Embedded Linux auf FPGA-Hardware zur Bildverarbeitung

Studienarbeit
im Rahmen des Diplomstudiengangs Informatik

eingereicht am Institut für Informatik
der Humboldt-Universität zu Berlin

von .... Martin Brückner
geb. am .... 19. März 1982
in .... Berlin

Betreuer:	Prof. Dr. Beate Meffert Dr.-Ing. Frank Winkler Dipl.-Inf. Marcus Ehrig, IHP Frankfurt/Oder
eingereicht am:	10. August 2007

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung

Das kostenlose und von Linus Torvalds entwickelte Betriebssystem Linux ist zur Zeit recht beliebt. Dieses wurde ursprünglich für den Intel 80386 entwickelt. In einer E-Mail [36] beschrieb er, dass es jedoch schwierig werden würde, dieses zu anderen Plattformen zu portieren. Entgegen seiner Aussage ist Linux inzwischen für viele verschiedene Architekturen verfügbar. Darunter sind ARM, StrongARM, Alpha, MIPS, PowerPC, Sun SPARC und viele weitere. Der größte Vorteil ist wohl, dass sich Linux auf jeder unterstützten Plattform gleich verhält, da dieselben C-Bibliotheken eingesetzt werden. Das heißt, der größte Teil des C-Quellcodes ist weiter verwendbar, da mit Hilfe eines speziellen Compilers (Cross-Compiler) den C-Code für die Plattform übersetzt werden kann. Dabei kann der Quelltext unangetastet bleiben.

Einige, der zur Zeit auf dem Markt verfügbaren FPGAs, haben entweder schon einen kompletten Prozessorkern integriert oder besitzen die Möglichkeit ihn als Hardwaredesign zu laden, so dass bei geeigneter Wahl des FPGAs bzw. des intgerierten CPU-Kerns ein Linux darauf laufen kann. Die Aufgabe ist, ein Linux zu finden, dass sich gut in die Struktur des FPGA-Boards einfügt. Es müssen dafür drei Voraussetzungen erfüllt sein.

Als erstes muss eine passende Konfiguration für die FPGAs gefunden werden, damit ein Linux darauf läuft. Es muss eine CPU eingebunden werden und eine geeignete Busstruktur im FPGA gewählt werden. Das so entstandene System (System on a chip) kann dann mit der externen Hardware verbunden werden. Des Weiteren benötigt man einen geeigneten Kernel, der die Unterstützung wichtiger Hardwarekomponenten (wie Ethernet, Massenspeicher, etc) enthält. Zu letzt benötigt Linux eine Umgebung, also wichtige Systemprogramme, Konfigurationsdateien, Bibliotheken und Bootskripte. Im eingebetteten Bereich muss dabei darauf geachtet werden, dass der Ressourcenverbrauch (Speicher und Energie) so gering wie möglich ist.

Zusätzlich wird im Rahmen der Arbeit die Frage untersucht, ob und in welchem Umfang ein Linux auf FPGA-Hardware für die Bildverarbeitung und andere Aufgaben geeignet ist. Dabei ist es wichtig zu wissen, ob und wie weit sich die Softwareentwicklungszeit reduzieren oder die Programmkomplexität erhöhen lässt.

Insbesondere wird darauf eingegangen, welche Komplikationen sich ergeben können und ob sich Linux als performant genug erweist, um in der Bildverarbeitung sinnvoll eingesetzt werden zu können. Am Ende des Kapitels wird auch die Implemetierungung eines Beispielsystems dargestellt.

Als nächstes wird der Unterschied von FPGAs gegenüber CPUs und ASICs erklärt. Dem folgt eine kurze Einführung zum Xilinx Virtex-II Pro FPGA und den zur Verfügung stehenden Boards. Das nächste Kapitel beschäftigt sich dann mit den möglichen Linuxvarianten und alternativen Betriebssystemen. Es geht auch auf die benötigte Hardware ein.
Danach wird eine Beispielimplementierungen auf den Boards mit den möglichen Hindernissen und Hürden beschrieben. Die Arbeit schließt mit einer Zusammenfassung und einem Ausblick ab.

1.1 Vergleich von CPU, DSP, FPGA und ASIC

Zunächst wird gezeigt, wo sich der FPGA im Verhältnis zur CPU (Central Processing Unit), dem DSP (Digital Signal Processor) und dem ASIC (Application Specific Integrated Circuit) in Bezug auf Flexibilität, Performance, Spezialisierung und Leistungsverbrauch einordnet. Es werden aber auch allgemeine Unterschiede genannt.

Eine CPU ist in der Lage einen bestimmten Programmcode auszuführen. Dazu wird dieser aus einem Speicher gelesen und eventuell weitere Daten angefordert. Somit ist es möglich einfache und komplexe Aufgaben auszuführen. Zu beachten ist dabei, dass das Programm sequentiell ausgeführt wird. Somit sind auch schon die Vor- und Nachteile einer CPU erkennbar. Sie erlaubt die Abarbeitung grundverschiedener Befehle und ist somit sehr flexibel einsetzbar. Früher wurde pro Takt immer nur ein kleiner Bereich der CPU genutzt. Dies hängt damit zusammen, dass kein Befehl den kompletten Prozessor ausnutzen kann. Somit ist eine CPU sehr ineffizient, da der ungenutzte Bereich weiterhin Energie verbraucht. Moderne Prozessoren nutzen mit Pipelines, mehrerer Recheneinheiten, und ähnlichem den Chip besser aus. Sie nutzen aber dennoch ein Programm welches erst aus dem Speicher geladen und verarbeitet werden muss. In der Bildverarbeitung werden viele Algorithmen parallel abgearbeitet. Performancesteigerungen lassen sich hier nur durch das Parallelschalten weiterer CPUs bewerkstelligen, was aber kosten- und wartungsintensiv ist.

Ein ASIC ist ein, in einem Chip integrierter, Schaltkreis für spezielle Anwendungen, der nicht verändert werden kann. Er wird direkt auf das Silizium aufgebracht. Somit ist er im Gegensatz zur CPU nicht in der Lage universelle Probleme zu lösen. Vielmehr ist es möglich, eine Aufgabe mit maximaler Performance durchzuführen, was nicht zuletzt an der Möglichkeit liegt, Algorithmen hochgradig parallel verarbeiten zu können. Somit wird die Chipfläche zu jeder Zeit zu einem Großteil ausgenutzt, was die Energiekosten senkt. Der Nachteil eines ASICs liegt, wie schon erwähnt, in der Unveränderlichkeit. Wird erst nach dem Produzieren des Chips ein Fehler im Design gefunden, so kann dieser nicht mehr nachträglich aus den ASICs entfernt werden. Sind im ASIC auch Bestandteile zur analogen Signalverarbeitung integriert, nennt man diesen Mixed-ASIC.

Ein DSP ähnelt in seiner Struktur mehr einer CPU als einem ASIC. Er ist programmierbar, besitzt gegenüber der CPU aber spezielle Register und Befehle zur Signalverarbeitung. Somit ist zum Beispiel das gleichzeitige Ausführen einer Addition und einer Multiplikation möglich. Auch verfügt er über eine erhöhte Parallelität durch die Harvard-Architektur (getrennter Daten- und Befehlsspeicher).

Ein FPGA (Field Programmable Gate Array) ist mit einem ASIC vergleichbar, kann aber trotzdem auf Hardwareebene rekonfiguriert werden. Somit wird ein Nachteil der ASICs aufgehoben. Dafür gibt es auf dem FPGA konfigurierbare Logikblöcke (CLBs) auf die ein Hardwaredesign abgebildet werden kann. Durch die freie Programmierung des gesamten Chips ist eine erhöhte Parallelität und Effizienz als bei einer CPU möglich. Ein FPGA eignet sich gut für die Bildverarbeitung. Es ist zum Beispiel möglich einen Kantenerkennungsalgorithmus in einen FPGA zu laden, damit dieser dann innerhalb von wenigen Takten, dafür aber parallel ein ganzes Bild verarbeiten kann. Für kompliziertere Rechnungen gibt es auch FPGAs die DSPs enthalten. Um auch Software ausführen zu können, sind FPGAs auf dem Markt, die bereits Prozessorkerne enthalten. Eine andere Möglichkeit ist, CPU-Kerne auf den CLBs abzubilden.

Um den Unterschied zwischen FPGA und CPU zu vergleichen, soll als Beispiel die Umrechnung vom RGB- zum YUV-Farbraum dienen. Als Umrechnungsformel können, um im Ganzzahlbereich zu bleiben, folgende Formeln [5] dienen:

$$
\begin{aligned}
Y &= ((66 * R + 129 * G + 25 * B + 128)/256) + 16 \\
U &= ((-38 * R - 74 * G + 112 * B + 128)/256) + 128 \\
V &= ((112 * R - 94 * G - 18 * B + 128)/256) + 128
\end{aligned}
$$

Die einzelnen Rechenschritte jedes Farbwertes sind demnach (exemplarisch am Y-Farbwert gezeigt):

$$
\begin{array}{llll}
e_1 = 66R & e_2 = 129B & e_3 = 25B & (1) \\
e_4 = e_1 + e_2 & & e_5 = e_3 + 128 & (2) \\
e_6 = e_4 + e_5 & & & (3) \\
e_7 = e_6/256 & & & (4) \\
e_8 = e_7 + 16 & & & (5)
\end{array}
$$

Die Inidizes der Ergebnisse zeigen die Takte für die CPU an (pro Ergebnis ein Takt). Somit werden ca. 24 Takte pro Pixel benötigt. Der FPGA kann die Operationen, die in einer Zeile stehen, unabhängig von einander ausführen und somit parallel abarbeiten. Hierbei ist die Lösung der Aufgabe in einem Takt möglich, wenn dieser nicht zu hoch gewählt wird. Wenn eine Operation 10ns benötigt (ausgenommen sei die Division mit 256, da Schiebeoperationen keine Zeit benötigen), beträgt die Gesamtzeit 40ns. Hinzu kommt eine Puffer für Signallaufzeiten und ähnliches, so dass der Maximaltakt ca. 20 MHz betragen kann. Eine andere Lösung ist das Verwenden einer Pipeline. Dabei werden die Ergebnisse zwischengespeichert. Somit ist ein Takt ca. 80 MHz (10ns Operationszeit+2ns Signallaufzeit) möglich. Es entsteht so allerdings eine Latenz von 4 Takten.

Für ein 100×100 Bild benötigt die CPU 240000 Takte. Bei der Annahme, dass der FPGA 10 Pixel parallel verarbeitet, benötigt er 1000 bzw. 1004 Takte, je nach Taktrate und verwendeter Lösung.

1.2 Xilinx Virtex-II Pro

Die Boards, die am Lehrstuhl zur Verfügung stehen, enthalten FPGAs von Xilinx. Dies ist eine US-amerikanische Firma. Da sie keine eigene Halbleiterfertigung besitzt, also ihre Chips von anderen Firmen produzieren lässt, nennt man sie auch „Fabless IC Hersteller“.
Xilinx produziert drei Arten von FPGAs, den Coolrunner (mit einfacher Logik), den Spartan (als Low-cost FPGA) sowie den Virtex (für High-End Anwendungen).

Für die Bild- und Signalverarbeitung sind vor allem die Virtex FPGAs interessant. Sie bieten mehr Platz für eigene Logik als die Spartan-FPGAs und haben die Möglichkeit über Multi-Gigabit Transceivers (MGTs) Daten mit Transferraten über 1 Gbit/s zu übertragen.

Der Virtex-II Pro hat als Vertreter der Virtex-Klasse auch die oben genannten Eigenschaften. Er enthält zusätzlich noch ein bis zwei IBM PowerPC-Kerne. Damit ist er für System-on-a-chip Systeme (SoC) und somit auch für Linux besonders interessant. Im FPGA muss also nicht zur eigentlichen Hardware noch eine CPU synthetisiert werden. Der PowerPC 405 besitzt einen 32 Bit breiten Adressbus und kann somit einen einen 4 Gigabyte großen Adressraum verwalten. Er besitzt eine MMU (Memory Management Unit) und kann damit das Umsetzen von physikalischen und virtuellen Adressen verwalten. Für Cachespeicher befinden sich RAM-Module (BRAM) auf dem FPGA. Zur Anbindung von Peripherie an die CPU sieht IBM die CoreConnect-Architektur vor. Dazu gehört der PL- und der OP-Bus.

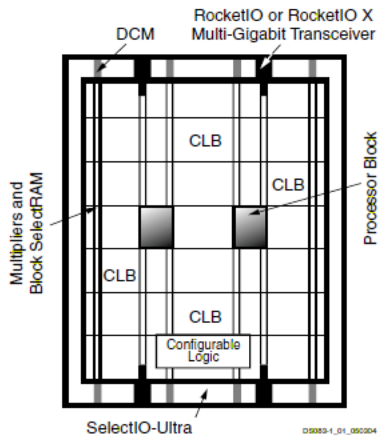

Abbildung 1: Schematischer Aufbau eines Virtex-II Pro [21]

In Abbildung 1 ist ein schematischer Aufbau eines Virtex-II Pro FPGA abgebildet. In der Mitte sind die zwei zuvor genannten PowerPC-Kerne zu sehen. Am Rand befinden sich die MGTs zur Übertragung mit hohen Datenraten. Dort befinden sich auch mehrere DCMs (Digital Clock Management), die zur Taktgenerierung eigener Hardware notwendig sind. Ein synthetisiertes Design wird auf die CLBs (Konfigurierbare Logikblöcke) abgebildet. Jeder CLB enthält vier Slices. Jede Slice wiederum beinhaltet ein FlipFlop, Multiplexer, arithmetrische Logik sowie einen 4 Bit SRAM-Block, der entweder als Distributed RAM, LookUpTable, ROM oder 16 Bit Schieberegister genutzt werden kann. Eine solche Slice ist in Abbildung 2 auf der nächsten Seite zu sehen. Da der SRAM-Block auch für Logik genutzt wird, reduziert sich die Menge an Distributed-RAM je größer ein Design wird. Deshalb gibt es zusätzlich Block-RAM. Auf ihn kann schnell zugegriffen werden und trotzdem werden nur wenige CLBs (für die Logik zum Block-RAM) benötigt. Ein solcher Block-RAM ist in Abbildung 3 auf der nächsten Seite abgebildet. Wie dort auch zu sehen ist, hat ein Block-RAM zwei unabhängige Ports um diesen ansprechen zu können. Der Vorteil besteht darin, dass man zwei Taktdomänen[1] auf diese Art verbinden kann oder asynchrone FIFOs erstellen kann.

[1]In einer Taktdomäne sind alle Komponenten, die die selbe Taktleitung benutzen.

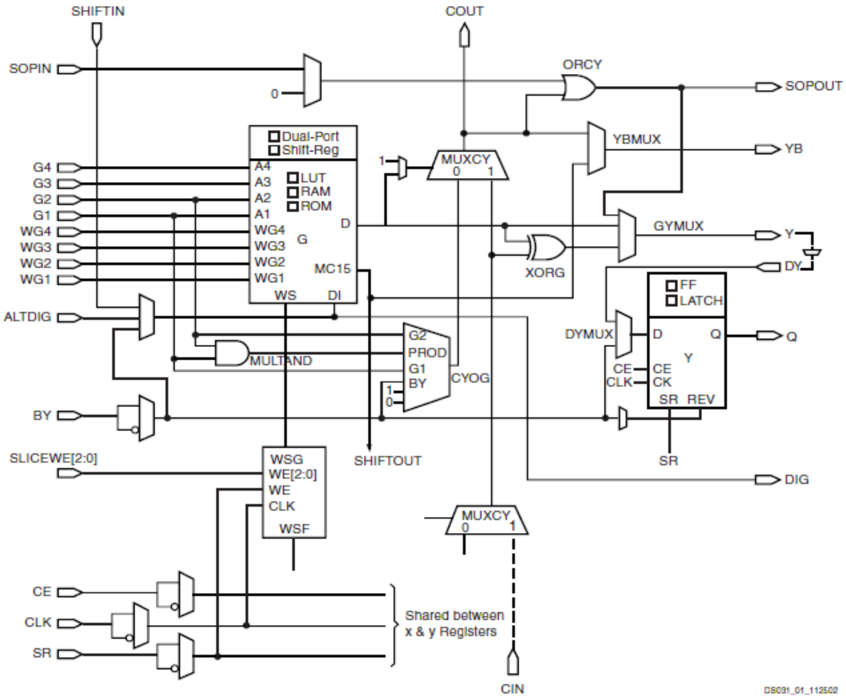

Abbildung 2: Aufbau eines Slices in einem konfigurierbaren Logikblock [21]

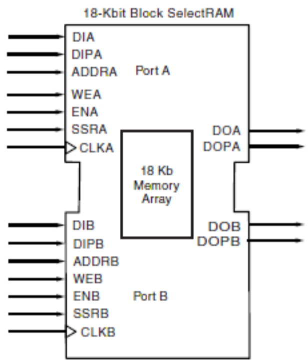

Abbildung 3: Block-RAM [21]

Die folgende Tabelle 1 auf der nächsten Seite zeigt nun die Unterschiede zwischen den einzelnen Virtex-II Pro FPGAs die auf den Boards im Lehrstuhl zur Verfügung stehen. Außerdem ist der zur Zeit größte und kleinste Vertreter mit in die Tabelle aufgenommen worden.

Um die FPGAs zu programmieren, stellt Xilinx das EDK (Entwicklungsumgebungen Embedded Development Kit) und ISE (Integrated Synthesis Environment) zur Verfügung. Letzteres erlaubt das Programmieren eigenen Hardware-Quellcodes, welcher dann synthetisiert und auf den FPGA geladen werden kann. Das EDK wird vor allem dafür genutzt, um integrierte oder synthetisierte Prozessoren an die Peripherie anbinden zu können. Dabei wird die CPU an einen lokalen Bus angeschlossen, der wiederum an so genannte IP-Cores angeschlossen werden muss. IP-Cores sind modulare Hardwareelemente, die im EDK einfach hinzugefügt werden können und nur noch verdrahtet werden müssen, wie zum Beispiel Speicherkontroller, Ethernetkontroller. Diese IP-Cores

Name	MGTs	PowerPCs	Slices	Block-RAM (je 18 Kbit)
XC2VP2	4	0	1.408	12
XC2VP20	8	2	9.280	88
XC2VP30	8	2	13.696	136
XC2VP70	16 oder 20	2	33.088	328
XC2VP100	0 oder 20	2	44.096	444

Tabelle 1: Resourcen verschiedener Virtex-II Pro FPGAs

selbst werden aber wieder in einer Hardwaresprache wie VHDL oder Verlog geschrieben und können im ISE bearbeitet werden. Zum Simulieren des Designs kann ModelSim benutzt werden [11].

1.3 Resourcenvergleich der untersuchten FPGA-Boards

Die in dieser Studienarbeit behandelten Boards sind das Amirix AP120, Alpha Data ADM-XP sowie Xilinx XUPV2P.

Alle Boards besitzen einen FPGA der Xilinx Virtex II Pro Familie. Sie sind immer mit RS232- und Ethernet-Schnittstellen ausgestattet. Auf den Platinen ist außerdem DDR-RAM zu finden. Dieser Speicher ist auch für das Linux notwendig, da die Größe der Block-RAMs für den Linuxkernel und die Umgebung zu klein ist. Die nächsten Unterkapitel befassen sich mit den Unterschieden der einzelnen Boards.

	Amirix AP120	Digilent XUPV2P	Alpha Data ADM-XP [24]
Virtex II Pro	XCVP20	XCVP30	XCVP70
System-ACE	über externen Bus	direkt am FPGA	nicht vorhanden
DDR-RAM	64 Megabyte	256 Megabyte	2× 64 Megabyte
SSRAM	nicht vorhanden	nicht vorhanden	4× 4 Megabyte
ROM	auf allen Boards vorhanden		
RS232	2 mal vorhanden	vorhanden	vorhanden
Ethernet	2 Gigabit PHYs	100 Megabit PHY	100 Megabit PHY
	1 Netzwerkkarte an PCI-Bridge		
Kamera	kein Anschluss	über Framegrabber	CameraLink
weitere Anschlüsse	keine	3 SATA-Ports	RocketIO

Tabelle 2: Unterschiedliche Ausstattung der FPGA-Boards

1.3.1 Amirix AP120

Wie in Tabelle 2 zu sehen ist, verfügt die AP120 über einen Virtex II Pro XCV2P20 FPGA. Dieser hat im Vergleich zu den anderen Boards nur 9280 Slices und 88 Block-RAMs. Damit ist er der kleinste FPGA. In der o.g. Abbildung ist auch zu sehen, dass die Karte zwei serielle Schnittstellen (RS232) und drei Ethernetschnittstellen besitzt. Zwei der drei Netzwerkschnittstellen sind mit PHYs[2] für Gigabit-Ethernet ausgerüstet, jedoch fehlt dafür bisher ein entsprechender IP-Core, um die volle Datenrate nutzen zu können. Bisher können auf allen drei Ports nur 100 Mbit/s genutzt werden. Auch für Linux gibt es nur 100 Mbit/s-Treiber. Die AP120 enthält 64 Megabyte DDR-SDRAM. Während der DDR-SDRAM, die RS232 Schnittstelle sowie zwei der drei Ethernetschnittstellen direkt mit dem FPGA verdrahtet sind, befindet sich auf dieser Karte noch ein externer lokaler Bus, an dem das System-ACE[3] sowie der Flash-ROM zum Booten angeschlossen

[2] PHYs dienen zur Aufbereitung der digitalen Daten für das jeweilige Übertragungsmedium.
[3] Ermöglicht die Konfiguration des FPGAs mittels CompactFlash-Karte

sind. Über eine PCI-Local-Bridge und einer PCI-PCI Bridge ist die Karte mit dem PCI-Bus des PCs verbunden.

Der Karte ist TimeSys Linux, ein Echtzeitlinux, mit beigelegt. Um dies nutzen zu können, gibt es ein Beispielprojekt, genannt *Baseline-Project*.

1.3.2 Digilent XUPV2P

Das XUPV2P wurde für den Lehrbetrieb entwickelt und enthält einen Virtex II Pro XCV2P30. Auf dem Board sind viele Schnittstellen herausgeführt, wie in Tabelle 2 auf der vorherigen Seite zu sehen ist. Sie werden durch die mitgelieferten IP-Cores auch unterstützt. Die Schnittstellen sind direkt mit dem FPGA verbunden. Außerdem enthält es einen DDR-SDRAM-Sockel, so dass eine variable Speichergröße eingesetzt werden kann. Da dieses Board weit verbreitet ist, gibt es auch im Internet viele hilfreiche Quellen, die helfen, wenn man bestimmte Projekte umsetzen will.

1.3.3 Alpha Data ADM-XP

Die ADM-XP ist eine auf einem PCI-Trägerboard befindliche PCI-Mezzanine-Karte. Sie verfügt über mehrere Hochgeschwindigkeitsanschlüsse für differenzielle Datenübertragung an, so dass man etwa CameraLink oder RocketIO anschließen kann.

Sie verfügt über einen Virtex II Pro XCV2P70, wie in Tabelle 2 auf der vorherigen Seite zu sehen. Außerdem sind auf dem Board sechs 256kb SSRAM[4]-Blöcke verfügbar. Zusätzlich sind weitere Anschlüsse über diverse Schnittstellen verfügbar. Die ADM-XP ist mit dem PC über eine PCI-Bridge verbunden.

Eine schematische Darstellung der Busse und der Anschlüsse der Boards sind im Anhang zu finden.

1.3.4 Hardwarevoraussetzungen für ein eingebettetes Betriebssystem

Um ein Betriebssystem auf einer Hardware laufen zu lassen, sind die folgende Grundvoraussetzungen notwendig:

Prozessor Zu allererst benötigt es ein Prozessor, der in der Lage ist, das Betriebssystem auszuführen. Er sollte in der Lage sein, einige Grundfunktionen, wie Speicheradressierung, einfache Rechenoperationen und ähnliches ausführen zu können. Als Beispiel für einen solchen Prozessor sei der, auf dem Virtex-II Pro vorhandene, PowerPC genannt. Um auf FPGAs, die keinen Prozessor enthalten, Software ausführen zu können, gibt es die Möglichkeit CPUs zu synthetisieren. Xilinx bietet zum Beispiel den Microblaze-Prozessor als Alternative an.

Speicher Neben dem Prozessor wird auch Speicher für den Programmcode und den Daten benötigt. Die FPGAs der Virtex Familie haben Block-RAM integriert. Für komplexere Betriebssysteme ist externer RAM notwendig und auf den Boards in Form von DDR-RAM bzw. SRAM vorhanden.

Peripherie Zur Interaktion mit dem Benutzer sollten Schnittstellen zur Verfügung stehen. Als Minimallösung kann die RS232-Schnittstelle dienen, um Benutzeranfragen zu ermöglich und Fehlersuchen zu erleichtern. Einige dieser Schnittstellen werden nachher noch genauer vorgestellt.

1.4 Bussysteme

In diesem Unterkapitel werden verschiedene Busse vorgestellt und darauf geachtet, wie gut sie sich für die Bildverarbeitung eignen. Hier muss vorallem darauf geachtet werden, dass die Datenrate den meist hohen Ansprüchen gerecht wird. Dies ist zum Beispiel essentiell für Videos.

[4] Synchronous Static Random Access Memory

Außerdem wird an den entsprechenden Stellen darauf eingegangen, ob es bereits ein IP-Core verfügbar ist, so dass man den Bus ohne großen Aufwand in die vorhandene Infrastruktur integrieren kann. Ist dies möglich, so kann leicht von einem Betriebssystem darauf zugegriffen werden.

Ein Einsatzgebiet ist die Übertragung von Videos. Je nach Bildgröße, Farbtiefe und Bildrate werden andere Anforderungen gestellt. Der CCD-Chip einer aktuellen Web-Kamera hat $640 \times 480 = 307200$ Pixel (VGA). Je Pixel werden hier 3×8 Bit (für Rot, Grün und Blau) verwendet. Somit ergibt sich ein Datenvolumen von $3 \times 8b \times 307200 \approx 7,3Mbit$ pro Bild. Bei 25 Bildern pro Sekunde wird schon eine Datenrate von $184Mbit/s$ benötigt. Um keine Hochgeschwindigkeitsbusse zur Übertragung nutzen zu müssen, werden die Daten weiter komprimiert. Die im IBB-Projekt genutzte Kamera liefert unkomprimierte 1000×1000 Pixel bei 25 Bildern pro Sekunde. Um diese Datenrate zu bewältigen (600 Mbit/s) wird die CameraLink-Schnittstelle benutzt.

1.4.1 Bussysteme für geringe Datenraten

IEEE 1284 IEEE 1284 ist ein Standard für eine parallele Schnittstelle. Diese wurde Ende der 1960er Jahre in den Wang Laboratories entwickelt und in den Druckern der sich dann abspaltenden Firma Centronics eingesetzt. Erst mit Einführung des IBM PCs 1982 wurde ein Quasi-Standard auf Hostseite geschaffen. IEEE 1284 sieht ein doppelt geschirmtes Kabel mit 18 Aderpaaren vor. Es gibt drei Steckertypen. Der D-Sub-Stecker eines PCs (Typ A) mit 25 Pins enthält aus Platzgründen nicht für jede Signalleitung eine Masseleitung. Der Typ B ist der bekannte Stecker der an die meisten (älteren) Drucker passt. Er ist 36-polig und enthält somit alle Leitungen genauso wie der Typ C Stecker, der sich aber nicht durchsetzen konnte.
Für diese Schnittstelle sind verschiedene Protokolle standardisiert:

- SPP (Standard Parallel Port):
  Es gibt 8 unidirektionale (Host zu Peripherie) Datenleitungen sowie 9 Steuerleitungen. Dieses Protokoll ist nur zum Anschluss eines Druckers geeignet.
  Die maximale Datenrate beträgt 150 Kilobyte pro Sekunde.
- EPP (Enhanced Parallel Port):
  Ende der 1980er haben u.a. Intel, Zenith und Xircom mit EPP die Möglichkeit geschaffen auch andere Geräte anzuschließen. Um dies zu erreichen, sind die Datenleitungen nun bidirektional, so dass auch die Peripherie Daten senden kann. Es gibt weiterhin die 9 Steuersignale.
  EPP erreicht Datenraten von bis zu 2 Megabyte pro Sekunde.
- ECP (Extended Capabilities Port):
  Hiermit sollte der Parallelport zu einer universellen Schnittstelle ausgebaut werden. Entwickelt wurde ECP von Microsoft als Vorläufer von USB und von Hewlett Packard zur Anbindung von Multifunktionsgeräte (Drucker mit Scanner und ähnliches). Höhere Datenraten werden durch eine Kompression erreicht. Diese beträgt maximal 64:1.

Dem Xilinx EDK liegt kein entsprechender IP-Core bei.

One-Wire-Bus Der One-Wire-Bus ist ein von Dalles Semiconductor entwickeltes Produkt zur Übertragung geringer Datenmengen. Es werden 2 Leitungen (Daten sowie Masse) benötigt.
Einige Geräte mit One-Wire-Bus Interface müssen sich mit Hilfe eines im IC enthaltenen 800pF Kondensator über die Datenleitung mit Strom versorgen.

Bei One-Wire können Geräte, die ohne eigene Stromversorgung auskommen, zurückgesetzt werden, indem man die Datenleitung für längere Zeit auf 0 setzt und somit die Stromzufuhr unterbricht. Die Datenübertragung funktioniert so, dass permanent Strom angelegt ist. Um eine Eins zu

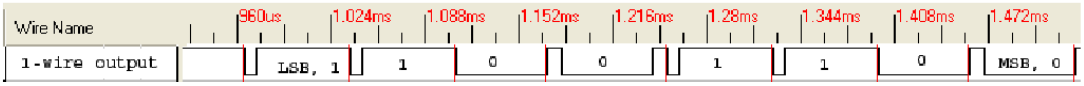

Abbildung 4: Signaldiagramm eines One-Wire Busses

übertragen wird für $5\mu s$ die Stromzufuhr unterbrochen. Für eine Null für $59\mu s$. Im Overdrivemodus werden die Abstände geringer und damit die Datenrate höher. In Abbildung 4 wird das Byte 0x33 auf diese Weise übetragen. One-Wire kann normaler Weise 16,3 kbit/s, im Overdrivemodus bis zu 142 kbit/s übertragen.
Angewendet wird One-Wire zur Abfrage von Sensoren. Jedes Gerät enthält außerdem einen „Unique Identifier“, um es eindeutig identifizieren zu können. Im Beispielprojekt zum Digilent XUPV2P Board wird der Bus zum Übertragen der Seriennummer (aus der dann die MAC-Adresse erstellt wird) genutzt. Daraus ergibt sich auch, dass ein IP-Core verfügbar ist.

CAN-Bus Controller Area Network (CAN) ist ein asynchrones serielles Bussystem welches zur Steuerung von Geräten in Automobilen gedacht ist. Dieser Bus wurde 1983 von Bosch enwickelt und 1985 zusammen mit Intel vorgestellt. Es ist echtzeitfähig und somit auch für sicherheitskritische Anwendungen wie bei Airbags, Bremsen, etc einsetzbar.
Zur Übertragung der Daten werden drei Leitungen benötigt: CAN_HIGH, CAN_LOW sowie CAN_GND, wobei CAN_HIGH immer das Komplementär zu CAN_LOW darstellt.
Der CAN-Bus kann in zwei verschiedenen Modi operieren. Im Lowspeed-Modus beträgt die Datenrate 125kbit/s, im Highspeed-Modus bei 1 Mbit/s.
Dem Xilinx-EDK liegt ein IP-Core mit Evaluationslizenz bei.

RS232/EIA-232 RS232 ist eine Schnittstelle, die seriell[5] Daten überträgt. Die Datenübertragung ist asynchron. Der Empfänger synchronisiert sich durch das Startbit und tastet dann mit seiner Baudrate die weiteren Bits ab. Die 0 entspricht +3V bis +15V und die 1 dementsprechend -3V bis -15V. RS232 überträgt nicht differenziell. Da diese Schnittstelle schon sehr gut etabliert ist, wird sie für diverse Aufgaben verwendet - im Embedded-Bereich vor allem zur Ausgabe von Debuginformationen.
Xilinx liefert eine vereinfachte Schnittstelle (UART-Lite) mit, die nur mit einer festen Baudrate funktioniert. Der UART-Lite-Core ist im EDK enthalten. Ein UART-16550-Core (also eine vollständige RS232-Schnittstelle) liegt nur als Evaluationslizenz bei.

I^{2}C Dieser Bus ist vor allem zur Übertragung geringer Datenmengen gedacht. Dies zeigt sich auch in den Geschwindigkeiten. Vor der Spezifikation 1.0 aus dem Jahre 1992 wurden Datenraten von bis zu 100 kbit/s unterstützt. Danach gab es auch einen schnellen Modus für Datenraten von 400 kbit/s. Mit Version 2.0 (1998) sind Datenraten von 3,4 Mbit/s möglich. I^{2}C benutzt zwei Leitungen SDA (Serial Data) und SCL(Serial Clock). Auf einigen Hochgeschwindigkeitsbussen gibt es Drähte, die mit SDA und SCL beschriftet sind und darauf hindeuten können, dass dort Steuerdaten über einen I^{2}C-Bus übertragen werden.
Es gibt bei diesem Bus einen (oder mehrere) Master sowie verschiedene Slaves (bis zu 1008 Knoten pro Bus). Der Master gibt den Takt durch die SCL-Leitung vor. Die Clients können, sollten sie nicht so schnell arbeiten, durch clock stretching die Geschwindigkeit heruntersetzen.
Eine Datenübertragung beginnt mit einer Start- und endet mit einer Stop-Bedingung. Genutzt wird I^{2}C zum Beispiel bei der Übertragung von Monitorkonfigurationsdaten an den PC (Display Data Channel oder DCC), Einstellen von Tuner-PLLs oder Sensorabfragen.
Dem EDK liegt ein IP-Core mit Evaluationslizenz bei.

Abbildung 5 auf der nächsten Seite zeigt einen typischen Signalverlauf eines I^{2}C-Busses beim Transfer von Daten. Am Anfang gibt es eine Start-Condition. Diese zeigt sich durch ein Signal-

[5] Zur Serialisierung wird ein UART-Baustein genutzt.

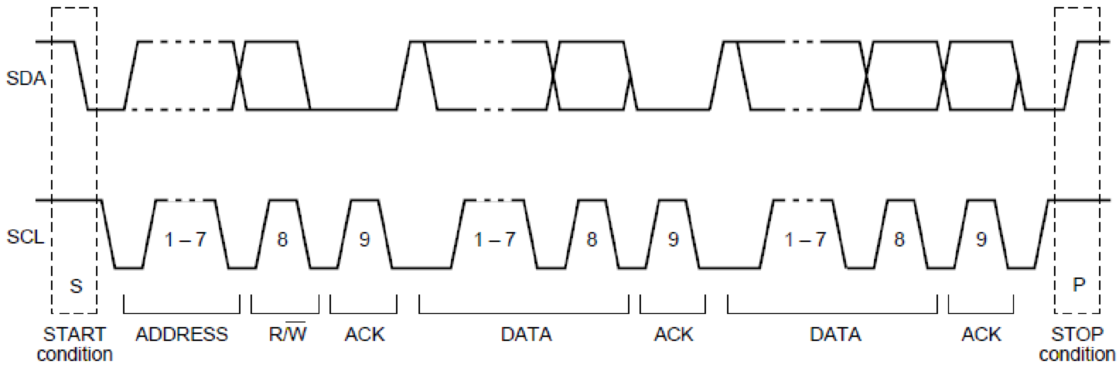

Abbildung 5: Transfer von Daten über den I^{2}C-Bus

wechsel auf SDA während SCL konstant bleibt. Analog funktioniert die Stop-Condition. Nach jeweils 8 Bit sendet der Slave eine Bestätigung.

Zusammenfassung Die o.g. Busse eignen sich nicht zur Übertragung von Bilddaten. Eher sind sie für Steuerungsdaten und Sensorabfragen mit geringen Datenmengen geeignet.

In Tabelle 3 sind die bisher vorgestellten Busse mit ihren Datenraten zusammengefasst.

Bus	par/ser	Geschwindigkeit
IEEE 1284 (SPP)	parallel	1,2 Mbit/s
IEEE 1284 (EPP/ECP)	parallel	16,7 Mbit/s
One Wire	seriell	16,3 kbit/s
CAN (Lowspeed)	seriell	125kbit/s
CAN (Highspeed)	seriell	1 Mbit/s
RS232	seriell	460,8 kbit/s
I^{2}C	seriell	3,4 Mbit/s

Tabelle 3: Übertragungsgeschwindigkeit von Bussen mit geringer Datenrate

1.4.2 Bussysteme für Datenraten über 100 Mbit/s

Die nächsen Busse nutzen LVDS als Übertragungstechnologie. Dies ist kein Bus ansich, sondern eine Technologie, die auf niedrige Spannungen basiert, um höhere Datenraten zu erzielen. Die danach folgenden Busse benutzen teilweise LVDS.
Bis auf Ethernet gibt es von Xilinx keine IP-Cores für die hier vorgestellten Busse. Von Fremdherstellern gibt es jedoch auch weitere IP-Cores.

LVDS Low Voltage Differential Signaling ist eine Schnittstelle, die Datenübertragung durch geringe Spannungen realisiert. Dies verringert bei hohen Datenraten elektrische und magnetische Felder und somit mögliche Fehlerquellen.
LVDS benutzt 2 Leitungen die, wie der Name schon sagt, differenziell gesteuert werden. Es wird eine Stromquelle genutzt und die so entstehenden Spannungen liegen ca. zwischen 1,1 Volt und 1,7 Volt. Um zwischen Null und Eins umzuschalten werden die beiden Leitungen einfach umgepolt.
Die theoretische Maximaldatenrate beträgt 1,923 Gbit/s.

USB Universial Serial Bus (USB) ist ein serieller Bus, der es ermöglichen soll, Hardware aus einem großen Spektrum mit einen PC verbinden zu können. Er wurde von Intel und Microsoft sowie

Compaq, DEC, IBM PC Company, NEC und Northern Telecom definiert. Dieser Bus benutzt zwei Leitungen zur differentiellen Datenübertragung. Zwei weitere Leitungen dienen der Stromversorgung der Peripherie mit maximal 500mA bei 5V.
USB 1.0 hat eine Brutto-Datenrate von 1,5 Mbit/s. In USB 1.1 wurde der so genannte FullSpeed-Modus hinzugefügt, der Datenraten von 12 Mbit/s (auch brutto) ermöglicht. Mit USB 2.0 gibt es noch eine dritte Geschwindigkeit. Im so genannten HighSpeed-Modus können bis zu 480 Mbit/s übertragen werden. Damit ist es möglich auch Festplatten mit hohen Datenraten ansprechen zu können. Dabei liegt die Nettorate von USB 2.0 allerdings nur bei 320 Mbit/s.
USB benutzt zur Übertragung eine Baumstruktur. Der PC ist dabei die Wurzel. Nur er darf einen Datentransfer initiieren. USB-Hubs dienen als Verteiler und sind bis USB 1.1 passive Geräte, die nur die Adressbits (7 Bit) auswerten. Mit USB 2.0 wird jede mögliche Verbindung im HighSpeed-Modus aufgebaut. Demnach müssen Hubs, an denen USB 1.x Geräte angeschlossen sind, die Pakete so transformieren, dass diese Geräte damit zurechtkommen.
USB kennt drei Übertragungsarten, den Bulk-, den isochronen und den Interrupt-Modus.

Im Bulk-Modus werden verpasste oder fehlerhafte Frames bis zu drei Mal erneut gesendet bevor die Datenübertragung als nicht erfolgreich abbricht.

Wohingegen der isochrone Modus dazu dient, Video- und Audiodaten zu übertragen. Hier ist es wichtig, dass die Daten rechtzeitig, jedoch nicht immer korrekt ankommen.

Der Interrupt-Modus ist für Geräte wie Maus, Tastatur und ähnliches gedacht. Diese geben bekannt, in welchen maximalen Zeitabständen sie abgefragt werden wollen. Pro Abfrage können dann bis zu 1 Kilobyte (USB 2.0) übertragen werden.

USB kann Ein- und Zwei-Bitfehler korrigieren. In Abbildung 6 ist der Transfer eines Pakets über USB dargestellt.

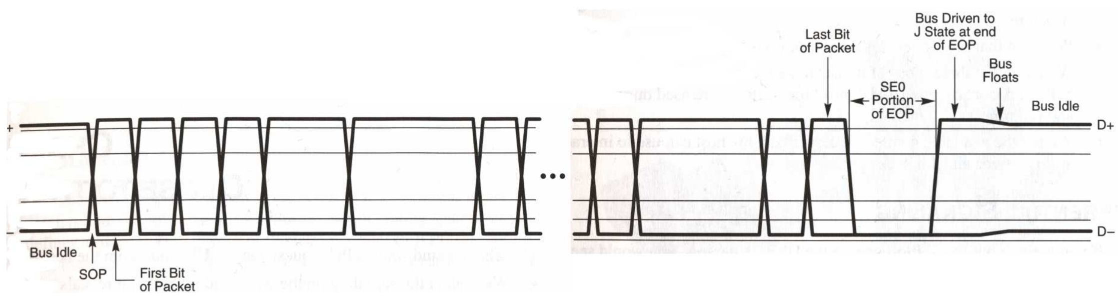

Abbildung 6: Transfer eines Pakets via USB [28]

i.link/Firewire/IEEE1394 Firewire wurde von Apple für den Macintosh als Nachfolger von SCSI entwickelt. Es dient vor allem zum Anschluss externer Geräte (Kamera, Festplatten, etc) an einen Rechner. Es werden bis zu 64 Geräte pro Bus sowie maximal 1023 Busse unterstützt. Dies führt zu einer maximalen Geräteanzahl von 64449 Geräten.
Die Übertragungsgeschwindigkeit der Basisversion liegt bei 98,304 Megabit pro Sekunde (S100). Im Standard sind auch die S200 und das bekannte S400 spezifiziert. Die in dem IEEE1394b-Standard definierten Nachfolger S800, S1600 und S3200 übertragen immer ein Vielfaches der Basisversion. Die Geschwindigkeit von S3200 liegt also bei 3134,728 Mbit/s.
Auf dem Markt erhältlich sind vorallem S400 (Firewire) und S800 (Firewire 800) Geräte.
Firewire ist hotplugfähig und benutzt bis zu 6 Adern. Es werden jeweils zwei Bit parallel und differenziell übertragen (4 Adern). Hinzu kommen 2 Pins zur Stromversorgung der Peripherie (8V-33V/1,5A, max. 48W).
Bei Notebooks ist häufig nur ein 4 Pin Stecker zu finden, so dass die Peripherie nicht den Notebookakku belasten kann.

Rocket IO Bussystem von Xilinx zur schnellen Übertragung von Daten. Datenraten von 600 Mbit/s bis zu 10 Gbit/s mit 4 bis 6 Adern möglich. Davon sind jeweils ein Paar für das Senden (jeweils das differentielle Signal) und ein Paar zum Empfangen notwendig. Bei hohen Geschwindigkeiten kann man zusätzlich noch ein Taktsignalpaar mitschicken. Rocket IO kann als Trägermedium zum Übertragen von verschiedenen Protokollen wie Ethernet oder Aurora benutzt werden.

HDMI High Definition Multimedia Interface (HDMI) ist eine serielle Schnittstelle mit der Audio- und Videodaten digital übertragen werden können. Sie wurde 2003 unter anderem von Hitachi, Panasonic, Philips, Silicon Image, Sony, Thomson und Toshiba für die Unterhaltungselektronik für den Heimgebrauch entwickelt.
HDMI überträgt Bild- und Tondaten mit bis zu 5 Gbit/s (HDMI 1.2) bzw. 10 Gbit/s (HDMI 1.3). Um dies zu ermöglichen werden drei Bits parallel und jedes differenziell übertragen.
Die Stecker haben 19 Pins, jeweils drei Pins pro Datenbit und nochmals drei für den Takt. Für Steuerdaten stehen weitere zwei Leitungen zur Verfügung (SDA und SCL).
Mit diesen hohen Übertragungsraten ist es möglich, 8 Audiokanäle mit einer Auflösung von jeweils 192 kHz bei 24 Bit sowie 25 Bilder pro Sekunde mit einer Auflösung von 1920x1080 (1080p) Bildpunkten mit jeweils bis zu 36 Bit (HDMI 1.2) bzw. 48 Bit (HDMI 1.3) zu übertragen. HDMI 1.3 unterstützt außerdem noch den deutlich größeren Farbraum xvYCC [9][34]. Dabei werden die Daten unkomprimiert übertragen.

SATA Serial Advanced Technology Attachment (SATA) wurde von den Firmen APT, Dell, IBM, Intel, Seagate und Maxtor (SATA 1,5 Gbit/s) bzw. Western Digital, Samsung, Hitachi und Seagate (SATA 3,0 Gbit/s) entwickelt, um Festplatten mit dünneren Kabeln und höheren Geschwindigkeiten als mit ATA anbinden zu können. Dazu nutzt SATA das LVDS. Die Datenraten, die von SATA unterstützt werden, stehen jeweils im Namen. Die häufig genutze Schreibweise SATA 3,0 Gbit/s als SATA II zu bezeichnen ist nicht korrekt. SATA II hieß die Arbeitsgruppe aus der SATA 3,0 Gbit/s hervorging. Es wurden aber auch andere Spezifikationen aus SATA II entwickelt. Die Gruppe selbst nennt sich jetzt Serial ATA International Organizazion (SATA-IO). Die Nettogeschwindigkeit liegt bei 80% der Bruttorate, da eine 8B/10B Kodierung [37] benutzt wird.
Der SATA-Bus besteht aus vier Leitungen bzw. zwei differentiellen Leitungspaaren. Ein Leitungspaar sendet Daten zum Gerät, das andere empfängt Daten. Hinzu kommen noch drei Masseleitungen. Darüber hinaus ist im SATA-Standard auch ein Stecker zur Stromversorgung spezifiziert. Dieser hat jeweils drei Pins, diese sind für die unterschiedlichen Spannungen (3,3V, 5V sowie 12V) sowie 5 Massepins.

SCSI/SAS Small Computer System Interface (SCSI) wurde 1979 von Shugart Technology entwickelt. SCSI-1 hat einen 8 Bit breiten Bus und kann synchron mit 5 MByte/s sowie asynchron mit 3,5 MByte/s laufen. SCSI-1 verwendete noch die High Voltage Differential Technologie.
SCSI-2 wurde 1989 als Standard verabschiedet. Die Transferrate wurde auf 10 MByte/s (für 8 Bit) erhöht. Durch die zusätzliche Möglichkeit den Bus doppelt so breit zu machen (Wide SCSI, 16 Bit) sind 20 MByte/s möglich. Ultra-SCSI (1992) verdoppelte die Raten noch einmal.
Ultra-2 SCSI (1997) führte dann LVD ein (Low Voltage Differential). Dadurch konnten die Kabellängen wieder erhöht und die Datenrate nochmals verdoppelt werden. Mit Ultra-160 (1999) wurden die Datenraten durch die Nutzung der abfallenden Taktflanke auf 160 MByte/s erhöht. Ab diesem Standard gibt es nur noch 16 Bit breite Busse.
2002 kam dann noch Ultra-320 heraus. Ultra-640 wird nicht mehr weiter entwickelt, da man sich auf Serial Attached SCSI (SAS) konzentriert.
Die eben genannten Varianten sind alle parallel und können pro SCSI-Bus maximal acht bzw. 16 (Wide) Geräte steuern.
Mit SAS wird die parallele SCSI-Schnittstelle abgelöst. SAS kann Daten nur noch seriell und über Punkt-zu-Punkt-Verbindungen an Geräte übertragen, ist aber deutlich schneller. Die Datenrate

beträgt 3 Gbit/s was 384 MByte/s entspricht. Geplant sind auch 6 Gbit/s bzw. 12 Gbit/s.
Eine Neuerung ist auch das Dual Porting. Eine Festplatte kann entweder mit Kanalbündelung an einem SAS-Controller angeschlossen werden oder an zwei unterschiedlichen. Weitere Informationen zu SCSI findet man in [27].

Ethernet Ethernet beschäftigt sich mit den unteren zwei Schichten des OSI-Netzwerk-Modells. Die MAC-Schicht ist bei allen Adaptern ähnlich. Bis zu den 100 Megabit Geräten wurden mehrere Geräte in einer Kollisionsdomäne unterstützt. Ab einem Gigabit ist dies nicht mehr möglich. Die weitverbeiteste Form der Datenübertragung findet mittels TwistedPair-Kabel statt. Dort sind jeweils zwei Adernpaare je Richtung vorgesehen. Die Spannungen befindet sich zwischen 0V für logisch 0 und 2,8V für logisch 1.
Um sich auf eine Geschwindigkeit zu einigen hilft das sogenannte AutoNegotiation. Interessant ist auch, dass Ethernet die 4B/5B Kodierung verwendet. Die brutto-Datenrate in einem 100 Megabit-Netz liegt also bei 125 Mbit/s. Weitere Informationen zu Ethernet sind in [32] zu finden.

CameraLink Die CameraLink-Schnittstelle wurde von National Semiconductors entwickelt, um digitale Videodaten seriell zu übertragen. Um dies zu erreichen, wurde das Hauptaugenmerk auf die hohen Datenrate gelegt. Dafür werden bis zu drei Kanäle gleichzeitig benutzt. Jeder Kanal überträgt 24 Bit pro Takt. Die maximale Datenrate beträgt 2,3 Gbit pro Sekunde. Die vierte Datenleitung die in Abbildung 7 zu sehen ist, wird zur Übertragung von Konfigurationsdaten genutzt und werden nicht in der oben genannten Geschwindigkeit übertragen. Außerdem ist noch eine Benutzer-API für Anwendungen in der Bildverarbeitung spezifiziert.

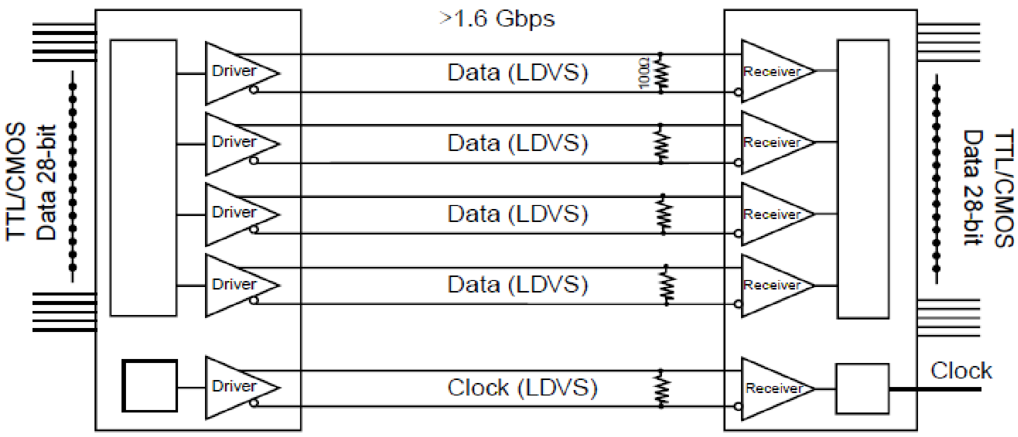

Abbildung 7: Schematischer Aufbau der CameraLink-Schnittstelle [4]

Zusammenfassung Die o.g. Busse sind bis auf USB 1.0 und USB 1.1 für die Übertragung von großen Datenmengen geeignet. HDMI, ein Bus zur Videoübertragung, zeigt, mit welchen Datenmengen gerechnet werden kann. In Tabelle 4 auf der nächsten Seite sind die Datenraten noch einmal aufgeführt.

1.4.3 Busse auf dem FPGA

Zur Übertragung von Daten von und zum Prozessor sowie zur Peripherie gibt es verschiedene Busse im FPGA. Hier wird auf die, für die Arbeit am wichtigsten, Busse hingewiesen. Im EDK unterstützt und vor allem von Xilinx eingesetzt, um den PowerPC mit Hardware zu verbinden, sind der PL- und der OPBus. Beide gehören zu IBMs CoreConnect. Außerdem werden noch der AMBA-Bus, der vor allem bei ARM-Prozessoren zu finden ist, und der Wishbone-Bus vorgestellt. Peripherie,

Bus	Geschwindigkeit
USB 2.0	480 Mbit/s
Firewire S800	800 Mbit/s
Rocket IO	600 Mbit/s - 10 Gbit/s
HDMI	10 Gbit/s
SATA	3 Gbit/s
SAS	3 Gbit/s
Ethernet	10 Mbit/s - 10 Gbit/s
CameraLink	2,3 Gbit/s

Tabelle 4: Übertragungsgeschwindigkeit von Bussen mit hoher Datenrate

die an einen dieser Busse angeschlossen wird, muss mit Hilfe einer Adresse vom PowerPC oder von DMA-fähigen Geräten angesprochen werden.

CoreConnect IBM hat mit der CoreConnect-Architektur ein Bussystem für System-on-a-Chip-Systeme eingeführt. Dieses richtet sich in erster Linie an den PowerPC, an dem man zwei der drei Busse direkt anschließen kann. CoreConnect besteht aus dem PLB (Processor Local Bus), dem OPB (On-Chip Peripheral Bus) und dem DCR-Bus (Device Control Register Bus).

- PLB (Processor Local Bus)
  Der PLB (Processor Local Bus) kann direkt mit dem PowerPC verbunden werden und dient zum Anschließen von zeitkritischer Hardware. Diese ist dann direkt mit dem Prozessor über Adressen erreichbar. Um eine hohe Datenrate zu erzielen sind der Adress- und der Datenbus entkoppelt. Dabei kann der Datenbus zwischen 32 Bit und 128 Bit groß sein (Tabelle 5). Es werden verschiedene Master- und Slave-Geräte an einem Bus unterstützt. Nur Master-Geräte können eigenständig einen Datentransfer anstoßen. Der PL-Bus ermöglicht durch eine zweistufige Schreib- und eine dreistufige Lesepipeline, dass weitere Schreib- bzw. Lesebefehle ausgeführt werden können, während die Ursprungstransaktion noch nicht beendet ist.

Feature	32 Bit	64 Bit	128 Bit
max. Taktrate	33 MHz	133 MHz	183 MHz
max. Bandbreite	264 MByte/s	1,1 GByte/s	2,9 GByte/s

Tabelle 5: Leistung des PL-Busses [24]

- OPB (On Chip Peripheral Bus)
  Der OPB (On Chip Peripheral Bus) wird über eine „PLB to OPB“-Brücke mit dem PL-Bus verbunden. Der Vorteil des OPB liegt in der Möglichkeit ihn über eine „LMB to OPB“-Brücke indirekt auch an einen Microblaze-Prozessor-Bus anschließen zu können. Somit ist es möglich die selben OPB-IP-Cores mit dem PowerPC als auch mit dem Microblaze zu verbinden. Durch die Brücke als auch durch den meist geringeren Takt des OP-Busses werden hier vor allem langsamere Geräte wie RS232, LEDs oder DIP-Schalter angeschlossen. Es werden 32 Bit Adress- und Datenleitungen zur Verfügung gestellt. Im Gegensatz zum APB gibt es hier auch die Möglichkeit mehrere Master-Geräte anzuschließen. OPB-Master können über eine „OPB to PLB“-Brücke auf PLB-Slaves zugreifen.

- DCR (Device Control Register)
  Mit Hilfe des DCR-Busses ist es möglich spezielle Steuerungsregister von Peripherie, die den DCR-Bus unterstüzt, direkt vom Prozessor aus zu beeinflussen. Der Vorteil liegt darin, dass weder der OP- noch der PL-Bus mit diesen (meist geringen) Datenmengen belastet werden. Der PowerPC 405 besitzt zur Ansteuerung des DCR-Busses spezielle Befehle.

Wishbone Whishbone ist ein Bus der aus der Open-Source Szene kommt. Sein Quellcode ist offen und für jeden zugänglich. Deshalb ist er für Entwickler von freien IP-Cores sehr gut geeignet. Bei den unter www.opencores.org verfügbaren Cores wird auf die Kompatibilität zum Wishbone-Bus geachtet.
Wishbone unterstützt 8 Bit, 16 Bit sowie 32 Bit breite Busse. Um Wishbone mit dem Xilinx EDK nutzen zu können, gibt es einen Wishbone-Wrapper, der auch auf der eben genannten Adresse heruntergeladen werden kann.
Im Rahmen der Arbeit wurde auch Wishbone in ein Projekt integriert. Weitere Informationen dazu sind im Implementierungsteil zu finden.

AMBA AMBA (Advanced Microcontroller Bus Architecture) wurde 1996 vom Hersteller ARM eingeführt. Er wird vor allem in FPGAs mit ARM-Prozessoren genutzt, um diese mit der Peripherie zu verbinden. Die AMBA-Spezifikationen Version 3 sehen die drei Buse APB (Advanced Peripheral Bus), AHB-Lite (Advanced High-performance Bus-Lite) und AXI (Advanced eXtensible Interface) vor.

- AXI (AMBA advanced eXtensible Interface)
  An diesem Bus sind Geräte angeschlossen, die eine Datenrate sowie geringe Latenzen benötigen. Pro unidirektionalen Schreib- und Lesekanal gibt es einen Adresskanal. Das AXI untersützt bis zu 16 Burst-Transfers in Folge. Des Weiteren gibt es Erweiterungen, um energiesparende Betriebszustände zur Verfügung zu stellen. Außerdem werden mehrere Master- und Slavegeräte am AXI-Bus unterstützt.
- AHB-Lite (AMBA advanced High-performance Bus-Lite)
  Im Unterschied zum AXI werden am AHB-Lite Bus Geräte angeschlossen, die zwar eine hohe Bandbreite erfordern, aber die Vorteile von Abarbeitungspipelines nicht nutzen können. Dazu gehört zum Beispiel externen und interner Speicher.
- APB (AMBA advanced Peripheral Bus)
  Dieser Bus ist ähnlich dem OPB und dient zur Ansteuerung von Geräten mit hoher Latenz und geringen Datenraten. Dazu gehört die RS232-Schnittstelle oder zum Beispiel die Tastatur. Der APB unterstützt kein Burst-Transfer und nur einen Master am Bus.

Das folgende Kapitel beschäftigt sich mit dem Linux, der notwendigen Infrastruktur und stellt einige Linuxdistributionen vor.

2 Embedded Linux für den PowerPC 405

Linux ist ein freies Betriebssystem, das ursprünglich von Linus Torvalds entwickelt wurde. Da es unter der GNU Public License steht, darf jeder den Quelltext benutzen und verändern. Inzwischen gibt es viele weitere Entwickler, die mit dem Linuxprojekt beschäftigt sind. Linux ist ein unixartiges Betriebssystem.

Für eingebettete Systeme gibt es verschiedene Anbieter von Komplettpaketen. Diese Pakete enthalten meist einen normalen Linuxkernel, eine Linuxumgebung, C-Bibliotheken sowie eine Entwicklungsumgebung inklusive Cross-Compiler.

Der Linuxkernel bildet eine Schicht zwischen der Hardware und der einsetzbaren Software (Hardware-Abstraction Layer). Zur Kommunikation zwischen den beiden Ebenen dient eine einheitliche API (Application Programming Interface). Dies hat den Vorteil, dass ein Programmquelltext nicht verändert werden muss, wenn die Rechnerarchitektur gewechselt wird. Der Linuxkernel ist monolithisch, d.h. er ist, im Gegensatz zu einem Microkernel, nicht in verschiedene Prozesse aufgeteilt. Alle wichtigen Funktionen (Scheduling, Prozessverwaltung, etc) sind als Routinen im Kernel enthalten und somit keine seperaten Prozesse. Abweichend davon ist es aber möglich, so genannte Kernelmodule nachzuladen, um die Funktionalität zu erweitern [35]. Für den eingebetteten Bereich wird der Kernel nicht komplett geändert, sondern mit Zusatzfunktionen erweitert. Zum Beispiel wird die Unterstützung für weitere Prozessortypen und weitere Hardware integriert. Andere Möglichkeiten liegen im Weglassen nicht benötigter Funktionen, um den Kernel speicherplatzschonend zu gestalten. Darauf aufsetzend gibt es die C-Bibliotheken, die unter anderem die Funktionen des Kernels an eigene Programme zur Verfügung stellen. Für den Start des Linux und zur Bedienung werden diverse Konfigurationsdateien, Startskripte sowie Werkzeuge benötigt. Dieses wurde in der Arbeit unter „Linuxumgebung“ zusammengefasst.

Um eigene Programme für ein eingebettetes Linux schreiben zu können, aber auch zum Übersetzen des Kernels wird ein Cross-Compiler benötigt. Mit diesem kann man Programme auf einem anderen Rechner für das eingebettete System compilieren.

Wie man hier sieht, sind die Unterschiede zwischen einer klassischen Linux-Distribution für den Desktop-PC oder Server und einem Linux für eingebettete Systeme gering. Insbesondere ist anzumerken, dass sich im FPGA-Bereich die Anpassungen an den Kernel - solange man keine Echtzeitanforderungen hat - in Grenzen halten. Hier steht genug freier Arbeitsspeicher (in Form von DDR-RAM, mehr als 4 Megabyte) zur Verfügung und durch System-ACE ist auch für die Umgebung genügend Platz vorhanden. Im frei erhältlichen Linuxkernel für den PowerPC 405 [10] ist sowohl die Unterstützung für den PowerPC 405 als auch für diverse Xilinx IP-Cores gegeben.

2.1 Hardwareanforderungen an ein Linux

Im Gegensatz zu anderen Betriebssystemen ist Linux sehr anpassungsfähig. Trotzdem benötigt es Grundhardware, die entweder nötig ist, oder die Bedienung erheblich erleichtern kann:

- Prozessor
  Zum Ausführen des Kernels

- Externer Arbeitsspeicher
  Dieser ist notwendig, da die Block-RAM Größe auf den Virtex FPGAs nicht ausreichend ist, um den Linuxkernel sowie Anwendungen zu halten.

- RS232-Schnittstelle (optional)
  Diese ist zwar nicht notwendig, hilft jedoch beim Debuggen und der Bedienung von Software. Alternativ kann man natürlich auch über telnet, ssh oder ähnliches auf das Linux zugreifen.

- Massenspeicher (optional)
  Um zusätzliche Software zu starten, Konfigurationsdateien zu speichern und ähnliches benötigt Linux einen nicht flüchtigen Speicher. Dieser kann natürlich entfallen, wenn nur der Kernel genutzt, oder die Daten über Netzwerk geladen werden. Auf einigen Xilinx-FPGA

Boards steht das System-ACE zur Verfügung und kann mit Hilfe des MontaVista-Kernels angesprochen werden.

2.2 Linuxvarianten

Im folgenden Abschnitt werden Softwareprojekte vorgestellt, die quelloffen sind und für jedermann zur Verfügung stehen. Danach werden Produkte einiger kommerzieller Anbieter beschrieben. Diese nutzen aber meist auch die frei verfügbaren Softwareprojekte oder unterstützen diese sogar.

2.2.1 Quelloffene Softwareprojekte

PowerPC Linuxkernel Der Linuxkernel 2.4 ist nicht kompatibel zum PowerPC 405. Daher gibt es im Internet ein Projekt [10], dass den Standardkernel angepasst hat, um ihn mit den Prozessoren zum Laufen zu bringen. Unterstüzt wird dieses Projekt von MontaVista, so dass sich auch viele Xilinx-Hardwaretreiber wiederfinden.

BusyBox BusyBox [3] ist ein Projekt, welches keinen Kernel enthält sondern nur die Linuxumgebung. Das Konzept besteht darin, alle wichtigen Programme in einer binären Datei zusammen zufassen. Möchte man ein Programm aufrufen, so geschieht das über einen Link der auf die BusyBox-Datei zeigt und den Namen des aufzurufenen Programms hat. Um also den Befehl „cat“ aufrufen zu können, wird ein Link von /bin/cat nach /bin/busybox gelegt und ausgeführt. Weitere Programme können natürlich dem Dateisystem hinzugefügt, aber nicht in die BusyBox-Datei integriert werden. BusyBox wird auch von einigen kommerziellen Anbietern genutzt, um ihre Linuxumgebung zu erstellen. Zum Beispiel läuft auch ein BusyBox auf der AVM Fritz!Box.

Dan Kegels Crosstools-Skript Mit Hilfe dieses Skriptes [29] ist es sehr einfach möglich, sich einen Crosscompiler sowie die Standard-C-Bibliotheken für den PowerPC zu erstellen.

Cross Linux from the Scratch Cross Linux From The Scratch [6] basiert auf Linux From The Scratch. Bei dieser Distribution schafft sich der Benutzer seine eigene Umgebung in dem er selbst jeden Schritt bei der Installation begleitet und jedes Paket selbst compiliert. Man hat durch diese reichen Einflussnahmen die Möglichkeit, das Linux genau so zu konfigurieren, dass es zur entsprechenden Hardware passt und genau den gewünschten Softwareumfang enthält. Cross Linux From The Scratch beschreibt, wie man eine Umgebung für eine andere Architektur erstellt. Für den PowerPC gibt es eine ausführliche Anleitung.
Durch die Notwendigkeit jede Software (darunter fallen auch kleine Werkzeuge wie find, tar, gzip, etc) selbst zu übersetzen, ist die Installation mühselig. Da die Programme für den PowerPC übersetzt werden müssen und dies nicht bei jedem Werkzeug problemlos klappt, ist es an einigen Stellen nötig selbst die Software zu modifizieren. Das System ist danach zwar individuell an die eigenen Wünsche angepasst, kann aber auch nur noch schwer verändert werden und ist schwerer wartbar.

uClibC Das uClibC-Projekt [17] enthält nur eine alternative C-Bibliothek. Diese ist bedeutend kleiner und enthält trotzdem die meisten Funktionen der glibc. Für Systeme mit wenig Arbeitsspeicher ist das sehr hilfreich. Die uClibC ist im Gegensatz zur Standard-C-Bibliothek nicht auf Performance ausgelegt. Auch wurden einige Features entfernt. Des Weiteren kann man einzelne Module wie zum Beispiel Remote Procedure Call hinzufügen, falls man sie wirklich braucht. Dies spart Platz.

uClinux uClinux [18] verwendet die eben genannte uClibC und einen Kernel der auf Prozessoren läuft, die keine Memory Management Unit (MMU) besitzen. Dieses Linux wird häufig auf eingebetteten Geräten verwendet, die wenig Speicher zur Verfügung haben. Der PowerPC 405 im Virtex II Pro besitzt hingegen eine MMU und ist somit nicht auf diesen speziellen Kernel angewiesen.

2.2.2 Kommerzielle Anbieter

Monta Vista Linux Monta Vista ist eine Firma die Linux für diverse Hardware anpasst. Da diese Firma unter anderem auch eng mit Xilinx zusammen arbeitet, gibt es spezielle Versionen für einige Xilinx Boards. Darunter sind das ML300 und das ML401. Um die Funktionalität dieser Boards zu verbessern hat Monta Vista eigene Treiber für verschiedene IP-Cores geschrieben. Darunter sind Treiber für die UARTLite Schnittstelle, dem System-ACE, dem Xilinx Ethernet Core, dem Framebuffer Core und einige weitere Hardware. Diese Treiber sind als Open Source verfügbar und demnach auch ohne MontaVista Linux nutzbar.
Zum Ausprobieren gibt es auf der MontaVista Homepage ein PreviewKit [12]. Als Host-PC können sowohl Cygwin [7] als auch ältere, unterstützte, RedHat und SuSE Distributionen genutzt werden. Das aktuelle MontaVista-Linuxpaket enthält einen 2.6er Linuxkernel. Die Entwicklungsumgebung DevRocket basiert früher auf Eclipse und ist heute eine Samlung von Plugins für Eclipse [8].

Beam FPGA-Linux Dies ist eine spezielle Linuxdistribution, die nur für die Alpha Data ADM-XPL gedacht ist. Geliefert wird eine Bit-Datei, die die Hardwarekonfiguration enthält und das Root-Dateisystem. Es wird zusätzlich ein Monta Vista Linux Kernel benötigt. Da man, durch fehlen zusätzlicher Quellen, keine eigene Hardware integrieren kann und da das Bitfile nur für die ADM-XPL gedacht ist, ist eine Erweiterung auf andere Hardware sowie ein Einfügen eigener Hardware nicht möglich [2].

TimeSys Linux TimeSys nutzt den PowerPC Linuxkernel und erweitert diesen um Echtzeitfähigkeiten. Das mit der Amirix-Karte ausgelieferte Linux ist ein Timesys-Linux. Im Unterschied zu anderen Anbietern ist TimeSys Linux modular aufgebaut. Je nach Wunsch des Kunden können weitere Module hinzugefügt werden. Dadurch, dass bekannt ist welches Linux und welcher Cross-Compiler zum Einsatz kommt, können bereits vorkompilierte Pakete bereitgestellt werden [15].

Kommerzielle Embedded Linux Anbieter nutzen vor allem Software aus der OpenSource-Szene. Durch die Erweiterungen und große Hardwareunterstützung eignen sie sich vor allem für Firmen, die ein fertiges Linux einsetzen wollen, ohne selbst viel Aufwand treiben zu müssen.

2.2.3 QNX Neutrino Realtime OS als Alternative

Das QNX Neutrino [14] ist ein Echtzeitbetriebsystem, welches den POSIX-Standard erfüllt. D.h. die meisten C-Funktionen haben dieselbe Syntax. Es gibt ein Board Support Package für das Xilinx ML300, ein Board mit dem Virtex-II Pro.

2.2.4 Betriebssystemlose Umgebung

Anstelle eines der oben genannten Betriebssysteme, besteht auch die Möglichkeit, die im Xilinx ISE bzw. EDK enthaltenen C-Bibliotheken zu nutzen. Die Funktionsnamen ähneln dabei denen in Unix-Betriebssystemen. Damit kann man ohne Betriebssystem Programme schreiben, die direkt auf der Hardware laufen. Die Benutzung von C mit den Xilinx-Bibliotheken und ohne Betriebssystem hat verschiedene Vor- und Nachteile.
Ein Vorteil besteht darin, dass man die Hardware durch Weglassen von - für Linux notwendigen - Elementen (Interrupt-Controller, DDR-RAM, etc), deutlich vereinfachen kann. Bei kleinen Programmen wird als Speicher nur Block-RAM benötigt. Dadurch sinkt die Komplexität zum Ansprechen der Hardware infolge des nicht mehr vorhandenen Speicherschutzes des Betriebssystems. Des Weiteren stellt Xilinx diverse Treiber für die IP-Cores bereit, die durch Einbinden der Header-Dateien verwendet werden können. Der Ansatz, kein Betriebssystem zu verwenden, hat aber auch Nachteile. Möchte man kompliziertere Software schreiben oder benötigt man Grundfunktionen eines Betriebssystems (Multitasking, Multiuser, Scheduling, etc) so ist dies in reinem C nicht mehr einfach möglich. Zum Beispiel benötigt ein Webserver einen TCP/IP-Stack sowie ein Programm zur Verabreitung der HTTP-Anfragen und einen Treiber für die Netzwerkschnittstelle.

Möchte man, dass zeitgleich mehrere Benutzer auf den Webserver zugreifen können, helfen die Funktionen eines Betriebssystems sehr. Dort ist es mit einfachen Mitteln möglich einen Webserver aufzusetzen. Für Linux gibt es gar schon fertige Server, so dass man sich nicht weiter darum kümmern muss. Ein anderer Punkt ist, dass die Hardware, die vom Betriebssystem unterstützt wird, einfacher zu nutzen ist. Vor allem der Umstand, dass schon fertige und meist ausgereifte Treiber zur Verfügung stehen, helfen dabei sehr. Xilinx liefert zu manchen seiner IP-Cores auch Treiber für den Linuxkernel mit (zum Beispiel für Ethernet, UARTLite und System-ACE).

Das nächste Kapitel beschäftigt sich mit er Wahl der Hardware und der Umsetzung eines geeigneten Linux auf dieser.

3 Implementierung

Dieses Kapitel beschreibt die Implementierung des Linux auf den vorgestellten Boards. Es wird dabei auf das Hardwaredesign wie auch auf die Softwarekonfiguration eingegangen.

Als erstes muss ein geeignetes Hardwaredesign für das jeweilige Board aufgebaut werden, so dass das Linux richtig mit der angeschlossenen Hardware agieren kann. Für die Alpha Data Karte wurde auf die Studienarbeit von Maxililian Buder [24] zurückgegriffen. In dessen Hardwareprojekt wurden bereits die wichtigsten Komponenten (SDRAM, Ethernet sowie SSRAM) angebunden.

Dem Amirix AP120 Board liegt das Baseline-Projekt bei, welches für das mitgelieferte TimeSys-Linux eine angemessene Plattform bereitstellt. Dieses Design wurde so modifiziert, dass der PowerPC-Kernel ohne Erweiterungen von TimeSys auf dem Board läuft. Dazu musste das Baseline-Projekt erheblich modifiziert werden. Einige Komponenten stehen nicht zur Verfügung, da die Beschreibung der Bridge, die den FPGA mit externer Hardware sowie der PCI Bridge verbindet, nicht vorlag. Daher wurde auf eine der zwei Gigabit-Ethernetschnittstellen zurückgegriffen. Das, über den externen Bus, angebundene System-ACE funktionierte erst durch Modifikation des *Board Support Packages*.

Dem XUPV2P liegt nur ein Beispieldesign mit dem uClinux bei, welches allerdings nicht die vorhandenen PowerPC Prozessoren nutzt, sondern einen Microblaze Prozessor synthetisiert. Da dessen Busstruktur mit dem des PowerPCs nicht übereinstimmt, wurde das Design verworfen und ein eigenes entwickelt. Dabei muss darauf geachtet werden, dass zeitkritische sowie zugriffsintensive Komponenten direkt an den PL-Bus angeschlossen werden. Zeitunkritische Hardware, wie die serielle Schnittstelle, wird dann an den OP-Bus angeschlossen.

In der Beispielimplementation wurden keine kommerziellen Produkte genutzt, sondern auf verschiedene oben genannten OpenSource-Projekte zurückgegriffen:

- dem Crosstool-Skript von Dan Kegel zum Erstellen eines GNU-C Crosscompilers und der C-Bibliotheken [29]
- der BusyBox von Denis Vlasenko als Linuxumgebung für den PowerPC [3]
- dem Linux Kernel 2.4 für den PowerPC 405 mit Treibern für Xilinx Hardware [10]

3.1 Das Hardwareprojekt

Um vergleichbare Daten zu erhalten, wurde für jedes Board ein Projekt erstellt, das nur die notwendige Hardware enthält, die Linux benötigt. Grundlage ist jeweils ein Prozessor, der zum einen an der JTAG[6]-Schnittstelle und zum anderen mit der CoreConnect-Architektur, bestehend aus PL- und OP-Bus, angeschlossen ist. Der PL-Bus (als Slavegerät) ist über eine Brücke mit dem OP-Bus (als Mastergerät) verbunden. Am PL-Bus wird der externe Arbeitsspeicher (DDR-RAM) sowie die Ethernetschnittstelle angeschlossen. Mit dem OP-Bus ist in jedem Fall die UARTLite[7]-Schnittstelle verbunden.
Da teilweise unterschiedliche IP-Cores verwendet wurden, differiert auch der absolute Resourcenverbrauch.

Tabelle 6 auf der nächsten Seite zeigt die verwendeten IP-Cores auf den jeweiligen Boards mit Angabe der benötigten Hardwareressourcen. Es wird auch der Gesamtverbrauch der einzelnen Projekte angegeben.

Amirix AP120 Da sich bei der AP120 ein externer Bus auf der Karte befindet und sich an diesem das System-ACE befindet, wurde er mit in das Projekt eingebunden. Dafür wurden IP-Cores plb2opb_guardian_bridge und opb_ext_bridge aus dem Amirix Baseline-Projekt weiterverwendet und die OPB2PLB-Bridge verwendet. Dies ist notwendig, um Geräten am externen Bus den direkten Speicherzugriff auf den PLB-DDR-RAM-Controller zu ermöglichen. Des Weiteren hatte

[6] Joint Test Action Group, ein Verfahren zum Testen und Debuggen von elektronischer Hardware
[7] Eine UART-Schnittstelle mit fester Baudrate

	Amirix AP120	Xilinx XUPV2P	AlphaData ADM-XP
PowerPC	1 PowerPC mit aktiver MMU		
PL-Bus	460 Slices (2 Master)	241 Slices (1 Master)	443 Slices (2 Master)
OP-Bus	202 Slices (2 Master)	48 Slices (1 Master)	48 Slices (1 Master)
PLB2OPB	538 Slices (von Amirix)	508 Slices	509 Slices
OPB2PLB	555 Slices	nicht verwendet	
OPB2EXT	362 Slices (von Amirix)	nicht vorhanden	
JTAG	2 Slices		
Uartlite (115200 8N1)	49 Slices	51 Slices	51 Slices
Ethernet	1726 Slices, 4 Block-RAMs	1710 Slices, 4 Block-RAMs	1643 Slices, 4 Block-RAMs
System-ACE-Contr.	in Hardware vorhanden	136 Slices	kein SystemACE vorhanden
DDR-RAM-Contr.	735 Slices (v. Amirix modifiziert, PLB)	957 Slices (PLB)	513 Slices (OPB, 2x)
Block-RAM-Contr.	198 Slices	254 Slices	200 Slices
Block-RAM	32 Block-RAMs		
SSRAM-Contr.	nicht vorhanden		566 Slices (4x)
LB2PLB	nicht vorhanden		2245 Slices, 1 Block-RAM
Gesamtverbrauch			
Slices	4947	3964	10437
Block-RAM	36	36	61

Tabelle 6: Verwendete IP-Cores mit Resourcenangaben

Amirix den DDR-RAM-Controller von Xilinx modifiziert, damit dieser mit dem AP120-Board arbeitet. Da der System-ACE-Controller bereits in Hardware verfügbar ist und an dem externen Bus angeschlossen ist, muss dieser nicht mehr ins Projekt integriert werden.

Digilent XUPV2P Im XUPV2P Design wurde der System-ACE-Controller hinzugefügt, um darüber das Linux von der CompactFlash-Karte booten zu können.

AlphaData ADM-XP Hierfür wurde auf das Projekt von Maximilian Buder [24] zurückgegriffen, dass er für seine Studienarbeit erstellt hatte. Im Unterschied zu den anderen ist der DDR-RAM über den OP-Bus angebunden. Dafür ist der auf der Karte vorhandene SSRAM am PL-Bus. Die AlphaData Karte hat kein System-ACE. Deshalb wurde bisher der Kernel immer manuell hochgeladen. Vom Flash-ROM zum Booten des Kernels wurde noch kein Gebrauch gemacht. Des Weiteren wird der lokale Bus der Karte über die LB2PLB-Bridge angebunden.

3.2 Linux

Um recht zügig ein lauffähiges Linux zu erstellen, scheiden die kommerziellen Varianten aus. Diese haben zwar meist eine komplette Toolchain-Umgebung integriert, welche aber einige Bedingungen an das System stellten (zum Beispiel veraltete Linuxdistribution auf dem Host), die nicht ohne Weiteres erfüllbar waren. Außerdem sind es meist nur Evaluationspakete, die nach einer bestimmten Zeit nicht mehr arbeiten würden. Des Weiteren kam bei der Recherche heraus, dass diese Anbieter zum großen Teil auch auf OpenSource Software setzen.

Da die Dokumentation für die Installation eines Linux für das XUPV2P-Board sehr ausführlich ist [31], wurde der dort beschriebene Dreisatz aus BusyBox [3], Standard-C-Bibliothek (glibc) und PowerPC-Linuxkernel 2.4 verwendet.

Um so ein Linuxsystem aufzubauen sind mehrere Schritte notwendig.

- Erstellen des Cross-Compilers und der glibc für den PowerPC
  Das Skript von Dan Kegel [29] lädt alle benötigten Programmquellen herunter und erstellt eine komplette Toolchain. Mit dem dabei entstehenden Compiler kann der Kernel und die BusyBox übersetzt werden. Für den PowerPC 405 gibt es ein spezielles Skript `demo-powerpc-405.sh`, welches angepasst werden muss. Die hier verwendete Version ist im Anhang zu finden.

- Erstellen des *Board Support Packages*
  Mit Hilfe des EDKs wurde nun zu den jeweiligen Projekten ein *Board Support Package* erstellt. Dazu muss im Programm unter „Software Platform Settings“ unter „OS“ „linux_ml31“ ausgewählt werden und danach unter „OS and Library“ diverse selbsterklärende Variablen gesetzt werden. Das *Board Support Package* enthält die Hardwareinformationen, die nötig sind, um den Linuxkernel zu übersetzen. Dazu gehören die Hardwareadressen, die Taktraten der einzelnen Busse, Interrupt-Leitungen sowie weitergehende Informationen zu einzelnen Hardwarekomponenten, wie die Baudrate der UART-Lite-Schnittstelle. Außerdem befinden sich die jeweiligen Treiber zur Xilinx-Hardware (Ethernet, UART-Lite, System-ACE, etc) in diesem Paket.

- Erzeugen eines Linux-Kernels
  Zu erst wird das *Board Support Package* über die Verzeichnisstruktur des Linuxkernels kopiert. Danach müssen noch Änderungen an mehreren Dateien vorgenommen werden. Diese Änderungen sind im Anhang beschrieben. Ebenso ist dort eine funktionierende Kernel-Konfigurationsdatei zu finden.

- Übersetzen der BusyBox
  Die BusyBox kann nun einfach übersetzt werden. Eine entsprechende Konfigurationsdatei ist im Anhang zu finden.

- Erstellen des Root-Dateisystems
  Zum Erstellen des Root-Dateisystems wurde das mkrootfs-Skript [30] verwendet.

Nachdem der Kernel erstellt wurde, gibt es mehrere Möglichkeiten ihn zu booten.

- Manuelles Laden
  Der am schnellsten verwendbare Weg ist, zuerst die Hardware auf den FPGA zu laden, um danach mittels Debug-Konsole eine Verbindung zum Prozessor herstellen zu können. Nun muss man über den Prozessor den Kernel in den DDR-RAM kopieren und von dort starten. Dieser Weg ist relativ problemlos aber auch unbequem und umständlich.

- System-ACE
  Der auf zwei der drei Boards verfügbare System-ACE Controller kann direkt mit dem Prozessor kommunizieren. Schiebt man eine Compact-Flash-Karte in den entsprechenden Einschub, so konfiguriert der Controller den FPGA, verbindet sich mit dem dann verfügbaren Prozessor und kopiert die Software (den Kernel) in den DDR-RAM. Er setzt auch selbständig den Program Counter auf die Startadresse und bootet den Kernel [33]. Diese Variante ist sehr angenehm, bedingt allerdings auch das Vorhandensein des Controllers. Ein weiterer Vorteil ist, dass die Karte auch gleich als Massenspeicher benutzt werden kann.
  Eine Compact-Flash Karte muss entsprechenden vorbereitet und formatiert werden. Für den System-ACE Controller muss eine FAT16 Partition mit 64 Sektoren pro Cluster erstellt werden. Um als Massenspeicher genutzt werden zu können, wurden zwei weitere Partitionen für

Linux erstellt (Eine Ext2- und eine Swap[8]-Partition). Nun kann mit Hilfe des TCL-Skripts „genace.tcl“ eine ACE-Datei erstellt und auf die FAT16 Partition kopiert werden.

- Spezieller Bootmanager
  Es gibt spezielle Bootmanager, wie U-Boot [16]. Diese enthalten einen Urloader, der klein genug ist, um in den Block-RAM zu passen und somit schon in das Hardwaredesign integriert werden können. Danach wird der eigentliche Bootmanager aus dem Flash-ROM geladen. Dieser Bootmanager enthält Treiber, um den Kernel über verschiedene Wege (ROM, Compact Flash, Netzwerk, etc) in den Speicher zu laden.

Nach dem Laden des Linux-Kernels wird die vorhandene Hardware initialisiert. War dies erfolgreich, wird der Init-Prozess gestartet, der selbst wieder ein Startskript ausführt. Dieses fährt dann zum Beispiel die Netzwerkschnittstelle hoch, startet den Webserver oder führt sonstige, beim Start notwenige, Prozesse aus. Zum Schluss wird „login“ geladen, so dass sich der Benutzer einloggen kann.

3.3 Erstellte Beispielprojekte mit Linux

Um weitere Informationen zur Fragestellung der Verwendbarkeit von Linux auf FPGAs zu erhalten, wurde das einfache Projekt erweitert und zusätzliche Hardware „eingebaut“. Dabei wurde sich aber auf das XUPV2P-Board beschränkt. Dies ist möglich, da sich die CPU und die Synthese sich nicht von Board zu Board ändert.

Folgende Hardware wurde hinzugefügt:

- Framebuffer: IP-Core zum Anschluss eines VGA-Monitors
- Auroraprotokoll: IP-Core zur Verwendung der SATA-Schnittstellen zur Datenübertragung
- Wishbone-Bus: IP-Core um Wishbone Cores (OpenCores) an den OP-Bus anschließen zu können
- Webserver

Framebuffer Der Framebuffercore arbeitet als Master auf dem PL-Bus. Er greift selbstständig auf den DDR-RAM, an eine ihm, entweder per DCR-Bus oder als Parameter, angegebene Adresse, zu. Um also ein Bild anzeigen zu lassen, muss an diesem Speicherbereich die Bildinformation gespeichert werden. Dieser Teil des DDR-RAMs sollte außerdem nicht mehr dem Linuxkernel als Arbeitsspeicher zur Verfügung stehen. Das Beschreiben des DDR-RAMs mit einem neuen Bild dauerte im Test mehrere Sekunden. Somit ist es vorteilhaft mehrere Bilder in den DDR-RAM zu kopieren, um dann dem Framebuffercore die neue Startadresse eines Bildes anzugeben. Letzteres funktionierte allerdings auf Grund eines *Bus Errors*[9] nicht.

Aurora Zur Anbindung einer schnellen Datenverbindung an das Linux wurde exemplarisch das Aurora-Protokoll über die SATA-Schnittstellen verwendet. Dazu wurden zwei SATA-Schnittstellen auf dem XUPV2P-Board verwendet, um Daten von einer auf die andere Schnittstelle zu übertragen. Zur Anbindung an den PL-Bus wurden zwei FIFOs verwendet. Ohne Betriebssystem funktioniert die Datenübertragung problemlos.

Bei der Verwendung des Linux wurde das Senden mit einem *Bus Error*-Fehler abgebrochen. Um trotzdem Daten übertragen zu können, ist die Verwendung des zweiten PowerPCs oder von Zwischenspeicher [26] als Puffer denkbar. Damit können dann aber keine hohen Datenraten mehr erzielt werden.

[8] Die Swap-Partition wurde auf Grund der Eigenschaften von Flash-Speicher nicht in Betrieb genommen
[9] Das Problem des *Bus Errors* konnt nicht abschließend geklärt werden, genaueres im nächsten Kapitel

Wishbone Da IBM CoreConnect-Architektur nicht frei ist und für die kommerzielle Nutzung lizensiert werden muss, wurde der Wishbone-Bus [19] als Alternative entwickelt. Viele der frei auf verfügbaren IP-Cores werden mit diesem Interface ausgestattet. Mit dem Wishbone-Wrapper [20] ist es möglich, diesen Bus an den OP-Bus anzuschließen. Der Nachteil der Wishbone-Lösung sind die nicht vorhandenen Linuxtreiber. Somit sind eigene Treiber zu schreiben und in den Kernel zu integrieren. Um den Wishbone Bus nutzen zu können sind einige Vorkehrungen zu treffen, die im Anhang beschrieben sind.

Webserver Diese vergleichsweise einfache Aufgabe wurde ohne Probleme vom Linux übernommen. Dabei konnte auf den vorhandenen httpd-Webserver, der in Busybox integriert ist, zurückgegriffen werden. Mit Hilfe dieses Designs wurden auch die unten genannten Performancemessungen durchgeführt. Der Speicherverbrauch (RAM) beträgt ungefähr fünf Megabyte.

3.4 Performanceanalyse

Es wurde überprüft wie der Prozessor sowie das Linux die Datenraten beeinflussen. Dafür wurde die angebundene Netzwerkschnittstelle genutzt, um Daten mit bis zu 100 Mbit/s zu übertragen. Für den Test wurde netio [13] verwendet. Dieses Programm sendet und empfängt wahlweise mittels UDP- und TCP-Pakete in unterschiedlicher Größe. Hier wurden mehrere Durchläufe nacheinander gestartet. Wie in Tabelle 7 zu sehen ist, wurden jeweils TCP- und UDP-Pakte mit 10 Mbit/s und 100 Mbit/s übertragen. Wie sich herausstellte, ändert sich die Geschwindigkeit nur geringfühgig, wenn die Netzwerkschnittstelle am OP-Bus angeschlossen ist. Durch Verwenden des CPU-internen Caches konnte die Datenrate aber deutlich gesteigert werden. Die Maximalleistung einer 100 MBit-Leitung liegt bei 12,5 MByte/s, die einer 10 MBit-Leitung bei 1,25 MByte/s. Diese wurden unter anderem deshalb nicht erreicht, da die Prozessorlast auf 100% anstieg.

Paketgr.		1		2		4	
Cache		-	√	-	√	-	√
TCP100	Tx	3129	7323	2448	7494	3132	7885
	Rx	3141	5982	3257	6119	4290	6758
UDP100	Tx	3549	4557	3672	4768	2450	6617
	Rx	4005	1484	147	142	130	128
TCP10	Tx	1156	1156	1155	1155	1176	1176
	Rx	1137	1137	900	953	732	1057
UDP10	Tx	1144	1144	992	1007	1063	1043
	Rx	1146	1003	143	156	74	63

Paketgr.		8		16		32	
Cache		-	√	-	√	-	√
TCP100	Tx	3627	8490	3975	8619	4301	8313
	Rx	4445	7245	4630	7423	4798	7812
UDP100	Tx	85	8648	14	8537	16	8027
	Rx	106	102	95	96	93	93
TCP10	Tx	1175	1175	1176	1175	1185	1185
	Rx	423	796	468	850	486	667
UDP10	Tx	1120	1122	607	612	1152	1152
	Rx	20	39	5	2	0	0

Tabelle 7: Datentransferrate bei unterschiedlicher Paketgrößen - Angaben in Kilobyte pro Sekunde

Des Weiteren wurde überprüft wie groß der Overhead eines eingesetzten Linux gegenüber einer betriebsystemlosen Umgebung ist. Die Frage ist, inwiefern sich Scheduler, virtuelles Speichermanagement und ähnliches auf die Geschwindigkeit des Systems auswirken. Dazu wurde 100 Mal ein vier Megabyte großes Speicherabbild im DDR-RAM kopiert. Um den Einfluss der MMU unter Linux analysieren zu können, wurde im zweiten Versuch - Linux (2) - der verwendbare Speicher mit dem Kernelbefehl `mem=100M` reduziert. Das Kopieren fand dann in dem Bereich über 100 MByte statt. Die Ergebnisse in Tabelle 8 zeigen, dass das Kopieren mittels memcpy, einem C-Befehl, der auf geeignete PowerPC-Befehle zurückgreift, um Daten im RAM zu kopieren, deutlich schneller ist, als das Kopieren durch Benutzen einer For-Schleife. Des Weiteren zeigt sich, dass Linux die

	Ohne Betriebssystem	Linux	Linux (2)
For-Schleife	55s	12s	
memcpy-Befehl	10s	1,5s	10s

Tabelle 8: Dauer eines 100 maligen Kopierens von vier Megabyte

Kopierdauer drastisch reduzieren kann. Dies geht aber nur dann, wenn sich der zu kopierende Bereich innerhalb des von Linux verwalteten Speicherbereichs befindet. Eine Erklärung ist, dass Linux durch Paging die selbe Speicherseite in den anderen Bereich einblendet, um somit den Aufwand zu minimieren. Dies ist ein weiterer Vorteil des Linux gegenüber der betriebsystemlosen Umgebung.

3.5 Probleme

Während der Implementierungsphase sind zwei schwerwiegendere Probleme aufgetreten.
Das erste betrifft die Tatsache, dass Linux nicht direkt mit den FIFOs des Aurora-Cores zur Ein- und Ausgabe kommunizieren konnte. Es konnte nichts hineingeschrieben werden (keine Fehlermeldung) und nichts gelesen werden (*Bus Error*). Um das Problem zu untersuchen wurden zwei eigene Hardwarekomponenten entwickelt. Zum einen ein OPB-Viewer der bestimmte Signale jeweils hochzählte, um sie dann über eine externe Schnittstelle ausgeben zu können und zum anderen ein FIFO-Viewer, der die Kommunikation zwischen dem FIFO und des Aurora-Cores ausgeben konnte. Da auch so das Problem nicht gefunden wurde, bleiben bisher nur Spekulationen. Eine Möglichkeit ist, dass der angeschlossene FIFO zu spät die Daten auf den Bus legt.

4 Zusammenfassung

Beim Einsatz von Bildverarbeitungsalgorithmen werden hohe Anforderungen an die Hardware gestellt. Ein Prozessor kann solche Aufgaben nicht gut lösen, da er nicht parallel arbeitet und bei jedem Schritt nur ein Bruchteil der Chipfläche genutzt wird. Solche Aufgaben lassen sich mit Hilfe von FPGAs effizienter lösen.

Die Bedienung von Geräten hingegen lässt sich einfacher gestalten und programmieren indem man vorhandene Software-Entwicklungswerkzeuge benutzt und eine Hardwareumgebung schafft, so dass Programme ausgeführt werden können. Mit dem Einsatz von Linux kann auf Bekanntes aufgebaut werden, so dass der Aufwand geringer wird. Da auf allen hier betrachteten FPGAs der PowerPC 405 enthalten ist, war es auch möglich ein Linux zu finden, dass ohne großen Aufwand auf allen Boards lauffähig ist. Hinzu kam, dass durch den vorhandenen DDR-RAM und dem System-ACE auch eine - für den Embedded-Bereich großzügige - Speicherausstattung vorhanden ist. Bei der Einarbeitung und Auswahl einer geeigneten Version hatte sich herausgestellt, dass kommerzielle Anbieter Teile ihrer Entwicklung kostenlos und mit Quelltext zur Verfügung stellen, OpenSource-Entwickler anstellen und bezahlen, sowie auch andere Projekte mit in ihre Distributionen aufnehmen. Somit war es ohne finanziellen Aufwand möglich ein Linux anzupassen, indem auf OpenSource-Projekte zurückgeriffen wurde. Dies ist aber auch der Grund warum Echtzeitfähigkeiten nicht in betracht gezogen wurden. Kommerzielle Embedded-Linux Anbieter stellen solche Erweiterungen nicht frei zur Verfügung.

Möchte man die nötige Infrastruktur verringern, so existiert weiterhin die Möglichkeit, ohne Betriebssystem Software zu schreiben. Dabei muss jedoch beachtet werden, dass man für manche Dinge das Rad neu erfinden muss, also bereits vorhandene stabile Software neu schreiben muss. Ähnliches gilt auch für alternative Betriebssysteme. Für diese ist die Programmauswahl nicht sehr groß.

Für die Bildverarbeitung spielt Linux nur eine untergeordnete Rolle, das Anbinden des PowerPCs und Bereitstellen der Infrastruktur bedarf eines relativ großen Aufwandes, der nur dann gerechtfertigt ist, wenn das System außer Bildverarbeitungsalgorithmen auch steuernde Aufgaben wahrnimmt. Um eigene Hardware an den PowerPC anzubinden, ist weiterer Aufwand nötig. In [26] wurde bereits eine Lösung für einen Waldbrandklassifikator entwickelt, indem Block-RAM als Zwischenpuffer und Schnittstelle zum PL-Bus genutzt wurde.

A Schematische Darstellungen der Karten

A.1 Amirix AP120

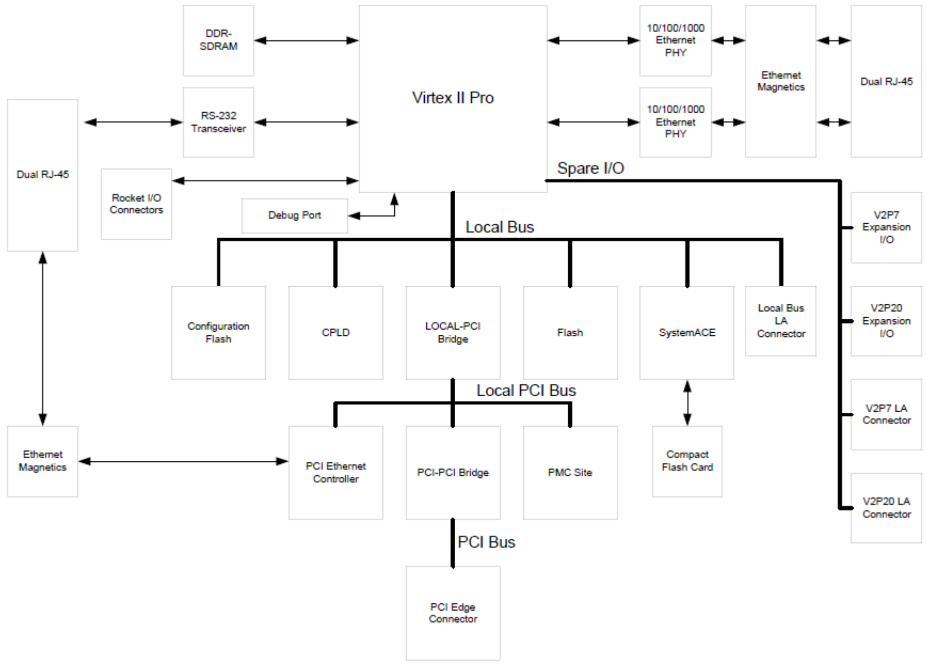

Abbildung 8: Schematischer Aufbau der Amirix AP120 [23]

A.2 Digilent XUPV2P

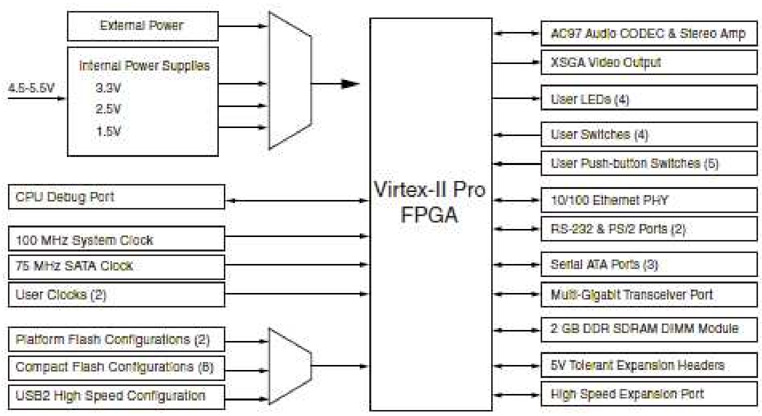

Abbildung 9: Schematischer Aufbau der Digilent XUPV2P [25]

A.3 AlphaData ADM-XP

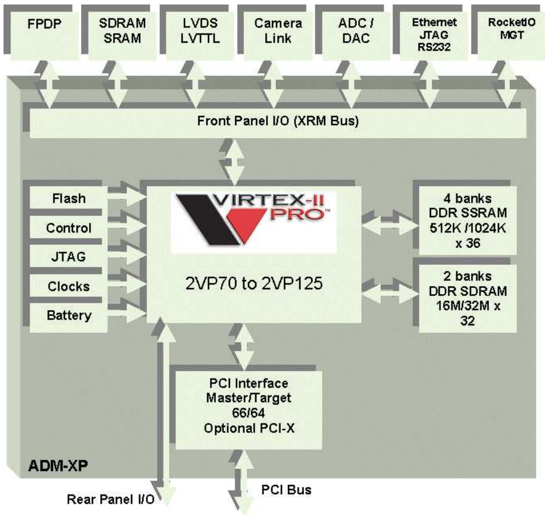

Abbildung 10: Schematischer Aufbau der Alpha Data ADM-XP [22]

B PowerPC-405 Skript, Kerneländerungen und Konfigurationsdateien

Das Skript `demo-powerpc-405.sh`:

```
#!/bin/sh
# This script has one line for each known working toolchain
# for this architecture.  Uncomment the one you want.
# Generated by generate-demo.pl from buildlogs/all.dats.txt

set -ex
TARBALLS_DIR=$HOME/downloads
RESULT_TOP=/vol/fob-vol3/mi02/brueckne/fpga/crosstool

export TARBALLS_DIR RESULT_TOP
GCC_LANGUAGES="c,c++"
export GCC_LANGUAGES

# Really, you should do the mkdir before running this,
# and chown /opt/crosstool to yourself so you don't need to run as root.
mkdir -p $RESULT_TOP
eval `cat powerpc-405.dat gcc-3.4.4-glibc-2.3.3.dat` sh all.sh --notest

echo Done.
```

Am Kernel wurden folgende Änderungen vorgenommen:

- Einstellen des Compilers und der Architektur in der `Makefile`:

```
ARCH := ppc
CROSS_COMPILE = powerpc-405-linux-gnu-
```

- Aktualisieren der `arch/ppc/platforms/xilinx_ocp/Makefile`, da jetzt Version 2.00 verwendet wird:

```
xilinx_ocp-objs += xbasic_types.o xdma_channel.o xdma_channel_sg.o \
            xipif_v1_23_b.o \
            xpacket_fifo_v2_00_a.o xpacket_fifo_l_v2_00_a.o \
            xversion.o
```

- Versionsnummern in der `arch/ppc/platforms/xilinx_ocp/xilinx_syms.c` aktualisieren :

```
#include "xpacket_fifo_v2_00_a.h"
EXPORT_SYMBOL(XPacketFifoV200a_Initialize);
EXPORT_SYMBOL(XPacketFifoV200a_Read);
EXPORT_SYMBOL(XPacketFifoV200a_SelfTest);
EXPORT_SYMBOL(XPacketFifoV200a_Write);
```

- Fehlermeldung beim Compilieren des Kernels ohne I2C-Bus durch Auskommentieren der Zeile 709 aus der `arch/ppc/boot/simple/embed_config.c` entfernen.

- Fehler in `arch/ppc/boot/simple/Makefile` entfernen. Dazu muss in Zeile 267 der Befehl von `mv` in `cp` geändert werden.

Die Konfigurationsdateien für BusyBox und den Kernel, sowie ein *Board Support Package* sind auf der CD zu finden.

C Hinweise zu Wishbone

Damit der opb2wishbone Wrapper funktioniert, müssen Anpassungen vorgenommen werden [1].

1. Der Daten- und Adressbus des Wishbone Cores müssen dem Wrapper angepasst werden (32 Bit).
2. Herunterladen und Installieren ins pcore Verzeichnisses des Projekts
3. Jetzt müssen in der bbd-Datei die Netzlisten umbenannt werden in opb2wb.edn.
4. In der mpd-Datei muss PORT opb_rst in PORT rst ändern.
5. Wichtig ist nachzusehen ob das rst-Signal richtig im globalen Design angeschlossen ist.
6. In der Netzliste muss noch „ (program "Xilinx ngc2edif"(version "G.31a")) “ umbenannt werden in „ (program "none") “
7. Als Adresse muss 0x80000000 genommen werden.

Literatur

[1] *Anleitung für den Wishbone-Wrapper.* `http://electronix.ru/forum/index.php?showtopic=19848`

[2] *Beam FpgaLinuxSys.* `http://portal.beam.ltd.uk/fpgalinux`

[3] *BusyBox.* `http://busybox.net`

[4] *CameraLink 1.1 Spezifikationen*

[5] *Converting Between YUV and RGB.* `http://msdn2.microsoft.com/en-us/library/ms847101.aspx`

[6] *Cross Linux From The Scratch für den PowerPC, Version 1.0.* `http://cross-lfs.org/view/1.0.0/ppc/`

[7] *Cygwin, a Linux-like environment for Windows.* `http://www.cygwin.com`

[8] *Get there faster with DevRocket.* `http://www.mvista.com/product_detail_devrocket.php`

[9] *HDMI.* `http://www.hdmi.org`

[10] *Linux PowerPC Kernelbaum.* `bk://ppc.bkbits.net/linuxppc-2.4`

[11] *ModelSim.* `http://www.model.com/`

[12] *Monta Vista Linux.* `http://www.mvista.com/`

[13] *Netzwerkperformance mit NetIO messen.* `http://www.nwlab.net/art/netio/netio.html`

[14] *QNX Neutrino Realtime Operating System.* `http://www.qnx.com/products/rtos/index.html`

[15] *TimeSys LinuxLink.* `https://linuxlink.timesys.com/`

[16] *U-Boot, Universal Bootloader project.* `http://u-boot.sf.net`

[17] *uClibc-Bibliothek.* `http://www.uclibc.org/`

[18] *uClinux.* `http://www.uclinux.org/`

[19] *Wishbone Bus Spezifikation.* `http://www.opencores.org/projects.cgi/web/wishbone/wishbone`

[20] *Wishbone Wrapper für den OPB-Bus.* `http://www.opencores.org/projects.cgi/web/opb_wb_wrapper/overv`

[21] *Xilinx Virtex II Pro Datenblatt.* `http://direct.xilinx.com/bvdocs/publications/ds083.pdf`

[22] Alpha Data: *Handbuch zur AlphaData ADM-XP*

[23] Amirix: *Handbuch zur Amirix AP120*

[24] Buder, M.: *Konzeption eines eingebetteten Systems für die Bildverarbeitung.* Studienarbeit am Institut für Informatik, 2007

[25] Digilent: *Handbuch zum Digilent XUPV2P-Board*

[26] Ehrig, M.: *Entwurf einer universell nutzbaren Kommunikationsschnittstelle zur Integration in Hardware realisierter blockorientierter Algorithmen in ein Prozessorsystem.* Studienarbeit am Institut für Informatik, Dezember 2004

[27] Field, G.: *SCSI.* MITP, 2001

[28] Hyde, J.: *USB Design by Example.* Wiley Computer Publishing, 1999

[29] Kegel, D.: *Crosstools.* `http://kegel.com/crosstool/`

[30] Klingauf, U.: *Klingaufs mkrootfs-Skript.* http://www.klingauf.de/v2p/mkrootfs.tgz

[31] Nelson, B. ; Baillio, B.: *Configuring and Installing Linux on Xilinx FPGA Boards.* http://splish.ee.byu.edu/projects/LinuxFPGA/configuring.htm

[32] Rech, J.: *Ethernet.* Heise, 2002

[33] Ryser, P. ; Baxter, M.: Embedded System á la Carte. In: *Embedded Linux Journal* August (2002). http://www.linuxdevices.com/articles/AT4160697622.html

[34] Schnick, D.: Hintergrundinfos zur HDMI-Schnittstelle. In: *Hifi-Regler* (2007)

[35] Tanenbaum, A.: *Modern Operating Systems.* Prentice Hall International, 2001

[36] Torvalds, L. ; Diamond, D.: *Just for Fun.* Hanser, 2001

[37] Widmer, A. X. ; Franaszek, P. A.: A DC-Balanced, Partitioned-Block, 8B/ 1 OB Transmission Code. http://www.research.ibm.com/journal/rd/275/ibmrd2705D.pdf