MPC

MULTI PROJEKT CHIP GRUPPE
BADEN-WÜRTTEMBERG

Herausgeber: Hochschule Ulm **Ausgabe:** 41 **ISSN** 1862-7102 **Workshop:** Künzelsau Februar 2009

Cooperating Organisation
Solid-State Circuit Society Chapter
IEEE German Section

**Tagungsband zum Workshop der Multiprojekt-Chip-Gruppe Baden-Württemberg,
Künzelsau, 6. Februar 2009**

Inhaltsverzeichnis

Diesen Workshopband und alle bisherigen Bände finden Sie im Internet unter:
http://www.mpc.belwue.de

Structured ASICs für Mixed-Signal Anwendungen

Dr. Christian Burwick, Ir. Cor Scherjon
Institut für Mikroelektronik Stuttgart, Allmandring 30a, 70569 Stuttgart
Telefon 0 711 / 21 855 -10, Fax -100, E-Mail info@ims-chips.de

Viele, am Institut für Mikroelektronik Stuttgart erfolgreich entwickelte Kunden-Schaltungen basieren maßgeblich auf dem GATE FOREST – Konzept. Dieser Gate-Array Ansatz wurde erstmals 1988 mit einer Strukturgröße von 2 µm eingeführt. Mit dem Übergang auf 1,2 µm und später dann auf 0,8 µm nahm die Komplexität entsprechend zu. Mit Einführung der 0,8 µm-Technologie wurden die Master um einen entsprechenden Analogteil ergänzt und zum kundenspezfischen Mixed-Signal Gate-Array erweitert. Gegenwärtig erfolgt die Umsetzung und Einfühung einer 0,5 µm-Technologie, einhergehend mit einer deutlichen Erweiterung des Analogteils sowie einigen Spezialfunktionen wie zum Beispiel dediziertes SRAM. Im Folgenden werden die Besonderheiten der neuen GATE FOREST-Technologie sowie ein Verschlüs-selungs-ASIC als Anwendungsbeispiel vorgestellt.

1. GATE FOREST

Die IMS GATE FOREST Technologie ist eine Gate-Array beziehungsweise Sea-of-Gates Architektur [1-2]. Der Aufbau ist schematisch in Abbildung 1 dargestellt. Die Grundzelle des Digitalteils besteht aus je zwei NMOS und PMOS-Transistoren mit vorgegebener Transistorlänge und Weite. Abhängig von der Größe des Masters umfasst das Array entsprechend viele Grundzellen. Ergänzt wird der Digitalteil durch einen Analogteil wie in Abbildung 2 gezeigt. Der obere Teil der Zelle umfasst Widerstandsbereiche mit einem Schichtwiderstand von 10 Ω/□ und 80 Ω/□. Im mittleren Bereich befinden sich N-Kanal und P-Kanal-Transistoren mit unterschiedlichen W/L-Verhältnissen. Der untere Teil schließlich umfasst 16 Einheitskapazitäten mit je 90 fF. Pro Grundzelle lassen sich damit Widerstände bis etwa 70 kΩ und Kapazitäten bis zu 1.44 pF realisieren. Die Realisierung von RAM im digitalen Bereich ist extrem flächenhungrig. Einige der Gate-Arrays enthalten daher dediziertes RAM, so zum Beispiel 12 Blöcke je 64x8 Bit auf dem GFQ060.

Mixed-Signal-Schaltungen können elegant in einer Hardwarebeschreibungssprache, vorzugsweise VHDL, beschrieben werden. Die anschließende Synthese behandelt analoge Schaltungsteile als erweiterte Black Boxe. Die analoge Schaltungsbibliothek kann bei Bedarf um weitere Funktionen erweitert werden. Nach einer abschließenden Gesamtsimulation erfolgt dann die Datenaufbereitung und Fertigung der Schaltung.

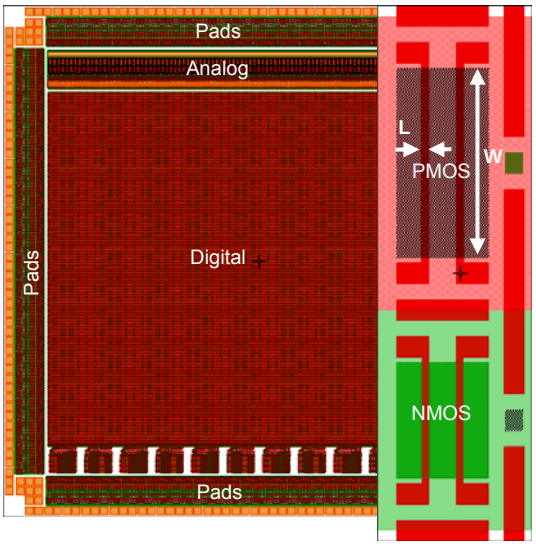


Abbildung 1: Aufbau des Gate-Arrays, Grundzelle des Digitalteils (rechts)

Typische Durchlaufzeiten für Prototypen ab Netzliste liegen bei etwa 13 Wochen entsprechend Abbildung

3. Bei Serienprodukten wird diese Zeit durch externes Verpacken um weitere 5 Wochen verlängert.

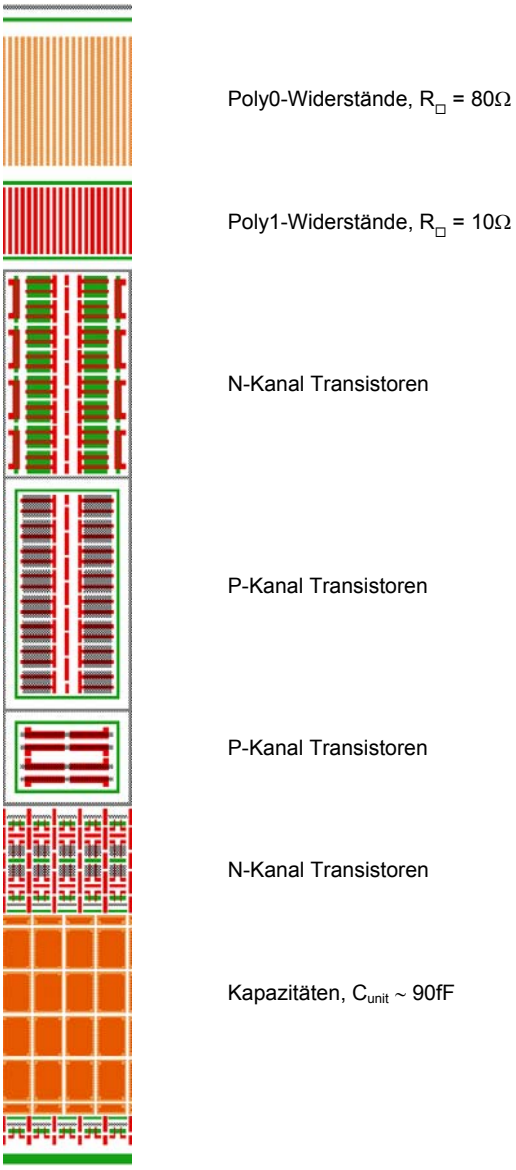


Abbildung 2: Aufbau einer Analogzelle

Die Kosten variieren je nach Mastergröße und Einstieg in den Designflow. Die Entwicklungskosten betragen bei einem Einstieg ab Netzliste 9.990 €. Für die MPC-Gruppe existiert ein Sonderpreis von 3.500 € für Multiprojekt-Wafer. Für verschiedene Schaltungsgrößen stehen entsprechende Master-größen zur Auswahl.

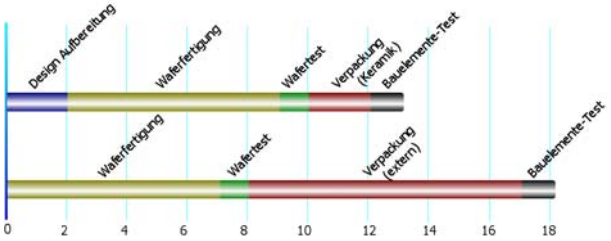


Abbildung 3: Durchlaufzeiten

Einen Überblick mit den charakteristischen Kenngrößen der Master gibt Tabelle 1.

Master	Digitale Zellen	Analoge Zellen	64x8 SRAM	Max. Pads	Chip-Größe	Prototype Gehäuse
GFN001	600	-	-	34	1.4 x 1.8	CLCC28
GFN002	2000	-	-	56	1.9 x 3.0	CLCC28
GFN004	4000	8*	-	82	3.0 x 3.3	CLCC44
GFN012	12.000	14*	-	136	4.6 x 4.6	CLCC68
GFN024	24.000	20*	-	184	5.9 x 5.9	CLCC84
GFN060	60.000	31*	-	284	8.6 x 8.6	CQFP100
GFN090	90.000	38*	-	320	10 x 10	CQFP160
GFN120	120.000	46*	-	408	12 x 12	CQFP208
GFQ010	10.000	40	-	82	3.0 x 3.3	CLCC44
GFQ032	32.000	78	10	136	4.6 x 4.6	CLCC68
GFQ060	60.000	101	12	184	5.9 x 5.9	CLCC84

* GFN-Analogteil, unterschiedlich zum neuen GFQ-Analogteil

Tabelle 1: Masterbezeichnung und Größen

Das IMS ist als Chiphersteller in Industriequalität nach ISO9001 und DIN EN 100 114 qualifiziert.

Die prozessierten Wafer werden im Anschluss getestet und in der IMS-internen Chipmontage zum gehäusten Bauelement im Keramikgehäuse weiterverarbeitet. Die Verpackung in Kunststoff-gehäuse wird für Serienprodukte von qualifizierten Zulieferern durchgeführt. Im abschließenden Bauelemente-Test wird das ASIC erneut geprüft und bei Fehlerfreiheit an den Kunden ausgeliefert.

Zukünftige Weiterentwicklungen des GATE FOREST zu einer "Structured ASIC"-Plattform können auch dedizierte Bereiche für Sensorarrays umfassen. So lassen sich zum Beispiel Detektoren für Licht und Magnetfelder gleichzeitig mit gemischt digital- und analoger Auswerteelektronik gemeinsam integrieren.

2. Mixed-Signal Design

2.1. Motivation

In einer Zeit, in der die Vernetzung von Geräten stetig voranschreitet, wird es von immer größerer Bedeutung, wichtige Daten sicher zu übertragen. Diese Sicherheit wird durch Verschlüsselung mittels geeigneter Algorithmen gewährleistet.

Es bestehen prinzipiell zwei verschiedene Arten von Verschlüsselungsalgorithmen, asymmetrische und symmetrische Verschlüsselung. Asymmetrische Verschlüsselung basiert auf einem sogenannten public-key Verfahren, bei dem der Absender die Daten mit dem öffentlichen Schlüssel verschlüsselt und der Empfänger diese mit seinem geheimen Schlüssel entschlüsselt.

Bei der symmetrischen Verschlüsselung werden die Daten mit ein und demselben geheimen Schlüssel ver- und entschlüsselt. Der Austausch des geheimen Schlüssels zwischen Absender und Empfänger geschieht in der Regel mittels asymmetrischer Verfahren.

Die Anforderungen an die Verschlüsselung sind einerseits die Sicherheit und andererseits die Geschwindigkeit des Algorithmus. 1997 hat deshalb das National *Institute for Standards and Technology* (NIST) einen Wettbewerb für einen neuen Verschlüsselungs-Standard, den *Advanced Encryption Standard* (AES) ausgeschrieben und die Gewinner waren die Entwickler des Rijndael Algorithmus [3].

Der Algorithmus eignet sich sehr gut zur Integration in Hardware und in Kombination mit analogen Schaltungen auf dem Chip, ist es nun möglich, analoge Signale, wie sie zum Beispiel von Sensoren geliefert werden, zu digitalisieren und direkt an der Quelle zu verschlüsseln.

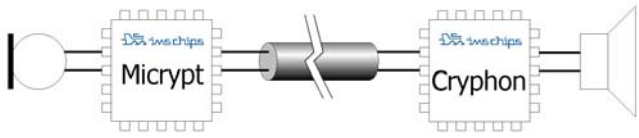


Abbildung 4: Applikationsbeispiel

Bei dem Mixed-Signal ASIC *"Micrypt"* das am IMS entworfen wurde, handelt es sich um eine Schaltung, die analoge Audiosignale in ein pulskodiertes digitales Signal wandelt, anschließend verschlüsselt und über eine digitale Schnittstelle überträgt. Ein Applikationsbeispiel zeigt Abbildung 4. Das Signal vom Mikrophon wird direkt vom ASIC verschlüsselt und über ein unsicheres Medium übertragen. Der Empfänger entschlüsselt den Datenstrom und wandelt die entschlüsselten Daten wieder in ein Audiosignal um.

2.2. Funktionalbeschreibung

Abbildung 5 zeigt das Blockdiagramm des modular aufgebauten ASICs. Es besteht aus einem AD-Wandler, einer seriellen Schnittstelle, Datenspeicher mit Busarbiter und dem AES-Kern.

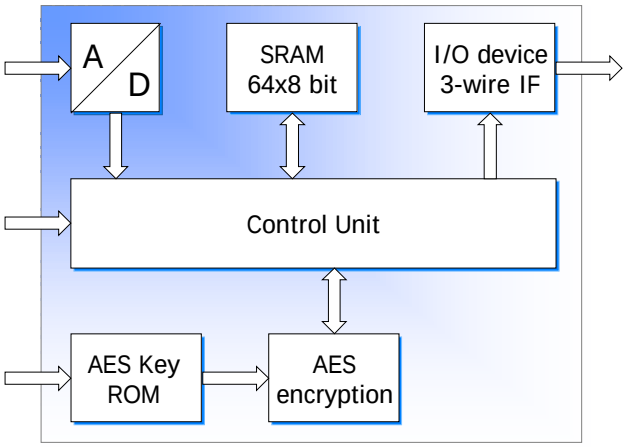


Abbildung 5: Blockdiagramm Micrypt

Der AD-Wandler setzt das analoge Audiosignal innerhalb einer Mikrosekunde nach dem SAR-Prinzip in ein digitales 8-Bit Signal um, wobei die Abtastrate bei 44.1 kHz liegt. Die Daten werden im SRAM zwischengespeichert, bis genug Daten für eine Verschlüsselung vorhanden sind. Der AES-Algorithmus erwartet eine Datenblockgröße von 128 Bit, die am Stück verschlüsselt werden.

Das Ausgabegerät liest die verschlüsselten Daten aus dem Speicher und überträgt diese seriell über eine 3-Draht Schnittstelle zur Applikation. Die Übertragung geschieht kontinuierlich und liefert daher Daten mit einer Geschwindigkeit von 44.1 kByte/s. Der Start einer verschlüsselten Übertragung wird durch das Senden von 16 Null-Bytes markiert, danach folgen die verschlüsselten Daten. Auf diese Weise kann sich die Empfängerseite auf eine verschlüsselte Datenübertragung synchronisieren.

Der AES-Kern liest unverschlüsselte Daten aus dem Speicher, verschlüsselt diese und legt sie wieder in den Speicher ab. Der Speicher wird außerdem gleichzeitig als Zwischenspeicher benutzt. Der Zugriff auf den Speicher durch AD-Wandler, Ausgabegerät

und AES-Kern wird vom Busarbiter koordiniert. Der AES-Kern benötigt für die Verschlüsselung einen geheimen 256-Bit Schlüssel. Einerseits befindet sich dieser 256-Bit Schlüssel in einem ROM, auf das von Außen nicht zugegriffen werden kann, andererseits kann die Applikation einen Schlüssel im ASIC ablegen.

Im Nachfolgenden wird auf den AES-Kern und die Implementierung des AD-Wandlers eingegangen.

2.2.1 AES-Kern

Das Herz des AES-Kerns in Abbildung 6 bildet der AES-Prozessor, der speziell für diese Aufgabe entwickelt wurde. Der Prozessor hat Zugriff auf einen externen Speicher über einen 8-Bit Datenbus und kommuniziert mit den speziellen arithmetischen Einheiten, die Bestandteil des Rijndael-Algorithmus sind.

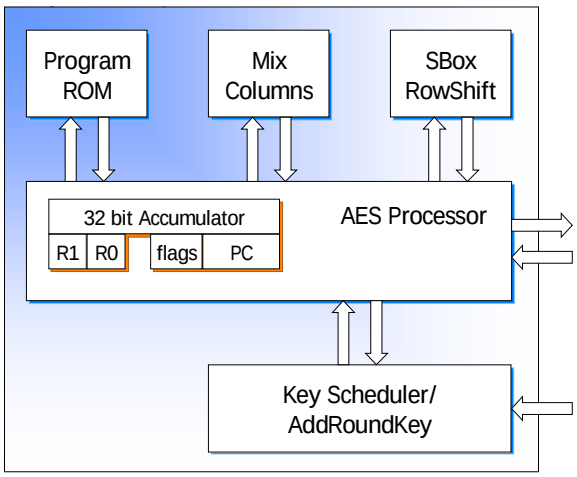


Abbildung 6: Blockdiagramm AES-Kern

Der Datenfluss und die Kommunikation werden vom Programm im ROM gesteuert. Da bei der Verschlüsselung und Entschlüsselung im Rijndael-Algorithmus die gleiche Datenmanipulation stattfindet, nur in anderer Reihenfolge, bestimmt das Programm im ROM, ob ver- oder entschlüsselt wird. Es wird also keine zusätzliche Hardware für eine Entschlüsselung benötigt.

Der AES-Algorithmus sieht 13 Datenmanipulationsrunden vor, bei der jeweils ein 128-Bit Schlüssel, der in jeder Runde geändert wird, verwendet wird. Die Berechnung des Rundenschlüssels findet im Block Key Scheduler statt.

Die 128-Bit Daten werden in einer 4x4-Matrix von jeweils einem Byte angeordnet, die in jeder Runde durch vier Funktionen manipuliert werden: SubByte, ShiftRow, MixColumns und AddRoundKey.

Die Funktion SubByte wird auf jedes Byte in der Matrix angewendet und besteht aus einer Lookup-Tabelle mit 256 Einträgen mit jeweils 8 Bit.

ShiftRow wird auf jeweils eine Zeile in der Matrix angewendet und rotiert den Inhalt der Zeile zyklisch um keine oder bis zu drei Positionen nach links, je nach Zeilenindex. Im AES-Kern wurde diese Funktion mit der Funktion SubByte kombiniert.

Die Funktion MixColumns multipliziert den Inhalt einer Spalte mit einem, für jede Spalte unterschiedlichen Polynom. Diese Bitoperation wurde mittels XOR-Verknüpfungen und Bitverschiebungen realisiert.

Zum Schluss wird der Rundenschlüssel mittels XOR mit der Matrix kombiniert.

Der Prozessor hat zwei Register R0 und R1, die als Zeiger und Zähler verwendet werden. Der 32-Bit Akkumulator kann eine Zeile oder Spalte der Matrix beinhalten. Der minimale Befehlssatz ist in Tabelle 2 aufgelistet:

Mnemonic	Kodierung
BNE <dir><val>	0dvvvvvv
LD R<0|1>, val4	100rvvvv
CMP R<0|1>, val4	101rvvvv
INC R<0|1>	110r----
LDROW	11100000
STROW	11100001
LDCOL	11100010
STCOL	11100011
SBOX	11100100
MIX	11100101
XOR	11100110
NEWKEY	11100111
HALT	11111111
NOP	11101110

Tabelle 2: Befehlssatz AES Prozessor

Das Programm im ROM umfasst 43 Bytes und wird im Folgenden dargestellt:

```
# encryption of one data block
# reset key scheduler
0000 NEWKEY
0001 NOP
# begin with a key addition (4 columns)
0002 LDREG R1, 0 # pointer for column
0003 LDCOL       # load column from RAM in accu
0004 XOR         # AddRoundKey, calc new Key
0005 STCOL       # store column in RAM
0006 INC R1
0007 CMP R1,4
0008 BNE -5
```

```
### start loop of 13 rounds
0009 LDREG R0, 0 # set round number
# SBox, RowShift (4 rows)
0010 LDREG R1, 0 # pointer for row
0011 LDROW       # load row from RAM in accu
0012 SBOX        # SBox + RowShift
0013 STROW       # store row in RAM
0014 INC R1
0015 CMP R1,4
0016 BNE -5
# Mix columns and add round key (4 columns)
0017 LDREG R1, 0 # pointer for column
0018 LDCOL       # load column from RAM in accu
0019 MIX         # MixColumns
0020 XOR         # AddRoundKey
0021 STCOL       # store column in RAM
0022 INC R1
0023 CMP R1,4
0024 BNE -6
#
0025 INC R0      # increment Round number
0026 CMP R0, 13
0027 BNE -17
### end of loop 13 rounds
# SBox, Rowshift
0028 LDREG R1, 0
0029 LDROW
0030 SBOX
0031 STROW
0032 INC R1
0033 CMP R1,4
0034 BNE -5
# Mix Columns and add round key
0035 LDREG R1, 0
0036 LDCOL
0037 XOR
0038 STCOL
0039 INC R1
0040 CMP R1,4
0041 BNE -5
#end of encryption
0042 HALT
```

Der Prozessor benötigt für eine Verschlüsselung 1770 Taktzyklen. Bei einer Taktrate von 80 MHz entspricht dieses etwa 22 µs pro verschlüsseltem Datenpaket.

2.2.2 AD-Wandler

Der auf dem Chip integrierte AD-Wandler aus Abbildung 7 beruht auf dem Prinzip der sukzessiven Approximation.

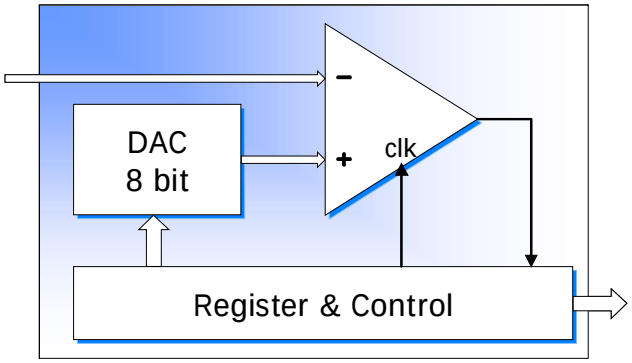


Abbildung 7: Blockdiagramm AD-Wandler

Der Komparator in dem AD-Wandler vergleicht das analoge Eingangssignal mit dem analogen Signal aus dem DA-Wandler. Der DA-Wandler wird von der digitalen Kontrolleinheit angesteuert. Eine Konvertierung beginnt damit, dass eine 1 an das höchstwertigste Bit angelegt wird, wobei die anderen Bits auf 0 liegen. Entscheidet der Komparator, dass das Eingangssignal kleiner ist als das Signal des DA-Wandlers, so wird das höchstwertigste Bit auf 0 gelegt, ansonsten bleibt es auf 1. Dasselbe Verfahren wird sukzessive für die restlichen Bits durchgeführt.

Der DA-Wandler wurde mittels R2R-Netzwerk, gesteuert von analogen Leistungsschaltern realisiert. Als Komparator wird ein schneller, getakteter Komparator, der als IMS IP-Block verfügbar ist, eingesetzt.

2.3. Mixed-Signal Designflow

Am IMS wurden in der Vergangenheit verschiedene analoge IP-Blöcke entwickelt und charakterisiert. Bei einer klassischen Entwicklung wird der Analogteil einer Schaltung mit Analogsimulatoren, wie z.B. SPICE simuliert und die digitale Schaltung in Verilog oder VHDL simuliert [4]. Eine reine Analogsimulation der Mixed-Signal Schaltung ist in der Regel nicht möglich, da die Simulationszeit extrem hoch ist oder der Speicherbedarf für eine derartige Simulation die Rechnerhardware übersteigt.

Eine Lösung für die Mixed-Signal Simulation ist die Modellierung von analogen Schaltungsblöcken in VHDL. Das Modell entspricht der Funktionalität der Blöcke bis zu einem gewissen Abstraktionsniveau das hinreichend für die Verifikation des Gesamtsystems ist.

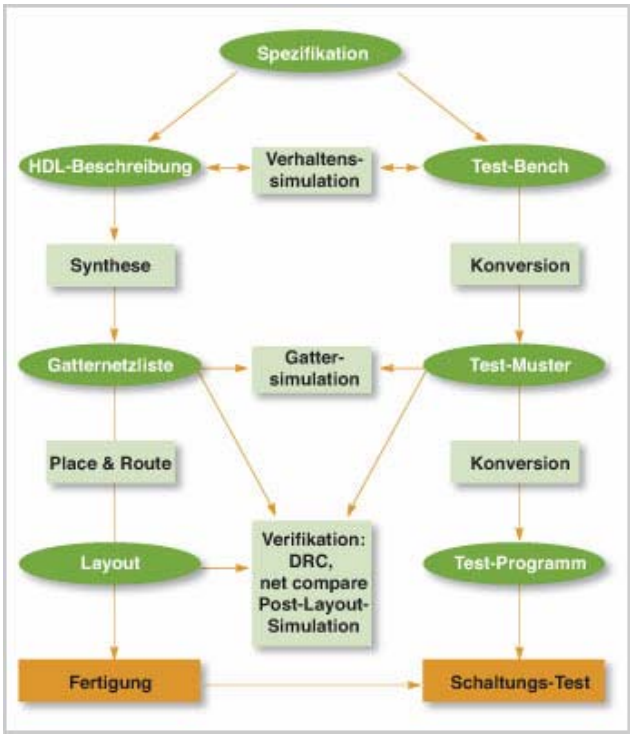


Abbildung 8: IMS Designflow

Ein anderer, großer Vorteil dieser VHDL-Modellierung besteht darin, dass der digitale Standard-IMS-Designflow, abgebildet in Abbildung 8 verwendet werden kann [5]. Die Synthesewerkzeuge betrachten die analogen Blöcke als Black Boxe. Die Timing-eigenschaften und die Eigenschaften der Ein- und Ausgangssignale dieser Blöcke sind den Synthesewerkzeugen bekannt. Hiermit kann eine Synthese wie bei einer rein digitalen Schaltung durchgeführt werden und die dabei generierte Netzliste wird von den Layoutwerkzeugen importiert und problemlos weiterverarbeitet.

Die Layoutwerkzeuge platzieren die analogen IP Module automatisch im analogen Bereich des jeweiligen Masters und verbinden diese mit den digitalen Zellen.

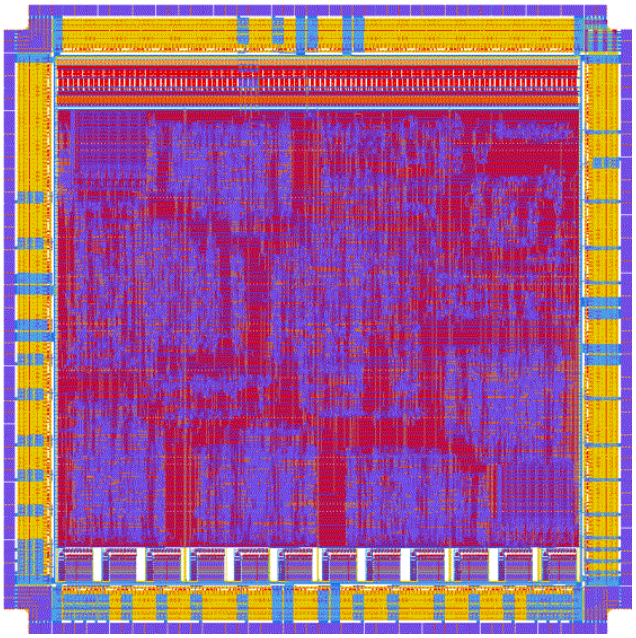


Abbildung 9: Layout des Micrypt ASICs

Das fertige Layout vom Micrypt ASIC zeigt Abbildung 9. Die Schaltungsgröße entspricht etwa 12.000 NAND2 Gatter und läuft bei einer Taktfrequenz von 10 MHz. Die maximale Betriebsfrequenz beträgt 80 MHz, bei der der analoge Eingang dann mit 352.800 Samples/s abgetastet wird.

3. Zusammenfassung

Die neue 0,5 µm-GATE FOREST Technologie bietet neben einer deutlichen Vergrößerung des digitalen Bereiches auch Platz für umfangreiche analoge Funktionen. Eine gemeinsame Integration von Sensoren und Auswerteelektronik bietet in einem Structured-ASIC Baukasten Raum für neue Ideen und Produkte. Ausführliche Informationen sind im Internet verfügbar.

Zu den Autoren

Dr.-Ing. Christian Burwick ist seit 2003 am IMS tätig. Schwerkpunkte seiner Arbeit liegen im Bereich

der Device- und Process-Simulation sowie der Designwerkzeuge.

Ir. Cor Scherjon ist seit 1997 im Bereich Bildsensorik und Mixed-Signal ASIC Entwicklung am IMS tätig.

Literatur

[1] W. Ludescher, **CMOS-Gate-Forest IMS-Foundry Service**, MPC-Workshop Dezember 1988, Heilbronn, Seite 6-16

[2] J. Burghartz, T. Deuble, **Schaltungsentwurf für GATE FOREST-Chips**, MPC-Workshop Februar 2006, Esslingen, Seite 5-9

[3] Joan Daemen and Vincent Rijmen, **The Design of Rijndael: AES - The Advanced Encryption Standard**. Springer-Verlag, 2002. ISBN 3-540-42580-2.

[4] R.D.M.Hunter and T.T.Johnson, **Introduction to VHDL**. Chapman & Hall, 1996, ISBN 0-412-73130-4.

[5] IMS Homepage Arbeitsgebiete, **ASIC-Entwurf** www.ims-chips.de/home.php?id=a3b2c2de

Testchipentwurf eines Sigma-Delta-A/D-Wandlers

Ante Trutin, Gerhard Forster

Institut für Kommunikationstechnik

Hochschule Ulm, Prittwitzstraße 10, 89075 Ulm

trutin@mail.hs-ulm.de forster@hs-ulm.de

Die Entwicklung eines Analog-Digital-Umsetzers stellt auch heute noch besondere Anforderungen an einen Mixed-Signal-Designflow. Analoge und digitale Schaltungskomponenten stehen hier in besonderer Wechselwirkung. Das Konzept des Sigma-Delta-Wandlers (ΣΔ) kommt dem Trend der weiteren Miniaturisierung entgegen, weil es die Fortschritte bei Packungsdichte, Taktfrequenz und Verlustleistung in digitalen Schaltungen nutzbar macht, um im Gegenzug die klassischen Anforderungen an analoge Schaltungen wie Offsetspannung, Rauschen und Linearität zu entspannen.

Der vorliegende Beitrag befasst sich mit der Entwicklung eines Sigma-Delta-Wandlers bis zum Tape Out für einen 0,35 µm-CMOS-Prozess in einem durchgängigen Design Flow. Die Auflösung des Wandlers ist auf 12 Bit bei einer Signalbandbreite von 20 kHz und einer Abtastrate von 20 MHz ausgelegt. Die Entwicklung des zweistufigen analogen Modulators erfolgte mit dem Simulator Spectre®. Das digitale Filter wurde auf der Basis einer High-Level-Synthese mit Matlab Simulink® entworfen und von dort automatisch in VHDL-Code umgesetzt. Der mit Hilfe des Analogsimulators erzeugte Bitstrom des Modulators wurde als Stimulus für die Verifikation des Filters in ModelSim® genutzt. Anschließend an die Schaltungssynthese mit Design Compiler® wurde das komplette Mixed-Signal-Back-End mit Cadence-Tools umgesetzt.

1. Grundlagen

1.1. Signalabtastung

Die Grundlage der Digital-Analog-Wandlung beruht auf dem von Shannon begründeten Abtasttheorem. Nach diesem lässt sich ein analoges Signal $s(t)$ eindeutig wieder rekonstruieren, falls die Wahl der Abtastfrequenz der Bedingung

$$f_s = 2f_\Delta$$

genügt. f_Δ bezeichnet hierbei die Bandbreite des Signals $s(t)$. Der ideale Abtastvorgang im Zeitbereich lässt sich beschreiben als eine Multiplikation des Signals mit einer Folge von Dirac-Impulsen. Mit der Ausblendeigenschaft der Dirac-Funktion erhält man somit für das abgetastete Signal die kompakte Darstellung:

$$s_a(t) = \sum_{n=-\infty}^{+\infty} s(nT)\delta(t-nT)$$

Der Parameter T bezeichnet hierbei das zeitlich konstante Abtastintervall und n einen Laufindex, der dem n-ten Abtastwert des Signals entspricht. Mit Hilfe der Fourierbeziehungen

$$s(t) \quad \circ\!\!-\!\!\bullet \quad S(f)$$

sowie dem Faltungstheorem

$$s(t)\cdot g(t) \quad \circ\!\!-\!\!\bullet \quad S(f) * G(f)$$

ergibt sich für die Fouriertransformierte des abgetasteten Signals $s_a(t)$:

$$\mathcal{F}\{s_a(t)\} = \frac{1}{T}\sum_{k=-\infty}^{+\infty} S\left(f-\frac{k}{T}\right)$$

Eine Abtastung von $s(t)$ mit einem zeitlichen Intervall T im Zeitbereich entspricht somit einer kontinuierlichen periodischen Wiederholung des Spektrums $S(f)$ mit dem Intervall $T^{-1} = f_s$. Genügt diese der von Shannon definierten Forderung nicht, so resultieren im Frequenzbereich Überlappungen der Spektren, was zu Mehrdeutigkeiten im Zeitbereich führt (Aliasing).

Typischerweise erfolgt die Signalabtastung nicht mit der Nyquistrate, sondern mit einer wesentlich höheren Abtastrate. Man definiert dazu ein Überabtastverhältnis (OSR)

$$OSR = \frac{f_s}{2f_\Delta}$$

Diese Überabtastung führt dazu, dass sich der Abstand der periodisch fortlaufenden Spektren $S(f)$ vergrößert. Ein Vorteil daran ist, dass hierdurch die An-

forderungen an ein Filter hinsichtlich der Flankensteilheit zur Unterdrückung der unerwünschten Frequenzbereiche deutlich reduziert werden. Dies wirkt sich beispielsweise beim Einsatz eines FIR-Filters direkt auf die Anzahl der benötigten Multiplikationen und somit auf den resultierenden Platzbedarf aus. Ferner lässt sich das durch die Quantisierung erzeugte Quantisierungsrauschen, wie nachfolgend gezeigt wird, reduzieren. Dies wiederum wirkt sich direkt auf das Signal-zu-Rauschleistungsverhältnis (SNR) und folglich auf das Auflösungsvermögen eines Analog-Digital-Wandlers aus. Die Überabtastung bildet die Grundlage des Sigma-Delta-Wandlungsprinzips.

1.2. Quantisierung

Das abgetastete Signal $s_a(t)$ besitzt noch einen kontinuierlichen Wertebereich. Für eine nachfolgende digitale Verarbeitung müssen daher die unendlich feinen Abstufungen der einzelnen Werte auf eine endliche Anzahl gerundet werden. Mögliche Werte, die das Signal $s(t)$ annehmen kann, entsprechen der Menge $M_s \subset \mathbb{R}$. Diese wird in eine bestimmte Anzahl gleichförmiger Amplitudenstufen der Größe Δ unterteilt. Aus folgender Beziehung erhält man nun die Anzahl von Bits, die für eine eindeutige Zuordnung benötigt werden:

$$K \geq \log_2\left(\frac{M_s}{\Delta}\right)$$

Die Amplitudenstufe Δ wird als Least Significant Bit LSB bezeichnet. Ein Abtastwert $s(nT)$ aus der Menge M_s wird nun zwischen zwei Amplitudenstufen

$$n\Delta \leq s(nT) < (n+1)\Delta$$

der jeweils näher liegenden zugeordnet. Die Grenze, in welcher sich der Quantisierer für einen Wert entscheidet, liegt offensichtlich bei $\pm\Delta/2$. Der Fehler e, welchen der Quantisierer gegenüber dem tatsächlichen Wert aus M_s durch diese Zuordnung verursacht, liegt folglich im Intervall

$$-\frac{\Delta}{2} \leq e \leq \frac{\Delta}{2}$$

Wird angenommen, dass der Quantisierungsfehler im oben angegebenen Intervall eine konstante Verteilungsdichte besitzt, so erhält man für die Rauschleistung [1]

$$P_e = \int_{-\frac{\Delta}{2}}^{+\frac{\Delta}{2}} \rho\,(e)^2\,de = \frac{\Delta^2}{12}$$

Dies resultiert in einer Rauschleistungsdichte, welche im Intervall der Breite f_s konstant ist. Wird die Abtastfrequenz f_s um einen Faktor K erhöht, so verbreitert sich die Bandbreite der Rauschleistungsdichte um den Faktor K, die Amplitude nimmt dagegen um den

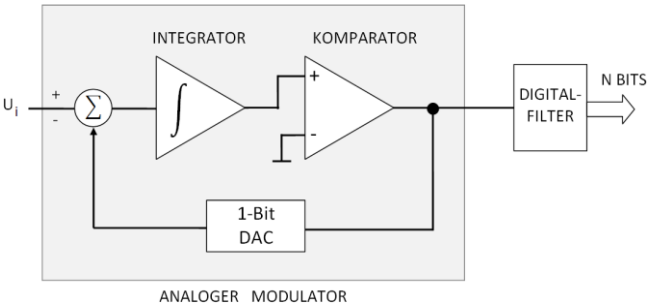


Bild 1: Sigma-Delta-Wandler erster Ordnung

gleichen Faktor ab. Erfolgt nun eine ideale Tiefpassfilterung mit der Grenzfrequenz f_Δ, so befindet sich eine geringere Rauschleistung im Nutzband. Der Signal/Rauschabstand SNR verbessert sich, wie in [1] gezeigt, um den Faktor

$$SNR_{OSR} = 10\log_{10}(OSR)$$

Mit dem Signal/Rauschabstand erhöht sich die effektive Anzahl der Bits (ENOB).

1.3. Sigma-Delta Modulation

Das Prinzipschaltbild eines ΣΔ-A/D-Wandlers erster Ordnung, auf welchen sich zur Vereinfachung auch die nachfolgenden Herleitungen beziehen, ist in Bild 1 dargestellt. Er besteht aus zwei Teilen, dem analogen Modulator und einem digitalen Filter. Die Ordnung eines Modulators ergibt sich aus der Anzahl der Integratoren. Mit dem ΣΔ-Modulator wird eine Reduzierung des Quantisierungsrauschens im Nutzband erreicht. Zur vereinfachten analytischen Erfassung kann der 1-Bit-Quantisierer durch eine unabhängige Rauschquelle ersetzt werden [2]. Das Ausgangssignal des Modulators besteht somit aus dem integrierten Differenzsignal und einem additiven Störfehler. Es lässt sich zeigen, dass unter diesen vereinfachenden Annahmen zwischen dem Ein- und Ausgangssignal des ΣΔ-Modulators folgender Zusammenhang besteht:

$$y[nT] = s_a[nT - T] + e[nT] - e[nT - T]$$

n bezeichnet hier eine Zählvariable und T die Taktperiode des ΣΔ-Modulators. Im Laplace-Bereich ergibt sich folgende Darstellung:

$$Y(z) = S_a(z)z^{-1} + E(z)(1 - z^{-1})$$

An dieser Funktion ist gut zu erkennen, dass Nutzsignal und Fehlersignal unterschiedlich übertragen werden. Für das Nutzsignal $H_x(z)$ sowie das Fehlersignal $H_e(z)$ gelten dabei folgende Übertragungsfunktionen:

$$H_x(z) = z^{-1}$$

$$H_e(z) = 1 - z^{-1}$$

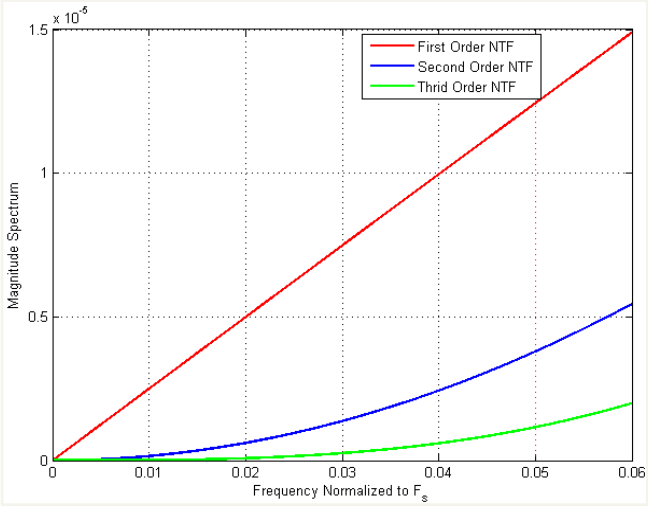


Bild 2: Rauschübertragungsfunktionen unterschiedlicher Ordnung

Der aktuelle Fehlerbeitrag ist also die Differenz des aktuellen und des vorhergehenden Fehlers. Hierin liegt bei einem ΣΔ-Wandler der entscheidende Vorteil gegenüber einem linearen Quantisierer. Je höher die Taktrate des Modulators, desto kleiner wird der additive Fehler. Nach [2] ergibt sich im Frequenzbereich die Übertragungsfunktion

$$|H_e(f)| = \sqrt{2\left(1-\cos 2\pi\frac{f}{f_s}\right)} = \sqrt{2}\sin\left(2\pi\frac{f}{f_s}\right)$$

Die übertragene Rauschleistung nach einem Filter erhält man über die verallgemeinerte Wiener-Lee-Beziehung [3]. Eine anschließende Tiefpassfilterung führt zu einem resultierenden Quantisierungsrauschen [1], [4] im Nutzband von

$$P_{e,\Sigma\Delta} = \int_{-f_\Delta}^{f_\Delta} \frac{\Delta^2}{12 f_s}\, 2\sin\left(2\pi\frac{f}{f_s}\right)^2 \cong P_e\frac{\pi^2}{3}OSR^{-3}$$

Daraus folgt eine Verbesserung des Signal/Rausch-abstands SNR um den Faktor

$$SNR_{\Sigma\Delta} = 10\log_{10}\left(\frac{OSR^3}{\pi^2/3}\right)$$

Folglich führt eine Verdopplung der Überabtastrate zu einer Erhöhung des Signal/Rauschabstands um 9 dB, was einer Erhöhung der effektiven Bitzahl ENOB um 1,5 Bit entspricht. Für einen Modulator zweiter Ordnung lassen sich die obigen Beziehungen analog herleiten [1], [4]. Man erhält dann einen weiter erhöhten Signal/Rauschabstand

$$SNR_{\Sigma\Delta} = 10\log_{10}\left(\frac{OSR^5}{\pi^4/5}\right)$$

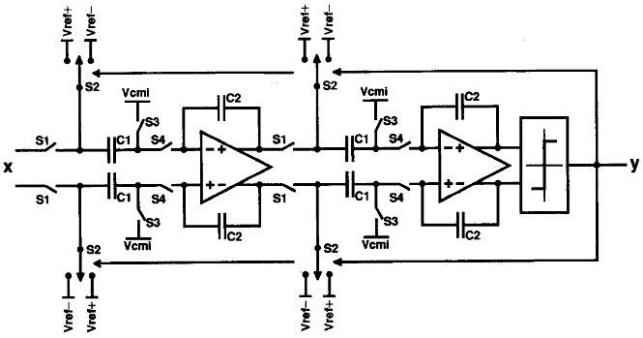


Bild 3: Systementwurf für den Modulator zweiter Ordnung

Die Auflösung erhöht sich in diesem Fall um 2,5 Bit. Beispielhaft sind in Bild 2 die Übertragungsfunktionen eines ΣΔ-Modulators erster, zweiter und dritter Ordnung dargestellt. Modulatoren dritter Ordnung neigen allerdings zu erhöhter Instabilität, weshalb man für eine konkrete Realisierung zu anderen Architekturen übergehen muss [4], [2].

2. Systementwurf

Zu Beginn des Schaltungsentwurfs des Modulators müssen zunächst dessen Entwurfsparameter näher spezifiziert werden. Diese lassen sich aus den obigen Beziehungen ableiten. Gemäß der Auflösungsanforderung von $N = 12$ Bit erhält man mit Hilfe der Formel

$$SNR = 6.02N + 1.76$$

einen ersten Richtwert für das benötigte SNR von etwa 74 dB. Man kann zeigen, dass zwischen dem Signal/Rauschabstand SNR, der Modulatorordnung n und der Überabtastrate OSR folgender Zusammenhang besteht:

$$SNR = 10\log_{10}\left(\frac{3(2n+1)}{2\pi^{2n}}\cdot OSR^{2n+1}\right)$$

Damit lassen sich geeignete Werte für n und OSR ermitteln.

Das digitale Filter, dargestellt in Bild 1, hat im Wesentlichen zwei Aufgaben, zum einen die gleitende Mittelwertbildung über das Ausgangssignal des Modulators und zum anderen eine anschließende Tiefpassfilterung. Aufgrund der vorgegebenen Nutzsignalbandbreite von 20 kHz ergibt sich für das digitale Filter eine entsprechende Grenzfrequenz f_g. Aus der zuvor abgeleiteten Überabtastrate OSR ergibt sich die notwendige Dezimation. Zusammenfassend sind in der Tabelle 1 die Spezifikationen des Modulators und des Filters zusammengefasst.

Der Systementwurf des Modulators in vollsymmetrischer Ausführung [5] ist in Bild 3 dargestellt. Die Integratoren sind, im Gegensatz zu einem zeitkontinuierlichen Konzept [6], in Switched-Capacitor-Technik realisiert. Diese Technologie ist bei der vor-

gegebenen Spezifikation besonders vorteilhaft einsetzbar.

Tabelle 1: Spezifikationen des Wandlers

Auflösung	$N = 12\ \text{Bit}$
Eingangsbandbreite	$f_g = 20\ \text{kHz}$
Überabtastrate	$OSR = 480$
Modulatorordnung	$n = 2$
Taktrate (Modulator)	$f_{Clk} = 20\ \text{MHz}$

Um Fehler durch Ladungsinjektion und kapazitiven Taktdurchgriff möglichst gering zu halten, werden als Schalter Transmission Gates mit ausgeglichenen Kapazitäten eingesetzt, die mittels nichtüberlappender Takte und in streuunempfindlicher Sequenz [5] angesteuert werden. Die Kondensatoren sind so dimensioniert, dass mit den verwendeten Transmission Gates minimaler Bauform die maximale Ladezeitkonstante τ_{TG} noch unter 1/10 der Taktperiode bleibt:

$$\tau_{TG} = R_{on} C < \frac{1}{10} T_{Clk}$$

Weiterhin muss verhindert werden, dass die Ausgänge der Integratoren übersteuern. Damit liegen alle Kondensatorwerte unter 1 pF. Über den vollsymmetrischen Aufbau des Modulators sollen verbleibende Fehler durch Ladungsinjektion, insbesondere aber Fehler durch Störeinkopplungen aus dem Digitalteil über das Substrat, unterdrückt werden.

Für den Aufbau der Integratoren ist ein vollsymmetrischer Operationsverstärker erforderlich. Entscheidend sind seine Auflösung und sein Settling-Verhalten [2], [4], [7]. Kritische Zieldaten sind hierbei die Leerlaufverstärkung V_0, die Transitfrequenz f_T, das Eingangsrauschen e_n sowie die Slew Rate SR. Angesetzt wird daher $V_0 >$ 80 dB, $f_T >$ 100 MHz und $e_n <$ 10 nV/sqrt(Hz). Die Stromaufnahme soll 500 µA nicht überschreiten. Wegen seiner Beschaltung mit SC-Elementen benötigt der Verstärker entsprechende Phasenreserve für eine Rückkopplung als Unity-Gain-Buffer.

Der Komparator als 1-Bit-A/D-Wandler hat im Wesentlichen nur die Anforderung einer kurzen Verzögerungszeit sowie einer geringen Verlustleistung zu erfüllen. Angesetzt wird $t_d < \frac{1}{10} T_{Clk}$.

Tabelle 2 zeigt zusammenfassend die Zieldaten der benötigten Komponenten im analogen Modulator.

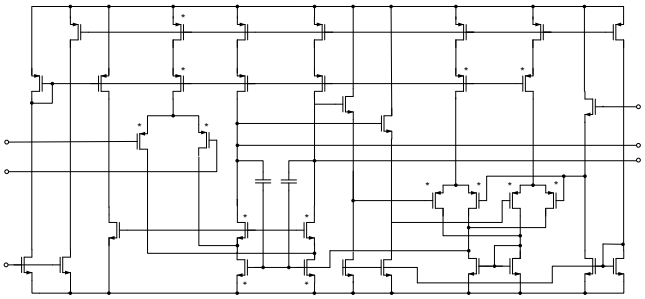


Bild 4: Symmetrischer Operationsverstärker

Tabelle 2: Systemparameter für den Modulator

OPV	Leerlaufverstärkung	$V_0 \geq 80\ \text{dB}$
	Transitfrequenz	$f_T \geq 100\ \text{MHz}$
	Phasenreserve	$\phi_R > 60°$
	Eingangsrauschen	$e_n < 10\ \frac{\text{nV}}{\sqrt{\text{Hz}}}$
	Settling Time	$t_{Settl} < 40\ \text{ns}$
	Stromaufnahme	$I_{DD} < 500\ \mu\text{A}$
KOMP	Verzögerungszeit	$t_{d,max} < 5\ \text{ns}$
	Ausgangsspannung	$V_o = \pm 1.65\ \text{V}$
	Stromaufnahme	$I_{DD} < 500\ \mu\text{A}$
TG	Zeitkonstante	$R_{on} C_{last} < \frac{T}{5}$

3. Schaltungsentwurf

Entwurf und Simulation der analogen Schaltungskomponenten erfolgte mit Cadence Virtuoso Schematic Composer sowie Spectre. Die Schaltungstopologie des symmetrischen Operationsverstärkers ist in Bild 4 dargestellt. Es handelt sich um einen einstufigen Verstärker mit hoher Phasenreserve, auch bei kapazitiver Last. Die PMOS-Differenzeingangsstufe arbeitet auf eine gefaltete Kaskode, deren Ausgänge unmittelbar die symmetrischen Ausgänge des Operationsverstärkers bilden. Die im rechten Teil angeordnete Schaltung dient der Gleichtaktregelung. Sie benötigt eine besondere Großsignal-Festigkeit, um Schwankungen der Gleichtaktspannung zu verhindern. Die mit einem Stern markierten Transistoren bestehen jeweils aus 2 parallel geschalteten Einzeltransistoren mit dem Ziel einer symmetrischen Layoutrealisierung (Kleeblattstruktur).

Der Komparator musste nicht erneut entwickelt werden. Hier konnte auf ein fertiges Design [8] zurückgegriffen werden, das die geforderten Spezifikationen erfüllt. Bei den Transmissiongates konnte ebenso auf fertige Standardzellen der Herstellerbibliothek zurückgegriffen werden.

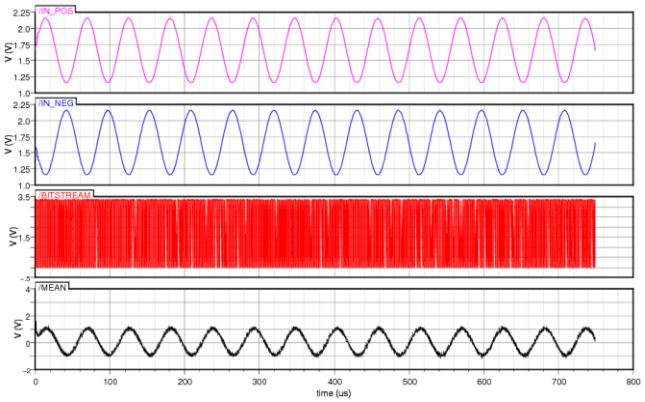


Bild 5: Simulation des Modulators mit einem Sinussignal

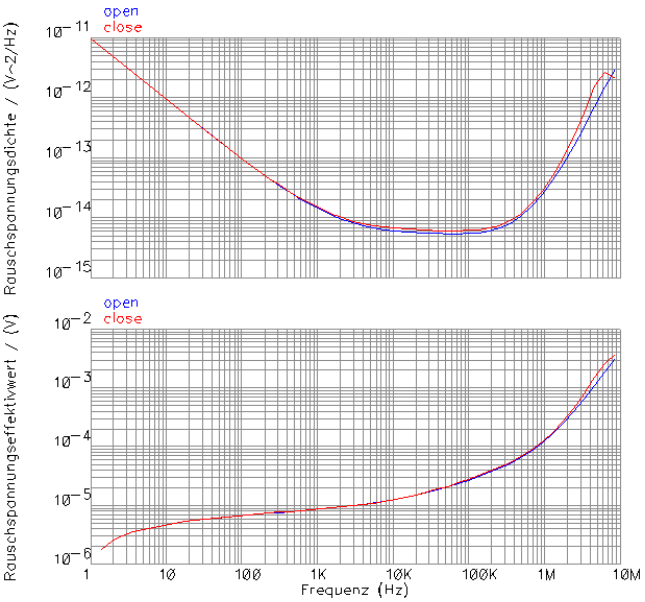


Bild 6: Rauschsimulation des Modulators

Ausgewählte Simulationsergebnisse des Operationsverstärkers sind in der Tabelle 3 dargestellt.

Tabelle 3: Simulationsergebnisse Operationsverstärker

Leerlaufverstärkung	$V_0 = 82{,}7\ \mathrm{dB}$
Transitfrequenz	$f_T = 169\ \mathrm{MHz}$
Stromaufnahme	$I_{DD} = 489\ \mu\mathrm{A}$
Phasenreserve	$\phi_R = 71.6°$
Settlingtime (DM)	$t_{Settl} = 25.8\ \mathrm{ns}$
Eingangsrauschen.	$e_n = 8.59\ \frac{\mathrm{nV}}{\sqrt{\mathrm{Hz}}}$

In umfangreichen Simulationen wurde die Funktionalität des Modulators verifiziert. Bild 5 zeigt das Ergebnis einer Gesamtsimulation. Dem Modulator wurde ein

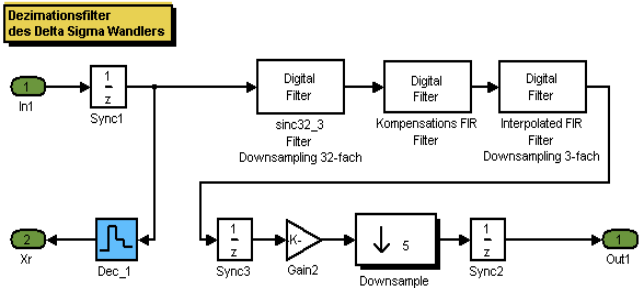


Bild 7: Digitale Filterstrecke

symmetrisches Sinussignal bei der Signalfrequenz $f = 20\ \mathrm{kHz}$ zugeführt. Der daraus resultierende hochfrequente 1-Bit-Strom wurde über einen nachgeschalteten RC-Tiefpass gemittelt. Wie zu erkennen ist, entspricht der gebildete Mittelwert dem Eingangsdifferenzsignal.

Das Ergebnis einer Rauschsimulation des Modulators ist in Bild 6 dargestellt, einmal als spektrale Rauschspannungsdichte und zum anderen als Rauschspannungseffektivwert, der sich als Integral bis zur angegebenen Frequenz ergibt [10].

Laut [2] ergibt sich ein idealer Rauscheffektivwert von

$$S_{n,ideal} = \frac{A_{max}}{2^N\sqrt{12}}$$

Für eine maximale Amplitude von 3,3 V sowie einer Auflösung von 12 Bit erhält man somit einen Wert von 233 µV. Die Simulation ergibt im Frequenzband bis 20 kHz einen Rauschspannungseffektivwert von 15 µV. Vom Rauschen des Modulators wird somit keine negative Auswirkung auf das angestrebte Auflösungsvermögen des Wandlers erwartet.

4. Filterentwurf

Die vollständige Entwicklung des digitalen Filters erfolgte im Rahmen einer High-Level-Synthese mit Hilfe von Matlab/Simulink. Durch den Einsatz des HDL Coders ist es möglich, die Realisierung auf einer hohen Abstraktionsebene durchzuführen, ohne direkt auf die sehr zeitaufwändige und kritische Hardwarebeschreibung eingehen zu müssen [11]. Zudem kann zu jedem Zeitpunkt des Entwicklungsprozesses das aktuelle Modell verifiziert und bei Bedarf geändert werden.

Die entwickelte Filterstrecke des Tiefpass- und Dezimationsfilters des ΣΔ-Wandlers ist in Bild 7 dargestellt [12]. Sie besteht aus einem Filter zur Mittelwertbildung sowie einer anschließenden Tiefpassfilterung. Ferner wurde zur Einhaltung des geforderten Auflösungsvermögens des Wandlers eine zusätzliche Kompensationsstufe implementiert [13]. Durch den 5-fachen Dezimationsblock erkennt man weiter, dass die angesetzte 480-fache Dezimation nicht in einem, sondern

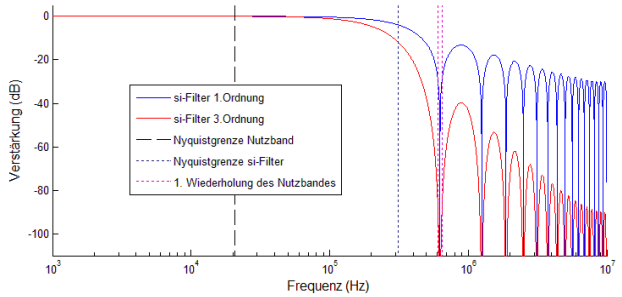


Bild 8: Übertragungsfunktionen des Si-Filters

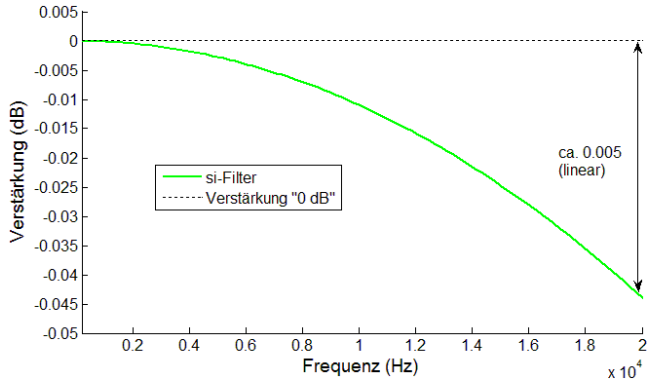


Bild 9: Vergrößerte Übertragungsfunktion des Si-Filters (bis 20 kHz)

in mehreren Schritten durchgeführt wird. Der Gain-Block dient zur korrekten Skalierung. Die eingesetzten Sync-Blöcke werden zur Taktsynchronisierung benötigt. Der erste Sync-Block steht an Stelle eines Abtastglieds, das aber im HDL-Coder nicht verfügbar ist.

4.1. Mittelwertfilter

Der vom Modulator erzeugte Bitstrom enthält Mittelwertinformationen über das anliegende Eingangssignal. Betrachtet man den Bitstrom als eine bipolare Folge, $b_i \in \{-1,1\}$, so erhält man durch Mittelwertbildung einen repräsentativen Wert für das am Modulatoreingang anliegende Signal.

$$V_o = V_R \left(\frac{1}{N} \sum_{i=1}^{N} b_i \right) \approx \overline{V_{in}}$$

N bezeichnet hierbei die Anzahl von Werten, über welche eine Mittelwertbildung erfolgt und V_R die Referenzspannung des Komparators. Voraussetzung für einen möglichst genauen Mittelwert ist eine deutlich höhere Taktgeschwindigkeit des Modulators im Vergleich zur maximalen Signalfrequenz ($f_{mod} \gg f_\Delta$).

Die Übertragungsfunktion eines Mittelwertfilters (Si-Filter) kann vergleichsweise einfach hergeleitet werden [9]. Die konkrete Realisierung des Mittelwertfilters erster Ordnung ist gegeben durch

$$H(z) = \frac{1 - z^{-k}}{1 - z^{-1}}$$

Wobei k die Zahl der in die Mittelung einbezogenen Samples bezeichnet.

Die dargestellte Übertragungsfunktion lässt sich bei näherer Betrachtung in das Produkt zweier unabhängiger Übertragungsfunktionen zerlegen – die eines Integrierers sowie eines Differenzierers. Diese lassen sich vergleichsweise einfach über Summations- und einfache Verzögerungsglieder realisieren. Anstatt eine k-fache Verzögerung zu realisieren, kann man geschickter Weise eine k-fache Dezimationsstufe zwischen Integrierer und Differenzierer einsetzen [13], was zu keiner Änderung der Funktionsweise führt. Insbesondere werden keine Multiplizierer benötigt. Dadurch wird der Entwurf vereinfacht, was bei einer späteren Codesynthese eine kompaktere Hardware und geringere Verlustleistung zur Folge hat. Nachteilig ist, dass durch die Dezimation die periodische Wiederholung der Übertragungsfunktion in das Nutzband zurückgefaltet wird. Aus diesem Grund ist man gezwungen, diesen Einfluss durch eine höhere Dämpfung gering zu halten.

Aus diesem Grund wurde ein Si-Filter dritter Ordnung gewählt. Mit $k = 32$ erhält man die Übertragungsfunktion

$$H(z) = \left(\frac{1 - z^{-32}}{1 - z^{-1}} \right)^3$$

Auch diese Übertragungsfunktion lässt sich in das Produkt zweier unabhängiger Übertragungsfunktionen zerlegen, wobei zwischen Integrierter und Differenzierer eine 32-fache Dezimationsstufe benötigt wird.

Die Übertragungsfunktionen der Si-Filter erster und dritter Ordnung sind in Bild 8 dargestellt. Man erkennt, dass die Rückfaltung ins Nutzband beim Filter dritter Ordnung deutlich geringere Auswirkungen hat als beim Filter erster Ordnung.

Die Wahl eines Si-Filters dritter Ordnung hat allerdings gegenüber dem Filter erster Ordnung den Nachteil einer größeren Dämpfung im Nutzband. Diese ist in Bild 9 vergrößert dargestellt. Die maximale Dämpfung im Nutzband bis 20 kHz beträgt ca. 0,05 dB. Dieser Wert ist zwar gering, er reicht allerdings bereits aus, das Auflösungsvermögen des Wandlers negativ zu beinträchtigen.

4.2. Kompensationsfilter

Die oben genannte Dämpfung des Si-Filters im Nutzband muss wieder ausgeglichen werden. Hierzu dient ein Kompensationsfilter, das im Bereich des Nutzbands gerade die inverse Übertragungsfunktion des Si-Filters besitzt. Im Hinblick auf einen möglichst geringen Aufwand wurde hierzu ein FIR-Filter mit nur

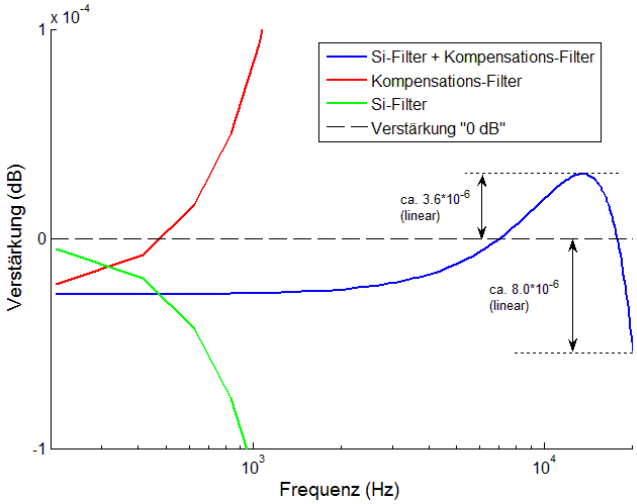


Bild 10: Übertragungsfunktionen des Si-Filters, des Kompensationsfilters sowie die Resultierende

drei Koeffizienten entworfen. Die Filterkoeffizienten wurden mittels folgender Optimierungsaufgabe [12] in Matlab bestimmt:

$$\min_{a_0,a_1}\{\|G(f,a_0,a_1)\|_2^2,\}$$

Die Funktion G stellt hierbei die Übertragungsfunktion dar. Die resultierende Übertragungsfunktion des Kompensationsfilters ist in Bild 10 über den Bereich von 20 kHz (Nutzband) dargestellt. Wie man erkennen kann, weist die Überlagerung mit der Übertragungsfunktion des Si-Filters nun wie gewünscht einen nahezu linearen Verlauf auf. Es verbleiben nur noch geringe Abweichungen von der 0 dB-Linie. Diese liegen in einem Bereich von 10^{-5} (linear) und können toleriert werden, da sie das Auflösungsvermögen des Sigma-Delta-Wandlers nicht mehr nennenswert beeinträchtigen.

4.3. Tiefpassfilter

Gewünscht ist ein Tiefpassfilter mit einer Grenzfrequenz von 20 kHz und möglichst hoher Flankensteilheit. Dies erfordert in der Regel entsprechend viele Filterkoeffizienten, verbunden mit hohem Platzbedarf und großer Verlustleistung in der Hardwarerealisierung. Abhilfe schafft an dieser Stelle ein Interpolated-FIR-Filter, das einen guten Kompromiss zwischen der Anzahl an benötigten Filterkoeffizienten und der Flankensteilheit bietet. Der Ansatz hierbei liegt darin, zunächst mit wenigen Filterkoeffizienten die Grenzfrequenz zu definieren. Man erhält dadurch eine gute Flankensteilheit, durch die Einsparung an Koeffizienten wird allerdings die Übertragungsfunktion schneller periodisch wiederholt, wodurch höhere Frequenzanteile nicht mehr gedämpft werden. Abhilfe schafft nun ein zweites Filter, an das nur sehr geringe Anforderungen bezüglich der Flankensteilheit gestellt werden.

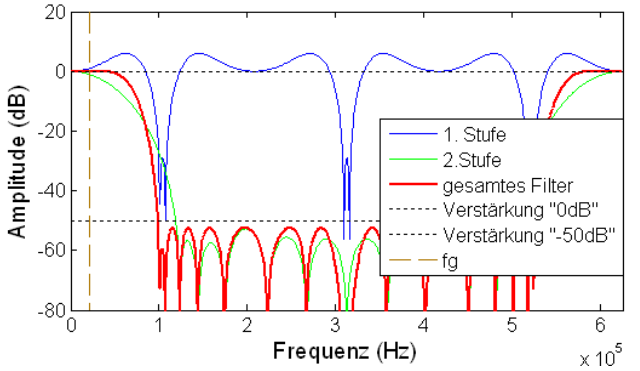


Bild 11: Übertragungsfunktionen des Interpolated FIR-Filters

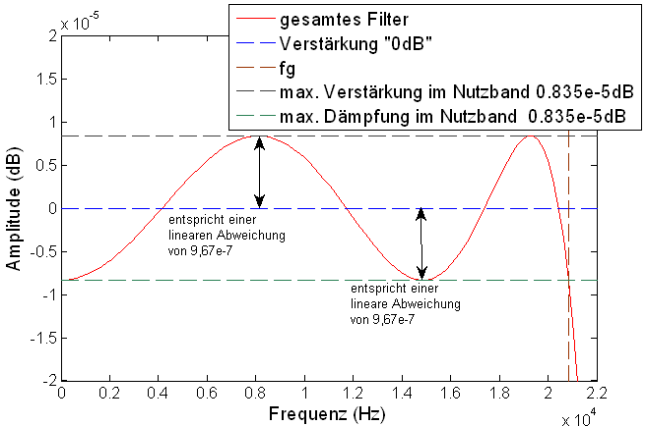


Bild 12: Vergrößerte Darstellung der Übertragungsfunktion des Interpolated FIR-Filters im Nutzband

Seine Aufgabe ist es lediglich, die periodischen Wiederholungen zu unterdrücken. Die daraus resultierende Übertragungsfunktion zeigt Bild 11. Die Übertragungsfunktion im Nutzband weicht in einem Bereich von ca 10^{-6} von der 0 dB-Linie ab (Bild 12). Auch diese Abweichung hat keine negativen Auswirkungen auf das Auflösungsvermögen des Wandlers.

4.4. Simulation der Filter

Das Filtermodell lässt sich innerhalb von Matlab nicht nur mittels darin vorhandener Signalquellen simulieren, sondern auch mit externen Daten. Damit ist es möglich, eine Simulation des Filters mit den Simulationsergebnissen des Modulators aus Spectre zu verifizieren. Dazu wurde das in Abschnitt 3 besprochene Simulationsbeispiel extrahiert und dem Simulink-Modell zugeführt. Nach Ablauf der Simulation kann das Ergebnis mit einem Scope angezeigt werden [14]. Damit konnte die Funktionalität des entworfenen Designs verifiziert werden.

5. VHDL-Codegenerierung

Die zeitintensive Arbeit vom Designentwurf bis zu einer Hardwareimplementierung lässt sich mit dem Programm Simulink HDL-Coder effektiv beschleunigen. Die Beschreibung auf RTL-Level in VHDL oder Verilog wird automatisch erzeugt. Der generierte VHDL-Code entspricht dem Aufbau des Simulink-Modells in einer Mischung aus Verhaltens- und Strukturbeschreibungen. Entsprechend ist der Code klar gegliedert, so dass eine Nachbearbeitung möglich ist. Davon wurde im vorliegenden Fall jedoch kein Gebrauch gemacht.

Neben dem Code zur Schaltungsbeschreibung ist der HDL-Coder auch in der Lage, die Testbench entsprechend der in Simulink eingesetzten Stimuligeneration und der Kontrolle der Ausgangssignale automatisch zu erzeugen. Darüber hinaus lassen sich vollständige Simulationsskripte erzeugen. Diese wurden genutzt, um die gesamte Testbench auf bequeme Weise in ModelSim (Mentor Graphics) zu verifizieren.

Der generierte Code nach IEEE-Standard ist vollständig synthesefähig. Auch für die Schaltungssynthese lässt sich mit HDL Coder ein Skript erzeugen. Dies führt zu einer erheblichen Zeitersparnis, da sich die Bearbeitung stets auf das Simulink-Modell konzentrieren kann. Im vorliegenden Fall wurde die Zeitersparnis dazu genutzt, zusätzliche Entwurfssicherheit über einen Rapid-Prototyping-Schritt zu gewinnen. Hierzu wurde das Design auf ein FPGA heruntergeladen und als Hardware getestet.

6. Layoutentwurf

6.1. Digitale Schaltungssynthese

Für die Synthese des VHDL-Codes und das Mapping auf die Zieltechnologie wurde das Programm Design Vision von Synopsys verwendet. Erwähnenswert ist, dass keinerlei Fehlermeldungen bezüglich der Codebeschreibung aufgetreten sind. Design Vision bietet zahlreiche Optimierungsmöglichkeiten hinsichtlich der Fläche oder auch der Schaltungsumgebung, wovon bei dieser Realisierung allerdings kein Gebrauch gemacht wurde. Es wurden vor allem zeitliche Vorgaben gesetzt, um eine einwandfreie Funktionsweise zu garantieren. Aufgrund der im Abschnitt 4 erläuterten stufenweisen Dezimation ergab sich ein Multifrequency-Design, das in der Parametrisierung berücksichtigt wurde. Der Synthesevorgang erreichte einen positiven Slackwert. Strengere Vorgaben bezüglich der Flächenausnutzung zeigten unmittelbar negative Einflüsse auf das Zeitverhalten des Designs.

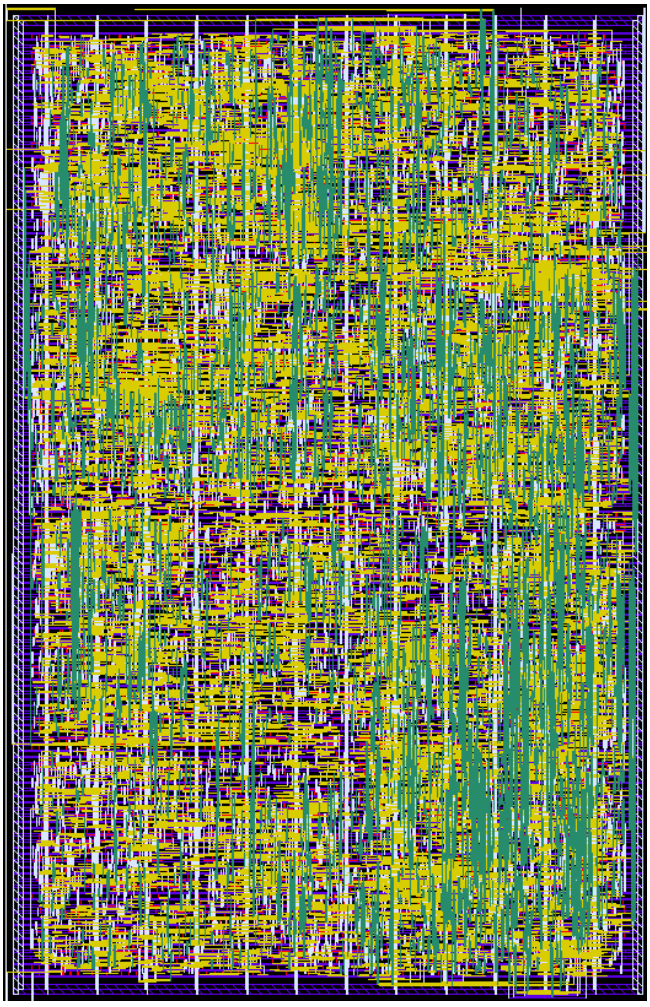


Bild 13: Layout des digitalen Filters

6.2. Digitallayout

Für den Place & Route-Prozess kam das Programm First Encounter von Cadence zu Einsatz. Durch die Kompatibilität zu Design Vision können bereits definierte Design Constraints wiederverwendet werden. Bei der Platzierung sowie dem physikalischen Routen wurden zeitliche Vorgaben gesetzt und vom Programm berücksichtigt. Das Layout kann mithilfe dieses Programms weitestgehend automatisiert erstellt werden (Bild 13). Auf eine zusätzliche Taktbaum-Synthese in Encounter wurde bei dieser Umsetzung verzichtet, da der erreichte Clock Skew nach einer zeitlichen Optimierung einen positiven Wert erreicht und somit akzeptabel ist. Ausgewählte Ergebnisse des Layoutentwurfs sind in Tabelle 4 dargestellt. Für den weiteren Prozess wurde das Layout in GDS II sowie die von Design Vision erzeugte Netzliste in Verilog exportiert.

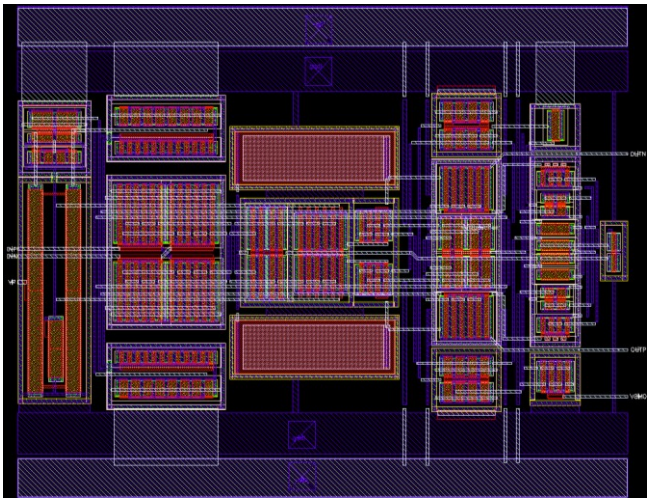


Bild 14: Layout des Operationsverstärkers

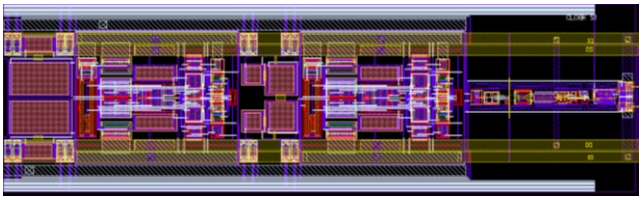


Bild 15: Layout des Modulators

Tabelle 4: Ergebnisse des digitalen Layouts

Core Size	$2.266 \cdot 10^6\ \mu m^2$
Chip Size (Total Area)	$2.587 \cdot 10^6\ \mu m^2$
Clock Skew	$15.745\ ns$
Anzahl Zellen	24011
IO Pins	17

6.3. Analoglayout

Der Entwurf der Analogzellen erfolgte in Virtuoso XL von Cadence als Full Custom Layout. Die wichtigste Zelle hierbei ist der Operationsverstärker. Besondere Sorgfalt wurde darauf verwendet, ein möglichst achsensymmetrisches Layout zu erzeugen. Bild 14 zeigt das Layout mit seiner horizontalen Symmetrieachse.

Hierzu wurden Transistoren teilweise doppelt ausgeführt und an offset-kritischen Stellen als Kleeblatt-Struktur abgebildet. Ebenfalls sind die Versorgungsleitungen doppelt ausgeführt. Mit diesem Layout wird erwartet, dass Fertigungstoleranzen, Temperaturgradienten und Bulk-Noise nur geringen Einfluss auf das Nutzsignal haben werden.

Mit einer abschließenden Post-Layout-Simulation unter Berücksichtigung der parasitären Elemente konnte

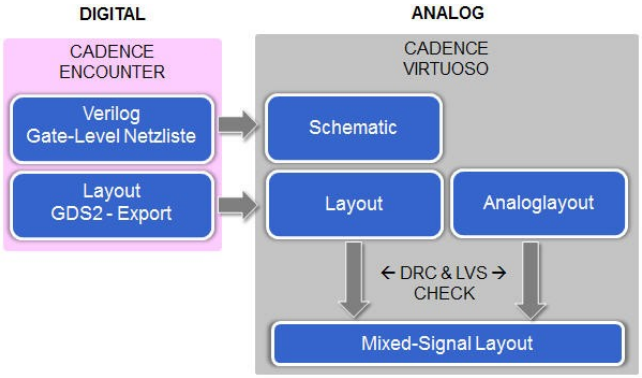


Bild 16: Entwurfsablauf für das Mixed-Signal-Layout

die korrekte Funktionsweise des Verstärkers bestätigt werden. Ausgewählte Ergebnisse [14] der Simulation sind in Tabelle 5 aufgelistet.

Mit gleicher Sorgfalt wurde auf Symmetrie im Layout des Komparators, in der Anordnung der Transmission Gates, der Kondensatoren und des gesamten Modulators geachtet (Bild 15). Damit zieht sich die Symmetrielinie durch den gesamten Modulator. Die Verifikation des Analoglayouts erfolgte mit dem Cadence-Programm ASSURA für den DRC (Design Rule Check) sowie den LVS (Layout-versus-Schematic).

Tabelle 5: Post-Layout-Simulationsergebnisse des Operationsverstärkers

Leerlaufverstärkung	$V_o = 81.2\ dB$
Transitfrequenz	$f_T = 152.5\ MHz$
Stromaufnahme	$I_{DD} = 498\ \mu A$

6.4. Design Import

Um das gesamte Mixed-Signal-Chiplayout mit Hilfe des Full Custom-Layoutprogramms Virtuoso XL von Cadence fertig stellen zu können, muss das von Encounter erzeugte Layout des digitalen Filters in Virtuoso eingelesen werden. Als Schnittstelle hierzu dienen die zuvor erstellte GDS II-Datei sowie die Verilog-Netzliste (Bild 16). Problemlos lassen sich die beiden Formate importieren. Man erhält daraus zwei Views (Layout und Schematic). Damit kann das in Virtuoso übertragene Digitaldesign auf etwaige DRC- bzw. LVS-Fehler geprüft und als Makrozelle abgelegt werden. Nach dem fehlerfreien Import des Digital-Layouts können die Analogteile hinzugefügt und das Mixed-Signal-Layout vervollständigt werden.

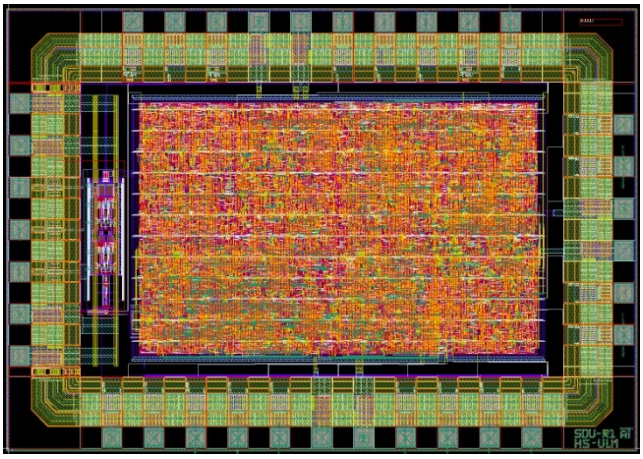


Bild 17: Layout des Testchips

Im Bild 17 ist der gesamte Testchip dargestellt. Das Chipmaß beträgt 2,10 * 3,00 mm^2. Im linken Bereich des Testchips ist der analoge Modulator (senkrecht) angeordnet. Die Größe des Testchips ist Core-Limited und im Wesentlichen durch die Layoutgröße des digitalen Filters bestimmt. Digitale und analoge Versorgungen sind bis in das Chiplayout hinein getrennt. Für das Packaging ist ein 48-poliges Plastikgehäuse vorgesehen. Dies ermöglicht den Einsatz zusätzlicher Signale für spätere messtechnische Zwecke.

Die Fläche des digitalen Filters könnte durch die Umstellung auf eine aktuellere Technologie, beispielsweise eine 90 nm-Technologie, um mehr als den Faktor 10 reduziert werden. Der Aufwand für die Entwicklung dafür wäre gering, da diese weitestgehend automatisiert erfolgen kann.

7. Zusammenfassung

Der Entwurf eines Testchips für einen Sigma-Delta-A/D-Wandler wurde vorgestellt. Bei einer Taktfrequenz von 20 MHz wird eine Auflösung von 12 Bit bis zu einer Signalbandbreite von 20 kHz erwartet. Wichtigste Komponenten des Chips sind der analoge Modulator und das digitale Filter. Der zweistufige Modulator in Switched-Capacitor-Technik beinhaltet zwei vollsymmetrische Operationsverstärker, einen Komparator sowie die Switched-Capacitor-Beschaltung mit entsprechender Taktansteuerung. Das digitale Filter ist in drei Stufen aufgebaut, einem Si-Filter, einem Tiefpassfilter und einem zusätzlichen Kompensationsfilter.

Mit diesem Projekt wurde ein vollständiger Design Flow für ein Mixed-Signal-IC auf der Basis der Cadence Design Tools erarbeitet. Der Entwurf der Analogteile erfolgte als Full Custom Design. Der Digitalteil wurde auf dem Wege einer High-Level-Synthese mit Matlab Simulink unmittelbar auf Systemebene entworfen. Der VHDL-Code wurde daraus vollautomatisch mit HDL Coder erzeugt und in ModelSim von Mentor Graphics verifiziert. Die Schaltungssynthese erfolgte mit Design Vision von Synopsys und die Layoutsynthese mit First Encounter. Die Fertigstellung des Layouts einschließlich DRC und LVS wurde mit Virtuoso XL durchgeführt. Zur Verifikation des Digitalteils wurden u.a. Modulator-Ausgangssignale mit Hilfe des Analogsimulators Spectre erzeugt. Der Digitalteil wurde darüber hinaus mit Hilfe eines Downloads auf ein FPGA verifiziert. Damit ist ein Weg gefunden, Mixed-Signal-ICs, auch mit komplexem Digitalanteil, zuverlässig und in begrenzter Entwicklungszeit zu realisieren.

8. Literatur

[1] A. Pervez, H. V. Sorensen, J. van der Spiegel, „An Overview of Sigma-Delta Converters“, IEEE Signal Processing Magazine, 01/1996

[2] J. Baker, “CMOS-Mixed-Signal Circuit Design”, IEEE Press Series on Microelectronic Systems, 2002 (ISBN: 0471227544)

[3] J. R. Ohm, H. D. Lüke, „Signalübertragung“, Springer Verlag, 2007 (ISBN: 3540587535)

[4] F. Maloberti, „Data Converters“, Springer Verlag, 2007 (ISBN: 0387324852)

[5] B. Brandt, D. Wingard, B. Wooley, „Second-Order Sigma-Delta Modulation for Digital-Audio Signal Acquisition“, IEEE Journal of Solid-State Circuits, 04/1991

[6] M. Ortmanns, F. Gerfers, Y. Manoli, „A Continuous-Time Sigma-Delta Modulator with Switched Capacitor Controlled Current Mode Feedback“, Institute for Microelectronics, Albert-Ludwigs-University Freiburg

[7] A. Leuciuc, C. Mitrea, „On the Effect of Op-Amp Finite Gain in Delta-Sigma Modulators“, IEEE International Symposium on Circuits and Systems, May 28-31, 2000, Geneva, Switzerland

[8] G. Vallant, G. Forster, „Designstudie und Testchipentwurf für ein integriertes Laser-Radar (LIDAR)“, Workshop der Multiprojekt-Chip-Gruppe, Konstanz 2008, (ISSN 1862-7102)

[9] J. Baker, „CMOS Circuit Design, Layout, and Simulation“, Second Edition, John Wiley & Sons, Inc. (ISBN: 0470229411)

[10] T. Burkhart, H. Brodka, „Entwicklungsstudie für einen Sigma-Delta-A/D-Umsetzer“, Hochschule Ulm, 2007

[11] F. Nagler, J. Wiest. „Mixed-Signal-Simulation mit MATLAB Simulink und automatischer

VHDL-Code-Erzeugung“, Hochschule Ulm, 2007

[12] T. Hartmann, R. Ritter. „Entwicklungsstudie für einen Sigma-Delta-A/D-Umsetzer“, Hochschule Ulm, 2008

[13] J. Hoffmann, F. Quint, „Signalverarbeitung mit MATLAB und Simulink“, Oldenbourg Wissenschaftsverlag, 2007 (ISBN: 3486584278)

[14] A. Trutin, „Testchipentwurf für einen Sigma-Delta-A/D-Wandler“, Hochschule Ulm, 2009

Danksagung

Diese Arbeit wäre nicht ohne eine Reihe von Vorarbeiten durch Studierende der Fakultät Elektrotechnik und Informationstechnik in der kurzen Zeit möglich gewesen. Die Autoren bedanken sich bei den Herren F. Nagler und J. Wiest für ihre Vorarbeiten zur High-Level-Synthese, bei den Herren T. Burkhart und H. Brodka für Simulationsarbeiten am Modulator und bei den Herren T. Hartmann und R. Ritter für ihre Arbeiten zum Filter-Systementwurf. Weiterer Dank gilt den Herren F. Mrugalla, G. Vallant und J. Schäfer für die wertvollen Hinweise sowie die großzügige Unterstützung.

Das Projekt wurde gefördert aus Mitteln der baden-württembergischen Multiprojekt-Chip-Gruppe.

IP-Core zur Ansteuerung eines CCD Sensors

Dipl.-Ing.(FH) Christoph Bayer

Hochschule Pforzheim, Tiefenbronner Straße 65, 75175 Pforzheim

IDS Imaging Development Systems GmbH & Co., Dimbacher Straße 6-8, 74182 Obersulm.

christoph@cbayer.info

In dieser Arbeit wird die schrittweise Erarbeitung einer konfigurierbaren Ansteuerung für verschiedenste CCD Sensoren beschrieben. Über die physikalischen Eigenschaften der Charge-Coupled Devices (Abkürzung: CCD, dt.: Ladungsgekoppelte Bauelemente) wird an die Komplexität des entwickelten Systems herangeführt. Hieraus werden die Systemanforderungen für die Ansteuerung hergeleitet.

Kern der Entwicklung bildet ein, auf einem FPGA integrierter, IP-Core (im folgenden auch Timingcore), zur Ansteuerung der vertikalen Taktsignale des CCD-Sensors. Dieser Timingcore ist zudem für die Bildsignalverarbeitung und Synchronisation des Systems zuständig. A/D Wandlung des CCD-Ausgangssignals und Ansteuerung der horizontalen Taktsignale werden mittels eines speziell für die Videosignalverarbeitung entworfenen Baustein von Analog Device vorgenommen.

1. Motivation

Bestehende Systeme zur Ansteuerungen von CCD-Sensoren sind meist in ihren Anwendungsgebieten oder ihrer maximalen Taktfrequenz begrenzt.

Der konfigurierbare A/D Wandler AD9895 von Analog Devices, welcher speziell zur komplexen Ansteuerung von CCD-Sensoren entwickelt worden ist, unterstützt beispielsweise nur Pixeltakte bis zu 30 MHz bei einer maximalen Sensorgröße von 4096x4096 Pixel und eine Auflösung von 12 Bit. Diese Eckdaten entsprechen einer Bildwiderholrate von 6,25 Bildern pro Sekunde bei einer Auflösung von fünf Millionen Pixeln. Sensoren dieser Größe, wie der ICX625 von Sony, unterstützen bereits einen Pixeltakt von 60 MHz, was einer Bildwiederholrate von zwölf Bildern pro Sekunde entspricht.

In den meisten existierenden Systemen findet keine Bilddatenvorverarbeitung statt. Diese Vorverarbeitung kann zur Komprimierung der übertragenen Datenmenge genutzt werden. Fortführend könnte hier schon Bildverarbeitung wie Subsampling oder Binning stattfinden.

Aus diesen Restriktionen heraus ist die Motivation entstanden, ein System zu entwickeln, welches mit höheren Pixeltakten eine große Anzahl unterschiedlichster Sensoren unterstützt und dabei auf die Möglichkeiten diverser Bildvorverarbeitungsmethoden zurückgreifen kann.

2. CCD Grundlagen

Eine der ersten Veröffentlichungen zu CCD Sensoren wurde im Jahre 1970 von den „Bell Laboratories“, unter den Namen der Erfinder W. S. Boyle und G. E. Smith präsentiert. Die dort erforschten Sensoren sollten zum Zweck der Videotelefonie eingesetzt werden.

Die ersten CCD-Sensoren wurden im Jahre 1980 zur Serienreife geführt und beispielsweise in allen Boeing 747 der Fluggesellschaft All Nippon Airways (ANA) verbaut. Der Preis dieses Sensors lag damals bei rund 2000$. Im Vergleich hierzu liegt der Preis von CCD-Sensoren heute im Bereich zwischen 10 und 500$.

2.1. Vom Photon zur Spannung

Die in Abbildung 1 dargestellten Illustration zeigt bildhaft die vier der CCD-Technologie zu Grunde liegenden Schritte der Bildaufnahme, die technisch betrachtet im Wesentlichen aus einer Wandlung von Licht in Spannungswerte besteht.

In Abbildung 1 ist die Messung des Niederschlags auf einer bestimmten Fläche dargestellt. Einige der Regentropfen werden in Eimern gesammelt. Die gefüllten Eimer werden auf Förderbändern erst nach rechts (zeilenweise) und dann nach unten (spaltenweise) zur Messstation transportiert, um dort ausgewertet zu werden.

Ein CCD-Sensor funktioniert analog hierzu wie folgt:

Die Photonen des einfallenden Lichts heben, auf Grund der übertragenen Strahlungsenergie, im Halbleiter Elektronen aus dem Valenz- in das Leitungsband.

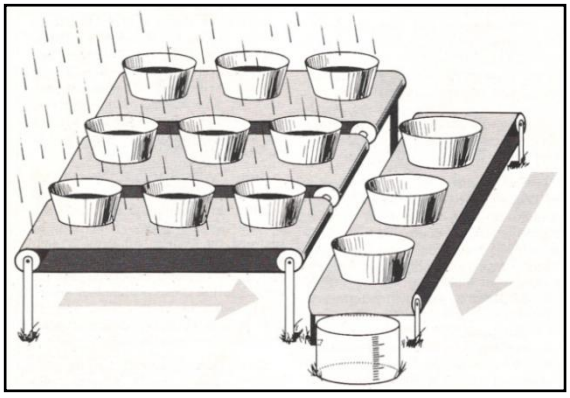


Abbildung 1 Prinzipielle Funktion eines CCD-Sensors nach (James Janesick, 1987)

Nach Ablauf der Belichtungszeit werden die Elektronen in Verarmungszonen (Eimern) unterhalb der aktiven Pixel verschoben. Je größer die Verarmungszonen ausgebildet sind, um so mehr Ladungen können gesammelt werden und um so empfindlicher ist der Sensor.

Durch Anlegen einer Spannung an entsprechend positionierte Gate-Elektroden (vgl. Feldeffekttransistor) werden die darunter liegenden Verarmungszonen (Potentialtöpfe) verformt. Durch geschicktes Beeinflussen dieser Potentialtöpfe können die Ladungen unterhalb der aktiven Pixel vertikal und horizontal zur „Messstation“ hin transportiert werden (Förderbänder).

In einem abschließenden Schritt werden die Ladungspakte an der „Messstation“ in eine analoge Spannung konvertiert. Dies erfolgt mittels einer „floating diode“- oder „floating diffusion“-Treiberstufe, welche eine Spannung proportional zur Größe der anliegenden Ladung generiert.

Zur korrekten Ansteuerung eines CCD-Sensors müssen Takte zur Ladungsintegration erzeugt werden; außerdem benötigt ein CCD-Sensor-Timing verschiede Takte, die den horizontalen und vertikalen Ladungstransport steuern.

2.2. Der Ladungstransport

Der Ladungstransport stellt einen essentiellen und hoch zeitkritischen Bestandteil der Ansteuerung eines Sensors darstellt dar.

Wie bereits im vorherigen Abschnitt erwähnt, kann durch das Anlegen einer Spannung an eine Gate-Elektrode die darunterliegende Verarmungszone verformt werden. Durch diese Spannungen werden sogenannte Potentialtöpfe ausgebildet. Die Elektronen sammeln sich an dem Punkt mit dem höchsten Potential. Somit sind die negativen Ladungsträger bestrebt, unter das Gate an welchem die höchste Spannung angelegt ist, zu strömen, hier ist der tiefste Potentialtopf ausgebildet. Abbildung 2 stellt anschaulich den Transport von Elektronen von einem Gate zum nächsten durch Verformung der Potentialtöpfe dar.

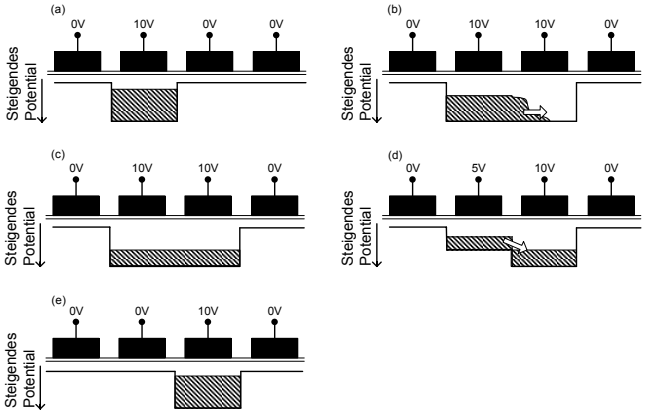


Abbildung 2 Prinzipieller Transport eines Ladungspakets von der Verarmungszone eines Gates zur nächste nach (Theuwissen, 1996)

In Teilbild (a) befinden sich die Elektronen unter dem zweiten Gate, an welchem die höchste Spannung angelegt ist. Durch Anlegen von Spannung an das dritte Gate wird der Potentialtopf breiter ausgebildet. Die Ladungen verteilen sich auf Grund des Bestrebens, das höchste Potential zu erreichen, (Teilbild (c)) gleichmäßig im gesamten Potentialtopf.

Wird die Spannung am zweiten Gate reduziert (d), fließen die Ladungen unter das dritte Gate, unter welcher sich in Teilbild (e) die Gesamtheit der Ladungen befindet.

Die vorausgehende Betrachtung bezieht sich auf Gates, von denen sich mehrere unter einem Pixel befinden können. Gewöhnlich liegt die Anzahl, je nach Ausführung des Sensors, zwischen einem und vier Gates pro Pixel wobei auch Realisierungen mit acht und mehr Gates existieren.

Abbildung 3 stellt ein Transportsystem mit vier Gates pro Pixel am Beispiel zwei nebeneinander liegender Pixel dar.

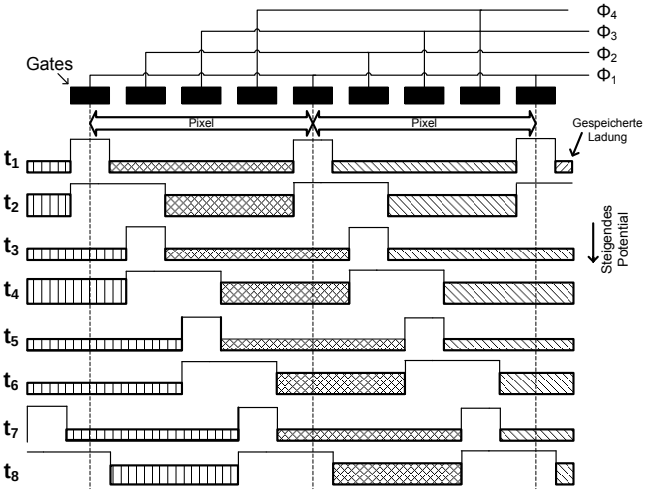


Abbildung 3 Vier Phasen Transportsystem nach (Theuwissen, 1996)

Durch geschicktes Schalten dieser Takte werden die gespeicherten Ladungen, welche hier unterschiedlich schraffiert dargestellt sind, von links nach rechts unter das nächste Pixel transportiert. Wie in der obigen Abbildung deutlich gezeigt wird, sind in einem vier-Phasen Transportsystem sechs Zyklen notwendig, um einen Transport vom einem zum nächsten Pixel auszuführen.

Folglich vereinfacht sich die Ansteuerung, wenn der Sensor mit weniger Elektroden pro Pixel ausgestattet ist. Hierdurch werden weniger Taktpulse und auch weniger schnelle Pulse benötigt. Je weniger Elektroden verbaut sind, um so kleiner werden die Bereiche, in welchen Ladungen gespeichert werden können. Kann in einem vier-Phasen System noch während der Zeit t_2, in welcher die wenigsten Elektroden aktiv sind (zwei pro Pixel), unter der Hälfte des Pixels Ladung gespeichert werden, so kann in einem drei Phasen System nur mehr unter einem Drittel des Pixels Ladung gespeichert werden.

2.3. CCD-Sensoraufbau

Aus fertigungstechnischen Gründen besteht ein CCD-Sensor aus drei grundsätzlich verschiedenen Pixel:

- Belanglose „dummy“-Pixel
- Optisch schwarze Pixel
- Aktive, lichtempfindliche Pixel

Diese sind in unterschiedlicher Anzahl auf dem Sensor nach dem Prinzip - wie in Abbildung 4 - dargestellt angeordnet. Hieraus entsteht sowohl im horizontalen, als auch im vertikalen Bereich eine Einteilung in jeweils fünf unterschiedlich große Regionen.

Von diesen drei Pixel Kategorien dienen nur die beiden letzten zur Erzeugung des Bildes. Der Wert der optisch schwarzen Pixel wird je nach Anwendung einmal pro Bild oder Zeile erfasst (geklemmt, engl.: clamp). Er dient zur Schwarzwertberechnung, welche zur Normierung der Werte der aktiven Pixel herangezogen wird.

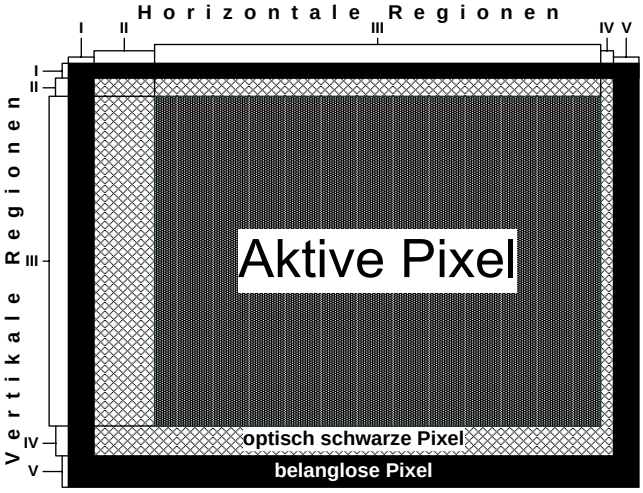


Abbildung 4 Einteilung eins CCD-Sensors in verschieden Regionen mit aktiven, optisch schwarzen und belanglosen „dummy“ Bildpunkten.

Damit das dem aktuellen Bild folgenden Bild keine Ladungen des aktuellen Bildes enthält, müssen alle Pixel eines Sensors ausgelesen werden, bevor eine neue Belichtung durch die Ladungsintegration beendet werden kann (Prinzip der Eimer-Leerung bevor neu gemessen werden kann). Folglich ist es naheliegend, für das Bild unnötige Pixelinhalte zeilenweise „wegzukippen“. Dies ist durch mehrmaliges vertikales Schieben möglich, was einem ausschließlichen Betreiben der vertikalen Förderbänder entspricht. Diese schnellere Auslesevariante impliziert ein anderes Pulsmuster, was vom Sensor als auch von der Regionengröße sehr stark abhängt. Wird nun der Bildausschnitt verändert bzw. verkleinert, so muss auch die zeitliche Abfolge der Schiebepulse angepasst werden.

Abschließend betrachtet liegt auf der Hand, dass ein universelles System zur adaptiven Ansteuerung verschiedenster CCD-Sensoren konform zu unterschiedlichsten Timing-Varianten konfigurierbar sein muss. Weiterhin müssen diese Varianten flankensynchron umschaltbar sein - während des Auslesens einer aktiven Region dürfen die Einstellungen trotz möglicher Konfigurationsvorgänge nicht verändert werden, da dies zu Bildverzerrungen führen würde.

3. Systemanforderungen und Entwurf

Die vorangegangenen Betrachtungen des CCD-Sensors zeigen auf, dass umfangreiche Systemanforderungen und Sensorspezifische Randbedingung zur Ansteuerung eingehalten werden müssen. Die Systemanforderungen sind:

- Unterstützung von Sensoren mit bis zu 8192x8192 Pixel.
- frei einstellbarer Pixeltakt bis zu 65 MHz.
- Generierung horizontaler und vertikaler Schiebetakte welche in Lage, Größe und Polarität konfigurierbar sind.
- Schnelles Umschalten zwischen verschiedenen AOI-, Trigger und anderen Betriebsmodi
- „Correlated double Sampling“ (CDS) A/D Wandlung mit einer Auflösung von 14 Bit. CDS wird mit zwei, innerhalb eines Pixeltaktes verschiebbaren, Abtastzeitpunkte realisiert.
- Parallelisieren der A/D gewandelten Daten und zugehörige Synchronisierung mit dem darüber liegenden System.
- Konfiguration des Systems über eine serielle Schnittstelle mit rücklesbaren Registerinhalten.
- Mögliche Bilddatenvorverarbeitung wie Subsampling oder Binning.

Auf Grund der geforderten Pixelfrequenzen (bis zu 65 MHz) und dem notwendigen Einsatz eines A/D Wandlers mit CDS-Abtastung wird der Einsatz eines CCD-Signalprozessors mit integrierter Ansteuerung der horizontalen Schiebepulse notwendig.

3.1. Horizontaler Taktgenerator + ADC

Der hier verwendete CCD-Signalprozessor wurde speziell für Ansteuerungen in Hochgeschwindigkeitskameras entwickelt. Er bietet eine maximale Pixelfrequenz von 65 MHz bei einer Auflösung von 8192x8192 Pixeln nebst CDS-A/D-Wandlung mit 14 Bit und entspricht somit exakt den Vorgaben. Durch die integrierte drei-Draht-Schnittstelle ist das Verhalten des Signalprozessors voll konfigurierbar und somit optimal an jeden Sensor anpassbar.

Es handelt sich jedoch um eine unidirektionale Schnittstelle, womit einer der Voraussetzungen nicht entsprochen wird. Die digitalisierten Bilddaten werden nur quasi-parallel über zwei LVDS Datenleitungen ausgegeben, eine Möglichkeit zur Bilddatenvorverarbeitung ist nicht vorgesehen.

Der hier behandelte Timingcore ergänzt den verwendeten CCD-Signalprozessor um die fehlenden Eigenschaften in Form einer FPGA-basierten IP-Core-Realisierung.

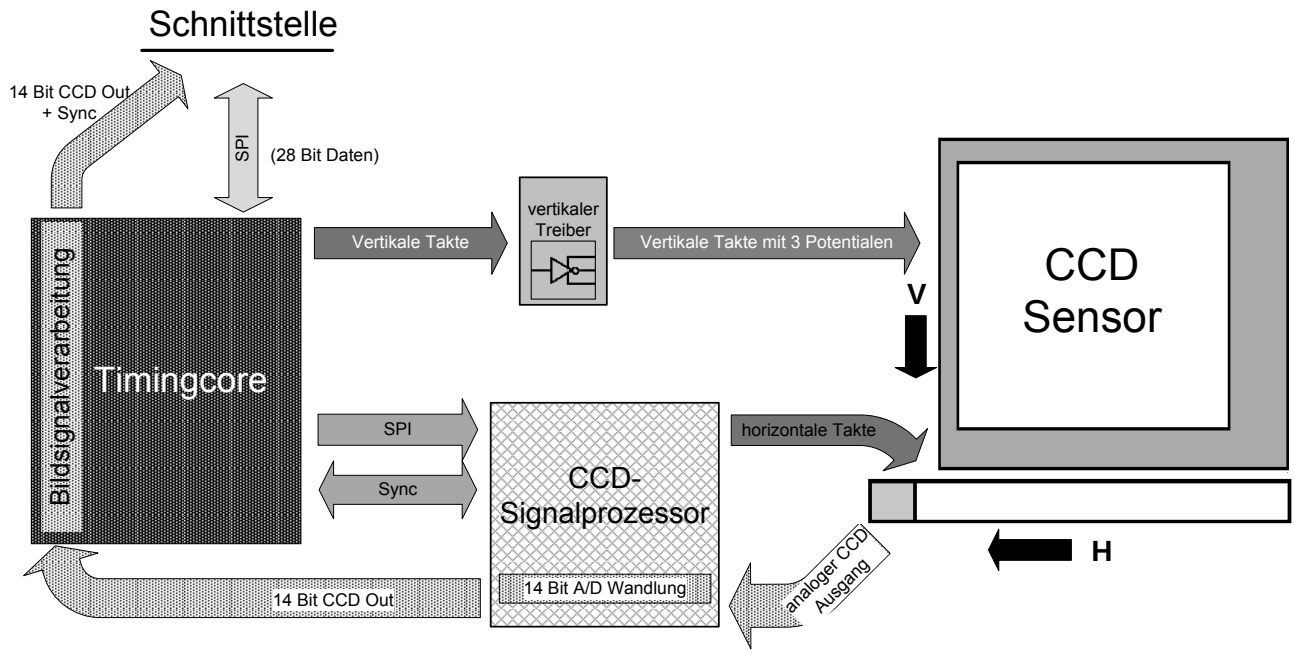


Abbildung 5: Funktionelles Blockschaltbild des entwickelten Systems zur Ansteuerung eines CCD-Sensors

4. Timingcore

Der entwickelte Core soll dazu dienen, alle benötigten vertikalen Takte zu generieren. Diese sind im speziellen:

- Ein Signal zum Löschen der angesammelten Ladung aus dem Sensor (Belichtungsstart).
- Signale zur Steuerung des vertikalen Sensorgates, um die Belichtung zu beenden. (Zur Erinnerung: hierdurch werden die Ladungen in das Transportsystem verschoben).
- Bis zu vier vertikale Schiebetakte.

Abbildung 6 Bereich „1“ zeigt die Generierung dieser Signale sowie die Synchronisierung des Timingcores mit dem AD9970 über Frame- und Linevalid-Signale, sowie die zugehörigen Konfigurationsregister.

Weiterhin ist der Timingcore für die Konfiguration des AD9970 via der geforderten drei-Draht Schnittstelle zuständig. Dies ermöglicht eine Speicherung der AD9970-Registerinhalte, um diese für den Systemmaster über die vier-Draht Schnittstelle auslesbar zu machen (Teil „4“ in Abbildung 6).

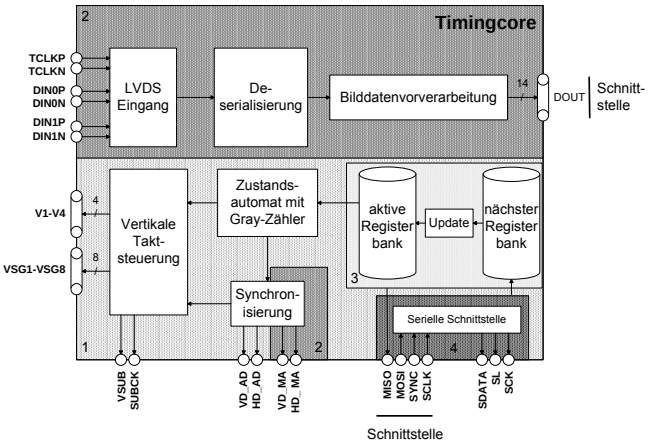


Abbildung 6 Blockschaltbild des entwickelten Timingcores zur Generierung des vertikalen Timings eines CCD-Sensors

Die Bilddatenvorverarbeitung, im zweiten Bereich der Abbildung, dient zur Deserialisierung der vom CCD-Signalprozessor erhaltenen Daten vorzunehmen. Zusätzlich ist in diesem Datenverarbeitungspfad weitere Bilddatenvorverarbeitung (Binning oder Subsampling) denkbar.

4.1. Befehlssatzarchitektur

Die Befehlssatzarchitektur (kurz: ISA, engl.: Instruction set Architecture) des Timingcores beschreibt im Wesentlichen die Konfigurationsschnittstelle, um ein komplettes Sensortiming einzurichten. Diese ist durch die serielle vier Draht Schnittstelle, welche über 1000 Register verwaltet, gegeben.

4.1.1 Serielle Schnittstelle

Die serielle Schnittstelle überträgt pro Schreib- bzw. Lesevorgang 28 Bit, welche mit 12 Bit adressiert werden; dieses Format wurde aus der Vorgabe des CCD-Signalprozessors übernommen. So ist die Anforderung einer Konfiguration und Steuerung des CCD-Signalprozessors durch den Timingcore erfüllt.

Um die Ansteuerung eines Sensors zu konfigurieren, sind meist eine Vielzahl an Schreibzugriffen notwendig. Diese erfolgen mit hoher Wahrscheinlichkeit an sukzessiv aufeinanderfolgenden Adressen. Ein Burst-Modus ermöglicht das Beschreiben aufeinanderfolgender Register ab einer anzugebenden Startadresse. Ein 3-Bit-Tag im Datenpaket klassifiziert übertragende Daten nach folgenden Gesichtspunkten:

- Frame ist gültig: frame valid = fv
- Burstmode aktiv: bm
- Differenzierung zwischen Schreib- und Leseanforderung (MOSI)
- Timingcore ist momentan nicht bereit Daten zu empfangen: iv = invalid (MISO)

Die sich ergebenden Schreib- bzw. Lesevorgänge (MOSI: Master out, Slave in/MISO: Master in Slave out)sind in Tabelle 1 und Tabelle 2 dargestellt:

Tabelle 1: MOSI Frameaufbau

0	Tag	2	3	Adressen	14	15	Daten	43
0	1	2	LSB	…	MSB	LSB	…	MSB
fv	bm	r/w	A0	…	A11	D0	…	D27

Tabelle 2 MISO Frameaufbau

0	Tag	2	3	Adressen	14	15	Daten	43
0	1	2	LSB	…	MSB	LSB	…	MSB
fv	bm	iv	A0	…	A11	D0	…	D27

Im Falle einer eingeleiteten Burst-Mode-Übertragung verkürzt sich der Frameaufbau um die Adressangabe.

4.1.2 Registerbank

Die Betrachtungen zur Architektur eines CCD Sensors aus Kapitel 2.3 haben Einfluss auf die Wahl der verwendeten Speicherarchitektur. Die in Abbildung 4 dargestellten Regionen sind meist unterschiedlich groß und benötigen in aller Regel unterschiedliche Abfolgen für die vertikalen Schiebepulse sowie für Lösch- und Sensorgate- Signale.

Zur Erinnerung: alle diese Signale sind mit verschiedenen Werten über Registerbänke konfigurierbar. Diese Register sind jedoch, wie im vorigen Unterpunkt beschrieben, nur über die serielle Schnittstelle be-

schreibbar. Dies impliziert, dass während des Beschreibens eines Registers metastabile Zustände auftreten können. So kann das Register während des Schreibvorgangs falsche Werte enthalten kann.

Da zur Definition der zeitlichen Abfolgen und der Größe einer Region bzw. eines Bildes mehrere Registereinstellungen notwendig sind, ist es notwendig, dass die betreffenden Registerinhalte zueinander konform sind. Dies bedeutet, dass die Einstellungen, die eine Region definieren, nicht geändert werden dürfen, solange diese Region ausgelesen wird. Die serielle Schnittstelle darf in ihrer Schreib- und Lesefunktion hierbei jedoch nicht beeinflusst werden.

Legt man die vorausgegangenen Betrachtungen zu Grunde, sollte jedes Register doppelt ausgeführt werden. Zu einem geeigneten Updatezeitpunkt kann zwischen den bestehenden und den neu konfigurierten Inhalten umgeschaltet werden. Die resultierende Speicherarchitektur ist in Abbildung 7 dargestellt.

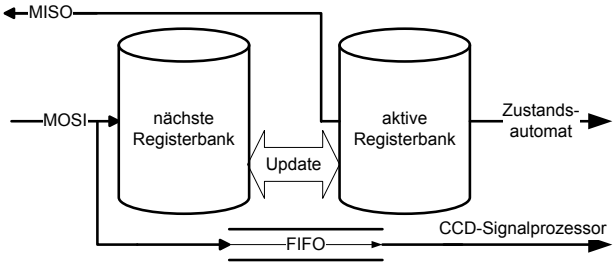


Abbildung 7 Funktionelle Anordnung der Aktiven und Schatten Register mit den schreibenden und lesenden Datenpfaden.

Die serielle Schnittstelle liest in dieser Architektur aus der aktiven Registerbank, während Schreibvorgänge in den inaktiven Speicherbereich erfolgen. Registerinhalte welche für den CCD-Signalprozessor bestimmt sind, werden zum einen in den Speicher des Timingcores geschrieben und über einen First in First out Speicher an den CCD-Signalprozessor weitergeleitet. So ist auch die Rücklesbarkeit der CCD-Signalprozessor-Register gewährleistet.

4.2. Vertikaler und horizontaler Zustandsautomat

Zur aktiven Steuerung des Sensors durch den Timingcore, mit der Ansteuerung des horizontalen Timings und der AD-Wandlung im CCD-Signalprozessor ist es notwendig mit diesem vollkommen synchron zu sein. Dies ist nur möglich, wenn diskrete Informationen über die genaue Position im auszulesenden Bild in Form von Zeilen- und Pixelwerten vorliegen.

Zur Positionsbestimmung sind zwei ineinander laufende Zähler implementiert. Diese sind mit einer Tiefe von 13 Bit für Sensoren mit einer maximalen Größe von 8192*8192 Pixel ausgelegt.

Die Bildqualität von CCD-Sensoren hängt maßgeblich von der Qualität der Versorgungsspannung ab, diese kann durch abrupte Lastwechsel enorm verschlechtert werden. Solche Lastwechsel können bei gewöhnlichen Binär-Zählern durch das gleichzeitige Umschalten mehrerer Bits auftreten. Aus diesem Grund sind die Zähler in Gray Kodierung ausgeführt.

Abhängig von der Bildposition werden die freigeschalteten Schiebeimpulse mittels Vergleicher getriggert. Die Verwendung Gray-codierter Zähler fordert hier erheblichen Mehraufwand.

4.3. Synchronisierung und Bilddatenübertragung

Der im verwendete CCD-Signalprozessor wandelt je nach Konfiguration -bis auf wenige Ausnahmenkonstant das analoge Ausgangssignal des CCD Sensors in ein digitales 14-Bit Signal. Das Ergebnis der Wandlung wird parallel über zwei LVDS Kanäle zu je sieben Bit pro Pixeltakt übertragen. Zur Übertragung an den Systemmaster wird dieses Ausgangssignal parallelisiert und kann so in einer geringeren Geschwindigkeit übertragen werden.

Durch Angabe des Bereiches mit den benötigten Bildinformationen ist es möglich, das Datenaufkommen zwischen dem Timingcore und dem darüber liegenden System weiter zu reduzieren. Folglich werden alle uninteressanten Pixelwerte vom Timingcore unterdrückt.

Die Synchronisation des Timingcores mit dem Systemmaster erfolgt anhand der Frame- und Linevalid-Signale.

5. Implementierung der Registerbank

Für einen CCD-Sensor können drei verschiedene Registerarten und dazugehörige Updatezeitpunkte festgelegt werden:

- Register, die keinen Einfluss auf die Ansteuerung des Sensors nehmen (beispielsweise Einstellungen der Verstärkung) können direkt nach Empfang über die serielle Schnittstelle übernommen werden.
- Register, die zur Definition einer Region beitragen und beispielsweise die Anzahl der vertikalen Pulse für eine Region definieren können zum Zeitpunkt eines Regionen- Wechsels übernommen werden.
- Einstellungen, die Einfluss auf das Verhalten des gesamten Bildes nehmen, (wie zum Beispiel die Anzahl der Regionen oder Zeilen pro

Bild) dürfen nur zum Zeitpunkt des Framewechsels übernommen werden.

Hieraus resultiert die Herausforderung, eine Ressourcenschonende Implementierung von ca. 1500 28-Bit-Registern zu entwerfen, welche zu verschiedenen Zeitpunkten - allerdings innerhalb eines Pixeltaktes - auf den aktuellen Stand gebracht werden können.

Auf Grund der geforderten Geschwindigkeit liegt nahe, jedes Register zweifach in FPGA-Logikelementen zu realisieren. Das SPI-Interface müsste dann, nach einer aufwendigen Adressdekodierung die Werte in das zugehörige Register schreiben. Tritt ein Updatezeitpunkt ein, so können innerhalb eines Zyklus die Werte vom passiven in den aktiven Registerbereich übernommen werden. Vergleichsoperationen für das Schalten von Signalen oder für die Steuerung des horizontalen und vertikalen Zählers können hier direkt auf den Registern ausgeführt werden. Berechnet man jedoch den Hardwareaufwand der Realisierung dieser Variation, so wird deutlich, dass dieser nicht in angemessener Relation zur gewünschten Funktion steht.

Da die Anzahl der eingebetteten Speicherblöcke im Verhältnis zu den Logikzellen in FPGAs in den letzten Jahren um ein Vielfaches gestiegen ist, ist ein Lösungsansatz unter Benutzung dieser Speicherblöcke plausibel. Der große Nachteil solcher RAM Blöcke liegt jedoch darin, dass der Inhalt nur seriell geschrieben oder gelesen werden kann. Um diese Tatsache mit den geforderten schnellen Updatekonzepten zu vereinbaren, ist es notwendig eine optimierte Speicherarchitektur zu entwerfen.

Im Folgenden werden drei Lösungsansätze vorgestellt und ihre Vor- sowie Nachteile beleuchtet.

5.1. Bitkodiertes Updateverfahren

In einem ersten Lösungsansatz wurden die Register in kleinstmögliche Gruppen eingeteilt und zu Registerbänken zusammengefasst. Das bedeutet hier, dass Register mit denselben Updatezeitpunkten gruppiert wurden. Hierbei ist darauf geachtet zu achten, dass beispielsweise alle Register, die zur Definition einer Region benötigt werden, in einer Speicherbank zusammengefasst sind. Dies soll bewirken, dass möglichst wenige Registerbänke aktualisiert werden müssen, da in aller Regel eine Konfiguration an einem Stück durchgeführt wird. Weiteres Entwurfsziel war, kleine Registerbänke zu entwerfen, um die Aktualisierungszeiten möglichst kurz zu halten. Bei einer solchen Einteilung ist jedoch wichtig, auf eine einfache Adressierungsmöglichkeit der zusammengefassten Blöcke zu achten, damit diese nicht zu viel Aufwand im seriellen Interface fordert.

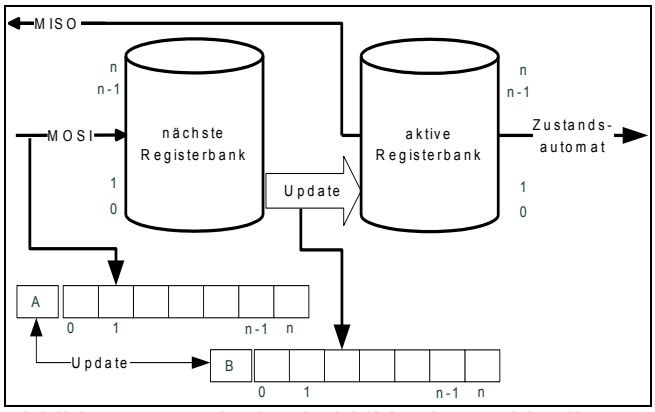


Abbildung 8 Blockschaltbild des Bitkodierten Speicher- Updateverfahrens

Für diese und die folgenden Realisierungen kommen im FPGA eingebettete „Dual Ported“ RAM Blöcke zum Einsatz. Zur Kennzeichnung welche Register zwischen zwei Aktualisierungszeitpunkten von der Seriellen Schnittstelle verändert wurden existiert für jedes Register ein Bit. Durch Auslesen dieser Bits wird in der Updateroutine entschieden welches Register aktualisiert werden muss. Dieser Aufbau führt zur folgenden zeitlichen Abfolge:

- Schreiben der inaktiven Register durch MOSI, dabei Kennzeichnung der veränderten Register.
- Lesen der aktiven Register durch den Zustandsautomat und MISO
- Tritt ein Updatezeitpunkt ein, so wird anhand der gesetzten Bits entschieden, welche Registerinhalte von der inaktiven zur aktiven Registerbank übertragen werden müssen. Hier ist eine Arbitrierung zwischen MISO- und Update- Zugriffen notwendig.

Durch die Verwendung der eingebetteten Speicherblöcke wird der Verbrauch der Logikelemente deutlich gesenkt. Zusätzlich muss jedoch für jedes Register ein Updatebit implementiert werden. Dieses muss, wie in der Abbildung gekennzeichnet doppelt ausgeführt werden und steigert somit merklich den Ressourcenverbrauch. Hinzu kommt, dass für die Arbitrierung zwischen MISO- und Updatezugriffen auf die aktive Registerbank weitere Logik aufgewendet werden muss.

Auch im zeitlichen Verhalten zeigt sich diese Implementierung als nicht optimal. Dies liegt daran, dass der Updatevorgang durch das Überprüfen jedes Bits mindestens so viele Takte benötigt, wie Register in einer Registerbank vorhanden sind. In dieser Zeit stehen auch den Zustandsautomaten keine gültigen Registerwerte zur Verfügung.

5.2. Adresskodiertes Updateverfahren

In einem nächsten Entwurf wurde auf die oben beschriebene Bitkodierung verzichtet. Zur Speicherung der Information, welche Registerinhalte zwischen den Aktualisierungszeitpunkten modifiziert wurden, wurde ein FIFO Speicher implementiert. Abbildung 9 zeigt das Blockschaltbild dieser Implementierung.

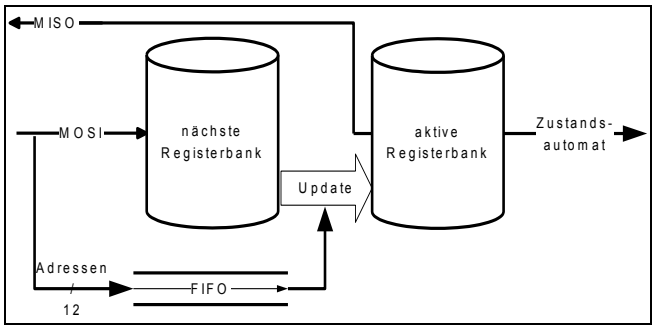


Abbildung 9 Blockschaltbild des Adresskodierten Speicher- Updateverfahrens

In diesem Speicher werden die Adressen der aktualisierten Register abgelegt. Tritt ein Updatezeitpunkt ein, so werden anhand dieser Adressen die betreffenden Register aktualisiert.

Da hier keine Statusbits überprüft werden müssen, ist die Dauer des Updateverfahrens nur von der Anzahl der modifizierten Adressen abhängig. Dies verkürzt die Zeit, in der keine gültigen Daten in der aktiven Registerbank vorliegen, um ein Wesentliches, jedoch entspricht auch diese Realisierung nicht den zeitlichen Vorgaben. Weiterhin besteht die Möglichkeit, dass Register zwischen den Updatezeitpunkten doppelt geschrieben werden. Dies führt dazu, dass diese auch doppelt aktualisiert werden, da ihre Adresse zweimal im FIFO gespeichert ist.

Zusammengefasst zeigt sich, dass diese Realisierung mit hoher Wahrscheinlichkeit kürzere Updatezeiten bietet. Auch der Ressourcenverbrauch im Bereich der Logikelemente wurde durch die Verwendung des FIFOs, welcher in eingebetteten Speicherblöcken realisiert wurde, optimiert. Von einer optimalen Realisierung ist dieser Entwurf durch die nicht eingehaltenen zeitlichen Randbedingungen jedoch weit entfernt.

5.3. Registerbankumschaltung

Um der zeitlichen Randbedingungen gerecht zu werden, dass die aktualisierten Registerwerte innerhalb eines Taktes den Zustandsautomaten zur Verfügung stehen müssen, wurde das Konzept nochmals überarbeitet. Die Randbedingung kann nur eingehalten werden, indem die kompletten Registerbänke während des Updatezeitpunktes ausgetauscht werden. Physikalisch wird dies dadurch realisiert, dass Schreib- und Lesezugriffe auf die jeweils aktiven bzw. inaktiven Speicher ‚geroutet' werden. Der Updatevorgang verläuft nun im Gegensatz zu den vorhergehenden Realisierungen vom aktiven zum inaktiven Register. Weiterhin muss nach wie vor der SPI Lesezugriff über MISO mit dem Aktualisierungsvorgang arbitriert werden.

5.3.1 Machbarkeitsanalyse

Die kürzest mögliche Zeit zwischen zwei Aktualisierungszeitpunkten gibt vor, wie groß eine Registerbank tatsächlich sein darf, um während dieser Zeit sicher und komplett von der aktiven Bank in die inaktive übertragen werden können.

Es sei:

n_{reg} = Größe der Registerbank in Registern;

n_{px}= Anzahl der Takte zwischen zwei Aktualisierungszeitpunkten;

f_{SPI} = Frequenz der SPI Schnittstelle;

f_{px} = Pixelfrequenz auch für Updateroutine.

Weiterhin gilt, dass ein Register mindestens 31 Takte benötigt um über die SPI Schnittstelle übertragen zu werden.

Hieraus entsteht folgender Zusammenhang:

$$n_{reg} = \frac{f_{SPI}}{31 f_{px}} n_{px}$$

Die Anzahl der Takte um eine komplette Registerbank zu aktualisieren (n_{upd}), hängt im Wesentlichen vom Maximum der Lesezugriffe über die SPI Schnittstelle ab welche folgendermaßen festgelegt werden kann:

$$n_{SPIrd} = n_{reg} \frac{f_{SPI}}{31 f_{px}}$$

Die Lesezugriffe fordern weiteren zeitlichen Aufwand, da bei jeder Unterbrechung die Pipeline zum Lesen des Speichers neu geladen werden muss (n_{pipe}).

$$n_{upd} = n_{reg} + n_{SPIrd}\left(n_{rdcycles} + n_{pipe}\right)$$

$n_{rdcycles}$ steht hier für die Dauer eines SPI Lesevorgangs.

Ersetzt man nun n_{reg} sowie n_{SPIrd} durch die vorigen Erkenntnisse, entstehen folgende Zusammenhänge für die Dauer eines Updatevorgangs:

$$n_{upd} = \frac{f_{SPI}}{31 f_{px}} n_{px} + \frac{f_{SPI}}{31 f_{px}} n_{px} \frac{f_{SPI}}{31 f_{px}} \left(n_{rdcycles} + n_{pipe}\right)$$

$$n_{upd} = \frac{f_{SPI}}{31 f_{px}} n_{px} \left(1 + \frac{f_{SPI}}{31 f_{px}} \left(n_{rdcycles} + n_{pipe}\right)\right)$$

Ein Aktualisierungsvorgang muss beendet sein, bevor ein darauffolgender Aktualisierungszeitpunkt eintritt. Dies bedeutet:

$$n_{upd} \leq n_{px\_nxt}$$

$$n_{px\_nx} \geq \left(1 + \frac{f_{SPI}}{31 f_{px}}\left(n_{rdcycles} + n_{pipe}\right)\right)\frac{f_{SPI}}{31 f_{px}} n_{px}$$

Dies führt zu folgendem Größenverhältnis zweier aufeinander folgender Regionen unter den Randbedingungen der Speicherzugriffszeiten $n_{rdcycles}$ und n_{pipe}, welche als n_{mem} weitergeführt werden. Diese liegen zusammen jedoch kaum über 10 Takte.

$$\frac{n_{px\_nx}}{n_{px}} \geq \left(1 + \frac{f_{SPI}}{31 f_{px}} n_{mem}\right)\frac{f_{SPI}}{31 f_{px}}$$

Für die Anzahl der Takte in einer folgenden Region (n_{px_nx}) kann man zu Grunde legen, dass eine Region mindestens eine Zeile beinhalten muss. Diese ist am kürzesten für den kleinsten betriebenen Sensor mit 692 Pixeln. Wird von einer maximalen Speichergröße mit 512 Registern ausgegangen, was in einem Cyclone III FPGA exakt in zwei Speicherblöcken implementiert werden kann, so ergibt sich folgender Zusammenhang:

$$\frac{692}{512} \geq \left(1 + \frac{f_{SPI}}{31 f_{px}} 10\right)\frac{f_{SPI}}{31 f_{px}}$$

$$1{,}35 \geq 0{,}042$$

und bestätigt somit die Machbarkeit der geplanten Realisierung.

5.3.2 Implementierung

Die zeitliche Abfolge des Updatevorgangs ist im Detail in Abbildung 10 dargestellt. Der Unterschied zu den vorangegangen Implementierungen liegt hier in Schritt (b), hier werden zur steigenden Flanke eines Updatesignals scheinbar die Registerbänke ausgetauscht. Somit sind einen Takt später alle Registerinhalte aktualisiert. Diese können in Zeitpunkt (c) schon während des andauernden Updatevorgangs von der aktiven zur passiven Registerbank von den Zustandsautomaten ausgelesen werden.

Um dafür zu sorgen, dass Registerinhalte die bereits über die SPI-Schnittstelle modifiziert wurden, nicht durch den Updatevorgang wieder mit alten Werten überschrieben werden, wird hier die Möglichkeit genutzt, die SPI-Schnittstelle zu sperren.

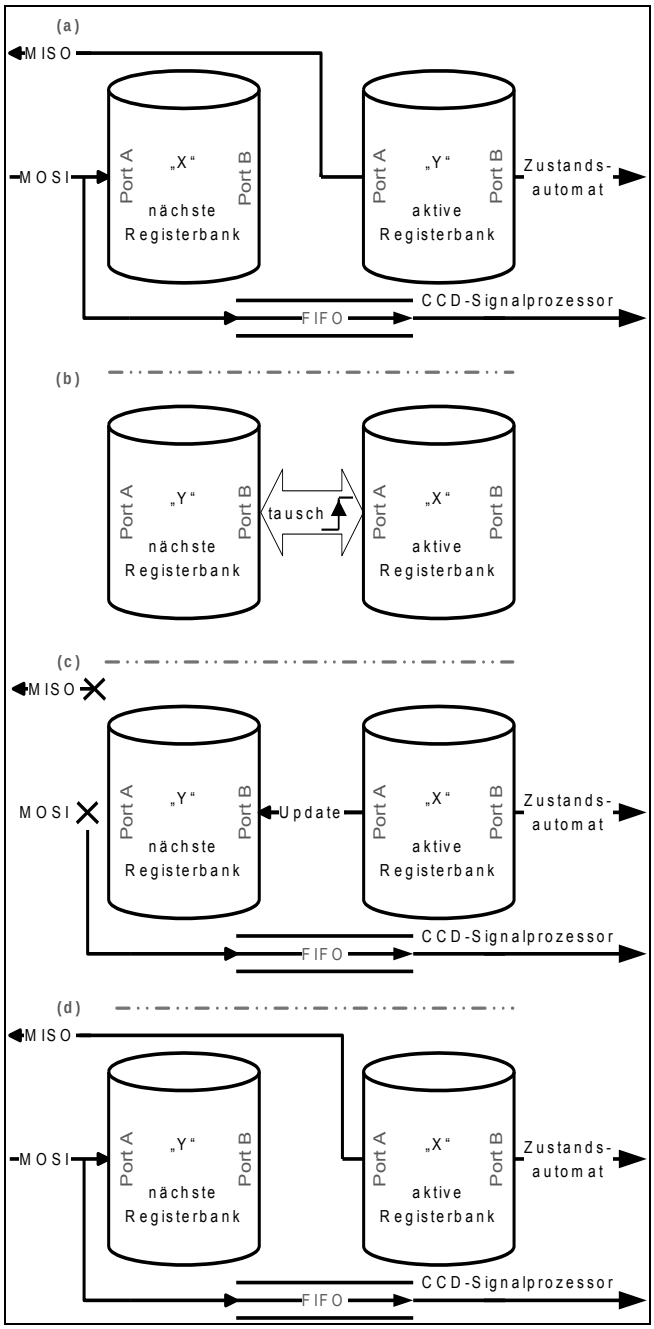


Abbildung 10 Darstellung der zeitlichen Abfolge der Zugriffe auf die aktiven und inaktiven Registerbänke in der Registerbankumschaltung Implementierung

6. Zusammenfassung

In der vorliegenden Arbeit wurde ein durch rund 1000 Register konfigurierbares System zur Ansteuerung komplexer und unterschiedlichster CCD Sensoren entworfen.

Neben der Ansteuerung der zeitkritischen und hochflexibel einsetzbaren Taktsignale nimmt die Speicherverwaltung einen großen Bestandteil ein. Hier liegt die Schwierigkeit darin, selbst beim Beschreiben einer großen Anzahl von Konfigurationsregistern zu garantieren, dass die Ansteuerung des Sensors kontinuierlich erfolgt.

Weiterhin werden die empfangen, A/D- gewandelten Ausgangsdaten des CCD-Sensors parallelisiert und möglicher Bilddatenvorverarbeitung unterzogen. Die Videodaten werden anschließend selektiert und nur die tatsächlich gewünschten Pixeldaten des „Area of interest“ an das darüber liegende System übertragen.

7. Ausblick

Der Trend in der Bildverarbeitung (beispielsweise bei digitalen Spielgelreflexkameras oder aber in der Überwachungs- und Medizintechnik) geht eindeutig zu höheren Auflösungen. Um hier dieselben Frameraten zu erhalten muss der Pixeltakt erhöht werden. Da dies auch Nachteile wie hochfrequente Störungen im Bild mit sich führt, sind sogenannte Mehrkanalsensoren auf dem Markt verfügbar.

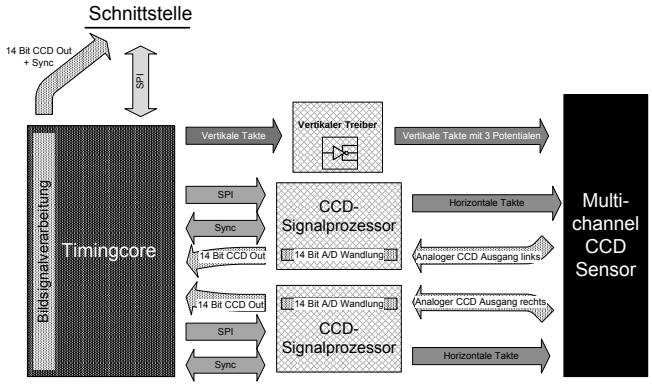


Abbildung 11 Funktionelles Blockschaltbild zur Ansteuerung von Multichannel CCD-Sensoren.

Bei einem Zweikanalsensor ist die horizontale Auslesezeile (vgl. horizontales Förderband) zweigeteilt. Eine Hälfte der Zeile wird nach links, die andere nach rechts ausgelesen. Folglich benötigt man zwei A/D Wandler und eine Logik, die die beiden Teilzeilen und bzw. die beiden Teilbilder wieder zusammenfügt.

Ein Vorteil dieser Architektur liegt darin, dass mit dem halben Pixeltakt dieselbe Menge an Bilddaten wie bei einem herkömmlichen Sensor mit einfachem Pixeltakt ausgelesen werden kann.

Für solch eine zukünftige Architektur stellt der entwickelte Timingcore eine ideale Basis darstellen.

8. Literaturverzeichnis

Beynon, J. D. E., Lamb, D. R. 1980. *Charge-coupled devices and their applications.* London : McGRAW-HILL Book Company (UK) Limited, 1980. ISBN: 0-07-984522-0.

Boyle, W. S. and Smith, G. E. 1970. Charge Coupled Semiconductor Devices. *Bell Syst. Tech. J..* April 1970, vol. 49, pp. 587-593.

Holst, Gerald C. 1996. *CCD Arrays Cameras and Displays.* Bellingham : JCD Publishing, 1996. ISBN: 0-9460000-2-4.

Janesick, James R. 2001. *Scientific Charge-Coupled Devices.* Bellingham, Washington : The Society of Photo-Optical Instrumentation Engineers, 2001. iSBN 0-8194-3698-4.

Melen, Roger, Buss, Dennis. 1977. *Charge-Coupled Devices: Technology and Applications.* New York : IEEE Press, 1977. ISBN: 0-87942-084-7.

Sony. *ICX414 Datasheet.* [*.pdf document] s.l. : Sony Corporation. Diagonal 8mm (Type 1/2) Progressive Scan CCD Solid-state Image Sensor with Square pixel.

Theuwissen, Albert J.P. 1996. *Solid-State Imaging with Charge-Coupled Devices.* Dordrecht, The Netherlands : Kluwer Academic Publishers, 1996. ISBN: 0-792-33456-6.

Low-price ARM7 development

Stefan Saulig

HTW Aalen, EDA Zentrum, Anton Huber Strasse 25, 73430 Aalen

Tel. 07361 / 576 - 4247, Fax 07361 / 576 - 444249

Zusammenfassung

Die Marktanforderungen an IT Produkte entwickeln sich generell vom Einfachen zum Komplizierten und vom Niederkomplexen zum Höherkomplexen. Im Bereich der Embedded Systems findet im Moment der Generationsübergang zu den 32 Bit Controllern statt. Dieser Übergang ist unabhängig davon, ob es Embedded Systems für regelungstechnische Aufgaben, Automatisierungsprozesse, Datenverarbeitungs- und Konvertierungsprozesse, Datenfusionsplattformen oder Systemintegrationsschnittstellen sind. Neben der Notwendigkeit einen höheren Datenbereich darzustellen und zu verarbeiten, spielt die bereits im Controller integrierte Peripherie eine erhebliche Rolle. So ermöglicht dies weniger komplexe, zuverlässigere, leistungsfähigere und kostengünstigere Designs gegenüber Controllern mit ausgelagerter Peripherie in Form zusätzlicher ICs auf der Boardebene. Nachteilig ist beim Einstieg bzw. Umstieg in den 32 Bit Controller Bereich, das anfänglich hohe Kosten entstehen, da sowohl JTAG Programmiergeräte als auch die benötigte IDE neu beschafft werden müssen. Die Anschaffungskosten von Development Boards sind im Vergleich zu 8 Bit Controller Boards um ein Vielfaches höher. In der Anfangsphase eines Generationsübergangs stehen nur wenige Informationen zur Verfügung. Diese Tatsache hemmt sehr häufig einen schnellen und kostengünstigen Umstieg. Aktuell stellt die IDE Eclipse mit ihren Plug-Ins für den ARM7, in Kombination mit einem JTAG Tool, eine kostengünstige Lösung dar. Die Herstellung eines eigenen Development Boards kann in vielen Fällen die wesentlich günstigere Variante sein.

1. Softwareumgebung Eclipse

1.1. Installation

Eclipse ist eine open Source Programmierumgebung zur Entwicklung von Software. In diesem Fall als IDE „integrated development environment“ verwendet um Programme für eine ARM7 Plattform zu schreiben, kompilieren und über JTAG zu übertragen.

Eclipse selbst stellt das Programmiergerüst (engl. framework) dar, in welches erst die benötigten Tools integriert werden müssen, um den Source-Code zu kompilieren und programmieren zu können. Nachfolgende Software muss installiert werden:

- Java Runtime Environment (JRE) 6
  http://java.sun.com
- AT91 In-System Programmer (ISP)
  http://www.atmel.com
- Open On-Chip Debugger
  http://www.yagarto.de
- YAGARTO GNU ARM Toolchain
  http://www.yagarto.de
- Eclipse IDE for C/C++ Developers
  http://www.eclipse.org

Als erstes muss die Java JRE installiert werden. Die erfolgreiche Installation kann durch die Eingabe „java –version“ in der Eingabeaufforderung überprüft werden.

Es folgt die Installation von OpenOCD „Open On-Chip-Debugger“.

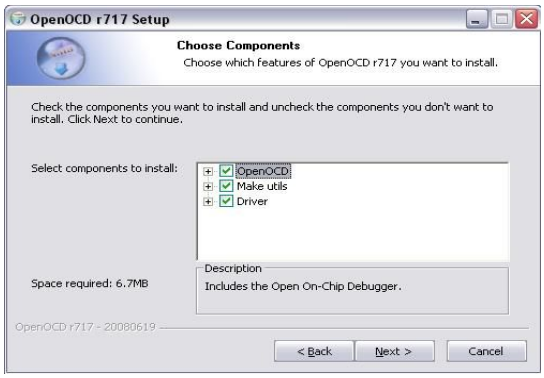


Abbildung 1 : OpenOCD Setup

In der Eingabeaufforderung mit „make –v“ den Erfolg überprüfen und es sollte „GNU Make 3.81 …“ erscheinen. OpenOCD unterstützt das Debugging, ISP „In-System-Programming“ und Boundary Scan Testing von Controllern. Dies erfolgt mittels JTAG IEEE 1149.1 konformer Hardware. Für die Programmiergeräte sind Treiber, die auf FTDI Treiber basieren, bereits im Softwarepacket integriert. Unterstützte Controller sind zum Beispiel ARM7 (ARM7TDMI und ARM720t), ARM9 (ARM920t, ARM922t, ARM926ej–s, ARM966e–s), XScale (PXA25x, IXP42x) und Cortex-M3 (Luminary Stellaris LM3 und ST STM32).

Zusätzlich kann zur Programmierung bei den meisten Controllern der oben genannten Familien über die serielle Schnittstelle programmiert werden. Hierzu muss vom Hersteller ein entsprechendes Tool installiert werden. Für den ARM7 von Atmel ist dies der AT91 In-System Programmer.

Abbildung 2 : AT91 ISP Setup

Nachfolgende Benutzeroberflächen stehen bereit.

Abbildung 3 : Benutzeroberfläche SAM-BA

Zusätzlich benötigte Toolchain „Programmier-werkzeuge“, wie z.B. GCC, GDB, Binutils und Newlib werden mit der YAGARTO GNU ARM Toolchain installiert.

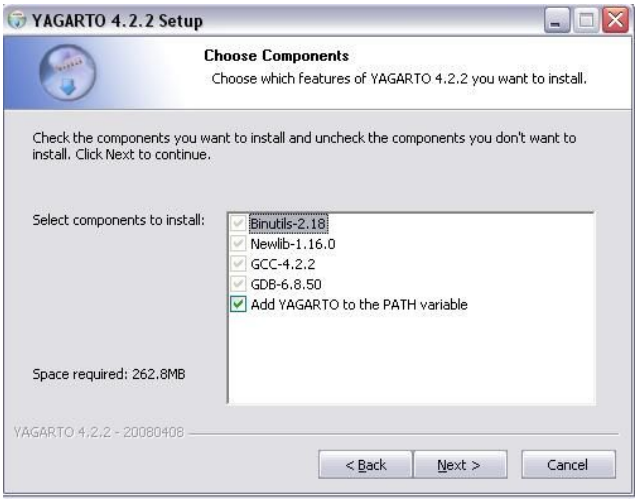


Abbildung 4 : Yagarto Toolchain Setup

In der Eingabeaufforderung wird mit „arm-elf-gcc -v“ die Installation überprüft und es sollte „Using built-in specs. Target: arm-elf Configured with: …“ erscheinen. GCC ist der C Compiler, GDB der

Debugger und Newlib ist eine C Bibliothek für Embedded Systems Prozessoren bzw. Controllern.

Binutils „GNU Binary Utilities“ enthält eine Sammlung von Programmierwerkzeugen zur Manipulation (anlegen, verändern, löschen) von Objektcode (Compilerzwischenergebnissen) und Objektdateien. Ebenfalls enthalten ist der Assembler und Linker.

Zuletzt wird Eclipse installiert.

Abbildung 5 : Oberfläche Eclipse

Da Eclipse selbst zur Programmierung von Java entwickelt wurde, ist die Installation von CDT (C/C++ Development Tool) notwendig. Hierzu sind zwei Internetquellen aufgeführt, welche das CDT bereitstellen.

Unter Help → Software Updates → Find and Install … → Search for new features to install → New Remote Site …

- ZylinCDT
  http://zylin.com/zylincdt
- EclipseOrgCDT
  http://download.eclipse.org/tools/cdt/releases/europa

Download und Installation der Updates bzw. Installationspakete.

Es folgt in 1.3 Konfiguration der weitaus unangenehmere Teil der Konfiguration von Eclipse und den Toolchain.

1.2. Toolchain Systemmodell

Das Zusammenwirken der installierten Software soll anhand der Abbildung 6 und Abbildung 7 erläutert werden.

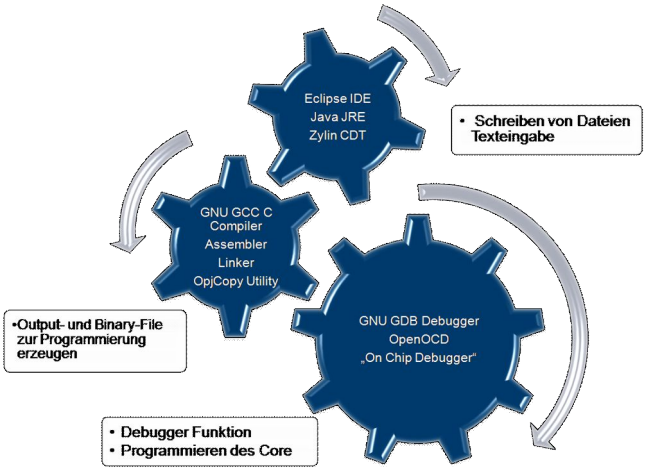


Abbildung 6 : Zusammenwirken der Toolchain

In der Eclipse IDE werden C- und Assembler-Dateien geschrieben, Dateien werden in das Projekt eingebunden, eine Projektstruktur wird erzeugt und das Projekt wird über die IDE gesteuert. Eclipse dient als Front-End.

Compiler, Assembler und Linker erzeugen das Output-File. Mittels Binutils wird ein Binary-File erzeugt.

Die Programmierung des Flash-Speichers des Controllers kann seriell über die RS232-Schnittstelle erfolgen, indem die Software AT91-ISP, für entsprechende Atmel Controller, von Atmel genutzt wird.

Direkt aus Eclipse heraus kann mit der JTAG-Hardware der Controller programmiert werden. Hierzu wird im Hintergrund der OpenOCD „Open On Chip Debugger“ genutzt, welcher als Daemon fungiert und die Softwareanbindung der JTAG-Hardware realisiert.

Um die Debuggerfunktionalität für Eclipse zu ermöglichen, wird der GDB Debugger verwendet, welcher auf den Daemon OpenOCD zugreift.

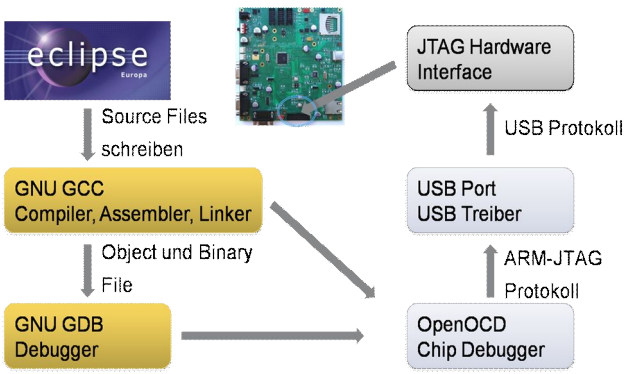


Abbildung 7 : Projektexport zum Controller

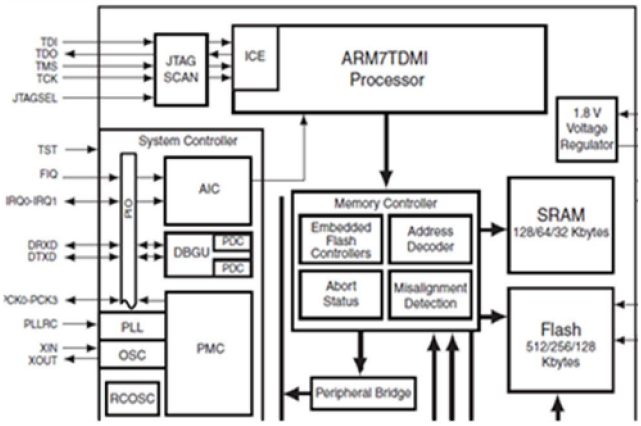


Abbildung 8 : Atmel AT91SAM7X512 [4]

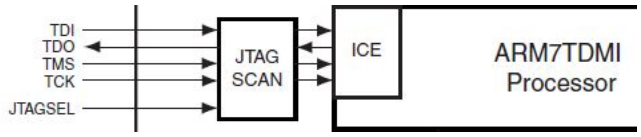


Abbildung 9 : Embedded ICE Macrocell [4]

ICE „In-Circuit Emulator “implementiert die Debugger Funktionen On-Chip und ist ein Hilfsmittel zur Softwareerstellung von Embedded Systems.

ICE Hauptfunktionen:

- Breakpoint - Programmausführung unterbrechen
- Single Step - Programm in Einzelschritten ausführen
- Register und Speicher auslesen
- Tracebuffer - Ereignisaufzeichnung

JTAG „Joint Test Action Group“ Interface stellt die Verbindung zu ICE bereit. JTAG ist unter IEEE-Standard 1149.1 bekannt. Es ist ein Verfahren zum Testen und Debuggen von Hardware (FPGA, CPLD, Mikrocontrollern), bekannt als Boundary Scan Test (Scantestkette).

JTAG Erweiterungen:

- Konfiguration von FPGA und CPLD
- Programmieren und Debuggen von Mikrocontrollern
- Programmierung von FPGA Programmspeicher (extern)

1.3. Konfiguration

Die Konfigurationsdateien aus dem Verzeichnis Script mit der Endung .CFG werden in das OpenOCD Verzeichnis BIN kopiert.

- at91sam7x512-armusbocd.cfg
- at91sam7x512-armusbocd-flash-program.cfg
- script.ocd

Die Scripte müssen selbst erstellt werden, angepasst werden oder liegen dem OpenOCD bei. Ggf. wird beim Hersteller des Programmiergerätes diese Information hinterlegt.

Konfiguration des OpenOCD (Run ohne Flash):

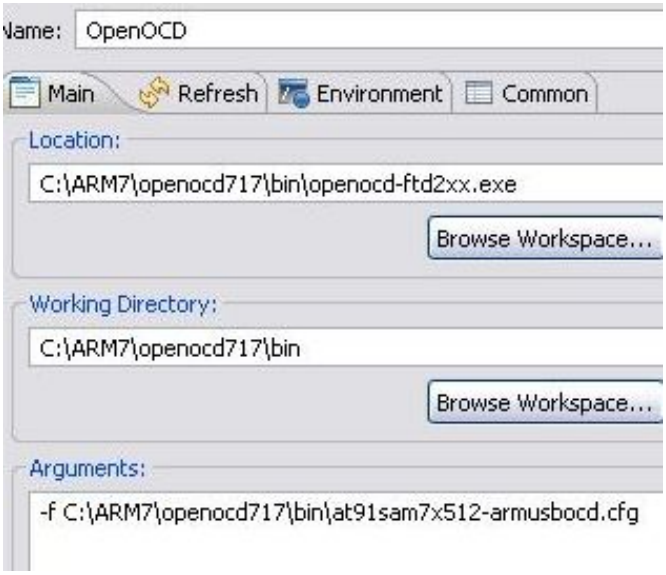


Abbildung 10 : Konfiguration OpenOCD Run

Konfiguration des OpenOCD (Run und Flash):

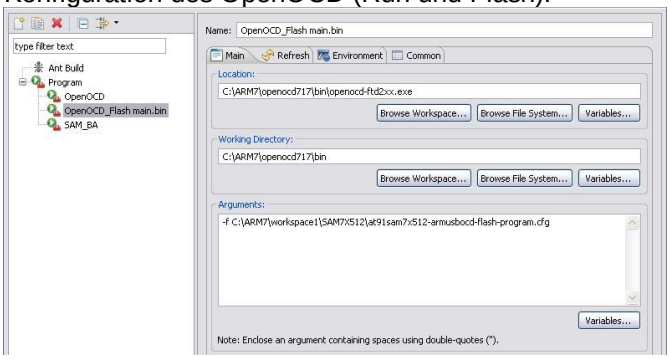


Abbildung 11 : Konfiguration OpenOCD Run Flash

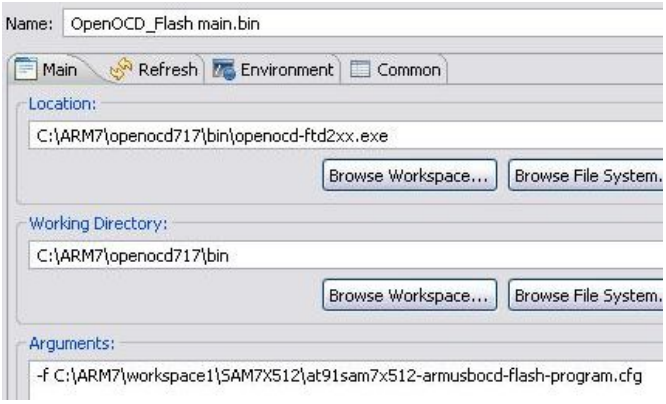
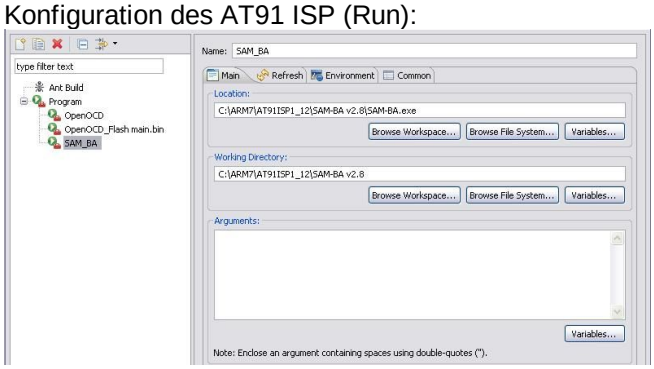


Abbildung 12 : Konfiguration OpenOCD Run Flash

Es ist wichtig entsprechende Verzeichnisangaben, sowohl in den Konfigurationen aus auch in den Skripten, auf die Installationsverzeichnisse anzupassen.

Konfiguration des AT91 ISP (Run):



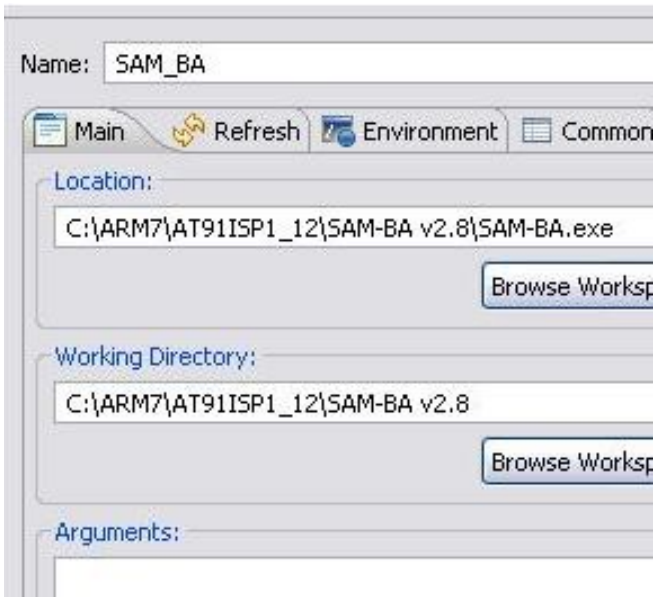


Abbildung 13 : Konfiguration SAM-BA / AT-91 ISP

Um festzustellen, dass die Konfiguration erfolgreich war, wird nun SAM_BA gestartet, Menü RUN→SAM_BA. Die Oberfläche des Programmes SAM-BA (AT91-ISP) der Firma Atmel wird gestartet.

Danach wird der OpenOCD aufgerufen und in der Eclipse Console müssen wie in Abbildung 14 Statusmeldung erscheinen:

```
Open On-Chip Debugger (2008-06-19 19:00) svn: 717

URL: http://svn.berlios.de/svnroot/repos/openocd/trunk

Info:    options.c:50 configuration_output_handler(): Open On-Chip
Debugger (2008-06-19 19:00) svn: 717

Info:    options.c:50 configuration_output_handler(): jtag_speed: 2, 2

Info:    options.c:50 configuration_output_handler(): Open On-Chip
Debugger (2008-06-19 19:00) svn: 717

Info:    jtag.c:1389 jtag_examine_chain(): JTAG device found:
0x3f0f0f0f (Manufacturer: 0x787, Part: 0xf0f0, Version: 0x3)

Info:    jtag.c:1389 jtag_examine_chain(): JTAG device found:
0x3f0f0f0f (Manufacturer: 0x787, Part: 0xf0f0, Version: 0x3)
```

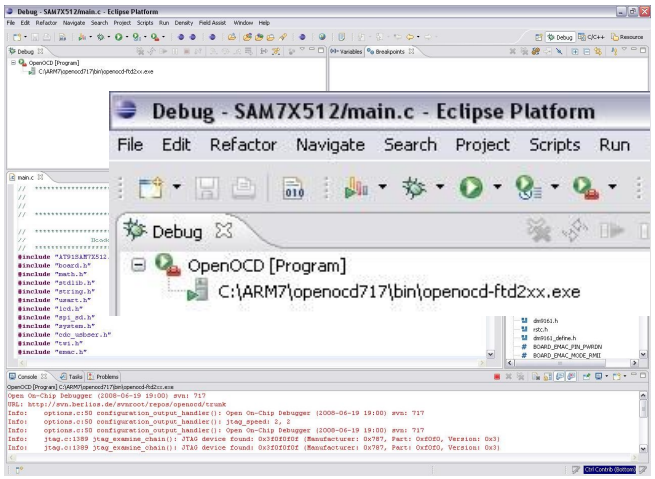


Abbildung 14 : OpenOCD Serverdienst

OpenOCD sollte als Serverdienst, wie in der Abbildung 14 gezeigt, laufen. Gleichzeitig muss die JTAG-Hardware an der angeschlossenen Platine diesen Zustand signalisieren, indem die Status-LED des Debuggers/Programmers blinkt.

Falls sich dieser Zustand einstellt wird die Erreichbarkeit des Serverdienstes getestet. Hierzu wird per Hyperterminal eine Verbindung aufgebaut, mit folgenden Einstellungen:

- TCP/IP - Hostadresse : x.x.x.x (eigene PC IP)
- PORT bzw. Anschlussnummer : 4444
- Emulation : ANSI bzw. Auto-Ermittlung

Nach dem Anruf des Ports meldet sich OpenOCD mit der Meldung:

```
++Open On-Chip Debugger
>
```

Gleichzeitig gibt die Console im Eclipse die Meldung:

```
Info:    server.c:78 add_connection(): accepting
'telnet' connection from 0
```

Alternativ kann per Eingabeaufforderung mit telnet xxx.xxx.xxx.xxx 4444 die Verbindung aufgebaut werden.

Falls dieser Zustand erfolgt, wird die Erreichbarkeit des Serverdienstes über den GDB getestet. Hierzu wird per Hyperterminal eine Verbindung aufgebaut, mit den Einstellungen:

- TCP/IP -> Hostadresse : x.x.x.x (eigene PC IP)
- PORT bzw. Anschlussnummer : 3333
- Emulation : ANSI bzw. Auto-Ermittlung

Nach Anruf des Ports 3333 meldet sich OpenOCD mit der Meldung:

```
+
```

Die Verbindung wird vom GDB abgebrochen (Time Out), da keine Konfiguration bzw. Daten übergeben werden. Es wurde aber festgestellt, dass der GDB erreichbar ist.

```
Info:    server.c:78 add_connection(): accepting 'gdb'
connection from 0
Error:   server.c:85 add_connection(): attempted 'gdb'
connection rejected
```

Konfiguration des Debugger (Run):

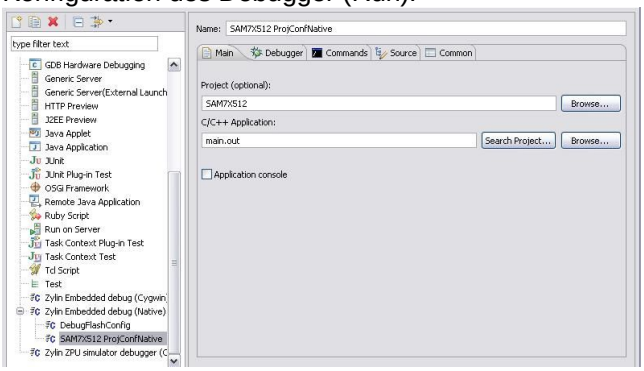
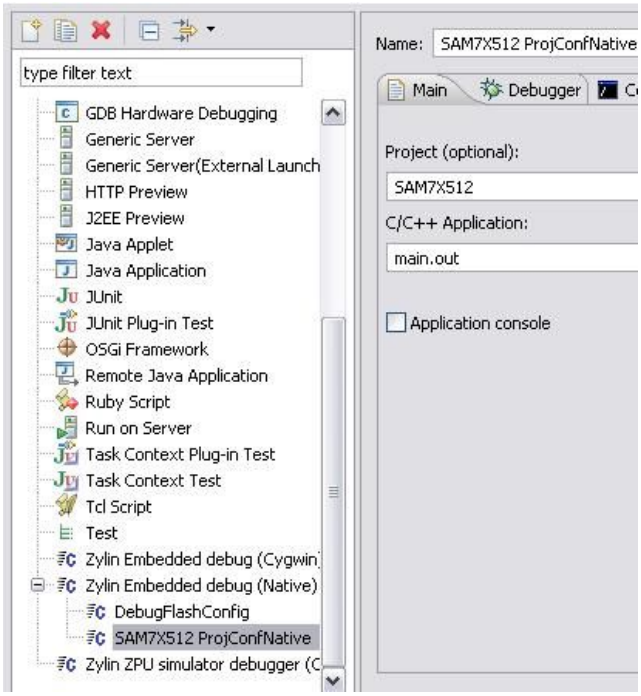


Abbildung 15 : Konfiguration Debugger



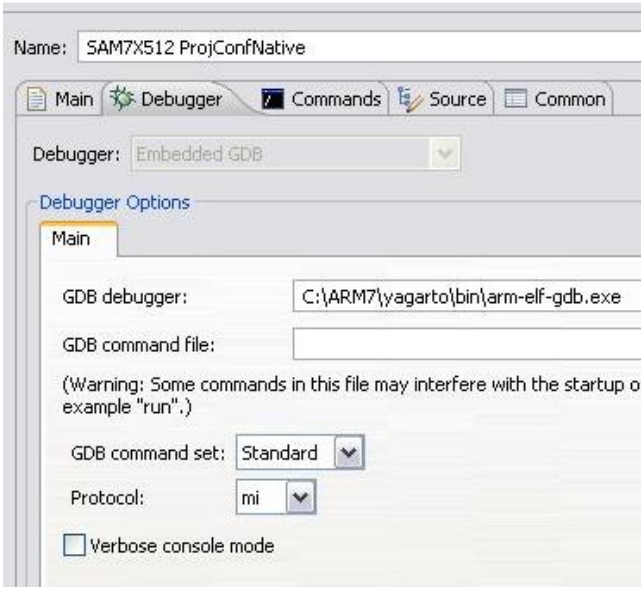


Abbildung 16 : GDB Debugger

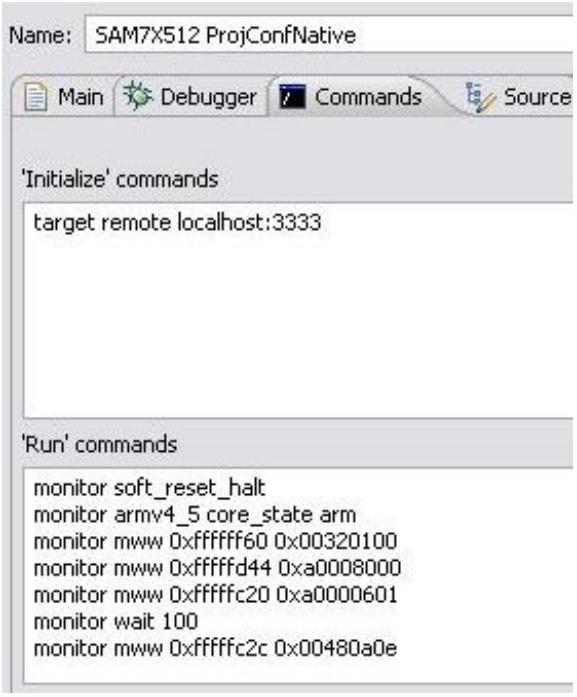


Abbildung 17 : Debugger Run Commands

```
monitor reset
monitor sleep 500
monitor poll
monitor soft_reset_halt
monitor arm7_9 force_hw_bkpts enable
monitor mww 0xFFE00000 0x1000213D
```

Entsprechende Möglichkeiten der Abfolge der Ausführungsbefehle für den GDB:

```
monitor soft_reset_halt
# OpenOCD command to halt the processor and wait
monitor armv4_5 core_state arm
# OpenOCD command to select the core state
monitor mww 0xffffff60 0x00320100
# set flash wait state (AT91C_MC_FMR)
monitor mww 0xfffffd44 0xa0008000
# watchdog disable (AT91C_WDTC_WDMR)
monitor mww 0xfffffc20 0xa0000601
# enable main oscillator (AT91C_PMC_MOR)
monitor wait 100
# wait 100 ms monitor
mww 0xfffffc2c 0x00480a0e
# set PLL register (AT91C_PMC_PLLR)
monitor wait 200
# wait 200 ms
monitor mww 0xfffffc30 0x7
# set master clock to PLL (AT91C_PMC_MCKR)
monitor wait 100
# wait 100 ms
monitor mww 0xfffffd08 0xa5000401
# enable user reset AT91C_RSTC_RMR
set remote memory-write-packet-size 1024
# Setup GDB for faster downloads
set remote memory-write-packet-size fixed
# Setup GDB for faster downloads
set remote memory-read-packet-size 1024
# Setup GDB for faster downloads
set remote memory-read-packet-size fixed
# Setup GDB for faster downloads
#monitor mww 0xfffffd00 0xa5000004
# force a peripheral RESET AT91C_RSTC_RCR
#monitor mww 0xffffff00 0x01
# toggle remap register to place RAM at 0x00000000
#monitor reg pc 0x00000000
# set the PC to 0x00000000
monitor arm7_9 force_hw_bkpts enable
# convert all breakpoints to hardware breakpoints
symbol-file main.out
# read the symbol information from main.out
#load
# download the application using file main.out
continue
# resume execution from reset vector, break at main
```

```
monitor soft_reset_halt
# OpenOCD command to halt the processor and wait
monitor armv4_5 core_state arm
monitor mww 0xffffff60 0x00320100
monitor mww 0xfffffd44 0xa0008000
monitor mww 0xfffffc20 0xa0000601
monitor wait 100
monitor mww 0xfffffc2c 0x00480a0e
monitor wait 200
monitor mww 0xfffffc30 0x7
monitor wait 100
monitor mww 0xfffffd08 0xa5000401
set remote memory-write-packet-size 1024
set remote memory-write-packet-size fixed
set remote memory-read-packet-size 1024
set remote memory-read-packet-size fixed
monitor mww 0xfffffd00 0xa5000004
monitor mww 0xffffff00 0x01
monitor arm7_9 force_hw_bkpts enable
symbol-file main.out
continue
```

Ein Starten des Debugmodes aus Eclipse sollte wie in Abbildung 18 aussehen:

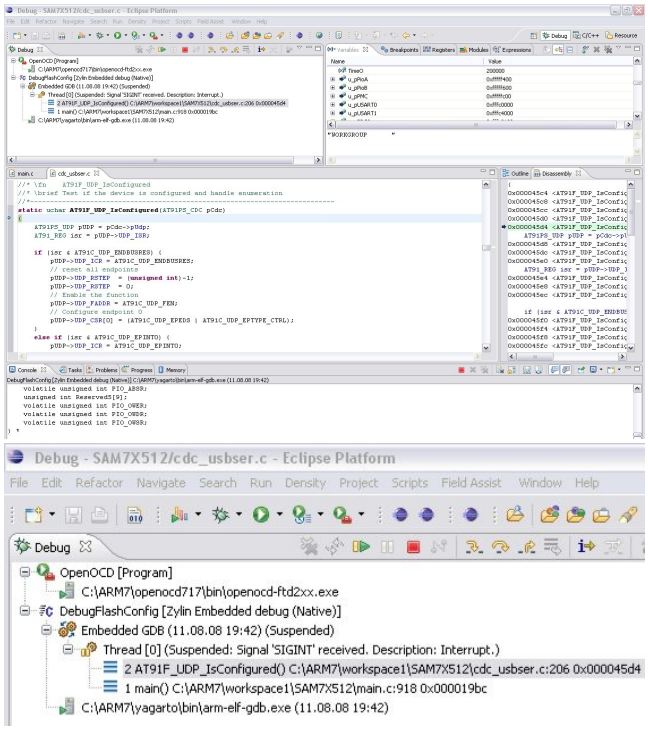


Abbildung 18 : Debugmodus gestartet

Wichtig:
Vor dem Starten des arm-elf-gdb (Debugger) muss der OpenOCD gestartet werden. Wenn der OpenOCD gestartet wurde, kann per Telnet oder Hyperterminal auf den ARM Core zugegriffen werden, indem Befehle über OpenOCD an den ARM-USB-OCD zum JTAG des Core gesendet werden. Ebenfalls kann nun über Eclipse der Debugger gestartet werden.

1.4. Projekt erstellen/implementieren

Die nun fertig konfigurierte Entwicklungsumgebung bietet die Möglichkeit, Projekte vollständig zu betreuen und auf den ARM Core zu übertragen. Auf eine stark detailierte Beschreibung der Konfiguration und Benutzung der Entwicklungssoftware und Tools wird hier verzichtet, da dies bereits Bücher füllen würde.

Zum Erstellen eines Projektes unter Menüpunkt File→New→Projekt den Menüpunkt C Projekt öffnen. In den nachfolgenden Selectionsregisterkarten werden Parameter konfiguriert und bestätigt. Wichtig hierbei ist darauf zu achten, dass ein Make C Projekt erzeugt wird.

Im Rahmen der Projektarbeit [3] wurde ein vollständiges Projekt entwickelt, das zu diesem Zeitpunkt zur Verfügung stand. Die detaillierten Eclipse Konfigurationspunkte sind in dieser Arbeit dokumentiert.

Nach dem Erstellen des Dateiensystems für das Projekt werden die eigentlichen Dateien importiert. Dazu File→Import→FileSystem→from Directory …. Select All wählen und mit Finish betätigen. Um die Richtigkeit der Übernahme des Projekts zu Prüfen, ob der Builder bzw. Compiler funktioniert, wird der Menüpunkt Build All „Ctrl+B“ aufgerufen.

Die Console von Eclipse sollte nachfolgendes anzeigen:

```
**** Build of configuration Default for project SAM7X512 ****

make all
...copying
arm-elf-objcopy --output-target=binary main.out main.bin
arm-elf-objdump -x --syms main.out > main.dmp
```

Sobald unter dem Ausgabefenster „Problems“ Fehler angezeigt werden, muss die Konfiguration des Projektes überprüft werden. Mögliche Fehler können die Einstellung für CDT, Makefile, Builder, Compiler und die installierten Softwarepakete sein.

Zu bemerken ist ebenfalls, dass eine richtige Installation von Eclipse keine große zusätzliche Konfiguration der Software Eclipse erfordert.

Builder Projekteinstellung:

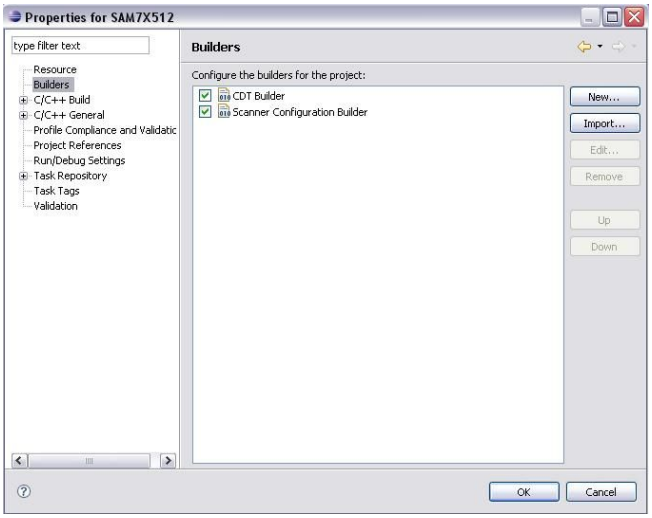


Abbildung 19 : Builder Konfiguration Projekt

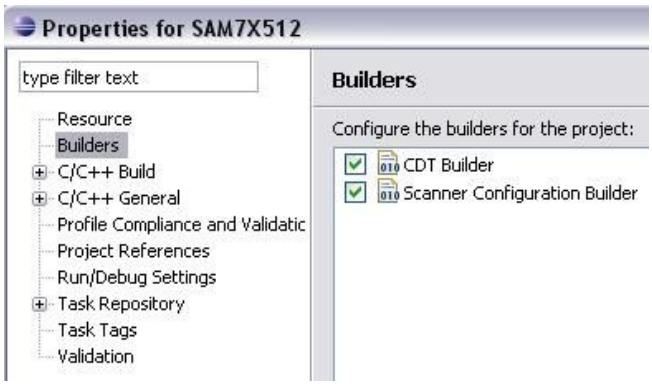


Abbildung 20 : Builder CDT

Wichtig:
Es sei ausdrücklich vor dem Ausprobieren verschiedener unsinniger Konfigurationen gewarnt, ebenfalls vor vielen fehlerhaften Beschreibungen im Internet, da dies zu starken Veränderungen des Projektes führt. Nicht erfahrene Eclipse Nutzer können damit ihr Projekt leicht unnutzbar machen.

1.5. Das Programmieren des ARM Core

Die wichtigste Funktion, nach dem das Projekt erfolgreich mit dem Menübefehl Build All übersetzt wurde, ist das Übertragen des Binary-Files, welches vom ARM Core sequenziell abgearbeitet wird.

Hierzu wird Run→OpenOCD_Flash[1] aufgerufen. Die hier konfigurierten Skripte wurden so geschrieben, dass der Flash Speicher des MC „Microcontroller“ als Zielmedium dient und nach einem Reset der MC seine Programmabarbeitung von dort aus beginnt. Im ROM des MC ist ein Bootloader[2] abgelegt, welcher durch das Setzen von Konfigurations-Bits aktiviert werden kann. Bei MC Start mit aktivem Bootloader kann das Programm per USB oder DEBUG-Port (RS232) übertragen werden, abweichend von der hier vorgestellten Methode. Um den Bootloader zu aktivieren, kann der Schalter für ERASE aktiviert werden, 10 Sekunden warten, Strom ausschalten, Schalter ERASE ausschalten, Strom einschalten und der Bootloader ist aktiv.

Der Befehl Run→OpenOCD_Flash startet gegebenenfalls Build All und Make All. Anschließend liegt als Ergebnis die Datei main.bin vor. Eclipse startet den OpenOCD openocd-ftd2xx.exe, welcher mit der Konfigurationsdatei at91sam7x512-armusbocd.cfg konfiguriert wird. Hieraus selbst wird das Core Programmierungsskript script.ocd gestartet. Dieses Skript setzt den Core in den Halt Mode, konfiguriert PLL und Clock, setzt den Core in DebugMode, schreibt die Datei main.bin in den Flash und konfiguriert wichtige Bits des Core. Nach dem Programmieren des Cores wird ein Reset ausgelöst und der Core startet im konfigurierten Speicherbereich.

Die entsprechenden Skripte[1] können Sie ordern und dienen zur entsprechenden Anpassung an andere Core.

Das übertragene Programm befindet sich im Flash Speicher des MC.

[1] http://openocd.berlios.de/web/
[2] Bootloader wird automatisch aktiviert, wenn Erase vor dem Reset aktiv war.

2. Hardware

2.1. Der ARM-USB-OCD von Olimex

Die gesamte vorgestellte Software benötigt einen entsprechenden Hardwareanteil um mit dem Core Daten auszutauschen. Hierzu dient Beispielsweise der Programmer/Debugger mit der Bezeichnung ARM-USB-OCD aus Abbildung 21.

Nachfolgend Bilder vom ARM-USB-OCD:

Abbildung 21 : Olimex ARM-USB-OCD [1]

Der von Olimex vermarktete USB-OCD wird mit entsprechender Software ausgeliefert. Wirtschaftlich gesehen entsteht eine Kosteneinsparung von bis zu 90% gegenüber vergleichbaren Geräten von Markenherstellern. Vergleicht man die benötigte

Software, welche Markenfirmen vertreiben, ist man bei der Kombination Eclipse, ARM-USB und Tools deutlich im wirtschaftlichen Vorteil.

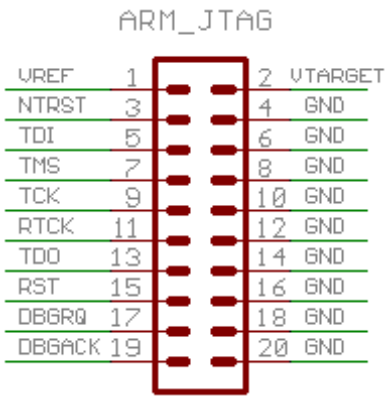


Abbildung 22 : JTAG Port

Die Belegung des JTAG Port ist der Abbildung 22 zu entnehmen.

Eine kurze Produktbeschreibung des Gerätes in Tabelle 1 legt die wesentlichen Produkteigenschaften dar:

ARM-USB-OCD 3-IN-1 FAST USB ARM JTAG, USB-TO-RS232 VIRTUAL PORT AND POWER SUPPLY 5-9-12VDC DEVICE (SUPPORTED BY OPENOCD OPEN SOURCE ARM DEBUGGER)
First on market three-in-one USB JTAG debugger - offers JTAG + RS232 (full modem signals supported) port + power supply all in one compact device
Fast speed USB 2.0 JTAG dongle interface, can be used with all ARM devices for programming and debugging.
uses ARM's standard 2x10 pin JTAG connector
supports ARM targets working in voltage range 2.0 – 5.0 V DC
software supported by OpenOCD (open source) debugger
adds virtual RS232 port to your computer with all modem signals like: DTR, DSR, DCD, RTS, CTS, Rx, Tx
can be used as power supply to your target board with three jumper selectable power supplies: 5V 9V and 12VDC, USB source current is limited with resetable fuse at 300mA, at the different output voltage the maximum current is different: 5V/200mA, 9V/100mA, 12V/70mA, note that this also depend on your USB host current capabilities, if other USB devices are attached to your computer or if the laptop is running on batteries these figures may be different and depend on your computer USB host.
comes with CD with Windows installer for full featured and open source tools as alternative to the commercial ARM development packages: GCC C compiler, openOCD debugger and Eclipse IDE.
dimensions 50x40 mm (2x1.6") + 20 cm (8") JTAG cable + 30 cm (12") power supply cable

Tabelle 1 : Olimex ARM-USB-OCD Parameter [1]

Aus den Produktunterlagen der Firma Olimex ist nicht sofort ersichtlich, dass die mitgelieferten Treiber relativ veraltet sind, womit sich zusätzlicher Arbeitsaufwand ergibt. Sie sind nicht konform mit dem OpenOCD Serverdienst (Befehlssatz unterschiedlich und veraltet). Der ARM-USB-OCD besteht im Wesentlichen aus einem FTDI[3] FT2322 und kann als solcher auch selbst abgeänderte Treiber von FTDI verwenden. Hierzu ist eine VID von 15ba und PID 0003 einzustellen. Die angepassten INF Dateien sind im Anhang dargelegt und gleichzeitig auf dem Datenträger enthalten.

Bei der Installation der Treiber vom ARM-USB-OCD ist manuell der Treiber zu wählen. Nach einer erfolgreichen Installation der Treiber sollte wie in Abbildung 23 nachfolgend aufgezeigte Hardware in der Systemsteuerung enthalten sein:

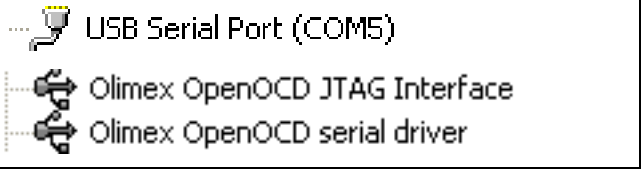


Abbildung 23 : System Hardware

Abbildung 24 und Abbildung 25 enthalten Konfigurationseinstellungen:

```
[Strings]
Ftdi="FTDI"
DESC="Olimex OpenOCD serial port"
DriversDisk="Olimex OpenOCD serial port drivers"
USB\VID_15ba&PID_0003.DeviceDesc="USB Serial Converter"
```

[3] Future Technology Devices International Limited; www.ftdichip.com

```
USB\VID_15ba&PID_0003&MI_00.DeviceDesc="Olimex OpenOCD JTAG Interface"
USB\VID_15ba&PID_0003&MI_01.DeviceDesc="Olimex OpenOCD serial driver"
SvcDesc="Olimex OpenOCD serial driver"
ClassName="USB"
```

Abbildung 24 : FTDIBUS.INF Strings [2]

```
[Strings]
FTDI="FTDI"
DESC="CDM Driver Package"
DriversDisk="Olimex OpenOCD driver 2.0"
PortsClassName = "Ports (COM & LPT)"
VID_15ba&PID_0003.DeviceDesc="USB Serial Port"
VID_15ba&PID_0004.DeviceDesc="USB Serial Port"
SvcDesc="USB Serial Port Driver"
SerEnum.SvcDesc="Serenum Filter Driver"
```

Abbildung 25 : FTDIPORT.INF Strings [2]

Der Olimex ARM-USB-OCD ist für circa 60 Euro erhältlich und somit wesentlich günstiger als vergleichbare Geräte die bei 150 Euro (ohne benötigte Softwaremodule zur vollständigen Programmierung) beginnen. Durchschnittlich lieget der Preis bei 300 € bis 700 € für das Gerät ohne entsprechende IDE und Toolchain.

2.2. Beispiel eines selbst erstellten Development Boards

Da viele Varianten der 32 Bit Controller als TQFP Package erhältlich sind, kann ein Design mit wenig Aufwand entwickelt werden und ggf. die Leiterplatte extern gefertigt werden.

Im Rahmen der Projektarbeit [3] wurden zwei Development Boards entworfen und gefertigt. Die Boards dienen als Systemschnittstelle und Systemsteuerung für eine fliegende Plattform. Eclipse IDE dient als Grundlage zur Programmierung und Softwareentwicklung.

Das Projekt mit dem Namen „QuadZeppCopter“ ist im Internet unter http://quadzeppcopter.eda.htw-aalen.de zu finden. Details können über das EDA-Zentrum der HTW Aalen erfahren werden.

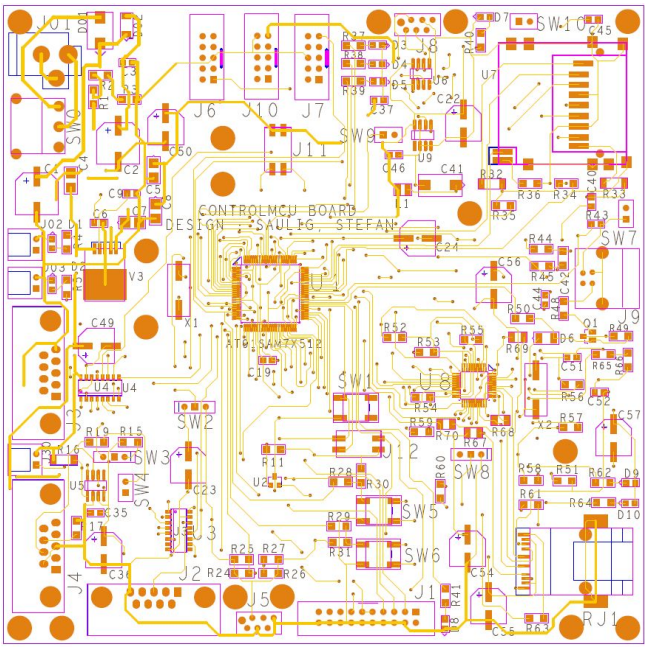


Abbildung 26 : TOP Design ARM7 Board [3]

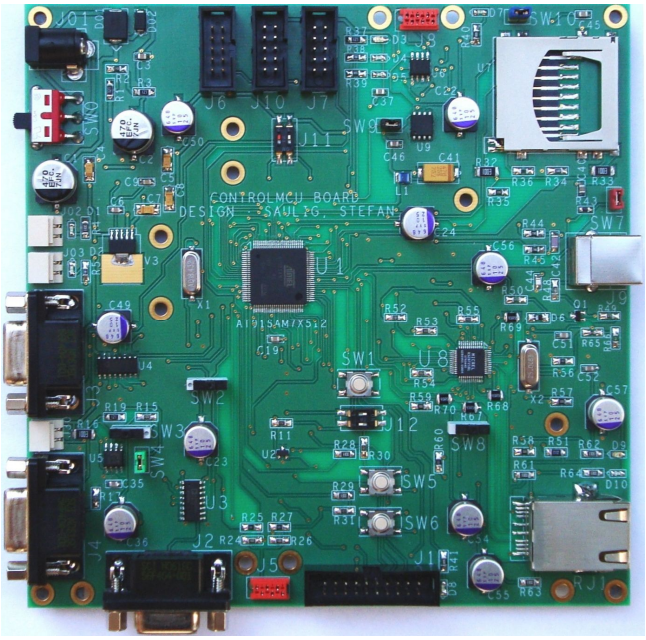


Abbildung 27 : TOP ARM7 Board [3]

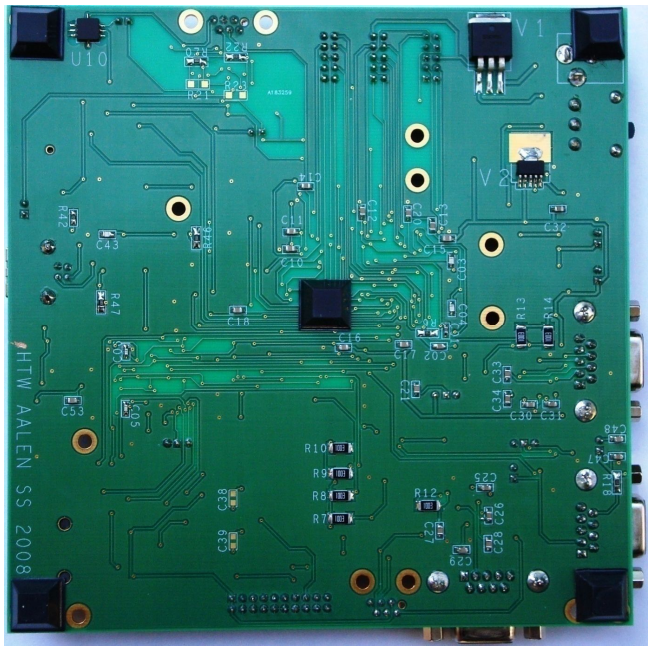


Abbildung 28 : BOT ARM7 Board [3]

Der verwendete AT91SAM7X512 [4] 32-Bit Controller hat nachfolgende Hauptmerkmale:

- 512KByte Flash Speicher
- 256 KByte SRAM
- JTAG-ICE Interface
- 802.3 Ethernet MAC 10/100
- CAN Controller
- 55 MHz bei 1.65V
- USB 2.0 Full Speed (12 Mbit/s)
- SPI, TWI, ADC, UART
- WDT, AIC, PIOA, PIOB, PMC, DMA
- SAM-BA (AT91-ISP)
- Security Protection

Abbildung 26 bis 29 verdeutlichen das entwickelte ARM7 Board und die realisierten Systemschnittstellen.

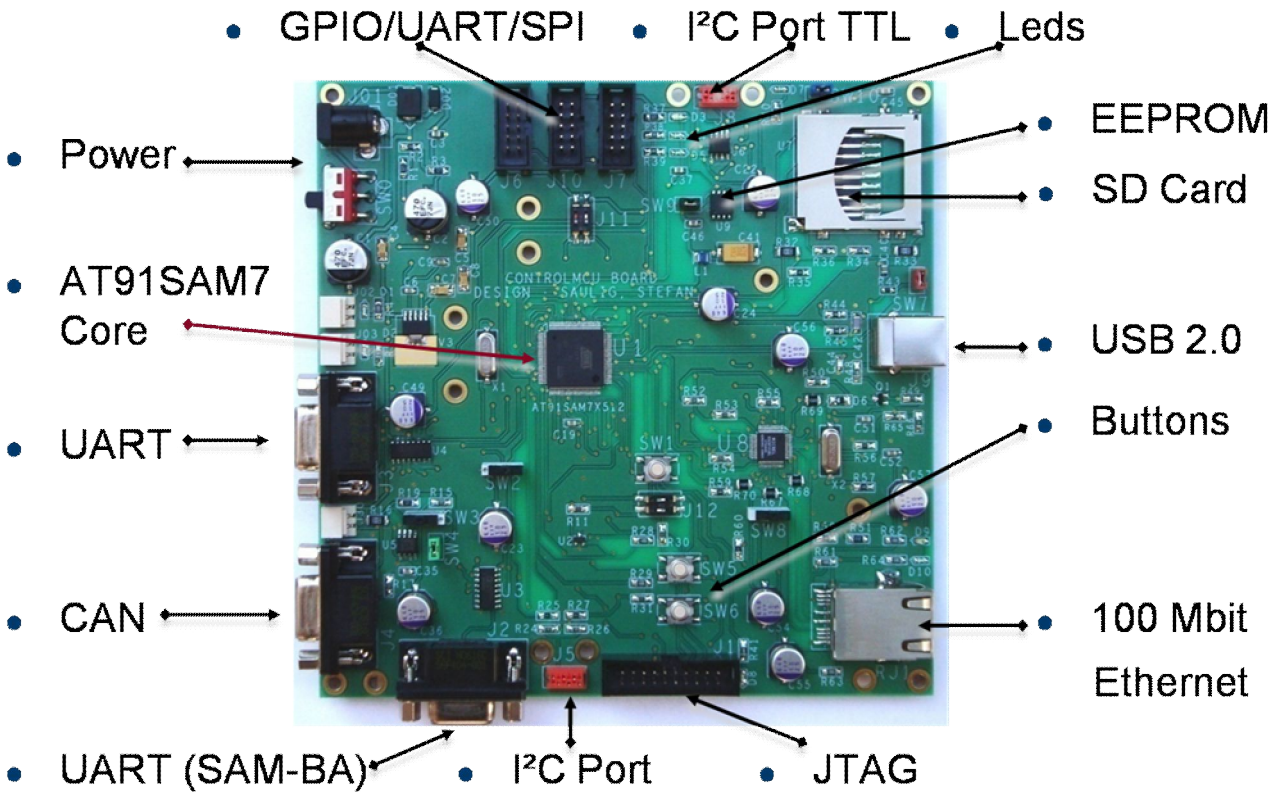


Abbildung 29 : TOP ARM7 Board Komponenten [3]

3. On-Board JTAG Interface

Die Software IDE benötigt ein entsprechendes Hardwareinterface zur Kommunikation mit dem JTAG Interface des Controllers. Hier ist die Realisierung eines Low-price JTAG Programmer/Debugger basierend auf den FTDI FT2232C [2] möglich. Nachfolgend aufgeführte Hauptmerkmale des ICs verdeutlichen die Möglichkeit einer einfachen Integration des JTAG für On-Board Entwicklungen.

- Umsetzung USB zu JTAG
- Support durch OpenOCD
- Unterstützung für SPI und I²C
- einfachste Systemanbindung durch Bibliotheken
- fertige DLL zur Implementierung
- fertige Gerätetreiber
- Multi-Protocol Synchronous Serial Engine (MPSSE)
- sehr geringe Kosten
- einfache Implementierung in eigene Designs

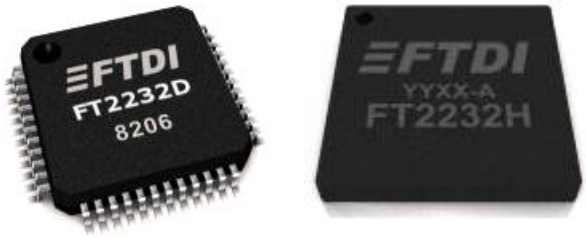


Abbildung 30 : FTDI FT2232D/H [2]

4. Development System Kosten

Die aufgezeigte Konzeption für die GNU IDE ist kostenlos. Ein selbstentwickeltes Board mit On-Board JTAG (FT2232C) ist für circa 80 Euro realisierbar. Ein Leiterplattendesign ist zweilagig realisierbar. Es entsteht kein Kostenzuwachs bei Mehrplatz-ausstattung. Die wesentliche Kosteneinsparung besteht durch die Verwendung eines On-Board JTAG und der kostenlosen IDE.

600 bis 1300 Euro sind für Development Board und IDE von Markenführern anzusetzen. Die Ausstattung mehrere Arbeitsplätze ist kostspielig und die Kosten steigen mit jedem zusätzlichen Arbeitsplatz an. Kostenanwuchs durch externes JTAG Interface und teure IDE mit Mehrplatz-Lizenzen entstehen.

5. Literatur

[1] Olimex.com - OLIMEX Ltd., 89 Slavjanska St., P.O.Box 237, Plovdiv 4000, BULGARIA

[2] Ftdichip.com - Future Technology Devices International Limited 373 Scotland Street, Glasgow, G5 8QB, United Kingdom

[3] Stefan Saulig - Projektarbeit SS 2008, Hochschule Aalen, Fakultät Elektronik und Informatik, Beethovenstr. 01, 73430 Aalen

[4] Atmel.com - Atmel Corporation, 2325 Orchard Parkway, San Jose, California 95131, USA

FPGA Implementierung der Linkadaption für ein OFDM-Modem

Dipl.-Ing. (FH) Andreas Utz

Telefunken RACOMS, Eberhard-Finckh-Strasse 55, 89075 Ulm
RWH Künzelsau, Daimlerstrasse 35, 74653 Künzelsau

andreas.utz@utz-elektro.de

Im Rahmen eines Entwicklungsvorhabens bei Telefunken RACOMS wird ein echtzeitfähiges OFDM-Modem für die Funkversorgung von Bahnsystemen entwickelt. Hierbei wurde die Implementierung einer Linkadaption auf dem FPGA des Modems als Diplomarbeit ausgeschrieben.

Als Grundlage für die Diplomarbeit stand ein Basissystem das eine Übertragung von IP-Paketen über 5,8 GHz ermöglichte, zur Verfügung. Ziel der Diplomarbeit war eine Erweiterung der Modemfunktionalität, speziell die Einbindung einer Kanalcodierung sowie die adaptive Wahl der Modulation über ein Feedback der Empfangsqualität.

1. Einleitung

Das Modem ist auf einem FPGA implementiert. Design, Simulation und Code- Generierung erfolgen mit dem DSP-Builder von Altera in einer MATLAB/Simulink-Umgebung, Synthese und Fitting in Quartus II. Für die Kanalcodierung konnte auf fertige IP-Cores zurückgegriffen werden. Für die adaptive Modulation ist die Empfangsqualität auszuwerten. Der Algorithmus für die Entscheidung sowie die Generierung der Header für das Feedback können auf dem Embedded Prozessor (NIOS-II) in C implementiert werden. Zum Test der adaptiven Modulation auf der Hardware-Plattform konnte ein Kanal-Emulator verwendet werden.

2. Verwendete Hardware

Die Hardware des OFDM-Modems MT5800 von Telefunken RACOMS setzt sich aus einem Embedded PC, dem FPGA[1] und der Sende- und Empfangseinheit (Frontend) zusammen. Der Aufbau ist in Abbildung 1 schematisch dargestellt.

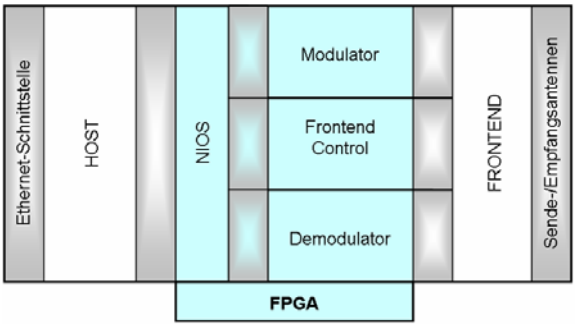


Abbildung 1: Schematischer Aufbau der Hardware

Der Embedded PC (Host) dient mittels der Ethernetschnittstelle zur Kommunikation mit dem vorhandenen Netzwerk. Über PCI kommuniziert er mit dem Soft-Core Prozessor auf dem FPGA, dem sog. NIOS. Darüber hinaus beinhaltet der FPGA den Modulator und den Demodulator. Diese Verarbeiten die zu sendenden oder empfangenen Daten. Hierbei können durch Modulator und Demodulator Übertragungen mit 4FSK[2], 2DPSK[3], 4DPSK und 8DPSK Modulation realisiert werden.

3. Konzept der Linkadaption

Das grundlegende Ziel einer Linkadaption ist die Optimierung der Datenübertragung unter den Randbedingungen der aktuellen Eigenschaften des Kanals. Es soll bei einem guten Kanal eine höhere Datenrate erreicht werden und bei einem gestörten Kanal eine sichere Übertragung.

Um eine Linkadaption zu realisieren, mussten Kriterien gefunden werden, anhand derer eine Aussage über die Eigenschaften des Kanals und die Qualität der Übertragung getroffen werden konnte. Mit Hilfe dieser Kriterien wird ein geeigneter Transmission-Mode gewählt.

[1] FPGA – Field Programmable Gate Array

[2] FSK – Frequency Shift Keying
[3] DPSK - Differential Phase Shift Keying

3.1. Übermittlung des Transmission-Modes

In Abbildung 2 ist die Kommunikation zwischen 2 Modems schematisch dargestellt. Im Empfänger des Modem II wird anhand des empfangenen Signals vom Sender des Modem I der LA-Mode[4] berechnet.

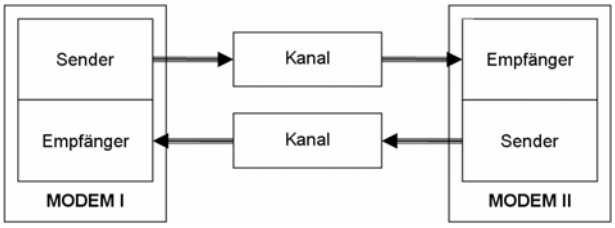


Abbildung 2: Kommunikation zwischen Sender und Empfänger

Dieser LA-Mode wird an den Sender des Modem II übergeben und an den Empfänger im Modem I übermittelt. Der LA-Mode wird in den Header des TTIs[5] geschrieben.

Im Empfänger des Modem I wird die Header-Information und damit auch der LA-Mode ausgewertet. Der LA-Mode kann nun vom Empfänger im Modem I als Transmission-Mode für den Sender im Modem I gesetzt werden.

3.2. Interne Übergabe des Transmission-Modes

In Abbildung 3 sind die Schnittstellen des Softcore-Prozessors (NIOS) mit dem Modulator und dem Demodulator abgebildet.

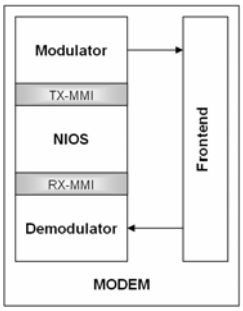


Abbildung 3: Schnittstellen Modulator, NIOS und Demodulator

Die Auswertung der letzten Übertragung wird im Demodulator durchgeführt. Der hierbei errechnete Wert wird an den NIOS übergeben. Der NIOS entscheidet über den daraus resultierenden Transmission-Mode für die nachfolgenden Pakete, die über diesen Zweig gesendet werden. Die Übergabe des Headers erfolgt vom NIOS an den Modulator.

Nach der Übermittlung dieses Paketes wird im Empfänger des anderen Modems der LA-Mode aus dem Header ausgelesen. Für die weitere Übertragung wird der LA-Mode als Transmission-Mode verwendet.

In beiden Übertragungsrichtungen sind unterschiedliche Transmission-Modes möglich.

3.3. Kriterium zur Ermittlung der Qualität des Übertragungskanals

Die Grundlage einer Linkadaption bildet das Kriterium nach dem die Wahl des Modulationsverfahrens entschieden wird. Unterschiedliche Eigenschaften des Übertragungskanals und Indikatoren aus dem Demodulator können hierfür verwendet werden.

Die Berechnung des Phase-Error (Phasenabweichung) erfolgt durch die Berechung der Phasenabstände der empfangenen Phasen zu den zu erwartenden Referenzphasenwerten. Hierbei werden die Phasenwerte der Übertragung direkt vor der Demodulation verwendet, siehe auch Abbildung 4.

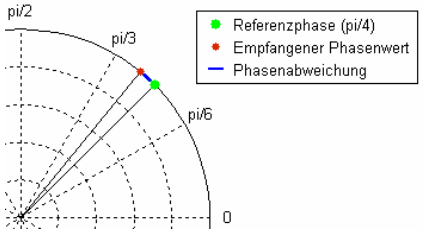


Abbildung 4: Phase-Error-Detection

Es wird aus den berechneten Phasenabweichungen $\Delta\varphi_A$ der Mittelwert $\tilde{m}_A$ und anschließend die Varianz σ_θ^2 ermittelt.

Es gilt:

$$\sigma_\theta^2 = \frac{1}{n}\sum_{i=0}^{n-1}\left|\Delta\varphi_{A,i} - \tilde{m}_A\right|^2$$

$$\tilde{m}_A = \frac{1}{n}\sum_{i=0}^{n-1}\Delta\varphi_{A,i}$$

Gemittelt über die Unterträger und die Anzahl an Datensymbolen pro TTI kann mit der Varianz des Phasenrauschens eine Aussage über die Qualität der Übertragung getroffen werden.

Es ist zu berücksichtigen, dass bei der Phase-Error-Detection nicht alle Störungen erkannt werden. Bei gravierenden Störungen in der Übertragungsstrecke, können Phasenab-

[4] LA-Mode – Link-adaption Transmission Mode
[5] TTI – Transmission Time Interval

weichungen auftreten, deren Betrag größer ist, als die Hälfte des Winkels zwischen zwei Referenzphasenwerten.

Der Detektionsalgorithmus ordnet diese dem nächsten Referenzwert zu und erkennt einen falschen Phasenabstand. Wenn dieser Fehler vermehrt auftritt, ist die Übertragung so stark beschädigt, dass eine korrekte Auswertung der Daten nicht mehr möglich ist. Die Linkadaption muss vor Auftreten derartiger Störungen agieren.

Bei der Varianzberechung des Phasenrauschens ist das Signal- zu Rausch- und Interferenzverhältnis berücksichtigt, da je größer das SINR desto kleiner die Phasenabweichung und deren Varianz.

Um Schwellwerte für die Varianz des Phasenrauschens festlegen zu können, wurde die in Abbildung 5 dargestellte Kurve verwendet werden.

In dem Diagramm wurden Wahrscheinlichkeitsdichtefunktionen der Schätzwerte der Phasenrauschvarianz bei unterschiedlichen BER-Werten für die Übertragungsmodi 2PSK, 4PSK und 8PSK dargestellt.

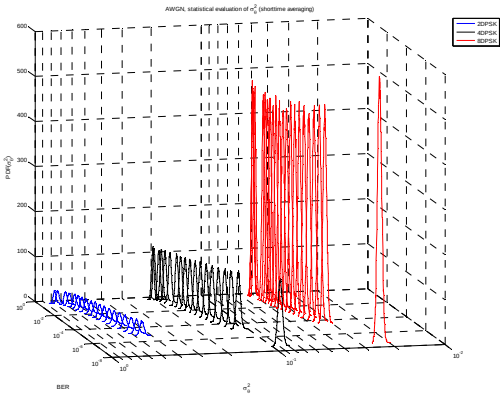


Abbildung 5: BER über Varianz des Phasenrauschens und über Wahrscheinlichkeitsdichtefunktion der Varianz

Quelle: Linkadaption – Alexander Linduska Telefunken RACOMS

Mit der Wahrscheinlichkeitsdichtefunktion bei einem gegebenen BER kann eine Aussage getroffen werden, mit welcher Wahrscheinlichkeit die Varianz des Phasenrauschen in einem gewissen Bereich liegt.

Im Umkehrschluss kann man aus dieser Grafik das wahrscheinliche BER einer gewissen Varianz des Phasenrauschens ermitteln.

4. Realisierung der Linkadaption

Für die Realisierung der Linkadaption musste im Modulator der Wechsel des Modulationsverfahrens während einer Übertragung ermöglicht werden und die Routine für den Start der Datenübertragung angepasst werden.

Die Abbildung 6 zeigt den grundlegenden Aufbau des Demodulators.

Im Verlauf der durchgeführten Arbeit wurde der Channel Emulator zur künstlichen Verschlechterung des Kanals verwendet.

Im Demodulator, parallel zu Errechung der Softbitinformation, ist die Berechnung der Varianz, die zur Bewertung der aktuellen Übertragungsqualität verwendet wird, implementiert.

In Abbildung 7 ist die schematische Berechnung der Varianz dargestellt.

Mit Hilfe der empfangenen Phasenwerte werden die Referenz-Phasenwerte bestimmt. Diese werden von den empfangenen Phasenwerten subtrahiert und der Betrag darüber gebildet. Anschließend werden die Abweichungen quadriert und eine Aufsummierung über ein TTI durchgeführt. Der errechnete Wert PhaseErrVar2 wird an den Softcore-Prozessor übergeben, der eine Normierung des Wertes durchführt. Hierbei wird eine Division durch die Anzahl der aufsummierten Phasenwerten durchgeführt.

Bei der Varianzberechung des Phasenrauschens, ist der berechnete Wert unabhängig vom aktuell verwendeten Modulationsverfahren. Somit kann zwischen

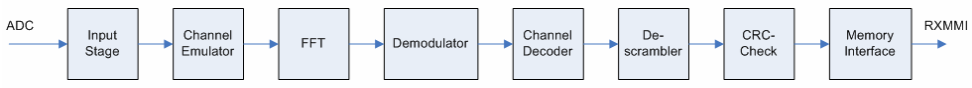


Abbildung 6: Schematischer Aufbau Demodulator

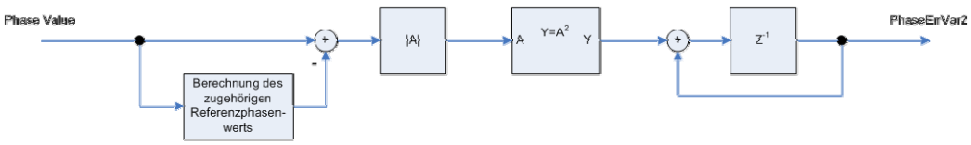


Abbildung 7: Schematische Darstellung der Varianzberechung

den unterschiedlichen DPSK Modulationen unabhängig von der aktuell verwendeten Modulation geschaltet werden.

Die Schaltschwellen, die hierfür verwendet werden, sind in einer XML-Datei hinterlegt und werden vom Host bei der Initialisierung an den Softcore-Prozessor übergeben.

Der durch die Schwellwertentscheidung festgelegte LA-Mode wird direkt als AHI[6] an den Header angefügt.

5. Messung der Eigenschaften der Linkadaption

Die Funktion der Linkadaption wurde in Matlab simuliert.

Die Simulation wurde im AWGN-Kanal[7] durchgeführt. In diesem Kanal wird zu dem Sendesignal ein weißes gaußsches Rauschen addiert. Für die Übertragung wurde eine Anzahl von 2500 TTIs gewählt mit jeweils 23 Datensymbolen. Die in der Simulation verwendete Varianz wurde gemittelt über die Anzahl der Symbole pro TTI.

Die Varianz des Phasenrauschens und die BER-Werte wurden für ein $\frac{E_b}{N_0}$ von 0 bis 20 dB errechnet.

Anschließend wurden die $\frac{E_b}{N_0}$-Werte in das dargestellte SNR [dB] umgerechnet.

Die Linkadaption kann zwischen den Transmission-Modes 4FSK, 2DPSK, 4DPSK und 8DPSK adaptieren.

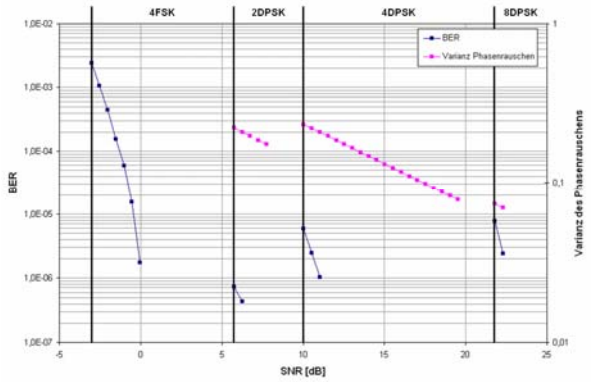


Abbildung 8: Ergebnis der Messung der Linkadaption

Das in Abbildung 8 dargestellte Diagramm zeigt die Adaption der Transmission- Modes anhand der ermittelten Varianz. Die verwendeten Transmission-Modes sollen, wenn das jeweilige BER eine Grenze von 10^{-5} erreicht, gewechselt werden. Die Schwellen der Phasenvarianz sind so angepasst, dass diese Grenze nicht überschritten wird. Das in der Abbildung blau aufgezeichnete BER ist dem jeweiligen Transmission-Mode zuzuordnen.

6. Zusammenfassung und Ausblick

In der Diplomarbeit wurde ein Konzept zur Erweiterung der Modemfunktionalität um eine Linkadaption erstellt und diese in die Hardware des Modems MT5800 von Telefunken RACOMS implementiert. Dabei wurde anhand der Varianz des Phasenrauschens die Qualität des Übertragungskanals ermittelt und in gegebenem Fall das Übertragungsverfahren angepasst.

Ein Wechsel des Übertragungsverfahrens aufgrund der aktuellen Eigenschaften des Kanals wird im empfangenden Modem entschieden. Das empfangende Modem übermittelt den neuen Transmission-Mode als zusätzliche Header-Information mit dem nächsten Paket zum Sender. Der Sender wechselt bei den nachfolgenden Paketen das Übertragungsverfahren.

Anhand der Simulationen konnte dieses Konzept verifiziert werden.

Die durchgeführten Versuche mit der auf der Hardware implementierten Linkadaption zeigten, dass unter Laborbedingungen im Digital Loop, dieses Konzept funktioniert.

Als nächstes müssen Tests in einer realen Übertragung beispielsweise im Tunnel durchgeführt werden. Es muss besonders das Verhalten beim Wechsel des Übertragungsverfahrens betrachtet werden. Diese Wechsel dürfen nicht zu Fehlern in der zu übertragenden Information führen.

Die implementierte Linkadaption kann die Transmission-Modes auf Paket-Ebene wechseln. Da sich die Eigenschaften des Kanals nicht derart sprunghaft ändern, könnte ein Algorithmus implementiert werden, der unter Berücksichtigung der gemessenen Varianz des Phasenrauschens, eine Vorhersage über die zu erwartende Qualität des Kanals durchführt.

Um einen solchen Algorithmus realisieren zu können, müssen die Messwerte über einen bestimmten Zeitraum betrachtet werden. Dadurch verliert die Linkadaption an Reaktionsgeschwindigkeit. Die Vorteile der Vorhersage und der dafür in kauf genommene Reaktionsverlust, müssten dann, anhand von Messungen abgewogen werden.

[6] AHI – Addidional Header Information
[7] AWGN – dt. Additives weißes gaußsches Rauschen

Realisierung eines FPGA-basierten Echtzeitdifferenzbildsensors für Verkehrsassistenzsysteme

Thomas Duttine, Konrad Doll

Hochschule Aschaffenburg

thomas.duttine@fh-aschaffenburg.de

In Verkehrsassistenzsystemen muss sehr häufig die Bewegung eines Objekts (z.B. Fahrzeug oder Fußgänger) erkannt werden. Dazu kann in einem intelligenten Bildverarbeitungssystem der Unterschied zweier Bilder – das sogenannte Differenzbild – benutzt werden. Auf einem herkömmlichen Rechnersystem benötigt diese Erstellung jedoch verhältnismäßig viel Rechenzeit. Aufgrund von Aspekten der Verkehrssicherheit muss die Objektbewegung allerdings sehr schnell erkannt und darauf reagiert werden. Durch die Auslagerung der Differenzbilderstellung auf einen FPGA ist es möglich, diese in Echtzeit, d.h. gleichzeitig mit dem Eingang der Videodaten, zu verwirklichen. Dadurch kann der Hauptprozessor merklich entlastet werden, was wiederum den Einsatz kleinerer und energiesparenderer Systeme ermöglicht oder dem Prozessor mehr Ressourcen für andere Aufgaben gibt.

1. Einführung

Die intelligente Bildverarbeitung hat heutzutage eine wichtigere Rolle denn je zuvor. Schon heute erlaubt es diese Technologie, Produktionsabläufe zu vereinfachen, indem z.B. Gegenstände vollautomatisch erkannt und anschließend sortiert werden können. Intelligente, personenerkennende Sicherheitssysteme erlauben eine effizientere Überwachung sicherheitsbedürftiger Bereiche und seit kurzem feiern diese Systeme auch ihren Einzug in das Straßenverkehrsgeschehen z.B. in Form eines Lane Departure Warning Systems[1].

Jedoch sind noch lange nicht alle Möglichkeiten dieser Technologie ausgeschöpft. So ist beispielsweise durch eine Technik zur Detektion und Klassifikation von Fahrzeugen eine Überwachung von Kreuzungsanlagen möglich, um deren Ampelschaltung dynamisch zu koordinieren oder sogar um Kollisionen mehrerer Fahrzeuge untereinander und mit Fußgänger oder Fahrradfahrer zu vermeiden.

Dies ist deswegen von besonderem Interesse, da etwa 25% aller Unfälle mit Verletzten an Kreuzungen und deren Umfeld auftreten. Etwa die Hälfte der Unfälle im Kreuzungsbereich ließe sich durch eine frühere Reaktion verhindern[2].

Allerdings benötigen viele der momentan existierenden Systeme enorme Rechenleistungen, um die umfangreiche Informationsvielfalt bearbeiten zu können. Dies konkurriert jedoch mit der Vorstellung, solche Anlagen klein und modular zu entwerfen, stromsparend zu betreiben, kostengünstig zu produzieren und vor allem echtzeitfähig zu gestalten.

Diese Anforderungen lassen sich erfüllen, wenn beispielsweise bestimmte rechenintensive Vorgänge auf externe Bausteine, wie z.B. FPGAs, ausgelagert werden. Diese Bausteine besitzen aufgrund ihrer Architektur die Möglichkeit, Abläufe parallel zu gestalten. Somit kann durch Kombination eines FPGAs und eines kleineren Mikroprozessors die Realisierung eines solchen Systems mit den oben genannten Kriterien erfolgen.

Das Ziel dieses Projektes bestand daher aus der Entwicklung und hardwaremäßigen Implementierung eines Differenzbildsensors, welcher einen Teil eines Kreuzungsassistenzsystems bilden soll. Dieser soll mithilfe von VHDL und Verilog Hardwarebeschreibungssprachen in ein FPGA-System implementiert werden.

Wie der Name sagt, liegt die Aufgabe eines Differenzbildsensors in der Erstellung eines Differenzbildes aus zwei unterschiedlichen Bildern. Hierbei werden die einzelnen, übereinander liegenden Pixelwerte der beiden Bilder voneinander subtrahiert, wodurch die Unterschiede zwischen den beiden Frames sichtbar werden. Angenommen beide Bilder sind absolut identisch, so ergibt das Differenzbild ein schwarzes Bild.

In der modernen Bildverarbeitung hat dieses Verfahren einen hohen Stellenwert, da hierdurch leicht überflüssige und unerwünschte Informationen aus den Bildern entfernt werden können. Weiterhin ist so die Detektion von Änderungen (speziell von bewegten Objekten) möglich.

In bisherigen Anwendungen erfolgt die Differenzbilderstellung jedoch hauptsächlich in prozessorbasierenden Systemen. Diese benötigen für ein NTSC-aufgelöstes Video ca. 750 Millionen Rechenoperationen pro Sekunde (720 * 487 Pixel * ca. 36 Rechenoperationen pro Pixel * 60 Frames pro Sekunde). Dies würde einen 1 GHz Prozessor ungeachtet der Tatsache, dass noch andere Rechenoperationen zusätzlich benötigt werden, schon zu 75% auslasten.

Die Notwendigkeit, diese Operation rein aus Geschwindigkeitsgründen auf einen externen Baustein (FPGA) auszulagern, wurde zwar schon in diversen Tools erkannt und implementiert, jedoch war zum Zeitpunkt der Erstellung dieser Arbeit kein frei verfügbarer und modular einsetzbarer Differenzbildsensor mit offenen Schnittstellen zur Implementierung in eigene FPGA Systeme erhältlich.

2. Anforderungsanalyse

Für die Planung und den Aufbau eines Prototyps, bzw. eines Entwicklungssystems für den Einsatz in einem Verkehrsassistenzsystem, musste zunächst eine Anforderungsanalyse klären, welche Mindest-anforderungen an das System gestellt werden.

Die Anforderungen an die Videoquelle behandeln in erster Linie die benötigte Mindestauflösung, um eine Detektion von Fahrzeugen zu gewährleisten. Weiterhin wird hier noch über die Framerate die Genauigkeit der Detektion mit einbezogen.

Die Anforderungen an das System beinhalten hauptsächlich die benötigte Größe des FPGAs und des Speichers. Weiterhin wird hier noch auf die minimal nötige Datenrate zwischen dem Speicher und dem FPGA eingegangen.

Die gesamte Analyse bezieht sich auf die Anwendung eines kompletten Erkennungssystems, da dieses Projekt in der Regel einen Teil eines solchen Systems darstellt.

2.1 Berechnung der Mindestauflösung

Betrachtet wird hierbei eine Kreuzung aus einer Perspektive, welche der späteren ähnlich kommt. Es wird ein Feld mit den Maßen von ca. 20 m * 60 m von einer Seite aus und aus einem erhöhten Stand beobachtet.

In dieser Annahme hat ein zu detektierendes Fahrzeug innerhalb der ROI (Region of Interest) ungefähr ein Verhältnis von 10% zur Breite des Gesamtbilds und ca. 8% zur Höhe des Gesamtbilds.

In Abbildung 1 ist dieser Zusammenhang gezeigt:

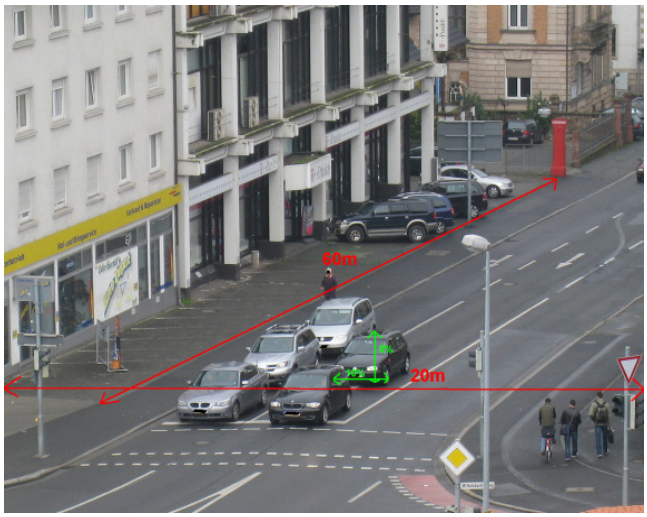


Abbildung 1: Der Zusammenhang zwischen der Größe eines Fahrzeugs und dem Bildausschnitt

Weiterhin wird in dieser Annahme das Fahrzeug aufgrund der Perspektive vereinfacht als Raute angenommen. Nachfolgend erfolgt die Berechnung der Fahrzeugfläche im Bild bei einer Auflösung von 720*487 Pixel (dies entspricht der sichtbaren Fläche des NTSC-Standards, welcher als Quelle für dieses Projekt infrage kommt):

$$100\% = 720\,px$$
$$\Rightarrow 10\% \cong 72\,px$$

$$100\% = 487\,px$$
$$\Rightarrow 8\% \cong 39\,px$$

Gemäß den oben festgestellten Verhältnissen berechnen sich die beiden Kantenlängen des betrachteten Fahrzeugs somit zu 72 und 39 Pixel.

Hieraus lässt sich die Objektfläche in Pixel ermitteln:

$$Objektfläche = 72\,px * 39\,px$$
$$= 2808\,px$$

Diese Objektfläche reicht für die erfolgreiche Detektion eines Objektes erfahrungsgemäß aus.

2.2 Berechnung der Framerate

Um die minimal benötigte Framerate für das Videosystem berechnen zu können, wurde eine Abschätzung aus dem Buch „Fahrzeugsicherheit: Personenwagen“ von Ulrich Seifert[3] herangezogen, die besagt, dass 96% aller Unfälle im Straßenverkehr bei Geschwindigkeiten unter 56 km/h (= 15,56 m/s) geschehen. Aufgrund dieser Annahme und der Forderung, dass ein weiterverarbeitendes System theoretisch eine Detektionsgenauigkeit von unter einem Meter erreichen soll, wurde mit der folgenden Berechnung die Framerate ermittelt:

$$Framerate = \frac{Geschwindigkeit}{Genauigkeit}$$

$$Framerate = \frac{15{,}56 m/s}{1m}$$

$$\underline{\underline{Framerate = 15{,}56\, fps}}$$

Bei diesen Überlegungen wird angenommen, dass keine Bewegungsschätzung vorgenommen wird.

Die Betrachtung stützt sich hierbei auf die Tatsache, dass sich das Objekt nicht sprunghaft bewegen bzw. seine Geschwindigkeit verändern kann.

2.3 Ergebnis der Anforderungsanalyse

Von der optischen Seite aus betrachtet erwiesen sich die berechneten Anforderungen als keine große Hürde. Bei der angenommenen NTSC-Auflösung (720*487 Pixel) und der Annahme eines geeigneten Videofeldes existiert eine genügend große Objektfläche, welche eine Detektion mit gängigen Algorithmen ermöglichen sollte.

Eine höhere Auflösung vergrößert natürlich die Zahl der Pixel für jedes zu detektierende Objekt, erhöht aber gleichzeitig auch die Anzahl der notwendigen Berechnungen. So sind beispielsweise bei einer NTSC-Auflösung 350.640 Bildpunkte und bei einer XGA-Auflösung (1.024*768 Pixel) schon 786.432 Bildpunkte zu berechnen, was einem Faktor von 2,24 entspricht.

Aufgrund der gegebenen Marktumstände ist hier zumindest im ersten Schritt eine kostengünstige Standard NTSC (oder PAL) Kamera empfehlenswert. Diese liegt mit ihrer Auflösung von 720*487 Pixel (350.640 Bildpunkte) im Mittelfeld der verfügbaren Systeme und besitzt damit genügend Reserven. Bei diesem Verfahren liegt die Bildwiederholrate bei 50 Hz mit Halbbildern, was 25 Vollbildern pro Sekunde entspricht. Hiermit wäre auf jeden Fall die Forderung von unter einem Meter Genauigkeit erfüllt. Bei Geschwindigkeiten unter 46 km/h ergibt sich sogar eine theoretische Genauigkeit von unter 0,5 m.

Weiterhin spricht für ein NTSC-System, dass viele auf dem Markt erhältliche Kamerasysteme diesen Standard unterstützen.

Außerdem kann vor allem im ersten Entwicklungsabschnitt die Kamera durch einen Computer oder einen Videorecorder simuliert werden, was Optimierungen und Entwicklungen, welche reproduzierbare Quelldaten benötigen, erst ermöglicht.

Für die Größe des FPGAs konnten erfahrungsbedingte Schätzwerte herangezogen werden. Diese stützen sich hierbei auf den Ressourcenbedarf ähnlicher Teilsysteme, welche zusammengerechnet ergaben, dass mindestens ein Virtex 2 ab Modell 2VP4 (6.768 Logikzellen) oder ein Spartan 3 FPGA ab Modell XC3S400 (8.064 Logikzellen) eingesetzt werden sollte.

Bei einem Blick auf die aktuell auf dem Markt verfügbaren Evaluationsboards zeigte sich jedoch schnell, dass diese Anforderungen keinerlei Problem darstellen, da die meisten geeigneten Systeme diese bei weitem erfüllten.

3. Hardware

Für die Implementierung des Systems wurde eine Kombination aus einem Digilent XUP-V2Pro Board[4] und einem ebenfalls von Digilent entwickelten externen Framegrabber VDEC1[5] gewählt. Zu Steuerung des Differenzbildsensors und für Testzwecke während der Entwicklungsphase wurde noch ein externes I/O-Erweiterungsboard entworfen, welches ebenfalls an das XUP-V2Pro Hauptboard angeschlossen werden kann. Die Komponenten sind entsprechend der Abbildung 2 miteinander verbunden und lassen sich in die folgenden Elemente gliedern:

- **Videodateneingang:** Hierfür ist hauptsächlich der Framegrabber VDEC1 zuständig.
- **Videodatenverarbeitung:** Diese Aufgabe übernimmt der FPGA-Chip auf der XUP V2Pro Hauptplatine.
- **Datenaus- bzw. -weitergabe:** Hierfür wird eine Kombination aus einem auf der Hauptplatine vorhandenen RAMDAC und einem externen VGA-Anzeigegerät benutzt.

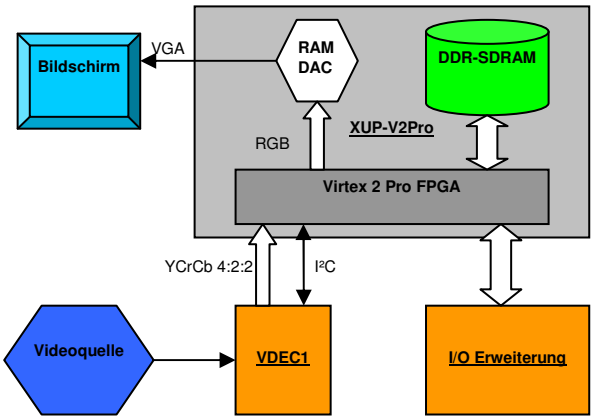


Abbildung 2: Der schematische Aufbau des Systems

3.1 Videodateneingang

Verantwortlich für den Videodateneingang ist, wie schon erwähnt, das VDEC1-Erweiterungsboard. Dieser Framegrabber erweitert das System um einen SVideo, einen Composite und einen Component Eingang zur Einspeisung von PAL, NTSC oder SECAM Signalen. Dadurch existiert eine große Anzahl an möglichen Videoschnittstellen für das System, da es sich hierbei um die momentan gängigen Standards für digitale Videoübertragung handelt.

Die Hauptfunktion des VDEC1 Framegrabbers liegt in der Wandlung der extern eingespeisten Signale in ein für den FPGA geeignetes Format. Diese Aufgabe übernimmt in diesem Fall ein AD7183B Chip von Analog Devices. Die Übertragung der Videodaten zu dem FPGA erfolgt in einem YCrCb 4:2:2 Format in TTL Pegel und wahlweise mit 8 Bit oder 16 Bit Datenbusbreite. Konfiguriert wird der Framegrabber Chip über einen I²C-Bus direkt von dem FPGA aus. Hierfür wurde ein entsprechendes Modul implementiert.

3.2 Videodatenverarbeitung

Das XUP V2Pro Entwicklungsboard enthält als Herzstück den Virtex 2 Pro FPGA vom Typ „XC2VP30“ mit 30.816 Logikzellen, welcher für die zentrale Bearbeitung der Videodaten bzw. die Erstellung des Differenzbildes zuständig ist. Für diese Aufgabe werden ebenfalls auf dem Board vorhandene Peripherieeinheiten wie der DDR-SDRAM für die Speicherung der Videodaten oder der Video RAMDAC mit angeschlossenem VGA-Anschluss für die Ausgabe des Differenzbildergebnisses benutzt.

Die Speicherung der Videodaten stellt ein zentrales und zugleich das wichtigste Element in der Erstellung des Differenzbildes dar. Da für die Erstellung jedes Differenzbildes der komplette Videoframe aus dem Speicher geladen werden muss, ist der Datendurchsatz hier entsprechend hoch. Zur Bewältigung dieser Aufgabe steht ein Standard DDR-SDRAM DIMM-Modul, welches baugleich auch in modernen PC-Systemen Einsatz findet, bereit. Bauartbedingt ergibt sich hierbei eine Datenbusbreite von 64 Bit bei einer variablen Speichergröße bis hin zu zwei Gigabyte. In diesem Projekt wurde ein 256 MB Modul der Firma AENEON mit einer Geschwindigkeit von 100 MHz eingesetzt.

3.3 Datenausgabe

Für die Visualisierung des berechneten Differenzbildes und für diverse Entwicklungszwecke wurde ein auf dem XUP V2Pro Board vorhandener RAMDAC-Chip der Firma Fairchild Semiconductor mit der Bezeichnung „FMS3818“ benutzt. Dieser wandelt die drei mit bis zu 8 Bit digital eingespeisten Grundfarbsignale in ein dem Computer-VGA-Standard entsprechendes Analogsignal. Neben den Farbleitungen werden noch die entsprechenden Synchronisationssignale über den RAMDAC ausgegeben. Die Signale werden ebenfalls direkt auf dem FPGA erzeugt und über Koppelwiderstände an den RAMDAC bzw. die VGA-Buchse ausgegeben. Zur Anzeige kann jedes VGA-kompatible Gerät verwendet werden.

Neben der Ausgabe auf einem externen Anzeigegerät ist natürlich auch die direkte Weiterverarbeitung des Videosignals (z.B. innerhalb des FPGAs) denkbar.

4. Implementierung

Wie schon beschrieben, stellt das FPGA das zentrale Element des Projektes dar. Hier geschieht die eigentliche Berechnung und Erstellung des Differenzbildes auf Hardwareebene. Neben dieser Funktionalität übernimmt das FPGA noch eine ganz Reihe weiterer Aufgaben, welche zur Vor- und Nachbearbeitung der Videodaten oder zur Steuerung externer Komponenten notwendig sind.

4.1 Datenfluss

Bevor auf die genaue Systemarchitektur eingegangen wird, soll zunächst der Datenfluss der Videosignale durch das System betrachtet werden.

Wie in Abbildung 3 zu erkennen ist, muss das von dem externen Framegrabber VDEC1 gelieferte Signal in einem ersten Schritt von dem YCrCb 4:2:2 Format

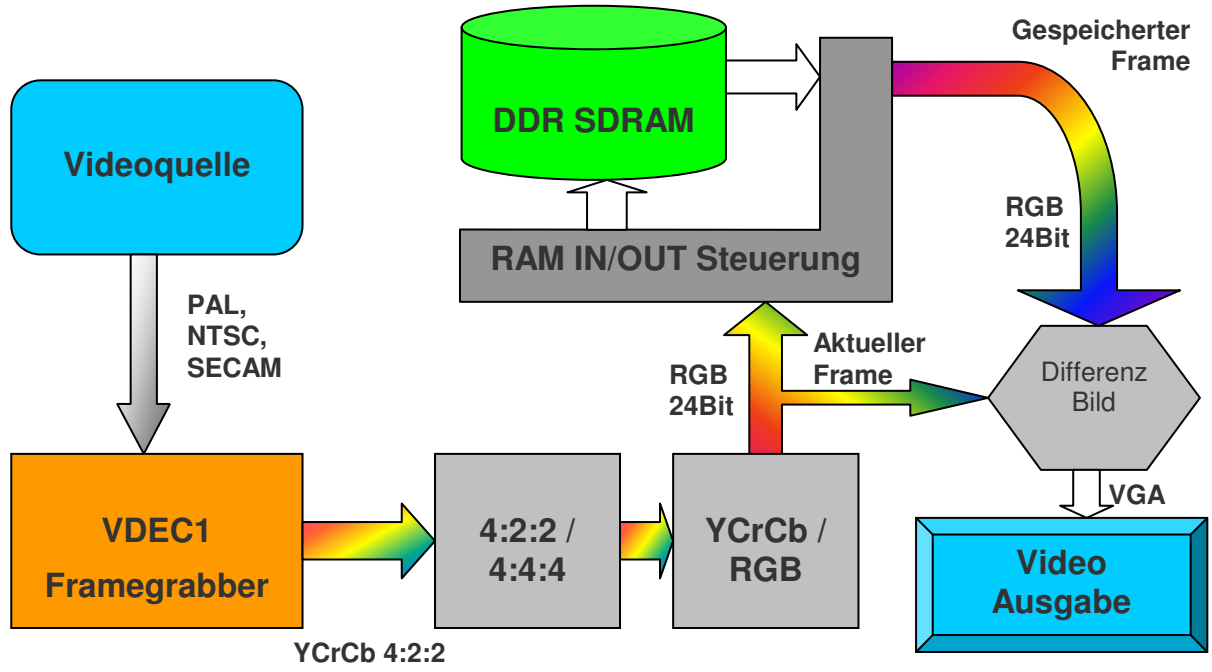


Abbildung 3: Der Datenfluss durch das System

in ein 24 Bit RGB-Signal (je 8 Bit pro Farbwert) gewandelt werden. Hierzu muss zunächst aus dem YCrCb 4:2:2 ein YCrCb 4:4:4 Signal erstellt werden. Da jetzt jedes Pixel einen eigenen Luminanz- und zwei eigene Crominanzanteile besitzt, kann nun mithilfe einer YCrCb-zu-RGB Wandlung diese vollzogen werden. Weiterhin werden aus dem von außen eingespielten Signal noch die gemäß ITU-R BT.656[6] zugefügten Informationen entnommen und ausgewertet. Die Wandlung in ein 24 Bit RGB-Format ist notwendig, da das restliche System aus Flexibilitätsgründen für die Verarbeitung von RGB-Signalen ausgelegt worden ist. Dies erleichterte unter anderem auch die Ausgabe der Videodaten während der Entwicklung auf ein externes Anzeigegerät. Außerdem erfolgt nach der erfolgreichen Bildung des Differenzbildes die Aus- bzw. Weitergabe des Signals ebenfalls in dem RGB-Format.

Das nun zur Verfügung stehende 24Bit RGB-Format wird an zwei unterschiedliche Systeme weitergegeben. Zum einen ist dies bereits der Differenzbildsensor, welcher diese Daten als eine Komponente des Differenzbildes benötigt, und zum anderen ist das die Speicherzugriffssteuerung.

Diese arbeitet in zwei verschiedenen Modi:

- **Auslesen des gespeicherten Bildes:** Diese Betriebsart stellt den Normalfall dar. Hierbei wird das in dem DDR-SDRAM hinterlegte Bild synchron mit jedem von außen eingespielten Bild wieder ausgelesen und als zweite Komponente an den Differenzbildsensor weitergegeben. Dieser kann nun aus den beiden synchronen Videosignalen das aktuelle Differenzbild in Echtzeit erstellen und dieses in einem dritten Videosignal nach außen, beispielsweise für eine Anzeige auf einem Monitor, weitergeben.
- **Speichern eines aktuellen Bildes:** Diese Betriebsart wird nur bei Bedarf oder je nach Einstellung in bestimmten zeitlichen Abständen aktiviert. Hierbei wird das Bild in den DDR-SDRAM Speicher geschrieben und dient als neues Referenzbild für den Differenzbildsensor.

4.2 Architektur

In diesem Kapitel wird die Implementierung des Systems auf dem FPGA beschrieben.

Das System gliedert sich in insgesamt vier Teilmodule, welche in einem übergeordneten fünften Modul mit dem eigentlichen Differenzbildsensor zusammengebunden werden. Dieses fünfte Modul stellt auch gleichzeitig das Systemmodul dar, in dem auch die einzelnen Schnittstellen des FPGAs definiert sind.

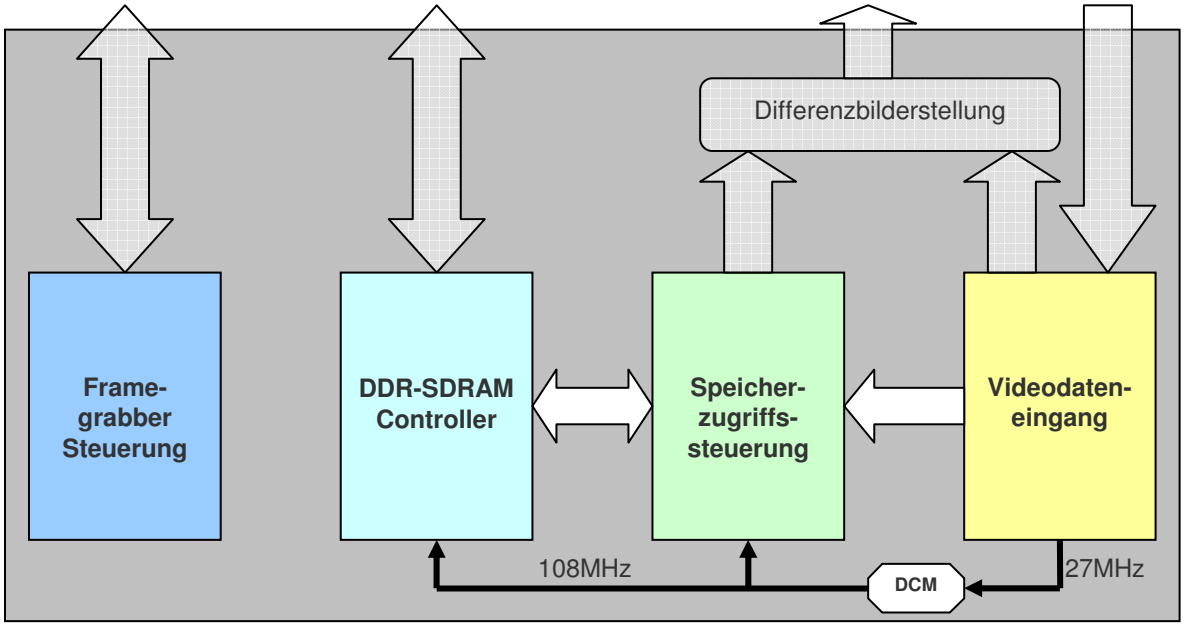


Abbildung 4: Die Systemarchitektur

Die einzelnen Module sind wie in Abbildung 4 dargestellt strukturiert und werden nachfolgend genauer erläutert.

Eine Besonderheit bildet hierbei der Systemtakt, welcher in diesem Fall aus dem Dateneingangstakt der Videosignale extrahiert wird. Ein digitaler Clock Manager (DCM) bereitet den Takt auf und generiert einen zu dem 27 MHz Dateneingangstakt synchronen 108 MHz Takt, welcher als Taktquelle für den DDR-SDRAM Controller der Speicherzugriffssteuerung dient. Hierdurch wird eine Synchronität bei der Weitergabe der Videodaten zwischen den einzelnen Modulen gewährleistet.

4.2.1 Systemmodul

Das Systemmodul ist das Grundgerüst des kompletten Systems. Alle weiteren Module sind hier als Komponenten deklariert und instanziiert. Darüber hinaus erfolgt hier auch die Verdrahtung der einzelnen Module untereinander oder nach außen.

Neben der Funktion als Grundgerüst enthält das Systemmodul noch den eigentlichen Differenzbildsensor. Dieser stellt zwar nur einen überraschend kleinen Teil des kompletten Projektes dar, jedoch liegt die eigentliche Komplexität in der Bereitstellung der geeigneten Videodaten. Der Sensor selbst ist rein kombinatorisch implementiert. Hierdurch wird die Bearbeitungsdauer und somit die Erstellung des Differenzbildes auf ein Minimum (Signallaufzeiten innerhalb der logischen Gatter) reduziert und ermöglicht somit den gewünschten Betrieb in Echtzeit.

Für die Differenzbilderstellung stehen diverse arithmetische Verfahren zur Verfügung. Klassisch ist z.B. das rein subtraktive Verfahren, das immer das niederwertigere Farbsignal von dem höherwertigeren subtrahiert. Hierzu werden die beiden Quellsignale (Videodaten aus dem Speicher und Videodaten von dem Eingang) zuerst mithilfe eines Komperators verglichen, um das höherwertigere ausfindig zu machen. Entsprechend dem Ergebnis wird über ein Subtraktionsgatter das Differenzbild erstellt. Dies wird für alle drei Farbsignalantiele durchgeführt.

Darüber hinaus gibt es beispielsweise noch Verfahren, welche ein Schwarz-Weiß-Bild anhand vorgegebener Schwellwerte der einzelnen Differenzen erstellen oder bestimmte Toleranzen aus dem Bild herausfiltern.

4.2.2 Framegrabber Steuerung

Das an sich eigenständige Steuermodul ist für die Konfiguration des AD7183B Chips auf dem VDEC1 Framegrabber Board zuständig. Hierfür ist innerhalb des Moduls ein einfacher I²C-Busmaster-Controller als Komponente eingebunden. Dieser Controller basiert auf einer Komponente von Opencores.org[7]. Die möglichen Funktionen beschränken sich hierbei auf einfache Schreib- und Lesezugriffe mit 100kHz Taktrate, jedoch ohne Multimaster- oder Slave-Fähigkeit. Darüber hinaus besitzt der Controller keine FIFO-Puffer für die zu sendenden oder empfangenden Daten, was bei der Erstellung des übergeordneten Steuersystems beachtet werden muss.

Da für die Konfiguration des AD7183B Chips jedoch nur einfache, einmalige Schreibzugriffe auf eben diesem Slave notwendig waren, erschien dieser Controller als die einfachste, schnellste und ressourcensparendste Möglichkeit, diese Aufgabe zu bewältigen.

Innerhalb des Systemmoduls befindet sich ein einfacher Zustandsautomat zur Ansteuerung des I²C-Controllers, welche speziell auf die benötigten Schreibzyklen des Framegrabber Chips angepasst wurde. Die zu konfigurierenden Register und die zugehörigen Daten sind neben dem Zustandsautomat ebenfalls fest in dem Systemmodul hinterlegt. Die Konfiguration selbst läuft hierbei komplett selbstständig ab, was somit einen Eingriff von außen nicht notwendig macht.

4.2.3 DDR-SDRAM Controller

Bei dem DDR-SDRAM Controller handelt es sich um einen von der Firma Array Electronic entwickelten und auf Opencores.org verfügbaren freien DDR-SDRAM-Controller[8]. Dieser stellt ein Derivat eines kommerziellen, leistungsfähigen RAM-Controllers dar, welcher in seiner Leistung stark eingeschränkt ist. So ist z.B. die maximal mögliche Taktfrequenz auf 100 MHz beschränkt und Burst Read bzw. Write Befehle sind nicht möglich. Außerdem ist der CAS Latency Parameter fest auf den Wert 2 gesetzt und kann nicht geändert werden.

Im Rahmen der Recherche wurden noch andere freie und kommerzielle Memory Controller Cores betrachtet. Jedoch erschien der Array Electronic Core am vielversprechendsten, da dieser einfach in der Ansteuerung war und bei Bedarf durch den leistungsfähigeren, kommerziellen Core ersetzt werden kann.

Für den Einsatz in diesem Projekt musste dieser modifiziert und erweitert werden, da die ursprüngliche Version nicht den Anforderungen genügte. Diese Änderungen bezogen sich hauptsächlich auf die Ein- und Ausgangsverschaltung der einzelnen Datenleitungen des DDR-SDRAM-Moduls. Hier mussten spezielle Double Date Rate D-FlipFlops zwischengeschaltet werden, da die Signalwege bis zur endgültigen Verarbeitung sonst zu lang gewesen wären. Des Weiteren wurden die Ausgangstaktsignale von einem auf insgesamt drei erweitert, da das verwendete DIMM-Modul drei Eingänge für differenzielle Taktleitungen besitzt.

Die eingebundene Komponente stellt neben den nach außen zu führenden DDR-SDRAM-Signalen noch eine Benutzerschnittstelle zur Steuerung des Controllers, einen Taktausgang, welcher synchron zu dem DDR-SDRAM-Takt läuft, einen Resetausgang und das nach außen geführte Lock-Signal des Clock Managers bereit.

Eine besondere Rolle spielen die beiden Double Data Rate D-FlipFlops, welche an jede Datenleitung angeschlossen werden. Diese sind essentiell für den korrekten Empfang und die Ausgabe der Daten zu dem externen RAM-Modul. Ohne diese käme es zu Verfälschungen der Daten, da die Laufzeit-unterschiede der einzelnen Signale eine leichte Asynchronität ergeben würden (Wichtig hierbei ist, dass diese FlipFlops direkt in dem jeweiligen I/O-Block der Leitungen implementiert werden).

Wie in Abbildung 5 zu sehen ist, sind die beiden Flip Flops direkt neben dem I/O-Pad lokalisiert. Hierdurch wird das ohnehin zeitkritische, zweiflankengesteuerte Eingangssignal auf zwei Leitungen aufgeteilt, die wiederum nur noch die Hälfte des ursprünglichen Datentaktes besitzen und sich nur noch auf einer Taktflanke ändern. Dadurch wird die weitere Verarbeitung wesentlich vereinfacht.

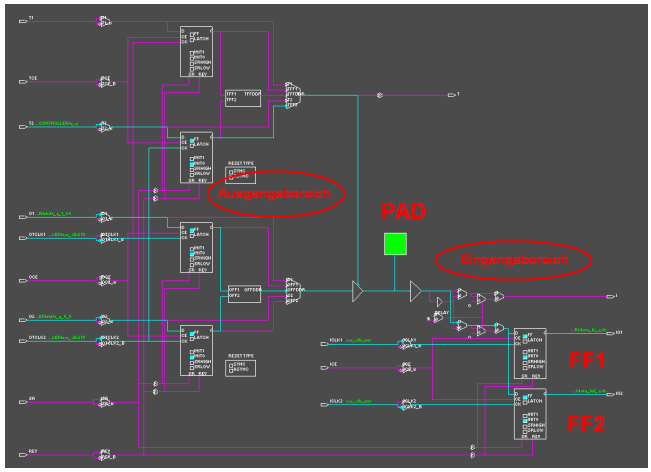


Abbildung 5: Der I/O Block

4.2.4 Speicherzugriffssteuerung

Die Aufgabe der Speicherzugriffssteuerung besteht darin, über einen Zustandsautomat den DDR-SDRAM Controller zu steuern und somit einzelne Frames auszulesen bzw. zu speichern. Die Speicherung erfolgt hierbei nur bei Bedarf bzw. in bestimmten zeitlichen Abständen, während das Auslesen permanent stattfindet, um einen kontinuierlichen Videostrom zu gewährleisten. Das Videosignal wird hierbei exakt synchron zu dem eingehenden Videostrom ausgegeben, um den nachgeschalteten Differenzbildsensor immer mit den passenden Daten zu versorgen. Die Steuerung des DDR-SDRAM Controllers erfolgt über die Signale einer zugehörigen Benutzerschnittstelle.

Neben diesen Leitungen benötigt das Modul noch die VGA Synchronisationssignale, um die Videodaten zu synchronisieren.

Die Struktur des Moduls ist wie in der folgenden Abbildung 6 zu sehen dargestellt. Die Steuersignale werden direkt von dem Zustandsautomat verwaltet, während die ein- und ausgehenden Videodaten wegen der Synchronisierung über jeweils einen Ein- und Ausgangs-FIFO laufen.

Die Synchronisierung mit den FIFO-Speichern ist aus drei verschiedenen Gründen notwendig:

- Erstens erfolgen das Speichern und Auslesen der Daten aus dem DDR-SDRAM mit dem 108 MHz-Systemtakt, während die Übergabe der Videodaten von dem Framegrabber und die Ausgabe an den Differenzbildsensor mit dem Pixeltakt von 27 MHz stattfinden.
- Zweitens werden bei jedem Schreib- und Lesevorgang des DDR-SDRAMs 128Bit Daten übertragen. Für das Videosignal sind jedoch nur 24 Bit pro Pixel notwendig. Somit können bis zu fünf Pixel pro Schreib- oder Lesezyklus übertragen werden. Da die erlaubte Datenbreite der FIFO's jedoch nur eine Potenz von 2 sein darf, wurde diese entsprechend auf 32 Bit gelegt, was dementsprechend nur die gleichzeitige Übertragung von vier Pixeln pro Zyklus ermöglicht.
- Der dritte Grund ist, dass aufgrund der Eigenschaften eines DDR-SDRAMs ein Schreib- oder Lesevorgang unterschiedlich lange dauern kann. Die Aus- und Eingabe der Videodaten erfolgt jedoch kontinuierlich, so dass diese in den FIFO's zwischengespeichert werden müssen.

Aufgrund dieser Forderungen ist der Einsatz eines asymmetrischen FIFO-Speichers notwendig. Dieser besitzt unterschiedlich breite Ein- und Ausgänge und darüber hinaus für die beiden Ports jeweils einen eigenen Datentakt. Die Größe wurde hierbei so gewählt, dass eine komplette Bildschirmzeile zwischengespeichert werden kann.

4.2.5 Videodateneingang

Der Videodateneingangsmodul wandelt die von dem externen VDEC1 gelieferten YCrCb 4:2:2 Videodaten in ein RGB Format um, und es werden die hierzu notwendigen Synchronisationssignale generiert. Diese

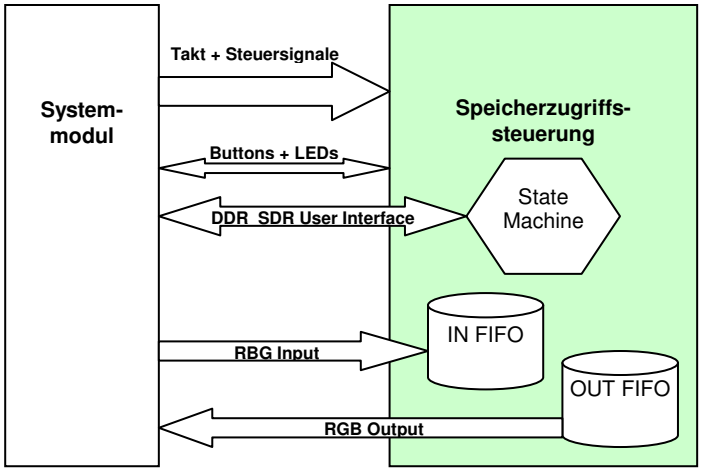


Abbildung 6: Die strukturelle Darstellung der Speicherzugriffssteuerung

Aufgaben übernimmt eine in dem Modul eingefügte Komponente der Firma Digilent[9], das angepasst wurde.

Dieses eingebettete Modul ist für die Wandlung des YCrCb 4:2:2 Signals in ein verarbeitbares RGB-Signal zuständig. Hierfür durchläuft das Signal die in Abbildung 7 dargestellten Stufen.

Direkt nach dem Eingang werden in einem Line Field Decoder zunächst die mit dem YCrCb-Signal übertragenen Statuswerte (horizontale Synchronisation, vertikale Synchronisation und Field-Signale) zur Detektion der einzelnen Halbbilder herausgefiltert. Diese werden zwischen den eigentlichen Videodaten übermittelt.

Danach wird aus dem 4:2:2-Signal ein 4:4:4-Signal generiert, d.h. die bei dem zweiten Luminanzwert Y fehlenden Crominanzanteile Cr und Cb werden durch die bei dem ersten Y-Wert übertragenen Werte ersetzt. Weiterhin wird das Signal jetzt in die drei Bestandteile aufgesplittet. Diese Werte werden in einem nächsten Schritt in einen RGB-Wert gewandelt.

Da jedoch bei diesem Verfahren zwei nacheinanderfolgende Halbbilder übertragen werden, wird immer abwechselnd jede Zeile in einen der zwei LineBuffer geschrieben.

Das Herauslesen der Daten aus dem LineBuffer erfolgt mit dem 27MHz-Pixeltakt.

Neben der Signalwandlung werden in einem Timinggenerator noch die für VGA nötigen Synchronisationssignale generiert. Dieser Generator wird über ein weiteres Modul, welches das Field Signal auswertet, synchronisiert.

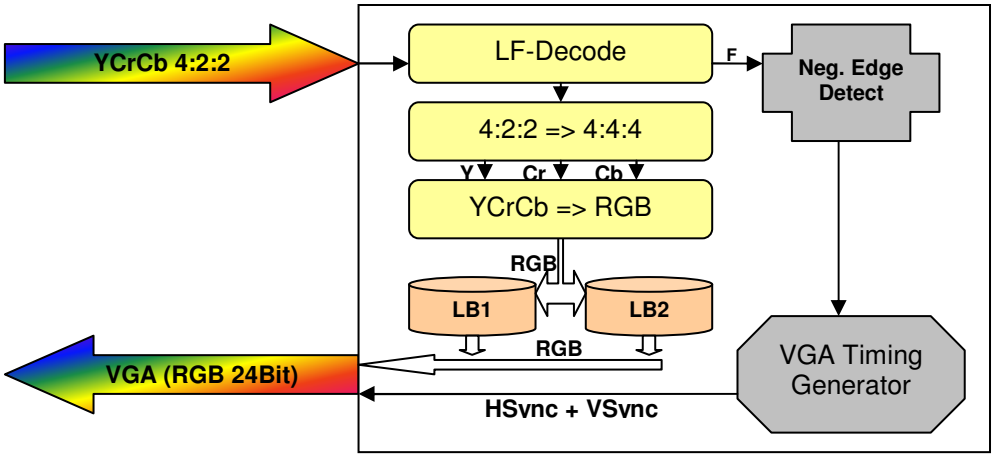


Abbildung 7: Der Formatwandlung von YCrCb 4:2:2 in ein 24Bit RGB Signal

5. Ergebnis

Als Ergebnis dieser Arbeit steht ein voll funktionsfähiger und eigenständiger Differenzbild-sensor zur Verfügung. In Abbildung 8 ist die Funktion des hier erstellten Sensors zu sehen. Hierbei wird eine Hintergrundsubtraktion ausgeführt, wodurch als Ergebnis nur noch die Fahrzeuge zu sehen sind.

Abbildung 8: Eine Anwendung des Differenzbild-sensors für eine Hintergrundsubtraktion

Der modulare Aufbau und der Verzicht auf spezielle, nur in diesem System verfügbare Komponenten ermöglichen ein breites Einsatzfeld. So kann dieses System in nahezu jeden verfügbaren FPGA implementiert und konfiguriert werden. Die einzigen Anforderungen sind hierbei die Bereitstellung eines DDR-SDRAM-Speichers, das Vorhandensein eines VGA-Ausgangs und der Anschluss einer Videoquelle mit YCrCb-Signal bzw. das Vorschalten eines Framegrabber Chips. Die Ressourcenanforderungen sind hierbei:

- ca. 2200 Slice Register (~7% des hier verwendeten FPGAs)
- 13 * 16 KBit Block RAM
- 3 Digital Clock Manager

Im Weiteren können die einzelnen Module auch an andere Gegebenheiten angepasst oder sogar als Teil eines größeren Systems eingebunden werden. So ist beispielsweise der Einsatz in Kombination mit einem Mikroprozessor denkbar, um die Effizienz und Leistungsfähigkeit von Objekterkennungsalgorithmen zu verbessern.

5.1 Ausblick

Der aus diesem Projekt entstandene, FPGA-basierte Differenzbildsensor stellt den ersten Teil einer ganzen Reihe möglicher Bildverarbeitungsoperatoren dar, welche durch die parallele Verarbeitung in einem programmierbaren Logikbaustein enorme Effizienz- und Geschwindigkeitssteigerungen im Vergleich zu einer sequenziellen Abarbeitung in einem herkömmlichen, prozessorbasierenden System verzeichnen können. Jedoch stellen FPGAs alleine auch nicht die beste Lösung dar. Vielmehr liegt die Zukunft in einer Kombination aus einem prozessorgesteuerten System und einem direkt angekoppelten FPGA. Somit könnten rechenintensive Prozesse und Operatoren ausgelagert werden, während der Prozessor die Steuerung und die Abarbeitung des eigentlichen Algorithmus übernimmt.

In Bezug auf dieses Projekt könnte z.B. ein bildverarbeitendes Prozessorsystem direkt von einem vorgeschalteten FPGA in Echtzeit mit dem benötigten Differenzbildergebnis versorgt werden und spart somit eine erhebliche Anzahl an sonst nötigen Rechenschritten.

Aber auch der Sensor selbst kann im Zuge einer Weiterentwicklung verbessert werden. Spielraum liegt vor allem im Bereich der Videoqualität bzw. der verwendeten Auflösung. Das momentan verwendete System, welches für den Anschluss einer NTSC-Videoquelle ausgelegt ist, stellt genau an dieser Stelle einen Engpass dar. Der universellen Einsetzbarkeit und der großen Anzahl möglicher Videoquellen steht eine aktuell nicht mehr zeitgemäße Videoqualität gegenüber. Auch die zwingend notwendige Verwendung externer Framegrabber Chips und die mehrmalige Wandlung in unterschiedliche Signalformate (unter anderem analog/digital) lassen Raum für Verbesserungen.

Denkbar wäre beispielsweise die Verwendung moderner digitaler Kamerasysteme mit CameraLink oder LVDS Anschluss. Diese Technik erlaubt die direkte Anbindung an den FPGA und erspart die verlustbehafteten Komprimierungen der momentan üblichen Systeme. Darüber hinaus lässt sich durch die direkte Anschlussmöglichkeit nicht nur ein externer Framegrabber einsparen, sondern auch interne Logikressourcen, da die rechenintensiven Wandlungen des YCrCb-Formates entfallen.

Dieses Projekt hat gezeigt, dass ein einfacher, aber sequenziell rechenintensiver Bearbeitungsschritt optimiert werden kann. Die Verwendung von FPGAs in Zusammenhang mit bildverarbeitenden Systemen erlaubt den Einsatz kleiner und weniger leistungsstarker Prozessoren. Dies ermöglicht wiederum die Entwicklung modularer intelligenter Kamerasysteme, welche sich beispielsweise für den Einsatz in einem Kreuzungsassistenzsystem eignen.

6. Quellen

[1] Colin Barnden: Driver Assistance Systems Pose FPGA Opportunities, Xcell Journal Issue 66 Fourth Quarter 2008

[2] K. Enke: Possibilities for Improving Safety within the Driver Vehicle Environment Loop, 7th Intl. Technical Conference on Experimental Safety Vehicle, Paris, 1979

[3] U. Seifert: Fahrzeugsicherheit: Personenwagen, VDI Verlag, 1992

[4] XUP V2Pro User Guide, April 2008: http://www.xilinx.com/univ/XUPV2P/Documentation/ug069.pdf

[5] VDEC1 Reference Manual, Revision 4/12/05: http://www.digilentinc.com/Data/Products/VDEC1/VDEC1-rm.pdf

[6] ITU-R BT.656 International Telecommunications Union, Dezember 2007

[7] I²C.Core:
Stand 20. Februar 2004
http://www.opencores.org/projects.cgi/web/i2c/overview

[8] DDR-SDRAM Controller Core:
Version 1.1 vom 19.März 2003
http://www.opencores.org/projects.cgi/web/ddr_sdr/overview

[9] Video Capture Core:
Stand 26.Juli 2005
http://www.digilentinc.com/Data/Products/XUPV2P/video_capture_rev_1_1.zip

Digitales Oszilloskop mit VGA-Anzeige und PS2-Maus Bedienung auf FPGA

Dennis Schlachter

Hochschule Ravensburg-Weingarten, Postfach 1261, 88241 Weingarten

Bei einem digitalen Oszilloskop ist es erforderlich, die eintreffenden Daten (die Abtastwerte des anliegenden Signals) schnell und in Echtzeit zu verarbeiten. Deshalb kommt ein reiner Mikrocontroller basierter Entwurf nicht in Frage. Für die Realisierung wird aus diesem Grunde ein FPGA eingesetzt. Dieser Chip steuert alle zeitkritischen Aufgaben, wie das schnelle Einlesen der Abtastwerte und die zeitlich richtige Darstellung auf einem Anzeigegerät. Für zeitlich unkritische Steuer- oder Berechnungsaufgaben kann ein Mikrocontroller eingesetzt werden.

1. Konzept

1.1.

Über einen schnellen AD Wandler wird ein anliegendes Signal abgetastet. Die Abtastwerte des Signals werden zum FPGA weitergeleitet und dort in einem internen Speicher abgelegt. Die Anzeige der Abtastwerte erfolgt über einen VGA Monitor. Bedient wird das Oszilloskop mit einer PS2 Maus durch anklicken entsprechender Steuerbuttons auf dem Monitor. Alle Steuer-, Berechnungs- und Anzeigefunktionen werden von einem FPGA übernommen. Ein zusätzlicher Mikrocontroller wird bei dieser Realisierung nicht benötigt.

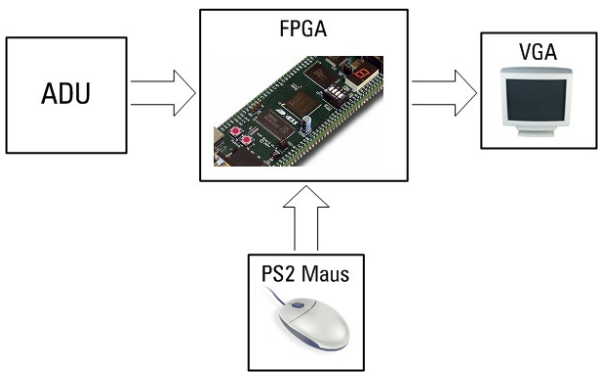


Abbildung 1: Blockdiagramm

Auf Abbildung 2 ist der Oszilloskop-Bildschirm zu sehen. Über den Pfeil an der linken Seite der Gitternetzlinien kann der Trigger-Level eingestellt werden. Der aktuell eingestellt Trigger-Level wird links unten angezeigt. Die Zeitbasis befindet sich rechts unten auf dem Bildschirm und kann über die Pfeilbuttons verändert werden. Der Verlauf eines angelegten Signals wird innerhalb der Gitternetzlinien angezeigt.

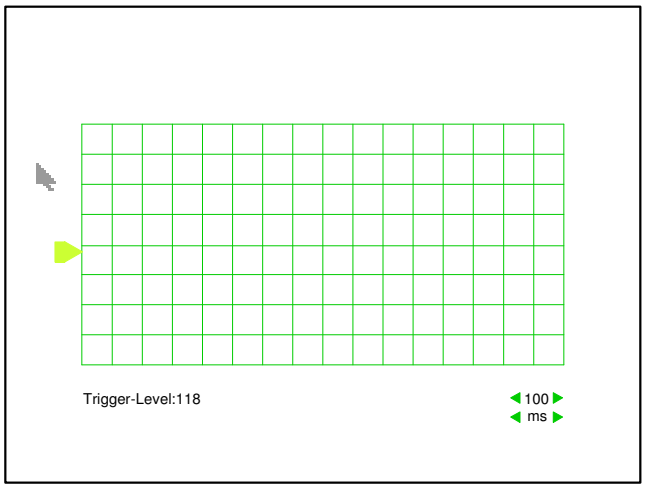


Abbildung 2: Oszilloskop-Bildschirm

2. Beschreibung der FPGA Komponenten

2.1. vga_sync

Die Komponente vga_sync in Abbildung 3 generiert die Video Synchronisation Signale für den VGA Monitor. Die Signale hsync und vsync werden mit dem Monitor verbunden und steuern die horizontale bzw. vertikale Abtastung des Monitors. Die Signale pixel_x und pixel_y geben die aktuelle Pixel Position an. Außerdem wird das Signal video_on generiert, welches bestimmt wann der Monitor dunkel getastet werden sollte, weil der Elektronenstrahl sich in der

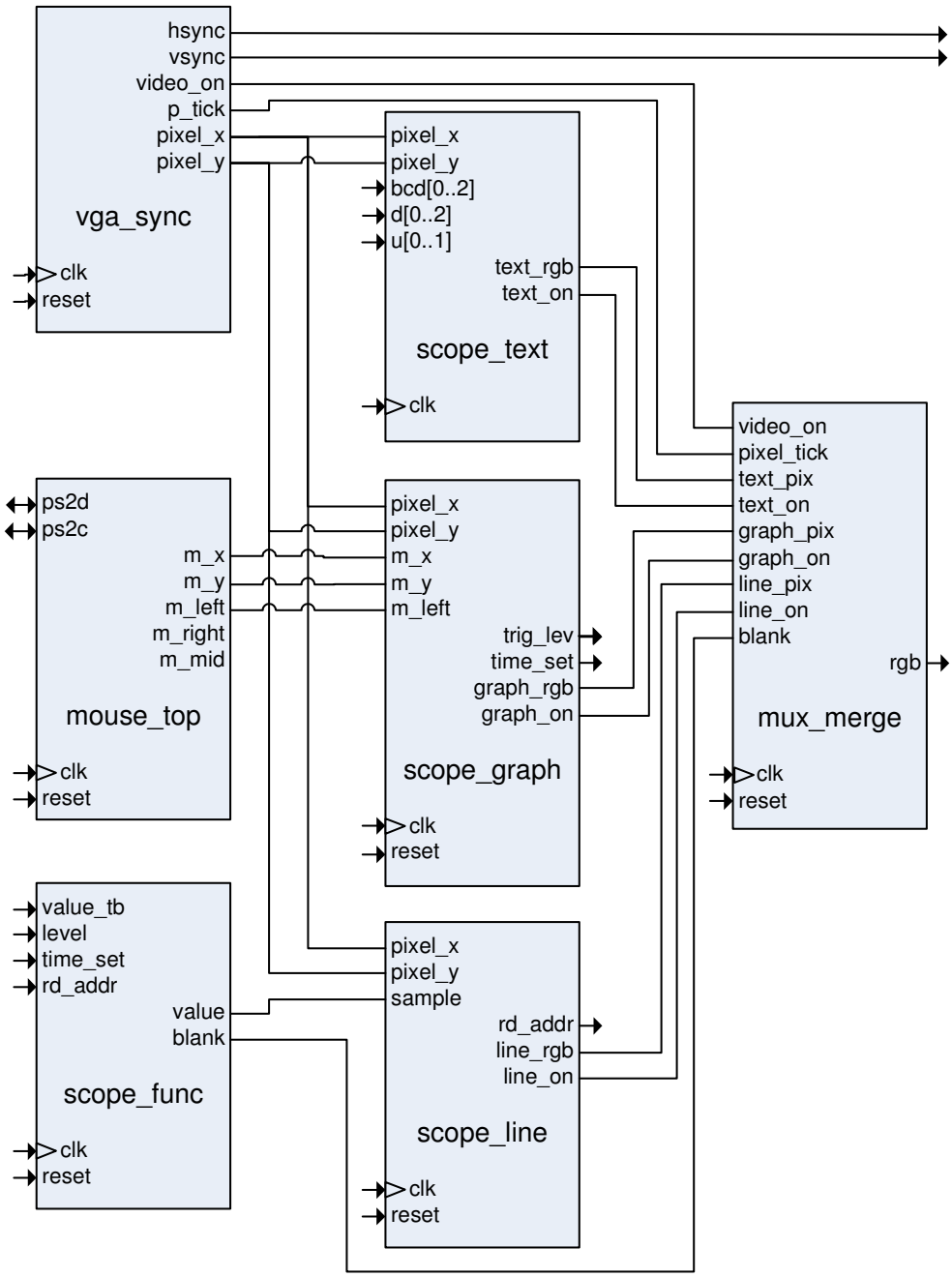


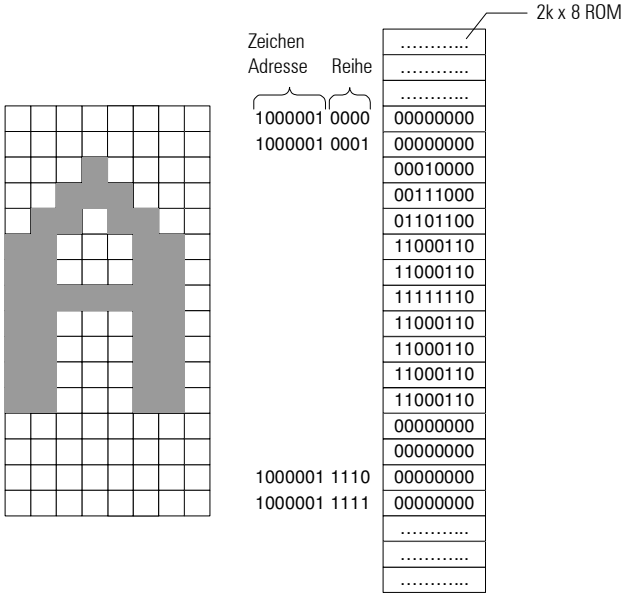
Abbildung 3: Vereinfachtes Blockschaltbild der FPGA Komponenten

vertikalen oder horizontalen Rückführung befindet oder weil das Bild außerhalb des sichtbaren Bereiches ist. Das Signal p_tick wird für Synchronisationszwecke verwendet und gibt den Pixeltakt an, der bei einer VGA Auflösung von 640 x 480 Pixel 25 MHz beträgt.

2.2. scope_text

Die Komponente scope_text in Abbildung 3 realisiert das Textsystem. Sie beinhaltet ein Font Rom, in dem die Bitmuster der ersten 128 ASCII Zeichen gespeichert sind. Jedes Zeichen besteht aus 8 x 16 Pixel (bzw. 8 Spalten und 16 Reihen). Wie diese Darstellung z.B. für den Buchstaben A realisiert wird ist in Abbildung 4 dargestellt. Bei einer Auflösung von 640 x 480 können somit maximal 80 x 30 Zeichen angezeigt werden. Das Rom ist als 2k x 8 Block Ram realisiert. Über den Eingang bcd wird der aktuelle Wert des Trigger-Levels eingelesen. Diesem Eingang ist ein binär zu bcd Konverter vorgeschaltet, der im Blockschaltbild aus Übersichtlichkeitsgründen nicht eingezeichnet ist. Der Trigger-Level, der als Binärwert

vorliegt, wird hiermit in die entsprechende bcd Zahl gewandelt. Über die Eingänge d[0..2] und u[0..1] wird der Anzeigewert der Zeitbasis eingelesen. Diese Werte, z.B. 100 an d[0..2] und ms an u[0..1] (→100ms), werden von der Komponente scope_func im für die Anzeige passenden ASCII Code geliefert. Durch die Signale pixel_x und pixel_y wird abgefragt, an welcher Stelle sich der Elektronenstrahl befindet. Mit Hilfe von Komparatoren kann somit festgelegt werden, an welcher Stelle Text angezeigt werden soll. Die entsprechende ASCII Adresse für das Font Rom wird ebenfalls aus pixel_x und pixel_y abgeleitet. Der Ausgang text_rgb liefert das 3 Bit RGB Signal und text_on wird zu '1', wenn Text angezeigt werden soll.



(a) Pixelmuster (b) ROM Inhalt

Abbildung 4: Textdarstellung

2.3. scope_graph

Die Komponente scope_graph stellt die graphischen Elemente des Oszilloskop-Bildschirms dar. Dies sind die Gitternetzlinien, der Mauszeiger und der Pfeil für den Trigger-Level. Der Mauszeiger erhält seine x- und y-Koordinaten über die Eingänge m_x und m_y. Die Signal m_x und m_y verändern sich entsprechend mit der Bewegung der extern angeschlossenen PS2 Maus und werden von der Komponente mouse_top generiert. Es ist somit möglich, den Maus Zeiger über den gesamten Bildschirm zu bewegen. Der Pfeil für den Trigger-Level befindet sich am linken Rand der Gitternetzlinien. Durch Anwählen mit dem Mauszeiger und Betätigen der linken Maustaste kann der Pfeil im Bereich der obersten und untersten Gitternetzlinie auf und ab bewegt werden. Über die Position des Pfeils wird der aktuelle Trigger-Level abgeleitet und am Ausgang trig_lev ausgegeben. Am Ausgang time_set wird angezeigt, ob einer der vier Buttons der Zeitbasis am rechten unteren Bildschirmrand mit dem Mauszeiger betätigt wurde. Hiermit lässt sich die Zeitbasis einstellen. Die festen Grenzen der Gitternetzlinien sind als Konstanten definiert. Ähnlich wie schon bei der Komponente scope_text wird mit Hilfe von Komparatoren, den Signalen pixel_x, pixel_y und den variablen oder konstanten x- und y-Signalen der graphischen Elemente ermittelt, an welcher Stelle die Elemente angezeigt werden sollen. Der Ausgang graph_on wird zu '1' wenn ein graphisches Element angezeigt werden soll und das drei Bit RGB Signal graph_rgb spezifiziert dessen Farbe.

2.4. scope_line

Die Komponente scope_line ist für die Anzeige der Oszilloskop-Linie bzw. des angelegten Spannungssignals verantwortlich. Hierfür wird zum jeweiligen Zeitpunkt die entsprechende Adresse erzeugt, die über den Ausgang rd_addr am Block Ram (in dem die Abtastwerte abgelegt sind) angelegt wird. Der Abtastwert erscheint dann am Eingang sample. Das Signal pixel_y wird nun mit dem anliegenden Abtastwert verglichen. Aufgrund dieses Vergleichs wird entschieden ob die aktuelle Stelle des Bildschirms Bestandteil der Oszilloskop-Linie ist. Ist dies der Fall wechselt das Signal line_on auf '1' und es wird ein Pixel auf den Bildschirm geschrieben. Über den Ausgang line_rgb kann die Farbe der Linie beeinflusst werden.

2.5. mux_merge

Die Komponenten mux_merge entscheidet, gesteuert über die Eingänge text_on, graph_on und line_on, ob Text, ein graphisches Element oder die Oszilloskop Linie angezeigt werden soll. Die jeweiligen RGB Signale erhält mux_merge von den Eingängen line_pix, text_pix und graph_pix. Der Eingang video_on wird von der schon beschriebenen Komponente vga_sync beliefert und sorgt für die erwähnte Dunkeltastung. Über den Eingang blank kann der Monitor ebenfalls dunkel getastet werden für den Fall das gerade ein Abtastwert ins Ram geschrieben wird. Da das Evaluation Board ein 9 Bit RGB Signal fordert wird das 3 Bit RGB Eingangssignal auf 9 Bit erweitert und am Ausgang rgb ausgegeben. Ferner ist noch ein Ausgangsbuffer für das Ausgangssignal rgb integriert, um eventuelle Störimpulse zu unterdrücken. Es ist zu berücksichtigen, dass sich das Ausgangssignal deshalb um eine Taktperiode verzögert.

2.6. mouse_top

Die Komponente mouse_top realisiert das PS2 Maus Interface. Für den Datenaustausch zwischen Maus und FPGA sind die 2 bidirektionale Leitungen ps2d und ps2c vorhanden. Über ps2d werden die Datenbits ausgetauscht, an ps2c liegt das Taktsignal an. In der Initialisierung Phase wird die Maus durch senden von F4'hex in den Sendemodus (stream mode) versetzt. Ab diesem Zeitpunkt sendet die Maus permanent ihre Daten (relative Bewegung und Status der Tasten). Die Komponente vga_sync liest nun das 3 Byte große Datenpacket der Maus ein und extrahiert hieraus die x- und y-Bewegung und die Tastenbetätigungen. Aus der relativen x- und y-Bewegung wird die absolute Position des Mauszeigers auf dem Bildschirm berechnet. Diese Werte werden an den Ausgängen m_x und m_y zur weiteren Verwendung bereitgestellt. Tastenbetätigungen werden an den Ausgängen m_left, m_mid und m_right erkannt.

2.7. scope_func

Die Komponente scope_func ist für die eigentliche Oszilloskop-Funktion zuständig. Am Eingang value_tb liegt der aktuelle Abtastwert, am Eingang level der Trigger-Level an. Diese beiden Werte werden nun permanent verglichen. Ist die Trigger Bedingung erfüllt (z.B. value_tb > level), löst der Trigger aus und es werden die Abtastwerte in den Speicher geschrieben. Der Speicher ist als 512 x 8 Block Ram realisiert. Da die Komponente scope_line über den Eingang rd_addr ebenfalls auf diesen Speicher zugreift, wurde ein Adressmultiplexer integriert, der zwischen Schreib- und Leseadresse auswählt. Es muss berücksichtigt werden, dass während eines Schreibvorgangs in den Speicher die Komponente scope_line einen falschen Abtastwert erhält. Deshalb wird während eines Schreibvorgangs das Ausgangsignal „blank" gesetzt und somit der Monitor für diesen Zeitraum dunkel getastet. Dem Betrachter fällt dies nicht auf, da ein Schreibvorgang im unteren Nanosekunden Bereich liegt. Über den Eingang time_set wird die Zeitbasis eingestellt.

3. Ergebnis

In den folgenden Abbildungen wird das Ergebnis der Bachelor Arbeit dargestellt. In Abbildung 5 ist ein Sinus Signal zu sehen, die Abbildungen 6 und 7 zeigen jeweils Details zur Einstellung des Trigger-Levels bzw. der Zeitbasis.

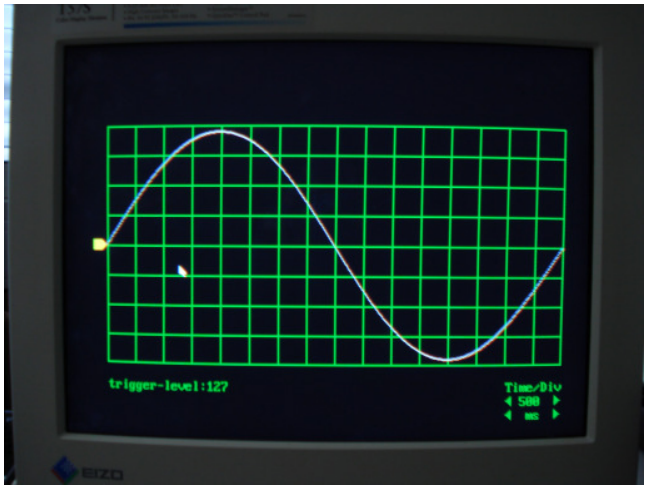


Abbildung 5: Sinus Signal mit einer Frequenz von 2 Hz

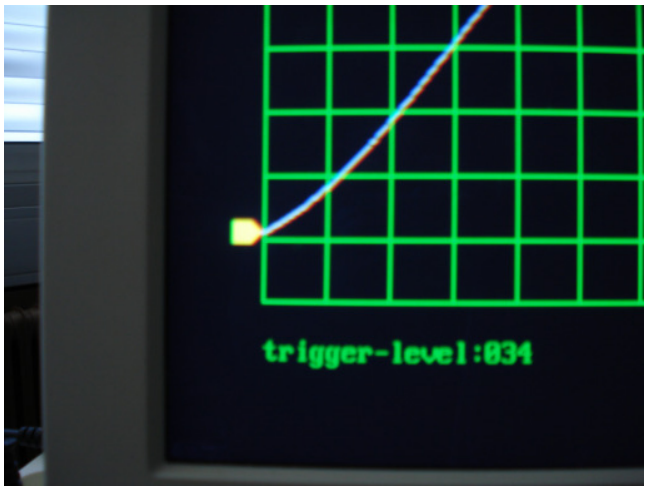


Abbildung 6: Einstellung Trigger-Level

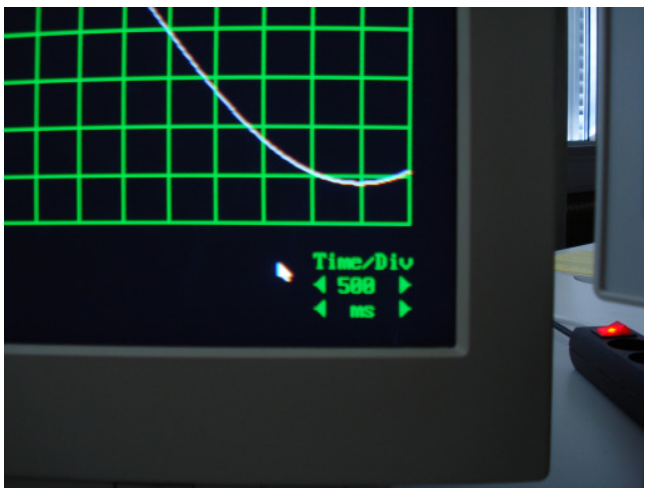


Abbildung 7: Einstellung Zeitbasis

4. Literatur

[1] The Designer's Guide to VHDL (Morgan Kaufmann Series in Systems on Silicon) - 3rd revised edition, Peter J. Ashenden

[2] FPGA Prototyping by VHDL Examples: Xilinx Spartan-3 Version, John Wiley & Sons, Pong P. Chu

[3] Entwurf von digitalen Schaltungen und Systemen mit HDLs und FPGAs, Oldenbourg, Frank Kesel und Ruben Bartholomä

[4] PS/2 Mouse/Keyboard Protocol, www.computer-engineering.org

Grafische Oberfläche für SIRIUS Prozessorkern auf FPGA

Andreas Kreker, Marc Durrenbeger,
Daniel Bau, Florian Zowislok, Dirk Jansen
Hochschule Offenburg, Badstraße 24
0781/ 205 179, akreker@stud.fh-offenburg.de

Zusammenfassung: Auf dem Markt existiert eine Vielzahl an PDAs. Alle haben einen sehr hohen Funktionsumfang und übertreffen sich von Generation zu Generation und erfordern einen hohen Entwicklungsaufwand von ganzen Entwicklerteams.
Der in dieser Arbeit entwickelte PDA mit seiner Hard- und Software soll kein Konkurrenzprodukt darstellen, sondern aufzeigen, was mit haus-internen Mitteln der Hochschule Offenburg möglich ist und gegebenenfalls eine Benutzer-oberfläche für bestehende oder noch kommende Projekte bilden.
Das hier entstandene Gerät ist im Akkumulator-Betrieb autonom und kann als eigenständiges System betrieben werden. Als Herzstück dient das Softcore SIRIUS Mikroprozessorsystem, das als VHDL-Modell in einem FPGA emuliert wird.
Zum Darstellen des grafischen Betriebsystems, welches speziell für dieses PDA entwickelt wurde, wird ein AMOLED-Display verwendet. Dieses besitzt ein Touchpanel, welches zur Steuerung des Systems genutzt wird. Softwareseitig sind Grundfunktionen zur Darstellung von Bildern und Texten entstanden, sowie Beispielanwendungen, die diese benutzen. Das grafische Betriebssystem ist modular und ermöglicht die direkte Weiterentwicklung von Anwendungen für das System.

1. Hardware: PDA-Platine

Die 6-Lagen Platine des PDAs wurde mit dem ALTIUM-Designer entwickelt. Sie wird einseitig bestückt und die Außenmaße betragen BxH: 52 x 54 mm. Die vollständig bestückte Platine ist in Bild 1 dargestellt. Das Herzstück bildet ein FPGA von Altera. Die Konfigurationsdaten des FPGA sind in einem Configuration Device gespeichert, welcher unterhalb des µSD-Karten-Slots zu sehen ist. Rechts daneben ist der 16 MBit-SRAM (Static RAM) von Cypress. Direkt darüber befinden sich im gelben Kasten die RTC mit ihrem 32 kHz – Quarz und im hell-grünen Kasten der Powertaster und ein RS-Flip-Flop. Im blauen Kasten ist der FPC – Connector (Flexible Printed Circuit) des Displays und der Touchcontroller zu sehen.

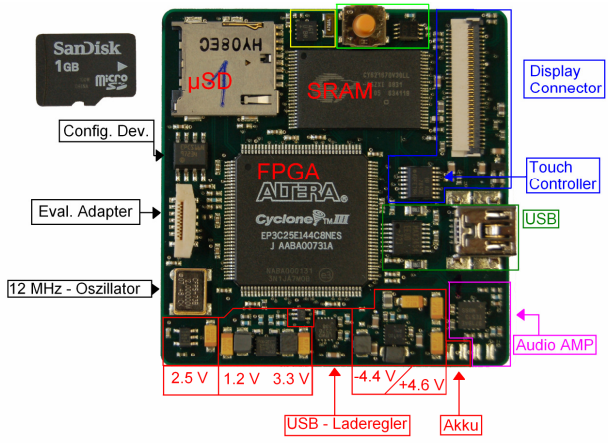


Bild 1: PDA-Platine

Der dunkel-grüne Kasten zeigt die USB-Buchse und den USB-Transceiver. Der pinke Rahmen beinhaltet den Audio-Verstärker und die Kontakte zum Lautsprecher. Neben diesen befinden sich die Kontakte, an denen der Akku angeschlossen wird. Alles was in Bild 1 rot umrandet ist, gehört zum Versorgungsteil. Oberhalb des 2,5 V – Linearreglers ist der 12 MHz Oszillator. Und darüber ist die Verbindungs-Buchse zum Evaluierungs-Adapter zu finden.

2. Konfiguration des FPGAs

2.1. Auslastung des FPGAs

Der verwendete FPGA (EP3C25) [9] gehört zur Familie der Cyclone III FPGAs von Altera. Er hat eine 144-Pin EQFP- Gehäuseform (Enhanced Quad Flat Pack). 83 der 144 Pins sind für den Benutzer frei verfügbar. Diese sind im PDA-Design, bis auf zehn Clock – Eingänge, alle verwendet worden. In Tab.1 sind weitere Daten zu den verwendeten Ressourcen des FPGAs.

Tab.1: Auslastung des FPGAs

	Verwendet	Verfügbar	Auslastung [%]
Logische Zellen	9.191	24.624	37
Pins	73	83	88
Speicher-Bits	525.312	608.256	86
PLL	1	4	25

2.2. SIRIUS Softcore

Das Herz des gesamten Systems ist der SIRIUS Softcore Prozessor. Er ist in VHDL (Very high speed integrated circuit Hardware Desciption Language) geschrieben und wird im FPGA emuliert. Neben dem SIRIUS existieren noch andere emulierte Controller wie der SPI- und USB- Controller. Fast die gesamten Speicherzellen des FPGAs werden für den internen 64 kB RAM des SIRIUS verwendet. In Bild 2 ist die gesamte Struktur des FPGA und seiner angeschlossenen Geräte dargestellt.

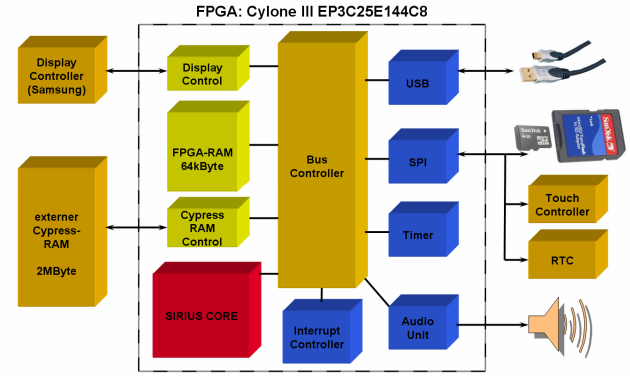


Bild 2: FPGA-Blockschaltbild

Neuerungen sind unter anderem die neue Audio-Unit, die es nun ermöglicht, Wave-Dateien auf dem Lautsprecher auszugeben. Sie entstand im Zuge einer Studienarbeit am IAF und ist mittels eines Delta-Sigma-Wandlers realisiert. Trotz dieses neuen Audio Interfaces ist das alte AHI (Accustic Human Interface) immer noch vorhanden und kann zur Ausgabe von Systemtönen benutzt werden. Da der gleiche Lautsprecher verwendet wird, kann jeweils nur eines der beiden Systeme genutzt werden.

Weitere Neuerungen sind die Komponenten externer SRAM und das Display, welches parallel über den Parallelbus-Controller angebunden ist. Am SPI-Bus befinden sich eine RTC (Real Time Clock), der Touch-Controller sowie die µSD-Karte.

Die Ergebnisse der Synthese des VHDL-Codes sind in Tab. 2 dargestellt.

Tab.2: Syntheseergebnisse

Komponenten	Logische Zellen	Memory Bits
SIRIUS Core	3547	-
USB	3843	1024
Interruptcontroller	415	-
Audio-Unit	356	-
FPGA-RAM	188	524288
SPI	158	-
Buscontroller	144	-
Timer	102	-
Restliche Komponenten	302	-

2.3. Speicheraufteilung des Systems

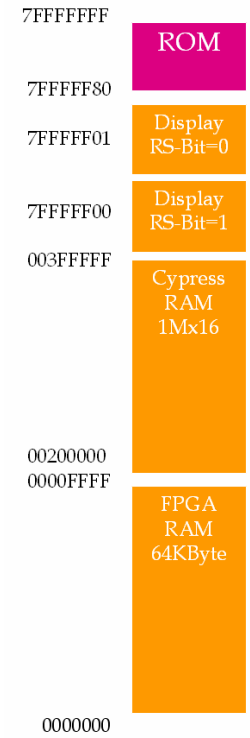


Bild 3: Speicheraufteilung

In Bild 3 ist die Speicher-Aufteilung des Systems dargestellt.

2.4. Externe SRAM

Für die externe RAM (CY62167DV30) sind zusätzlich 20-Adress-Bits nötig, um diesen adressieren zu können. Dies folgt aus der Größe des RAMs: 2^20 * 16 Bit = 16777216 Bit, also 16777216/(1024*1024) = 16 MBit. 16 MBit/8 = 2 MByte. Der SIRIUS ist prinzipiell ein 32-Bit Prozessor, dennoch wurde das alte Betriebssystem mit einem 16-Bit Compiler kompiliert. Um auch den oberen Speicherbereich adressieren zu können, muss dieser auf 32-Bit umgestellt werden. Das zweite Problem liegt darin, dass die externe RAM eine Latenzzeit von 45 ns besitzt und da der SIRIUS mit einem Takt von 48 MHz läuft, also für einen Takt 20,84 ns benötigt, ist er zu schnell für die RAM. Um nicht den gesamten Systemtakt zu vermindern, wurde ein Ready-Signal im SIRIUS Prozessor eingeführt. Dieses wird von einer Ansteuereinheit gesetzt, sobald die RAM-Operationen durchgeführt werden. Zum Beispiel lässt sie den SIRIUS bei einem externen RAM-Zugriff auf die Daten

des RAMs warten. In Bild 4 ist die Simulation des Ready-Signals zu sehen.

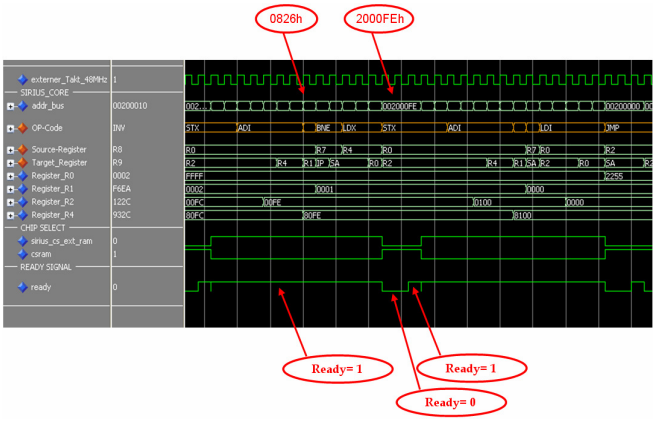


Bild 4: Simulation

Wird, wie das Beispiel in Bild 4 zeigt, auf die Adresse 0826h zugegriffen, handelt es sich um eine Adresse die kleiner 10000h ist, liegt sie also im internen RAM. Daraus folgt, dass das Ready-Signal auf logisch “1“ ist und somit der Prozessor ohne Unterbrechung weiterarbeiten kann. Bei Operationen die auf die interne RAM zugreifen, zum Beispiel arithmetische Operationen, läuft der SIRIUS mit seinem vollen Takt. Handelt es sich um eine Adresse die größer als eine 16-Bit Adresse ist, zum Beispiel um 2000FEh, ist das Ready-Bit auf logisch “0“ und der Ablauf wird um drei Takte verzögert. Der Vorteil liegt darin, dass das System nur verzögert wird, wenn auf das externe RAM zugegriffen wird.

2.5. Display

Bei der Ansteuerung des Displays ist eine direkte Adressierung nicht nötig. Hier reicht es, die Start- und Endposition des GRAM (Graphical RAM) per Datenbus dem Display-Controller (S6E63D6) zu übermitteln. Nach jedem geschriebenem Pixel inkrementiert der Display-Controller die RAM-Adresse automatisch. Dies ist möglich, da das GRAM selten wahllos beschrieben wird, sondern eher zeilenweise mit aufeinander folgenden Pixeln, die dann auch aufeinander folgende Adressen besitzen. Es ist am Parallel-Bus angeschlossen.

Das in Kapitel 2.4. beschriebene Problem mit den Antwortzeiten der Komponenten am Parallel-Bus wirkt sich beim Display-Controller genau so aus. Die Register- und RAM- Zugriffszeiten des Displaycontrollers variieren je nach Schreib- oder Lesevorgang zwischen 30 und 500ns. Bei einem Systemtakt von 48 MHz muss der SIRIUS Prozessor, um die Displayansteuerungszeiten einhalten zu können, angehalten werden. Auch hier kommt das READY Signal des SIRIUS zum Einsatz.

3. Software

3.1. Grafisches Betriebssystem

Wird der PDA gestartet, erscheint kurz das SIRIUS-Logo auf dem Display. Danach wird das Hauptmenu wie in Bild 5 angezeigt. Es zeigt maximal 20 Icons an, die bei Berührung eine Anwendung starten. Die Anwendungen können mit der PDA-Version der SIRIUS IDE kompiliert und als HEX-Datei auf die MicroSD-Karte kopiert werden. Das Icon für die Anwendung kann frei gewählt werden, sollte jedoch eine Auflösung von 48x48 Pixel haben und eine Farbtiefe von 32-Bit. Das Hauptprogramm, welches im “OS“ Ordner der SD-Karte liegt, erwartet die Anwendungen sowie die Icons im Wurzelverzeichnis. Um die beste Position für die Anwendung im Menü auszuwählen, ist eine alphabetische Einteilung realisiert, die es verlangt einen Buchstaben von A bis T sowohl für den Namen der Anwendung als auch für den des Icons zu wählen. Diese Aufteilung ist in Bild 6 dargestellt.

Bild 5: PDA-Menü

Aus dem Vergleich von Bild 5 und Bild 6 geht hervor, dass die Sonnenblume das Icon mit dem Namen “N.ICO“ sein muss, was auch bedeutet, dass die Anwendung, die mit dem Berühren dieser Sonnenblume gestartet wird, den Namen “N.HEX“ trägt. Die noch nicht belegten Platzhalter sind mit transparenten Icons versehen. Die HEX-Dateien der nicht benutzten Plätze müssen nicht vorhanden sein. Diese Art des Zugriffes auf die Anwendung ermöglicht eine leichtere Entwicklung von neuen Anwendungen, da das Hauptprogramm nicht erneut kompiliert werden muss.

A	B	C	D
E	F	G	H
I	J	K	L
M	N	O	P
Q	R	S	T

Bild 6: Menu-Aufteilung

3.2. Beispielanwendung

Der Taschenrechner ist eine der Beispielanwendungen, die entstanden sind, um den Umgang mit den Funktionen und Treibern zu demonstrieren. Er ist auf die Grundrechenarten beschränkt und kann lediglich mit ganzen Zahlen rechnen. Der Aufbau des Taschenrechners ist einfach gestaltet. Die Oberfläche ist eine Bitmap-Datei, sie wird mit der bmp() – Funktion auf dem Display dargestellt und ist in Bild 7 zu sehen. Erfolgen Eingaben, so werden diese im Display des Taschenrechners dargestellt. Zum Aktualisieren der Anzeige wird aus Performancegründen nicht die ganze Bitmap erneut gezeichnet, sondern nur die durch die Eingaben veränderte Fläche.

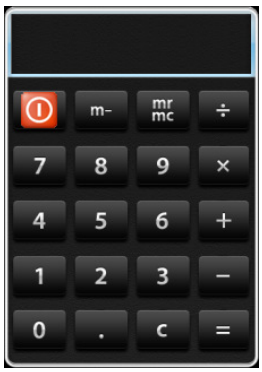


Bild 7: Taschenrechner

Um die eingegebenen Zahlen und die Ergebnisse darzustellen, ist die Funktion writeString() zum Einsatz gekommen, welche Stringketten auf dem Display darstellt und zu der erstellten Schriftarten-Bibliothek gehört.

Die Bedienung des Rechners erfolgt über das Touchpanel. Zur Auswertung der Koordinaten und Rückgabe der gedrückten Taste wird die Funktion get_calc_pos() verwendet. Sie gehört zum erstellten Touch-Treiber.

Durch Eingabe des ersten Operanden wird eine Berechnung eröffnet. Danach muss ein Operator folgen. Nach diesem wird der zweite Operand erwartet. Ist dieser eingegeben muss mit dem “Gleichheitszeichen“ das Ergebnis abgerufen werden. Dieses steht automatisch als neuer erster Operand für eine weitere Berechnung zur Verfügung. Soll eine vollkommen neue Berechnung durchgeführt werden, ist die “C-Taste“ zu betätigen. Die Funktion der Tasten “.“, “m-“ und “mr/mc“ sind nicht implementiert. Beendet wird das Programm mit der roten Ausschalttaste.

3.3. Treiber und Bibliotheken

Für die Kommunikation mit dem Display-Controller werden vier Routinen benötigt. Diese sind in Assembler geschrieben, im BIOS verankert und können aus jeder späteren Anwendung, sofern als externe Routine deklariert, aufgerufen werden.

Routine 1: setIndex(U16)

Mit dieser Routine wird im Display-Controller festgelegt, welches seiner Register momentan angesprochen werden soll. Sie erwartet einen vorzeichenlosen 16-Bit Wert als Übergabe, welcher die anzusprechende Adresse beinhaltet. Diese kann sowohl eines der Register als auch das GRAM ansprechen.

Routine 2: setData(U16)

Diese Routine setzt einen Wert in das momentan gewählte Register oder in den GRAM. Im Falle eines Registers wird hier die Konfiguration des Registers übertragen. Ist die gewählte Adresse das GRAM wird hiermit der 16-Bit Farbwert eines Pixels übertragen.

Routine 3: DisplayReadStatus()

Wählt ein Register aus, das gelesen werden soll.

Routine 4: DisplayReadData()

Hiermit kann die Konfiguration eines Registers des Display-Controllers oder ein Pixelwert wieder ausgelesen werden, abhängig von dem mit DisplayReadStatus() gewähltem Register oder GRAM.

Bevor das Display nach einem Neustart funktionieren kann, muss es initialisiert werden. Dies wird beim Bootvorgang des grafischen Betriebssystems erledigt.

Aufbauend auf den Display-Treiber sind unter anderem Bibliotheken entstanden zum Darstellen von Windows Bitmaps und Icons, verschiedenen Schriftarten und geometrischen Figuren wie Rechtecke und Kreise.

Um das System bedienen zu können ist ein Touch-Treiber mit Auswerte-Logik entstanden, die in Kapitel 3.1. beschrieben wird.

4. Eckdaten

Hardware	
Display	
Größe	2.8"
Technologie	AMOLED
Auflösung HxB	320x240 Pixel
Anzahl Farben	65536
CPU	
Prozessor	SIRIUS
Takt	48 MHz
Speicher	
internes RAM	FPGA 64 kB
externes RAM	Cypress 2 MB
Daten	SanDisk 1 GB micro SD
Steuerung	
Typ	resistives Touchpanel
Akku	
Akkutechnologie	Lithium-Ionen
Akkukapazität	1150 mAh
Dauerbetrieb	5h
Anschlüsse	
USB	Aufladen des Akkus und Verbindung mit USB-Konsole
Größe und Gewicht	
Abmessungen	107 x 60 x 10,5 mm
Gewicht	134 g

Software	
Betriebssystem	grafisches PDA-OS
Unterstützte Dateiformate	
Ausführtbar	HEX (Intel)
Bild	BMP (1,4,8,16,24-Bit), ICO (32-Bit)
Audio	WAV 44,1 kHz 16-Bit
Text	TXT

Verbesserung der Lokalisationsgenauigkeit in selbstorganisierenden Sensornetzwerken durch Ausnutzung des frequenzselektiven Verhaltens der Signalstärke

Markus Rollinger, Dirk Benyoucef

Digital Communications & Signal Processing Lab (DCSP-Lab)

Hochschule Furtwangen, Robert-Gerwig-Platz 1, 78120 Furtwangen

ros@hs-furtwangen.de,benyoucef@hs-furtwangen.de

Kurzfassung

Zur Lokalisation von Sensorknoten werden in drahtlosen Sensornetzwerken häufig Distanzen benötigt. Da die Funkmodule der Sensorknoten bei der Demodulation empfangener Signale bereits eine Signalstärkemessung durchführen und das Ergebnis in Form eines RSSI-Wertes zur Verfügung stellen, ist die Auswertung der Signalstärke zur Distanzbestimmung ohne zusätzliche Hardware möglich. Dieses Verfahren liefert jedoch nur ungenaue Ergebnisse.

Um die Genauigkeit dieses Verfahrens verbessern zu können, wurden im DCSP-Lab der Hochschule Furtwangen Signalstärkemessungen durchgeführt. Die Messung der empfangenen Signalstärke unter Verwendung verschiedener Übertragungskanäle ermöglichte die Untersuchung ihres frequenzselektiven Verhaltens.

Die Auswertung der gewonnenen Messdaten ergab, dass das frequenzselektive Verhalten der empfangenen Signalstärke dazu genutzt werden kann, die Auswirkungen des Fadings auf die Signalstärke deutlich zu reduzieren und dadurch die Genauigkeit des RSSI-Verfahrens zu steigern.

1 Einführung

Der Zusammenschluss hunderter oder tausender Sensorknoten in einem drahtlosen Ad-Hoc Netzwerk wird als Sensornetzwerk bezeichnet. Seine Ausbringung in einem zu untersuchendem Gebiet ermöglicht die Überwachung einer oder mehrerer physikalischer Parameter.

Die Anwendungsmöglichkeiten sind vielfältig. Beispielsweise könnten Sensornetzwerke in Wäldern verteilt und durch Temperaturmessungen dazu verwendet werden, Waldbrände frühzeitig zu erkennen und effektiver zu bekämpfen [Rei07]. Ein weiteres Anwendungsbeispiel ist der Einsatz von Sensornetzwerken in der Laborautomatisierung. Die Überwachung der Vital-Funktionen oder die Detektion von Gefahrenstoffen, die in Schutzanzüge eindringen, könnten dazu beitragen, Labormitarbeiter rechtzeitig aus einem Gefahrenbereich zu evakuieren [RBT04]. Neben den genannten Beispielen ist der Einsatz von Sensornetzwerken in der Gebäudeautomatisierung [MR03], Steigerung der Verkehrssicherheit [TM07], Erforschung der Verhaltensweise bestimmter Tiere [MR03], Ozeanographie [RHT05], etc. denkbar.

Alle genannten Einsatzmöglichkeiten haben eine Sache gemeinsam: Um einem Messwert einen Ort zuordnen zu können, muss den Sensorknoten ihre eigene Position bekannt sein. Andernfalls wäre die Nutzbarkeit der Messwerte deutlich eingeschränkt. Die Lokalisation der Sensorknoten in einem drahtlosen Sensornetzwerk ist daher von entscheidender Bedeutung für dessen Verwendbarkeit.

Mit Hilfe exakter mathematischer Verfahren wie beispielsweise der Trilateration oder Triangulation ist eine exakte Ortsbestimmung von Sensorknoten möglich. Allerdings setzt dies voraus, dass fehlerfreie geometrische Abhängigkeiten vorliegen. Die Genauigkeit einer Lokalisation hängt stark von deren Qualität ab — je genauer Distanzen und Winkel sind, desto präziser kann die Position eines Sensorknotens ermittelt werden.

Es gibt eine Vielzahl von verschiedenen Verfahren um geometrische Abhängigkeiten zwischen benachbarten Sensorknoten zu bestimmen. Eines davon stellt das RSSI-Verfahren dar, welches die Distanz durch Auswertung der empfangenen Signalstärke ermittelt. Der Vorteil hierbei ist, dass keinerlei zusätzliche Hardware benötigt wird, da Funkmodule für die Demodulation die Signalstärkemessungen übernehmen. Die geringe Genauigkeit ist jedoch nachteilig.

Diese Arbeit beschäftigt sich mit der Verbesserung des RSSI-Verfahrens. Sie gliedert sich wie folgt: In Kapitel 2 werden die Grundlagen der Distanzbestimmung erläutert. Kapitel 3 beschäftigt sich mit der Beschreibung ei-

nes Messaufbaus, der dazu diente, die Messwerte der empfangenen Signalstärke zu ermitteln, die dieser Arbeit zu Grunde liegen. In Kapitel 4 werden eine Verbesserung des Verfahrens beschrieben und Simulationsergebnisse dargestellt. Das letzte Kapitel ist eine Zusammenfassung der Ergebnisse.

2 Grundlagen

2.1 Distanzbestimmung aus der empfangenen Signalstärke

Beim Empfang einer Nachricht messen die Funkmodule der Sensorknoten die Signalstärke und stellen einen RSSI-Wert zur Verfügung. Um aus diesem Wert die Distanz zum Sender bestimmen zu können, kann Gleichung (1), welche die empfangene Signalstärke als Funktion der Distanz beschreibt, verwendet werden [PAOH03]. Mit diesem Kanalmodell wird der Einfluss des Large-Scale Fadings auf die empfangene Signalstärke berücksichtigt. Dieses setzt sich aus dem mittleren Pfadverlust und Schwankungen um diesen Mittelwert zusammen [Skl97]. Das Small-Scale Fading bleibt bei diesem Modell jedoch unberücksichtigt.

$$P_r(d)(dB) = P_0(dB) - 10n \log\left(\frac{d}{d_0}\right) + X_\sigma \qquad (1)$$

In Gleichung (1) ist $P_r(d)$ die empfangene Signalstärke in Dezibel, die in einer Entfernung d zum Sender gemessen wird. P_0 entspricht der Referenzsignalstärke in Dezibel im Abstand d_0. In Abhängigkeit der Umgebung werden für d_0 unterschiedliche Werte gewählt. In großen Zellen wird der Wert 1 km, in Mikrozellen 100 m und in Innenraumumgebungen 1 m verwendet [Skl97]. Der Faktor n ist abhängig vom Übertragungskanal. Er wird häufig auch als Pfadverlustexponent bezeichnet. Im Freiraum kann für n der Wert 2 gewählt werden. In Abhängigkeit der Umgebung werden kleinere oder größere Werte verwendet. $X_\sigma \sim \mathcal{N}(0;\sigma^2)$ ist eine gaußsche Zufallsvariable mit einem Mittelwert vom Betrag 0 und der Varianz σ^2. Sie dient zur Berücksichtigung der Abweichung der Signalstärke von ihrem Mittelwert [Skl97].

Durch Umformung des Mittelwerts von (1) erhält man Gleichung (2), mit der die Distanz zweier benachbarter Sensorknoten berechnet werden kann.

$$d(P_r) = 10^{\left(\frac{P_0(dB)-P_r(dB)}{10n}\right)} \qquad (2)$$

Um die Distanz unter Verwendung von Gleichung (2) ermitteln zu können, müssen geeignete Werte für P_0 und n gewählt werden. Da diese beiden Parameter jedoch von der Umgebung abhängig sind, schätzt man sie für jeden Ort.

2.2 Genauigkeit der Distanzbestimmung

Abbildung 1 zeigt die Darstellung von Messdaten, die im DCSP-Lab ermittelt wurden. Der blaue Graph stellt dabei die gemessene Signalstärke dar, während der rote Graph aus Gleichung (1) bestimmt wurde.

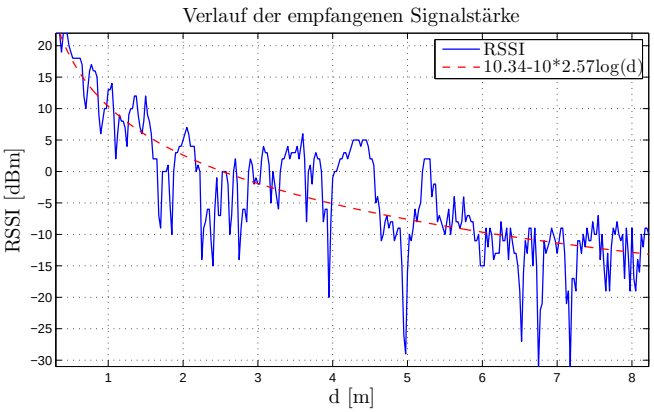


Abbildung 1: Verlauf der Signalstärke, Gleichung 1 entsprechend (rot) und aus Messungen ermittelt (blau)

Der Verlauf der gemessenen Signalstärke ist von Schwankungen und zum Teil tiefen Einbrüchen geprägt. Dieses Verhalten wird durch das Autreten von Fading verursacht [Skl97]. Die Auswirkungen auf die Genauigkeit einer Distanzbestimmung können durch Auswertung des absoluten Distanzfehlers, der in Gleichung (4) definiert ist, beobachtet werden.

$$\Delta d = |d - d(P_r)| \qquad (3)$$

$$= \left| d - 10^{\left(\frac{P_0(dB)-P_r(dB)}{10n}\right)} \right| \qquad (4)$$

Dabei ist Δd der Betrag des Distanzfehlers und P_r die gemessene Signalstärke im Abstand d zum Sender.

Die Anwendung von Gleichung (4) auf die Messdaten aus Abbildung 1 liefert das in Abbildung 2 dargestellte Schaubild. Entlang der Ordinate kann der absolute Distanzfehler abgelesen werden, der bei einer Distanzbestimmung in einer Entfernung d zum Sender auftritt.

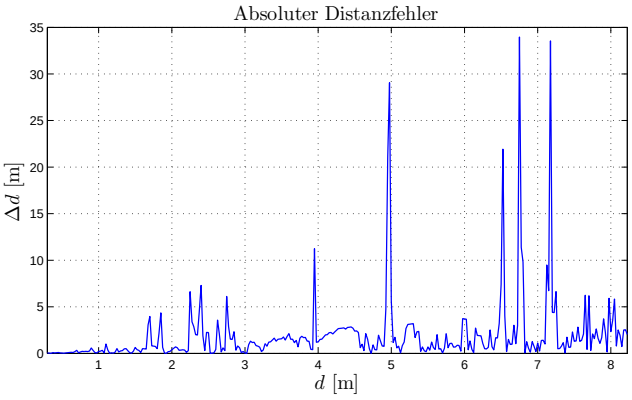


Abbildung 2: Absoluter Distanzfehler

Es wird deutlich, dass die Schwankungen der Signalstärke zu Distanzfehlern führen, die bis zu mehreren Metern betragen können. Besonders große Fehler werden dabei durch tiefe Einbrüche der Signalleistung, wie es beispielsweise an der Stelle $d = 5$ Meter der Fall ist, verursacht. Je größer die Entfernung zum Sender ist, desto stärker wirken sich die Abweichungen der Signalstärke vom roten Graphen in Abbildung 1 auf den Distanzfehler aus.

3 Messaufbau

Die Untersuchung der empfangenen Signalstärke ist Gegenstand unterschiedlicher Publikationen [WLT+08, GF06, SKPP07]. In [GF06, SKPP07] liegt der Fokus auf Messungen der Signalstärke mit verschiedenenen Übertragungsfrequenzen. Dabei hat sich gezeigt, dass die Signalstärke ein frequenzabhängiges Verhalten aufweist. Um das frequenzselektive Verhalten der Signalstärke im Hinblick auf die Verbesserung des RSSI-Verfahrens zu untersuchen, werden weitere Messungen der Signalstärke unter der Verwendung verschiedener Übertragungsfrequenzen benötigt. Hierzu wurde im DCSP-Lab der Hochschule Furtwangen ein Messaufbau erstellt, der in Abbildung 3 schematisch dargestellt ist.

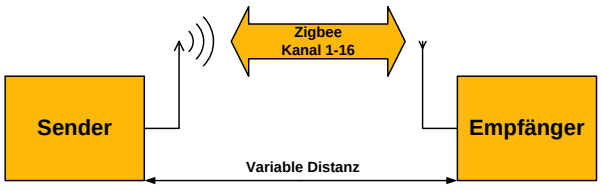


Abbildung 3: Schematische Darstellung des Messaufbaus

Der Messaufbau besteht aus zwei Modulen — einem Sender, und einem Empfänger. Beide Module sind baugleich und bestehen aus je einem Evaluierungsboard mit der Bezeichnung „CC2420DBK“ [Ti:08b]. Diese verfügen über einen Zigbee-Transceiver des Typs CC2420 [Ti:08a] und werden für die Signalstärkemessungen benötigt.

Der in Abbildung 3 dargestellte Sender sendet periodisch und mit konstanter Leistung ein Datenpaket. Der verwendete Zigbee-Transceiver arbeitet mit Übertragungsfrequenzen von 2,405 GHz bis 2,480 GHz im lizenzfreien ISM-Band. Dabei wird nacheinander zwischen allen 16 zur Verfügung stehenden Übertragungskanälen gewechselt. Der Empfänger, der in Abbildung 3 rechts dargestellt ist, misst die Signalstärke aller eingehenden Nachrichten und berechnet ihre RSSI-Werte.

Mit dem Testaufbau wurden verschiedene Messreihen durchgeführt. Der Ablauf war bei allen Messreihen identisch: Die Signalstärke wurde vom Empfänger mit unterschiedlichen Distanzen[1] und Übertragungskanälen gemessen. Es wurden Messreihen sowohl im Innenraum als auch im Freien zu verschiedenen Tageszeiten durchgeführt.

4 Verbesserung der Distanzgenauigkeit

Abbildung 4 zeigt eine dreidimensionale Darstellung von Messdaten, die im Rahmen dieser Arbeit ermittelt worden sind.

Die Distanz steht für den Abstand zwischen Sender und Empfänger. Entlang der k-Achse sind die verwendeten Übertragungskanäle aufgetragen. An der RSSI-Achse kann die Signalstärke, die bei einer Distanz d unter Verwendung der Übertragungskanals k gemessen wurde, abgelesen werden.

Die Nutzung verschiedener Übertragungskanäle führt dazu, dass bei der Messung der Signalstärke bei gleichen Entfernungen unterschiedliche Werte ermittelt werden. Dieses Verhalten äußert sich in Abbildung 4 durch den ungleichmäßigen Verlauf bei konstanten Werten für d.

Das Senden von Nachrichten mit wechselnden Übertragungskanälen entspricht beim CC2420 dem Gebrauch unterschiedlicher Übertragungsfrequenzen. Die Frequenz des ersten Kanals beträgt 2.405 GHz. Alle weiteren 15 Kanäle sind aufsteigend angeordnet und haben zueinander einen Abstand von 5 MHz.

[1] Mit einer Auflösung von 2,5cm

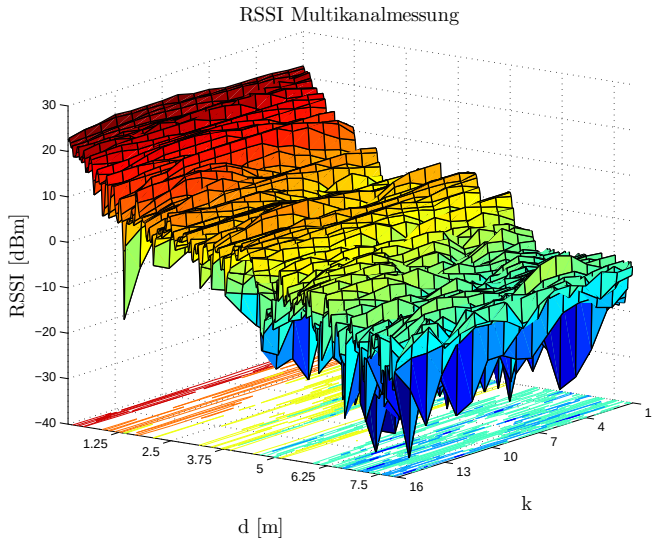


Abbildung 4: Dreidimensionale Darstellung von Messdaten

Das unterschiedliche Verhalten der Signalstärke ist dadurch zu erklären, dass Signale verschiedener Wellenlänge ausgesendet werden. Bei gleichen Umgebungsbedingungen interferieren die Mehrwegekomponenten von Signalen aufgrund anderer Feldstärkeverteilungen unterschiedlich beim Empfänger.

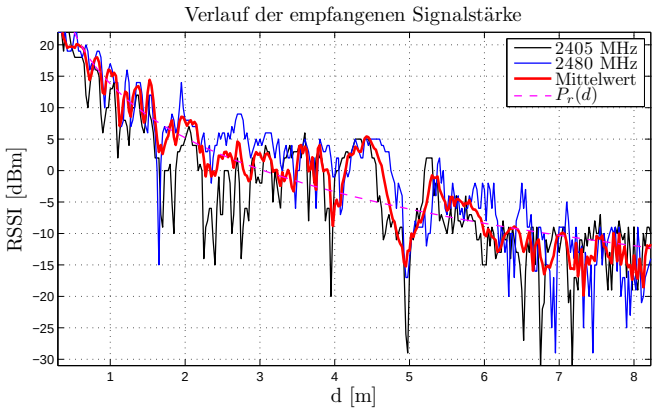


Abbildung 5: Verlauf des RSSI (schwarz, blau) und dessen Mittelwert über alle Kanäle (rot)

Abbildung 5 zeigt den Verlauf des RSSI für die Übertragungsfrequenzen 2,405 GHz und 2,480 GHz. Aufgrund des Wellenlängenunterschieds treten Fading-Löcher des schwarzen und blauen Graphen zueinander verschoben und mit unterschiedlicher Intensität auf. Dieses Verhalten kann man im Bereich von $d = 6$ bis $d = 8$ Meter sehr deutlich beobachten. Stellenweise driften die Fading-Löcher soweit auseinander, dass an manchen Orten nur bei einem Graphen, aber nicht bei beiden, ein Fading-Loch auftritt.

	absoluter Distanzfehler [m]	
	Mittelwert	Maximum
Kanal 1	2,2	54,65
Kanal 16	2,23	48,91
Mittelwert Kanal 1-16	0,87	2,86

Tabelle 1: Mittel- und Maximalwert des absoluten Distanzfehlers

Dieses frequenzselektive Verhalten der Signalstärke kann dazu genutzt werden, den Einfluss von Fading-Löchern auf den Verlauf der Signalstärke zu reduzieren. Die Berechnung des Mittelwertes aller an einem Ort vorliegender Messwerte führt zu dem roten Graphen in Abbildung 5. Es werden dadurch Fading-Löcher herausfiltert und der gesamte Verlauf deutlich geglättet.

Abbildung 6 zeigt die absoluten Distanzfehler der in Abbildung 5 dargestellten Daten.

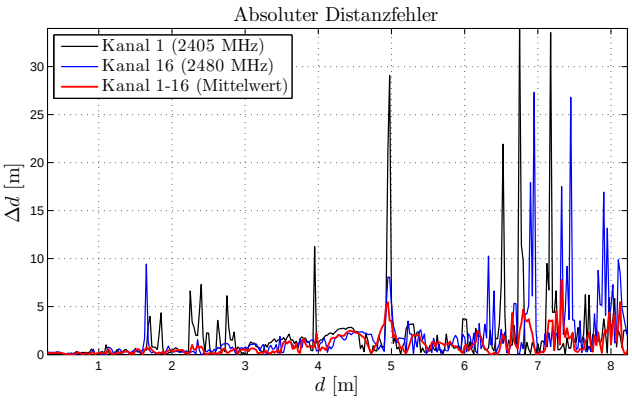


Abbildung 6: Absoluter Distanzfehler des RSSI für einzelne Kanäle (schwarz, blau) und den Mittelwert über alle Kanäle (rot)

Die Mittelwertbildung des RSSI durch Einbeziehung der Daten aller Übertragungskanäle führt zu einer deutlichen Verringerung des absoluten Distanzfehlers.

Aus der Anwendung eines Algorithmus, der den Mittelwert über alle Übertragungskanäle bildet, auf alle Messdaten erhält man die Ergebnisse in Tabelle 1.

Die Mittelwertbildung liefert gegenüber der Verwendung einzelner Übertragungskanäle deutlich geringere Distanzfehler. Dies gilt sowohl für den Mittelwert, als auch für den Maximalwert des absoluten Distanzfehlers.

5 Zusammenfassung

In diesem Artikel wurde gezeigt, wie die Genauigkeit des RSSI-Verfahrens der Signalstärke verbessert werden kann. Das frequenzselektive Verhalten der Signalstärke führt dazu, dass ihre Messung mit verschiedenen Frequenzen unterschiedliche Ergebnisse liefert. Die Mittelwertbildung des RSSI über alle Übertragungskanäle bewirkt eine Glättung seines Verlaufs und eine Reduzierung des Distanzfehlers.

Es muss noch geklärt werden, ob der Distanzfehler durch die Nutzung eines intelligenteren Verfahrens weiter vermindert werden kann.

Literatur

[GF06] GALBREATH, Jake ; FROLIK, Jeff:
Channel allocation strategies for wireless sensors statically deployed in multipath environments.
In: *IPSN '06: Proceedings of the 5th international conference on Information processing in sensor networks*.
New York, NY, USA : ACM, 2006. –
ISBN 1–59593–334–4, S. 334–341

[MR03] MATTERN, Friedemann ; RÖMER, Kay:
Drahtlose Sensornetze.
In: *Informatik-Spektrum* 26 (2003), Juni, Nr. 3, S. 191–194

[PAOH03] PATWARI, Neal ; ALFRED O. HERO, III:
Using proximity and quantized RSS for sensor localization in wireless networks.
In: *WSNA '03: Proceedings of the 2nd ACM international conference on Wireless sensor networks and applications*.
New York, NY, USA : ACM, 2003. –
ISBN 1–58113–764–8, S. 20–29

[RBT04] REICHENBACH, Frank ; BLUMENTHAL, Jan ; TIMMERMANN, Dirk:
Drahtlose Sensornetzwerke in der Laborautomatisierung.
In: *GIT Laboratory IT User Service (LITUS)* (2004), November, S. 20–21. –
ISSN 16148622. –
Darmstadt, Germany

[Rei07] REICHENBACH, Frank:
Ressourcensparende Algorithmen zur exakten Lokalisierung in drahtlosen Sensornetzwerken.
Fakultät für Informatik und Elektrotechnik, Universität Rostock, Dissertation, 2007

[RHT05] REICHENBACH, Frank ; HANDY, Matthias ; TIMMERMANN, Dirk:
Monitoring the Ocean Environment with Large-Area Wireless Sensor Networks.
In: *8th EUROMICRO Conference on Digital System Design*, Institute for Systems Engineering and Automation, September 2005. –
ISBN 3902457090, S. 57–58. –
Porto, Portugal

[Skl97] SKLAR, B.:
Rayleigh Fading Channels in Mobile Digital Communication Systems Part I: Characterization.
In: *Communications Magazine, IEEE* 35 (1997), S. 90–100

[SKPP07] STOYANOVA, Tsenka ; KERASIOTIS, Fotis ; PRAYATI, Aggeliki ; PAPADOPOULOS, George:
Evaluation of impact factors on RSS accuracy for localization and tracking applications.
In: *MobiWac '07: Proceedings of the 5th ACM international workshop on Mobility management and wireless access*.
New York, NY, USA : ACM, 2007. –
ISBN 978–1–59593–809–1, S. 9–16

[Ti:08a] TI:WEBLINK:
CC2420 Zigbee Transceiver Datasheet.
`http://www.ti.com/lit/gpn/cc2420`, 2008. –
Stand: 06.02.2009

[Ti:08b] TI:WEBLINK:
CC2420DBK Demonstration Board Kit.
`http://www.ti.com/lit/pdf/swru043`, 2008. –
Stand: 06.02.2009

[TM07] TORRENT-MORENO, Marc:
Inter-vehicle communications: assessing information dissemination under safety constraints.
In: *Wireless on Demand Network Systems and Services, 2007. WONS '07. Fourth Annual Conference on*, 2007, S. 59–64

[WLT+08] WU, Rong-Hou ; LEE, Yang-Han ; TSENG, Hsien-Wei ; JAN, Yih-Guang ; CHUANG, Ming-Hsueh:
Study of characteristics of RSSI signal.
In: *Industrial Technology, 2008. ICIT 2008. IEEE International Conference on* (2008), April, S. 1–3.
`http://dx.doi.org/10.1109/ICIT.2008.4608603`. –
DOI 10.1109/ICIT.2008.4608603

Hochschule Reutlingen
Reutlingen University

Mechatronik
Hochschule Reutlingen

Entwicklung und Implementierung eines Algorithmus zur Eliminierung des Leckeffektes bei der Diskreten Fourier-Transformation

Timo Zawischka
Hochschule Reutlingen, Studiengang Mechatronik

1. Einführung

Die Fourier-Transformation ist ein mathematisches Instrument zur Berechnung der Spektralanteile eines Signales. Sie ist dem französischen Mathematiker und Physiker Jean Baptiste Joseph Fourier (1768 – 1830) zuzuschreiben, der herausfand, dass sich jedes periodische Signal in eine Summe cosinusförmiger Teilsignale zerlegen lässt. Mit der Digitalisierung kam der Wunsch auf, die Fourier-Transformation auch in Computersystemen und der Digitalen Signalverarbeitung einzusetzen. Daraufhin wurden die Diskrete Fourier-Transformation (DFT) und die Fast Fourier Transformation (FFT) zur Transformation zeitdiskreter, periodischer Signale entwickelt. Beide erfüllen dieselbe Aufgabe, die FFT ist jedoch, auf Grund der Anwendung einiger mathematischer Tricks, wesentlich weniger rechenaufwändig (und somit schneller) als die DFT.

Damit die Transformationsmechanismen korrekt arbeiten, müssen einige Rahmenbedingungen eingehalten werden. Zu den Wichtigsten gehören die Einhaltung des Abtasttheorems und das Transformieren eines ganzzahligen Periodenvielfachen des Signals. Das Abtasttheorem nach Shannon verlangt eine mindestens doppelt so hohe Abtastrate (Abtastfrequenz) als die höchste zu erwartende Frequenz des zu transformierenden Signales.

Wesentlich wichtiger ist jedoch das Transformieren eines ganzzahligen Periodenvielfachen. Diese Bedingung ist in der Realität jedoch praktisch nicht erfüllbar, da im Regelfall die Abtastrate fest und die Signalfrequenz unbekannt ist. Wird diese Forderung nicht eingehalten, tritt der sog. Leckeffekt auf, der das mit der DFT bzw. FFT berechnete Spektrum verfälscht.

Diese Verfälschung muss reduziert werden. Hierzu wird verbreitet das sog. Windowing verwendet, das die abgetasteten Signalwerte gewichtet und so den Leckeffekt mindert, jedoch nicht eliminiert.

Die vorliegende Arbeit stellt eine alternative, mathematische Lösungsmethode vor, die den Leckeffekt vollständig eliminieren und ein unverfälschtes Spektrum berechnen soll. Zusätzlich wird diese Methode auf ihre Leistungsfähigkeit hin untersucht und optimiert. Abschließend wird der entwickelte Algorithmus in die Hardwarebeschreibungssprache VHDL umgesetzt.

2. Der Leckeffekt

Wendet man die DFT oder die FFT auf ein nicht ganzzahliges Vielfaches einer Signalperiode an, entsteht der sog. Leckeffekt. Dadurch wird das errechnete Spektrum unscharf, die eigentlich im Signal enthaltenen Frequenzanteile werden nicht als Hauptpeaks dargestellt, sondern durch mehrere Nebenpeaks angedeutet. Weiterhin entstehen im durch den Leckeffekt verunreinigten Spektrum Frequenzanteile, die im Ursprungssignal nicht enthalten sind.

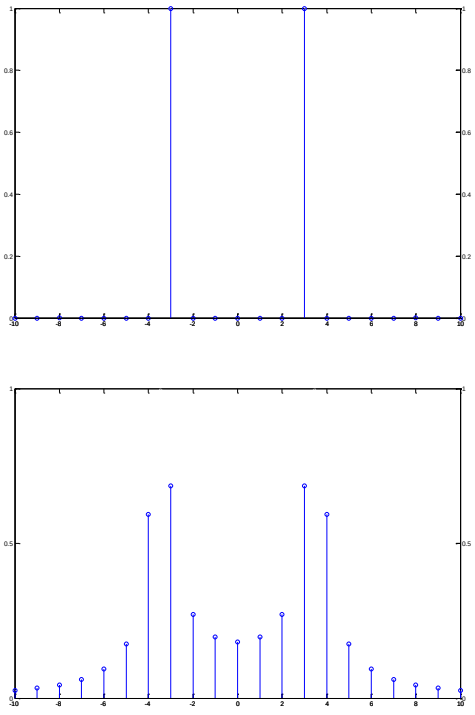


Abb. 2.1 – Frequenzspektren zweier Signale ohne und mit Leckeffekt

Abb. 2.1 zeigt oben das Spektrum eines Sinussignals mit $\omega_0 = 3 \cdot \omega_P$ (kein Leckeffekt) und unten das Spektrum eines Sinussignals mit $\omega_0 = 3{,}5 \cdot \omega_P$ (mit Leckeffekt). Bereits hier lässt sich erkennen, dass der Quotient $v = \frac{\omega_0}{\omega_P}$ das Vielfache einer abgetasteten Signalperiode ist. Formt man ω_0 und ω_P nach T_0 (Periodendauer des Signals) und T_P (Dauer des Abtastvorgangs) um, ergibt sich $T_P = v \cdot T_0$. Somit wurden v Signalperioden abgetastet.

2.1. *Entstehung (nach /1/)*

Mathematisch gesehen ist das zu transformierende Signal das Produkt (im Frequenzbereich das Faltungsprodukt) aus dem unendlich andauernden Ursprungssignal und einer Rechteckfunktion, mit $r(t) = 1$ für $0 < t < T_P$. Rechteckfunktionen im Zeitbereich korrespondieren mit der sog. Sinus-Cardinalis-Funktion (Spaltfunktion, si-Funktion) mit $si(x) = \frac{\sin(x)}{x}$ und $si(0) = 1$ im Frequenzbereich. Das Frequenzspektrum ist also nichts anderes als eine si-Funktion[1] mit Scheitelpunkt bei $x_S = \omega_0$. Für ein gewöhnlich von $-\frac{\omega_A}{2}$ bis $\frac{\omega_A}{2}$ (oder von 0 bis ω_A) betrachtetes Spektrum ergeben sich durch die Periodizität pro Spektralanteil zwei si-Funktionen, die sich additiv überlagern. Dies verdeutlicht folgender Beweis nach /1/. Es wird ein cosinusförmiges Signal transformiert:

$$x(t) = r_{\frac{T_p}{2}}(t) \cdot \cos(\omega_0 \cdot t) \quad \text{(II.I)}$$

$$r_{\frac{T_p}{2}}(t) \circ\!\!-\!\!\bullet\, T_P \cdot si(\omega \cdot \frac{T_P}{2})$$

$$\cos(\omega_0 \cdot t) \circ\!\!-\!\!\bullet\, \pi \cdot (\delta(\omega - \omega_0) + \delta(\omega + \omega_0))$$

Transformation von (II.I): Faltung im Frequenzbereich

$$X(\omega) = \frac{1}{2\pi} \int_{-\infty}^{\infty} X_1(y) \cdot X_2(\omega - y) dy$$

$$= \frac{T_P \cdot \pi}{2\pi} \int_{-\infty}^{\infty} si(y \cdot \frac{T_P}{2}) \cdot (\delta(\omega - \omega_0 - y) + \delta(\omega + \omega_0 - y)) dy$$

$$X(\omega) = \frac{T_P}{2} \left(si\left((\omega - \omega_0) \cdot \frac{T_P}{2} \right) + si\left((\omega + \omega_0) \cdot \frac{T_P}{2} \right) \right) \quad \text{(II.II)}$$

Da DFT und FFT aber nur diskrete Spektren kennen, ist auch eine si-Funktion nur an diskreten, ganzzahligen Stützstellen definiert. Im Spektrum wird die si-Funktion nun über ω an den Stützstellen $k = n \cdot \omega_P$ mit $-\frac{N}{2} \cdot \omega_P \le \omega \le \frac{N}{2} \cdot \omega_P$ oder $0 \le \omega_P \le N \cdot \omega_P$ aufgetragen. Ist nun der Quotient $\frac{\omega_0}{\omega_P}$ ganzzahlig, fällt der Scheitelpunkt der si-Funktion exakt auf eine Stützstelle und es entsteht dort eine Spektrallinie. Da $\sin(k \cdot \pi) = 0$ ist, ist die si-Funktion an allen anderen Stützstellen null. Ist $\frac{\omega_0}{\omega_P}$ nicht ganzzahlig, verschieben sich Scheitelpunkt und Nullstellen der si-Funktion von den Stützstellen weg in die undefinierten Bereiche und der Leckeffekt entsteht. Abbildung 2.2 veranschaulicht dies:

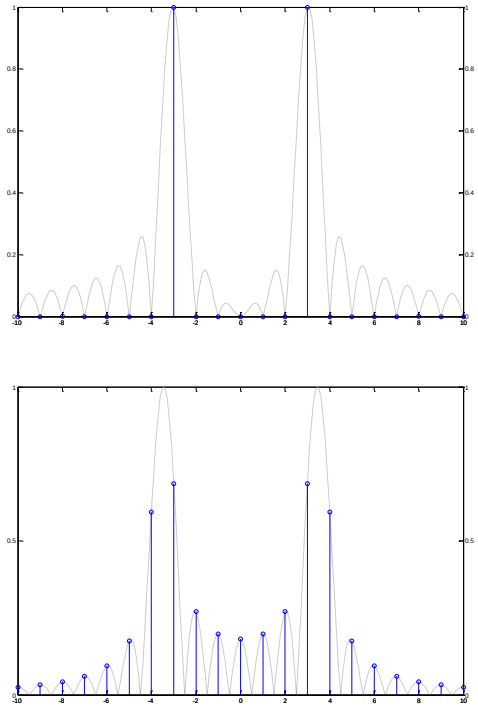


Abb. 2.2 – Frequenzspektren zweier Signale mit einhüllender si-Funktion (DarstellungLeckeffekt.m)

2.2. *Abhilfe*

Eine verbreitete Methode, den Leckeffekt zu reduzieren ist das sog. Windowing. Hierbei wird das Eingangssignal nicht mit einem Rechteckfenster, sondern mit sinusartigen (z.B. Hanning-Fenster) oder dreieckförmigen Fenstern multipliziert. Indem die Abtastwerte zu Beginn und am Ende der Abtastperiode mit Werten um null (und nicht wie beim Rechteckfenster sofort mit eins) multipliziert werden, sollen die Unstetigkeiten, die durch das Auswerten eines nicht ganzzahligen Vielfachen der Signalperiode entstanden sind, eliminiert werden. Die auszuwertende Zahlenfolge beginnt und endet somit bei oder nahe null. Durch das Windowing sind die Hauptmaxima zwar deutlicher von den Leckeffekt-Peaks zu unterscheiden, das Spektrum an sich wird jedoch unschärfer, da das Hauptmaximum breiter wird./1/

[1] Genau genommen handelt es sich um die Betragsfunktion einer si-Funktion. Dies spielt für die weiteren Betrachtungen jedoch keine Rolle.

3. Der Algorithmus zur Eliminierung des Leckeffektes

Zur Vermeidung des Leckeffektes muss die Anzahl der Abtastwerte so angepasst werden, dass die Transformation eines ganzzahligen Periodenvielfachen möglich wird. Dieses Ziel lässt sich mit folgendem Algorithmus erreichen:

- Transformation des Eingangssignals und Berechnung der Anzahl der abgetasteten Signalperioden
- Ermitteln der Anzahl der Abtastwerte für eine Signalperiode
- Errechnen der maximalen Anzahl der Abtastwerte für ein ganzzahliges Vielfaches der Signalperiode
- Transformation des Signals mit korrigierter Anzahl der Abtastwerte

3.1. Transformation des Eingangssignals und Berechnung des abgetasteten Signal-periodenvielfachen

Nach Kap. 2.1 ist der x-Wert des Scheitelpunktes der einhüllenden si-Funktion das abgetastete Periodenvielfache. Um diesen zu berechnen, muss die si-Funktion genauer analysiert werden.

Die in Abb. 5.1 dargestellte si-Funktion lässt sich mit der Funktionsgleichung

$$f(x) = si(x - x_0)$$
$$= si(x - (k + \alpha)) \quad | k \in \aleph, \alpha \in \Re, 0 \le \alpha < 1$$

beschreiben, wobei der Scheitelpunkt bei $x_S = (k + \alpha)$ liegt und k der größte ganzzahlige x-Wert (die größte ganzzahlige Stützstelle) vor dem Scheitelpunkt und $\alpha = x_S - k$ ist. Da der Scheitelpunkt der si-Funktion immer zwischen den zwei höchsten Spektrallinien zu suchen ist, kann der Wert von k einfach ermittelt werden: Sein Wert ist der x-Wert der ersten der beiden den Scheitelpunkt umgebenden Spektrallinien.

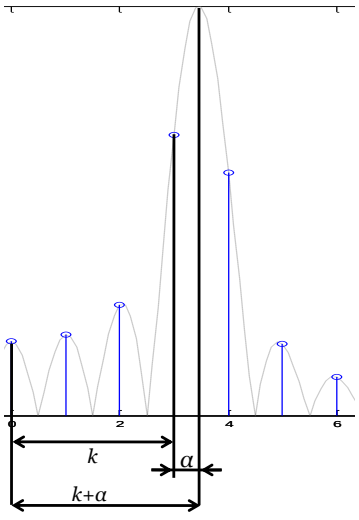


Abb. 3.1 – Mathematische Darstellung einer si-Funktion

Die Berechnung von α wird nachfolgend erläutert.

Allgemein lässt sich α aus dem Verhältnis zweier nebeneinander liegender Spektrallinien im Spektrum bestimmen, was mit folgender Herleitung veranschaulicht werden soll:

$$\frac{f(k+1)}{f(k)} = \frac{si(k+1-k-\alpha)}{si(k-k-\alpha)}$$
$$= \frac{si(1-\alpha)}{si(-\alpha)}$$
$$= \frac{si(\pi \cdot (1-\alpha))}{si(-\pi \cdot \alpha)}$$
$$= \frac{\sin(\pi \cdot (1-\alpha)) \cdot (-\pi \cdot \alpha)}{(\pi \cdot (1-\alpha)) \cdot \sin(-\pi \cdot \alpha)}$$

Symmetriebetrachtungen zeigen:
$\sin(-\pi \cdot \alpha) = -\sin(\pi \cdot (1-\alpha))$

$$\frac{f(k+1)}{f(k)} = \frac{-\alpha}{\alpha - 1} \quad \left| \cdot \frac{-1}{-1} \right.$$
$$\frac{f(k+1)}{f(k)} = \frac{\alpha}{1-\alpha} \quad \text{(III.I)}$$

Daraus folgt:

$$\frac{f(k)}{f(k+1)} = \frac{1-\alpha}{\alpha}$$
$$\frac{f(k)}{f(k+1)} + 1 = \frac{1}{\alpha}$$
$$\alpha = \frac{1}{\frac{f(k)}{f(k+1)} + 1}$$
$$\alpha = \frac{f(k+1)}{f(k) + f(k+1)} \quad \text{(III.II)}$$

Nun kann das Periodenvielfache $\nu = k + \alpha$ berechnet werden.

Die Abszisse des Scheitelpunktes befindet sich zwischen den beiden höchsten Spektrallinien im Spektrum, so dass gilt $k < x_S < k+1$. Zur Berechnung von α müssen die Spektrallinien mit den beiden größten Ordinatenwerten im Intervall $\left[0, \frac{N}{2}\right]$ herangezogen werden.

Da die Anteile der si-Funktion bei $-x_S$ im Verhältnis zu den untersuchten Ordinatenwerten zunehmend kleiner werden, können sie – selbst bei niedrigen Frequenzen – in der Rechnung vernachlässigt werden.

3.2. Ermitteln der Anzahl der Abtastwerte für eine Signalperiode

Da nun sowohl die Anzahl der abgetasteten Signalperioden als auch die Anzahl der Abtastwerte pro Abtastzeitraum bekannt bzw. vorgegeben sind, lässt sich die Anzahl der Abtastwerte, die eine Signalperiode repräsentieren, errechnen.

$$N_{1P} = \frac{N}{k+\alpha} = \frac{N}{\nu}$$

3.3. *Errechnen der maximalen Anzahl der Abtastwerte für ein ganzzahliges Vielfaches der Signalperiode*

Mit k ist das maximale ganzzahlige Periodenvielfache (die Anzahl der komplett abgetasteten Signalperioden) gegeben. Multipliziert mit N_{1P} erhält man die Anzahl der Abtastwerte, die für eine zweite Transformationsrechnung herangezogen werden sollte:

$$N_{neu} = k \cdot N_{1P}$$

Da N_{neu} in der Regel nicht ganzzahlig ist, muss für die weitere Arbeit eine Rundung von N_{neu} auf die jeweils nächste ganze Zahl vorgenommen werden.

3.4. *Probleme*

Hinsichtlich der Wirksamkeit des Algorithmus stellen sich verschiedene Probleme:

3.4.1. Rundungsfehler Δ_R

Bei einem zu großen Rundungsfehler $\Delta_R = | N_{neu,ungerundet} - N_{neu,gerundet} |$ ist der oben beschriebene Algorithmus in seiner Wirkung – besonders bei höheren Frequenzen – deutlich vermindert. Dies ist darauf zurückzuführen, dass eine Signalperiode nicht auf oder unmittelbar in der Nähe eines Abtastpunktes endet, sondern im Extremfall genau zwischen zwei Abtastpunkten. Auf diese Weise wird die Transformation wiederum mit einem nicht ganzzahligem Periodenvielfachen durchgeführt und der Leckeffekt bleibt, wenn auch vermindert, existent.

Beobachtungen haben gezeigt, dass Δ_R bis zu einem Wert von etwa $0{,}15$ tolerierbar ist. Bei einer Überschreitung müssen Korrekturmaßnahmen getroffen werden:

Im Regelfall wird, begünstigt durch eine entsprechend lange Beobachtungszeit, deutlich mehr als eine Signalperiode zur Transformationsrechnung herangezogen. Diese Redundanz ist für die Minimierung von Δ_R sehr nützlich. Da das Ergebnis der Transformation unabhängig von der herangezogenen Periodenzahl ist, ist es möglich, für die zweite Transformationsrechnung eine Periodenzahl kleiner k mit einem tolerierbaren Δ_R heranzuziehen.

Die neue Anzahl der Abtastwerte wird so lange um die Anzahl der Abtastwerte für eine Periode verringert, bis Δ_R im Toleranzbereich liegt.

3.4.2. Versagen beim zufälligen Abtasten eines ganzzahligen Periodenvielfachen

Wird zufällig ein exakt ganzzahliges Periodenvielfaches des zu überprüfenden Signals abgetastet, hat die zweithöchste Spektrallinie einen Wert nahe oder gleich null. Da nicht auszuschließen ist, dass $f(k+1)$ die höchste Spektrallinie repräsentiert (und somit $f(k) \approx 0$ ist), kann $\alpha = \frac{f(k+1)}{f(k)+f(k+1)} = 1$ werden, was jedoch nicht zulässig ist. Nach Kap. 0 ist $k+\alpha$ das exakte Vielfache einer abgetasteten Signalperiode. Weiterhin muss für exakt ganzzahlige Periodenvielfache nach Kap. 3.1 $\alpha = 0$ sein. Ist $\alpha = 1$, wird mit einem fälschlicherweise um 1 zu hohen Periodenvielfachen gerechnet, was zu einem fehlerhaften N_{neu} und somit zu einem falschen Ergebnis führt. Abhilfe schafft das Nullsetzen von α.

In der Praxis macht sich dieses Problem bereits ab $\alpha \approx 0{,}95$ bemerkbar. Daher werden bereits alle $\alpha > 0{,}95$ nullgesetzt. Auf Grund der Symmetrie einer si-Funktion werden alle $\alpha < 0{,}05$ ebenfalls nullgesetzt (nicht zuletzt, um Rechenaufwand zu sparen).

Für $\alpha = 0$ ist eine zweite Transformation natürlich nicht notwendig, das Spektrum aus der ersten Transformationsrechnung kann somit direkt ausgegeben werden.

3.4.3. Anzahl der Abtastwerte für eine FFT

Damit eine FFT angewendet werden kann, muss die Anzahl der Abtastwerte unbedingt einer Zweierpotenz entsprechen. Diese Bedingung wird vom vorgestellten Algorithmus natürlich verletzt. Zur Abhilfe ließe sich ggf. eine Interpolation von 2^m Stützstellen durchführen. Diese Methode versagt allerdings zunehmend bei höheren Frequenzen, das errechnete Frequenzspektrum wäre erneut fehlerhaft.

Auch das Auffüllen der übrigen Abtastwerte mit Nullen – das sog. Zero-Padding – ist hier nutzlos, weil sich hierdurch ein vom Ursprungssignal abweichendes Signal ergeben würde, das das Spektrum abermals verfälscht.

Die scheinbare Unvereinbarkeit der zwei Bedingungen „ganzzahliges Periodenvielfaches“ und „2^m Stützstellen“ wird somit zum zentralen Problem hinsichtlich der Anwendbarkeit des Algorithmus bei einer FFT.

Unberührt davon bleibt hingegen die Anwendbarkeit bei einer DFT. Da die Stützstellenanzahl jedoch variabel sein muss, stellt der Algorithmus zur Eliminierung des Leckeffektes hohe Anforderungen an die Flexibilität des DFT-Algorithmus. Deshalb wird für die DFT die Transformationsvorschrift

$$X[k] = \frac{1}{N} \sum_{n=0}^{N-1} x[n] \cdot e^{-j2\pi \frac{nk}{N}} = \frac{1}{N} \sum_{n=0}^{N-1} x[n] \cdot W_N^{nk}$$

1:1 implementiert. Der Nutzer muss daher einen wesentlich höheren Hardware-Aufwand und deutlich mehr Rechenzeit

in Kauf nehmen. Letzteres kann durch den Einsatz schneller Zielhardware teilweise kompensiert werden.

4. Realisierung des Algorithmus in VHDL

Die VHDL-Implementierung des Algorithmus muss folgendes leisten können:

- Einlesen der Abtastwerte des A/D-Wandlers
- Berechnung einer DFT mit dynamisch variierbarer Länge
- Neuberechnung der Länge der zweiten DFT (eigentliche Eliminierung des Leckeffektes)
- Ausgabe des Spektrums mit eliminiertem Leckeffekt

Sinnvollerweise teilt man diese vier Aufgaben je einem eigenständig funktionierenden VHDL-Design zu. Abbildung 4.1 veranschaulicht das Zusammenspiel der einzelnen VHDL-Designs. Dabei werden zunächst die Abtastwerte des A/D-Wandlers eingelesen und in RAM 0 abgelegt. Anschließend wird aus diesen Werten das Spektrum berechnet und in RAM 1 gespeichert. Hieraus werden die zwei größten Spektrallinien ermittelt und zur Berechnung der Länge der zweiten DFT herangezogen. Nun erfolgt die zweite Transformationsrechnung, deren Ergebnis abermals in RAM 1 abgelegt wird. Zum Schluss wird das Ergebnis aus RAM 1 ausgelesen und ausgegeben. Das Zusammenspiel der einzelnen Module koordiniert eine übergeordnete Finite State Machine (FSM), die, abgesehen von den Speichermodulen, jedes Modul ansteuert.

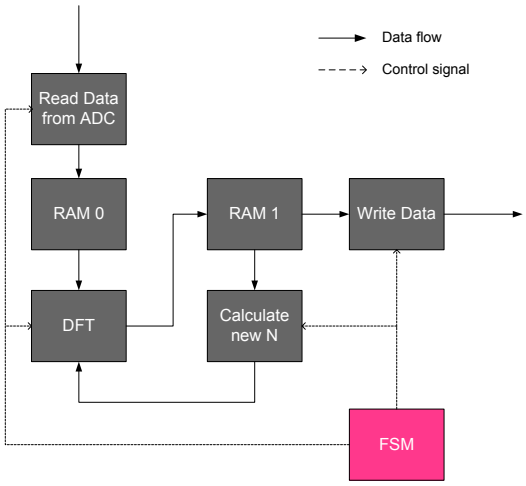


Abb. 4.1 – Schematische Funktionsweise des VHDL-Designs

4.1. Implementierung der einzelnen Module

Zum tieferen Verständnis des VHDL-Designs werden nachfolgend wichtige Aspekte zu den Funktionen der einzelnen im Design eingesetzten Module erläutert. Da die Implementierung des Einlesens der Daten vom A/D-Wandler und der Ausgabe des bereinigten Spektrums sehr einfach ist und auf viele verschiedene Arten (FSM, Zählvariable usw.) erfolgen kann, wird nicht näher darauf eingegangen. Im Fokus stehen die Module zur Berechnung der DFT und der neuen Länge, sowie die FSM zur Prozesssteuerung.

4.1.1. Modul zur Berechnung der Diskreten Fourier-Transformation

Zunächst werden Aufbau und Funktionsweise des DFT-Moduls als eines der drei zentralen VHDL-Module beschrieben.

4.1.1.1. Einschränkungen und Rahmenbedingungen

Wie in Kap. 3.4.3 erläutert, muss für eine ausreichende Flexibilität die Transformationsvorschrift

$$X[k] = \frac{1}{N}\sum_{n=0}^{N-1} x[n]\cdot e^{-j2\pi\frac{nk}{N}} = \frac{1}{N}\sum_{n=0}^{N-1} x[n]\cdot W_N^{nk}$$

direkt implementiert werden.

Die Forderung nach einem synthetisierbaren und lauffähigen VHDL-Design erzwingt zudem diverse Rahmenbedingungen und Einschränkungen, die beim Programmieren zu beachten sind.

a) Realisierung der Schleifenstrukturen

Bereits aus der Transformationsvorschrift ist zu erkennen, dass sich im Laufe eines Transformationszyklusses die Rechenoperationen wiederholen. Bei Verwendung einer klassischen Programmierhochsprache wie C würde man den Algorithmus mittels Schleifen implementieren. In VHDL ist dies jedoch nicht ohne weiteres möglich. While-Schleifen scheiden wegen ihrer Syntheseunfähigkeit sofort aus. For-Schleifen hingegen sind zwar ohne weiteres synthetisierbar, allerdings muss ihre Länge zum Kompilationszeitpunkt bekannt sein, da für jeden Schleifendurchgang Ressourcen auf der Zielhardware benötigt werden. Dies impliziert, dass sich die Anzahl der Schleifendurchgänge zur Laufzeit nicht mehr ändern kann. Da der Eliminierungsalgorithmus jedoch zwei Transformationsrechnungen mit jeweils unterschiedlicher Länge (und somit unterschiedlich vielen Schleifendurchgängen) verlangt, scheidet auch die Anwendung von For-Schleifen aus.

Stattdessen kommt für die Transformationsberechnung eine Finite State Machine als leichte Abwandlung eines Zustandsautomaten zum Einsatz. Sie ist in ihrer Implementierung zwar deutlich aufwändiger als eine Schleife, bietet dafür aber wesentlich mehr Flexibilität. Besonders interessant ist die Anwendung von Steuersignalen zur Triggerung der Zustandsübergänge. So lassen sich auch Operationen ausführen, die mehr Zeit als einen Taktzyklus in Anspruch nehmen, was mit einer Schleife, die einmal pro Taktzyklus komplett durchläuft, unmöglich wäre.

b) Umsetzung der Rechenoperationen in Binärarithmetik

Da es zurzeit nur möglich ist, Berechnungen mit Fließkommazahlen in VHDL zu simulieren, jedoch nicht zu synthetisieren, müssen alle Rechenoperationen in Binärarithmetik erfolgen. Da sowohl positive als auch negative Zahlen in die Berechnung einfließen, ist die

Interpretation der Binärwerte als Zweierkomplementzahlen zweckmäßig. Speziell für Rechenoperationen mit Zweierkomplementzahlen bietet das vom IEEE genormte VHDL-Package `std_logic_signed` die nötigen Implementierungen der Grundrechenarten Addition, Subtraktion und Multiplikation an. Möglichkeiten der Division und der Exponentialrechnung fehlen jedoch.

Nach Euler kann $e^{j\varphi}$ in $\cos(\varphi) + j \cdot \sin(\varphi)$ überführt werden, was eine Exponentialoperation unnötig macht. Trigonometrische Berechnungen sind zwar auch kein Bestandteil des Packages `std_logic_signed`, allerdings können sie mittels des sog. CORDIC[1]-Algorithmus durchgeführt werden. Dieser Algorithmus ermöglicht neben den Berechnungen von Sinus und Cosinus auch das Berechnen von Potenzen und Logarithmen. Im Gegensatz zu einer genauso möglichen Reihenentwicklung ist der CORDIC-Algorithmus jedoch um ein Vielfaches leistungsfähiger.

Die Industrie (z.B. Altera®) bietet über das Internet käufliche CORDIC-Module an, es gibt aber auch einige wenige frei verfügbare. Für die Implementierung des DFT-Moduls wurde ein frei erhältlicher, in VHDL geschriebener CORDIC-Baustein verwendet, dessen Funktionsumfang sich allerdings auf die trigonometrischen Berechnungen beschränkt (was aber völlig ausreicht). Er ist im Internet unter *www.ht-lab.com/freecores/cordic/cordic.html* (Stand: 01.Oktober 2008) kostenlos herunterzuladen.

Das exakte Verständnis des CORDIC-Algorithmus ist für vorliegende Arbeit unwichtig. Es sei nur soviel gesagt, dass die Sinus- und Cosinusberechnungen iterativ durch Rotation von Koordinatensystemen erfolgen. Nachfolgend kann das CORDIC-Modul als Black Box betrachtet werden, die aus einem (im Zweierkomplement und im Bogenmaß dargestellten) Winkel die dazugehörigen Sinus- und Cosinuswerte berechnet.

Zur Berechnung der Division ist in die Entwicklungsumgebung Quartus®, die die Firma Altera® für die Entwicklung von Software mit ihren FPGAs mitliefert, eine Bibliothek integriert, die dem Entwickler eine Reihe häufig benötigter Funktionen (sog. Mega Functions) in Form fertig implementierter VHDL-Module zur Verfügung stellt. Dies ist auf den ersten Blick sehr hilfreich für den Entwickler, allerdings wird er durch den Einbau einer Mega Function in sein Design an die Verwendung von Altera®-FPGAs gebunden. Auf FPGAs anderer Hersteller sind diese Mega Functions höchstwahrscheinlich nicht lauffähig. Allerdings liefern alle großen Hersteller solche Bibliotheken mit, so dass die Portierung des hier entwickelten Designs auf die Hardware anderer Hersteller keine größeren Probleme bereiten dürfte.

c) Komplexe Zahlen

Da j bzw. i nicht binär darstellbar ist, müssen die reellen und komplexen Spektralanteile während der Berechnung der DFT getrennt berechnet werden. Erst bei der Berechnung des Betragsspektrums werden sie über $|X[k]| = \sqrt{\Re^2(X[k]) + \Im^2(X[k])}$ zusammengeführt. Daraus wird eine Wurzelberechnung notwendig. Auch dies wird von einem Mega-Function-Modul der Firma Altera® ausgeführt.

d) Festkommaarithmetik

Da während des gesamten Prozesses die Rechenergebnisse als Binärwerte vorliegen, muss berücksichtigt werden, wie sich das – rein fiktive – Komma abhängig von der ausgeführten Rechenoperation verschiebt. Weiterhin ist die sich ständig ändernde Wortbreite der Zwischenergebnisse zu beachten. Damit keine Überläufe entstehen, muss sie für jede durchzuführende Rechnung passend gewählt werden.

4.1.1.2. Design der Finite State Machine

Die Rechenvorschriften

$$\Re(X[k]) = \sum_{n=0}^{N-1} x[n] \cdot \cos(2\pi \frac{nk}{N}), \quad \Im(X[k]) = -\sum_{n=0}^{N-1} x[n] \cdot \sin(2\pi \frac{nk}{N})$$

und $|X[k]| = \sqrt{\Re^2(X[k]) + \Im^2(X[k])}$ erfordern eine schrittweise Berechnung des Ergebnisses. Sinnvollerweise sieht man für die einzelnen Schritte jeweils entsprechende Zustände vor. Zunächst werden die Argumente für die Sinus- und Cosinusberechnung errechnet und anschließend der Sinus und Cosinus selbst. Nun erfolgt die Aufsummierung des Produktes von Sinus bzw. Cosinus mit dem Eingangssignal, zuletzt die Berechnung des Betrages. Zusätzlich zu diesen vier Zuständen werden ein Ruhezustand, sowie ein Reset-Zustand und ein Zustand zum Abspeichern der Rechenergebnisse in ein RAM benötigt. Abbildung 4.2 veranschaulicht den Aufbau der hier zum Einsatz kommenden FSM:

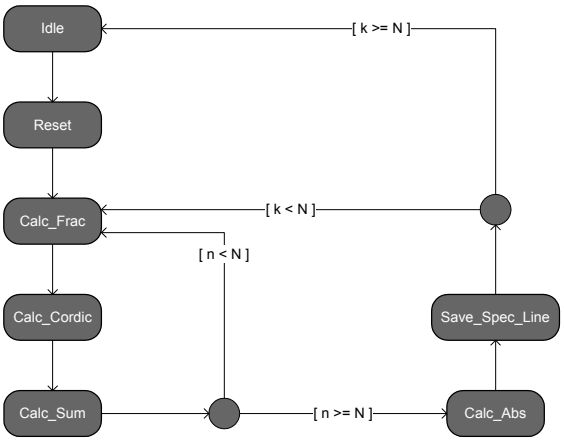


Abb. 4.2 – Aufbau der FSM zur Berechnung der DFT

4.1.2. Modul zur Neuberechnung der DFT-Länge

Als zweites zentrales VHDL-Modul wertet das Modul `calcnewlength` das zuvor berechnete Frequenzspektrum aus und berechnet die neue Länge für den zweiten Transformationsvorgang. Zur weiteren Verarbeitung wird dem DFT-Modul ein Begrenzer als Differenz zwischen

[1] Coordinate Rotation Digital Computer

ursprünglicher und neuer Länge übermittelt. Dies bringt Vorteile bei der Implementierung des DFT-Moduls, da sich die jeweilige Länge der DFT immer über die Differenz der Ursprungslänge und des Längenbegrenzers errechnet. Soll eine DFT mit Ursprungslänge durchgeführt werden, ist der Längenbegrenzer null.

4.1.2.1. *Einschränkungen und Rahmenbedingungen*

Da auch hier wiederholte Anweisungen und bedingte Aweisungssequenzen ausgeführt werden müssen, wird ebenfalls auf das Entwurfsmuster „Zustandsautomat" (FSM) zurückgegriffen, das sich bereits bei der Implementation des DFT-Moduls bewährt hat (s. Kap.4.1.1.1 a)).

Für die Arithmetik stehen nach wie vor die Mega-Function-Module von Altera® zur Verfügung. Nicht nötig hingegen sind vorzeichenbehaftete Arithmetikoperationen, da alle vorkommenden Zahlenwerte positiv sind. Dies ermöglicht die Anwendung des Packages `std_logic_unsigned`, das dieselben Funktionen wie das Package `std_logic_signed` implementiert, die Argumente jedoch als nicht vorzeichenbehaftet interpretiert. Logischerweise müssen die Variablen kein Vorzeichenbit mehr mitführen. Somit ist auch der Hardwareaufwand im Vergleich zur vorzeichenbehafteten Arithmetik (wenn auch nur geringfügig) geringer.

4.1.2.2. *Design der Finite State Machine*

Für die Neuberechnung der Länge sind zunächst einmal drei Schritte zwingend notwendig, denen man sinnvollerweise je einen Zustand zuspricht:

- Ermittlung der höchsten Spektrallinien
- Berechnung von α
- Neuberechnung der DFT-Länge

Weiterhin werden Zustände zur Auswertung von α (s. Kap.3.4.2) und des Rundungsfehlers (s. Kap.3.4.1) benötigt, ehe der Begrenzer in einem eigenen Zustand ausgegeben wird. Dazu sind ein Ruhezustand sowie ein Reset-Zustand zur Sicherstellung eines definierten Ausgangszustandes obligatorisch. Abb. 4.3 zeigt den schematischen Aufbau der FSM zur Neuberechnung der DFT-Länge.

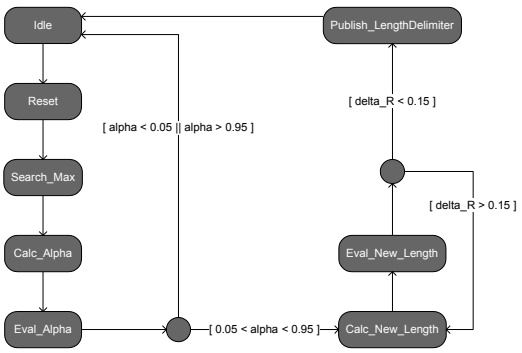


Abb. 4.3 – FSM zur Neuberechnung der DFT-Länge

4.1.3. FSM zur Steuerung des Gesamtablaufes

Damit die einzelnen VHDL-Module korrekt zusammenarbeiten können, müssen sie miteinander kommunizieren. Da die Implementierung von direkten Kommunikationsmöglichkeiten zwischen den einzelnen Modulen sehr aufwändig wäre und auch deren Wiederverwendbarkeit (insbesondere die des DFT-Moduls) stark einschränken würde, werden die Kommunikation und die Prozesssteuerung von einer übergeordneten Finite State Machine abgewickelt.

4.1.3.1. *Prozesssteuerung und Kommunikation*

Die FSM übernimmt die Ansteuerung jedes einzelnen Modules (abgesehen von den Speicherblöcken, die direkt von den einzelnen Modulen angesteuert werden) und ist zentraler Kommunikationsknotenpunkt. Dabei kommunizieren die einzelnen Module niemals direkt untereinander, sondern stets über die FSM. Zur Ansteuerung der einzelnen Module existiert pro Modul eine Leitung zur Übermittlung des Enable-Signals (`mod_ena`; die ersten drei Buchstaben identifizieren das Modul, ena = enable). Wird dieses gesetzt, nimmt das Modul seine Arbeit auf, vorausgesetzt, es befindet sich zum Zeitpunkt des Setzens von `mod_ena` im Ruhezustand. Hat ein Modul seine Arbeit aufgenommen, kann es im laufenden Betrieb nur durch das Setzen des Signals `mod_reset` angehalten werden, worauf das gesamte Modul wieder den Ruhezustand einnimmt (und später automatisch zurückgesetzt wird). Die einzelnen Reset-Signale werden entweder durch die FSM selbst, unmittelbar vor der Aktivierung eines Modules, oder durch einen globalen Reset über den Reset-Zustand der FSM gesetzt.

Zur Kommunikation besitzt jedes Modul eine Handshake-Leitung `mod_ready`. Sie signalisiert der FSM die Bereitschaft für eine erneute Aktivierung des Moduls. Sobald ein Modul aktiviert wird, wird `mod_ready` automatisch zurückgesetzt, bis es nach Abschluss der Arbeit wieder gesetzt wird. So erkennt die FSM, ob ein Modul aktiviert werden kann und wann ein Modul seine Arbeit beendet hat und das nächste Modul aktiviert werden kann. Standardmäßig ist das Ready-Flag im Ruhezustand eines Modules gesetzt. Sobald jedoch die Aktivierung durch ein Enable-Signal erfolgt, wird es asynchron zurückgesetzt und bleibt es solange, bis der Ruhezustand wieder erreicht und das Enable-Signal zurückgesetzt wird. So wird eine versehentliche zweite Aktivierung des Modules nach Abschluss der Arbeit verhindert. Dies kann durch die Taktflankentriggerung passieren, wenn ein Modul nach Abschluss seiner Arbeit sein Ready-Flag setzt und zu diesem Zeitpunkt noch das Enable-Signal der FSM am Modul anliegt.

Zusätzlich besitzen das DFT-Modul, das Modul zur Berechnung der neuen DFT-Länge sowie das Modul zur Ausgabe des bereinigten Spektrums eine Leitung zur Übermittlung des Längenbegrenzers. Damit der Wert des Längenbegrenzers dauerhaft und konsistent verfügbar ist, wird er in der FSM abgespeichert und von dort aus zur Verfügung gestellt.
Im hier beschriebenen VHDL-Design ist die übergeordnete FSM für die Koordinierung der DFT-Länge zuständig. Beim

ersten Durchgang übermittelt sie dem DFT-Block den Wert `0` (somit wird eine DFT über die volle Länge berechnet), im zweiten Durchgang übermittelt sie den Wert des Längenbegrenzers.

4.1.3.2. Design und Funktion

Für das Design der FSM sind drei Aspekte von zentraler Bedeutung.

a) Zustände

Da im Betrieb zu keiner Zeit zwei Module gleichzeitig aktiv sind, bietet es sich an, pro Modul einen Zustand vorzusehen, in dem das entsprechende Modul aktiv ist. Dazu werden ein Reset-Zustand zur Schaffung eines definierten Ausgangszustandes, sowohl der FSM als auch der angesteuerten Module, und ein Ruhezustand benötigt, in dem alle Module inaktiv sind.

b) Betriebsmodi

Der Betrieb kann in zwei unterschiedlichen Betriebsmodi erfolgen, die über das Signal `mode` eingestellt werden können. Ist `mode` zurückgesetzt, wird auf eine Anforderung ein Arbeitszyklus durchgeführt. Anschließend versetzt sich die FSM wieder in den Ruhezustand. Bei gesetztem `mode` arbeitet die FSM kontinuierlich, d.h. nach der Ausgabe des bereinigten Spektrums erfolgt sofort ein erneutes Einlesen der Werte des A/D-Wandlers. Eine Änderung von `mode` darf zu beliebigen Zeitpunkten erfolgen, da sein Zustand im laufenden Betrieb nicht von Bedeutung ist. Lediglich im Zustand `write_data` entscheidet der Zustand von `mode`, ob ein erneuter Arbeitszyklus gestartet werden (Folgezustand ist `read_data` bei gesetztem `mode`) oder sich die FSM zur Ruhe setzen soll (Folgezustand ist `idle` bei zurückgesetztem `mode`). Wird `mode` im Ruhezustand der FSM gesetzt, bewirkt dies eine erneute Aktivierung der FSM.

c) Zustandsübergänge

Für einen Zustandsübergang aus Zustand A in Zustand B müssen im Wesentlichen zwei Bedingungen erfüllt werden: Das in Zustand A aktive Modul muss signalisieren, dass seine Arbeit beendet wurde, während das in Zustand B aktive Modul seine Arbeitsbereitschaft anzeigen muss. Da ein Zustand in der Regel mehr als einen Folgezustand hat, müssen die Übergangsbedingungen zur eindeutigen Identifizierung des Folgezustandes erweitert werden. Hierfür werden die Zustände folgender Signale ausgelesen:

- `mode`

  Entscheidet, was nach erfolgreichem Abschluss eines Arbeitszyklus, bzw. nach einem globalen Reset geschieht.
- `cnl_done`

  Ist gesetzt, wenn die DFT-Länge berechnet wurde. Dann erfolgt nach der (ggf. zweiten) DFT direkt die Datenausgabe.
- `cnl_lengthdelim`

  Ist ungleich `0`, wenn eine zweite DFT nötig ist.

Aufbau und Funktionsweise werden in Abb. 4.4 noch einmal schematisch dargestellt:

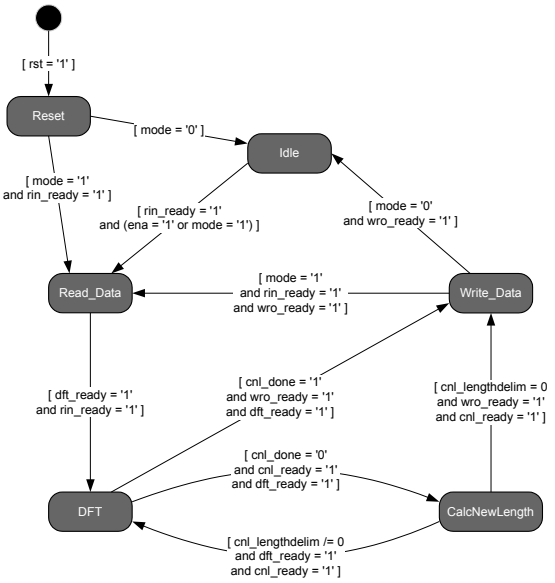


Abb. 4.4 – Aufbau der FSM zur Steuerung des Gesamtablaufes

5. Zusammenfassung

Der hier vorgestellte Algorithmus ist in der Lage, den Leckeffekt, der bei der DFT und der FFT entsteht, bis auf ein Mindestmaß zu reduzieren bzw. ihn gänzlich zu eliminieren. Dabei wird auf das verbreitet angewandte Windowing verzichtet, vielmehr wird der Leckeffekt durch die Veränderung der Anzahl der zur Transformation herangezogenen Signalabtastwerte unterdrückt.

Der Algorithmus wurde unter Zuhilfenahme des Mathematikprogrammes MATLAB® erfolgreich entwickelt, verifiziert und optimiert.

Zur Verwendung in der Praxis wurde der Algorithmus in der Hardwarebeschreibungssprache VHDL synthesefähig implementiert. Mit ca. 6.600 Logischen Einheiten ist der Platzbedarf auf einem FPGA noch überschaubar. Es muss jedoch noch zusätzlicher Platz für das Erfassen der Abtastwerte des A/D-Wandlers und deren Speicherung in einen Speicher vorgesehen werden.

Fester Bestandteil des hier entwickelten VHDL-Designs ist ein Modul zur Berechnung der Diskreten Fourier-Transformation. Diese musste der Fast Fourier Transformation vorgezogen werden, da die Fast Fourier Transformation mit festen Längen von 2^m Stützstellen arbeitet, die Länge für den beschriebenen Algorithmus jedoch variabel sein muss. Der mit der Diskreten Fourier-Transformation einhergehende hohe Rechenaufwand äußert sich in einer Zykluszeit von 3,5 ms bei einer Taktfrequenz von 50 MHz. Dies erschwert den Einsatz des Algorithmus in zeit- und ressourcenkritischen Systemen (Mobiltelefon) erheblich.

Da vorliegende Arbeit auf vorimplementierte VHDL-Module, sog. Mega Functions der Firma Altera® zurückgreift, kann das vorgestellte Design nur für FPGAs

dieses Herstellers verwendet werden. Bei Verwendung von Hardware anderer Hersteller müssen diese Module gegen entsprechende Äquivalente ausgetauscht werden. Dies sollte jedoch problemlos möglich sein.

6. Ausblick

Die vorliegende Arbeit zeigt, dass der Algorithmus zur Eliminierung des Leckeffektes funktionsfähig ist. Es können jedoch noch einige Optimierungen vorgenommen werden.

Gegenwärtig variiert die Wortbreite der verwendeten Variablen sehr stark, um die Berechnungen möglichst genau durchzuführen. Damit verbunden ist ein gewisser Hardwareaufwand, besonders bei den Divisionsmodulen und dem Modul zur Wurzelberechnung, da die Wortbreite der berechneten Werte mit maximal 76bit durchaus beachtlich ist. Durch Untersuchungen, inwieweit sich eine verringerte Wortbreite auf die Genauigkeit der Ergebnisse auswirkt und einer anschließenden Modifikation des Designs ließe sich der Hardwareaufwand des Designs reduzieren.

Eine weitere Optimierungsmöglichkeit wäre die Realisierung der Portierbarkeit des entwickelten Designs auf Zielhardware anderer FPGA-Hersteller. Dazu müssten alle verwendeten Mega-Function-Module durch eigene, herstellerunabhängige Implementierungen ersetzt werden. Inwieweit sich dieser (insbesondere bei den Arithmetikmodulen beträchtliche) Aufwand jedoch rechtfertigt, ist fraglich (s. Kap. 5).

Interessant wäre die Optimierung des Designs durch die Entwicklung eines ausreichend flexiblen Transformationsalgorithmus, der weniger Rechenzeit und möglichst auch weniger Hardwareressourcen benötigt. Dies wäre z.B. möglich durch eine der Transformation vorausgehende Berechnung aller benötigten Sinus- und Cosinuswerte (deren Anzahl auf Grund der Periodizität stets endlich ist) und deren Abspeicherung in einem Speicher. So würde jeder benötigte Wert nur einmal berechnet werden. Der auf diese Weise eingesparten Rechenzeit stünde allerdings ein geringfügig höherer Ressourcenaufwand gegenüber. Es wäre auch zu untersuchen, ob sich ein für dieses Design geeigneter FFT-Algorithmus entwickeln ließe.

Der reizvollste Optimierungsansatz ist jedoch der völlige Verzicht auf eine zweite Transformationsrechnung. Kapitel 3 hat gezeigt, dass bereits aus dem leckeffektbehafteten Spektrum die Grundfrequenz eines Signales berechnet werden kann. Es müsste untersucht werden, ob auf diese Weise auch die Spektralanteile komplexerer Signale errechnet werden können. Dies brächte zwei entscheidende Gewinne. Erstens wäre zur Ermittlung aller benötigten Signalinformationen nur ein Transformationsvorgang nötig, zweitens ist hier auch eine echte Fast-Fourier-Transformation einsetzbar, was abermals den Hardware- und Rechenaufwand deutlich mindert. Diese gewichtigen Vorteile und die scheinbare Umsetzbarkeit machen diese Idee zu einem vielversprechenden Ansatz zur effektiven und genauen Berechnung der Spektralanteile eines Signales.

7. Literatur

/1/ Kreutzer, Hans — Digitale Signalverarbeitung
Arbeitsunterlagen zur Vorlesung
Hochschule Reutlingen, 2008

FE-RFID 15693

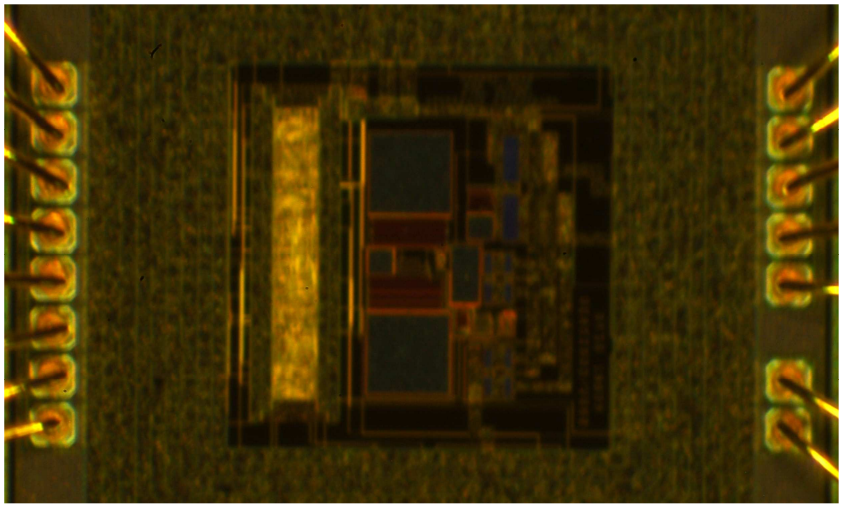
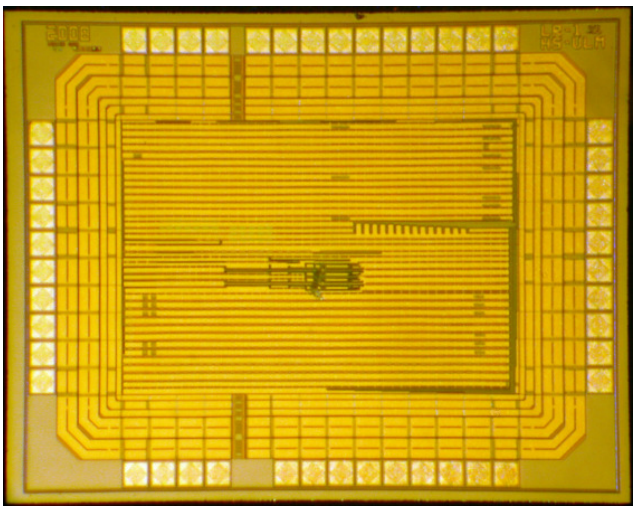


Abbildung 1: Fertiger Chip im Vergleich mit einem SOT23-Transistor

Entwurf:	Hochschule Offenburg Bearbeiter: Dipl.-Ing. Tobias Volk M.Sc. Daniel Bau Betreuer: Prof. Dr.-Ing. Dirk Jansen
Layouterstellung:	Hochschule Offenburg (Standardzellenentwurf & analog Design)
Technologie:	AMIS 0.35 µm CMOS A
Chipfertigung:	Europractice, MPW Run 1904 (Mini@sic)
Herstelldatum:	April 2008
Kostenträger:	MPC-Mittel HS-Verbund Baden-Württemberg
Chipdaten:	Chipgröße: 1314 µm x 1314 µm Gehäuse: QFN32 (5x5)
Funktion:	RFID Frontend für den ISO-Standard 15693. Mit Hilfe des ASICs kann ein Microcontroller über die RFID-Schnittstelle kommunizieren. Zur Kommunikation ist lediglich ein embedded Prozessor mit 4MHz Taktrate nötig.
Testergebnisse:	Bei der Inbetriebnahme wurde der digitale und analoge Schaltungsteil erfolgreich verifiziert.

Testchip für ein Laser-Radar



Entwurf:	Hochschule Ulm, Institut für Kommunikationstechnik Bearbeiter: Georg Vallant Betreuer: Prof. Dipl.-Phys. Gerhard Forster
Layouterstellung:	Hochschule Ulm, Labor Mikroelektronik (Mixed Signal-Entwurf) Analogteil: Full Custom Design Digitalteil: Standardzellen-Entwurf
Technologie:	C35B4C3 0,35 µm CMOS 4 Metal / 2 Poly / HR
Chipfertigung:	Fa. AMS, Österreich, über Europractice
Herstelldatum:	III. Quartal 2008
Kostenträger:	MPC-Gruppe Baden-Württemberg
Chipdaten:	Chipfläche: 2,15 x 1,70 mm^2 Gehäuse: QFN 48 Funktionsblöcke: Analogteil: 5 Transimpedanzverstärker, 5 Komparatoren, 1 DPLL Digitalteil: 5 Zähler, Ausleselogik
Funktion:	Mit dem Testchip sollten kritische Komponenten eines Laserradar-Empfängers untersucht werden. Der spätere Chip soll einmal 64 Empfangskanäle zur gleichzeitigen Laufzeitbestimmung eines reflektierten Laserpulses enthalten. Jeder Kanal besteht aus einem Transimpedanzverstärker (Transimpedanz 100 kΩ, Bandbreite 380 MHz), einem Komparator (Laufzeit 1,4 ns) und einem Zähler. Das System arbeitet mit einem internen Takt von 640 MHz, der mittels Frequenzsynthese gewonnen wird. Der Testchip enthält 4 Slices sowie einen Referenzkanal mit zusätzlichen Diagnoseanschlüssen. Er konnte bereits erfolgreich getestet werden. Näheres siehe Workshopband 40, Seite 19.

MULTI PROJEKT CHIP GRUPPE

Hochschule Aalen
Prof. Dr. Bartel, (07361) 576-4182
manfred.bartel@htw-aalen.de

Hochschule Albstadt-Sigmaringen
Prof. Dr. Rieger, (07431) 579-124
rieger@hs-albsig.de

Hochschule Esslingen
Prof. Dr. Lindermeir, (0711) 397-4221
walter.lindermeir@hs-esslingen.de

Hochschule Furtwangen
Prof. Dr. Rülling, (07723) 920-2503
rue@hs-furtwangen.de

Hochschule Heilbronn
Prof. Dr. Gessler, (07940) 1306-184
gessler@hs-heilbronn.de

Hochschule Karlsruhe
Prof. Dr. Koblitz, (0721) 925-2238
rudolf.koblitz@hs-karlsruhe.de

Hochschule Konstanz
Prof. Dr. Freudenberger, (07531) 206-647
juergen.freudenberger@htwg-konstanz.de

Hochschule Mannheim
Prof. Dr. Paul, (0621) 292-6351
g.paul@hs-mannheim.de

Hochschule Offenburg
Prof. Dr. Jansen, (0781) 205-267
d.jansen@fh-offenburg.de

Hochschule Pforzheim
Prof. Dr. Kesel, (07231) 28-6567
frank.kesel@hs-pforzheim.de

Hochschule Ravensburg-Weingarten
Prof. Dr. Ludescher, (0751) 501-9685
ludescher@hs-weingarten.de

Hochschule Reutlingen
Prof. Dr. Kreutzer, (07121) 271-7059
hans.kreutzer@hochschule-reutlingen.de

Hochschule Ulm
Prof. Dipl.-Phys. Forster, (0731) 50-28180
forster@hs-ulm.de

www.mpc.belwue.de