FHS
HOCHSCHULE FÜR TECHNIK, WIRTSCHAFT
UND SOZIALE ARBEIT ST.GALLEN

FACHBEREICH ELEKTROTECHNIK

FPGA Wavelet Transformation für Bildübertragung

Diplomarbeit im Fach Signalverarbeitung

erstellt von: Daniel Bachofen
Brühlstrasse 100
9320 Arbon

betreut von: Dipl. Ing. ETH Björn Schaltegger

Experte: Dipl. Ing. HTL Hans Rissi

Eingereicht: 3. Oktober 2001

FHS
Hochschule für Technik, Wirtschaft
und Soziale Arbeit St.Gallen
Fachbereich Elektrotechnik

Abstract

Diese Diplomarbeit untersucht die Implementierung einer 2D Wavelet Transformation in einem FPGA. Im ersten Teil der Arbeit werden verschiedene Integer to Integer Wavelet Transformations-algorithmen besprochen. Auf dem Lifting Schema basierend wird im zweiten Teil eine Wavelet Transformation in einem FPGA realisiert und getestet. Der dritte Teil der Arbeit beschäftigt sich mit den Möglichkeiten der verlustfreien und verlustbehafteten Bildkomprimierung. Im vierten Teil wird, aufbauend auf der Wavelet Transformation, eine serielle Bildübertragung mit Entropiecodierung entworfen. Dieser Entwurf wird ebenfalls in einem FPGA implementiert. Im letzten Teil der Arbeit wird ein Hardware Konzept für eine Wavelet basierte Kompression von Videodaten vorgestellt.

Vorwort

Durch die enorme Leistungssteigerung, welche im Bereich der konfigurierbaren Logik in den letzten Jahren erreicht wurde, eröffnen sich neue Möglichkeiten für die digitale Schaltungstechnik. Insbesondere kleine und mittlere Unternehmen erhalten die Chance auch für komplexere Signalverarbeitungsalgorithmen schnelle, digitale Logik einzusetzen.
Die Motivation zu dieser Arbeit entstand aus der Idee das interessante Gebiet der konfigurierbaren Logik mit einem neuen, mathematischen Werkzeug – der Wavelet Transformation – zu verknüpfen.

Die gleichzeitige Einarbeitung in zwei neue Themenkreise war eine Herausforderung. Dank der Ressourcen die das Internet bietet, war es jedoch möglich, in relativ kurzer Zeit einen Überblick über die Themengebiete und die neusten Trends zu gewinnen.

An dieser Stelle möchte ich mich herzlich bei jenen bedanken die mich während der Diplomarbeit unterstützt haben. Ein besonderer Dank für die hilfreichen Ratschläge und die zur Verfügung gestellte Literatur geht an Herrn B. Schaltegger. Auch an M. Geser ein spezielles Dankeschön – die interessanten Diskussionen über die Wavelet Transformation haben mir den Einstieg in das Thema um Einiges erleichtert.
Das letzte Dankeswort ist für meine Freundin bestimmt. Ihre Unterstützung hat es mir ermöglicht, mich praktisch ausschliesslich auf die Diplomarbeit zu konzentrieren.

Inhalt

1 Einleitung

1.1 Problemstellung

Die Wavelet Transformation hat sich seit Anfang der neunziger Jahre zu einem wichtigen neuen Werkzeug der Signalverarbeitung entwickelt. Die Wavelet Transformation ermöglicht eine Zerlegung von Signalen in wenige Teilkomponenten, die anschliessend mit geringerem Aufwand gespeichert werden können.
Die Wavelet Transformation gewinnt bei der Bildübertragung zunehmend an Bedeutung. Leistungsfähige Wavelettransformationsalgorithmen für Bildübertragungen erfordern aufgrund der zu verarbeitenden grossen Datenmengen den Einsatz dezidierter Hardware. Im Rahmen dieser Diplomarbeit soll für die Bildverarbeitung ein leistungsfähiger Wavelet Transformations-algorithmus in einem FPGA implementiert werden.

1.2 Aufgabenstellung

Es sind für die Bildübertragung geeignete Wavelet Transformationsalgorithmen auf ihre Implementierbarkeit in FPGA's zu untersuchen und zu bewerten. Ein für FPGA geeigneter, leistungsfähiger Wavelet Transformationsalgorithmus ist auf einem Virtex XSV-Entwicklungs-board zu implementieren. Die Möglichkeiten verlustfreier und verlustbehafteter Bildübertragung sind zu untersuchen.

1.3 Voraussetzungen

Die Herleitung der theoretischen Grundlagen der Wavelet Transformation und des Lifting Schemas würde den Rahmen dieser Diplomarbeit sprengen. Aus diesem Grund wird in dieser Arbeit darauf verzichtet diese theoretischen Grundlagen detailliert zu besprechen.
Die Herleitung der Theorie zur Wavelet Transformation und zum Lifting Schema ist detailliert in verschiedenen Veröffentlichungen vorhanden. In Anhang B finden sich einführende Publikationen sowohl zur Wavelet Transformation als auch zum Lifting Schema.
In den einzelnen Kapiteln wird dann nochmals konkret auf die entsprechenden, einführenden Veröffentlichungen hingewiesen.

1.4 Wieso Wavelets zur Bildkompression

Ob sich Wavelets zur Bildverarbeitung besser eignen als bereits länger bekannte Methoden wie beispielweise die Fourier Transformation oder die Cosinus Transformation, darüber streiten sich gem. B. Burke Hubbard [BUR96] selbst die Experten. Untersuchungen im Zusammenhang mit der Kompression von Fingerabdrücken für das FBI haben jedenfalls gezeigt, dass – zumindest für diese speziellen Bilder – Wavelet-basierte Methoden anderen Kompressionsmethoden überlegen sind.
Bei Wavelet-basierten Kompressionsmethoden wird der Effekt ausgenützt, dass durch die Wavelet Transformation die Anzahl der Bildpunkte mit Werten nahe Null zunimmt. Die folgenden Histogramme (Bild 2) zeigen diesen Effekt deutlich.

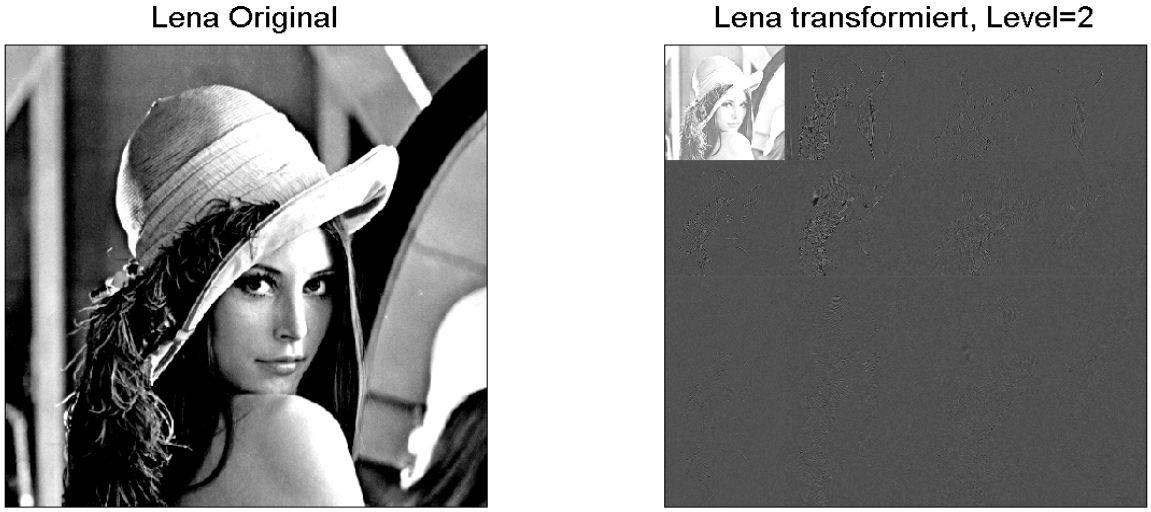


Bild 1: ‚Lena' im Originalbild und transformiert mit 5-3 Wavelet
(Leider sind die Feinheiten im grauen Bereich des transformierten Bildes selbst bei guter Druckqualität schlecht sichtbar)

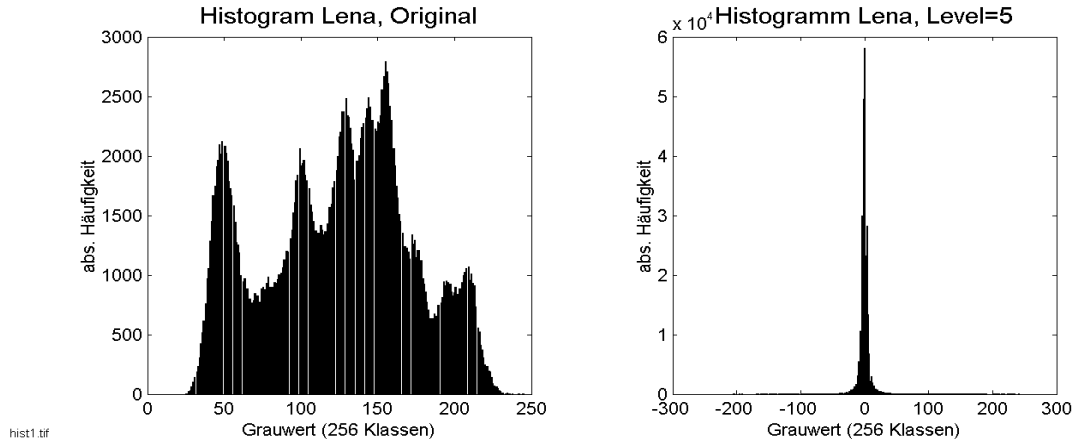


Bild 2: Histogramm zu ‚Lena' Original und ‚Lena' transformiert mit 5-3 Wavelet

Die Anhäufung von Bildpunkten mit Werten nahe der Zahl Null hat zur Folge, dass die Information des Bildes konzentrierter vorhanden ist.
Als zweiter Vorteil der Wavelet transformierten Bilder entstehen im Bild oft sehr grosse Bereiche, in denen die Bildpunkte den gleichen Wert haben. Es ergibt sich also auch in der Bildgeometrie eine Konzentration (siehe Bild 1 rechts).
Durch die oben beschriebenen Effekte können nachfolgende Algorithmen zur Codierung des Bildes effektiver arbeiten. Das Bild kann stärker komprimiert werden.

1.5 Entwicklungsumgebung

1.5.1 XSV-300 Entwicklungsboard

Um die entwickelten Lösungsansätze zu testen stand ein FPGA XSV-Entwicklungsboard der Firma XESS zur Verfügung. Dieses Board enthält als Kern einen Xilinx Virtex 300 FPGA mit diverser Peripherie, welche unter anderem auf Bildverarbeitung ausgelegt ist. Tabelle 1 zeigt eine Aufstellung der zur Verfügung stehenden Peripherie.

Tabelle 1: wichtigste Elemente des XSV-Boards

Bauteil	**Beschreibung**
XC95108	CPLD von Xilinx
DS1075	programmierbarer Clockgenerator (100MHz)
28F016S5	16 Mbit Flash RAM
AS7C4096	1MByte (16Bit) SRAM
SAA7113	PAL/SECAM/NTSC Video Decoder mit I^2C-Bus
BT481A	VGA- RAMDAC
AK4520A	Stereo In-Out Audio Codec
LXT970A	Ethernet PHY
PDIUSB11A	USB-Schnittstellentreiber
	Parallele Schnittstelle
	PS-2 Schnittstelle
MAX232A	Serielle Schnittstelle

Weitere Informationen zum XSV-Board können dem Manual [XES00] entnommen werden.

Die Videodaten wurden mit einer handelsüblichen, analogen Videokamera über den Video-Eingang auf das XSV-Board übermittelt. Es wurde eine CVBS Schnittstelle benutzt.
Über den VGA-RAMDAC wurden die verarbeiteten Daten auf einem VGA-Monitor dargestellt.

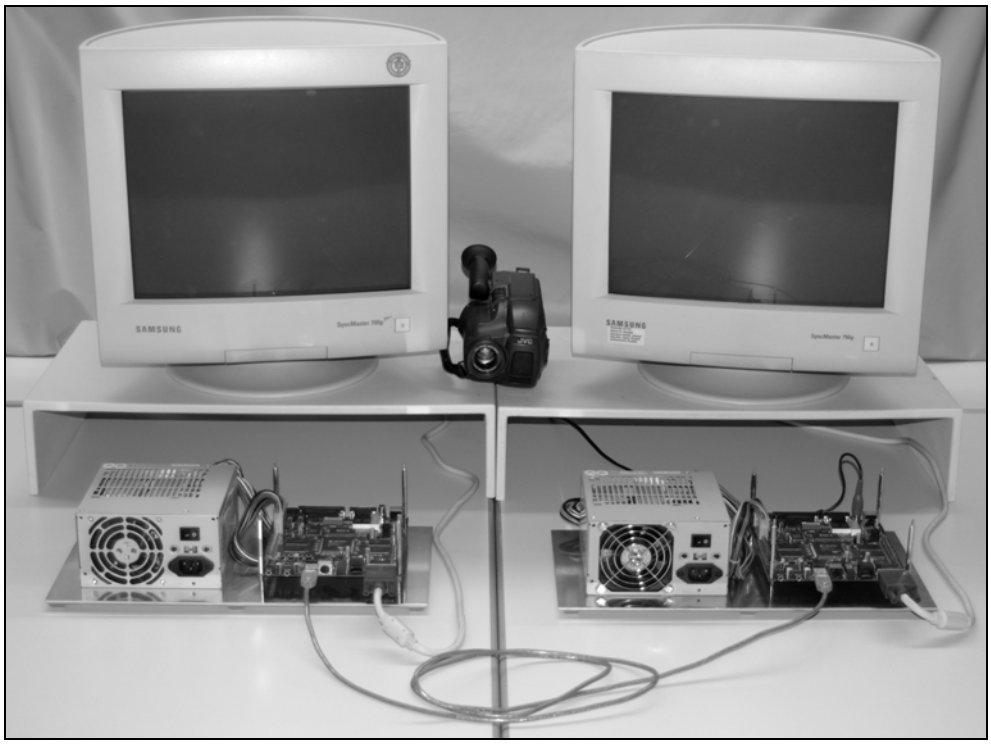


Bild 3: Bildverarbeitungssystem mit XSV-300 Entwicklungsboard

1.5.2 VHDL

VHDL ist eine Hardwarebeschreibungssprache welche mit dem Konzept entwickelt wurde auf allen Ebenen des Entwurfes elektronischer Schaltungen einsetzbar zu sein.
Dies hat zu einer extrem mächtigen und komplexen Programmiersprache geführt und nicht alle verfügbaren Befehle können auch für die Synthese von digitaler Logik benutzt werden.
Zusätzlich kommt erschwerend dazu, dass der selbe VHDL Code von verschiedenen Syntheseprogrammen unterschiedlich in Logik umgesetzt wird. Aus diesem Grund sollte man sich beim Programmieren in VHDL möglichst stark an die Richtlinien des Herstellers des Synthese Programms halten.

Der Designprozess für digitale Schaltungen mit Hilfe von CAD Systemen ist in Bild 4 dargestellt.

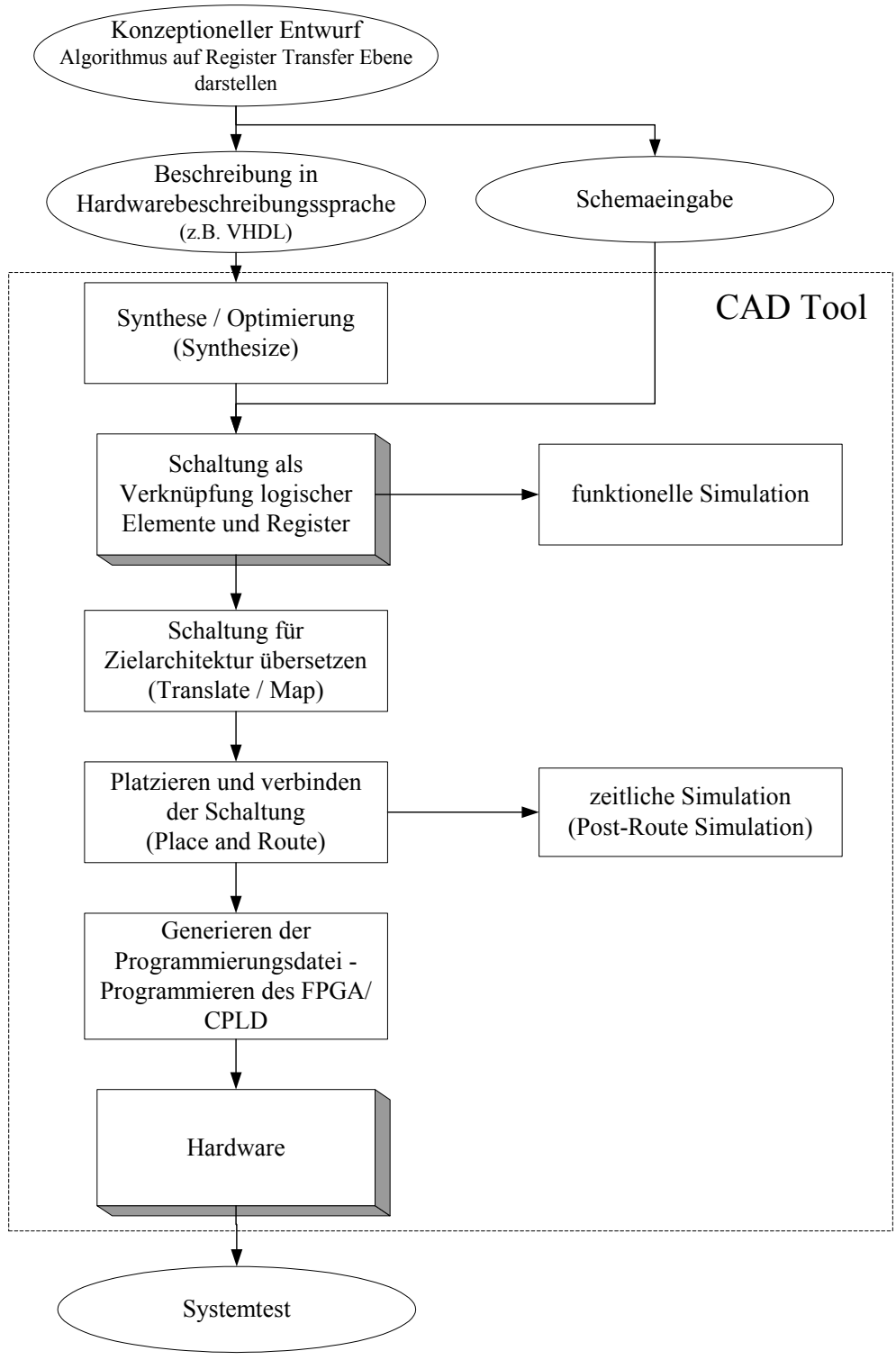


Bild 4: Entwicklungsablauf beim Entwurf digitaler Schaltungen mit Hilfe von CAD Systemen

1.5.3 FPGA Entwicklungsumgebung

Die VHDL Programme wurden mit dem Softwarepacket „Xilinx Foundation ISE 3.1i“ erstellt. Dieses Packet enthält auch eine „Starter“-Lizenz des Simulationsprogrammes „ModelSim“. Die maximale Anzahl Programmzeilen ist bei dieser Lizenz jedoch beschränkt und bei umfangreichen VHDL Programmen dauert die Simulation wesentlich länger als mit einer voll lizenzierten „ModelSim“ Version. Die Testprogramme wurden mit dem Programm „HDL Bencher“ generiert. Dieses Programm ist ebenfalls Teil der Entwicklungsumgebung „Xilinx Foundation ISE 3.1i“.
Für die VHDL Synthese wurde das XST Synthese Tool der Firma Xilinx verwendet.
Da die verschiedenen Designschritte im Softwarepacket „Xilinx Foundation ISE 3.1i“ auf mehrere Produkte (ModelSim, HDL Bencher) dritter Anbieter verteilt sind, braucht der Einstieg in die Entwicklungsumgebung etwas mehr Zeit als bei andern Entwicklungsumgebungen.

2 Analyse der Aufgabenstellung

2.1 Voraussetzungen für eine Hardwareimplementierung der Wavelet Transformation für Bildübertragung

Aufgrund der hohen Datenmengen, die bei einer Bildübertragung anfallen, sind die Anforderungen bezüglich Speicherplatz und Geschwindigkeit hoch. Im Folgenden wird untersucht wie viel Speicher für eine Hardware Implementierung der Wavelet Transformation benötigt wird und wie viele Speicherzugriffe für die Transformation eines Bildes ausgeführt werden müssen.

Bildgrösse:	480 x 640 Pixel	=	307 200	Pixel
Speicherbedarf bei gleichbleibender Bitbreite:				
2D Transformation, 480 x 640 Pixel S/W 8Bit			307 200	Bytes

Nötige Speicherzugriffe wenn 1Pixel mit einem Lese-/Schreibzyklus um
1 Transformationslevel in einer Dimension transformiert werden kann:

Speicherzugriffe für 2D Transformation eines Pixels			4	
1 Level:	307 200 x 4	=	1 228 800	Zugriffe
2 Level:	307 200 x 4 /4	=	307 200	Zugriffe
3 Level:	307 200 x 4 /16	=	76 800	Zugriffe
4 Level:	307 200 x 4 /32	=	19 200	Zugriffe
5 Level:	307 200 x 4 /64	=	4 800	Zugriffe
bei 5 Levels:			1 636 800	Zugriffe

Bei einem Transformationstakt von 16 MHz würde die Transformation eines Bildes etwas länger als 100ms dauern.

Die Anzahl Multiplikationen und Additionen (MAC) für die Wavelet Transformation hängt vom verwendeten Wavelet ab. Geht man von acht Multiplikationen für die Transformation aus, so werden 13 094 400 MAC’s benötigt um ein Bild zu transformieren.

2.2 Anforderungen an eine Wavelet Transformation zur Hardwareimplementierung

Die in 2.1 ausgeführten Berechnungen gingen davon aus, dass es möglich sein würde die Transformation eines Pixels mit einem Lese-/Schreibzyklus auszuführen. Die Anforderung geht allerdings noch weiter. Damit die Abfolge von Lese- und Schreibvorgängen nicht ins Stocken gerät darf die Berechnungszeit der Transformation maximal die Dauer von zwei Speicherzugriffen haben. Um einen Transformationstakt von 16 MHz erreichen zu können, darf die Berechnung also maximal 125 ns dauern.
Die Berechnung der Wavelet Transformation eines Pixels beinhaltet mehrere Multiplikationen. Die Anzahl der Multiplikationen hängt vom Typ des Wavelets ab. Damit die Anforderungen an die Geschwindigkeit der Hardware nicht beliebig hoch werden, müssen die Multiplikationen parallel berechnet werden können.
Um den Aufwand für einen Multiplizierer zu beschränken, sollten die Multiplikationen nach Möglichkeit in einem Integer Format durchgeführt werden. Sollte dies nicht möglich sein, so sind Fixpoint-Multiplizierer eine Alternative. Floatingpoint-Multiplizierer sollten jedenfalls vermieden werden, da der Hardware-Aufwand für solche Multiplizierer sehr gross, und damit die Berechnungsgeschwindigkeit eher klein ist.

2.3 Stand der Technik

In der Literatur tauchen Wavelet-basierte Video-Kompressions Codecs bereits ab 1991 [LEW91] auf.
Im Handel ist seit 1996 von Analog Devices [ANA96] ein solcher Codec erhältlich. Dieser benötigt allerdings zusätzlich einen DSP und benützt eine 9-7 Daubechies (aus [KOL99] S.1) Wavelet Transformation mit anschliessender Run-Length und Huffmann-Codierung. Es werden Kompressionsraten bis maximal Faktor 350:1 erreicht. Für Bildübertragung ohne sichtbaren Qualitätsverlust sinkt die Kompressionsrate auf Faktor 4:1 (typischer Wert für natürliche Bilder).
Kolarov und Lynch [KOL99] haben 1996 ein Konzept für einen Codec vorgestellt der ein spezielles, vom Haar-Wavelet abgeleitetes, 2-6 Integer Wavelet benutzt. Dieses Wavelet ist einerseits sehr einfach zu implementieren und hat andererseits trotzdem gute Eigenschaften für die Kompression von Bildern. Gemäss Kolarov und Lynch erreicht ihr Codec etwas höhere Kompressionsraten als der Codec von Analog Devices (siehe [KOL99] S.10).
Ab 1999 erscheinen erste Publikationen [RIT01, SUT99] welche sich mit der Implementation von Wavelet-basierten Kompressionen in FPGA's auseinandersetzen. Die Autoren dieser Publikationen arbeiten ausschliesslich mit speziellen, über das Lifting Schema [SWE95, SWE96] einfach in Hardware realisierbaren Wavelets.

2.4 Bildkompressionsstandards der Zukunft

Einer der neusten Standards ist JPEG2000. Dieser Standard benutzt die Wavelet Transformation als Teil der Kompression. In Part I des Standards sind zwei verschiedene Wavelets vorgesehen. Für Anwendungen mit möglichst hoher Kompressionsrate ist dies das 9-7 Daubechies Wavelet, für Anwendungen bei denen Geschwindigkeit oder verlustfreie Kompression im Vordergrund steht, soll ein 5-3 Integer Wavelet verwendet werden. In der Erweiterung von Part II sind zusätzliche, zur Zeit noch nicht spezifizierte, Wavelets vorgesehen.
Weitere Informationen zu JPEG2000 finden sich in [STRU00, S.192ff] sowie [MAR00].

2.5 Pflichtenheft der Diplomarbeit

Die Aufgabenstellung lässt sich in drei Teile zerlegen.

2.5.1 Untersuchung von Wavelet Transformationsalgorithmen für FPGA's

Ziel: Das Ziel dieser Untersuchung ist die Bewertung verschiedener Algorithmen. Aufgrund dieser Bewertung wird ein Algorithmus ausgesucht und in einem FPGA implementiert.
Bei der Wahl des zu implementierenden Algorithmus sind neben der allgemeinen Leistungsfähigkeit auch die Möglichkeiten der Zielhardware (Virtex XSV Board) zu berücksichtigen.

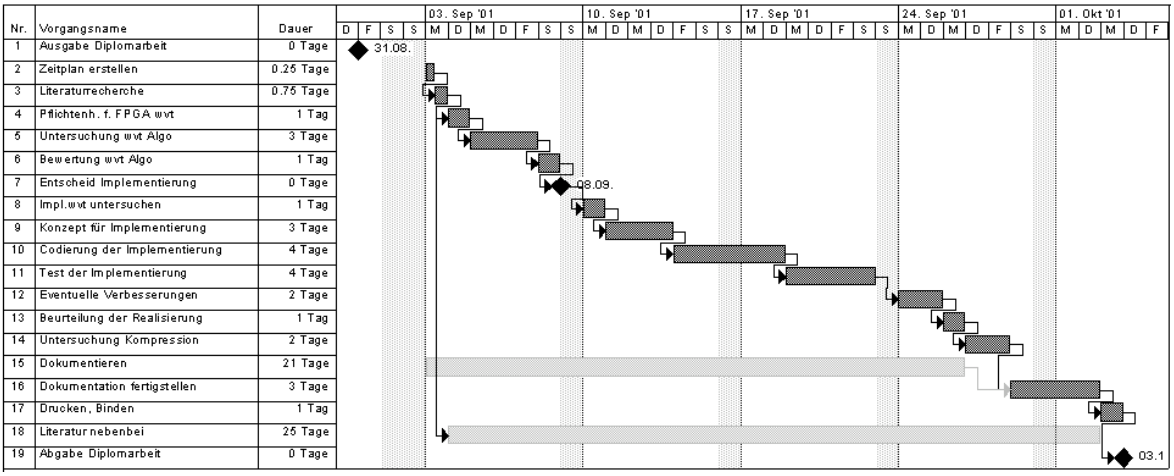
2.5.2 Implementation eines leistungsfähigen Wavelet Transformationsalgorithmus in einem FPGA

Ziel: Der im ersten Teil der Diplomarbeit ausgewählte Wavelet Transformationsalgorithmus wird auf einem Virtex XSV Board implementiert. Dabei soll ein wiederverwendbarer Core entstehen, welcher möglichst einfach auf eine andere Hardware portiert werden kann. Aus diesem Grund ist die Implementierung in der Hardwarebeschreibungssprache VHDL zu realisieren.
Für die Verifikation der Implementierung sind Testverfahren zu definieren. Aufgrund dieser Tests sollen korrekte Funktion und Leistungsfähigkeit des realisierten Algorithmus überprüft werden.

2.5.3 Untersuchung der Möglichkeiten verlustfreier und verlustbehafteter Bildübertragung

Ziel: Es soll untersucht werden welche verlustfreien und verlustbehafteten Verfahren sich zur Codierung und Kompression von Bildern eignen.
Dabei soll die Implementierbarkeit in einem FPGA und die zu erwartende Kompressionsrate beurteilt werden.

2.6 Projektplan



3 Untersuchung von Wavelet Transformationsalgorithmen für FPGA's

Wollte man verschiedene Wavelet Transformationsalgorithmen detailliert auf Ihre Eigenschaften und Eignung zur Implementierung in einem FPGA untersuchen, so wäre dies allein schon eine Arbeit, die den Zeitrahmen dieser Diplomarbeit sprengen würde. Aus diesem Grund wurde in einer Literaturrecherche nach möglichen Algorithmen gesucht. Deren Eignung zur Implementierung wurde ebenfalls auf dieser Recherche basierend abgeschätzt.

3.1 Beurteilungskriterien

3.1.1 Informationstheoretischer Entropiebegriff

Die folgende Definition des Entropiebegriffs ist dem Buch „Bilddatenkompression – Grundlagen, Codierung, MPEG, JPEG" von T. Strutz [STRU00] entnommen.

Ein Zeichen oder Symbol sei mit s_i bezeichnet. Somit umfasst das Alphabet $\mathbf{Z} = \{s_i\}$ mit $i = 1,2,...,K$ die Menge aller vorkommenden, unterschiedlichen Symbole. Ein endliches, zeitdiskretes Signal $x[n]$ kann als eine Folge von Symbolen aus $\mathbf{Z}$ betrachtet werden.
Jedes Symbol s_i besitzt eine Auftretenswahrscheinlichkeit p_i. In Abhängigkeit von p_i wird der Informationsgehalt $I(s_i)$ des Symbols s_i ermittelt:

$$I(s_i) = \log_2 \frac{1}{p_i} \quad \text{[bit]} \tag{3.1}$$

Gleichung 3.1 besagt, dass der Informationsgehalt eines Symbols umso kleiner ist, je häufiger das Symbol auftritt.

Für die Berechnung des mittleren Informationsgehalts einer Folge von statistisch unabhängigen Symbolen verwendet man den Begriff der Entropie H. Sie berechnet sich aus der Summe der gewichteten Einzelinformationen.

$$H = \sum_{i=1}^{K} p_i \log_2 \frac{1}{p_i} = -\sum_{i=1}^{K} p_i \log_2 p_i \quad \text{[bit/Symbol]} \tag{3.2}$$

Der Wertebereich der Entropie ist durch die Anzahl verschiedener Symbole K definiert. Die Entropie wird maximal, wenn alle Symbole gleichverteilt sind (p_i=1/K).

$$H_{\max} = \sum_{i=1}^{K} \frac{1}{K} \log_2 K = \log_2 K \tag{3.3}$$

Je ungleichmässiger die Symbole verteilt sind, desto geringer wird die Entropie eines Signals. Kommt nur ein Symbol in einem Signal vor, so wird die Entropie minimal H=0.

$$H_{\min} = 0 \quad wenn \quad p_i = \begin{cases} 1 & i = j \\ 0 & i \neq j \end{cases} \tag{3.4}$$

Am Extremfall H=0 lässt sich gut veranschaulichen, dass die erreichbare Kompressionsrate und die Entropie eines Signals umgekehrt proportional zu einander sind.
Nehmen wir ein Signal $x[n]$ mit N Zeichen und Entropie H=0. Es kommt also nur ein Symbol im Signal vor. Anstatt die ganze Signalfolge von N Zeichen zu übertragen würde es genügen das Symbol und die Anzahl N zu übertragen. Wäre die Entropie $H=H_{max}$, so liesse sich diese Art der Kompression nicht anwenden.

Je kleiner die Entropie eines Signals ist, desto grösser ist die erreichbare Kompressionsrate.

3.1.2 Zusammenstellung der Beurteilungskriterien

Da bei der Implementierung einer Wavelet Transformation in einem FPGA verschiedene Zielkonflikte bestehen, sind die Transformationsalgorithmen aufgrund von mehreren Kriterien zu beurteilen. Im Folgenden sind die verschiedenen Kriterien und ihre Bedeutung aufgelistet.

Tabelle 2: Beurteilungskriterien für Wavelet Transformationsalgorithmen

Kriterium	Beschreibung
Geschwindigkeit	Dauer einer Transformation eines Pixels
Ressourcenbedarf	Aufwand bei der Realisierung des Algorithmus in einem FPGA
Entropie	Je kleiner die Entropie des Bildes, desto grösser ist die mögliche Kompression

3.2 Algorithmen

Im folgenden Abschnitt wird kurz die Wavelet Transformation mit Filterbank vorgestellt. Danach wird das Lifting Schema am Beispiel des 5-3 Wavelets gezeigt. Auf dem Lifting Schema basierend werden Integer Wavelet Transformationen dargestellt.

Zur Berechnung der Wavelet Transformation gibt es verschiedene Möglichkeiten. Die einfachste und naheliegendste ist die Verwendung eines Hochpass- und eines Tiefpassfilters. Die entsprechenden Filter können direkt aus der Wavelet Transformation hergeleitet werden. Diese Herleitung wird hier nicht gezeigt, eine Einführung in die Wavelet Transformation findet sich in [REZ99] (siehe Anhang B).

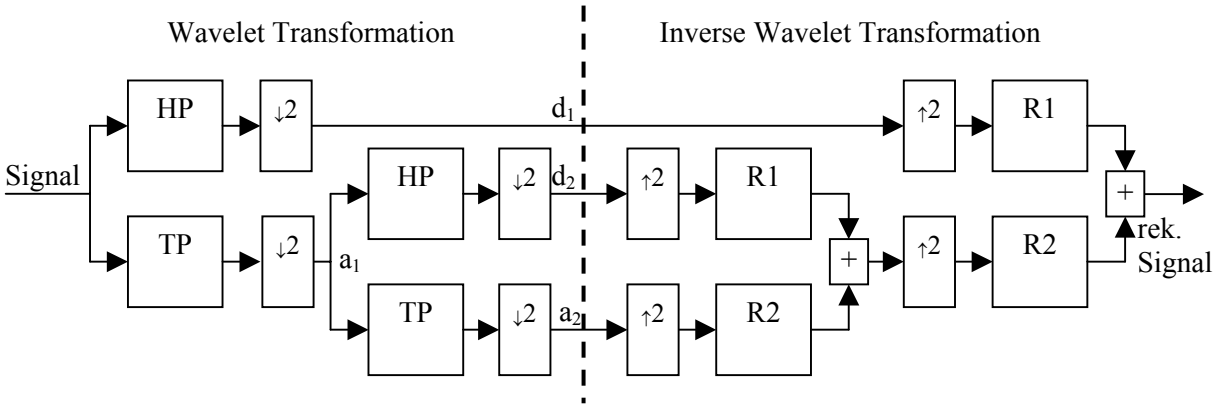


Bild 5: Wavelet Transformation mit Hoch-/ Tiefpasskaskade

Für einen Transformationslevel wird das Signal mit dem Hochpass und dem Tiefpass gefiltert. Das Resultat der Hochpassfilterung sind die Detailkoeffizienten d_n. Resultat der Tiefpassfilterung sind die Approximationskoeffizienten a_n, wobei n den Level der Transformation angibt. Für den nächsten Level werden die Approximationskoeffizienten a_n nach einer Unterabtastung mit denselben Hoch- und Tiefpässen gefiltert. Es entstehen die Detailkoeffizienten d_{n+1} und die Approximationskoeffizienten a_{n+1}.
Für die Rücktransformation werden die Koeffizienten überabgetastet, mit den Rekonstruktionsfiltern R1 und R2 gefiltert und danach addiert. Dies wird ebenfalls für jede Stufe durchgeführt.

Der Aufwand der Berechnung eines Punktes hängt also direkt von der Filterlänge der jeweiligen Hoch- und Tiefpässe ab. Bei dem 5-3 Wavelet (siehe Formel 3.5 und 3.6) sind dementsprechend 5+3 = 8 Multiplikationen und Additionen nötig.

3.2.1 Lifting Schema nach Sweldens

1995 wurde von Wim Sweldens eine effiziente Methode zur Berechnung der Wavelet Transformation vorgeschlagen [SWE95, SWE96]. Mit Hilfe des Lifting Schemas können die Hoch- und Tiefpassfilter jeder Wavelet Transformation umgeschrieben werden. Für die meisten Wavelet Transformationen ergeben sich mit dem Lifting Schema wesentlich einfachere Berechnungsalgorithmen. In Anhang B findet sich eine Veröffentlichung [SWE95], in der eine detaillierte Einführung in das Lifting Schema stattfindet.

Anhand des 5-3 Wavelets sollen die Vorteile des Lifting Schemas gezeigt werden. Das 5-3 Wavelet führt zu folgenden Tief- und Hochpassfiltern:

$$\textit{Tiefpass:} \qquad x'_{2i} = \frac{1}{8}(-x_{2i-2} + 2x_{2i-1} + 6x_{2i} + 2x_{2i+1} - x_{2i+2}) \tag{3.5}$$

$$\textit{Hochpass:} \qquad x'_{2i+1} = \frac{1}{4}(x_{2i} - 2x_{2i+1} + x_{2i+2}) \tag{3.6}$$

Durch Anwendung des Lifting Schemas ergibt sich folgende Berechnung:

$$\textit{Hochpass:} \qquad x'_{2i+1} = -x_{2i+1} + \frac{1}{2}(x_{2i} + x_{2i+2}) \tag{3.7}$$

$$\textit{Tiefpass:} \qquad x'_{2i} = x_{2i} - \frac{1}{4}(x'_{2i-1} + x'_{2i+1}) \tag{3.8}$$

Die Hochpassberechnung ist dabei umformuliert und mit Faktor 2 multipliziert. Die Berechnung des Tiefpasses verwendet bereits berechnete Hochpasswerte. Werden diese in Formel 3.8 eingesetzt, so erhält man die ursprüngliche Formel des Tiefpassfilters. Die Wavelet Transformation mit oder ohne Lifting Schema führt für den Tiefpass zu denselben, transformierten Koeffizienten. Die Koeffizienten des Hochpasses sind bei der Transformation mit dem Lifting Schema um Faktor zwei grösser.

Damit hat dieser Algorithmus wesentliche Vorteile gegenüber der herkömmlichen Berechnung:

Die Transformation kann **schneller berechnet** werden. Da bei der Berechnung des Tiefpasses bereits berechnete Hochpasswerte verwendet werden, fallen weniger Rechenoperationen als bei der herkömmlichen Wavelet Transformation an.

Die Transformation kann auf **einfache Weise invertiert** werden. Für die Rücktransformation können die Formeln einfach nach den Originalwerten umgestellt werden. Man erhält folgende Rücktransformationsvorschrift:

$$\textit{Tiefpass:} \qquad x_{2i} = x'_{2i} + \frac{1}{4}(x'_{2i-1} + x'_{2i+1}) \tag{3.9}$$

$$\textit{Hochpass:} \qquad x_{2i+1} = -x'_{2i+1} + \frac{1}{2}(x_{2i} + x_{2i+2}) \tag{3.10}$$

Die Rücktransformation kann also mit demselben Kern, durch Umschalten der Subtraktion in eine Addition, realisiert werden.

Wie in Kapitel 3.2.4 noch gezeigt wird, entsteht durch das Lifting Schema die Möglichkeit, die Wavelet Transformation mit ganzen Zahlen, also Integer to Integer, durchzuführen. Dadurch entsteht die Möglichkeit eine wirklich verlustfreie Wavelet Transformation zu realisieren.
Für gewisse Wavelets (z.B. 5-3 Wavelet) kann mit Hilfe des Lifting Schemas die gesamte Transformation mit Integer Multiplikatoren berechnet werden. Daher entstehen für die Realisierung der Transformation in Hardware grosse Vorteile.

Obwohl die oben aufgeführten Vorteile nicht für alle in der Bildverarbeitung wesentlichen Wavelets gleich gross sind, ist die Berechnung der Wavelet Transformation in jedem Fall einfacher als mit der herkömmlichen Berechnung. Aus diesem Grund wird für die folgenden Algorithmen die Verwendung des Lifting Schemas vorausgesetzt.

3.2.2 2D Wavelet Transformation

Da Bilder zweidimensional sind, sollte auch eine zweidimensionale Wavelet Transformation durchgeführt werden.
Die vielleicht verbreitetste 2D Wavelet Transformation funktioniert nach dem folgenden Schema. Zuerst werden alle Zeilen in das entsprechende Approximationssignal (A) und Detailsignal (D) zerlegt (horizontale Transformation). Im nächsten Schritt werden die Spalten ebenfalls in das Approximationssignal (a) und das Detailsignal (d) aufgeteilt (vertikale Transformation). Man erhält ein Bild das auf dem ersten Level Wavelet transformiert ist und aus vier Teilbildern besteht. Das Teilbild mit der Bezeichnung Aa1 wird nun wieder auf die gleiche Art zerlegt.

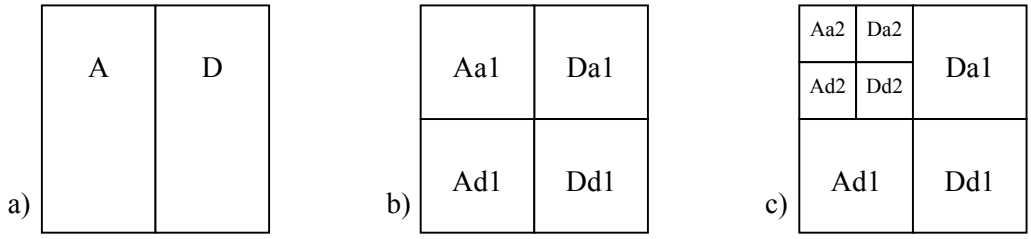


Bild 6: 2 D Wavelet Transformation: a) horiz. Transformation;
b) horiz. und vert. Transformation 1 Level; c) horiz. und vert. Transformation 2 Level

Eine leicht abgeänderte 2D Wavelet Transformation erhält man, indem zuerst alle Transformationslevels in horizontaler Richtung berechnet werden und erst danach alle Transformationen in vertikaler Richtung durchgeführt werden.

Bild 7: 2 D Wavelet Transformation a) zuerst vollständige horizontale Transformation
b) danach vollständige vertikale Transformation

Diese Art der 2 D Wavelet Transformation hat den Nachteil, dass mehr Rechenaufwand entsteht. Sie weist jedoch den Vorteil auf, dass jede Zeile und Spalte nur einmal aus dem Speicher geladen werden muss, um das ganze Bild zu transformieren. Es sind also wesentlich weniger Speicherzugriffe nötig.

3.2.3 Blockweise Wavelet Transformation

Da der Zugriff vom FPGA auf den externen Speicher einen „Flaschenhals“ der 2D Wavelet Transformation darstellt, gibt es Algorithmen die Teilbilder in einen internen Speicher des FPGA’s laden, fertig transformieren, eventuell codieren und dann wieder zurück in den externen Speicher schreiben. Der interne Speicher kann viel schneller getaktet werden und eignet sich vor allem für die Parallelisierung hervorragend. Gerade bei Codierungsalgorithmen wie dem Embedded Zerotree (EZW) Algorithmus, welche effizienten Zugriff auf das gesamte Bild erfordern, ist dies von grossem Vorteil. S. Sutter hat in seiner Diplomarbeit [SUT99] eine solche blockweise Wavelet Transformation getestet.
Ein Nachteil der blockweisen Wavelet Transformation liegt im grösseren Platzbedarf, hervorgerufen durch den RAM-Block im FPGA.
Wird nach der Wavelet Transformation das Bild verlustbehaftet komprimiert, können bei einer Blockverarbeitung an den Rändern der Blöcke Verzerrungen auftreten. Dies kann verhindert werden, wenn am Rand eines Blockes die Nachbarpixel des nächsten Blockes mit in die Transformation einbezogen werden. Der Speicherbedarf für einen Block nimmt quadratisch mit der Breite und Länge des Blocks zu. Die oben vorgeschlagene Erweiterung am Rand verursacht deshalb einen relativ starken Anstieg des Speicherbedarfs. Die folgende Rechnung soll dies am Beispiel des 5-3 Wavelets verdeutlichen. Der Block muss um die Filterlänge minus Eins erweitert werden:

Blockgrösse	32 x 32 Pixel	=	1024Pixel
Erweiterter Block	36 x 36 Pixel	=	1296Pixel
Speicherzunahme in Prozent		=	26.6%

3.2.4 Wavelet Transformation Integer to Integer

Die Grundlagen für die Integer to Integer Wavelet Transformation finden sich in den Schriften von Calderbank et. al. [CAL98] sowie Chao, Fisher und Hua [CHA96]. Die Veröffentlichung von Chao, Fisher und Hua [CHA96] ist in Anhang B vorhanden.

Ein Bild besteht aus Pixeln, welche normalerweise in einem Integerformat gespeichert sind. In dieser Arbeit wurde mit einem Graustufenformat von acht Bit / Pixel gearbeitet. Jeder Bildpunkt ist also Element der ganzen Zahlen und im Intervall [0,255] zu finden.
Anhand des 5-3 Wavelets (siehe Gleichung 3.7 und 3.8) können folgende Probleme gezeigt werden:

- Die transformierten Koeffizienten sind nicht mehr zwingend im Intervall [0,255].
- Die transformierten Koeffizienten sind nicht mehr zwingend ganzzahlig.

Es findet also eine Vergrösserung des Dynamikbereichs statt. Beim Speichern des transformierten Bildes führt dies nun zu Problemen, da für ein Pixel neu neun oder zehn Bit benötigt werden. Damit die Ausgangswerte der Transformation ganzzahlig und innerhalb des Intervalls [0,255] bleiben, muss die Berechnung abgeändert werden. Will man die Dynamik des Bildes und die perfekte Rekonstruierbarkeit erhalten, so kann dies mit Runden und Modularisieren geschehen [CAL98, CHA96].
Die Berechnungsvorschrift wird dazu folgendermassen abgeändert:

$$\textit{Hochpass:} \qquad x'_{2i+1} = \left(-x_{2i+1} + \left\lfloor \frac{x_{2i} + x_{2i+2}}{2} \right\rfloor \right) \bmod 256 \tag{3.11}$$

$$\textit{Tiefpass:} \qquad x'_{2i} = \left(x_{2i} - \left\lfloor \frac{x'_{2i-1} + x'_{2i+1}}{4} \right\rfloor \right) \bmod 256 \tag{3.12}$$

Trotz Runden und den Modulo Operationen ist das Bild ohne Probleme rekonstruierbar, da bei der Rücktransformation der Rundungsfehler wieder rückgängig gemacht wird. Ebenso wird ein Sprung von 0 auf 255 bei der Rücktransformation durch den inversen Sprung von 255 auf 0 wieder aufgehoben.
Die so entstandene Wavelet Transformation hat jedoch spezielle Eigenschaften, die es zu beachten gilt. Die Linearität der Filterung geht wegen der Rundung verloren. Auf Grund der modularen Arithmetik darf im transformierten Bild keine Quantisierung vorgenommen werden. Da kleine Werte durch die Modularisierung zu grossen Werten werden können, würde eine Quantisierung die Rücktransformation verfälschen.
Die Filter der Gleichungen 3.11 und 3.12 lassen sich unter Verwendung des Zweierkomplements sehr einfach in Hardware bauen. Rundung und modulare Arithmetik sind durch reine Verdrahtung realisierbar und führen zu keinem zusätzlichen Aufwand an Logik.

Wie in Bild 8 zu sehen ist entsteht, durch die Abbildung auf das Intervall [0,255] ein anderes transformiertes Bild als mit der normalen Wavelet Transformation. Vor allem die Konzentration in der Bildgeometrie fällt weniger stark aus.

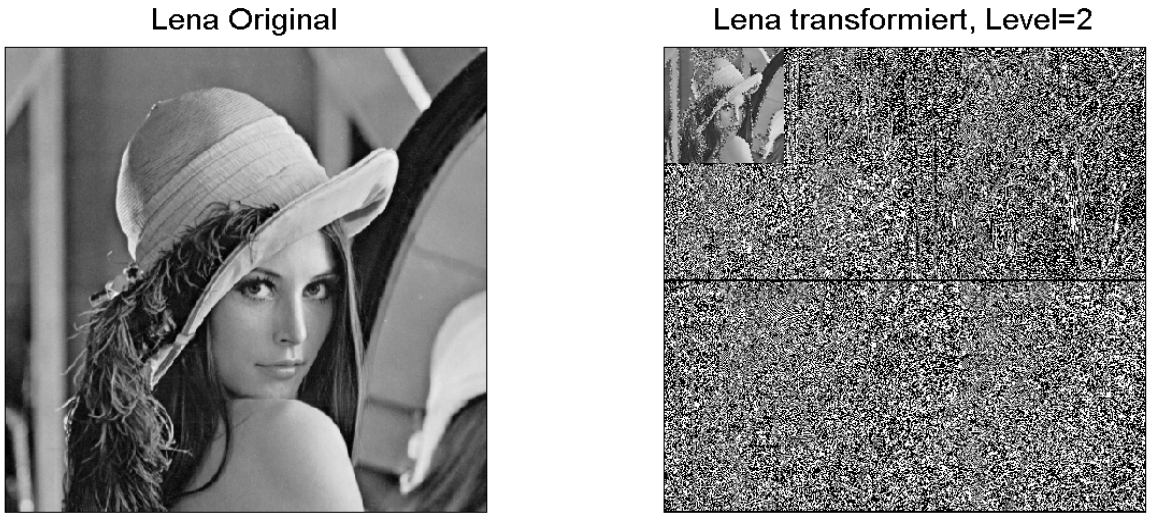


Bild 8: ‚Lena' transformiert mit 5-3 Wavelet (Integer to Integer)

Dem Vorteil der konstanten Dynamik dieser Integer to Integer Transformation steht der Nachteil gegenüber, dass diese Transformation die Entropie weniger verringert als eine konventionelle Wavelet Transformation.

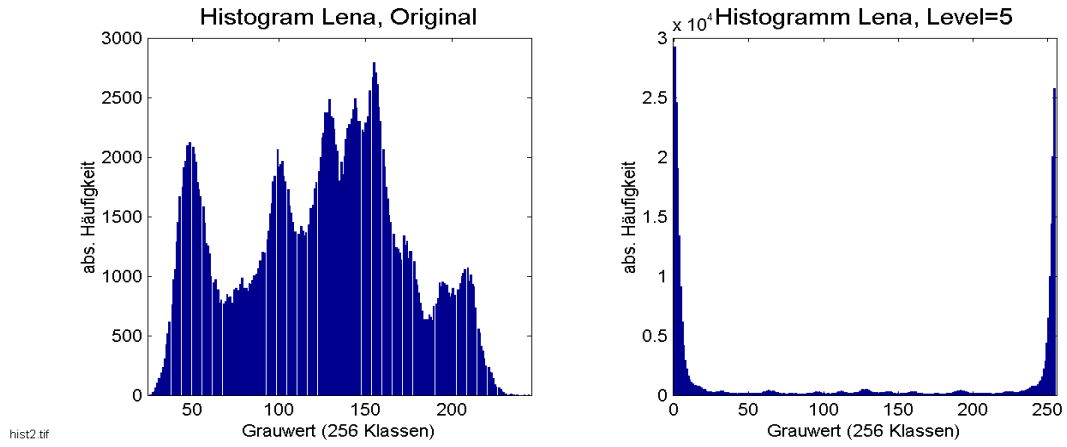


Bild 9: Histogramm zu ‚Lena' Original und ‚Lena' transformiert mit 5-3 Wavelet Integer to Integer

In Bild 9 ist deutlich sichtbar, dass mit der Integer to Integer Transformation eine wesentlich schwächere Konzentration der Werte stattfindet (zum Vergleich Bild 2). In der Folge können Codierungsverfahren weniger effektiv arbeiten als bei der Wavelet Transformationen mit erhöhter Dynamik.

3.2.5 Wavelet Transformation Integer to Integer mit erhöhter Dynamik

Verzichtet man auf die Forderung, dass der Dynamikbereich nicht erhöht werden darf, kann die Modularithmetik weggelassen werden. Wir betrachten wieder das 5-3 Wavelet. Lassen wir die Modularisierung in den Gleichungen 3.11 und 3.12 weg, so entsteht eine Transformation bei der Integerzahlen auf Integerzahlen abgebildet werden. Die Dynamik ist bei dieser Transformation jedoch nicht mehr für alle Koeffizienten gleich.

$$\textit{Hochpass:} \qquad x'_{2i+1} = -x_{2i+1} + \left\lfloor \frac{x_{2i} + x_{2i+2}}{2} \right\rfloor \tag{3.13}$$

$$\textit{Tiefpass:} \qquad x'_{2i} = x_{2i} - \left\lfloor \frac{x'_{2i-1} + x'_{2i+1}}{4} \right\rfloor \tag{3.14}$$

Die Wavelet Transformation Integer to Integer mit erhöhter Dynamik erlaubt auch eine nachfolgende Quantisierung des transformierten Bildes. Der Speicherbedarf pro Pixel kann dadurch wieder auf acht Bit reduziert werden. Findet eine solche Quantisierung statt, muss jedoch in Kauf genommen werden, dass das rücktransformierte Bild nicht mehr vollständig dem Original entspricht. Es entsteht also eine verlustbehaftete Transformation. Die Fehler welche durch die Quantisierung bei der Rücktransformation entstehen, sind bei einer Integer to Integer Transformation mit erhöhter Dynamik grösser als bei einer Integer to Floatingpoint Transformation. Dies trifft gemäss Reichel [REI1 S.7] speziell bei tiefen Kompressionsraten zu, bei höheren Kompressionsraten ist der Unterschied wesentlich kleiner.
Auch wenn keine Quantisierung vorgenommen wird, ist mit einer guten Verringerung der Entropie zu rechnen.

3.2.6 Wavelet Transformation Integer to Floatingpoint

Die Wavelet Transformation Integer to Floatingpoint ist eigentlich nur ein Spezialfall der diskreten Wavelet Transformation (DWT). Dieser Spezialfall zeichnet sich dadurch aus, dass alle Eingangswerte im Bereich der ganzen Zahlen liegen. Auf das transformierte Signal hat dies jedoch keinen Einfluss.
Wavelets weisen im Allgemeinen irrationale Koeffizienten auf. Auch wenn man das Lifting Schema anwendet, entstehen deshalb bei den meisten Wavelets nach der Transformation irrationale Signalwerte. Diese können anschliessend durch Quantisierung wieder auf Integerzahlen abgebildet werden. Damit wird jedoch eine perfekte Rekonstruktion unmöglich.
Wie bereits oben erwähnt hat die DWT für verlustbehaftete Signalkompressionen die besseren Eigenschaften als die Integer to Integer Transformationen.
Diesen guten Eigenschaften steht jedoch der viel grössere Aufwand bei der Realisierung gegenüber. Da im Allgemeinen bei der Berechnung der DWT trotz des Lifting Schemas Floatingpoint Operationen vorkommen, müssen Floatingpoint Multiplizierer realisiert werden. Der Hardwareaufwand für Floatingpoint Arithmetik ist wesentlich höher als für Integer Arithmetik. Damit verbunden sind normalerweise auch längere Rechenzeiten. Zusätzlich wird durch den grösseren Platzbedarf die Parallelisierbarkeit reduziert.

3.2.7 Eigenschaften der Wavelets

Die Wavelet Transformation kann mit verschiedenen Wavelets durchgeführt werden. Deshalb muss die Entscheidung getroffen werden, welches Wavelet für die Bildtransformation verwendet wird. Da die hier besprochene Wavelet Transformation später zur Bildkompression verwendet wird, sollte durch die Wavelet Transformation die Entropie des Bildes möglichst klein werden. In der Literatur ist das 9-7 Daubechies Wavelet das am meisten verbreitete Wavelet für Bildkompressionen.
Aufgrund der irrationalen Koeffizienten dieses Wavelets ist es jedoch nicht besonders gut zur Implementation in einem FPGA geeignet.
Es wird also ein Wavelet gesucht, welches mit ganzzahligen Koeffizienten oder besser noch mit Koeffizienten $a_k=2^n$ beschrieben werden kann.
Das 5-3 Wavelet weist diese Eigenschaft auf und ist in der Literatur ebenfalls weit verbreitet. Zudem wird dieses Wavelet im JPEG2000 Standard verwendet.
Weitere Wavelets wurden von Calderbank et al. in [CAL1] vorgestellt. Die von Calderbank untersuchten 4-2 und 2+2,2 Wavelets weisen für verlustfreie Kompression bessere Eigenschaften auf als das populäre 9-7 Daubechies Wavelet [CAL1, S.2].

In Tabelle 3 sind die oben erwähnten Wavelets im Lifting Schema dargestellt. Für das 2+2,2 Wavelet und das 9-7 Wavelet sind mehrere Lifting Schritte nötig.

Tabelle 3: Wavelet Transformationen im Lifting Schema (nach [CAL1, S.2])

Name	Wavelet Transformation	
5-3	$d'_l = -s_{2l+1} + \lfloor \frac{1}{2}(s_{2l} + s_{2l+2}) \rfloor$ $s'_l = s_{2l} - \lfloor \frac{1}{4}(d'_{l-1} + d'_l) \rfloor$	
4-2	$d'_l = s_{2l+1} - \lfloor \frac{9}{16}(s_{2l} + s_{2l+2}) - \frac{1}{16}(s_{2l-2} + s_{2l+4}) + \frac{1}{2} \rfloor$ $s'_l = s_{2l} + \lfloor \frac{1}{4}(d'_{l-1} + d'_l + \frac{1}{2} \rfloor$	
2+2,2	$d''_l = s_{2l+1} - \lfloor \frac{1}{2}(s_{2l} + s_{2l+2}) + \frac{1}{2} \rfloor$ $s'_l = s_{2l} + \lfloor \frac{1}{4}(d''_{l-1} + d''_l + \frac{1}{2} \rfloor$ $d'_l = d''_l - \lfloor \frac{1}{8}(-\frac{1}{2}s'_{l-1} + s'_l - \frac{1}{2}s'_{l+1}) + \frac{1}{8}(-\frac{1}{2}s'_l + s'_{l+1} - \frac{1}{2}s'_{l+2}) + \frac{1}{2} \rfloor$	
9-7	$d''_l = s_{2l+1} - \lfloor \alpha(s_{2l} + s_{2l+2}) + \frac{1}{2} \rfloor$ $s''_l = s_{2l} + \lfloor \beta(d''_l + d''_{l-1}) + \frac{1}{2} \rfloor$ $d'_l = d''_l + \lfloor \gamma(s''_l + s''_{l+1}) + \frac{1}{2} \rfloor$ $s'_l = s''_l + \lfloor \delta(d'_l + d'_{l-1}) + \frac{1}{2} \rfloor$	$\alpha \approx -1.586$ $\beta \approx -0.053$ $\gamma \approx 0.883$ $\delta \approx 0.444$

3.3 Gegenüberstellung der Algorithmen

Im Folgenden wird eine Auswahl von Kombinationen der oben aufgeführten, verschiedenen Algorithmen und Wavelets nach den Kriterien aus Tabelle 2 beurteilt.

Tabelle 4: Vergleich der Algorithmen

Kombination	Geschwindigkeit	Ressourcenbedarf	Entropie	Summe
5-3 i2oD, hvhv	gut	sehr gut	mittel	12
9-7 i2oD, hvhv	mittel	schlecht	gut	8
4-2 i2oD, hvhv	gut	gut	sehr gut	13
5-3 i2oD, hhvv	sehr gut	gut	mittel	12
9-7 i2oD, hhvv	mittel	schlecht	gut	8
4-2 i2oD, hhvv	gut	mässig	sehr gut	11
9-7 i2f, hvhv	mässig	schlecht	sehr gut	8
5-3 i2i, hhvv	sehr gut	sehr gut	mässig	12
5-3 i2oD, Block	sehr gut	mässig	mittel	10

Legende:	
hvhv:	-2D Transformation als Abfolge von horizontal-vertikal-horizontal-vertikal Transformationen
hhvv:	-2D Transformation als Abfolge von horizontal- horizontal - vertikal - vertikal Transformationen
Block:	-Transformation auf Teilbildern als Block im FPGA gespeichert
i2i:	-Integer to Integer Transformation mit konstanter Dynamik
i2oD:	-Integer to Integer Transformation mit erhöhter Dynamik
i2f:	-Integer to Floatingpoint Transformation

Um einen besseren Überblick über die Leistungsfähigkeit der einzelnen Algorithmen zu erhalten, wurde in der Spalte „Summe“ jedem der verbalen Beurteilungskriterien ein Zahlenwert zugeordnet und dann die Summe der verschiedenen Kriterien ermittelt. Die Zuordnung geht von 1=schlecht bis 5=sehr gut.

3.3.1 Wahl eines Algorithmus für die Implementation

In Tabelle 4 wird der Algorithmus mit dem 4-2 Wavelet und einer 2D Transformation mit hvhv Ablauf als am leistungsfähigsten beurteilt.
Für die Implementierung auf der Zielhardware, würde sich jedoch eine 2D Transformation mit hhvv Ablauf besser eignen. Dies vor allem, weil der Adressbus nur in zweifacher Ausführung vorhanden ist, und mit einer maximalen Schreibfrequenz von 25MHz gerechnet werden darf. Der Speicherzugriff wird also voraussichtlich die erreichbare Transformationsgeschwindigkeit am meisten beeinflussen. Da bei einer 2D Transformation mit hhvv Abfolge wesentlich weniger Speicherzugriffe nötig sind, sollte versucht werden eine solche Transformation zu realisieren.
Das 4-2 Wavelet benötigt Integer Multiplikationen, die Implementierung des 5-3 Wavelets hingegen kann ausschliesslich mit Schiebeoperationen und Additionen durchgeführt werden. Dies führt zu einem wesentlich kleinerem Hardware-Aufwand bei der Implementierung.
Bei einer hhvv-2D Transformation können alle Transformationslevels gleichzeitig bearbeitet werden. Es kann also eine enorme Parallelisierung stattfinden. Dies bedeutet jedoch, dass ca. fünf Transformationsstufen parallel realisiert werden müssen.
Im FPGA muss neben der Wavelet Transformation auch noch die Bedienung der Peripherie (Video Codec, VGA RAMDAC) Platz haben. Aus diesem Grund soll das einfachere 5-3 Wavelet implementiert werden. Dass das 5-3 Wavelet auch im JPEG2000 Standard verwendet wird, unterstützt den Entscheid zugunsten dieses Wavelets zusätzlich.

Für die Implementierung soll das 5-3 Wavelet verwendet werden. Die Berechnung erfolgt nach dem Lifting Schema. In einem ersten Schritt soll eine normale hvhv Transformation implementiert werden. Danach soll, in einer Erweiterung, die hhvv Transformation realisiert werden.

4 Implementierung der Wavelet Transformation in einem FPGA

4.1 Vorgehensweise

Als Erstes werden die für die Verifikation der Implementierung nötigen Testverfahren definiert. Diese Überlegungen sollen zuerst ausgeführt werden, damit eventuell nötige Teststrukturen gleich von Anfang an in das Konzept eingeplant werden können.
Im nächsten Schritt wird die Wavelet Transformation in Teilblöcke zerlegt. Es soll ein Blockschaltbild entstehen welches die 2D Bildtransformation in Unterfunktionen aufteilt. Danach werden diese Unterfunktionen im „Top-Down" Verfahren weiter definiert. Jede Unterfunktion wird soweit zerlegt, bis sie auf der Register Transfer Ebene beschrieben werden kann.
Sind die Teilblöcke soweit bekannt, kann die Codierung in VHDL beginnen. Hier wird mit der „Bottom – Up" Strategie vorgegangen. Die hierarchisch tiefer liegenden Teilblöcke werden zuerst programmiert und in der Simulation soweit wie möglich ausgetestet.
Am Schluss werden die vorgeprüften Funktionsblöcke zusammengefügt und die Systemtests durchgeführt.

Im Folgenden sind die einzelnen Teilschritte nochmals stichwortartig dargestellt:

- Definieren der Testverfahren
- Wavelet Transformation in Teilblöcke zerlegen
- Teilblöcke für digitale Logik aufbereiten
- Codierung der Teilblöcke in VHDL, sowie Test der Teilblöcke
- Zusammenführen der Teilblöcke und Test des Gesamtsystems

4.2 Testverfahren

4.2.1 Zur Verfügung stehende Testwerkzeuge

Die Tests können mit vier verschiedenen Werkzeugen durchgeführt werden.

Der VHDL Code kann mit der **funktionellen Simulation** auf logische Fehler überprüft werden. Mit der **Post-Route Simulation** wird neben der logischen Funktion auch das Timing im FPGA getestet.
Auf dem Entwicklungsboard können Tests entweder mit einem **Oszilloskop** oder dem **Logik-Analysator** durchgeführt werden. Für Messungen mit dem Logik-Analysator stehen die Daten-, Adress- und Steuerleitungen des externen Speichers zur Verfügung.

Letztes Testmittel ist das menschliche Auge. Auch wenn damit keine reproduzierbaren und vergleichbaren Messungen durchgeführt werden können, sollen die transformierten Bilder auch visuell beurteilt werden.

4.2.2 Funktionelle Kontrolle

Mit der funktionellen Kontrolle soll festgestellt werden, ob die Berechnungen der realisierten Wavelet Transformation korrekt sind. Um dies zu überprüfen wird ein Testvektor definiert, welcher auf einem PC Wavelet transformiert wird. Die Resultate der FPGA Wavelet Transformation

können dann mit diesem Testvektor verifiziert werden. Ist das Resultat beider Berechungen identisch, funktioniert die FPGA Wavelet Transformation korrekt.

Anforderungen an den Testvektor:

- Der Testvektor soll Zahlen aus allen Bereichen des Intervalls [0,255] enthalten.
- Es ist darauf zu achten, dass bei der Transformation Zahlen entstehen, welche das Intervall [0,255] verlassen. Damit wird das korrekte Funktionieren der Modulo Operationen getestet.
- Bei der Transformation des Testvektors sollen rationale Zahlen entstehen. Damit wird das richtige Funktionieren der Rundungs-Operationen getestet.
- Damit die FPGA Simulationswerkzeuge effizient genutzt werden können, soll der Testvektor möglichst kompakt sein.

Testvektor:

[255, 240, 0, 0, 128, 30, 50, 102, 173, 247, 247, 255]

Transformiert mit dem 5-3 Wavelet in 8 Bit Signed Integer Transformation:

Hochpass: [15, 192, 187, 140, 209, 248]
Tiefpass: [248, 13, 162, 97, 219, 5]

Die Fehlersuche in den VHDL Programmen wird mit den Simulationswerkzeugen und dem Testvektor durchgeführt.

Für die funktionelle Kontrolle auf der Zielhardware wird ein VHDL Programm geschrieben, mit welchem der Testvektor im Speicher des Entwicklungsboards abgelegt werden kann. Die Wavelet Transformation und die Rücktransformation können dann mit Hilfe des Logik-Analysators überprüft werden.

4.2.3 Evaluation der Leistungsfähigkeit

Um die Leistungsfähigkeit der FPGA Wavelet Transformation beurteilen zu können, sollen bestimmte Kenngrössen ermittelt werden. Dabei steht vor allem die Berechnungsgeschwindigkeit im Vordergrund. Um die maximale Transformationsgeschwindigkeit zu ermitteln, soll die Berechnungszeit für die Transformation eines Koeffizienten bestimmt werden. Weiter soll gemessen werden, wie lange die Transformation eines Bildes dauert.
Damit können Transformationskern und der gesamte Transformationsalgorithmus getrennt beurteilt werden.

Bestimmen der Berechnungszeit für die Transformation eines Koeffizienten:
Die Berechnungszeit soll mit Hilfe der Post-Route Simulation, unter Verwendung der Worst-Case Timings bestimmt werden.

Messen der Transformationsdauer für ein Bild:
Während der Transformation soll ein Ausgang des FPGA's auf High geschaltet werden. Mit einem Speicheroszilloskop wird dann die Transformationsdauer bestimmt.

4.3 Zerlegung der Wavelet Transformation in Teilblöcke

Aufgrund der Peripherie des Evaluationsboards wurde ein Top-Level Blockschaltbild entworfen. Dieses Blockschaltbild ist in Bild 10 auf Seite 22 zu sehen, es zeigt alle Teilelemente die im FPGA enthalten sind. Zudem ist die, für die Bildverarbeitung benötigte Peripherie des FPGA's aufgezeichnet.
Die Framegrabber Einheit sowie die Ansteuerung des VGA RAMDAC's wurden im Vorfeld der Diplomarbeit, während einer Semesterarbeit [BA01] aufgebaut.

4.3.1 Beschreibung der Funktionen im Top-Level Blockschaltbild:

Der CVBS Ausgang einer Standard Videokamera wird im Videodecoder in ein digitales YCrCb Signal gewandelt. Dieses wird über eine acht Bit Schnittstelle an den FPGA gesendet. Ausgelöst durch Drücken einer Taste wird mit dem Framegrabbermodul ein Bild im externen RAM abgelegt. Dabei wird nur der Luminanz Anteil des Videosignals verwendet. Im Speicher ist nach einem Grab Vorgang ein acht Bit, schwarzweiss Bild vorhanden.
Für die Wavelet Transformation wird dieses Bild zuerst Zeile um Zeile und danach Spalte um Spalte aus dem Speicher gelesen, transformiert und wieder zurück in den Speicher geschrieben. Während der Transformation einer Zeile werden die transformierten Koeffizienten im internen RAM des FPGA's zwischengespeichert.
Die inverse Transformation arbeitet in der gleichen Weise, ausser dass das Bild zuerst vertikal und danach horizontal zurücktransformiert wird.
Das VGA Modul liest fortlaufend das Bild aus dem externen RAM und sendet die Bilddaten an den VGA RAMDAC. Dieser wandelt die digitale Information in analoge Signale, welche dem VGA Standard entsprechen. Auf einem VGA Bildschirm kann das Bild nun dargestellt werden. Greifen die anderen Teilsysteme auf das externe RAM zu, wird ein schwarzes Bild auf dem Monitor dargestellt.
An die Steuereinheit sind drei Taster angeschlossen. Über diese Taster wird das Lesen eines Bildes sowie die Transformation und die Rücktransformation ausgelöst. Die Steuereinheit schaltet über den Multiplexer den Steuer-, Adress- und Datenbus des aktiven Teilblocks auf das externe RAM. Die Konfiguration der Wavelet Transformationsblöcke wird ebenfalls von der Steuereinheit übernommen.

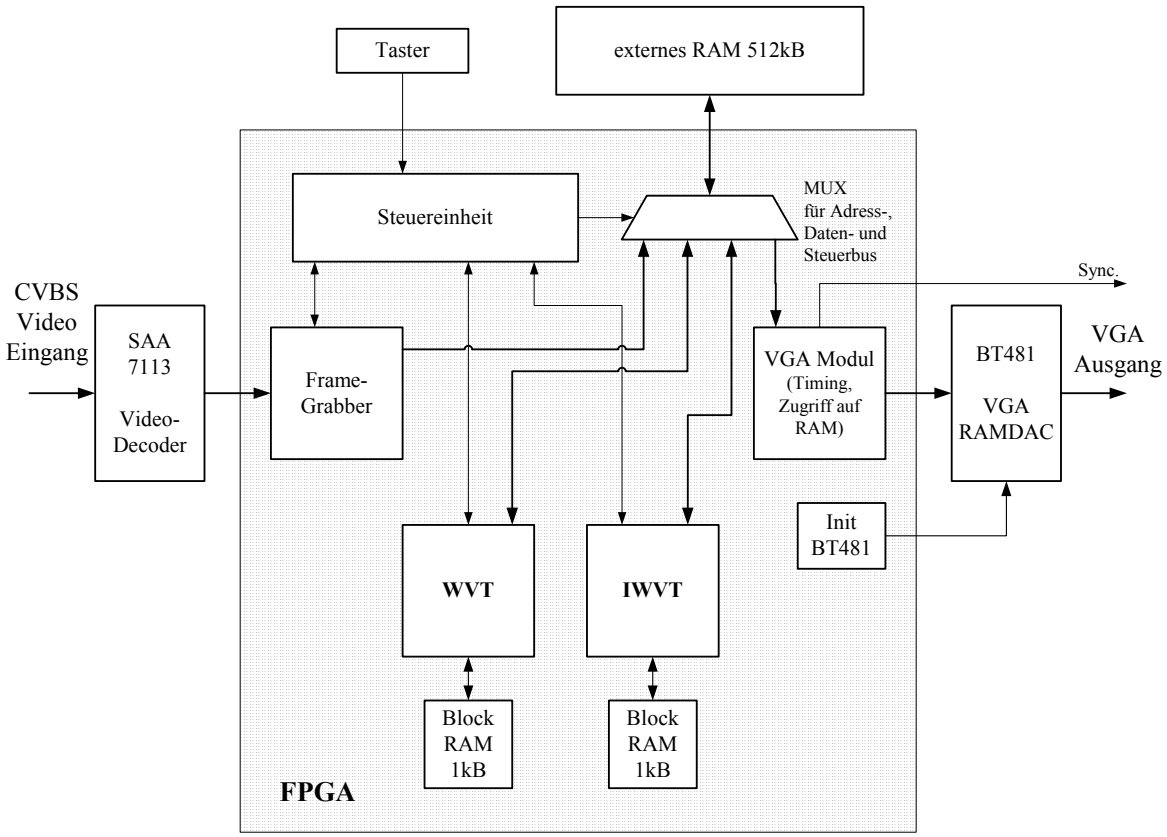


Bild 10: Blockschaltbild der im FPGA enthaltenen Teilsysteme

4.3.2 Wavelet Transformation

Beim Entwurf der Wavelet Transformationseinheit wurden primär drei Ziele verfolgt:

- Verwendung der gleichen Einheit für horizontale und vertikale Transformation
- Einfache Anpassung an einen anderen Wavelettyp
- Einfache Portierung auf eine andere Hardware

Um für horizontale und vertikale Transformation das selbe Modul verwenden zu können, muss dieses über vier Konfigurationsregister verfügen. In diesen Registern können Position (horizontaler und vertikaler Offset) und Grösse (Anzahl Zeilen und Spalten) des zu transformierenden Bildes eingestellt werden.
Die Pixeladresse wurde in eine Spaltenadresse und eine Zeilenadresse aufgeteilt. Von der Steuereinheit können diese beiden Adressen dann, auf horizontale oder vertikale Transformation angepasst, zur Adresse im externen Speicher kombiniert werden.
Die Arithmetik des 5-3 Wavelets wurde in einem separaten Block realisiert. Wenn die Transformationseinheit auf ein anderes Wavelet angepasst werden muss, genügt es diesen Block umzuschreiben und mit der Arithmetik des neuen Wavelets zu ersetzen.

In Bild 11 ist die Wavelet Transformation in einem Blockschaltbild weiter aufgelöst. Die Transformation wird durch die Haupt-Statemachine gesteuert. Diese Statemachine kontrolliert das Laden und Speichern der Daten, erzeugt die Steuersignale für das externe RAM und steuert den Transformationscore, welcher die Arithmetik zur Berechnung der Transformation enthält. Von der Haupt-Statemachine werden zudem die Zähler für Zeile und Bildpunkt gesteuert. Das Zurücksetzen aller Zähler wird ebenfalls durch die Haupt-Statemachine ausgelöst.
Die transformierten Daten werden im internen RAM des FPGA's abgelegt. Dieses RAM ist so gross gewählt, dass eine vollständige Zeile gespeichert werden kann. Die Approximations-koeffizienten werden im linken, die Detailkoeffizienten im rechten Bereich der Zeile gespeichert. Dies entspricht der Darstellung des Wavelet transformierten Bildes wie sie in Kapitel 3.2.2 vorgestellt wurde.
Um diese Organisation der transformierten Daten realisieren zu können, werden zwei Zähler benötigt, welche von einer zweiten Statemachine gesteuert werden. Diese Statemachine ist ausschliesslich für das Ablegen der transformierten Daten im internen Speicher zuständig.
Nachdem eine Zeile transformiert ist, wird sie ins externe RAM zurückgeschrieben. Danach wird der Zeilenzähler um eins erhöht und die anderen Zähler werden zurückgesetzt. Nun kann die nächste Zeile transformiert werden.
Dies wird solange fortgesetzt, bis die letzte Zeile transformiert wurde. Danach werden alle Zähler zurückgesetzt und die Haupt-Statemachine geht in den Wartezustand.

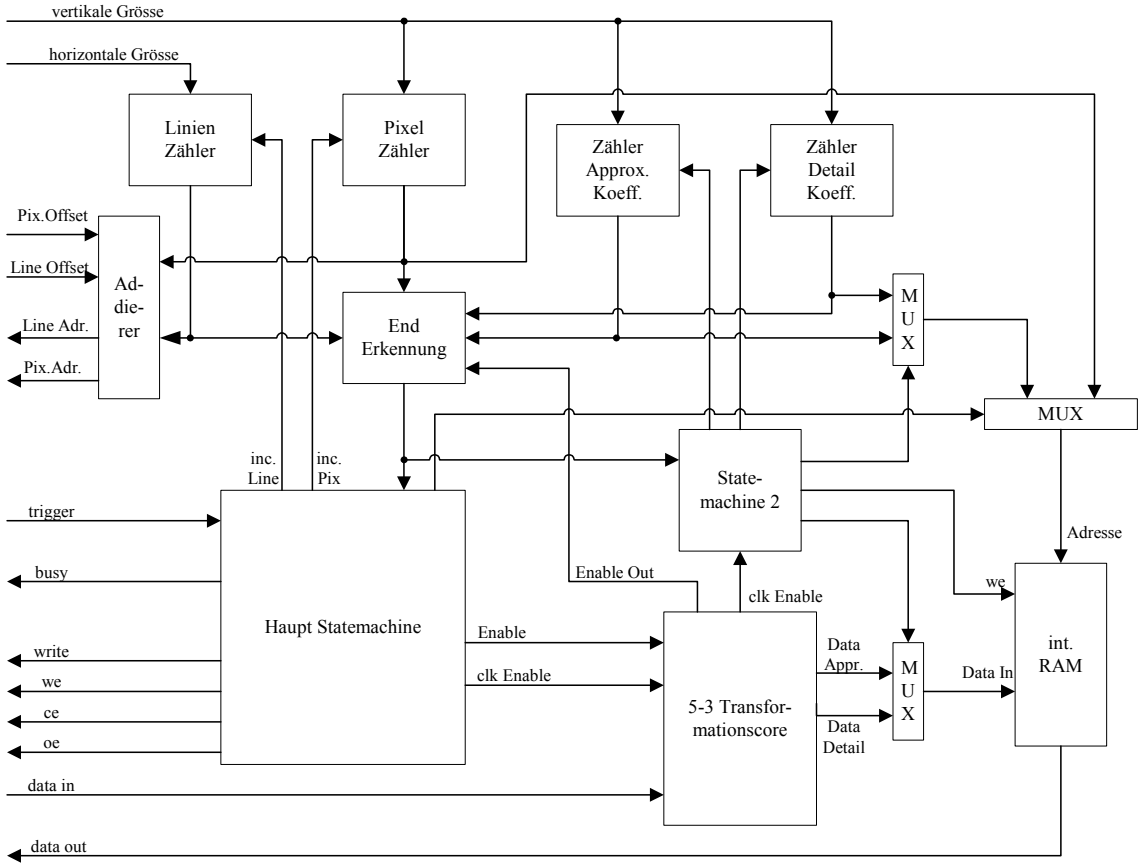
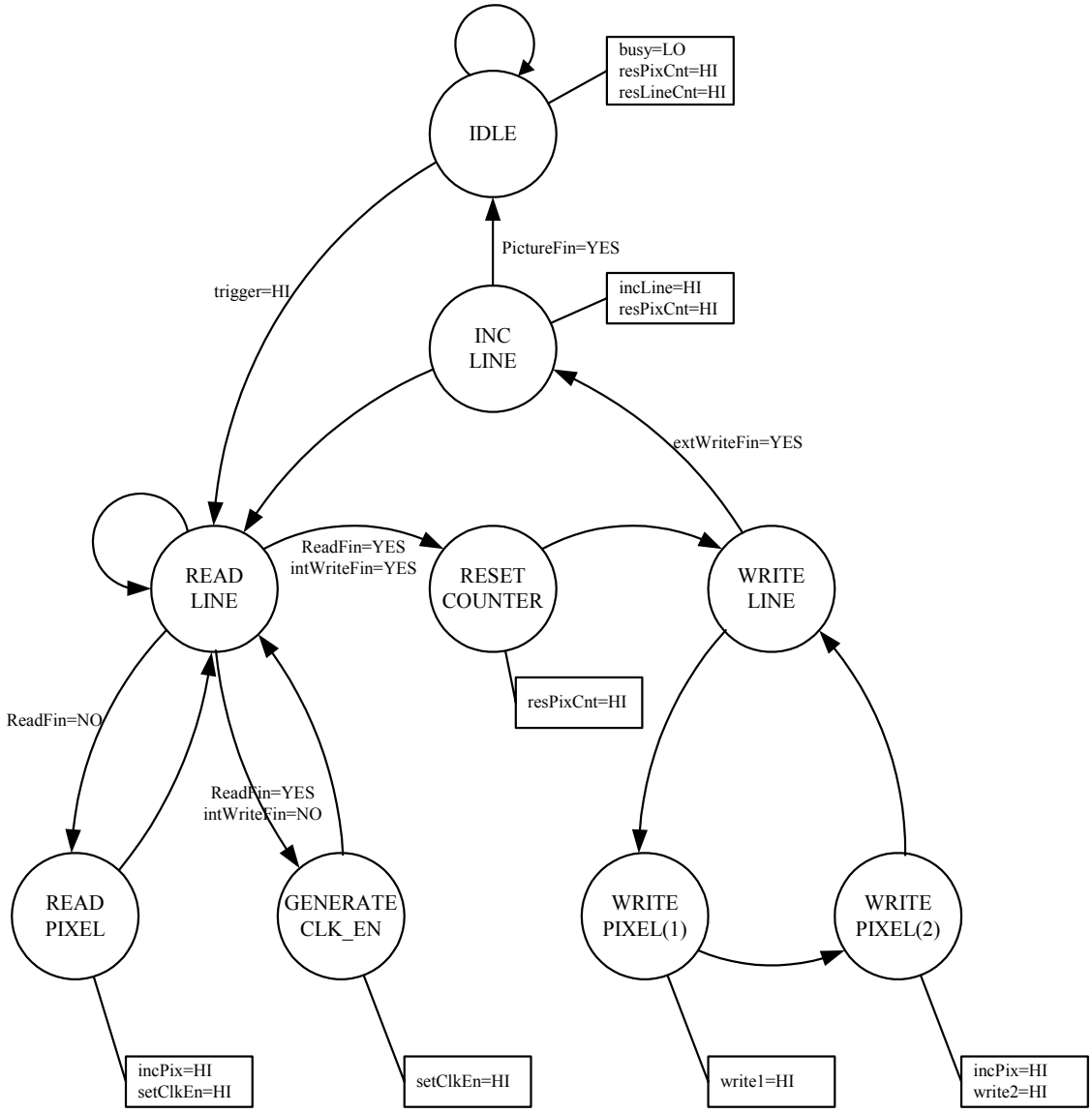


Bild 11: Blockschaltbild der Wavelet Transformation

4.3.3 Haupt-Statemachine der Wavelet Transformation

In Bild 12 ist das Zustandsdiagramm der Haupt-Statemachine der Wavelet Transformation dargestellt. Die nicht weiter bezeichneten Übergänge finden dann statt, wenn keine der anderen Bedingungen erfüllt ist. Zu jedem Zustand werden jeweils die, von der Grundstellung abweichenden, Ausgänge angegeben.



Grundzustand der Ausgänge:

busy=LO	setClkEn=LO	write1=LO
incPix=LO	resPixCnt=LO	write2=LO
incLine=LO	resLineCnt=LO	

Bild 12: Haupt-Statemachine der Wavelet Transformation

4.3.4 Arithmetik der 5-3 Wavelet Transformation

Um die Wavelet Transformation einfach auf andere Wavelets anpassen zu können wurde die Berechnungsarithmetik in einen separaten Block gefasst. Im folgenden Abschnitt ist dieser Block beschrieben.

Die Transformationsvorschrift des 5-3 Wavelets wird hier nochmals wiederholt:

$$\textit{Hochpass:} \qquad x'_{2i+1} = -x_{2i+1} + \frac{1}{2}(x_{2i} + x_{2i+2}) \tag{4.1}$$

$$\textit{Tiefpass:} \qquad x'_{2i} = x_{2i} - \frac{1}{4}(x'_{2i-1} + x'_{2i+1}) \tag{4.2}$$

Bild 13 zeigt die Umsetzung dieser Formel in eine Struktur, die sich in Hardware realisieren lässt. Der Aufwand dafür ist relativ gering, es werden zwei Addierer, zwei Subtrahierer und vier Register benötigt.

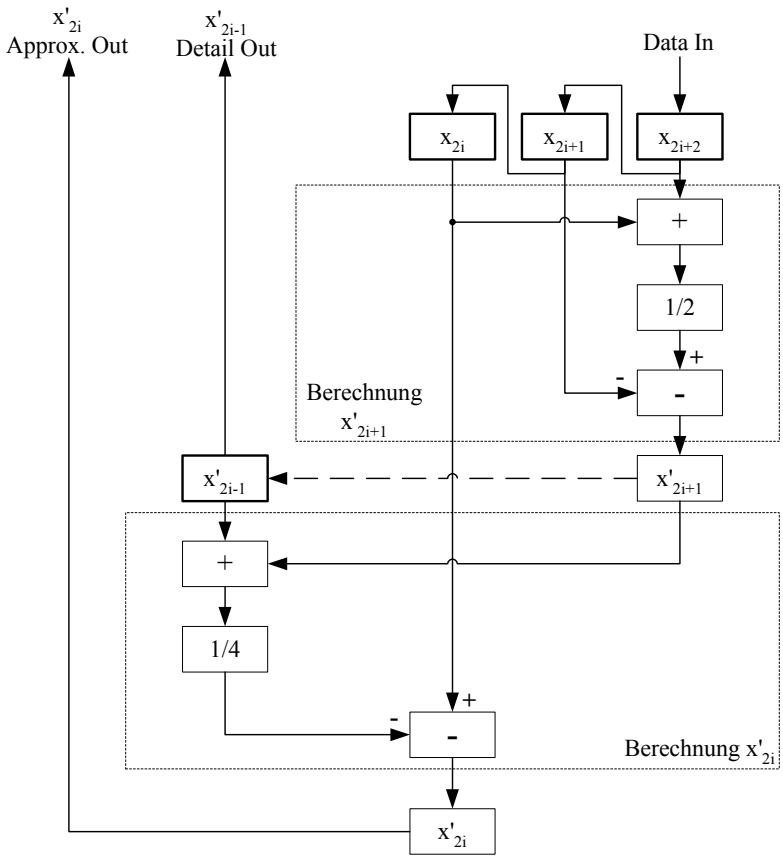


Bild 13: Hardware für Arithmetik der 5-3 Wavelet Transformation

Im gesamten Berechnungspfad sind vier Addierer oder Subtrahierer in Serie vorhanden. Die Berechnungsgeschwindigkeit setzt sich aus den Berechnungszeiten dieser vier Elemente zusammen. Die Divisionen könnte durch Schiebeoperationen ausgeführt werden. Wie in Bild 14 dargestellt ist, können diese Schiebeoperationen durch direkte Verdrahtung realisiert werden und benötigen deshalb keine Rechenzeit.

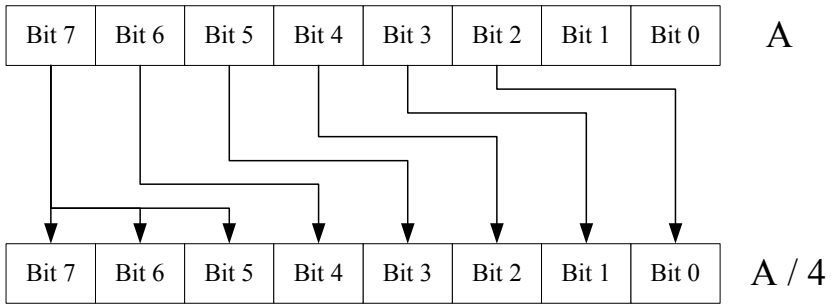


Bild 14: Acht Bit Signed Integer Division durch Vier

Beim Entwurf des 5-3 Transformationscores wurde die Idee verfolgt, die Transformationscores zu kaskadieren. Eine solche Kaskade von Transformationscores kann wesentlich einfacher realisiert werden, wenn die einzelnen Transformationsstufen autonom arbeiten können.
Um diese Unabhängigkeit zu erreichen, muss die reine Arithmetik mit einer Steuerlogik ergänzt werden. Diese Steuerlogik kontrolliert die internen Register, die Ergänzung der Bildränder und erzeugt die Steuersignale für die folgende Transformationsstufe.
Damit die Daten am Ausgang länger stabil bleiben, wurde eine zweite Reihe mit Registern eingebaut. Diese Register werden nur mit jedem zweiten Bildpunkt getriggert. Dadurch bleiben die Daten am Ausgang für mindestens zwei Clock Perioden im gleichen Zustand.
Die Daten können dem Transformationscore bei einer Kaskadierung nicht mit einer fixen Frequenz übermittelt werden. Um im Core trotzdem eine synchrone Schaltung aufbauen zu können, wurde für das Einlesen der Daten ein „Clock-Enable“ Signal eingeführt. Ist dieses Signal High und weist der Clock eine positive Flanke auf, werden die Daten am Eingang übernommen.

In Bild 15 ist das erweiterte Blockschaltbild des 5-3 Transformationscores zu sehen.

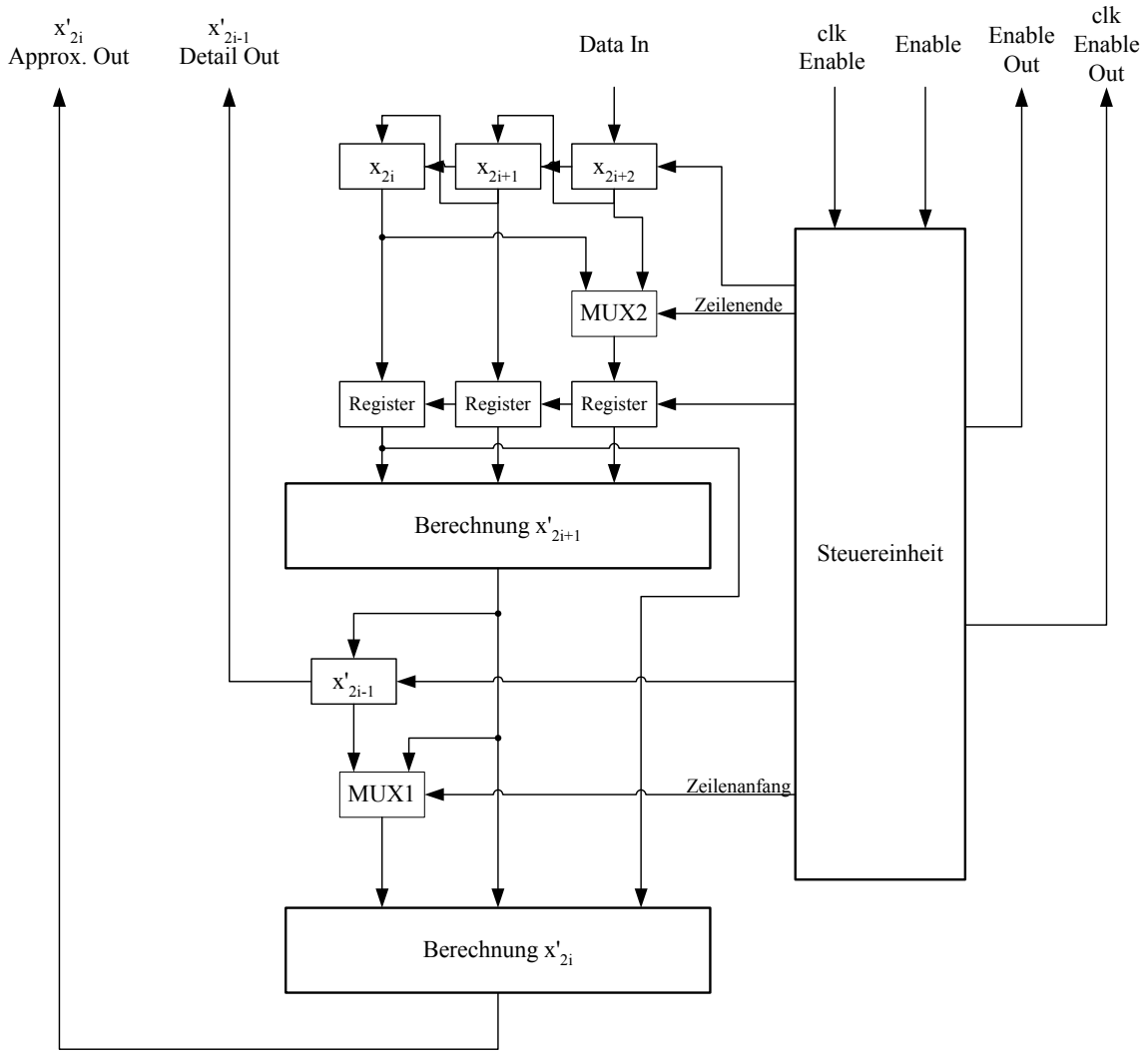


Bild 15: Erweiterte Hardware für die 5-3 Transformation

4.3.5 Ergänzung des Bildes an den Rändern:

Um nach der Transformation wieder gleich viele Bildpunkte zu erhalten, muss das Bild an den Rändern um die Länge der verwendeten Filter minus Eins erweitert werden. Dies soll anhand der diskreten Faltung gezeigt werden.

Wir gehen von einem Signal (beispielsweise eine Bildzeile) $x[k]$ mit $k = 0..M$ aus. Dieses Signal wird mit dem FIR Filter $h[i]$ mit $i = 0..N$ gefiltert. Das gefilterte Signal $y[k]$ wird nach Formel 4.3 berechnet:

$$y[k] = \sum_{i=0}^{N} x[k-i] \cdot h[i] \qquad (4.3)$$

Formel 4.3 zeigt, dass das gefilterte Signal nur noch die Länge *M-N+1* hat. Will man erreichen, dass $x[k]$ und $y[k]$ die gleiche Länge haben, muss das Signal an den Rändern um *N-1* Koeffizienten erweitert werden. In der Bildverarbeitung wird diese Erweiterung häufig symmetrisch, durch Spiegelung an den Rändern, vorgenommen. Das erweiterte Signal $x_e[k]$ lautet:

$$x_e = \left[x[\,\lceil \tfrac{N-1}{2} \rceil\,], x[\,\lceil \tfrac{N-1}{2} \rceil - 1], ..., x[1], x[0], x[1], ..., x[M], x[M-1], ..., x[\,M - \lfloor \tfrac{N-1}{2} \rfloor\,] \right] \qquad (4.4)$$

Da die Ergänzung von der Länge des Wavelets abhängig ist, soll sie direkt im Transformationscore vorgenommen werden. Am linken Rand (i=0) des Bildes lautet die Berechnung:

Hochpass: $$x_1' = -x_1 + \frac{1}{2}(x_0 + x_2) \qquad (4.5)$$

Tiefpass: $$x_0' = x_0 - \frac{1}{4}(x_1' + x_1') \qquad (4.6)$$

Dies entspricht einer symmetrischen Erweiterung um zwei Bildpunkte.
Am rechten Rand (i=*M)* des Bildes lautet die Berechnung:

M ist ungerade, wenn das Bild eine gerade Anzahl Bildpunkte hat.

Hochpass: $$x_M' = -x_M + \frac{1}{2}(x_{M-1} + x_{M-1}) \qquad (4.7)$$

Tiefpass: $$x_{M-1}' = x_{M-1} - \frac{1}{4}(x_{M-2}' + x_M') \qquad (4.8)$$

Dies entspricht einer Ergänzung am rechten Rand um einen Bildpunkt.

Diese Erweiterung ist im Blockschaltbild Bild 15 bereits eingebaut. Durch die Multiplexer wird die Ergänzung implementiert. Mit Hilfe des Enable Signals kann der Transformationscore feststellen, ob er sich am linken oder rechten Rand des Bildes befindet. Dieser Aufbau führt dazu, dass keine zusätzliche Rechenzeit für die Randbehandlung benötigt wird.

4.3.6 Kaskadierung des Transformationscores

Um die Transformationscores für mehrere Transformationslevels kaskadieren zu können, wird in der Steuereinheit für die Ausgangsdaten ebenfalls ein „Enable“ und ein „Clock-Enable“ Signal erzeugt. Mit diesem Konzept kann die Transformation in einer Dimension (horizontal oder vertikal) in allen Transformationslevels gleichzeitig durchgeführt werden.

In Bild 16 ist eine solche Kaskadierung für die gleichzeitige Transformation von drei Levels dargestellt.

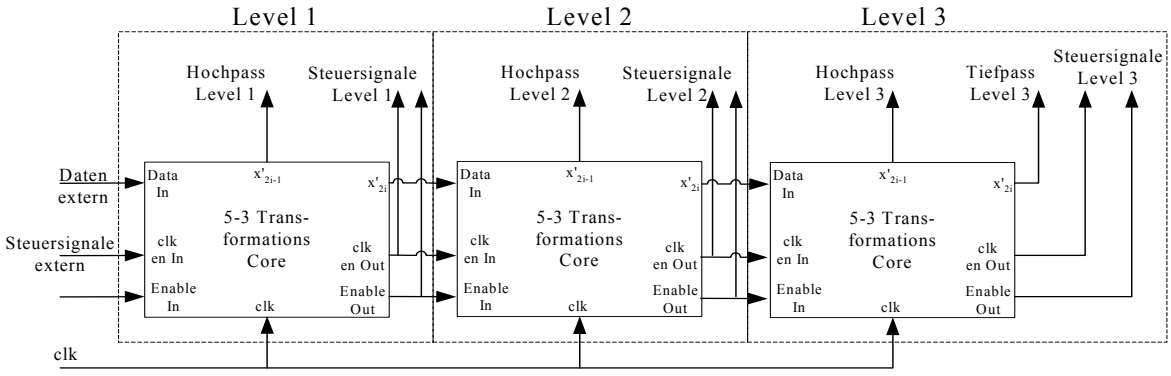


Bild 16: Kaskade der Transformationscores

Eine solche Kaskade von Transformationscores wurde ebenfalls realisiert. Um die transformierten Daten in der richtigen Reihenfolge ablegen zu können, musste das Blockschaltbild in Bild 11 um zwei weitere Zähler ergänzt werden. Der Platzbedarf steigt deshalb nicht nur weil die Transformationscores mehrfach vorhanden sind, sondern auch, weil die Peripherie der Transformationscores komplizierter wird.

4.3.7 Inverse Wavelet Transformation

Für die inverse Wavelet Transformation wurde das gleiche Konzept wie für die Wavelet Transformation benutzt.

In Bild 17 ist das Blockschaltbild der inversen Wavelet Transformation zu sehen. Um Unterschied zur Wavelet Transformation muss bei der Rücktransformation das Bild aus zwei verschiedenen Bereichen des externen Speichers gelesen werden. Dies ist der Grund, wieso die Zähler in Bild 17 etwas anders angeordnet sind als im Blockschaltbild der Wavelet Transformation.

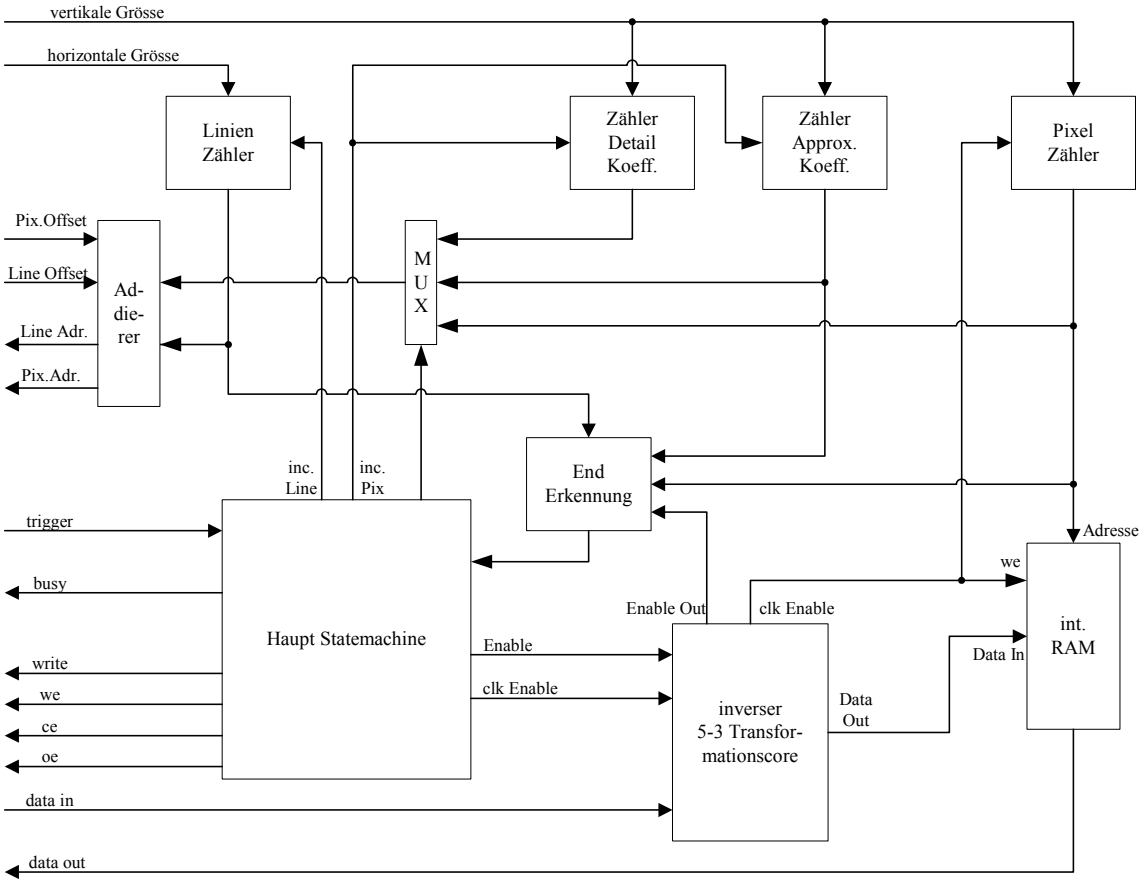


Bild 17: Blockschaltbild der inversen Wavelet Transformation

4.3.8 Arithmetik der Inversen 5-3 Wavelet Transformation

Die Berechnungsvorschrift der Rücktransformation wird hier noch einmal wiederholt. Bei der inversen Wavelet Transformation wird zuerst die Tiefpassfilterung rückgängig gemacht. Danach kann die Hochpassfilterung invertiert werden.

$$\textit{Tiefpass:} \qquad x_{2i} = x'_{2i} + \frac{1}{4}(x'_{2i-1} + x'_{2i+1}) \tag{4.9}$$

$$\textit{Hochpass:} \qquad x_{2i+1} = -x'_{2i+1} + \frac{1}{2}(x_{2i} + x_{2i+2}) \tag{4.10}$$

In Bild 18 wird wiederum die Umsetzung der Formel 4.9 und 4.10 in Hardwarearithmetik gezeigt.

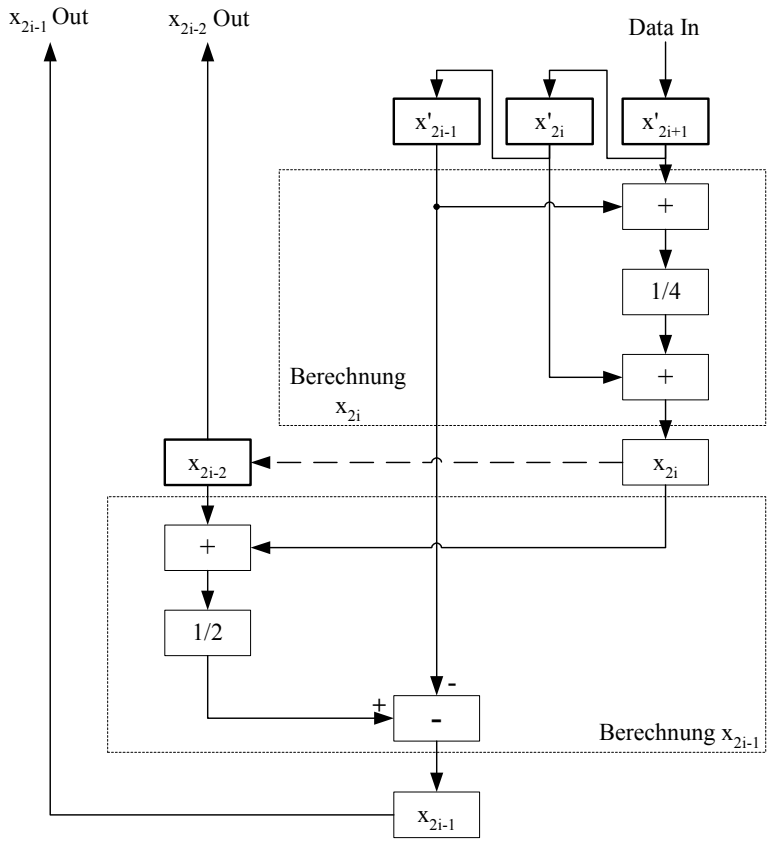


Bild 18: Arithmetik der Inversen 5-3 Transformation

Wie bei der Wavelet Transformation muss die reine Arithmetik mit einer Steuerung ergänzt werden. In Bild 19 ist die erweiterte Logik zu sehen. Die Grundstruktur ist dabei dieselbe wie bei der Wavelet Transformation. Der wesentliche Unterschied besteht darin, dass die Multiplexer an anderen Positionen vorkommen.
Bei der inversen Wavelet Transformation werden die Ausgangsdaten hintereinander in der Zeile abgelegt. Die Aufteilung in einen rechten und linken Bereich, wie sie bei der Wavelet Transformation vorkommt, entfällt. Aus diesem Grund wurde im Ausgang ein Multiplexer eingefügt, welcher zwischen den Resultaten der Tiefpass und Hochpass Rekonstruktion umschaltet.

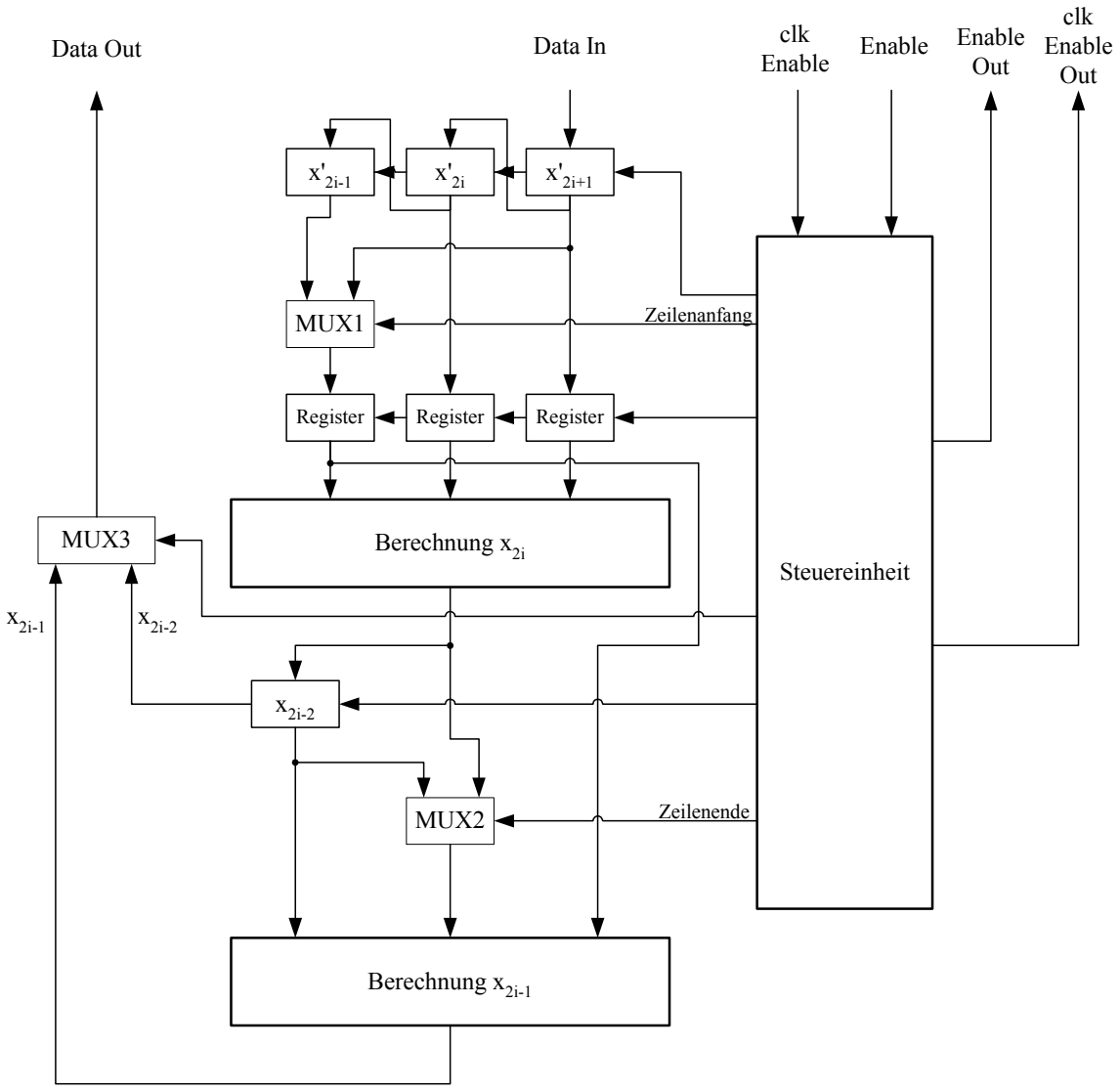


Bild 19: Erweiterte Arithmetik der inversen Transformation

Auch bei der Rücktransformation müssen die Ränder des Bildes ergänzt werden. Dabei ist die Erweiterung so durchzuführen, dass die Ergänzung der Transformation berücksichtigt und wieder rückgängig gemacht wird.
Die entsprechende Randerweiterung kann durch Umstellen der Formeln 4.5 bis 4.8 gefunden werden.

Am linken Bildrand muss eine symmetrische Erweiterung um einen Bildpunkt durchgeführt werden:

$$\textit{Tiefpass:} \qquad x_0 = x'_0 - \frac{1}{4}(x'_1 + x'_1) \tag{4.11}$$

$$\textit{Hochpass:} \qquad x_1 = -x'_1 + \frac{1}{2}(x_0 + x_2) \tag{4.12}$$

Am rechten Bildrand muss eine symmetrische Erweiterung um zwei Bildpunkte durchgeführt werden:

$$\textit{Tiefpass:} \qquad x_{M-1} = x'_{M-1} - \frac{1}{4}(x'_{M-2} + x'_M) \tag{4.14}$$

$$\textit{Hochpass:} \qquad x_M = -x'_M + \frac{1}{2}(x_{M-1} + x_{M-1}) \tag{4.13}$$

Auch der Transformationscore für die inverse Wavelet Transformation kann zur gleichzeitigen Berechnung mehrerer Transformationslevels kaskadiert werden. Dies ist in Bild 20 zu sehen. Da jeder Transformationscore Daten aus dem externen Speicher benötigt, werden die Eingangsregister aus Bild 19 so abgeändert, dass jeweils zwei Bildpunkte gleichzeitig in den Transformationscore eingelesen werden. Der Transformationscore des höchsten Levels erhält beide Bildpunkte aus dem externen Speicher. Die restlichen Transformationscores erhalten jeweils einen Bildpunkt vom übergeordneten Level und einen Bildpunkt aus dem externen Speicher.

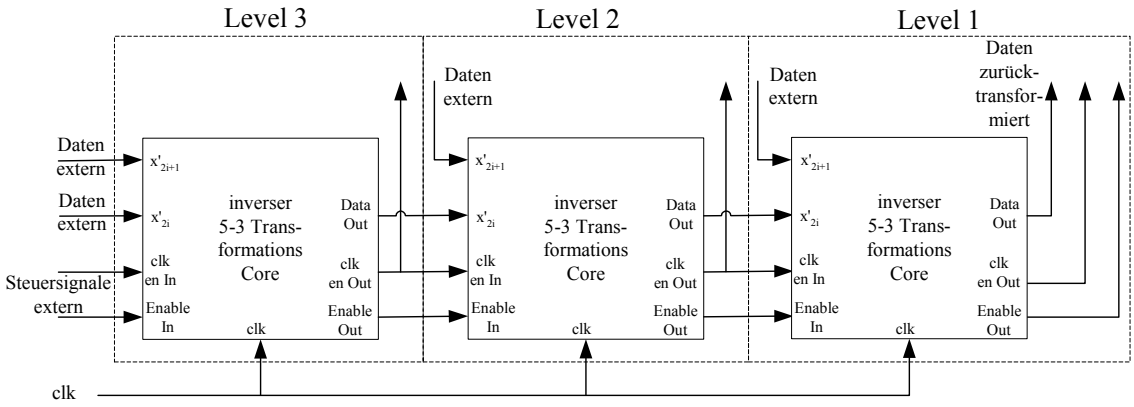


Bild 20: Kaskade der inversen Transformationscores

4.3.9 Internes Block RAM

Der Virtex 300 FPGA verfügt über Block RAM, welches zusätzlich zur konfigurierbaren Logik vorhanden ist. Damit die Ressourcen der konfigurierbaren Logik nicht unnötig verbraucht werden, wurde dieses RAM verwendet um den internen Speicher der Transformationseinheit zu realisieren. Der VHDL Code für den RAM Block wurde gemäss der Dokumentation des XST Synthese Programms aufgebaut [XIL00].

4.4 VHDL Programme für die Wavelet Transformation

Die Wavelet Transformation und die Rücktransformation wurden in je drei Programmteile zerlegt. In Tabelle 5 sind die Bezeichnungen und eine Kurzbeschreibung des jeweiligen Programms aufgelistet. Zudem sind die Hauptprogramme (Top-Level) und die Testprogramme angegeben.

Tabelle 5: VHDL Programme für die Wavelet Transformation und Rücktransformation

Bezeichnung	Beschreibung
wt_ud_1_block8.vhd	Hauptprogramm für Wavelet Transformation und inverse Wavelet Transformation
wt_down_3_8Bit.vhd	Hauptprogramm für Wavelet Transformation mit Kaskade von Transformationscores
wt_1down_8.vhd	Transformationseinheit für die Wavelet Transformation mit Haupt Statemachine, Zählern, Generierung der Steuersignale für das externe RAM usw.
wt_3level_8Bit.vhd	Transformationseinheit für die Wavelet Transformation mit Kaskade der Transformationscores
wt_1up_8.vhd	Transformationseinheit für die inverse Wavelet Transformation
cd53_down_8bit.vhd	5-3 Transformationscore mit Arithmetik
cd53_up_8bit.vhd	Inverser 5-3 Transformationscore mit Arithmetik
ramBlock1024k.vhd	Interner RAM Block
test_wvt.vhd	Hauptprogramm für die Testumgebung
write_test_data.vhd	Schreibt den Testvektor ins externe RAM

4.5 Test der Wavelet Transformation

4.5.1 Kontrolle mit dem Testvektor

Alle VHDL Programme wurden in der Simulation erfolgreich mit dem Testvektor geprüft. Die Testbenches zu den VHDL Entitys wurden mit dem Programm HDL Bencher erstellt. Die Namen der Testbenches wurden aus den Namen der zugehörigen VHDL Programmen und der Erweiterung „_TB“ vergeben. Die zum VHDL Programm „cd53_down_8bit.vhd“ gehörende Testbench heisst demnach „cd53_down_8bit_TB.vhd“.
Auch der Test auf dem Bord, bei dem der Testvektor über ein Testprogramm in den externen Speicher des FPGA’s geschrieben wurde, konnte erfolgreich durchgeführt werden.

4.5.2 Visuelle Verifikation

Da es nicht möglich war den Inhalt des RAM’s auf einen PC zu laden und mit einem Testprogramm die Transformation eines ganzen Bildes zu überprüfen, mussten die weiteren Tests visuell erfolgen. Es hat sich gezeigt, dass praktisch alle Fehler, die während der Entwicklung der Wavelet Transformation aufgetreten sind, zu sichtbaren Artefakten in den Bildern führten. Stimmen beispielsweise die Adressen bei der Rücktransformation nicht vollständig mit den Adressen bei der Transformation überein, so entstehen gut sichtbare Fehler im Bild.
Um eine visuelle Kontrolle durchführen zu können wurde das Bild „Lena“ ausgedruckt, mit der Videokamera aufgenommen und Wavelet transformiert. Parallel dazu wurde für das selbe Bild mit einem C-Programm die Wavelet Transformation berechnet. Die Resultate der beiden Transformationen stimmen überein.
Danach wurde das Bild wieder zurücktransformiert und genau auf Artefakte untersucht. Soweit dies visuell beurteilbar ist, traten auch bei dieser Kontrolle keine Fehler auf.

4.5.3 Evaluation der maximalen Berechnungsgeschwindigkeit

Die maximal zulässige Berechnungsgeschwindigkeit wurde für die Transformationscores ohne Peripherie bestimmt.
Die Daten hierzu stammen aus der Post-Route Simulation und dem Pre-Route und Post-Route Timing Report. Die Timingsimulation für den 5-3 Transformationscore ist in Bild 21 zu sehen.

Tabelle 6: Berechnungsdauer in den Transformationscores

	cd53_down_8bit.vhd	cd53_up_8bit.vhd
Pre-Route Static Timing	16.58 ns oder 60.3 MHz	17.4 ns oder 57.5 MHz
Post-Route Timing	31.25 ns oder 32MHz	29.25 ns oder 34.2 MHz
Simulation	24.7 ns oder 40.5MHz	28.3 ns oder 35.3 MHz

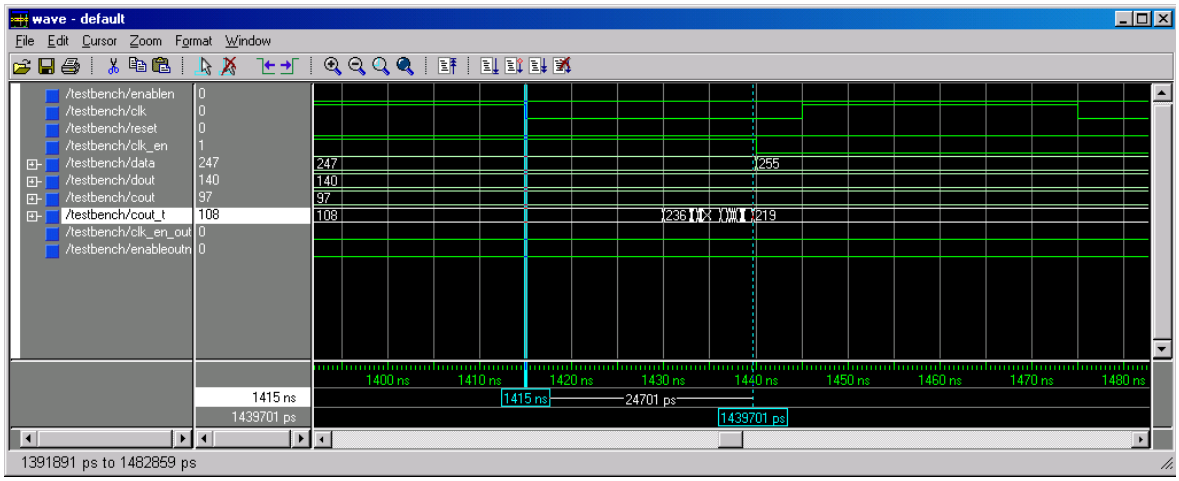


Bild 21: Timingsimulation für 5-3 Simulationscore

4.5.4 Evaluation der Transformationsdauer

Um die Berechnungszeit der Transformation eines ganzen Bildes messen zu können, wurde das Busy-Signal auf einen unbenutzten Ausgang des FPGA geführt. Mit einem Speicheroszilloskop wurde an diesem Ausgang die Transformationszeit gemessen.

Tabelle 7 zeigt eine Übersicht über die gemessenen Zeiten. Dabei ist zu beachten, dass die Transformationsdauer vor allem durch den Speicherzugriff bestimmt wird.

Tabelle 7: Dauer für die Transformation eines Bildes

	Transformation	Rücktransformation
1. Level horizontal - vertikal	247.5 ms	247.5 ms
2. Level horizontal - vertikal	62.3 ms	62.3 ms
3. Level horizontal - vertikal	15.8 ms	15.8 ms
4. Level horizontal - vertikal	4 ms	4.04 ms
Total 4 Levels	329.6 ms	329.64 ms
Total 3 Levels	325.6 ms	325.6 ms
3 Levels kaskadiert	247.7 ms	--
Performancesteigerung durch Kaskadierung	31%	--

4.5.5 Beurteilung der Leistungsfähigkeit

Die ermittelten Berechnungszeiten der Transformationscores zeigen, dass das Herzstück der Transformation auch für die Transformation eines Videostreams geeignet wäre. Die Taktfrequenz des SAA7113 Videodecoder liegt bei 24.57 MHz, somit liegen selbst die schlechtesten Werte aus Tabelle 6 noch gut 28 Prozent darüber.

Als Vergleich soll hier berechnet werden, welche Taktrate auf einem DSP nötig wäre um die gleiche Performance zu erreichen. Wir gehen davon aus, dass die Transformation mit neun Befehlen im DSP berechnet werden könnte (zwei Befehle für das Laden der Daten aus dem externen RAM, fünf Befehle für die Berechnung, zwei Befehle für Speicherung). Zusätzliche Befehle für die Berechnung der Adressen, sowie weiterer Overhaed wird dabei vernachlässigt.

Transformation 1 Level	32 MHz x 9Takte	=	288 MHz Taktrate

Geht man von der Taktrate des Videosignals aus, beträgt die minimal benötigte Taktfrequenz des DSP's 225 MHz.

Die Berechnungszeiten für die Transformation eines ganzen Bildes sind relativ hoch. Dies liegt daran, dass die Speicherzugriffe nur mit einer Taktrate von 12MHz durchgeführt werden. Beim Design des Speicherzugriffs wurden Reserven eingebaut um Probleme beim Speichern zu vermeiden. In einem weiteren Schritt könnten diese Reserven überprüft und wahrscheinlich entfernt werden. Damit könnte die Transformationszeit eines Bildes reduziert werden.

Die Kaskadierung der Transformationscores hat auf Anhieb funktioniert und stellt eine interessante Alternative, vor allem bei Videostreamverarbeitung, dar. Die nötigen externen Speicherzugriffe können markant gesenkt werden. In der Folge sinkt auch die benötigte Zeit um ein Bild zu transformieren. Mit der Kaskade kann ein Bild in der selben Zeit vollständig transformiert werden, die normalerweise für die Transformation auf dem ersten Level benötigt wird.

5 Untersuchung der Möglichkeiten verlustfreier und verlustbehafteter Bildübertragung

In einer Wavelet basierten Bildkompression ist die Wavelet Transformation nur ein Glied einer mehrstufigen Signalverarbeitungskette. Thema dieses Kapitels sind die weiteren Elemente dieser Signalverarbeitungskette.
Anhand der 5-3 Wavelet Transformation wird die Eignung verschiedener Integer to Integer Wavelet Transformationen für verlustfreie und verlustbehaftete Bildkompression untersucht. Dabei werden die unterschiedlichen Eigenschaften der Integer to Integer Transformation sowie der Integer to Integer Transformation mit erhöhter Dynamik aufgezeigt.

5.1 Elemente der Bilddatenkompression

Die Signalverarbeitungskette der Bildkompression kann in Standardelemente zerlegt werden. Diese Aufteilung ist in Bild 22 zu sehen. Nicht jede Kompression verwendet alle Glieder der Signalverarbeitungskette. Bei einer verlustfreien Kompression beispielsweise darf der Quantisierungsschritt nicht durchgeführt werden, weil sonst das Bild nicht mehr perfekt rekonstruierbar wäre. Da es für jeden Teilschritt der Bildkompression verschiedene Varianten mit spezifischen Eigenschaften und Stärken gibt, ist es nötig, beim Entwurf einer neuen Bildkompression die einzelnen Elemente möglichst gut aufeinander und auf die zu komprimierenden Bilder abzustimmen.

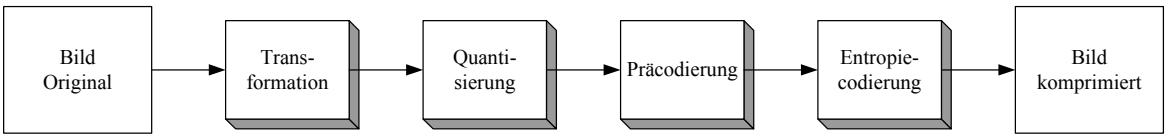


Bild 22: Elemente der Bildkompression

In den nächsten Abschnitten werden diese Teilschritte kurz vorgestellt und deren Aufgabe erklärt.

5.1.1 Transformation

Transformationen für die Bildkompression haben die Aufgabe eine Dekorrelation der Signalwerte des Bildes durchzuführen. Damit soll die Konzentration der Symbole des Bildsignals sowie die Konzentration der geometrischen Verteilung dieser Symbole erhöht werden. Über die Transformation soll das Bild in einer Weise dargestellt werden, welche sich für die Kompression besser eignet. Das Bild wird sozusagen für die Kompression aufbereitet.
Neben der Wavelet Transformation gibt es verschiedene andere Transformationen die gute Eigenschaften für die Bildkompression aufweisen. Eine dieser Transformationen ist die diskrete Cosinus Transformation. Die diskrete Cosinus Transformation wurde im JPEG Kompressions-Algorithmus verwendet.

5.1.2 Quantisierung

Durch die Quantisierung werden bei der Bildkompression mehrere Symbole zu einem einzigen Symbol zusammengefasst. Je mehr Symbole zusammengefasst werden, desto höher ist die erreichbare Kompressionsrate.
Die Quantisierungsvorschrift kann für verschiedene Ziele optimiert werden. Bei der wahrnehmungsoptimierten Quantisierung sollen die entstehenden Signalverzerrungen für das menschliche Auge möglichst nicht sichtbar werden.

5.1.3 Entropiecodierung

Die Entropiecodierung nutzt die statistische Verteilung der Symbole eines Bildes. Dabei wird die mittlere Bitrate minimiert.
Das Grundprinzip der Entropiecodierung besteht darin, jene Symbole, welche in einer Signalfolge häufig vorkommen, mit wenigen Bits darzustellen. Dafür werden die Symbole, welche in einer Signalfolge selten auftreten, mit mehr Bits dargestellt. Dadurch kann der mittlere Aufwand zur Darstellung der Symbole reduziert werden.
Die heute wohl wichtigsten Verfahren für die Entropiecodierung sind die Huffman Codierung [STRU00, S.31ff] und die Arithmetische Codierung [STRU00, S.36ff].

Da die statistische Verteilung der Symbole nach der Wavelet Transformation in den meisten Fällen ähnlich ausfällt, kann die Entropie Codierung mit einer festen Huffmann Codierung durchgeführt werden. Der Aufwand, für jedes Bild die statistische Verteilung zu bestimmen und dann dieser Verteilung entsprechend eine Huffman Codierung zu erstellen, entfällt dadurch.
Für Wavelet basierte Kompressionsverfahren kann aus diesem Grund die Huffman Codierung in einem FPGA sehr effizient und mit wenig Aufwand implementiert werden.

5.1.4 Präcodierung

Die Entropiecodierung stützt sich nur auf die Häufigkeiten einzelner, voneinander unabhängiger Symbole. In einem Bild bestehen jedoch oft unter den Symbolen Abhängigkeiten und Beziehungen (Intersymbolredundanz). Mit der Präcodierung wird versucht diese Abhängigkeiten für die Kompression auszunützen.
Ein bekanntes Beispiel für die Präcodierung ist die Lauflängencodierung (Run Length Coding). Diese Methode eignet sich für Signale in denen oft mehrere gleiche Symbole hintereinander vorkommen. Sind aufeinanderfolgend N Symbole gleich, wird dies mit dem Symbol, gefolgt von der Anzahl N übertragen.

5.2 Untersuchung der Möglichkeiten zur Quantisierung bei der Integer to Integer Wavelet Transformation

Verwendet man für die Wavelet Transformation eine Integer to Integer Transformation, so ist aus den Formeln 3.11 und 3.12 ersichtlich, dass durch eine nachfolgende Quantisierung im Bild Artefakte entstehen. Es ist jedoch nicht sofort klar welcher Art diese Artefakte sind und wie die Bildqualität beeinflusst wird. In diesem Abschnitt werden verschiedene Quantisierungsmethoden untersucht und deren Effekt auf die Bildqualität wird beurteilt.
Die Beurteilung der Bildqualität wird nicht über Kenngrössen, sondern visuell durchgeführt. Für die Untersuchung wird ein C Programm verwendet, welches die Arithmetik der FPGA Wavelet Transformation nachbildet. Die Visualisierung und Auswertung der Daten wird mit Matlab durchgeführt.

5.2.1 Quantisierung mit Division und anschliessender Rundung

Das Bild wird quantisiert indem die Bildkoeffizienten durch einen Faktor *k* dividiert werden. Danach werden die Koeffizienten auf die nächst tiefere ganze Zahl abgerundet.
Ist beispielsweise *k*=2 so wird für die Darstellung der transformierten Koeffizienten ein Bit weniger benötigt. Vor der Rücktransformation wird das quantisierte Bild mit *k* multipliziert. Dadurch kann die mittlere Helligkeit des Bildes ungefähr erhalten werden. Trotzdem entstehen im rücktransformierten Bild sehr gut sichtbare Artefakte. Diese sind in Bild 23 zu sehen.
Die Approximationskoeffizienten enthalten vermutlich mehr Information über das Bild als die Detailkoeffizienten. Deshalb wurde vermutet, dass der Einfluss der Quantisierung in den Detailbereichen des Bildes (Da1, Dd1, Ad1 in Bild 6) kleiner sein könnte als im Approximationsbereich. Aus diesem Grund wurde in einem Versuch die Quantisierung nur in den Detailbereichen vorgenommen. Auch mit dieser abgeänderten Quantisierung konnte die Bildqualität des rücktransformierten Bildes nicht verbessert werden.

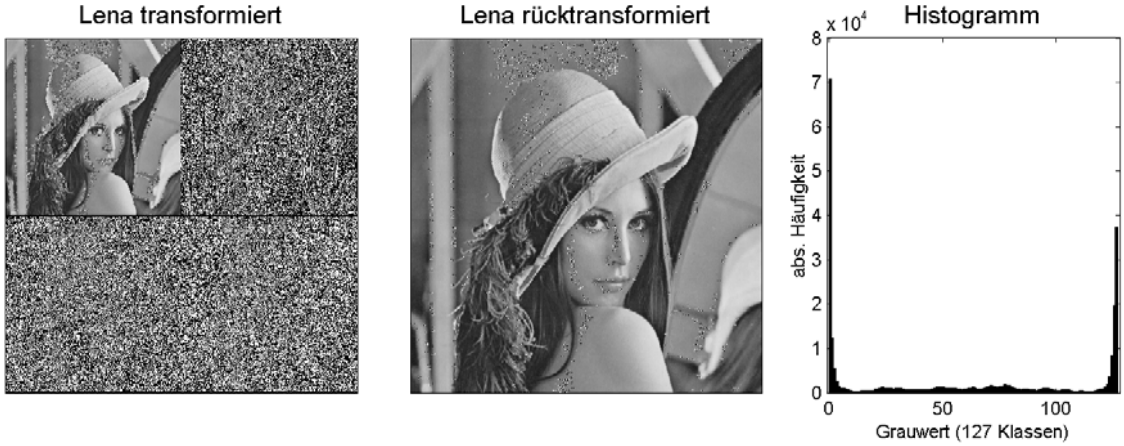


Bild 23: Quantisierung im ganzen Bild mit Division *k*=2

5.2.2 Quantisierung mit Nullbereich-Erweiterung

Im transformierten Bild finden sich in den Detailbereichen viele Koeffizienten, die durch die Modulo Operation auf 255 respektive 0 abgebildet werden. Aus diesem Grund sehen diese Bereiche sehr verrauscht aus. Die Präcodierung könnte effizienter durchgeführt werden, wenn in den Detailbereichen grössere Flächen den gleichen Wert hätten. Um dies zu erreichen wird bei der Quantisierung der Wavelet Transformierten Bilder oft eine Nullbereich-Erweiterung durchgeführt. Alle Zahlen, die in einem Intervall nahe Null liegen, werden Null gesetzt.
Für die Integer to Integer Transformation muss diese Vorschrift etwas abgeändert werden, da auch Zahlen welche nahe von 255 liegen auf Null abgebildet werden müssen.

$$k^*_i = \begin{cases} 0, \text{wenn } k_i \in [0, M] \text{ oder } k_i \in [255 - M, 255] \\ k_i, \text{sonst} \end{cases} \qquad (5.1)$$

In Formel 5.1 meint k^*_i den *i*-ten, quantifizierten Koeffizienten; k_i meint den *i*-ten, nicht quantifizierten Koeffizienten. M ist die Intervallgrenze.

Mit dieser Quantisierung entstehen im transformierten Bild grosse Bereiche mit identischen Koeffizienten. Bild 24 zeigt aber auch, dass auch diese Art der Quantisierung zu gut sichtbaren Artefakten im rücktransformierten Bild führt.

Bild 24: Quantisierung mit Nullbereich-Erweiterung im Detailbereich (M=5)

5.2.3 Schlussfolgerung

Es konnte keine Quantisierungsmethode für die Integer to Integer Wavelet Transformation gefunden werden, welche zu akzeptablen Qualitätsverlusten im rücktransformierten Bild führt. Um dies mathematisch erklären zu können, wären weitergehende Untersuchungen bezüglich der benutzten Modul-Arithmetik nötig. Ein heuristischer Erklärungsversuch könnte folgendermassen aussehen:
Die Koeffizienten des transformierten Bildes können verschiedene Zahlen aus dem ursprünglichen Zahlenstrang repräsentieren. Die Zahlen -257, -1, 255, 511 werden jeweils mit dem Wert 255 dargestellt. Wird nun beispielsweise eine Quantisierung mit Nullbereich-Erweiterung vorgenommen, so entstehen unterschiedlich grosse Quantisierungsfehler. War der ursprüngliche Wert des Koeffizienten –1, so ist der Quantisierungsfehler relativ klein. Falls der Wert des Koeffizienten 255 war, so entsteht jedoch ein viel grösserer Quantisierungsfehler. Dieser grosse Quantisierungsfehler wird auch zu stärkeren Artefakten im rücktransformierten Bild führen.

Für eine Bildkompression mit Quantisierungsstufe ist die Integer to Integer Wavelet Transformation ohne erweiterte Dynamik schlecht geeignet.

5.3 Untersuchung der Möglichkeiten zur Quantisierung der Integer to Integer Wavelet Transformation mit erhöhter Dynamik

Da sich die Integer to Integer Wavelet Transformation für eine nachfolgende Quantisierung nicht eignet, werden in diesem Kapitel Quantisierungsmethoden für die Integer to Integer Wavelet Transformation mit erhöhter Dynamik untersucht.
Basierend auf der Integer to Integer Wavelet Transformation kann die Integer to Integer Wavelet Transformation mit erhöhter Dynamik mit relativ wenig Aufwand in einem FPGA realisiert werden. Dabei muss die Wortbreite der verwendeten Speicher an die erhöhte Dynamik angepasst werden. In der Arithmetik selbst müssen die Modulo Operationen entfernt werden.

5.3.1 Quantisierung mit Division und anschliessender Rundung

Die Quantisierung wird auf die gleiche Art wie in Kapitel 5.2.1 durchgeführt. Die Quantisierung führt zu wesentlich weniger starkem Qualitätsverlust im rücktransformierten Bild. Bis zu einem Divisionsfaktor k=8 treten praktisch keine sichtbaren Qualitätsverluste auf. Bei der hier verwendeten Auflösung der Bilder ist selbst bei Faktor 16 kein grosser Qualitätsverlust zu sehen.

Die folgenden Bilder zeigen den steigenden Qualitätsverlust bei zunehmender Quantisierung. Bei feinerer Auflösung sind die Unterschiede entsprechend schneller zu sehen

Bild 25: Lena ohne Quantisierung

Bild 26: Quantisierung im ganzen Bild mit Division k=4

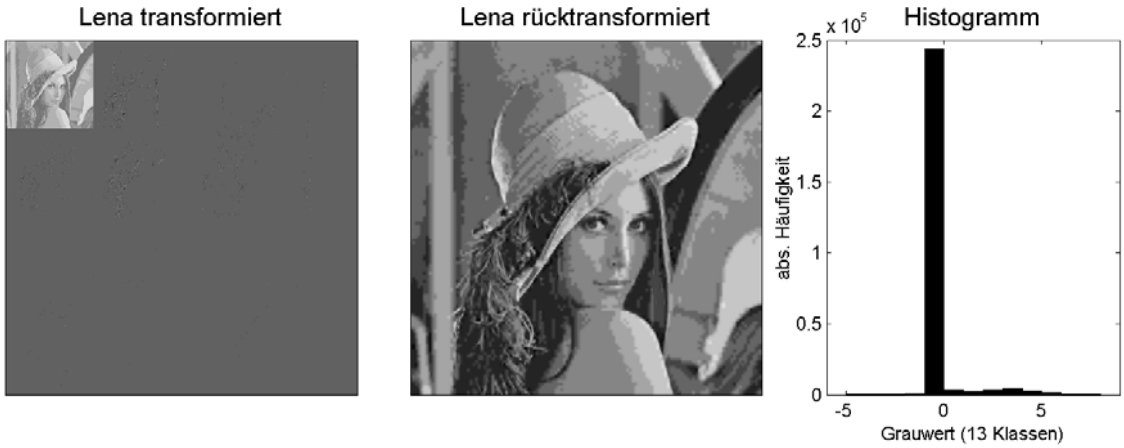


Bild 27: Quantisierung im ganzen Bild mit Division k=32

Die Quantisierung wurde in einem zweiten Versuch für den Approximationsbereich und den Detailbereich unterschiedlich vorgenommen. Damit lassen sich bei gleicher Bildqualität für den Detailbereich wesentlich höhere Quantisierungen realisieren. Im Vergleich zeigt Bild 22 eine Quantisierung im Detailbereich mit k_D= 64, im Approximationsbereich wurde mit k_A=4 quantisiert.

Bild 28: Quantisierung im Detail- und Approximationsbereich unterschiedlich: k_D= 64, k_A=4

Werden Detail- und Approximationsbereich unterschiedlich behandelt, so nimmt die Gesamtdynamik nur dem kleineren Divisionsfaktor entsprechend ab.

5.3.2 Quantisierung mit Nullbereich-Erweiterung

Da bei der Integer to Integer Wavelet Transformation mit erhöhter Dynamik keine Modularithmetik verwendet wird, muss die Formel 5.1 angepasst werden.

$$k^*_i = \begin{cases} 0, \text{wenn } k_i \in [-M, M] \\ k_i, \text{sonst} \end{cases} \qquad (5.2)$$

Auch die Quantisierung mit Nullbereich-Erweiterung führt bei der Integer to Integer Transformation mit erhöhter Dynamik zu akzeptablen Qualitätsverlusten im Bild.

Beim Bild Lena entstehen bis M=3 keine sichtbaren Qualitätsverluste, die Anzahl Nullen wird jedoch um Faktor 5 grösser. Für $M \geq 11$ nimmt die Anzahl Nullen im transformierten Bild nicht mehr wesentlich zu. Das Bild Lena wirkt bei M=11 leicht verschwommen, in der hier verwendeten Auflösung ist dies jedoch nicht zu sehen.

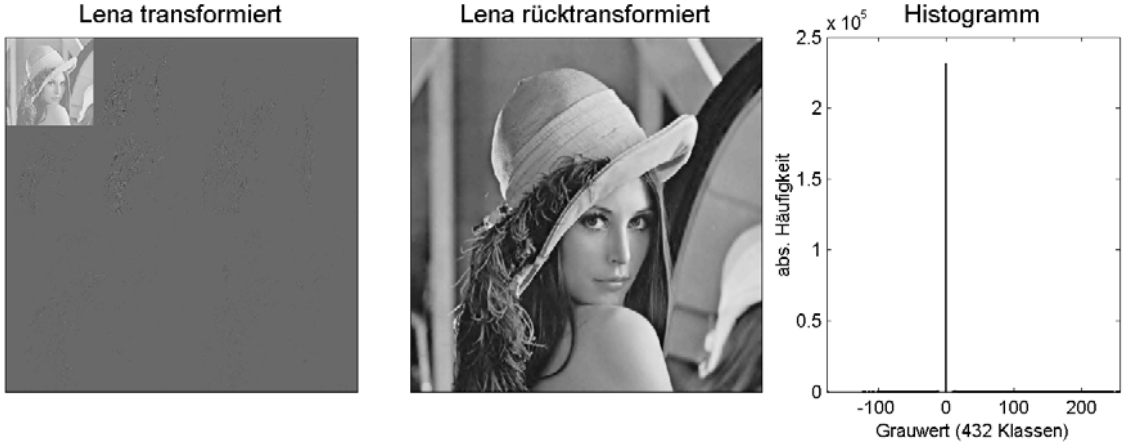


Bild 29: Quantisierung mit Nullbereich-Erweiterung im Detailbereich (M=11)

5.3.3 Schlussfolgerung

Bei der Integer to Integer Wavelet Transformation mit erhöhter Dynamik können die untersuchten Quantisierungen erfolgreich eingesetzt werden. Bei stärkerer Quantisierung entstehen im Bild erwartungsgemäss grössere Fehler. Es bilden sich jedoch keine Kanten oder andere auffällige Artefakte. Dadurch können die Bilder auch bei starker Quantisierung noch mit akzeptabler Qualität rekonstruiert werden.
Die Quantisierung lässt sich mit wenig Aufwand in einem FPGA realisieren. Am einfachsten wird sie beim Einlesen der Daten für die Präcodierung durchgeführt. Für die Quantisierung mit Division sollten Divisionsfaktoren $k=2^N$ gewählt werden, da damit die Dynamik jeweils um N Bits verringert wird. Die Nullbereich-Erweiterung kann mit digitalen Vergleichsoperatoren realisiert werden.

Wird eine Quantisierungsstufe im Bildkompressionssystem vorgesehen, so drängt sich die Integer to Integer Wavelet Transformation mit erhöhter Dynamik auf. Eine unterschiedliche Quantisierung von Approximations- und Detailbereich scheint erfolgversprechend zu sein. Eventuell würde es sich lohnen, die Detailbereiche für jeden Level unterschiedlich zu behandeln.

5.4 Präcodierung auf einem FPGA

Die Verwendung einer Präcodierung kann die Kompressionsrate um Faktoren vergrössern. Der verwendete Präcodierungsalgorithmus sollte möglichst gut an die statistischen Eigenschaften der zu komprimierenden Bilder oder der verwendeten Bildtransformation angepasst werden.
Im folgenden werden zwei, auf die Wavelet Transformation ausgelegte Präcodierungsalgorithmen kurz vorgestellt und auf Ihre Eigenschaft zur Implementierung in einem FPGA untersucht.

5.4.1 Quadtree Codierung

Mit der Quadtree Codierung können zweidimensionale Signale, in denen ein bestimmtes Symbol (im folgenden Spezialsymbol genannt) viel häufiger vorkommt als alle anderen Symbole, effizient codiert werden. Da bei der Wavelet Transformation das Symbol Null viel häufiger auftritt als die restlichen Symbole, eignet sich diese Codierung für Wavelet transformierte Bilder gut.
Die Quadtree Codierung arbeitet hierarchisch. Die Codierung beginnt auf der untersten Hierarchiestufe H_0. Dafür wird das Originalbild zuerst in Unterbilder der Grösse 2x2 partitioniert. Jedes dieser Teilbilder wird dahingehend untersucht, ob ausschliesslich das Spezialsymbol vorkommt. Entsprechen alle vier Zeichen des Teilbildes dem Spezialsymbol, so wird dieses Teilbild auf der nächsten Ebene H_1 mit einer Null repräsentiert. Entsprechen nicht alle vier Zeichen dem Spezialsymbol so wird das Teilbild auf der Ebene H_1 mit einer Eins dargestellt.
Auf der Hierarchiestufe H_1 werden wieder Unterbilder der Grösse 2x2 gebildet. Diese Unterbilder werden in der Ebene H_2 als Null dargestellt, wenn alle Zeichen des Unterbildes Nullen enthalten. Sind ein oder mehrere Zeichen eines Unterbildes nicht gleich Null, so wird dieses Unterbild in der Ebene H_2 als Eins eingetragen. Dieses Verfahren wird solange fortgesetzt, bis keine Unterbilder mehr gebildet werden können.
In der Folge entsteht eine Baumstruktur, welche ausgehend von der Wurzel (oberste Hierarchiestufe) übertragen wird. Die Übertragung der Äste wird abgebrochen, wenn in einem Ast eine Null auftritt.
Bild 30 zeigt ein Beispiel für die Quadtree-Codierung. Detailliertere Erklärungen zur Quadtree Codierung und deren Anwendung auf Wavelet transformierte Bilder finden sich in [STRU00, S.63ff sowie S.188f].

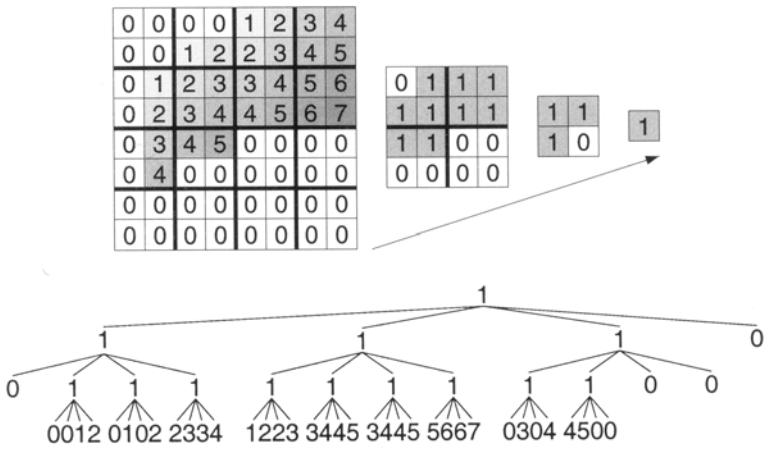


Bild 30: Beispiel einer Quadtree Codierung mit zugehöriger Baumstruktur (Quelle: [STRU00], S.64)

Die Quadtree Codierung erfordert für alle Ebenen H_k mit $k>0$ das Speichern einer Bit-Karte. Um effizient codieren zu können, sollte diese Bit-Karte im internen RAM des FPGA's abgelegt sein. Ausgehend von einem Bild der Grösse 640x480 wird hier der Speicherbedarf für die Quadtree-Codierung ermittelt.

Ebene H_1:	640 x 480 / 16	=	19 200 Bits
Ebene H_2:	19 200 / 16	=	1 200 Bits
Ebene H_3:	1 200 / 16	=	75 Bits
Ebene H_4:	75 / 16	=	5 Bits
Total:			**20 480 Bits**

Der Virtex XCV 300 FPGA enthält 65 536 Bits Block RAM, der Speicherbedarf für die Quadtree Codierung kann demzufolge abgedeckt werden. Mit der Quadtree Codierung könnte ein Bild der Grösse 640 x 480 in einem Schritt codiert werden. Die Grundstruktur des Algorithmus würde dabei folgendermassen aussehen:

- Lesen aller 2x2 Teilbilder aus dem externen Speicher und erstellen der Bit-Karte für Ebene H_1
- Erstellen der Bit-Karten H_2 bis H_4
- Übertragen der signifikanten Teile der Bitkarten H_4 bis H_1, angefangen bei H_4
- Lesen und übertragen der signifikanten Bildpunkte

Um ein Bild codieren zu können, müssten dementsprechend alle Bildpunkte im Maximum zweimal aus dem externen Speicher gelesen werden.
Die Quadtree Codierung lässt sich also mit vertretbarem Aufwand in einem FPGA realisieren. Voraussetzung für eine effiziente Quadtree Codierung ist genügend interner Speicher um die Bitkarten abzulegen.

5.4.2 Embedded Zerotree Codierung

Die Embedded Zerotree Wavelet Codierung (EZW) wurde 1993 von J. M. Saphiro [SHAP93] vorgestellt. Drei Jahre später publizierten A. Said und W. A. Pearlman [SAI96] eine abgeänderte und effizientere Version des ursprünglichen EZW Algorithmus. Said's abgeänderter EZW Algorithmus weist den Vorteil auf, dass anstelle einer Arithmetischen Codierung eine Binäre Codierung verwendet wird.
Die Embedded Zerotree Codierung baut auf den Eigenschaften der Wavelet Transformation auf und wird heute in sehr vielen Wavelet basierten Bildkompressionen verwendet [CAL1, REI1, RIT01, SUT99]. Auch der JPEG2000 Standard verwendet eine, der EZW Codierung ähnliche Präcodierung [STRU00, S.195ff].

Der EZW Algorithmus nach Said und Pearlman wird im folgenden grob beschrieben, eine genauere Beschreibung findet sich in [SAI96] (siehe Anhang B).

Als erstes wird das Wavelet transformierte Bild in Teilbilder zerlegt. Diese Zerlegung erfolgt hierarchisch, den Transformationslevels entsprechend. Das durch den höchsten Transformationslevel entstehende Teilbild wird als Wurzel (Root) bezeichnet. Jeder Punkt im Rootbild hat entweder kein Kind oder vier Kinder. Dadurch entsteht ein hierarchischer Baum in dem jeder Bildpunkt einer Wurzel zugeordnet werden kann. Bild 31 zeigt diese hierarchische Struktur.

Pixel 1 in Bild 31 ist Element des Rootbildes und hat folgende Kinder: 11, 12, 13, 14. Die vier Kinder eines Pixels werden als Unterbild oder Set des entsprechenden Pixels bezeichnet. Pixel 11, 12, 13, 14 bilden dementsprechend das Unterbild von Pixel 1.
Pixel 2 und 3 sind ebenfalls Element des Rootbildes. Pixel 2 hat folgende Kinder: 21, 22, 23, 24. Pixel 11 ist Kind von Pixel 1, hat aber selber auch wieder Kinder: 111, 112, 113, 114. Ausgehend vom Rootbild entsteht auf diese Weise eine Baumstruktur, die mit jedem Unterbild verfeinert wird.

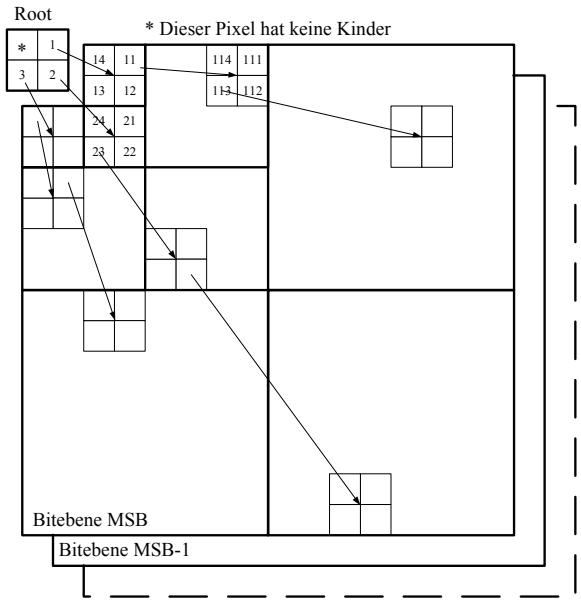


Bild 31: Hierarchische Struktur der Zerotree Methode

Im EZW Algorithmus werden die Bitebenen der Bildpunkte nacheinander untersucht. Bitebene *N* enthält das *N*-te Bit aller Bildpunkte. Als Erstes wird die Bitebene mit den MSB's aller Bildpunkte analysiert, *N* ist zu Beginn also maximal.
Für jede Bitebene werden die Pixel, der Hierarchie in Bild 1 folgend, analysiert. Die Untersuchung eines Astes wird abgebrochen, wenn alle Unterbilder nur Nullen enthalten. Sind alle Äste untersucht, so wird die nächst tiefere Bitebene analysiert.
Es werden die Beträge der Bildpunkte verwendet, die Vorzeichen sind in einer separaten Ebene abgelegt.

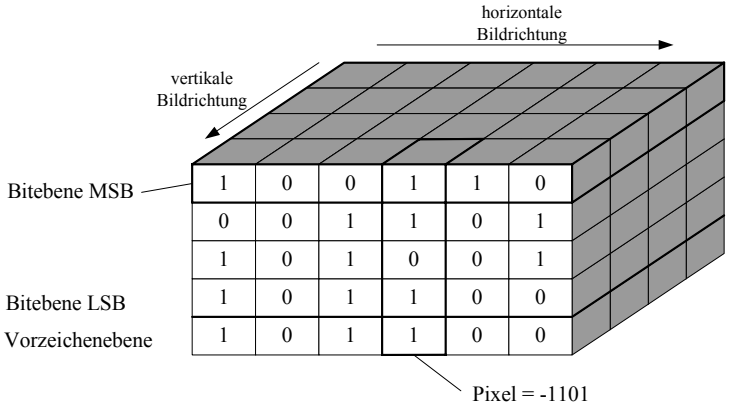


Bild 32: Bitebenen eines Bildes mit 4x6 Bildpunkten und 4+1 Bit Auflösung

Zu Beginn der Codierung gelten alle Pixel als nicht signifikant. Ein Pixel wird dann als signifikant erklärt, wenn er in der Bitebene eine Eins aufweist. Hat der Status von nicht signifikant auf signifikant gewechselt, so bleibt dieser Status bis zum Ende der Codierung erhalten.

Für die Durchführung des EZW Algorithmus werden drei verschiedene Listen geführt:

- Liste der nicht signifikanten Sets LIS
  In dieser Liste werden die Unterbilder abgelegt, in denen kein signifikanter Pixel enthalten ist.

- Liste der signifikanten Pixel LSP
  Enthält ein Unterbild signifikante Pixel, so wird es in der Liste LIS gelöscht und die signifikanten Pixel werden in der Liste LSP abgelegt.

- Liste der nicht signifikanten Pixel LIP
  Enthält ein Unterbild signifikante Pixel werden die restlichen, nicht signifikanten Pixel in dieser Liste gespeichert.

Der EZW Algorithmus enthält vier Teilschritte:

- **Initialisierung:**
  N wird auf die Bitbreite der Daten (MSB) gesetzt und übertragen.
  Die Liste LSP enthält keine Einträge. In die Liste LIP werden die Elemente des Root-Bildes eingetragen. Die Unterbilder des Rootbildes werden in der Liste LIS eingetragen.

- **Sortieren:**
  - Alle Einträge in der Liste LIP, welche in der Bitebene *N* eine Eins enthalten werden in LIP gelöscht und in LSP eingetragen.
  - Die Unterbilder in der Liste LIS werden untersucht. Für Unterbilder bei denen ein oder mehrere Pixel in der Bitebene *N* eine Eins enthalten, wird folgendes durchgeführt:
    - Diejenigen Pixel des Unterbildes, welche in der Bitebene *N* eine Eins enthalten werden in der Liste LSP eingetragen. Die restlichen Pixel des Unterbildes werden in der Liste LIP eingetragen.
    - Der Eintrag des Unterbildes wird in der Liste LIS gelöscht. Dafür werden die Unterbilder des Unterbildes in der Liste LIS eingetragen.

- **Verfeinern:**
  - Für alle Einträge, welche im letzten Sortierungsschritt in die Liste LSP eingetragen wurden, wird deren Vorzeichen und Position übertragen.
  - Für alle anderen Einträge der Liste LSP wird deren Bit in der *N*-ten Bitebene übertragen.

- **Bitebene wechseln:**
  N wird um Eins verringert. Danach wird bei Schritt *Sortieren* weitergearbeitet.

In dieser Beschreibung ist der EZW Algorithmus vereinfacht dargestellt. Die grundlegende Arbeitsweise kann jedoch trotzdem verstanden werden. Mit dem EZW Algorithmus werden zuerst jene Informationen übertragen, welche eine grosse Amplitude enthalten. Pixel, welche eine Null enthalten, werden nicht übertragen.

Ein grosser Vorteil der EZW Codierung liegt darin, dass die Qualität des Übertragenen Bildes mit zunehmender Übertragungsdauer steigt. Deshalb kann die Übertragung eines Bildes an einem beliebigen Punkt abgebrochen werden. Das Rücktransformierte Bild hat dann eine Qualität die der Übertragungsdauer entspricht.
Ein Nachteil der EZW Codierung liegt in den vielen Speicherzugriffen, welche während des Codierungsvorgangs ausgeführt werden müssen. Jeder Bildpunkt muss im Maximum *M* mal gelesen werden. *M* entspricht der Bitbreite des zu codierenden Bildes. Bei der Implementierung in einem FPGA sollte aus diesem Grund das gesamte zu codierende Bild im internen RAM des FPGA vorhanden sein.
Zur Zeit gibt es keine FPGA's mit entsprechend grossen RAM Blöcken. Will man den EZW Algorithmus in einem FPGA realisieren, so muss das Bild in Teilbilder partitioniert werden. Damit die hierarchische Darstellung der Bilder den Anforderungen des EZW Algorithmus entspricht, muss die Wavelet Transformation ebenfalls in diesen Teilbildern durchgeführt werden.

5.5 Elemente der Signalverarbeitungskette für verlustfreie Bildübertragung

Soll ein Bild auf der Empfängerseite perfekt rekonstruiert werden, so muss die Bildübertragung verlustfrei erfolgen. In diesem Fall darf keine Quantisierung angewendet werden. Aus diesem Grund kann für die verlustfreie Bildübertragung die Integer to Integer Wavelet Transformation problemlos eingesetzt werden.

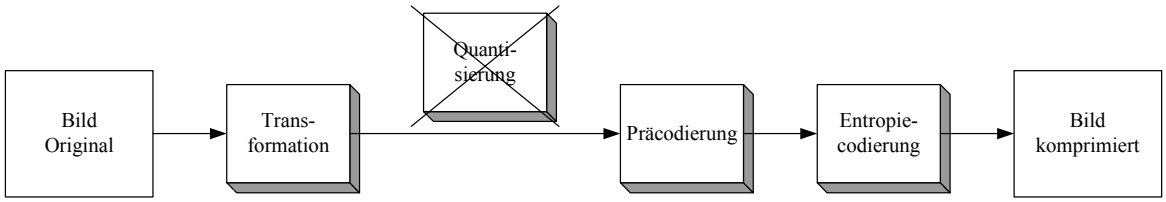


Bild 33: Elemente der verlustfreien Bildkompression

Ohne Quantisierung entstehen in der Bildgeometrie nur kleine Bereiche in denen ausschliesslich Nullen vorhanden sind. Aus diesem Grund wird die Quadtree Codierung nicht sehr effizient angewendet werden können. Die Embedded Zerotree Codierung führt gemäss Calderbank et al. sowie Reichel et al. [CAL1, REI1] auch für verlustfreie Bildkompression zu guten Resultaten. Die Übertragung des Bildes darf bei der EZW Codierung für verlustfreie Bildübertragung nicht vorzeitig abgebrochen werden, da sonst Verluste im Bild entstehen.

Mit der Entropie Codierung werden keine Informationen aus dem Bild entfernt. Deshalb lässt sich die Entropie Codierung auch für verlustfreie Bildübertragung anwenden.

Die erreichbare Kompressionsrate ist für die verlustfreie Bildübertragung stark beschränkt. In Sahni et al. [SAH97, Tabelle 10, S.36] findet sich ein Vergleich verschiedener verlustfreier Bildkompressionen. Der Vergleich wurde mit 26 unterschiedlichen Bildern durchgeführt. Dabei variieren die Kompressionsraten von 1.34 bis 5.5. Die meisten Bilder konnten jedoch mit Kompressionsraten von 1.75 bis 2.5 komprimiert werden.

Für eine verlustfreie Bildübertragung können, bis auf die Quantisierung, alle Elemente der Signalverarbeitungskette für Bildkompression verwendet werden. Da keine Quantisierung durchgeführt werden kann, sind die erreichbaren Kompressionsraten stark beschränkt.

5.6 Elemente der Signalverarbeitungskette für verlustbehaftete Bildübertragung

Die Quantisierung stellt für die verlustbehaftete Bildübertragung ein wesentliches Teilelement dar. Deshalb muss für eine Wavelet basierte, verlustbehaftete Bildübertragung mit einer Integer to Integer Wavelet Transformation mit erhöhter Dynamik, oder einer Floatingpoint Wavelet Transformation gearbeitet werden. Bild 34 zeigt die Elemente der verlustbehafteten Bildkompression. Diese ist mit der in Bild 22 gezeigten Signalverarbeitungskette identisch.

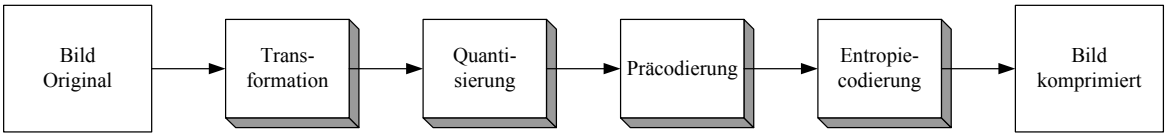


Bild 34: Elemente der verlustbehafteten Bildkompression

Sollen höhere Kompressionsraten erreicht werden, so muss eine verlustbehaftete Bildübertragung verwendet werden. Nimmt man einen entsprechenden Qualitätsverlust in Kauf, so können praktisch beliebig hohe Kompressionsraten erreicht werden. Bei akzeptabler Bildqualität können gemäss B. Burke Hubbard [BUR96] Kompressionsfaktoren bis zu 35:1 realisiert werden.

6 Aufbau einer seriellen Bildübertragungsstrecke mit Datenkompression

Um die in Kapitel 5 ausgeführten Überlegungen in der Praxis zu testen, wurde ein Konzept für eine serielle Schnittstelle entwickelt und im FPGA implementiert. Zusätzlich wurde eine Entropiecodierung entworfen, welche die Eigenschaften der Integer to Integer Wavelet Transformation möglichst optimal ausnützt.

6.1 Wahl der seriellen Schnittstelle und Design des Protokolls

Das XSV-300 Entwicklungsboard bietet verschiedene Schnittstellen um eine serielle Verbindung aufzubauen. Neben einer Standard RS232 gibt es eine PS-2 Schnittstelle und einen USB Bustreiber. Der Entscheid, welche Schnittstelle für die Bildübertragung benutzt werden soll, fiel Aufgrund der erreichbaren Übertragungsrate zu Gunsten des USB Ports aus.

6.1.1 Beschreibung der USB Schnittstelle

Der USB Standard beschreibt eigentlich keine serielle Schnittstelle, sondern ein serielles Bussystem. Auf dem XSV-300 Entwicklungsboard ist jedoch nur ein Treiberbaustein vom Typ PDIUSB11A vorhanden, welcher die Logikpegel am Ausgang des FPGA's in ein USB konformes, differentielles Signal wandelt. Dieser Treiberbaustein stellt nur die physikalische Ebene des USB Busses zur Verfügung.
Wenn das XSV-300 Entwicklungsboard an einen USB Bus angeschlossen werden soll, müssen die höheren Kommunikationsebenen im FPGA realisiert werden.
Um eine serielle Übertragung zwischen zwei XSV-300 Boards zu realisieren wird die Busfunktionalität des USB jedoch nicht zwingend benötigt. Um eine einfach zu handhabende und möglichst schlanke Schnittstelle mit hoher Übertragungsrate zu erhalten, wurde ein eigenes Übertragungsprotokoll entworfen. Dabei wird auf der physikalischen Ebene der USB Bustreiber benutzt.

Die physikalische Ebene der USB Schnittstelle ist im USB Standard 1.1 für zwei verschiedene Übertragungsraten definiert. Im Low Speed Modus beträgt die Übertragungsrate 1.5Mbps, im High Speed Modus ist sie auf 10Mbps festgelegt. Der Treiber PDIUSB11A unterstützt beide Übertragungsraten. Für die Bildübertragung wurde der Treiber auf 10Mbps programmiert. Damit wird jedoch lediglich die Flankensteilheit des differentiellen Bussignals verändert.

Bild 35 zeigt den Aufbau der USB Verbindung mit dem Treiberbaustein PDIUSB11A. Die USB Verbindungskabel sind mit vier Leitungen ausgestattet. Neben zwei Leitungen für das differentielle Signal werden Masse und eine 3.3V Speisung geführt. Mit diesen zwei Leitungen besteht die Möglichkeit ein angeschlossenes Gerät mit Strom zu versorgen.

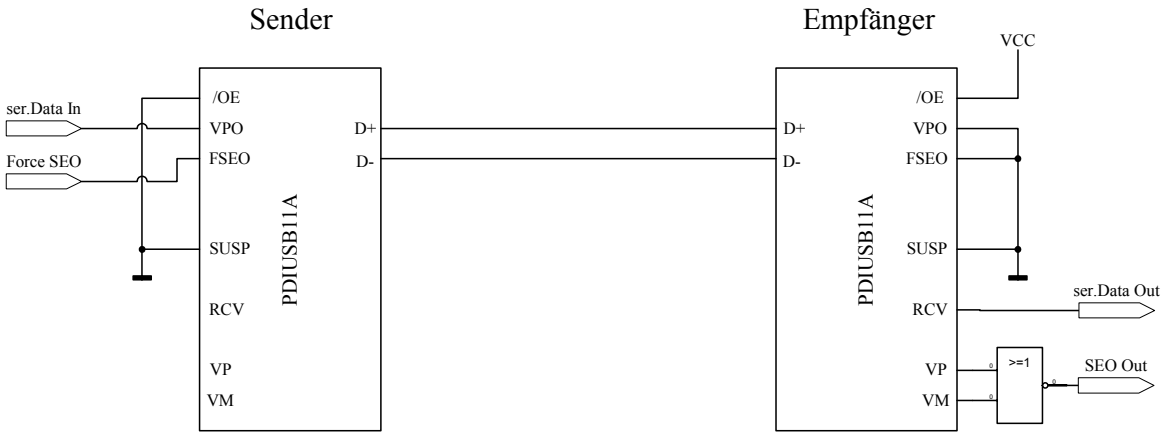


Bild 35: Schnittstellentreiber PDIUSB11A

In Tabelle 8 sind die Funktionen der Anschlüsse des Treiberbausteins PDIUSB11A aufgelistet. Neben den Ein- und Ausgängen für die serielle Übertragung ist vor allem der Eingang FSEO (Force Single Ended Zero) von Bedeutung. Mit diesem Eingang können beide differentiellen Leitungen auf negatives Potential gebracht werden. Dieser Zustand kann auf der Empfängerseite über die Anschlüsse VP und VM detektiert werden. Damit steht für die Übertragung neben logisch Null und logisch Eins ein dritter Zustand – SEO – zur Verfügung.

Tabelle 8: Anschlüsse des USB Schnittstellentreibers PDIUSB11A

Anschluss	Typ	Zustand		Beschreibung
/OE Output Enable	Eingang	0		Ausgänge freigeschaltet; Treiber arbeitet als Sender
		1		Ausgänge auf Tristate; Treiber arbeitet als Empfänger
SUSP Suspend	Eingang	0		Treiber aktiv
		1		Treiber nicht aktiv, Ausgänge auf Tristate
VPO	Eingang	0		Eingang für serielle Daten; Logisch 0 wird übermittelt
		1		Eingang für serielle Daten; Logisch 1 wird übermittelt
FSEO Force SEO	Eingang	0		Ausgang gemäss VPO
		1		Ausgang auf SEO (Single Ended Zero)
RCV Receive	Ausgang	0		Entspricht einem Logisch 0 am differentiellen Eingang
		1		Entspricht einem Logisch 1 am differentiellen Eingang
VP, VM	Ausgang	VP	VM	
		0	0	SEO (Single Ended Zero)
		0	1	Low Speed Verbindung erkannt
		1	0	High Speed Verbindung erkannt
		1	1	Fehler
D+	Eingang / Ausgang			Differentieller Ein-/ Ausgang Plus
D-	Eingang / Ausgang			Differentieller Ein-/ Ausgang Minus

6.1.2 Schnittstellenprotokoll

Das verwendete Schnittstellenprotokoll stützt sich auf die Eigenschaft des USB Treibers, drei verschiedene Zustände übertragen zu können. Dabei wird der Anfang und das Ende eines Datenwortes durch den Zustand SEO angezeigt. Auf diese Weise kann auf das Start- und das Stopbit verzichtet werden. Ausserdem wird es möglich Datenworte ohne vordefinierte Wortlängen zu übertragen.

In Bild 36 ist der zeitliche Ablauf einer Übertragung von zwei Datenworten unterschiedlicher Länge dargestellt. Bei der Übermittlung wird das MSB zuerst gesendet. Wenn die Datenlänge *N* nicht acht Bit beträgt, werden die ersten *N* Bits gesendet und danach die Übertragung abgebrochen. Daraus folgt, dass das empfangene Datenwort nicht mehr mit dem gesendeten Datenwort übereinstimmt, da das gesendete MSB im Schieberegister des Empfängers noch nicht die MSB Position erreicht hat. Aufgrund der Variable „Bitwidth Out“ kann die Information jedoch wieder rekonstruiert werden.

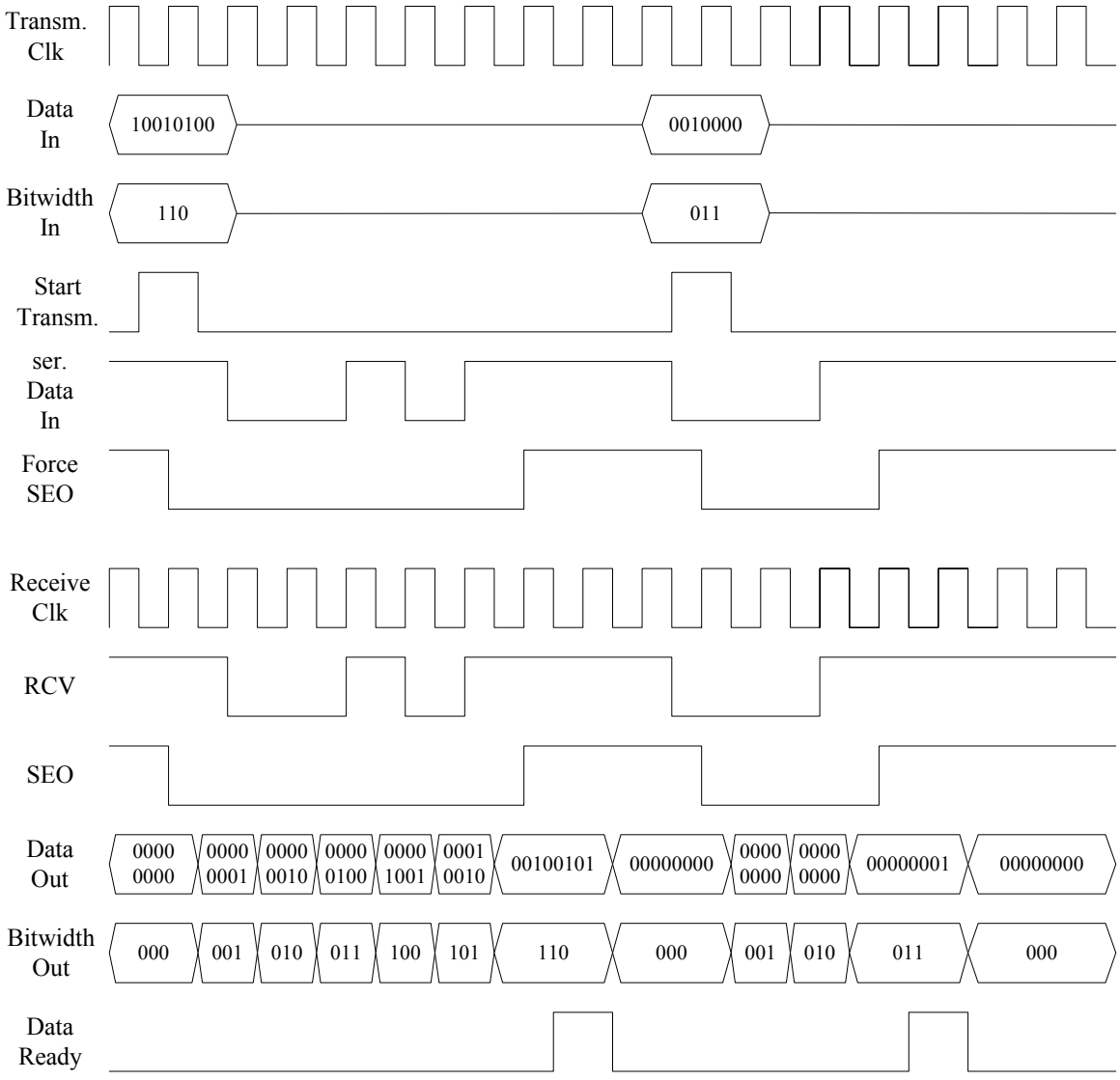


Bild 36: Timingdiagramm der seriellen Übertragung

6.1.3 Aufbau der Sende und Empfangseinheit

Die USB Verbindung arbeitet asynchron, es wird also kein Taktsignal übertragen. Aus diesem Grund muss im Empfänger eine Synchronisation vorgenommen werden. Um diese Synchronisation vornehmen zu können, muss auf der Senderseite und auf der Empfangsseite ein Taktsignal mit der gleichen Frequenz vorhanden sein. Diese Taktfrequenz f_0 sollte höher liegen als die Übertragungsfrequenz $f_Ü$. Beim Start einer Übertragung werden in der Sendeeinheit und in der Empfangseinheit je ein Zähler gestartet. Dieser Start findet im Sender und im Empfänger um maximal eine Taktperiode verschoben statt, die Zähler sind also praktisch synchron. Nun wird in der Sendeeinheit mit jedem *N*-ten Takt das nächste Bit des zu übertragenden Wortes an den Ausgang gelegt. Im Empfänger wird jeweils in der Mitte von *N* Takten das serielle Signal in das Schieberegister eingelesen. Da die Oszillatoren in Sender und Empfänger nie eine exakt gleiche Frequenz erzeugen, können Sende und Empfangseinheit nach einer bestimmten Zeit nicht mehr als synchron betrachtet werden. Mit zunehmender Wortlänge nimmt die Gefahr von Fehlern in der Übertragung zu.
Der prinzipielle Aufbau des Senders und des Empfängers ist in Bild 37 dargestellt. In Tabelle 9 findet sich eine Zusammenstellung der Eigenschaften der realisierten Übertragungsstrecke.

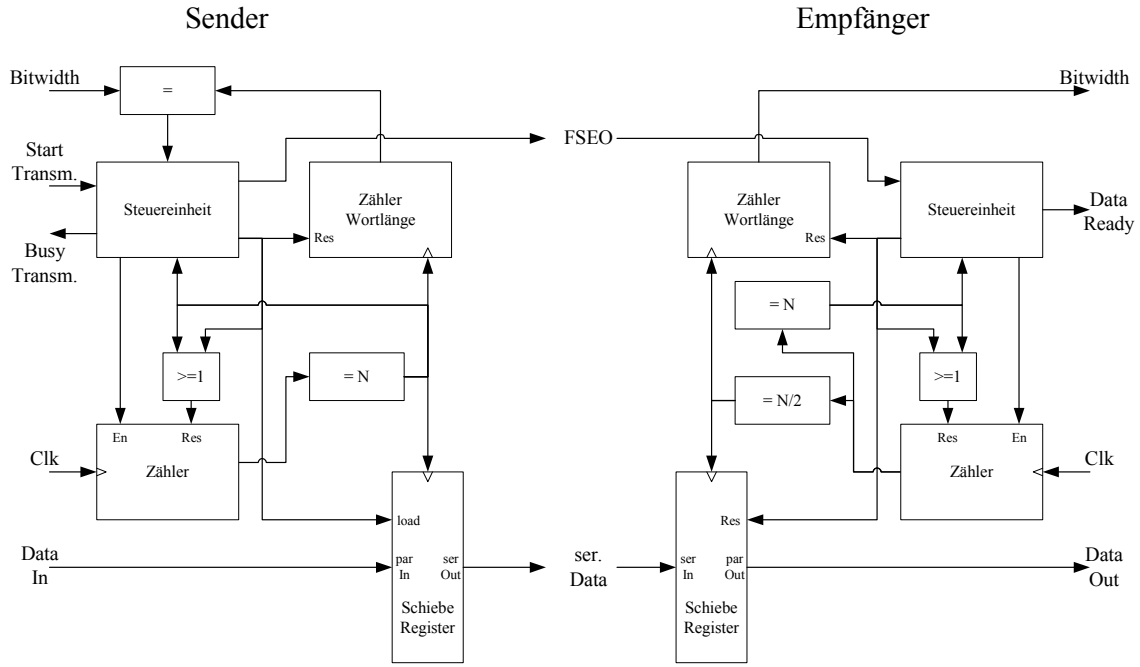


Bild 37: Aufbau der Sende und Empfangseinheit im FPGA

Tabelle 9: Eigenschaften der seriellen Übertragungsstrecke

Merkmal	Wert	Beschreibung
f_0	50 MHz	Sender und Empfänger sind auf eine Taktrate von 50MHz ausgelegt
$f_Ü$	782kHz < < 12MHz	Der Teilerfaktor *N* ist zwischen 4 und 64 einstellbar
Wortlänge	1-8 Bit	Im Register *Bitwidth* kann die zu übertragende Wortlänge eingestellt werden

Um ein Bild übertragen zu können, müssen Sender und Empfänger um eine Steuereinheit erweitert werden. Diese organisiert den Zugriff auf den Bildspeicher. Das ergänzte Übertragungssystem ist in Bild 38 zu sehen. Damit die Bildübertragung funktioniert, müssen die Register *horizontale Grösse* und *vertikale Grösse* auf beiden Seiten den gleichen Wert enthalten. Stimmen diese Register nicht überein, so kann auf der Empfängerseite das Ende der Übertragung nicht korrekt erkannt werden.

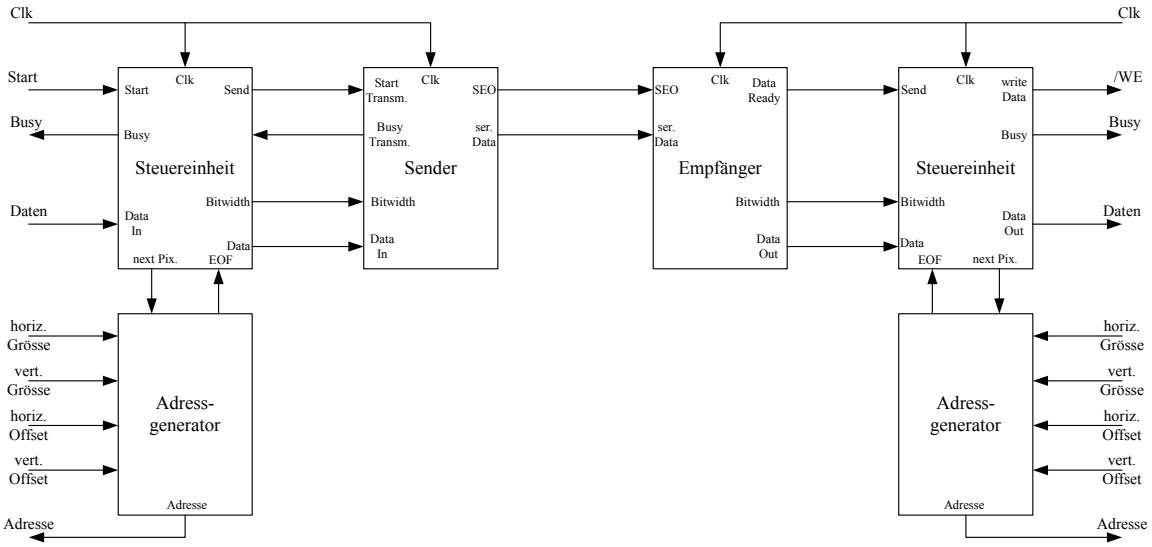


Bild 38: Übersicht über das Bildübertragungssystem

6.2 Entwurf der Entropiecodierung

Der Entwurf der Entropiecodierung stützt sich auf statistische Eigenschaften der Wavelet transformierten Bilder. Dabei wird erwartet, dass die Wavelet transformierten Bilder eine Symbolverteilung haben, bei der die Symbole 255 und 0 am häufigsten auftreten. Die Häufigkeit der Symbole nimmt jeweils in Richtung des Symbols 128 ab und erreicht bei den Symbolen 127 und 128 das Minimum. Eine solche Verteilung ist in Bild 39 dargestellt.

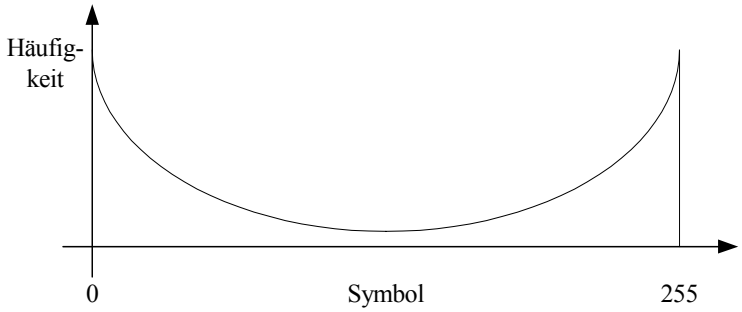


Bild 39: Erwartete Symbolverteilung nach der Wavelet Transformation

Von dieser Verteilung der Symbole ausgehend könnte nun beispielsweise eine Huffman Codierung implementiert werden. Um alle Vorteile der in Kapitel 6.1 vorgestellten Datenübertragung ausnützen zu können, wurde eine Spezialcodierung entworfen.
Diese Codierung bildet alle Symbole auf eine Kombination von zwei Symbolen ab. Jedem Symbol wird ein Wert und eine Wortlänge zugeordnet. Den Symbolen mit der grössten Auftretenswahrscheinlichkeit werden die kleinsten Wortlängen zugeordnet. Den Symbolen mit der kleinsten Auftretenswahrscheinlichkeit hingegen werden die grössten Wortlängen zugeordnet.

Tabelle 10 zeigt einen Teil der Zuordnungsvorschrift des Entropiecoders. Es werden jeweils nur die ersten *N* Bits (MSB zuerst) des Wertes übertragen. Der Vorteil dieser Art Codierung gegenüber einer normalen Huffmancodierung besteht darin, dass im Durchschnitt weniger lange Datenworte entstehen. Dies wird erreicht weil für jede Wortlänge alle möglichen Wörter ein Symbol des Ursprungsalphabets darstellen können. Bei der Huffmancodierung können für jede Wortlänge nur jeweils die Hälfte der möglichen Wörter ein Symbol des Ursprungsalphabets repräsentieren.

Tabelle 10: Ausschnitt aus der Zuordnungsvorschrift der Entropiecodierung

Originalsymbol	Wortlänge *N*	Wert
0	1	00000000
255	1	10000000
1	2	01000000
2	2	00000000
254	2	10000000
253	2	11000000
...	...	...
126	7	01111100
129	7	10000010
127	8	00000000
128	8	10000000

Der Decoder enthält die inverse Abbildungsvorschrift. In Bild 40 ist das Bildübertragungssystem mit Entropiecodierung zu sehen.

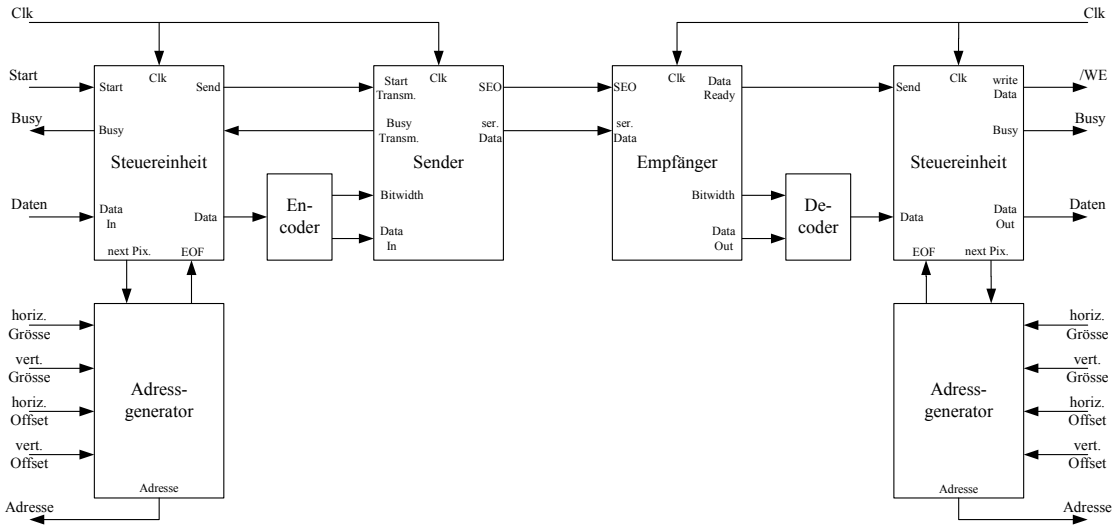


Bild 40: Bildübertragungssystem mit Entropiecodierung

6.3 VHDL Programme für die serielle Bildübertragung

In Tabelle 11 sind die zusätzlichen VHDL Einheiten für die serielle Bildübertragung und Entropiecodierung aufgeführt. Für die Transformation wurden die selben VHDL Programme wie in Tabelle 5 verwendet.

Tabelle 11: VHDL Programme für serielle Bildübertragung

Bezeichnung	Beschreibung
wvt_usb1.vhd	Hauptprogramm mit Wavelet Transformation und USB Sender (Top Level)
iwvt_usb1.vhd	Hauptprogramm mit Wavelet Transformation, inverser Wavelet Transformation und USB Empfänger (Top Level)
transmit_frame.vhd	Organisation der Bildübertragung auf Senderseite: Speicherzugriff, Start und Ende der Übertragung, usw.
receive_frame.vhd	Organisation der Bildübertragung auf Empfängerseite: Speicherzugriff, Start und Ende der Übertragung, usw.
coder.vhd	Entropiecodierung
decoder.vhd	Entropiedecodierung
usb_transmitter.vhd	Sendeeinheit für ein Datenwort
usb_receiver.vhd	Empfangseinheit für ein Datenwort
usb_test.vhd	Testumgebung für Transmitter und Receiver
test_coder.vhd	Testumgebung für Codierung und Decodierung
usb_test_all.vhd	Testumgebung für gesamte Übertragungsstrecke mit transmit_frame und receive_frame

6.4 Test der Übertragungsstrecke

Um die Bildübertragung zu testen wurden bei verschiedenen Übertragungsgeschwindigkeiten Wavelet transformierte Bilder übermittelt, auf der Empfängerseite rücktransformiert und das Resultat auf Artefakte untersucht. Durch die Rücktransformation werden Übertragungsfehler besser sichtbar. Die Bildübertragung funktioniert bis zu einer Übertragungsgeschwindigkeit von 10Mbps problemlos. Bei der maximalen Übertragungsgeschwindigkeit von 12Mbps entstehen teilweise Fehler im Bild. Der USB Treiber ist für maximal 10Mbps spezifiziert. Wird mit 12Mbps gearbeitet, so liegt die Übertragungsrate 20% über dem spezifizierten Wert. Der Grund für die fehlerhafte Übertragung kann darin liegen, dass die Spezifikationen nicht eingehalten werden.

6.5 Test des Entropiecoders

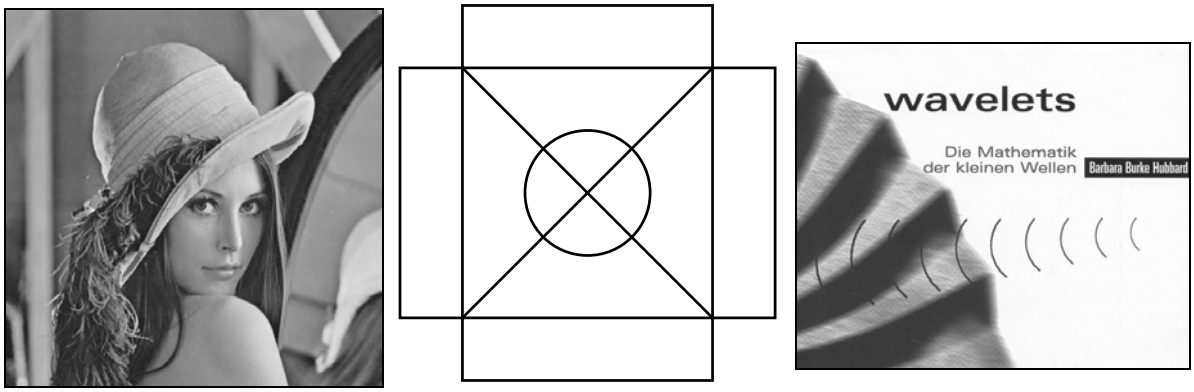
Codierung und Decodierung wurden in einer VHDL Testumgebung auf ihre Funktionalität getestet. Danach wurde das korrekte Funktionieren des Encoders und Decoders anhand von Testbildern auf dem Board getestet. Hier wurde wieder ein transformiertes Bild übermittelt und dann auf der Empfängerseite zurücktransformiert. Das rücktransformierte Bild wurde anschliessend auf Artefakte untersucht. Bei Übertragungsraten von bis 10Mbps traten keine Fehler auf.

6.5.1 Untersuchung der Kompressionsrate

Um eine Aussage über die erreichbare Kompressionsrate zu erhalten, wurde die Übertragungsdauer verschiedener Testbilder mit und ohne Entropiecodierung gemessen. In Tabelle 12 sind die Resultate dieser Messungen dargestellt. In Bild 41 finden sich die zugehörigen Bilder. Als Beurteilungskriterien wurden die Kompressions- und Bitrate berechnet. Die Kompressionsrate errechnet sich aus dem Verhältnis der Übertragungszeiten. Die Bitrate gibt an, wie viele Bits im Durchschnitt zur Darstellung eines Bildpunktes benötigt werden. Da bei der Übertragung ohne Codierung die Bitrate konstant acht Bit beträgt, lässt sich die Bitrate direkt aus der Kompressionsrate berechnen.

Kompressionsrate: $$CR = \frac{t_{\ddot{U}0}}{t_{\ddot{U}C}} \qquad (6.1)$$

Bitrate: $$BR = \frac{b}{CR} = \frac{8\text{Bit}}{CR} \qquad (6.2)$$



Lena Rechtecke Buchdeckel

Bild 41: Testbilder für Untersuchung der Kompressionsrate

Tabelle 12: Kompressionsrate des verwendeten Entropiecoders

Bild	Übertragungsdauer ohne Codierung ($t_{Ü0}$)	Übertragungsdauer mit Codierung ($t_{ÜC}$)	Kompressions-Rate CR	Bitrate* BR
Lena (1 Level)	234.5 ms	121.2 ms	1.94 : 1	4.12
Lena (4 Level)	234.5 ms	98.7 ms	2.38 : 1	3.361
Rechtecke (1 Level)	234.5 ms	120.1 ms	1.95 : 1	4.1
Rechtecke (4 Level)	234.5 ms	91.2 ms	2.57 : 1	3.11
Buchdeckel (1 Level)	234.5 ms	127.9 ms	1.83 : 1	4.36
Buchdeckel (4 Level)	234.5 ms	104.5 ms	2.24 : 1	3.57

* Da für die Übertragung ein zusätzliches Signal (SEO) benutzt wurde, kann die Bitrate nicht direkt mit der Bitrate verglichen werden, welche normalerweise in Untersuchungen für Kompressionen verwendet wird.

Zum Vergleich sei hier nochmals die typische Kompressionsrate des Kompressionscodecs ADV601 von Analog Devices [ANA96] angegeben. Dieser Codec erreicht eine typische Kompressionsrate von 4:1 bei Übertragungen ohne sichtbaren Qualitätsverlust. Im ADV601 wird neben der Huffmancodierung zusätzlich eine Quantisierung und eine Run Length Codierung verwendet.
Da diese zusätzlichen Elemente in der hier verwendeten Kompression nicht angewendet wurden, ist die erreichte Kompressionsrate von 2.4:1 für verlustfreie Kompression als gut zu bewerten.

Werden die Bilder anstatt über einen Level über vier Levels Wavelet transformiert, so steigt die erreichte Kompressionsrate im Durchschnitt um 26% an. Der Rechenaufwand hingegen steigt für eine Transformation über vier Levels um 33% gegenüber einer Transformation über einen Level. Da das Verhältnis zwischen Aufwand und Ertrag mit zunehmender Transformationstiefe schlechter wird, lohnt es sich wahrscheinlich nicht, die Wavelet Transformation über alle möglichen Levels durchzuführen.

7 Ausblick

7.1 Kompression von Standbildern

Diese Diplomarbeit hat gezeigt, dass es möglich ist ein Wavelet basiertes Bildkompressionssystem mit vertretbarem Aufwand in einem FPGA zu realisieren. Dabei wurde das Gewicht hauptsächlich auf einen schnellen Transformationskern gelegt. In einem weiteren Schritt sollte nun versucht werden die Speicherzugriffe auf den externen Speicher zu optimieren. Damit könnte die Berechnungszeit für die Transformation eines Bildes erheblich gesenkt werden.

Die implementierte Bildkompression verwendet keine Präcodierung. In dieser Arbeit wurden jedoch zwei verschiedene Präcodierungsalgorithmen vorgestellt. Es wäre interessant einen dieser Algorithmen in einem FPGA zu implementieren. Die erreichbare Kompressionsrate könnte damit markant gesteigert werden.

Am Schluss der Diplomarbeit wurden die VHDL Programme noch einmal überarbeitet. Diese Überarbeitung hatte zum Ziel, die Wortbreite der Transformation generisch einstellen zu können. Durch umstellen der Bitbreite von acht Bit auf 16 Bit kann so eine Integer to Integer Transformation mit erhöhter Dynamik realisiert werden. Dies funktioniert allerdings nur für Bilder mit acht Bit Wortbreite.
Aus Mangel an Zeit konnten die Programme nicht mehr vollständig ausgetestet werden. Für eine Wortbreite von acht Bit wurden die Tests erfolgreich abgeschlossen. Bei grösseren Wortbreiten entstehen jedoch Artefakte im Bild. Als nächster Schritt sollte mit einer Post-Route Simulation eruiert werden, welcher Verarbeitungsblock für die Artefakte verantwortlich ist.
In Tabelle 13 sind die geänderten Programme aufgelistet.

Tabelle 13: VHDL Programme mit generisch konfigurierbarer Wortbreite

Bezeichnung	Beschreibung
wt_ud_1_block.vhd	Hauptprogramm für Wavelet Transformation und inverse Wavelet Transformation
wt_1down.vhd	Transformationseinheit für die Wavelet Transformation mit Haupt Statemachine, Zählern, Generierung der Steuersignale für das externe RAM usw.
wt_1up.vhd	Transformationseinheit für die inverse Wavelet Transformation
cd53_down.vhd	5-3 Transformationscore mit Arithmetik
cd53_up.vhd	Inverser 5-3 Transformationscore mit Arithmetik
ramBlock1024kFlex.vhd	Interner RAM Block mit flexibler Wortbreite

7.2 Kompression von Videostream

In der vorliegenden Diplomarbeit wurden nur Standbilder verarbeitet. Die erreichbare Transformationsgeschwindigkeit der FPGA Wavelet Transformation prädestinieren diese jedoch für den Einsatz in der Videosignalverarbeitung.

Im folgenden wird ein Konzept für eine FPGA basierte Videostreamverarbeitung vorgestellt. Als erstes wird skizziert, wie eine ideale Hardware für Videostreamverarbeitung aufgebaut sein müsste. Danach wird dieses Konzept auf das XSV-300 Entwicklungsboard angepasst.
Das vorgestellte Konzept enthält keine Kompression aufeinanderfolgender Bilder. Bei Videostream weisen zeitlich nahe beieinander liegende Bilder eine sehr hohe Redundanz auf. Aus diesem Grund kann durch eine Kompression der aufeinanderfolgenden Bilder die Kompressionsrate stark erhöht werden. Es wäre deshalb sinnvoll, das hier vorgestellte Konzept so zu erweitern, dass bei der Kompression die Redundanz der aufeinanderfolgenden Bilder berücksichtigt wird.

7.2.1 Ideale Hardware für Videostreamverarbeitung

Die Hardware wird in eine Kompressions- und eine Dekompressionseinheit unterteilt. Beide Einheiten werden getrennt besprochen. Kernstück für Kompressions- und Dekompressionseinheit bildet ein FPGA. Dieser wird für die Realisierung sämtlicher Signalverarbeitungsblöcke der Videokompression und Dekompression eingesetzt. Je nach Signalquelle und Signalsenke müssen entsprechende Peripheriebausteine verwendet werden. Um die Bilder während der Verarbeitung zu speichern wird SRAM eingesetzt. Die Wavelet Transformation wird zuerst in allen horizontalen Transformationsebenen und danach in allen vertikalen Transformationsebenen (hhvv Transformation) durchgeführt.

Kompressionseinheit:
Die in Bild 42 gezeigte Kompressionseinheit ermöglicht eine Präcodierung, bei der mehrmals auf das Bild zugegriffen wird. Dadurch wird es möglich Präcodierungen wie die Quadtree Codierung zu realisieren.
Wird eine Präcodierung verwendet, die direkt den Ausgang der vertikalen Transformation verarbeiten kann, so entfällt der Speicherblock C. Die vertikal transformierten Bilddaten werden dann nicht mehr in Speicherblock B zurückgeschrieben, sondern direkt an die Quantisierung gesendet. Mit jedem neuen Bild werden die Speicherblöcke um eine Stelle rotiert. Bei diesem Konzept wird vorausgesetzt, dass die Daten in den einzelnen Verarbeitungsblöcken mit der Taktrate des Videostreams verarbeitet werden.

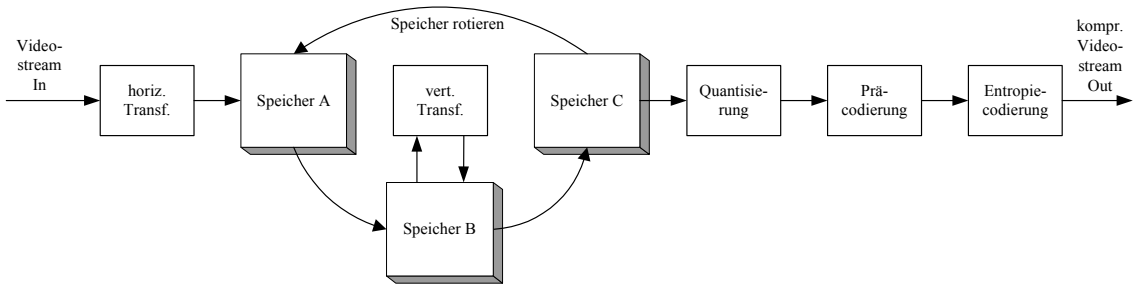


Bild 42: Kompressionseinheit für Videostream

Dieses Konzept erfordert genügend Speicher um drei Bilder ablegen zu können. Damit gleichzeitig auf alle drei Speicherblöcke zugegriffen werden kann, müssen Adress-, Daten-, und Steuerbus jedes Speicherblocks separat vorhanden sein.

Dekompressionseinheit:
In Bild 43 ist die Struktur der Dekompressionseinheit dargestellt. Da am Videoausgang ein kontinuierlicher Datenstrom erzeugt werden muss, ist ein zusätzlicher, vierter Speicherblock nötig. Dieser Speicherblock entfällt, wenn es gelingt die horizontale Transformation mit dem Videoausgang zu synchronisieren. Der Speicherblock A entfällt, wenn die vertikale Transformation direkt an die Dequantisierung angeschlossen werden kann.
Auch in der Dekompressionseinheit müssen alle Speicherblöcke unabhängig ansprechbar sein.

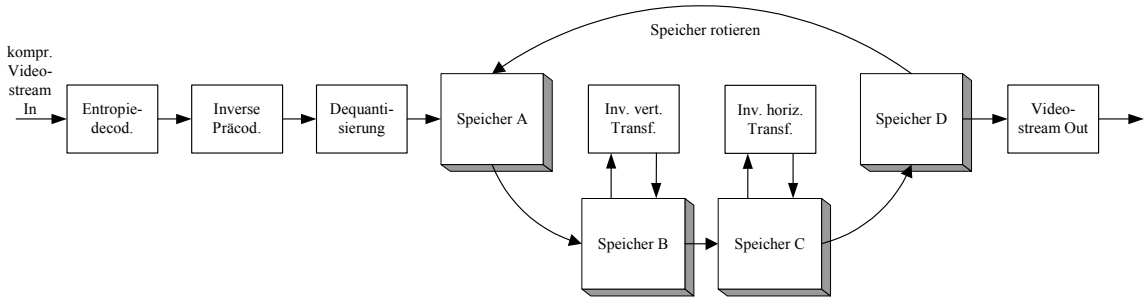


Bild 43: Dekompressionseinheit für Videostream

Das in Bild 42 und Bild 43 vorgestellte Konzept hat den Nachteil, dass es sehr viel Speicher benötigt. Wird das Bild partitioniert, so kann man mit wesentlich weniger Speicher auskommen. Bei einer Embedded Zerotree Codierung sollte das Bild ohnehin partitioniert werden. Aus diesem Grund wird hier eine zweite Variante für Videostreamverarbeitung gezeigt.

Kompressionseinheit für partitionierte Verarbeitung:
Bild 44 zeigt das Konzept einer partitionierten Videokompression. Wird das Bild in Blöcke mit Höhe *N* aufgeteilt, so müssen die beiden Speicherblöcke A und B *N* Zeilen des Bildes aufnehmen können. Wiederum müssen beide Speicherblöcke unabhängig ansprechbar sein.

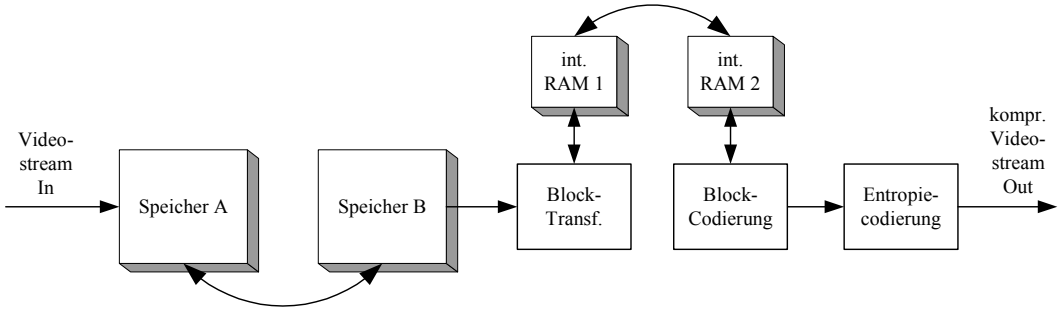


Bild 44: Kompressionseinheit für partitionierte Verarbeitung

Dekompressionseinheit für partitionierte Verarbeitung:
Für die Dekompression werden die Signalverarbeitungsblöcke der Kompressionseinheit in der umgekehrten Reihenfolge angeordnet. Um am Videoausgang einen kontinuierlichen Datenstrom zu erhalten, muss in den Speicherblöcken A und B jeweils ein vollständiges Bild abgelegt werden können.

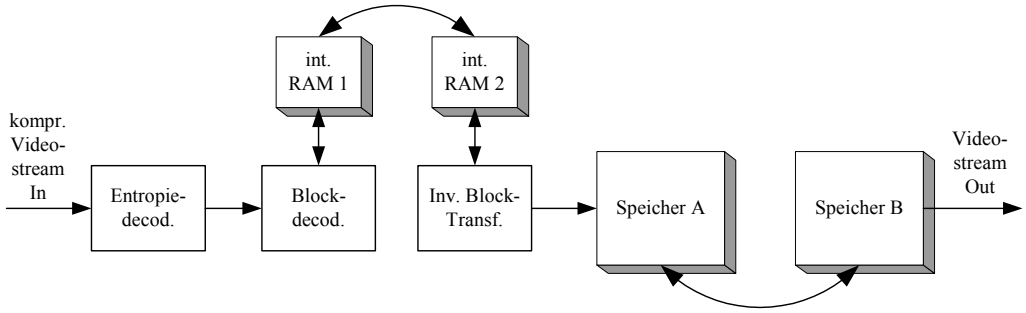


Bild 45: Dekompressionseinheit für partitionierte Verarbeitung

Ausgehend von einem acht Bit schwarzweiss Bild der Grösse 640x480 ist in Tabelle 14 der Speicherbedarf der verschiedenen Varianten aufgelistet.

Tabelle 14: Speicherbedarf der verschiedenen Videostream Konzepte

Konzept	Speicherbedarf Kompression	Speicherbedarf Dekompression
Gesamtes Bild	3 x 512kB= 1536kB	4 x 512kB = 2048kB
Partitioniertes Bild (N=32)	2 x 32kB = 64kB	2 x 512kB = 1024kB

7.2.2 Konzept für Videostreamverarbeitung mit dem XSV-300 Entwicklungsboard

Das XSV-300 Entwicklungsboard weist genügend Speicherplatz für vier schwarzweiss Bilder auf. Es können jedoch nur zwei unabhängige Speicherblöcke angesprochen werden. Zudem wird der eine Datenbus mit dem Videoausgang geteilt. Aus diesen Gründen ist es nicht möglich auf dem XSV Board das Konzept der nicht partitionierten Verarbeitung aus Kapitel 7.2.1 zu realisieren. Wird eine partitionierte Verarbeitung gewählt, so kann diese realisiert werden.
In diesem Kapitel soll das Konzept für eine nicht partitionierte Verarbeitung so angepasst werden, dass es sich auf dem XSV Board implementieren lässt.

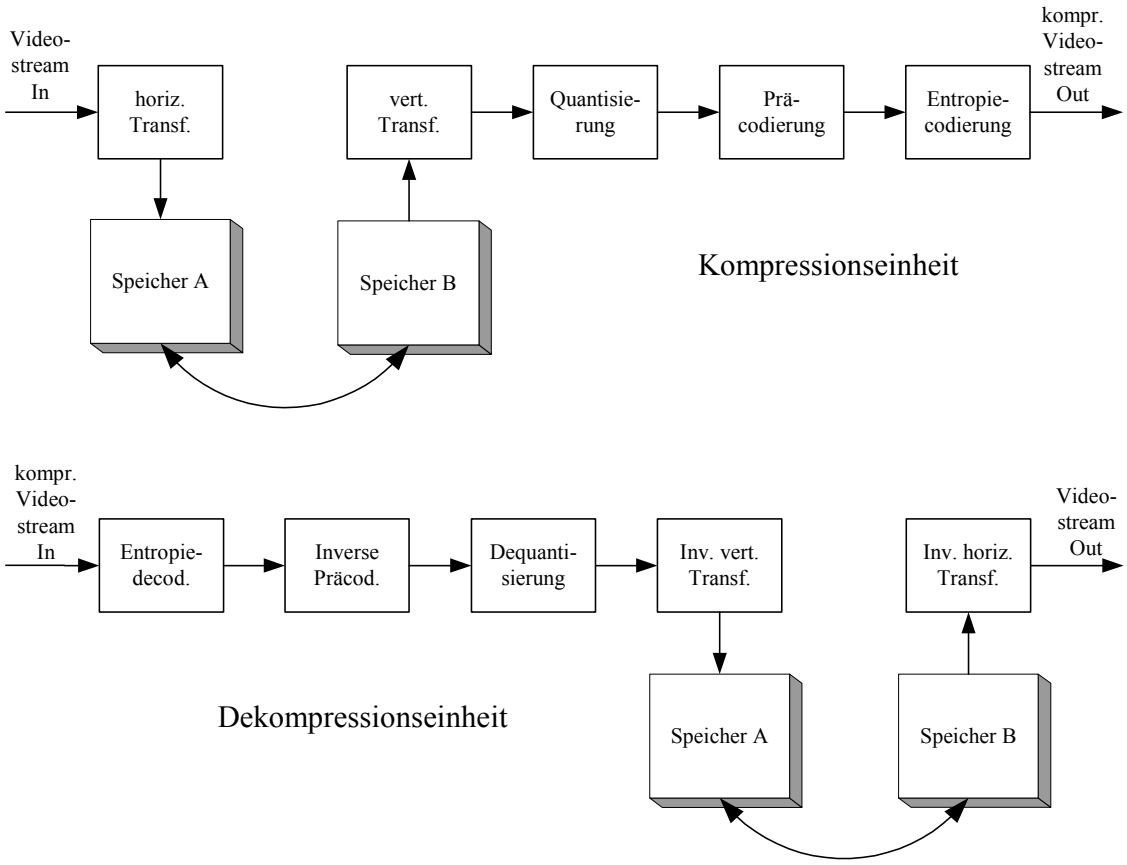


Bild 46: Konzept für Videostreamverarbeitung mit dem XSV Board

Bild 46 zeigt das Konzept einer Videostreamverarbeitung mit dem XSV-300 Entwicklungsboard. Bei diesem Konzept muss eine Präcodierung verwendet werden, welche Spalte für Spalte bearbeitet. Dafür könnte beispielsweise eine Lauflängencodierung eingesetzt werden. Ein problematischer Punkt könnte die Synchronisation der inversen, horizontalen Transformation mit dem VGA Ausgang darstellen.
Gelingt diese Synchronisation nicht, so existiert folgende Alternative:
Es wird nur jedes zweite Bild übertragen. Der VGA-Ausgang stellt alternierend nur jede zweite Zeile dar. So entsteht zwischen zwei Zeilen genügend Zeit, um eine Zeile aus Speicherblock A zu lesen, diese horizontal zu transformieren und dann in Speicherblock B abzulegen.

8 Zusammenfassung

Die vorliegende Diplomarbeit beschäftigt sich mit der Hardware-Implementierung der Wavelet Transformation für Bildübertragung. Dabei stellt die Wavelet Transformation ein Glied in der Signalverarbeitungskette der Bildkompression dar. Aufgrund der grossen Datenmengen, die in der Bildverarbeitung anfallen, erfordern leistungsfähige Wavelet Transformationen den Einsatz dezidierter Hardware.

Es wurden verschiedene Algorithmen auf Ihre Eignung zur Implementierung in einem FPGA untersucht. Dabei wurde, auf dem Lifting Schema basierend, der Schwerpunkt auf unterschiedliche Varianten der Integer to Integer Wavelet Transformation gelegt. Weiter wurden zwei verschiedene Möglichkeiten der 2D Wavelet Transformation aufgezeigt. Die Vor- und Nachteile für die Hardware-Implementierung wurden für beide Varianten beurteilt.

Am Beispiel des 5-3 Wavelets wurde ein Konzept für die Implementierung einer verlustfreien Wavelet Transformation in einem FPGA vorgestellt. Durch Anwendung des Lifting Schemas kann diese Transformation mit einer konstanten Dynamik durchgeführt werden.
Das Konzept der Hardware-Implementierung wurde in einem Xilinx FPGA umgesetzt und erfolgreich ausgetestet. Der realisierte Transformationscore lässt sich bis zu einer Taktrate von 32MHz einsetzen. Um die gleiche Performance zu erreichen, müsste ein konventioneller DSP mit mindestens 300MHz betrieben werden. Durch Kaskadierung der Transformationscores konnte die Berechnungszeit für die Transformation eines Bildes um 30% gesenkt werden.

Basierend auf dem physikalischen Layer einer USB Schnittstelle wurde eine verlustfreie Bildübertragungsstrecke aufgebaut. Die Wavelet transformierten Bilder konnten durch den Einsatz einer Entropiecodierung mit Kompressionsraten in der Grössenordnung 2.5 : 1 komprimiert werden.

Die Möglichkeiten verlustfreier und verlustbehafteter Bildkompressionen wurden analysiert und auf ihre Eignung zur Implementierung in einem FPGA untersucht. Es konnte gezeigt werden, dass sich die Integer to Integer Wavelet Transformation mit konstanter Dynamik nicht für die Realisierung einer verlustbehafteten Bildkompression eignet. Als Alternative zu dieser Transformation bietet sich die Integer to Integer Wavelet Transformation mit erhöhter Dynamik an.

Abschliessend wurde ein Konzept für die Implementierung einer Videokompression in einem FPGA entwickelt. Basierend auf dieser Diplomarbeit könnte in einer weiteren Phase eine solche Kompression für Videodaten realisiert werden.

9 Abbildungsverzeichnis

10 Tabellenverzeichnis

11 Glossar

Codec	Kodierungs- / Dekodierungseinheit
Core	In einer Hardware-Beschreibungssprache (z.B. VHDL) beschriebener digitaler Verarbeitungsblock.
CPLD	Complex Programmable Logic Device
CVBS	Color, Video, Blanking, and Sync. Europäische Bezeichnung für Videosignal mit Synchronisations, Luminanz und Chrominanz Anteil.
DSP	Digitaler Signal Prozessor
EZW	Embedded Zerotree Wavelet Coding
Fixpoint	Zahlenformat auf digitalen Rechnern. Der Kommapunkt ist dabei an einer fixen Stelle.
Floatingpoint	Zahlenformat auf digitalen Rechnern. Die Zahl wird aus einem Exponent, einer Mantisse und einem Vorzeichen gebildet. Dadurch wird eine hohe Dynamik erreicht.
FPGA	Field Programmable Gate Array
Integer	Zahlenformat auf digitalen Rechnern. Es können nur ganze Zahlen dargestellt werden.
JPEG2000	Kompressionsstandard für Bildkompression. Verwendet eine Wavelet Transformation.
Lifting Schema	Berechnungsschema für die Wavelet Transformation.
Post-Route Simulation	Simulation für konfigurierbare digitale Logik. Es wird das Zeitverhalten des verwendeten Bauteils berücksichtigt.
Pre-Route Simulation	Simulation für konfigurierbare digitale Logik. Das zeitliche Verhalten des verwendeten Bauteils wird nicht berücksichtigt.
RAMDAC	RAM Digital-Analog Converter. Die analogen Ausgangswerte werden aufgrund einer im internen RAM des Konverters abgelegten Tabelle erzeugt.
Run-Length Codierung	Lauflängenkodierung
USB	Universal Serial Bus. Für PC entwickeltes Serielles Bussystem.
VHDL	VHSIC Hardware Description Language. Beschreibungssprache für digitale Logik. Neben Verilog am meisten verbreitet.
VHSIC	Very High Speed Integrated Circuit
YCrCb	Videosignal mit Farbaufteilung in Luminanz (Grauwert) und zwei Chrominanz (Farbe) Anteilen.

12 Literatur

[ANA96]	N.N.	ADV601 Low Cost Multiformat Video Codec. ADV601 Preliminary Data Sheet, Analog Devices, 1996
[BA01]	Bachofen D. und Rietmann C.	Inbetriebnahme eines XSV Entwicklungsboards. unveröffentlichte Semesterarbeit, Hochschule für Technik Wirtschaft und soziale Arbeit, St.Gallen, 2001
[BRO00]	Brown, S. D. und Vranesic, Z.	Fundamentals of digital logic with VHDL design. Boston, McGraw-Hill, 2000
[BUR96]	Burke-Hubbard, B.	The world according to wavelets – The story of a mathematical technique in the making. Wellesley, MA, U.S.A., A.K. Peters Ltd, 1996
[CAL1]	Calderbank, A. R., Daubechies, I., Sweldens, W. und Yeo, B.-L.	Lossless Image Compression Using Integer To Integer Wavelet Transforms. Erhältlich unter url: citeseer.nj.nec.com/91145.html
[CAL98]	Calderbank, A. R., Daubechies, I., Sweldens, W. und Yeo, B.-L.	Wavelet transforms that map integers to integers. Applied and Computational Harmonic Analysis, vol. 5, no. 3, pages 332-369, July 1998.
[CHA96]	Chao, H., Fisher, P. und Hua, Z.	An approach to integer wavelet transformation for lossless image compression. Technical Paper, University of North Denton, Texas, U.S.A., 1996
[DAU92]	Daubechies, I.	Ten lectures on wavelets. CBMS-NSF Regional Conf. Series in Appl. Math., Vol.61, Society for Industrial and Applied Mathematics, Philadelphia, PA, 1992
[GES00]	Geser, M.	Bildkompression mit Wavelets. unveröffentlichte Diplomarbeit, Hochschule für Technik Wirtschaft und soziale Arbeit St. Gallen, 2000
[KOL99]	Kolarov, K. und Lynch, W.	Very low cost video wavelet codec. In: SPIE Conference on Applications of Digital Image Processing, Vol. 3808, Denver, 1999
[LEW91]	Lewis A. und Knowles G.	A 64Kb/s Video Codec using the 2-d Wavelet Transform. In: Proceedings of the Data Compression Conference, Snowbird, Utah, 1991
[MAR00]	Marcellin, M. W. et al.	An overview of JPEG-2000. In: Data Compression Conference (2000), pages 523-544, 2000
[MEY00]	Meyer-Bäse, U.	Schnelle digitale Signalverarbeitung – Algorithmen, Architekturen, Anwendungen. Berlin, Springer, 2000

[POY96]	Poynton C. A.	A Technical Introduction to Digital Video. Chichester UK, John Wiley & Sons – Verlag, 1996
[REI1]	Reichel, J. et al.	Integer Wavelet Transform for Embedded Lossy to Lossless Image Compression. Signal Processing Laboratory, Swiss Federal Institute of Technology, Lausanne, Switzerland, Erhältlich unter url: citeseer.nj.com/420073.html
[REZ99]	Reza, A. M.	From Fourier Transform to Wavelet Transform – Basic Concepts. White Paper, Xilinx Corp., 1999 Erhältlich unter url: http://www.xilinx.com/products/logicore/dsp/fft_to_wavelet.pdf
[RIT01]	Ritter, J. und Molitor, P.	A pipelined architecture for partitioned DWT based lossy image compression using FPGA's. Seminarunterlage 2001, FPGA 2001, Februar 11-13, 2001, Monterey, CA
[RUS99]	Rushton, A.	VHDL for Logic Synthesis. 2nd ed.. Chichester UK, John Wiley & Sons – Verlag, 1999
[SAH97]	Sahni, S. et al.	State of The Art Lossless Image Compression Algorithms. 1997, Erhältlich unter: http://softlab.od.ua/algo/dsp/index_image.html
[SAI96]	Said, A. und Pearlman, W. A.	A new, fast, and efficient image codec based on set partitioning in hierarchical trees. In: Trans. Circuits and Systems for Video Technology, Vol. 6, IEEE, June 1996
[STRU00]	Strutz, T.	Bilddatenkompression – Grundlagen, Codierung, MPEG, JPEG. 1. Aufl., Braunschweig, Vieweg-Verlag, 2000
[SUT99]	Sutter, S.	Vergleich von FPGA-Architekturen für partitionierte Wavelet-Transformationen auf Bildern. unveröffentlichte Diplomarbeit, Martin-Luther-Universität Halle-Wittenberg, Halle, 1999
[SWE95]	Sweldens, W.	The lifting scheme: A New Philosophy in Biorthogonal Wavelet Constructions. In: SPIE, Wavelet Applications in Signal and Image Processing III (1995), S. 68-79, 1995
[SWE96]	Sweldens, W.	The lifting scheme: A custom-design construction of biorthogonal wavelets. In: Applied and Computational Harmonic Analysis, volume 3, no.2, pages 186-200, 1996
[SHAP93]	Shapiro, J. M.	Embedded image coding using zerotrees of wavelet coefficients. In: IEEE Trans. Signal Processing, vol. 41. S.3445-3462, 1993
[XES00]	N.N.	XSV Board V1.0 Manual. XESS Corp., Apex NC 27502, U.S.A., 2000
[XIL00]	N.N.	Xilinx Synthesis Technology (XST) User Guide. Vers. 3.1i, Xilinx Corp., San Jose, U.S.A., 2000

13 Ehrenwörtliche Versicherung

Ich versichere hiermit ehrenwörtlich, dass ich die vorliegende Arbeit selbständig und nur unter Benützung der angeführten Quellen und Hilfsmittel angefertigt habe. Sämtliche Entlehnungen sind durch Quellenangaben kenntlich gemacht.

Ort, Datum: Arbon, 2. Oktober 2001

Unterschrift: Daniel Bachofen

Anhang A Eingesetzte Mittel und Systeme

Video Kamera
JVC
Videomovie GR- AX40
VHS C

FPGA Entwicklungsboard
XESS
XSV-300
Board mit Xilinx Virtex FPGA 300kGates
XESS Tools V3.3

Xilinx Entwicklungsumgebung
Xilinx Foundation ISE3.3.06i
ModelSimXE 5.3d
HDL Bencher 1.02x Xilinx Edition

Messgeräte
Logic Analyzer:
PM 3585 Philips
200MHz

Digitales Speicheroszilloskop:
LeCroy
WaveRunner LT264M
350MHz - 1GS/s

Computer
Notebook
Compaq Armada E500
Intel PIII Prozessor
196 MByte RAM

Betriebssystem: Microsoft Windows 2000

Weitere Software
Für die Dokumentation wurde folgende Software benützt:

- Microsoft Word 2000
- Microsoft Project 2000
- Microsoft Visio 2000
- Matlab 6.2
- Microsoft Visual C++ 6.0

Die Wavelet Transformierten Bilder für die Dokumentation wurden mit einem in C selbstgeschriebenen Programm berechnet. Für die Visualisierung dieser Bilder wurde Matlab benutzt.

Anhang B Einführende Literatur

Liste der Veröffentlichungen:

1. From Fourier Transform to Wavelet Transform – Basic Concepts
Reza, A. M.

2. The lifting scheme: A New Philosophy in Biorthogonal Wavelet Constructions
Sweldens, W.

3. An approach to integer wavelet transformation for lossless image compression
Chao, H. , Fisher, P. und Hua, Z.

4. A new, fast, and efficient image codec based on set partitioning in hierarchical trees
Said, A. und Pearlman, W. A.