt sich die Bedeutung jedes Bits den Spezifikationen [17] entnehmen. Das Source- (bei MM2S) bzw. Destination- (bei 2SMM) Register enthält die Adresse im RAM, an denen die Daten beginnen. Bei MM2S liegen dort die Daten, die gesendet werden sollen, bei MM2S soll dorthin geschrieben werden. Über einen Offset lassen sich relativ zu dieser Adresse die Speicherzellen ansprechen.
Das vierte Register enthält die Anzahl der Bytes, die bei MM2S geschrieben bzw. bei 2SMM empfangen werden.
Vivado stellt ein DMA-IP-Core bereit, der es schnell ermöglicht, ein DMA in das FPGA Design zu implementieren. Diese verbindet das FPGA mit dem DMA-Controller in der CPU. Adresse und Größe werden über den „Adress-Editor“ in Vivado eingerichtet.
Läuft die AXI-Schnittstellen des DMA mit einem anderen Takt, wie die FPGA Einheit, die die Daten bereitstellt bzw. benötigt, so kann dazwischen ein FIFIO geschaltet werden, das die Daten zwischenspeichert und so die Taktdifferenz ausgleicht.

Bei den meisten Betriebssystemen ist es Anwendungen nicht erlaubt, beliebig in den RAM zu schreiben [18]. Daher muss dem Betriebssystem das DMA als Gerät bekannt gemacht werden, um so einen festen, frei beschreibbaren RAM Bereich zu bekommen. Dazu wird ein Universal IO Device (UIODevice) eingerichtet. Dieses wird in dem verwendeten Linux erkannt und kann von einem Programm angesprochen werden. Dazu muss die Datei DeviceTree.dts angepasst werden, die beim Start des ZedBoards geladen wird. In ihr muss Name, Adresse und Größe des gewünschten Speicherbereiches eingetragen werde; Adresse und Größe muss mit den Einstellungen des DMA übereinstimmen. Sollen einzelne Werte direkt über AXI4 bzw. AXI4-Light gelesen bzw. geschrieben werden, so muss auch dafür ein UIODevice eingerichtet werden.

5 Inbetriebnahme des Adapters

Da das ZedBoard über keine Camera link Schnittstelle verfügt, wird ein Adapter diese über den FMC Stecker nachgerüstet. Die in Betracht gezogenen Lösungen der Firmen, Alpha Data, sowie Integre Technologies sind in der Lage zwei Base oder eine Medium bzw. Full Verbindung herzustellen. Die Lösung von Integre Technologies hat darüber hinaus ein eigenes FPGA verbaut. Da die verwendete Kamera nur über ein Base-Anschluss verfügt und somit die zusätzlichen Features der fertigen Lösungen nicht benötigt werden und diese darüber hinaus mit bis zu 1.400 $ recht kostspielig sind, wird ein eigener Adapter entwickelt. Dies geschah bereits über vier vergangene Studienarbeiten hinweg, in denen das Layout entworfen, verbessert und diverse Prototypen gefertigt wurden. Alle diese Prototypen wiesen gewisse Funktionsstörungen auf, die zunächst behoben werden müssen.

5.1 Aufbau der Platine

Idee hinter dem Adapter war es, neben einem Eingang für Camera Link auch eine Möglichkeit zum Testen und Debuggen zu schaffen. Dazu befindet sich neben der Hardware zum Empfangen auch Hardware zum Senden von Daten auf der Platine. Der Sendeteil simuliert dann eine Kamera, so kann der Empfangsteil getestet werden.

5.1.1 Empfangsseite

Kern der Empfangsseite ist ein Framegrabber in Form des DS90CR288AMTD [19] von Texas Instruments. Er empfängt die vier LVDS Leitungen sowie das Taktsignal und dekodiert sie in die 28 parallele Ausgangssignale. Den Eingangspins für die LVDS Leitungen sind noch die 100 Ohm Widerstände zwischengeschaltet, da diese nicht im IC integriert sind. Versorgt wird er mit 3,3 Volt. Mit dieser Spannung werden auch die Ausgangssignale ausgegeben. Er ist für eine maximale Frequenz von 85 MHz ausgelegt. Somit kann die Camera link Schnittstelle mit maximaler Frequenz

betrieben werden. Neben dem Framegrabber befindet sich auf der Empfangsseite der Sende-IC DS90LV048ATMTC [20] von Texas Instuments für das Camera Control Signal. Er codiert vier Eingangssignale auf je eine LVDS Leitung. Die dritte Komponente ist der DS90LV019TMTC [21] von Texas Instruments für die serielle Kommunikation. Zum einen empfängt er ein LVDS Signal und dekodiert es. Zum anderen sendet er auch ein LVDS Signal aus. So ist die bidirektionale Kommunikation über eine Full-Duplex Verbindung möglich. Auch hier ist für die Empfangsseite ein zusätzlicher 100 Ohm Widerstand nötig.
Zusätzliche Kondensatoren zwischen Versorgungsspannung und Masse tragen zur Spannungsstabilisierung bei.

5.1.2 Sendeseite

Die Sendeseite dient nur zum Testen der Empfangsseite und wird daher im Einsatz mit einer Kamera nicht mehr benötigt. Da sie eine Kamera simulieren soll, stellt sie eine Spiegelung der Empfangsseite dar. Der Framegrabber wird zu einem Framepusher in Form des DS90CR287MTD [19] von Texas Instruments. Er setzt 28 Eingangssignale in vier LVDS Signale und ein LVDS Taktsignal um. Das vom Empfänger ausgesendete CC Signal wird von dem DS90LV047ATMTC [22] von Texas Instruments empfangen und decodiert. Da der IC für die serielle Kommunikation sowohl senden als auch empfangen kann, kommt auf der Sendeseite der gleiche Chip wie auf der Empfangsseite zum Einsatz.

5.2 Messen und Beheben der vorhanden Störungen

Die bisher gefertigten Prototypen der Platine schafften es nicht, alle Signale fehlerfrei zu übertragen. Es ist daher zunächst zu prüfen, welche Signale Fehler aufweisen und wodurch diese verursacht werden. Dazu stand ein Testprogramm auf dem ZedBoard bereit, dass das Bitmuster 0x0000FFFF über jedes Ausgangsbit rotiert. Die Geschwindigkeit der Rotation kann über die Schalter des ZedBoards eingestellt werden. Die empfangenen Signale werden über die LEDs ausgegeben. Über die Schalter kann festgelegt werden, welche Eingangssignale ausgegeben werden.

Sollten alle Signale richtig empfangen werden, sollten die LEDs gemäß dem rotierenden Bitmuster wie ein Lauflicht durchlaufen. Nähere Informationen zu der Funktion des Testprogramms, sowie der Bedeutung der Schalterstellungen kann unter [23][3] entnommen werden.

Für die weitere Untersuchung wurde der Prototyp genommen, der laut der vorherigen Studienarbeit die wenigsten Fehlfunktionen hat. Das Testprogramm hat ergeben, dass die LVDS Verbindung X_0 für die Nutzdaten, sowie die serielle Verbindung fehlerhaft sind.

Abb. 5-1 zeigt ein Bit der funktionierende Verbindung X1. Die Phasenverschiebung entsteht dadurch, dass ein anderes Eingangs- als Ausgangsbits gewählt wurde. Bei Wahl des gleichen Bits, liegen Eingangs- und Ausgangssignal übereinander, wodurch die Darstellung in dieser Dokumentation erschwert wird. Abb. 5-2 zeigt dagegen ein Bit über den gestörten Kanal X0. Zu erkennen ist, dass der Ausgang immer auf HIGH ist und somit kein Signal empfangen wird.

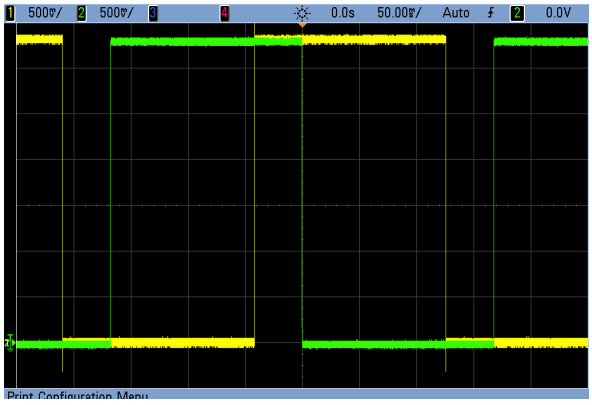


Abb. 5-1: Vergleich funktionierendes Signal; Grün: Referenz; Gelb: Ausgangsignal

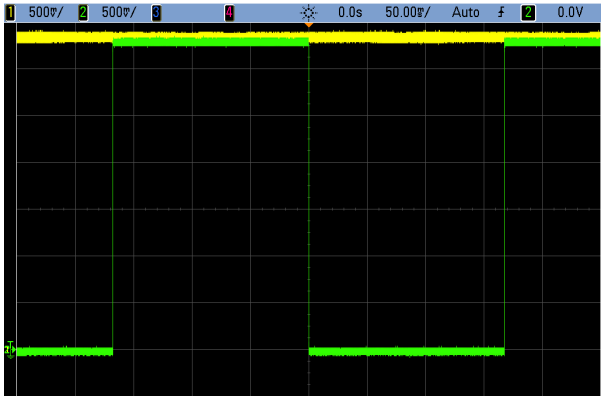


Abb. 5-2: Vergleich fehlerhaftes Signal; Grün: Referenz; Gelb: Ausgangsignal

Die Messung der LVDS Signale selbst erfolgt über ein Oszilloskop mit einer Grenzfrequenz von 100 MHz. Zum Testen wurde der Takt der Übertragung auf 25 MHz reduziert. So liegt der Takt auf ein Viertel der Grenzfrequenz und lässt sich so noch genau messen. Durch die 7:1 Seriealisierung ergibt sich allerdings ein Takt von 25 MHZ * 7 = 175 MHz auf den vier LVDS Leitungen der Nutzdaten, so dass das Signal nicht mehr korrekt dargestellt werden kann. Das Eingangs- und das

[3] Als PDF auf DVD

angezeigte Signal unterscheiden sich bei der Grenzfrequenz von 100 MHZ bereits um 3 dB. Wie Abb. 5-3 und Abb. 5-4 zeigen, lässt sich trotzdem gut der Unterschied zwischen einem funktionierenden und einem fehlerhaften Signal erkennen. Die Messung der differentiellen Signale erfolgt über die 100 Ohm Widerstände. Wird eine Seite des Widerstandes als Referenz angenommen, so misst die Messspitze des Oszilloskops nur die Spannungsdifferenz über den Widerstand. Da für die Funktionsprüfung auch die absoluten Werte eine Rolle spielen, erfolgt die Messung über zwei Messspitzen, die beide einen Masse-Pin des ZedBoardes als Referenz nutzen. Die Differenz kann von dem Oszilloskop berechnet und als zusätzlicher Graph ausgegeben werden. Über eine dritte Messspitze wurde auch der Eingangstakt aufgenommen, um das LVDS Signal einem Kontext zuweisen zu können, sowie eine regelmäßige und saubere Triggerung zu bekommen.

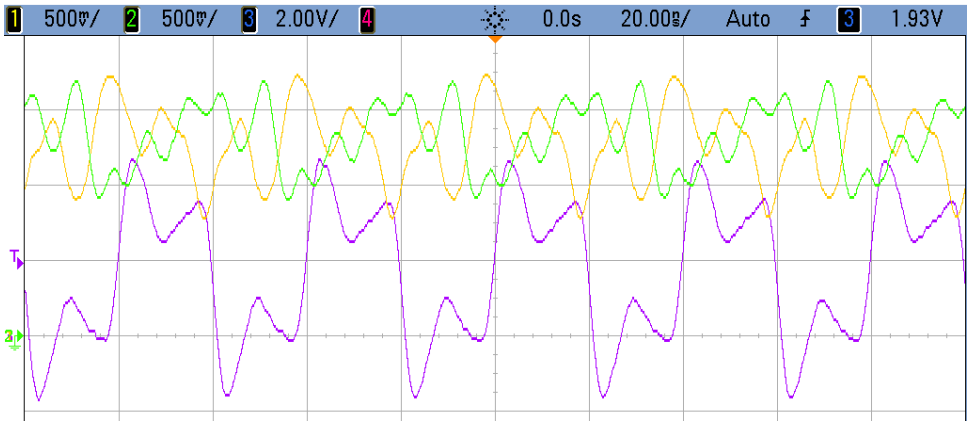


Abb. 5-3: LVDS Verbindung X1 über den Widerstand auf der Empfängerseite; Gelb:X1_P; Grün: X1_N; Violett: Takt

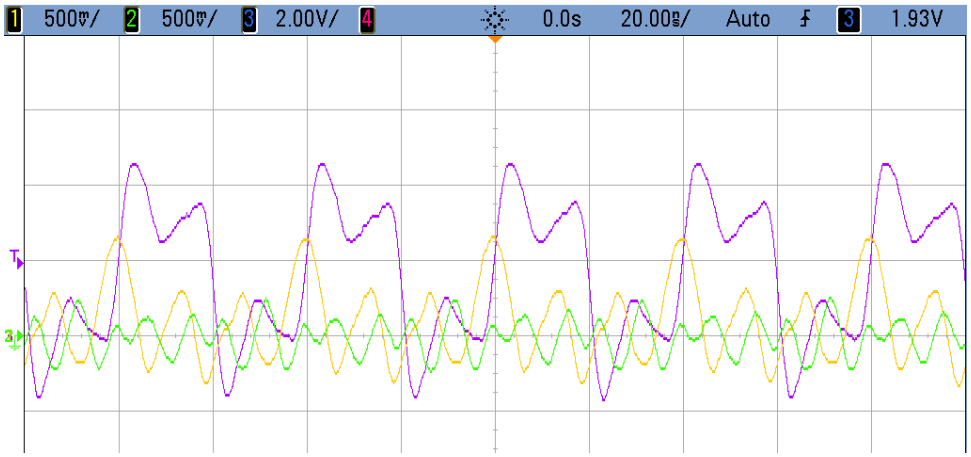


Abb. 5-4: LVDS Verbindung X0 über den Widerstand auf der Empfängerseite; Gelb:X0_P; Grün: X0_N; Violett: Takt

Abb. 5-3 zeigt die funktionierende LVDS Verbindung X1 über den Widerstand auf der Empfangsseite. Zu erkennen ist, dass die Spannungen gemäß der LVDS Spezifikationen mit ca. ±300mV um 1,2 Volt schwanken. In Abb. 5-4 dagegen schwanken die Spannungen um 0 Volt. Dass das Signal nicht vollständig auf NULL liegt, bedeutet, dass das Signal eine niederohmige Verbindung zu Masse hat, die das Signal nach unten zieht. Eine Messung ohne Verbindung von Sender und Empfänger über einen 100 Ohm Messwiderstand auf der Senderseite ergab, dass bereits dort das Signal gegen Masse gezogen wird. Eine Ursache auf der Empfängerseite oder im Kabel wird damit ausgeschlossen. Da durch eine optische Prüfung keine Verbindung von Signal und Masse festgestellt werden konnte, ist ein interner Defekt im Framepusher wahrscheinlich. Dies bestätigt die Annahme, die Karmann in [23] trifft.
Ein Tausch des Framepushers durch Entfernen und Auflöten eines Ersatzchips behob diese Fehler, wodurch nun alle vier Datenleitungen korrekt Daten übertragen.

Die LEDs, die die Signale der seriellen Verbindung ausgeben, sind dauerhaft an. Dies zeigt an, dass die Kommunikation in keine der beiden Richtungen funktioniert. Das Messen der beiden LVDS Signale ergab die in Abb. 5-5 und Abb. 5-6 dargestellten Signalverläufe.

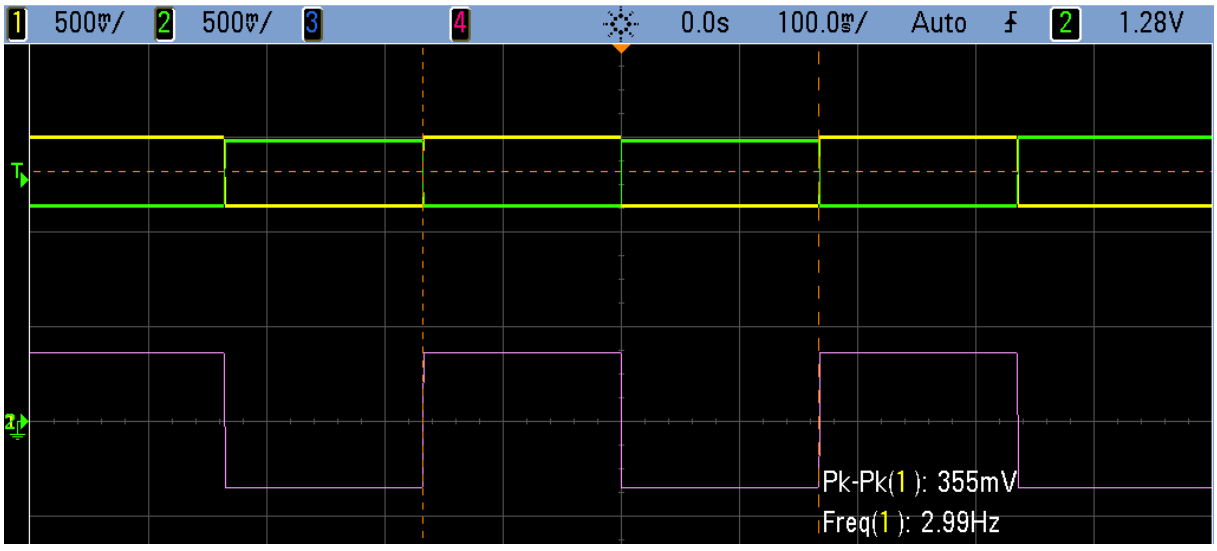


Abb. 5-5: Funktionierendes serielles Signal G_SERFC; Gelb: G_SERFC_N; Grün: G_SERFC_P; Violett: Differenz

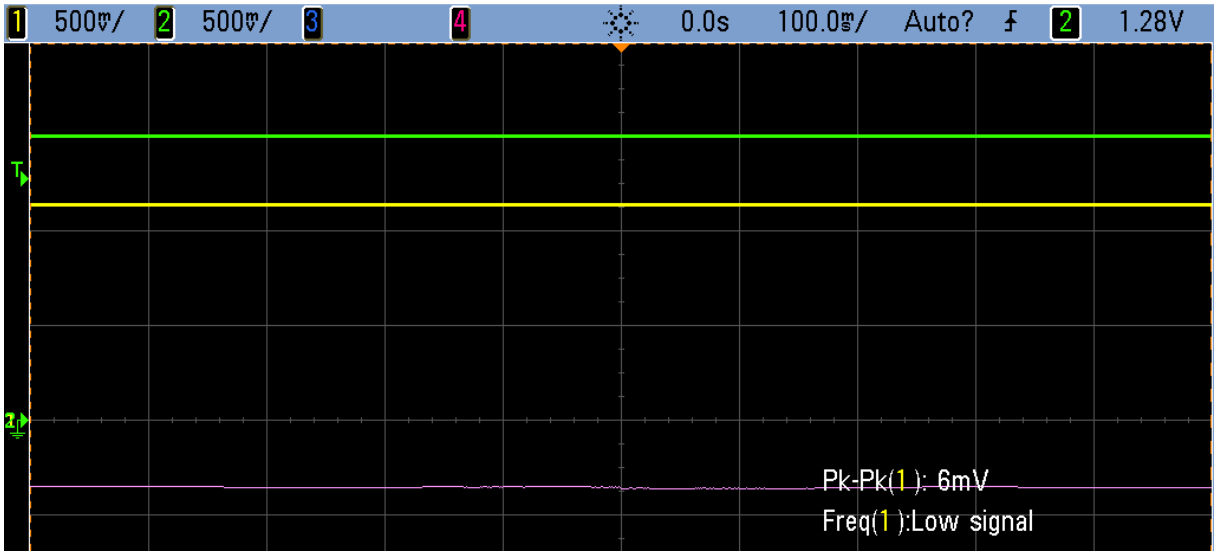


Abb. 5-6: Fehlerhaftes serielles Signal C_SERFFG; Gelb: C_SERFFG_N; Grün: C_SERFFG_P; Violett: Differenz

Abb. 5-5 zeigt das serielle Signal von der Senderseite kommend über den Widerstand auf der Empfängerseite. Der violette Graph zeigt die vom Oszilloskop gebildete Differenz zwischen den beiden Spannungen am Widerstand. Der Signalverlauf des rotierenden Bitmusters ist in Form eines Rechtecksignals zu erkennen. Abb. 5-6 zeigt die Gegenrichtung. Das Differenzsignal ist hier immer auf low. Eine Messung über einen 100 Ohm Widerstand direkt am Ausgang hat aufgezeigt, dass schon hier das Signal falsch ausgesendet wird. Da zwar das G_SERFC Signal richtig am IC auf der Empfängerseite ankommt, der IC aber kein richtiges Signal aussendet, ist zunächst von einem weiteren defekten IC ausgegangen worden. Weitere Messungen ergaben allerdings, dass am Ausgang des IC, an dem eigentlich das decodierte LVDS Signal an das ZedBoard weitergeleitet werden soll, ein Takt anlag. Dagegen lag an dem Eingang des IC kein Takt an. Diese Vertauschung zeigte sich auch bei der Messung der Ausgangspins direkt am FMC-Stecker ohne Adapterplatine. Damit konnte ein Fehler durch die Platine ausgeschlossen werden.

Behoben wurde dieser Fehler durch das Tauschen der Pinbelegung in der Konfiguration des FPGA. Nach der erneuten Synthese des Projektes zeigten nun auch die LEDs das gewünschte Blinkmuster.

Damit ist nun jeder Kommunikationskanal funktionsfähig. Da bisher nur mit 25 MHz sowie relativ wenigen Flanken der Eingangsbits getestet wurde, ist nun der Test mit

den maximalen 85 MHz sowie mit einer Erhöhung der Eingangsflanken durchzuführen. Abb. 5-1 und Abb. 5-5 zeigen, dass das Bitmuster bisher mit 3 Hz rotiert wurde. Dies führt zu einem noch erkennbaren Blinken der LEDs. Diese Frequenz ist nun auf 42,5 MHz erhöht worden. Das Ergebnis dieser Messung zeigt Abb. 5-7. Die Phasenverschiebung zwischen Eingang und Ausgang entsteht durch die Serialiserung. Es dauert genau ein Takt den Eingang zum Ausgang zu senden. Bei 85 MHz bedeutet dies eine Verzögerung von $\frac{1}{85\ MHz} \approx 11\ ns$. Diese Messung wurde mit allen Kanälen wiederholt, die alle das in Abb. 5-7. dargestellte Verhalten aufwiesen. Damit ist nun gezeigt, dass der Adapter auch bei hohen Frequenzen und Takten funktioniert.

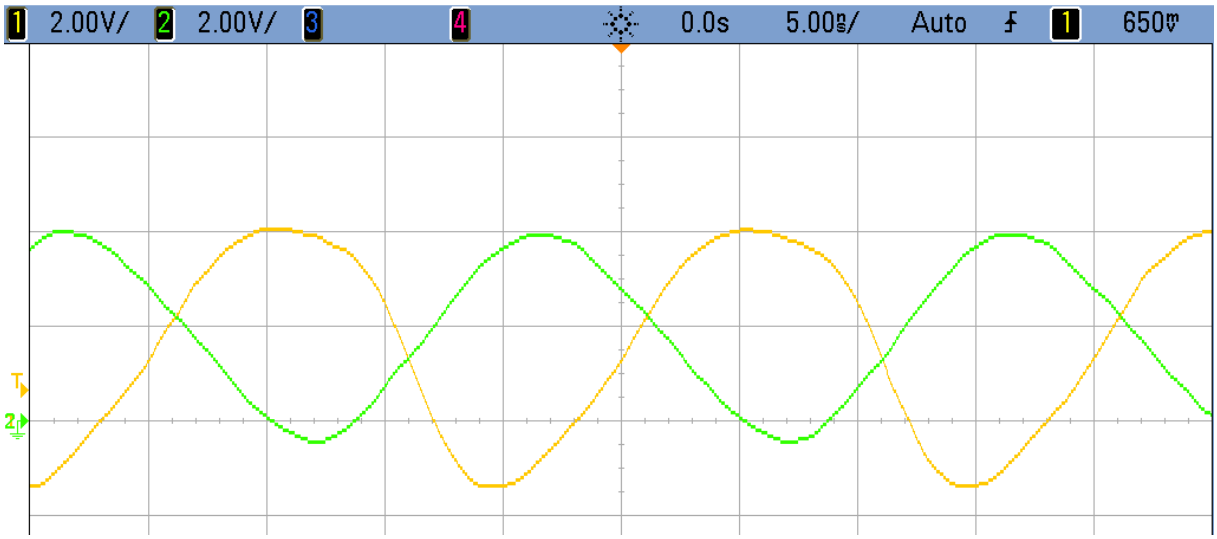


Abb. 5-7: Ein- und Ausgangssignal bei 85 MHz; Gelb: Eingang; Grün: Ausgang

6 Anbinden der Kamera

Nachdem nun gezeigt wurde, dass der Adapter funktionsfähig ist, ist die Kamera selbst in Betrieb zu nehmen. Ziel ist es, aus dem ARM Prozessor laufenden Linux über das FPGA und die Adapterplatine ein Signal an die Kamera zu senden, die daraufhin mit Bilddaten antwortet, die über ein Programm verarbeitet werden können. Dazu muss das FPGA so konfiguriert werden, dass die eingehenden Bilddaten über DMA in dem RAM gespeichert werden und das Camera Control Signal an die Kamera geschickt wird. Darüber hinaus muss das Programm so angepasst werden, dass es die ankommenden Daten auslesen und das Camera Control Signal ausgeben kann.

6.1 Verhalten und Eigenschaften der Kamera

Die verwendete Zeilenkamera von e2v mit der Bezeichnung AViiVA SC2 CL verfügt über 4096 einzelne Pixel, die in 1365 RGB Gruppen zusammengefasst sind. Die Belichtungsdauer kann zwischen einer und 32.000 µs betragen. Die Belichtung kann zyklisch oder extern ausgelöst über ein Camera Control Signal erfolgen. Auch die Belichtungszeit kann extern gesteuert werden. Als maximale Frequenz sind 60 MHz zulässig. Konfiguriert wird die Kamera über die asynchrone serielle Verbindung. Da diese Funktion noch nicht auf dem ZedBaord implementiert ist, erfolgt dies über einen PC mit eingebautem Framegrabber [1].
Ziel ist es, dass sowohl Belichtungszeit, als auch Belichtungszeitpunkt extern gesteuert werden. Dies erfolgt über die erste der vier Camera Control Leitungen. In der Dokumentation der Kamera ist dieses Signal *TRIG1* genannt. Abb. 6-1 zeigt den Verlauf, wenn ein einzelne Zeile aufgenommen werden soll.

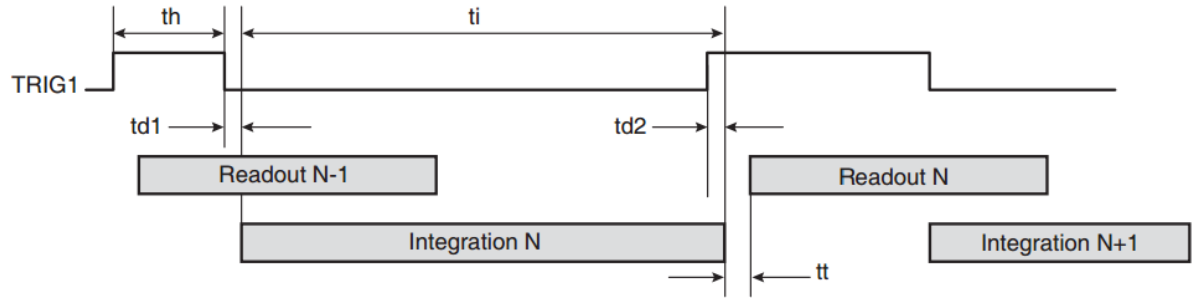


Abb. 6-1: Zeitdiagramm bei Aufnahme einer Zeile, wenn Aufnahmezeitpunkt und Aufnahmedauer extern über eine Quelle gesteuert werden
http://www.e2v.com/content/uploads/2014/02/doc0874D.pdf

Bei fallender Flanke von *TRIG1* beginnt die Kamera zu belichten. Mit der nächsten steigenden Flanke stoppt die Kamera die Aufnahme und sendet die aufgenommene Zeile. Die Anzahl der Takte, die für eine bestimmte Belichtungszeit notwendig sind, berechnet sich über folgende Formel:

$$Takte = Belichtungszeit * Frequenz$$

Abb. 6-2 stellt die Signale beim Senden dar. Strobe ist der gesendete Takt. Wenn das Line-Valid-Bit (LVAL) und das Data-Valid-Bit (DVAL) gesetzt sind, werden gültige Daten übertragen. LVAL steigt mit der ersten zu übertragenen RGB Gruppe an und fällt mit der letzten ab. DVAL zeigt für jede RGB Gruppe einzeln dessen Gültigkeit an, um undefinierte Werte zwischen zwei Gruppen zu vermeiden. Da es sich um eine Zeilenkamera handelt, wird auf ein Frame-Valid-Bit verzichtet.

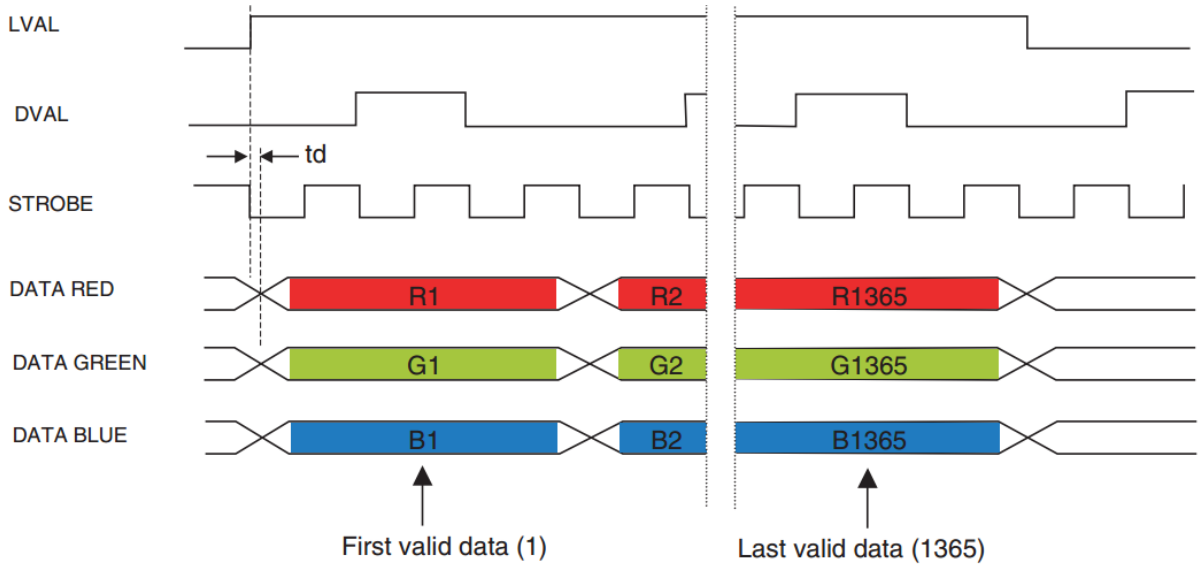


Abb. 6-2: Senden der Bilddaten über Camera Link von der Kamera zum Framegrabber
http://www.e2v.com/content/uploads/2014/02/doc0874D.pdf

6.2 FPGA Design

Dieser Abschnitt beschreibt das aktuelle Design des FPGA. Es besteht aus zwei Teilen, dem Hauptprojekt „Camera Link“ sowie dem IP-Core „CLReceive“. Im Hauptprojekt werden alle IP-Cores wie der DMA, FIFO, AXI-Interconects und dem CLReceive untereinander verbunden. Die im ursprünglichen Projekt enthaltenen IP-Cores für die Sendeseite wurden nach dem erfolgreichen Test in Abschnitt 5.2 entfernt. Auch werden Ports, Schnittstellen zu Hardware wie die LEDs, Switches oder dem FMC-Stecker eingerichtet und verbunden. Hier wird auch im Adresseditor die Adressen für das DMA und die Memory Mapped AXI4 festgelegt.
Der IP-Core „CLReceive“ besteht aus zwei Teilen. Der eine kümmert sich um die eingehenden Kameradaten, der andere steuert das ausgehende Camera Control Signal. Ein weiterer dritter Teil misst die aktuelle Eingangsfrequenz.

6.2.1 Daten Input

Dieser Teil verwaltet die eingehenden Bilddaten. Die 28 Camera Link Eingangsbits werden über den Port DATA_O bereitgestellt. Da die Eingangsbits von der Kamera gemischt werden, muss aus diesem Signal zunächst ein RGB-Wert zusammengebaut werden. Abb. 6-3 zeigt die Zuordnung von Eingangsbits zu ihrer Bedeutung.

Bit	DS90CR285 Pin Name	Bit	DS90CR285 Pin Name	Bit	DS90CR285 Pin Name	Bit	DS90CR285 Pin Name
RED-00	Tx0	RED-07	Tx5	BLUE-02	Tx19	GREEN-05	Tx14
RED-01	Tx1	GREEN-00	Tx7	BLUE-03	Tx20	GREEN-06	Tx10
RED-02	Tx2	GREEN-01	Tx8	BLUE-04	Tx21	GREEN-07	Tx11
RED-03	Tx3	GREEN-02	Tx9	BLUE-05	Tx22	STROBE	TxCLK
RED-04	Tx4	GREEN-03	Tx12	BLUE-06	Tx16	LVAL	Tx24
RED-05	Tx6	BLUE-00	Tx15	BLUE-07	Tx17	DVAL	Tx26
RED-06	Tx27	BLUE-01	Tx18	GREEN-04	Tx13	–	–

Abb. 6-3: Zuordnung von Signalen und Eingangsbits bei der verwendeten Kamera im parallelen RGB-Modus. TxA entspricht DATA_O(A)
http://www.e2v.com/content/uploads/2014/02/doc0874D.pdf

Sie werden jeweils zu einem roten, blauen bzw. grünen Wert zusammengesetzt, die zusammen in der Reichenfolge rot, grün, blau zu einem 24 Bit Wert kombiniert werden. Dieser wird über einen AXI4-Stream Master über ein FIFO an das DMA geschickt. Die Bits 24 (LVAL) bzw. 26 (DVAL) werden verundet und auf das Valid-signal des AXI4-Stream Masters gegeben. Somit sendet der Master nur gültige Daten, wenn auch gültige Daten von der Kamera kommen.
Zusätzlich zur Ausgabe der Daten über das DMA werden die Werte direkt über die LEDs ausgegeben. Dazu werden immer, wenn gültige Daten ankommen, die drei Farbkanäle addiert und durch drei geteilt. Das so entstandene, acht Bit große Signal wird auf die LEDs gegeben, so dass jede LED ein Bit repräsentiert. Der Schnitt kann also binär abgelesen werden. Da die Ausgabe mit jeder RGB-Gruppe erfolgt, entsteht ein blinkendes Muster. Eine LED ist umso heller, je öfter ihr Wert auf eins ist. Durch eine sich veränderte Helligkeit verändert sich das Muster. Viele (dunkel)

leuchtende LEDs zeigen eine hohe Helligkeit an, wenige dagegen ein dunkles Signal. Über den Switch 0 kann diese Ausgabe zu- oder abgeschaltet werden.

6.2.2 Camera Control Output

Das Design für das Camera Control Signal besteht aus zwei Elementen, die über den Switch 1 ausgewählt werden können. Ist der Switch EIN so wird ein rein hardwareseitiges Signal ausgegeben. Dafür wird für bei jeder negative Taktflanke ein Zähler hochgezählt. Ist der Zähler kleiner als 0x7FFF, so sind alle CC-Leitungen HIGH. Nach 0x7FFF Takten werden die Leitungen auf LOW gesetzt. Dieser Zustand wird für 18.000 Takte beibehalten. In dieser Zeit belichtet die Kamera. Danach werden die CC-Signale wieder für 0x7FFF auf HIGH gesetzt. Durch die Aufteilung der beiden HIGH Zyklen wird garantiert, dass in einem Durchlauf ein fallende sowie eine steigende Flanke vorhanden ist.
Ist der Switch AUS, so wird das Signal über die Software und die AXI4-Register verwaltet. Dieser Teil funktioniert zum Ende dieser Arbeit noch nicht, weshalb hier nur die gedachte Funktionsweise beschreiben wird. Ziel ist es Dauer, Wert und Wiederholungen des CC-Signales zu steuern.
Dazu kommen zwei, je 32 Bit breite, Register zum Einsatz. Das eine enthält die Anzahl der Takte, die das Camara Signal HIGH und wie viele Takte es auf LOW ist. Dafür sind je 16 Bit reserviert. Verwendet wird dafür das AXI4 Register mit der Adresse 4. Das zweite Register, mit der Adresse 3, enthält die in Tabelle 1 dargestellten vier Informationen.

Tabelle 1: Bedeutung der Bits in Camera Control Register 3

Bits	Bedeutung
31…10	Anzahl Wiederholungen falls im „loop mode“
9..8	Modus: b“00“ bzw. b“10“ -> Kein Trigger bzw. Reset b“01“ -> Trigger wird bis zum Reset wiederholt -> „infinity mode“ b“11“ -> Trigger wird X mal wiederholt – „loop mode“
7..4	Wert der über Camera Control während LOW gesendet werden soll (i.d.R. b“0000“
3..0	Wert der über Camera Control während HIGH gesendet werden soll (i.d.R. b“0111“)

Für das Register 3 ist zusätzlich eine Flagge *slv_reg3_written* eingeführt, die anzeigt, ob in das Register geschrieben wurde. Dies ist für den „loop mode“ notwendig, da sonst die Anzahl der Wiederholungen immer wieder neu gesetzt wird. Der Prozess zur CC Steuerung wird von vier Variablen und zwei Signalen unterstützt. Diese sind in Tabelle 2 aufgeführt. Die Variablen speichern die Werte der Register zwischen, um eine Änderung der Werte während eines Zyklus zu vermeiden. Sie werden nur unter bestimmten Bedingungen gesetzt. So können zu einer beliebigen Zeit die Register geändert werden, ohne dass der aktuelle Ablauf gestört wird.

Tabelle 2: Typ und Bedeutung der Signale und Variablen zur softwareseitigen Steuerung des Camera Control Signals

Typ	Name	Bedeutung
Signal	loops	Zähler für die Wiederholungen. Ist Bit 22 gesetzt, so befindet sich das System im *infinity mode.* Ist Bit 22 nicht gesetzt und loops nicht gleich null, so wird nach jedem Triggerzyklus der Wert um eins reduziert
Signal	CCCounter	CCCounter: Zähler für jeden Triggerzyklus
Variable	CCHighTicks	Anzahl Takte in denen Das CC Signal HIGH ist
Variable	CCLowTicks	Anzahl Takte in denen Das CC Signal LOW ist
Variable	CCHighValue	Wert der während der LOW Phase gesendet wird
Variable	CCLowValue	Wert der während der LOW Phase gesendet wird

Das Register 4 wird dann abgefragt, wenn sowohl *CCCounter* als auch *loops* gleich null sind. Somit werden *CCHighTicks* bzw. *CCLowTicks* nur dann geändert, wenn aktuell kein Signal ausgegeben wird.
Wann Register 3 ausgelesen wird, hängt vom gesetzten Modus ab. Für alle Modi gilt, dass nur gelesen wird, wenn *CCCounter* null ist, also zwischen zwei Zyklen. So kann ein Zyklus nicht verändert oder abgebrochen werden. Ist der Modus *Reset*, so wird das Triggersignal gestoppt und alle Werte auf ihren Standardwert gesetzt.
Ist der Modus der *infinity mode*, so wird erst dann gelesen wenn bereits ein Wert für *CCLowTicks* und *CCHighTicks* gesetzt wurde. Hier werden die Werte für *CCLowValue* und *CCHighValue* gelesen, sowie Bit 22 in *loops* gesetzt.
Für den *loop mode* gilt, dass nur gelesen wird, wenn neben gesetzten Werten für *CCLowTicks* und *CCHighTicks* auch loops gleich null und die *slv_reg3_written* Flagge gesetzt ist. So kann erst eine neue Anzahl an Wiederholungen vorgegeben werden, wenn die bisherige Anzahl abgearbeitet wurde. Durch die Flagge wird verhindert, dass sobald *loops* auf NULL fällt, das Register erneut gelesen und dadurch *loops* wieder gesetzt wird.
Von *Loops* hängt es ab, ob ein Camera Control Signal ausgegeben wird. Ist loops gleich null, also weder Bit 22 im *infinity mode* noch die anderen im *loop mode* mit einem Wert gefüllt, wird kein Signal ausgegeben. Ist *loops* ungleich null, wird das am Anfang des Kapitels beschriebene Signal ausgegeben, nur dass diesmal Grenzen und Werte nicht mehr fest vorgegeben sind. Jedes Mal wenn der *CCCounter* das Maximum erreicht, wird geprüft, ob sich das System im *infinity mode* befindet. Wenn nicht, wird *loop* um eins reduziert. Die Ausgabe stoppt daher, wenn *loops* null erreicht oder *loops* über ein Resetsignal auf null gesetzt wird.
Da die vollständige Kontrolle des CC-Signals über Register noch nicht funktioniert, ist ein Workaround eingeführt worden. Zwischen den beiden Varianten lässt sich, neben dem Schalter 1, auch über das *slv_reg2* umschalten. Ist das Bit 0 gesetzt, so wird die feste Implementierung ausgeführt, ist es nicht gesetzt, so wird die Register gesteuerte Variante ausgeführt.

6.3 C++ Programm

Die auf dem Linux und der ARM CPU laufende Software soll die von dem FPGA empfangen und aufbereiteten Bildsignale verarbeiten. Die verwendete Software basiert auf einer von Herrn Dr. Heintz zu Verfügung gestellten Version. Zum Steuern des Systems muss sie zum einen die Aufnahme starten und zum anderen die Bilddaten empfangen können. Schnittstelle zwischen beiden Systemen ist der RAM. Damit die Software, die im *user-mode* läuft, die Adresse, die mit dem DMA bzw. dem AXI4 verbunden ist ansprechen kann, muss als erster Schritt in der Software die dafür angelegen UIODevices geöffnet werden. Dazu werden zwei Objekte der Klasse *UIODevice* instanziiert. Die Klasse enthält Informationen wie Adresse und Name des *UIODevice,* sowie Methoden zum Schreiben und Auswerten des RAM-Bereiches. Um saubere Werte zu erhalten wird zunächst der RAM-Bereich des DMA mit Nullen befüllt. Anschließend werden die Register des Empfangs-DMA gesetzt. Der Befehl *Receive.writeReg(CCOverride,ALLONE)* setzt das slv_reg2 und startet somit das Trigger Signal. Das Programm wartet nun, bis ein Interrupt signalisiert, dass Daten in den DMA geschrieben wurden. Jetzt wird das Trigger-Signal wieder beendet und der Inhalt des RAMs sowie Zusatzinformationen wie ein Durchschnitt oder der gemessene Takt ausgegeben. Um die Konsole nicht zu überfüllen, wird nur jede 16. der 1365 eingehenden RGB-Gruppen ausgegeben.

7 Ergebnis und Ausblick

Als Ergebnis existiert nun eine Verbindung zwischen dem ZedBoard und der Kamera. Zum einen ist der Adapter zwischen FMC und Camera Link fertig gestellt und funktionsbereit, zum anderen ist es möglich Signale zur Kamera zu schicken bzw. von ihr zu empfangen. Nach dem Beheben der gefundenen Fehler der Adapterplatine haben Messungen gezeigt, dass jene auch bei hohen Frequenzen ordnungsgemäß arbeitet. Durch die Anpassung des FPGA Designes ist es darüber hinaus möglich, Bilddaten von der Kamera zu empfangen und sowohl über die LED als auch über ein DMA auszugeben. Auch lassen sich Camera Signale an die Kamera senden. Dadurch ist es möglich, auf Befehl Bilder aufzunehmen. Zur Zeit ist es allerdings nur möglich dieses Signal über einen Schalter bzw. ein AXI4-Register ein und aus zu schalten. Auch ist es nicht möglich die Belichtungszeit zu ändern. Daher ist zunächst der VHDL-Code so zu ändern, dass eine vollständige Kontrolle des Camera Control Signals über AXI4-Register möglich ist. Danach sind folgende Punkte zu untersuchen und zu realisieren:

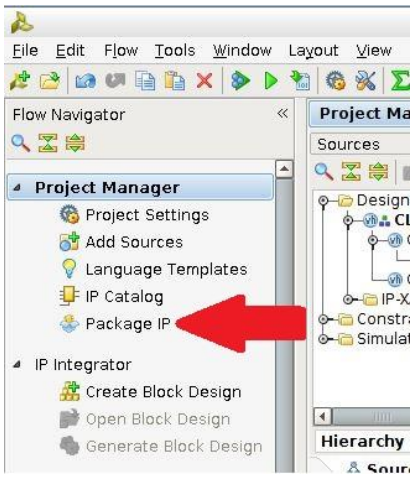
- Herstellung der seriellen Verbindung zwischen Kamera und ZedBoard.
  z.Z. ist es nicht möglich Daten über die serielle Verbindung zu senden bzw. zu empfangen. Daher ist das FGPA so zu designen, dass über AXI Daten gesendet und empfangen werden können. Hier muss eine Sende- und Empfangseinheit implementiert werden.

- Vorverarbeitung der Bilddaten.
  Aktuell werden die eingehen Bilddaten direkt über DMA in das RAM geschrieben. Für eine effektive Nutzung der ARM-CPU ist es nötig, die Daten im FPGA vorzubereiten. Es ist daher zu untersuchen, welche Informationen die CPU benötigt, um möglichst effizient die Objekterkennung durchzuführen und wie das FPGA diese aus den eingehen Bilddaten gewinnen kann.

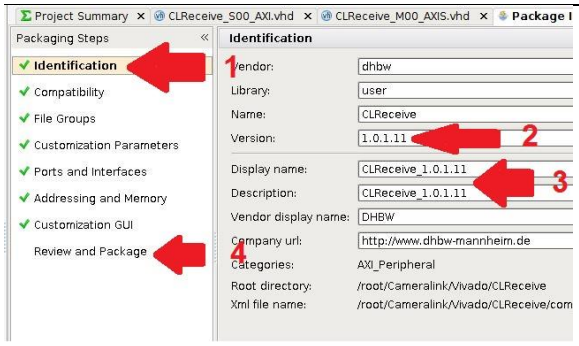
8 Anhang

8.1 Von VHDL zur SD-Karte

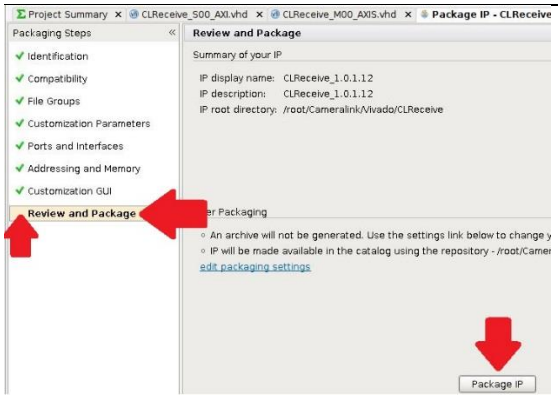
Dieser Abschnitt zeigt, welche Schritte unternommen werden müssen, damit eine Änderung in einer IP übernommen wird. Es wird angenommen, dass der VHDL Code fehlerfrei ist. Begriffe in eckiger Klammer [] bezeichnen Schaltflächen in Vivado. Zahlen in runden Klammern () bezeichnen Pfeile im Bild (falls vorhanden)



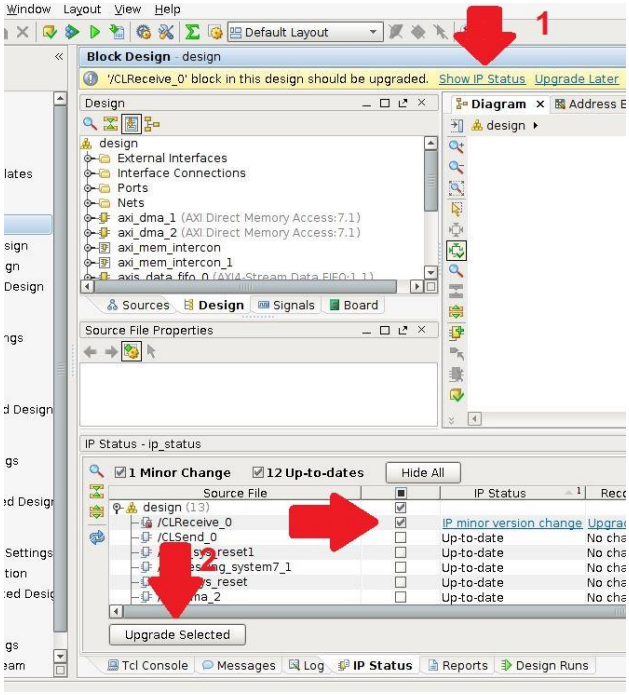
1. Klick auf [Package IP]



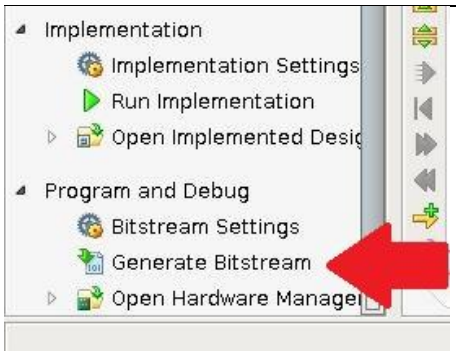
2. Unter [Identifikation] (1), die Versionsnummer (2) anpassen.
3. Optional Name und Beschreibung anpassen (3)
4. Klick auf Review and Package (4)



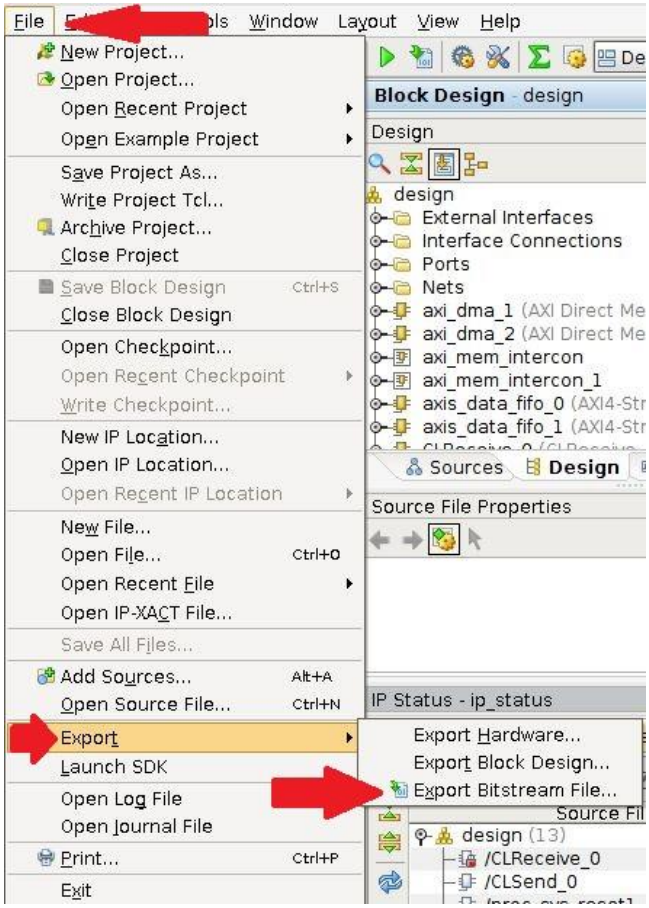
5. Sind Ports geändert worden, müssen diese über den Punkt [Port und Interfaces] geladen werden.
6. Wenn alle Punkte einen grünen Haken haben, kann über [Package IP] die IP gepackt werden.



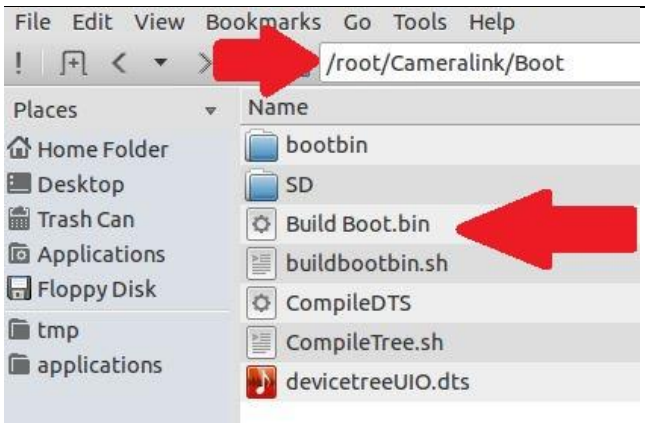
7. Projekt laden, in dem die IP verwendet wird.
8. Block Design öffnen
9. Klick auf [Show IP Status] (1)
10. Prüfen ob geänderte IP mit Haken ausgewählt ist
11. Klick auf [Upgrade Selected] (2)



12. Klick auf [Generate Bitstream]. Dies dauert, je nach System, aktuell 15 bis 30 Minuten.



13. Exportieren des Bitstreams über [File] -> [Export] -> [Export Bitstream File…]
14. Speichern unter <Pfad zu Projekt>/Boot/bootbin
15. Vorhandene system.bit ersetzten



16. Das Skript „Build Boot.bin“ packt die Dateien aus „bootbin“ und legt sie im Ordner SD ab
17. Soll ein DMA bzw. UIODevice hinzugefügt oder entfernt werden, muss die Datei „devicetreeIO.dts“ angepasst und über „CompileDTS“ gepackt werden

Anschließend werden die Dateien aus dem Ordner SD auf die Bootpartition der SD Karte kopieren und die vorhandenen Dateien ersetzt.

8.2 Inhalt DVD

- /SD_Image
  Enthält das aktuelle Abbild der SD Karte
- /Studienarbeiten
  Enthält diese Studienarbeit, sowie die von Karmann und Specht
- /Eagle
  Eagle Dateien mit Schaltplan und Design des Adapters
- /Cameralink
  Aktuelle Projektdateien
- /Quellcode
  Der aktuelle Quellcode als .vhd und .cpp Datein

9 Literaturverzeichnis

Stelle	Quelle	
[1]	o.V. (2009):	AViiVA® SC2 CL Camera Link® Color Linescan Camera Datasheet, http://www.e2v.com/content/uploads/2014/02/doc0874D.pdf, Abgerufen am 12.02.2015
[2]	o.V. (2014):	Avnet Product Brief ZedBoard, http://ZedBoard.org/sites/default/files/product_briefs/ZedBoard%20Brochure%20%28English%29.pdf, Abgerufen am 02.03.2015
[3]	o.V. (2012):	Getting Started Guide, Version 7.0, http://ZedBoard.org/sites/default/files/documentations/GS-AES-Z7EV-7Z020-G-V7.pdf, Abgerufen am 02.03.2015
[4]	Reichert, J. / Schwarz, Bernd (2007):	VHDL-Synthese, Entwurf digitaler Schaltungen und Systeme, 5.,aktualisierte Auflage, München 2009
[5]	o.V. o.J.:	Boccaccio - FMC Camera Link, http://www.toyon.com/downloads/FMC_CameraLink.pdf, Abgerufen am 15.01.2015
[6]	o.V. (2012):	FMC-CAMERALINK, Datasheet Revision 1.1, http://www.alpha-data.com/pdfs/fmc-cameralink.pdf, Abgerufen am 15.01.2015
[7]	o.V. (2012):	FMC-200 Programmable Camera Interface FMC Card, http://integretek.com/products/FMC200.html, Abgerufen am 15.02.2015
[8]	Specht, T. (2014):	Aufbau einer kleinen Sortiereinheit
[9]	o.V. (2002):	LVDS Application and Data Handbook, http://www.ti.com/lit/ug/slld009/slld009.pdf, Abgerufen am 12.02.2015

[10]	o.V. (2000):	Specifications of the Camera Link Interface Standard for Digital Cameras and Frame Grabbers, http://www.fastvideo.ru/info/cameralink/CameraLinkOfficial.pdf, Abgerufen am 15.01.2015
[11]	Wikipedia contributors (2014):	Camera Link. Wikipedia, The Free Encyclopedia. Wikipedia, The Free Encyclopedia, 21 Dec. 2014. Web. 12 Mar. 2015.
[12]	Wikipedia contributors (2015):	Low-voltage differential signaling. Wikipedia, The Free Encyclopedia. Wikipedia, The Free Encyclopedia, 11 Feb. 2015. Web. 12 Mar. 2015.
[13]	o.V. (2011):	National Semiconductor Channel Link Design Guide, http://www.ti.com/lit/ml/snla167/snla167.pdf, Abgerufen am 01.03.2015
[14]	o.V. o.J.:	Zynq-7000 Silicon Devices, http://www.xilinx.com/products/silicon-devices/soc/zynq-7000/silicon-devices.html, Abgerufen am 13.02.2015
[15]	Himpe, V. (2009)	Digitale Logik selbst entwickeln, Aachen 2012
[16]	o.V. (2014):	Vivado Design Suite AXI Reference Guide, Revision v2.1, http://www.xilinx.com/support/documentation/ip_documentation/axi_ref_guide/latest/ug1037-vivado-axi-reference-guide.pdf, Abgerufen am 30.01.2015
[17]	o.V. (2014)	LogiCORE IP AXI DMA v7.1 Product Guide, http://www.xilinx.com/support/documentation/ip_documentation/axi_dma/v7_1/pg021_axi_dma.pdf, Abgerufen am 20.02.2015
[18]	Beck, M. / u.A. (1994):	Linux Kernel-Programmierung, Algorithmen und Strukturen der Version 2.4. 6 Auflage, München

		2001
[19]	o.V. (2013):	DS90CR287/DS90CR288A +3.3V Rising Edge Data Strobe LVDS 28-Bit Channel Link - 85MHz, http://www.ti.com/lit/ds/symlink/ds90cr287.pdf, Abgerufen am 01.03.2015
[20]	o.V. (2013):	DS90LV048A 3V LVDS Quad CMOS Differential Line Receiver, http://www.ti.com/lit/ds/symlink/ds90lv048a.pdf Abgerufen am 01.03.2015
[21]	o.V. (2000):	DS90LV019 3.3V or 5V LVDS Driver/Receiver, http://www.ti.com/lit/ds/symlink/ds90lv019.pdf, Abgerufen am 01.03.2015
[22]	o.V. (2013):	DS90LV047A 3V LVDS Quad CMOS Differential Line Driver, http://www.ti.com/lit/ds/symlink/ds90lv047a.pdf, Abgerufen am 01.03.2015
[23]	Karmann, S. (2015):	Inbetriebnahme einer Cameralink-Schnittstelle für das ZedBoard
[24]	Beierlein, T. / Hagenbruch, O. (1999):	Taschenbuch Mikroprozessortechnik, 4. Auflage, München 2011