Lehrstuhl
Kommunikationstechnik

Brandenburgische
Technische Universität
Cottbus - Senftenberg

Brandenburgische Technische Universität Cottbus-Senftenberg

Fakultät 3 - Mathematik, Naturwissenschaften, Informatik

Lehrstuhl Kommunikationstechnik

Prof. Dr.-Ing. habil. Matthias Wolff

– Bachelorarbeit –

Entwicklung eines Vorlesungsexperiments "LTI-System"

Develop of a lecture experiment "LTI-System"

Andre Jan Richter

Matrikelnummer 2906399

Informations- und Medientechnik

Abgabe: *17. Dezember 2013*

Prüfer: Prof. Dr.-Ing. habil. Matthias Wolff

Kurzfassung/Abstract

Diese Arbeit beschäftigt sich mit der Umsetzung eines LTI-Systems auf einem digitalen Signalprozessor. Anfangs werden theoretische Mittel zur Lösung des Problems vorgestellt. Danach schließt sich eine Einführung in die digitale Signalverarbeitung an, gefolgt von der Beschreibung der benutzen Soft- und Hardware. Abschließend befindet sich die Dokumentation der Entwicklung und eine Zusammenfassung.

This Thesis is composed to provide an explanation about the execution of a LTI-system to a digital signal processor. The short prelude in the beginning gets followed by a description of the topic's main problems, which get solved through the application of several adequate approaches in a theoretical manner. Subsequently an introduction in the digital signal processing is given. This part gets followed by a detailed list of the applied soft- and hardware as well as the report of development. The end presents a summary, which brings the thesis to a conclusion.

Inhaltsverzeichnis

Abbildungsverzeichnis

Abkürzungsverzeichnis

AAF	Antialaisingfehler
ADU	Analog-Digital-Umwandler
CPU	Central Processing Unit
DAU	Digital-Analog-Umsetzer
DSP	Digitaler Signalprozessor
DSV	Digitale Signalverarbeitung
FFT	Fast Fourier-Transformation
FPGA	Field Programmable Gate Array
FRAM	Ferroelectric Random Access Memeory
LTI-System	lineares zeitinvariantes System
MAC	Multiplay-Accumulate
SNR	Signal-Rauschabstand

1 Einleitung

Die Aufgabe dieser Arbeit ist es, ein System zu entwickeln, welches eingehende Signale modifiziert und ausgibt. Um diese zu bewältigen, erfolgt am Anfang der Arbeit eine Einleitung, in der theoretische und mathematische Hintergründe für die Umsetzung thematisiert werden. Dabei wird beginnend beschrieben, was Signale sind und wie sie sich klassifizieren lassen.

Befindet man sich in einer Kathedrale und klatscht in die Hände, so wird ein großräumiger Hall erzeugt. Dieser Vorgang lässt sich innerhalb eines Raumes mit allen akustischen Signalen beliebig oft wiederholen. Demnach wird auf ein Eingangssignal (das Klatschen) mit einem veränderten Ausgangssignal (das Klatschen mit Hall) geantwortet. Dabei wird das Originalsignal scheinbar modifiziert. Doch wer oder was verändert die akustischen Merkmale des Signales? Die Antwort darauf ist die Kathedrale selbst. Den eben beschriebenen Vorgang kann man auf jeden anderen Raum übertragen. Dabei klingt die Systemantwort auf eine Anregung immer dem Raum entsprechend.
Unabhängig davon, zu welchem Zeitpunkt man in einem Raum Laute erzeugt, ist der zeitliche Abstand bis zur Antwort immer konstant und unabhängig davon, welche Laute man in diesem Raum erzeugt, die Antwort wird immer als eine Abwandlung davon erkennbar sein. Man stellt fest, dass die Veränderung eines Signals innerhalb eines abgeschlossenen Systems von statten geht. Diese Beschreibung benennt man in der Kommunikationstechnik als ein LTI-System, ein lineares zeitinvariantes System. Ein solches System kann abstrakt als ein Schema beschrieben werden, welches auf ein Eingangssignal mit einem Ausgangssignal antwortet und dabei dem Prinzip der Linearität (Wiedererkennung des Eingangs) und dem Prinzip der Zeitinvarianz (zeitliche Konstanz) gerecht wird.
Dieser Effekt lässt sich elektronisch nachmodellieren. Dazu wird im Rahmen dieser Arbeit ein leistungsstarker digitaler Signalprozessor zum Einsatz kommen, auf dem versucht wird eine Software zu implementieren, die unter anderem das LTI-System *Kathedrale* simuliert. Dabei benötigt man die Impulsantwort der Kathedrale. Als Impuls wird dabei zum einfacheren Verständnis das Klatschen gesehen, was man intuitiv als kurzen kraftvollen Akustikstoß beschreiben kann. Praktisch kommt hier ein Dirac-Stoß zum Einsatz (Erklärung Kapitel 2.3). Das Eingangssignal wird mit der Impulsantwort gefaltet und man erhält dadurch die Systemantwort. Die so genannte Faltung stellt hier eine mathematische Operation dar.
Um ein LTI-System zu beschreiben, ist es von Nöten, ein Signal nicht nur im Zeitbereich zu kennen, sondern auch frequenzmäßige Eigenschaften zu ermitteln. Um vom Zeitbereich in den Frequenzbereich zu transformieren, wird die Theorie der Fourier-Transformation genutzt, welche ebenfalls im theoretischen Teil kurz erläutert wird. Danach wird ein Ein-

blick in die digitale Signalverarbeitung gegeben. Dieser beinhaltet auch den Vorgang der Digitalisierung der Signale und welche Probleme dabei auftreten können.
Eine weitere Einsatzmöglichkeit eines LTI-Systems betrifft speziell die schon angesprochenen Frequenzen eines Signals. Hält man sich beispielsweise vor Augen, wie minderwertig die Qualität der Stimme während eines Telefonates gegenüber dem Original ist, so muss man sich die Frage stellen, wieso dies so ist. Dies hat mit den Frequenzen des Signals zu tun. Das menschliche Ohr kann Frequenzen von $200\,Hz$ bis circa $20\,kHz$ wahrnehmen. Menschliche Sprache bewegt sich dabei im Bereich von $500\,Hz$ bis $2\,kHz$. Um die Übertragung der Signale eines Telefonats effizienter zu gestalten, kann man sich auf den Frequenzbereich der Sprache konzentrieren. Mit einem LTI-System ist es möglich, aus dem Originalsignal den eben angesprochenen Bereich zu Filtern. Um dies durchzuführen, werden bestimmte Frequenzfilter benötigt, die ebenfalls im Rahmen dieser Arbeit implementiert werden sollen.
Neben der Entwicklung der Software, soll der digitale Signalprozessor in eine Black-Box verbaut werden, die eine ansprechende und einfache Bedienung bietet. Die Arbeit hat außerdem den Zweck, das Endergebnis im Vorlesungsbetrieb des Moduls *Grundzüge der Kommunikationstechnik* als Demonstrator zu nutzen.
Der letzte Teil dieser Arbeit beschäftigt sich mit der Darstellung der Umsetzung und der dazu benötigten Verfahren. Weiterhin wird die benutzte Hard- und Software beschrieben. Abschließend erfolgt die Dokumentation des Quellcodes, sowie einige Tests, die mit aussagekräftigen Grafiken belegt werden.

2 Signale und Systeme

Das Wort Signal ist von den lateinischen Wörtern *signalis* (deutsch *dazu bestimmt*) und *signum* (deutsch *ein Zeichen*) abgeleitet. Es repräsentiert eine physikalische Darstellung von Informationen, die von einem Sender zu einem Empfänger, mittels einem Kanal, übertragen werden. Es stellt dabei den Informationsträger dar. Ein Signal ist eine nachweisbare und stets messbare Größe (insofern technisches Know-how und benötigte Geräte verfügbar sind). Signale können bezüglich vieler Eigenschaften klassifiziert werden. Diese sind zum Beispiel in allgemeiner Form in Nutzsignal, also ein Signal welches Informationen für den Empfänger bereit hält, und in Störsignal, also unerwünschte Überlagerungen des nützlichen Anteils, gruppiert.
Bei einem Signal wird ein Amplitudengang über eine physikalische Größe beschrieben. Diese Größe entspricht allgemein hin der Zeit. In dieser Arbeit wird sich neben Signalen mit zeitlichem Bezug auch auf ihre Transformation im betreffenden Frequenzbereich konzentriert. In der Abbildung 2.1 auf Seite 3 wird ein Signal mit einer Amplitude U über einen zeitlichen Verlauf t dargestellt. Die im Ausschnitt gezeigte Kosinusfunktion ist eine der wichtigsten Signale für das Themengebiet der digitalen Signalverarbeitung. Diese

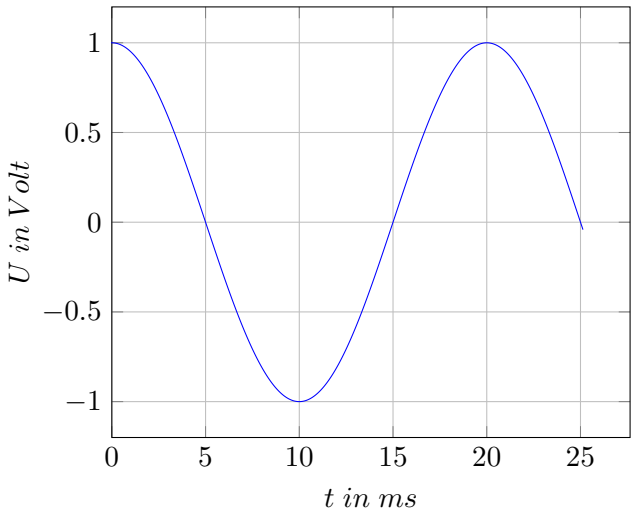


Abbildung 2.1: Analoges Signal, Spannung $U\ in\ Volt$ über Zeit $t\ in\ ms$
Eigene Darstellung

Funktion hat die Grundform $x(t) = Acos(\omega t + \theta)$. Die Größe A beschreibt dabei die Amplitude des Signals, ω repräsentiert die Frequenz und θ ist der Parameter der Phase. Das in Abbildung 2.1 auf Seite 3 gezeigte Kosinussignal hat somit die Amplitude $A = 1\,V$, die Frequenz $\omega = 2\pi 50\,Hz$ und ist in seiner Phase nicht verschoben ($\theta = 0\,°$).
Man kann den Werte- beziehungsweise Definitionsbereich auf zweierlei Weisen unterteilen.

Zum einen kann das Signal für eine kontinuierliche Zeit t definiert sein oder es ist für diskrete ganzzahlige Werte k beschrieben. Zum anderen kann auch die Amplitude kontinuierlich oder für eine begrenzte Anzahl an Werten definiert sein.
Daraus ergeben sich vier Arten von Signalen: $x(t)$, $x_Q(t)$, $x(k)$ und $x_Q(k)$. Die vier genannten Unterarten sind in Abbildung 2.2 auf Seite 5 dargestellt. Dabei wurde der Signalart $x(t)$ (zeit- und wertekontinuierliches Signal) bis heute die meiste Aufmerksamkeit geschenkt. Ein Grund dafür ist, dass vorhandene Bauelemente, mit denen Signale verarbeitet werden konnten, nur für diese Signalart geeignet war. Dieses Verhältnis hat sich, wie in Kapitel 3.1 beschrieben wird, in den letzten Jahren geändert, denn die zeitdiskreten Signale haben enorm an Bedeutung gewonnen.[enden]
Signale lassen sich, wie bereits erwähnt, nicht nur im Zeitbereich beschreiben. Für die Verwendung in der Signalverarbeitung ist es notwendig, Signale auch im Frequenzbereich darstellen und verstehen zu können. Das Signal im Frequenzbereich wird auch als Spektrum des Signals bezeichnet.
Neben der gebräuchlichen Form, kann man Signale auch in komplexer Form beschreiben. Nehmen wir das Signal $x(t) = Acos(\omega t)$, wobei $\omega = 2\pi f_0$. Diese Kosinusfunktion kann man in komplexer Schreibweise, beziehungsweise eulerschen Form, auch beschreiben als $x(t) = Acos(\omega t) + jAsin(\omega t) = Ae^{j\omega t}$. Dabei kann man sich das Signal als Zeiger eines Einheitskreises vorstellen. Die Länge des Zeigers wird durch die Amplitude A und die Winkelgeschwindigkeit, mit der der Zeiger rotiert, durch ω repräsentiert.

2.1 Analoge Signale

Ist ein Signal $x(t)$, mit $x : \mathbb{R} \Rightarrow \mathbb{R}$ und $t \in \mathbb{R}$, zeitkontinuierlich, bedeutet dies, dass ein Funktionswert x für jede Stelle t eines Zeitbereiches definiert ist. Wird es zusätzlich noch als wertkontinuierlich bestimmt, die Amplitudenänderung kann unendlich klein sein, so spricht man von einem analogen oder auch kontinuierlichen Signal. Ein Beispiel dafür ist die Sprache des Menschen.

2.2 Digitale Signale

Kann man den Funktionswert von $x(t_k)$ mathematisch als eine äquidistante (mathematisch: Punkte gleichen Abstands) und zeitliche Folge $x(k)$, mit $k \in \mathbb{N}$ beschreiben, nennt man das Signal zeitdiskret. Sozusagen ist x meist nur für bestimmte (äquidistante) Zeitpunkte definiert. Ist es dazu noch wertediskret, nennt man es ein digitales Signal ($x_Q(k)$). Ein Beispiel hierfür ist eine Rechteckspannung. Digitale Signale werden meist durch die Umwandlung von analogen Signalen gewonnen. Dies ist jedoch nicht immer der Fall. Manche digitalen Signale werden gezielt produziert. Beispielsweise liefert ein digitaler Tongenerator eine Folge von Zahlen, die zusammen ein stufiges und kosinusförmiges Signal darstellen.

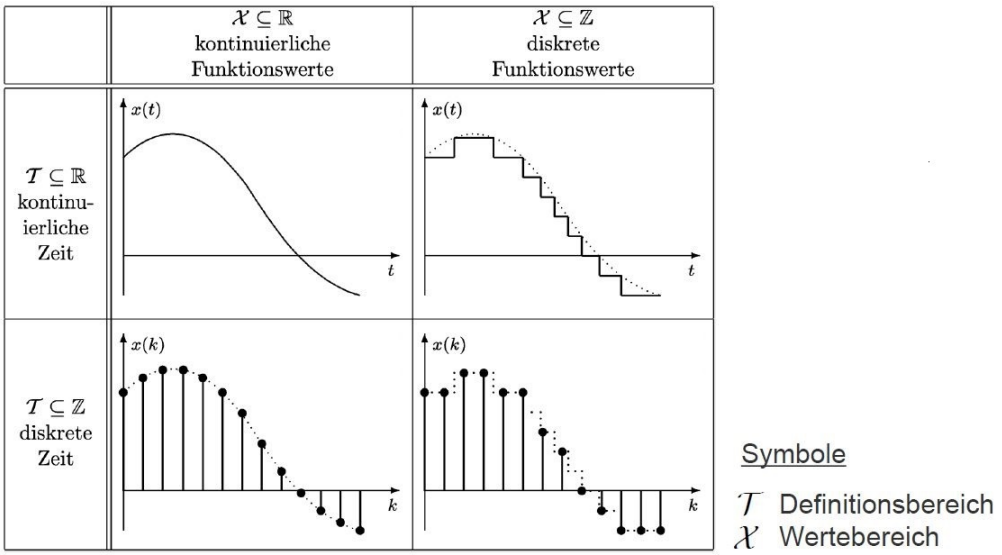


Abbildung 2.2: Übersicht über Umwandlung von Signalen
Bild:[hoffmann]

2.2.1 Festlegung

In dieser Arbeit sollen folgende Notationen verwendet werden. Wird von einem kontinuierlichem Signal gesprochen, so ist die Zeitkonstante t - zum Beispiel $x(t) = cos(x)$. Ist die Rede von den diskreten Abtastwerten eines Signals oder von einer Folge davon, so ändert sich die Zeitvariable in k und meint damit äquidistante diskrete Zeitpunkte, zum Beispiel $x(k) = cos(x)$. In Kapitel 3 wird zu Hervorhebung der Amplitudenquantisierung auch $x_Q(k)$ verwendet. Ob es sich dabei um den Wert oder um das Signals selbst handelt, wird aus dem Kontext ersichtlich sein.

2.3 Elementarsignal Dirac-Impuls

Man stelle sich ein Rechtecksignal wie in Abbildung 2.3a auf Seite 7 vor. Integriert man unter dieser Funktion, erhält man den Flächeninhalt 1. Schneidet man nun den negativen Teil ab und fügt ihn oberhalb des positiven an, erhält man ebenfalls einen Flächeninhalt von 1 (vgl. Abbildung 2.3b auf Seite 7). Dieses Verfahren kann Wort wörtlich bis auf die Spitze getrieben werden, gerade solange, bis die Amplitude unendlich groß und die Breite des Signals unendlich klein ist, die Fläche bleibt 1. Mathematisch beschrieben multipliziert man eine Rechteckfunktion mit der Höhe 1 und der Breite T_0 mit $\frac{1}{T_0}$. Für T_0 gegen unendlich entsteht dabei der Dirac-Stoß. Man nennt diese Funktion[1] auch Delta-Stoß $\delta(t)$, -Impuls, $\delta(t)$-Funktion oder Nadelimpuls mit:

$$\int_{-\infty}^{\infty} \delta(t)dt = 1. \tag{2.1}$$

Da die 1 das neutrale Glied der Multiplikation ist, nutzt man den δ-Puls oder die Multiplikation dazu, um ein Signal an einer Stelle t_0 abzutasten. Daraus kann man folgende Formel ableiten:

$$\int_{-\infty}^{\infty} x(t) \cdot \delta(t - t_0)dt = x(t_0). \tag{2.2}$$

Addiert man zum Dirac-Impuls zeitlich verschobene Abbilder, so erhält man die so genannte Abtastfunktion, welche auch als Dirac-Kamm bezeichnet wird:

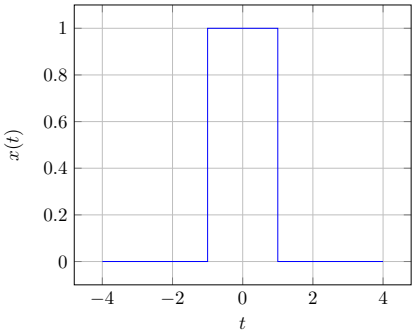
$$\delta_T(t) = \sum_{n=-\infty}^{\infty} \delta(t - nT). \tag{2.3}$$

Das Spektrum eines Dirac-Impulses erhalten wir durch die Frequenzanalyse, welche eine Fourier-Transformation darstellt. Diese wird im späteren Verlauf noch genauer beschrieben. Man kommt zu dem Ergebnis, dass der Dirac-Impuls alle vorkommenden Frequenzen besitzt, da dessen Fourier-Transformierte stets durch den Graphen $y = 1$ beschrieben werden kann.

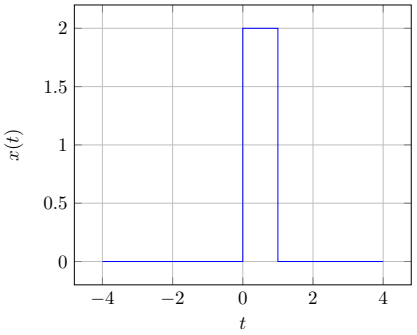
2.4 Transformation von Zeit- und Frequenzbereich

Wie bereits am Anfang des Kapitels erwähnt lassen sich Signale nicht nur zeitabhängig, sondern auch frequenzabhängig darstellen. Dafür muss das Signal in den Frequenzbereich transformiert werden. Die folgenden Abschnitte beschreiben oder nennen Vorgehensweisen um verschiedene Signalarten vom Zeitbereich in den Frequenzbereich zu überführen. Da sich diese Arbeit jedoch nicht auf die Ausarbeitung der mathematischen Herleitungen der benötigten Rechenvorschriften stützt, sollen diese nicht hergeleitet werden.

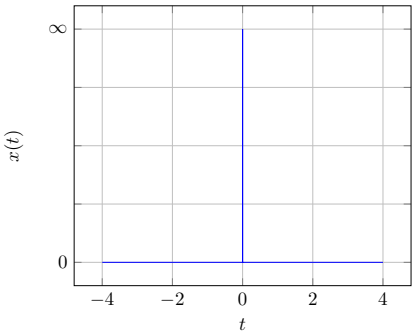
[1] In der Literatur [grueningen] wird hier mathematisch gesehen von einer Distribution gesprochen.



(a) Rechtecksignal mit Fläche=1



(b) Gestauchtes Signal mit Fläche=1



(c) Dirac-Impuls mit Fläche=1

Abbildung 2.3: Herleitung des Dirac-Impulses
Bild: eigene Darstellung

2.4.1 Fourier-Reihe und Fourier-Transformation

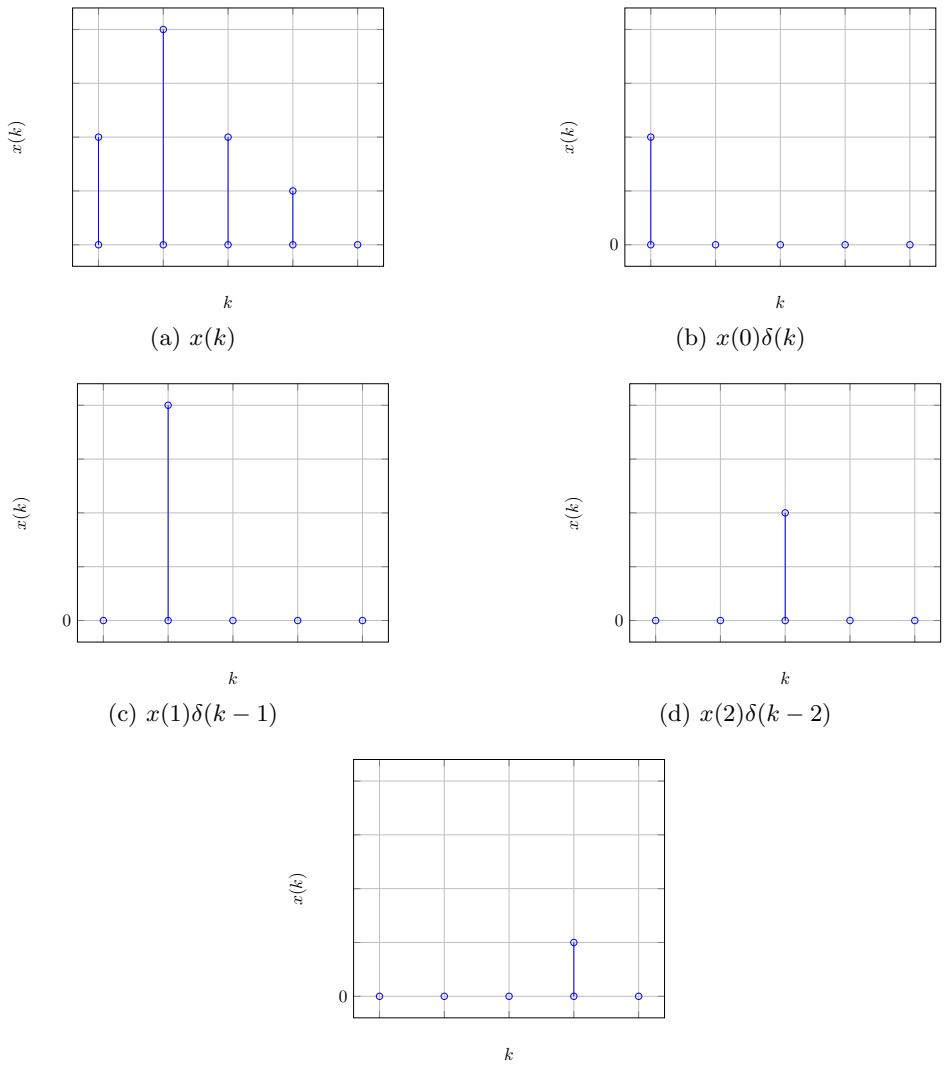
Der französische Mathematiker und Physiker Joseph Fourier fand am Anfang des 19. Jahrhunderts heraus, dass sich ein kontinuierliches periodisches Signal als Kombinationen aus Sinus- und Kosinusschwingungen mit der Formel:

$$x(t) = \frac{a_0}{2} + \sum_{k=1}^{\infty}[a_k cos(2\pi k f_0 t) + b_k sin(2\pi k f_0 t)], \tag{2.4}$$

darstellen lässt[grueningen], wobei a_k und b_k als Fourier-Koeffizienten bezeichnet werden. Oder zusammengefasst:

$$x(t) = A_0 + \sum_{k=1}^{\infty} A_k cos(2\pi k f_0 t + \alpha_k). \tag{2.5}$$

Diese Summe heißt Fourier-Reihe. Die Grundfrequenz f_0 setzt sich aus der Periode T des Signals zusammen ($f_0 = \frac{1}{T}$). Der Signalanteil mit dieser speziellen Frequenz wird Grundschwingung genannt. Alle anderen Anteile sind Harmonische dieses Signals. Eine Harmonische ist eine Schwingung, die sich als ganzzahlige Vielfache einer Grundschwingung darstellen lässt. Man nennt diese auch Obertöne der Grundschwingung. A_0 bezeichnet den Gleichstromanteil. Es gibt allerdings noch eine weitere Form, mit der sich eine



(a) $x(k)$ (b) $x(0)\delta(k)$

(c) $x(1)\delta(k-1)$ (d) $x(2)\delta(k-2)$

(e) $x(3)\delta(k-3)$

Abbildung 2.4: Zerlegung des Signals $x(k)$ in verschobene Dirac-Impulse
Bild: eigene Darstellung

Fourier-Reihe darstellen lässt:

$$x(t) = \sum_{k=-\infty}^{\infty} c_k e^{jk2\pi f_0 t}. \tag{2.6}$$

Dieser Ausdruck wird komplexe Fourier-Reihe genannt und c_k heißt komplexer Fourier-Koeffizient. Die verschiedenen Koeffizienten lassen sich ineinander umrechnen. [grueningen] Da meist die komplexe Variante (komplexe Fourier-Reihe) bevorzugt wird, soll nur die Formel zur Berechnung des komplexen Koeffizienten aufgezeigt werden:

$$c_k = \frac{1}{T_0} \int_{\frac{-T_0}{2}}^{\frac{T_0}{2}} x(t) e^{-jk2\pi f_0 t} dt. \tag{2.7}$$

Die eben aufgezeigten Formeln bezogen sich ausschließlich auf periodische Signale. Hat man es jedoch mit aperiodischen Schwingungen zu tun, braucht man die Fourier-Transformierte (FT) des Signals:

$$X(f) = \int_{-\infty}^{\infty} x(t)e^{-j2\pi ft}dt. \tag{2.8}$$

Kongruent dazu ist die inverse Fourier-Transformation, um vom Frequenzbereich wieder in den Zeitbereich zu überführen:

$$x(t) = \int_{-\infty}^{\infty} X(f)e^{j2\pi ft}df. \tag{2.9}$$

Durch die Fourier-Transformation eines Signals $x(t)$ erhält man das Spektrum $X(f)$ und durch die inverse Fourier-Transformation von $X(f)$ erhält man $x(t)$, sodass man diese Beziehung als Transformationspaar bezeichnen kann.

$$x(t) \circ\!\!-\!\!\bullet X(f) \tag{2.10}$$

Das Spektrum $X(f)$ eines Signals $x(t)$ repräsentiert das Betrags- und Phasenspektrum einer komplexen Schwingung, mit der Frequenz f.
Die Fourier-Transformation besitzt eine Reihe von Eigenschaften, die den Gebrauch unter Umständen vereinfachen können. Für ein besseres Verständnis der FT werden die Veröffentlichungen [grueningen] und [pearson] nahegelegt.
Abschließend soll noch ein Beispiel der Fourier-Transformation gegeben werden. Dabei handelt es sich um die Transformierte eines Rechteckimpulses. Man nehme an, dass ein Rechtecksignal (Abbildung 2.5 auf Seite 10) der Breite τ definiert sei als:

$$x(t) = \begin{cases} x_0 & : -\frac{\tau}{2} < t < \frac{\tau}{2} \\ 0 & : sonst. \end{cases} \tag{2.11}$$

Gesucht ist das Spektrum des Signals. Es berechnet sich wie folgt:

$$X(\omega) = x_0 \int_{-\frac{\tau}{2}}^{\frac{\tau}{2}} e^{-j\omega t}dt \tag{2.12}$$

$$= x_0\tau \frac{e^{j\omega\frac{\tau}{2}} - e^{-j\omega\frac{\tau}{2}}}{j\omega\tau} \tag{2.13}$$

$$= x_0\tau sinc\frac{\omega\tau}{2}. \tag{2.14}$$

Zeichnet man dieses (Betrags-)Spektrum, so ist der Verlauf wie in Abbildung 2.6 auf Seite 10. Es ist zu sehen, dass der Verlauf eine so genannte *sinc*-Funktion darstellt, welches einen abklingenden Charakter hat, jedoch nie komplett ausschwingt. Ein damit zusammenhängendes Problem wird zu einem späteren Zeitpunkt erläutert - denn das Spektrum muss *gefenstert* werden.

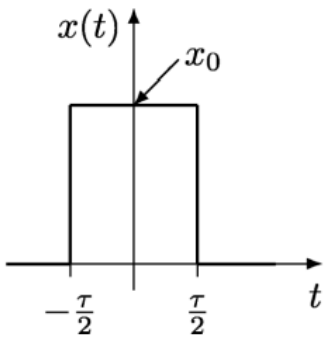


Abbildung 2.5: Rechtecksignal
Bild: [wolff]

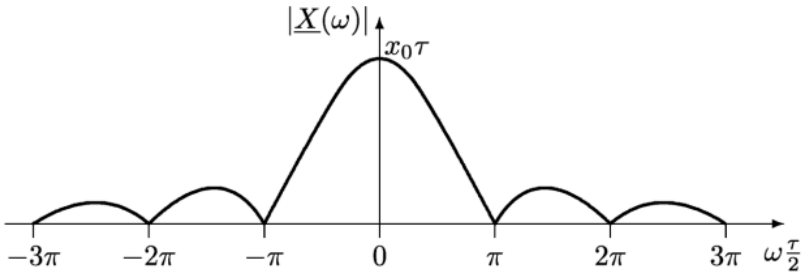


Abbildung 2.6: *sinc*-Funktion
Bild: [wolff]

2.4.2 Zeitdiskrete Transformationen

Da zur Verarbeitung im DSP abgetastete Signale vorliegen, soll beschrieben werden, wie sich diese Signale frequenzabhängig darstellen lassen. Bekannter Weise lassen sich kontinuierliche Signale mittels der Fourier-Reihe oder -Transformation berechnen. Setzt man ein abgetastetes Signal (mittels der Abtastfunktion von Formel 2.3 auf Seite 6 multipliziert mit dem Signal $x(t)$) in das Fourier-Integral ein und fasst es zusammen, so erhält man für das abgetastete Signal $x_s(t)$:

$$x_s(t) = \sum_{n=-\infty}^{\infty} x(nT)\delta(t - nT), \tag{2.15}$$

das Spektrum $X_s(f)$ mit der Rechenvorschrift

$$X_s(f) = \sum_{n=-\infty}^{\infty} x(nT)e^{-j2\pi fnT}, \tag{2.16}$$

unter der Annahme, dass der Dirac-Kamm nur bei $t = nT$ einen Wert, nämlich 1, besitzt und sonst immer 0 ist.[grueningen]

2.4.3 Zeitdiskrete Fourier-Transformation (DFT)

Da im praktischen Teil dieser Arbeit ausschließlich mit digitalen Signalen gearbeitet wird, muss die Transformation im Frequenzbereich für diese Klasse von Signalen beschrieben werden. Um das gewünschte Ergebnis zu erzielen, wird für periodische zeitdiskrete Signale die diskrete Fourier-Transformation, kurz DFT, angewendet. Sie spielt eine entscheidende Rolle in der digitalen Signalverarbeitung, da sich mit ihr nicht nur das Signal im Bildbereich darstellen lässt, sondern auch wichtige Operationen, wie beispielsweise die Faltung, vereinfachen.[2] Sie lässt sich weiterhin mit verschiedenen Algorithmen in ein Computerprogramm implementieren.
Die Aufgabe dieser Bachelorarbeit besteht darin, ein Echtzeitsystem zu entwickeln, mit dem man Sprachsignale modifizieren kann. Sprache verkörpert aperiodische Signale, da sie ein Gemisch von verschiedenen Lauten mit unterschiedlicher Länge und unterschiedlichen Frequenzen darstellen. Solche Signale haben eine unendliche lange Periodendauer. Da die DFT jedoch zur Frequenzanalyse von periodischen Signalen verwendet wird, muss das Signal gefenstert werden. Fenstern bedeutet, dass nur ein Teil des Signals betrachtet, beziehungsweise ausgeschnitten wird. Die Fensterlänge wird mit dem Parameter N angegeben und gibt die Anzahl an Abtastwerten an, die pro Fenster verwendet werden. Man geht davon aus, dass das gefensterte Signal N-periodisch ist.
Betrachtet man die Formel der FT (Formel 2.8 auf Seite 9), setzt dort das abgetastete Signal (Formel 2.15 auf Seite 10) ein, legt fest, dass man die diskreten Abtastwerte mit $x(k)$ benennt und stellt geeignet um, so erhält man für die DFT eines Signals die Formel:

$$X(f_k) = \sum_{k=0}^{N-1} x(k) e^{-jkf_k \frac{2\pi}{N}}, \tag{2.17}$$

wobei k die diskrete Zeitvariable und analog dazu, f_k die diskrete Frequenzvariable darstellt. Die inverse DFT ist:

$$x(k) = \frac{1}{N} \sum_{f_k=0}^{N-1} X(f_k) e^{jf_k k \frac{2\pi}{N}}. \tag{2.18}$$

Dieses Verfahren ist deshalb so wichtig, weil man damit digitale Filter realisieren kann, denn eine Faltung (digitales Filter) im Zeitbereich, entspricht einer einfachen Multiplikation im Frequenzbereich. Diese Aussage wird auch als Faltungstheorem bezeichnet.
Die DFT selbst hat einen Rechenaufwand von $N \cdot N$ Multiplikationen und $N \cdot (N-1)$ Additionen. Da digitale Signalprozessoren für diese beiden Operationen optimiert sind, betrachtet man die Multiplikationen inklusive Additionen und man erhält einen Aufwand von $O(N^2)$. Zusammengefasst liefert die DFT für ein zeitdiskretes Signal $x(k)$ ein frequenzdiskretes Spektrum $X(f_k)$.[werner] Die DFT kann mittels einer schnellen Fourier-Transformation implementiert.

[2] Unter Verwendung des Faltungstheorems.

2.4.4 Schnelle Fourier-Transformation

Die schnelle Fourier-Transformation (engl. Fast Fourier-Transformation (FFT)) ist ein Verfahren, um die DFT effizienter durchzuführen. Der entscheidende Unterschied liegt beim Rechenaufwand, der bei einer Implementierung in ein Computerprogramm ein kalkuliert werden muss und erheblichen Einfluss auf die Entwicklung eines Programms hat. Es bedarf allerdings einigen Grundvoraussetzungen, die erfüllt sein müssen, um eine FFT durchführen zu können. Die Hauptidee dieses Algorithmus ist es, das Problem in Teilprobleme zu zerlegen.[3]
Begonnen wird mit der Zerteilung des Eingangsignals in zwei Teilfolgen. Die eine enthält nur die geraden Abtastwerte ($x_1(k) = x(2k)$) und die andere folglich die Ungeraden ($x_2(k) = x(2k+1)$). So kann man auch die Summe der DFT (Formel 2.17 auf Seite 11) in zwei Teilsummen zerlegen. Diese Zerlegung führt man oft solange durch, bis die Rechnung nur noch aus so genannten 2-Punkte-DFTs besteht. Um diese Zerlegung bis zum Schluss durchführen zu können, ist es also notwendig die Fensterlänge N so zu wählen, dass sie den Wert einer Zweierpotenz annimmt. Ist dies nicht möglich, kann mittels *zero padding* das Fenster mit so vielen Nullen aufgefüllt werden, bis N eine solche Potenz darstellt. Der Rechenaufwand der gesamten FFT ist $O(N \cdot log_2(N))$, welcher sich im Gegensatz zu $O(N^2)$ erheblich verringert hat. Grafisch kann man die Zerlegungen in so genannten Butterfly-Graphen veranschaulichen. Weitere Informationen zu Herleitung, Rechenwegen und Beispielen findet man in [grueningen] und [koblf].

2.4.5 z-Transformation

Die Fourier-Transformierte eines zeitdiskreten Signals wurde schon in Gleichung 2.16 auf Seite 10 aufgezeigt. Da $x(k)$ die Abtastwerte des Signals darstellen, kann Gleichung 2.16 als:

$$X(f) = \sum_{k=-\infty}^{\infty} x(k) e^{-j2\pi fkT} \tag{2.19}$$

dargestellt werden. Ersetzt man die Frequenzvariable durch die normierte Kreisfrequenz $\Omega = 2\pi fT$, so erhalten wir bekanntermaßen ebenfalls die Fourier-Transformierte eines zeitdiskreten Signals. Allerdings gibt es Signale, bei denen die Summe dort nicht konvergiert, also $X(\Omega) = \infty$. Für diese Fälle multipliziert man die Signale mit der Exponentialfolge r^{-k}. Damit kann erreicht werden, dass die Summation doch konvergiert und man zielführende Ergebnisse für die Fourier-Transformierte bekommt. Die FT hat dann die Form:

$$X(\Omega) = \sum_{k=-\infty}^{\infty} x(k) e^{-jk\Omega} r^{-k}. \tag{2.20}$$

Die beiden letzten Faktoren fasst man zu einer komplexen Variable z zusammen und erhält somit die z-Transformierte:

$$X(z) = \sum_{k=-\infty}^{\infty} x(k) z^{-k} \tag{2.21}$$

[3] Man spricht hier auch vom Prinzip *devide and conquer*, zu deutsch *teile und herrsche*.

und durch die Anwendung von dem Cauchy Integral Theorem die inverse Rechenvorschrift [grueningen]:

$$x(k) = \frac{1}{2\pi j} \oint X(z) z^{k-1} dz. \tag{2.22}$$

2.5 LTI-System

Mit einem linearen, zeitinvarianten System (LTI-System) wird ein Ordnungsprinzip beschrieben, welches Eingangs- in Ausgangsgrößen überführt und dabei Linearität und Zeitinvarianz gerecht wird. Dabei ist die Linearität und die Zeitinvarianz oder viel mehr deren Bedingungen definiert als [kobll]:

$$x(t) = k_1 x_1(t) + k_2 x_2(t) \Rightarrow y(t) = k_1 y_1(t) + k_2 y_2(t) \tag{2.23}$$

und

$$x(t - i) \Rightarrow y(t - i). \tag{2.24}$$

Speist man ein solches System mit dem Dirac-Impuls von Abschnitt 2.3 auf Seite 6, so erhält man die so genannte Impulsantwort $h(t)$ des Systems (Abbildung 2.7 auf Seite 13). Mit ihr lässt sich das gesamte Systemverhalten beschreiben, sodass man resultieren kann, dass ein angelegtes Eingangssignal $x(t)$ verknüpft mit der Impulsantwort $h(t)$, das Ausgangssignal $y(t)$ repräsentiert. Analog dazu sind diskrete LTI-Systeme beschrieben. Sie müssen die gleichen Bedingungen bezüglich Linearität und Zeitinvarianz genügen, mit dem Unterschied, dass die Zeitkonstante diskret ist.

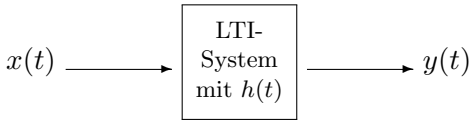


Abbildung 2.7: Analoge Signalverarbeitungsprozess
Bild: eigene Darstellung, Anlehung an [grueningen]

2.5.1 Kenngrößen und Eigenschaften eines LTI-Systems

Lineare zeitinvariante Systeme besitzen neben der eben beschriebenen Impulsantwort, weitere Möglichkeiten um sie zu beschreiben. Mit unter ist die Rede von Frequenz-, Amplituden- und Phasengang eines Systems. Der Frequenzgang ist die Fourier-Transformierte der Impulsantwort. Durch die Speisung eines LTI-Systems mit einer Kosinusschwinung und der nachzulesenden Umformung in [kobll], lässt sich ableiten, dass man aus dem Frequenzgang $H(\Omega)$ den Amplitudengang $|H(\Omega)|$ eines Systems und den Phasengang $\angle H(\Omega)$ erhalten kann. Da $\Omega = 2\pi f T$, ist der Frequenzgang und somit auch Amplituden- und Phasengang am Ausgang, von der Frequenz am Eingang abhängig. Für Frequenzen, welche ein vielfaches von 2π sind, ergeben sich gleiche Kenngrößen. In der Praxis wird der Amplitudengang oftmals als Dämpfung benutzt. Diese lässt sich wie folgt

in die Pseudoeinheit Dezibel umrechnen[kobll]:

$$A(\Omega) = -20log(|H(\Omega)|)\, dB. \tag{2.25}$$

Aus dem Phasengang lässt sich weiterhin die Information der Phasenlaufzeit τ_p erschließen, welche die benötigte Zeit darstellt, die ein Signal im Filter verbringt. Sie lässt sich berechnen mit:

$$\tau_p = -\frac{\angle H(\Omega)}{\Omega}. \tag{2.26}$$

Im Gegenzug zum Frequenzgang, gibt es die Übertragungsfunktion $H(z)$ eines diskreten LTI-Systems. Sie wird durch die z-Transformation gewonnen. Daher ist sie die z-Transformierte der Impulsantwort eines diskreten LTI-Systems. Für kontinuierliche Systeme gilt analog dazu die Laplace-Transformierte der Impulsantwort $H(s)$[grueningen], welche auch Übertragungsfunktion genannt wird. Aus dieser können die Filterkoeffizienten gewonnen werden. Sie ist nicht gleichzusetzen mit dem Frequenzgang eines LTI-Systems.[pearson]
Auch Systeme lassen sich in verschiedene Klassen unterteilen. Dabei stellen kausale Systeme eine wichtige Klasse dar, da ein System sonst als hellseherisch angesehen werden kann. Ein zeitdiskretes kausales LTI-System, welches durch eine Differenzengleichung der Form:

$$y(k) = -\sum_{i=1}^{M} a_i y(k-i) + \sum_{i=0}^{N} b_i x(k-i) \tag{2.27}$$

beschrieben werden kann, stellt die wichtigste Klasse dieser Systeme dar. N und M repräsentieren dabei die Anzahlen der Summanden und die größere von beiden definiert die Ordnung der Gleichung. Da a_i mit dem Ausgangswerten verknüpft ist, wird dieser auch rekursiver Koeffizient genannt, b_i ist der Nichtrekursive. Ist der rekursive Anteil null (alle $a_i = 0$), spricht man von nicht rekursiven Systemen, andernfalls von rekursiven Systemen. Überführt man die Differenzengleichung in den z-Bereich und löst nach $y(z)$ auf [grueningen], so erhält man für $H(z)$ (Übertragungsfunktion) eine Division mit dem nichtrekursiven Anteil im Zähler und dem rekursiven Anteil im Nenner, in der Form:

$$H(z) = \frac{b_0 + b_1 z^{-1} + \cdots + b_N z^{-N}}{1 + a_1 z^{-1} + \cdots + a_M z^{-M}}^{4}. \tag{2.28}$$

Die Differenzengleichung kann also aus der Übertragungsfunktion $H(z)$ gewonnen werden.

2.6 Faltung

In einem vorherigen Abschnitt ist von einer Verknüpfung vom Eingangssignal $x(t)$ mit der Impulsantwort $h(t)$ die Rede. Diese Verknüpfung wird Faltung genannt und besitzt die

[4]Hier wird ersichtlich, dass bei allen $a_i = 0$, sich die Filterkoeffizienten b_i aus der Übertragungsfunktion ablesen lassen.

Rechenvorschrift:

$$x(t) * h(t) = \int_{-\infty}^{\infty} x(\tau)h(t-\tau)d\tau = y(t). \tag{2.29}$$

Diese oben genannte Formel heißt Faltungsintegral und ist eine essenzielle Beziehung in der DSV. Intuitiv kann man ausdrücken, dass ein Signal mit einer Impulsantwort gefaltet wird. Das Operationszeichen lautet $*$. Es gilt also die Beziehung:

$$y(t) = \{x * h\}(t)^5. \tag{2.30}$$

Es gilt außerdem das Kommutativgesetz, welches erlaubt, Terme und Operation untereinander zu vertauschen.
Durch einige Überlegungen, die beispielsweise in [grueningen] dargestellt werden, kommt man zu der wichtigen Erkenntnis, dass man die Faltung in den Bildbereich transformieren (mittels Fourier-Transformation) kann und folgende Aussage erhält:

$$Y(f) = H(f)X(f). \tag{2.31}$$

Die Faltung eines Eingangssignals $x(t)$ mit einer Impulsantwort $h(t)$ im Zeitbereich ist identisch und der Multiplikation des Spektrums des Eingangssignals $X(f)$ mit der Fourier-Transformierten Impulsantwort $H(f)$ des Systems. Die soeben in Worten beschriebene Beziehung:

$$h(t) * x(t) \circ\!\!-\!\!\bullet H(f)X(f), \tag{2.32}$$

unterliegt dem Prinzip der Dualität, sodass ebenfalls gilt:

$$h(t)x(t) \circ\!\!-\!\!\bullet H(f) * X(f). \tag{2.33}$$

Diese Transformationspaare ergeben das Faltungstheorem, welches zum besseren Verständnis noch einmal in Worten wiedergegeben werden soll:

> *"Eine Faltung im Zeitbereich entspricht einer Multiplikation im Frequenzbereich und eine Multiplikation im Zeitbereich entspricht einer Faltung im Frequenzbereich."([grueningen], S. 46)*

Zusammengefasst bedeutet dies, dass man das aufwendige Integral der Faltung im Zeitbereich durch eine durchaus kostengünstigere Multiplikation im Frequenzbereich realisieren kann. Diesen Vorteil nutzt man vor allem bei der Umsetzung von digitalen Filtern.
Da man es in der DSV ausschließlich mit zeitdiskreten Signalen zu tun hat, wendet man eine spezielle Form der Faltung an, die diskrete Faltung. Aus dem Integral der allgemeinen

[5] Die Faltungsgleichung ist auch bekannt als $y(t) = x(t) * h(t)$, jedoch soll kenntlich gemacht werden, dass x und h an der gleichen Stelle t gemeint sind.

Faltung wird eine Summe, sodass gilt:

$$x(k) * h(k) = \sum_{n=-\infty}^{\infty} x(n)h(k-n) = y(k). \tag{2.34}$$

Dies ist auch die Formel, die zu implementieren ist, um eine Faltung im Zeitbereich zu realisieren. Ein grafisches Beispiel einer Faltung wird in Abbildung 2.8 auf Seite 16 dargestellt.

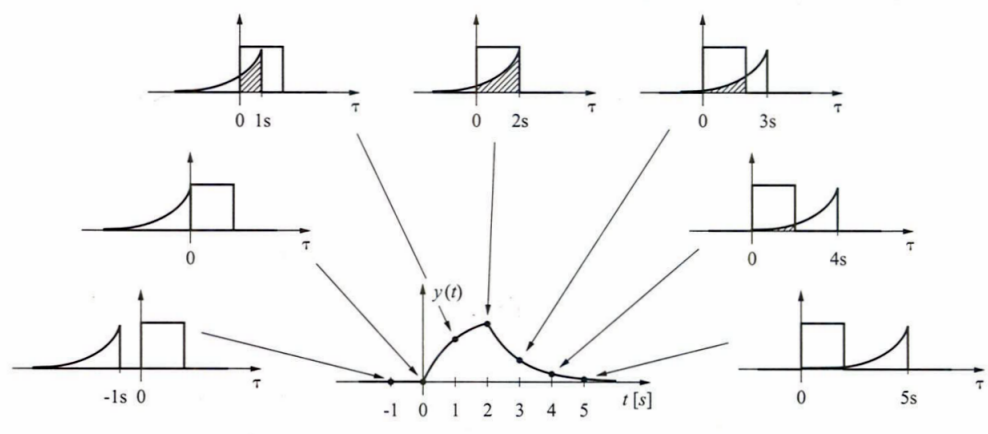


Abbildung 2.8: Faltungsprozess
Bild: [grueningen]

3 Signalverarbeitung

3.1 Geschichte der Signalverarbeitung

Da technisch nutzbarer Strom erst Mitte des 19. Jahrhunderts entwickelt wurde, ist der Ursprung der signalverarbeitenden Geschichte nicht in all zu weiter Ferne. Anfangs beschäftigte sich die Wissenschaft der Signalverarbeitung ausschließlich mit der analogen Materie. Ein Meilenstein der Entwicklung war die Übertragung von Signalen durch kabelgebundene Telegrafie, um 1800.[mfk]
Bereits 1928 publizierte der schwedische Physiker H. Nyquist eine Arbeit, aus der die Idee des Abtasttheorems hervorging. Diese Sentenz wurde 1948 von C.E. Shannon in seinem Lebenswerk *A Mathematical Theory of Communication* (deutsch *Mathematische Grundlagen in der Informationstheorie*) wissenschaftlich beschrieben.[bell] Seine dortigen Behauptungen besagen, dass ein analoges Signal verlustfrei digitalisiert (Übergang von zeit- und wertkontinuierlichen Signal zu zeit- und wertdiskreten Signal) werden kann, wenn:

$$f_{Abtast} > 2f_{max}. \tag{3.1}$$

Die Formel 3.1 auf Seite 17 beschreibt, dass die Abtastfrequenz f_{Abtast}, mit der ein analoges Signal abgetastet wird, mindestens doppelt so groß sein muss, wie die maximale Frequenz f_{max} des Signals.
Die anfänglichen Versuche mit plenardiskreten Signalen hatten noch keinen Echtzeitcharakter. Dem entgegenwirkend (jedoch nicht speziell dafür konzipiert) wurde unter anderem die Harvard-Architektur entwickelt. Diese machte es möglich, Befehlssätze von Programmdaten auf verschiedene Busse und Speicher zu trennen. Damit konnte die Rechenfähigkeit mit digitalen Signalen und die Datendurchsätze der Rechner stark erhöht werden.
In Abbildung 3.1 auf Seite 18 kann man den Prozess der Entwicklung von Rechnern betrachten. Man erkennt, dass sich der Vorgang in den letzten 20 Jahren immens beschleunigt hat. Dies hat eine Reihe von Gründen, wovon zwei genannt werden. Zum einen sind die schon bestehenden Rechnersysteme so rechenstark geworden, dass Simulationen im großen Maß (Schaltung mit über $7\,Mrd.$ Transistoren [nvidia]) bewältigt werden können. Zum anderen stellen Fertigungen in Atomgrößen und in vielfältigen Stückzahlen keine Barriere mehr dar. Leider ist das Ende der aktuellen Technologie absehbar. Zur Zeit wird an neuen Techniken geforscht. Dazu zählen beispielsweise FRAM (Ferroelectric Random Access Memeory), SET (Single Electron Transistor) oder die Forschung an Galiumnitrid.
Der heute zur Zeit leistungsfähigste Rechner der Welt ist der *Titan* der Firma *Cray* und steht im US-Energieministerium in Oak Ridge (Tennessee). Er hat eine Rechenleistung von mehr als 20 Petaflops pro Sekunde, welche er mit Hilfe der mehr als 18.500 verbauten

Technik	Charakteritisches Element	Zeitspanne	Anzahl der Addiotionen pro Sekunde
Mechanik	Zahnrad	1650-1900	1
Elektromechanik	Relais, Schalter	1900-1945	10
Elektronik	Elektronenröhre	1945-1955	10^4
Halbleiterelektronik	Transistor	1955-1965	10^5
Mikroelektronik	Integrierter Schaltkreis	1965-1975	10^6
VLSI Mikroelektronik	Mikroprozessor	1975-heute	10^{15} und mehr

Tabelle 3.1: Übersicht über den Stand der Technik
Eigene Darstellung, Anlehnung an [enden], S. 2

Grafikkarten und mehreren 6-Kern-AMD-Prozessoren erreicht. [diewelt]

3.2 Digitale Signalverarbeitung

Da sich diese Arbeit mit digitaler Signalverarbeitung (DSV) beschäftigt, wird diese hier näher erläutert.
Die DSV beschreibt auschließlich die Bearbeitung digitaler Signale. Dabei wird eine Eingangsfolge von Zahlen (digitales Signal) zu einer Ausgangsfolge verarbeitet (siehe Abbildung 3.1 auf Seite 18). Die Instruktionen der Verarbeitung, sprich die Anleitung, nach dem die Ausgangsfolge berechnet wird, nennt sich Algorithmus und ist als Programm auf dem Digitalrechner gespeichert.[grueningen] Eine viel verwendete Operation, die auf die

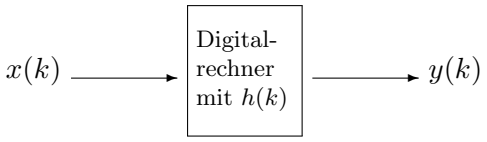


Abbildung 3.1: Digitaler Signalverarbeitungsprozess
eigene Darstellung, Anlehung an: [grueningen]

Daten bei DSV angewendet wird, ist eine Multiplikation zweier Parameter und das Aufaddieren zu dem Ergebnis. Diese Operationen nennt man im Verbund Multiply-Accumulate (deutsch *Multiplikationsakkumulator*, kurz MAC) (siehe Formel 3.2).

$$a = a + b \cdot c \tag{3.2}$$

Man benötigt sie unter anderem für die schnelle Fourier-Transformation (FFT) oder die Faltung. Dadurch ist man bestrebt, eine MAC-Operation in einem Taktzyklus durchzuführen. Gern werden spezialisierte Digitalrechner, digitale Signalprozessoren (DSP), an der Anzahl dieser Operation pro Sekunde gemessen.
Außerdem sind in den meisten DSPs noch andere Rechenwerke, wie ALUs (*arithmetic*

logic unit, deutsch *arithmetisch-logische Einheit*) oder Ähnliches zu finden. Um DSV noch effizienter zu gestalten, werden die Rechner in der sogenannten Harvard-Architektur gefertigt.
Alle DSPs kann man in Assembler[1] programmieren. Entwickelt man jedoch ein komplexes Programm, welches den DSP mit voller Rechenleistung nutzt, so sollten verschiedene Operationen möglichst parallel ablaufen, was in Assembler aufwändig werden kann. Oft wird deshalb mit einem C-Compiler gearbeitet, der es erlaubt Quellcode in der Sprache C zu formulieren, was wesentlich komfortabler ist.
Es lassen sich auch analoge Signale bearbeiten, indem sie vor der DSV entsprechend vorbehandelt und digitalisiert werden. Nach der Verarbeitung wird das Signal analogisiert und entsprechend nachbehandelt (Abbildung 3.2 Seite 19). Der erste Bearbeitungsschritt besteht im Wesentlichen aus Verstärken, Filtern und in ihrer Amplitude Begrenzen. Wie die Umwandlung und deren Rückumwandlung zwischen analog und digital im Detail funktioniert, wird später beschrieben. Die Einheiten, also den Analog-Digital- beziehungsweise Digital-Analog-Wandler, nennt man zusammen auch Codec (Zusammengesetzt aus CODieren und DECodieren).[grueningen]
Aus praktischen Gründen kann oft auf die Rücktransformation zu einem analogen Signal verzichten werden, da die Speicherung oder der Datentransfer dadurch wirtschaftlicher wird.[enden]

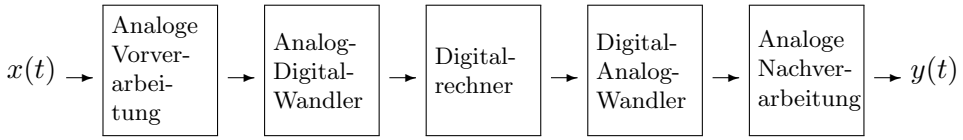


Abbildung 3.2: Signalverarbeitungsprozess
eigene Darstellung, Anlehung an: [grueningen]

3.2.1 Vorteile digitaler Signalverarbeitung

Einer der Vorteile der DSV besteht in der Langzeit- und Temperaturstabilität. Digitale Schaltungen sind auch bei längerem Betrieb in ihren Ergebnissen stabil. Ein Grund dafür ist, dass die Bauteile nicht überschwänglich auf Wärme reagieren. Bei einer analogen Schaltung sehe dies anders aus, denn ein Widerstand verändert seinen Wirkung ablesbar mit variierender Betriebstemperatur. Für einen Kupferdraht mit einem Eigenwiderstand von 1Ω verändert sich bei dem Übergang der Temperatur von 20°C auf 21°C der Widerstandswert zwar nur um $\sim 0,005\,\Omega$, jedoch sind das bei einer Veränderung von 100°C (beispielsweise durch lange Laufzeiten von Wärme entwickelnden Bauteilen) schon $\sim 0,5\Omega$.
Einen vorteilhaften Effekt hat auch der Adaptionscharakter der digitalen Filter.[grueningen] Das bedeutet, dass sich die Filter selbst anpassen und dementsprechend ihre Koeffizienten einstellen können.

[1] Assembler meint hier die maschinennahe Programmiersprache und nicht den Assemblierer, der den Programmcode in Maschinencode umwandelt.

Auch die Fähigkeit der Datenkompressionen wirkt sich positiv auf die Position der DSV gegenüber der analogen Datenverarbeitung aus. So können Daten beziehungsweise Informationen dauerhaft und vor allem effizienter gespeichert werden.
Überlegen ist die Verarbeitung in digitalen Systemen weiterhin, weil mehrdimensionale Signale, wie etwa Bilder, verarbeitet werden können. Neben der mehrdimensionalen Verarbeitung können auch Datenfelder behandelt werden. Dies nutzt man beispielsweise zur Lokalisierung einer Schallquelle, mit Hilfe von mehreren Mikrofonen, die sich alle in einem Datenfeld befinden.[grueningen]
Ein weiterer Vorteil ist die Programmierbarkeit der digitalen Signalprozessoren. Somit kann ein einziges System unzählig viele verschiedene Aufgaben ausführen, je nachdem welche Algorithmen beziehungsweise welche Programme oder Schnittstellen gewählt werden.
Ein weiteres Argument, welches für die Nutzung von DSV spricht, ist die kostengünstige Fertigung der Komponenten, Bausteine und Systeme.

3.2.2 Nachteile digitaler Signalverarbeitung

Jede neue Technologie birgt natürlich auch ihre Nachteile. Obwohl sich auch analoge Signale mittels DSV behandeln lassen, erfordert dies doch einen zusätzlichen Mehraufwand. Neben benötigtem Raum für zusätzliche Bauteile, wie Multiplexer oder RAM (*random-access memory*, deutsch *Direktzugriffsspeicher*), muss auch der dadurch verursachte erhöhte Stromverbrauch und andere Kosten einkalkuliert werden.
Des Weiteren muss man sich vor Augen halten, in welch enormer Geschwindigkeit das Umschalten von Spannungen und Strömen vor sich geht. Diese Vorgänge können dafür sorgen, dass es zu kleinen Störungen kommt.
Außerdem erfordert die Arbeit mit DSV ein enormes Fachwissen. Dabei werden neben mathematischen Kompetenzen auch Kenntnisse in der Programmierung, der Elektrotechnik und spezielles Wissen über Signalprozessoren benötigt.

3.3 Umwandlung eines Signals

Ziel dieses Unterkapitels ist es, ein wert- und zeitkontinuierliches Signal in ein wert- und zeitdiskretes Signal zu transformieren - siehe Abbildung 3.2 auf Seite 19.
In den folgenden Unterpunkten wird eine sequentielle Beschreibung des Signals nach Abbildung 3.2 auf Seite 19 gegeben. Beginnend wird der Antialaisingfilter (AAF) beschrieben, der Informationsverlust vermeiden soll und einen Punkt der Vorverarbeitung darstellt. Im nächsten Schritt, wird das Signal digitalisiert, um nach der Verarbeitung im DSP, analogisiert und nachbehandelt zu werden.

3.3.1 Anti-Alaising-Filter

Bevor das Eingangsignal den Analog-Digital-Wandler erreicht, wird er durch einen AAF geleitet. Dort wird das Signal tiefpassgefiltert und somit bandbegrenzt. Das bedeutet, dass zu hohe Frequenzen gefiltert werden und das Abtasttheorem von Formel 3.1 auf

Seite 17 anspruchsloser zu erfüllen ist. Aliasingfehler sind irreversibel, sie sind also nicht korrigierbar. Deswegen ist eine Vermeidung unbedingt notwendig.
In dieser Arbeit liegt die Abtastfrequenz bei $16\,kHz$, da das fertige Modul Sprachsignale verarbeiten soll. Da sich die menschliche Sprache im Frequenzbereich zwischen $500\,Hz - 2\,kHz$ befindet, ist ein hier ein AAF theoretisch nicht von Nöten.

3.3.2 Analog-Digital-Wandlung

Bei der Umsetzung von analogen zu digitalen Signalen bedarf es zwei Schritte. Zum einen muss eine Abtastung des zeitkontinuierlichen Signals erfolgen und zum anderen müssen die Amplitudenwerte des analogen Signals in diskrete Werte umgewandelt werden (Quantisierung). Das Bauteil, was diesen Vorgang durchführt, wird Analog-Digital-Umsetzer (-Wandler), AD-Umsetzer (-Wandler) oder kurz ADU genannt. In dieser Arbeit soll die Abkürzung ADU verwendet werden. In Abbildung 3.3 auf Seite 21 ist ein solcher ADU abgebildet, der eine Wortlänge von bis zu 12 Bit unterstützt.
In Abbildung 3.2 auf Seite 21 wird eine Tabelle angegeben, die einen Eindruck der im

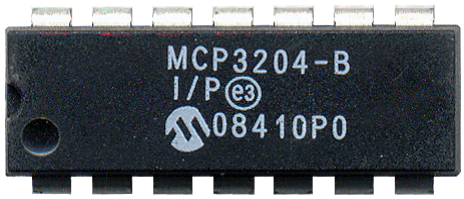


Abbildung 3.3: 12 *Bit*-Analog-Digital-Wandler
Bild:[parallax]

Alltag benutzten Abtastfrequenzen, Wortlängen, damit errechnete Quantisierungsschritte und deren Datenmenge pro Sekunde, in der Audiobranche zeigt.

	f_{abtast}	n-Bit	Quantisierungsschritte	Datenmenge/s
PC	8-22 kHz	8	256	64-176 kBit
CD	44,1 kHz	16	65536	705,6 MBit
Tonstudio	88,2 kHz	24	16777216	2,1168 GBit

Tabelle 3.2: Qualitätsübersicht von Medien
Eigene Darstellung, Anlehnung an [wedel]

Abtastung

Die Abtastung eines Signals bedeutet trivial beschrieben eine Stichprobe des Signals an äquidistanten Abtastpunkten, die alle das gleiche Abtastintervall T besitzen. Das Ergebnis dieser Abtastung ist ein zeitlich diskretes Signal mit kontinuierlicher Amplitude. Dabei ist die Frequenz, mit der das Signal abgetastet wird, eine entscheidende Größe. Allgemein gilt das Prinzip "*So klein wie möglich, aber so groß wie nötig*". Ziel dabei ist es, bestimmte

Fehler zu vermeiden, da sonst eine Rekonstruktion des Signals nicht mehr ohne Informationsverlust möglich ist. Um dies zu erreichen ist es zunächst von Nöten, dass in Kapitel 3.1 angesprochene Abtasttheorem einzuhalten. Dieses besagt, dass ein bandbegrenztes Signal mindestens mit der doppelten Frequenz der maximal im Signal auftretenden Frequenz abgetastet werden muss. In anderen Worten:

> *"Ein Signal ist eindeutig durch seine Abtastwerte x(nT) bestimmt, wenn die Abtastfrequenz f_s größer ist als das Doppelte der Grenzfrequenz f_{max}."([grueningen], S.57)*

Bei Verletzung dieses Theorems wird es in den aller meisten Fällen zu Fehlern, wie beispielsweise bei zu geringer Abtastfrequenz dem Aliasing-Effekt, kommen. Ist dies der Fall, spricht man von einer Unterabtastung.[grueningen]
Aliasing kommt jedoch für gewöhnlich immer vor, da Anti-Aliasing-Filter keine idealen Bauteile darstellen und nur begrenzte Flankensteilheit besitzen. Diesem Phänomen kann mit einem überabgetasteten ADU entgegengewirkt werden.[grueningen]
Wird die Äquidistanz der Abtastpunkte nicht eingehalten (Takt-Jitter), so tritt die so genannte Jitter-Noise auf. [dobl]
Eine ideale Abtastung kann man sich als einfache Multiplikation vorstellen. Dabei wird das Eingangssignal $x(t)$ mit einer Dirac-Impulsfolge $\delta_T(t)$ multipliziert. Veranschaulicht wird dies in Abbildung 3.4 auf Seite 22. Das abgetastete Signal lässt sich dann beschreiben durch:

$$\begin{aligned} x_s(t) &= x(t) \cdot \delta_T(t) \\ &= x(t) \sum_{n=-\infty}^{+\infty} \delta(t - nT) \\ &= \sum_{n=-\infty}^{+\infty} x(nT)\delta(T - nT) \end{aligned}$$

Da die im folgenden Abschnitt beschriebene Quantisierung einige Zeit in Anspruch nimmt,

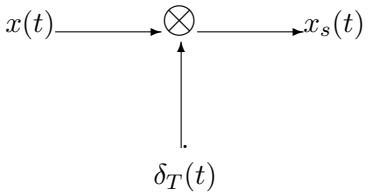


Abbildung 3.4: Idealer Abtaster
eigene Darstellung, Anlehnung an:[grueningen] S.54

ist es im Allgemeinen üblich, dass im ADU eine Haltschaltung (engl. sample-and-hold) integriert ist. Ist die Umsetzzeit des AD-Wandlers kleiner als die Änderungszeit des Signals, benötigt man solch ein Bauteil nicht.[kurz] Diese Schaltung, bestehend aus aktiven Bauelementen, sorgt dafür, dass Eingangsspannungen für eine bestimmte Zeit gehalten werden,

also zwischengespeichert werden. Dafür wird prinzipiell ein Schalter und ein hochwertiger[2] Kondensator gebraucht. Ist der Schalter, der beispielsweise durch einen taktgesteuerten Transistor realisiert werden kann, geschlossen, lädt sich der Kondensator auf. Ist der Schalter geöffnet, hält die Kondensatorschaltung die Spannung, mit der sie zuvor aufgeladen wurde. Es gibt Halteschaltungen verschiedener Ordnung. Die genaue Funktionsweise kann in [kurz] nachgelesen werden.

Amplitudenquantisierung

Die Amplitudenquantisierung wandelt die kontinuierlichen Amplitudenwerte des Signals in diskrete Zahlenwerte (mit bestimmter Wortlänge in Bit) um. Quantisierungen mit $16Bit$ ($2^{16} = 65536$ Quantisierungsstufen) im Audio- und $8Bit$ im Bildverarbeitungsbereich sind üblich. Die Zahlenwerte sind gebräuchlicher Weise eine Folge von binären Werten in Zweierkomplementdarstellung.[proch]
Die Amplitude wird in Amplitudenintervalle geteilt. Alle Werte, die in diesem Intervall kursieren, bekommen den Indexwert dieses Intervalls. Dabei wird zu dem Intervall gerundet, an dem der abgetastete Wert am nähesten liegt. Im Zuge dieser Operation wird klar, dass eine Quantisierung immer einen Informationsverlust zur Folge hat, da Rundungen verwendet werden. Man könnte annehmen, diese Abweichung wäre vorhersehbar, da es zu einer maximalen Abweichung von $\frac{p}{2}$[3] kommen kann. Doch die Abfolge von Aufrunden und Abrunden, zu den jeweiligen Quantisierungsstufen ist statistisch zufällig. Man nennt dieses Zufallssignal auch Fehlersignal $e(k)$.[grueningen]
Man kann festlegen, dass sich das digitalisierte Signal $x_Q(k)$ aus dem Fehlersignal, was auch als Quantisierungsrauschen bezeichnet wird, und dem Eingangssignal zusammensetzt. [enden]
Mathematisch:

$$x_Q(k) = e(k) + x(k). \tag{3.3}$$

Aus [grueningen] geht hervor, dass dieses Rauschen als weißes Rauschen angenommen werden kann - denn alle Frequenzanteile sind gleich verteilt. Durch diese Überlagerung verschlechtert sich die Signaldynamik (auch Dynamikbereich) des Signals. Unter Dynamikbereich versteht man einfach gesagt, den Abstand zwischen den größten und dem niedrigsten, darzustellenden Wert. Je höher der Dynamikbereich eines ADUs, desto genauer können Werte dargestellt werden.
Das Verhältnis von Nutzsignal $x(k)$ und Störsignal $e(k)$ wird als Signal-Rauschabstand (SNR) bezeichnet, englisch *Signal-to-Noise-Ratio*.[kester] Dieses Verhältnis dient als Qualitätsmaß von ADUs, deren Abhängigkeit jedoch von Signalart zu Signalart variiert. Dabei spielt der Crest-Faktor eine Rolle, welcher beschreibt, wie groß das Verhältnis zwischen Spitzenwert und Effektivwert des Signals ist. [dobl]
Näheres dazu und zum SNR, kann in [grueningen], [dobl] oder [enden] nachgelesen werden.

[2] Hochwertig meint hier einen Kondensator mit kurzen Ladezeiten.
[3] p repräsentiert hier der gemessene Abtastwert.

3.3.3 Grafisches Umwandlungsbeispiel

In den Abbildungen 3.6a bis 3.6d ist ein grafisches Beispiel mit kurzen Erläuterungen zu einer Digitalisierung eines analogen Signals dargestellt. Das Abtastintervall beträgt $t_s = 0.2ms$, woraus sich mit der Formel 3.4 auf Seite 24

$$f_a = \frac{1}{t_s} \tag{3.4}$$

eine Abtastfrequenz von $f_a = 5kHz$ ergibt.
Als erstes wird das Signal von Abbildung 3.5 auf Seite 24 vorbearbeitet. Um das Signal nicht unterabzutasten wird es im AAF tiefpassgefiltert. Dabei werden zu hohe Frequenzanteile, die zu der gewählten Abtastfrequenz dem Abtasttheorem (Formel 3.1 auf Seite 17) nicht genügen, heraus gefiltert. Dazu muss das Tiefpassfilter dementsprechend dimensioniert sein.
In Abbildung 3.6a auf Seite 24 ist das Signal von Abbildung 3.5 auf Seite 24 nach diesem

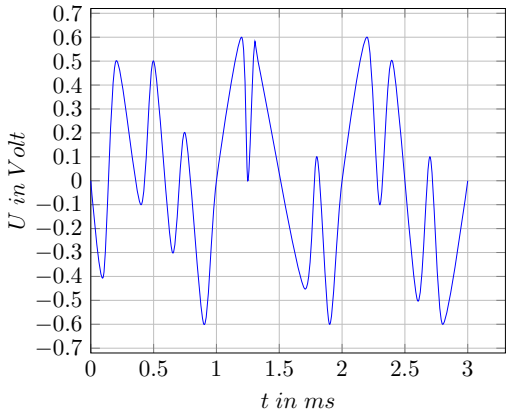


Abbildung 3.5: Analoges Signal, Spannung $U\ in\ Volt$ über Zeit $t\ in\ ms$
Eigene Darstellung

Vorgang abgebildet. Die Werte sind rein anschaulicher Natur.
Der nächste Schritt entspricht der zeitlichen Abtastung des Signals. Dabei werden dem Signal, in Abhängigkeit von der Abtastfrequenz (auch Samplefrequenz), in bestimmten Zeitabständen Messwerte entnommen. In Abbildung 3.6c auf Seite 26 ist der Abtastvorgang, beziehungsweise viel mehr die Abtastpunkte, dargestellt.
Nun muss das zeitlich begrenzte Signal noch wertediskret gestaltet werden. Dafür wird zunächst eine geeignete Bitwortlänge gewählt. In diesem Beispiel ist die Wortlänge $n = 3$, daraus folgen $N = 2^n = 8$ Quantisierungsstufen. Zunächst sollen jedoch noch einige Begriffe erläutert und berechnet werden.

Quantisierungsfehler

Der Quantisierungsfehler A_{quant} eines Wertes wird mit Formel 3.5 auf Seite 25 berechnet:

$$A_{quant} = U_M - U_W \tag{3.5}$$

Dabei ist U_M der errechnete Digitalwert und U_W der gemessene Wert der analogen Eingangsspannung. U_W lässt sich auch durch die Beziehung der Formel

$$U_W = U_q \cdot p \tag{3.6}$$

berechnen, wobei U_q die Breite der Quantisierungsstufe und p den gemessenen Digitalwert repräsentiert.[mikroq]
U_q wird durch Formel 3.7 auf Seite 25 beschrieben.

$$U_q = \frac{Messbereich\,des\,ADU}{N} \tag{3.7}$$

In diesem Beispiel wäre der Qunatisierungsfehler für Abtastpunkt 3 wie in den Gleichungen 3.8 auf Seite 25 bis Gleichung 3.11 auf Seite 25 zu berechnen:

$$U_q = \frac{0.5V - (-0.5V)}{8} = 125mV. \tag{3.8}$$

Aus Formel 3.6 auf Seite 25 folgt:

$$p = \frac{U_W}{U_q} = \frac{0.458V}{0.125V} = 3.664, \tag{3.9}$$

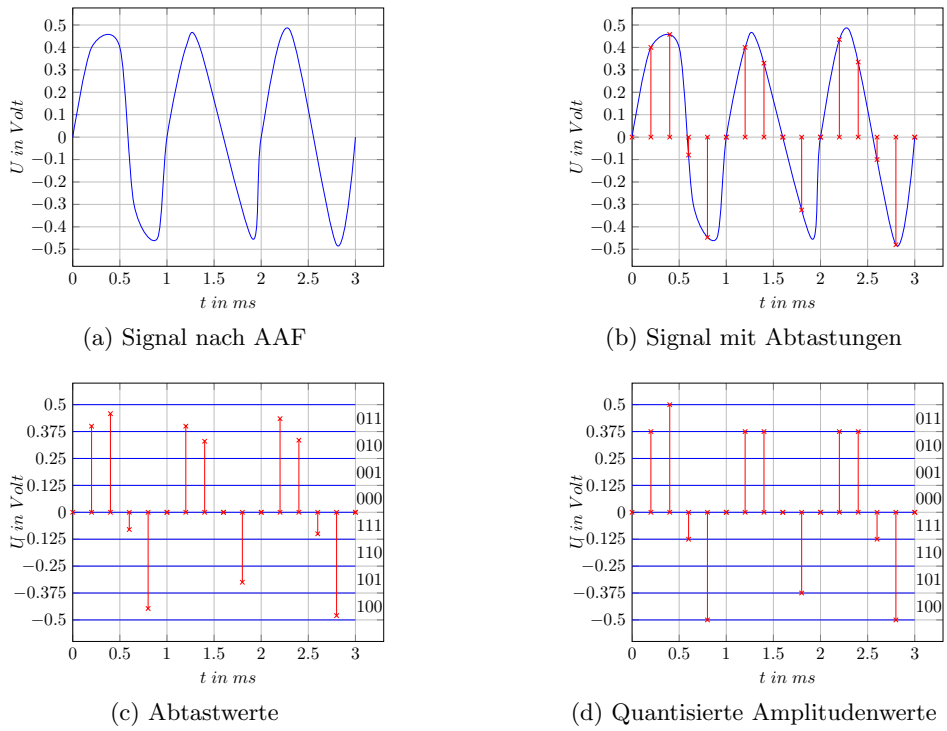
p wird gerundet auf: $p = 4$.
U_M lässt sich jetzt berechnen durch:

$$U_M = 0.125V \cdot 4 = 0.5V, \tag{3.10}$$

was der obersten Quantisierungsstufe (011) entspricht. Abschließend lässt sich nun der Quantisierungsfehler berechnen:

$$A_{quant} = U_M - U_W = 0.5V - 0.458V = 0.042V. \tag{3.11}$$

Dies liegt im Rahmen des zulässigen Maximalfehlers, welcher durch $+/-0.5\,LSB$ beschrieben wird.[mikroq] LSB ist dabei das niedrigste Bit im Quantisierungswort (hier $0.125V$). So lässt sich nun auf Grundlage der Rechnung und dem Signal von Abbildung 3.6d auf Seite 26 jeder Abtastwert berechnen und liefert das Ergebnis von Abbildung 3.6d auf Seite 26. Das digitale Signal ist dann $x_Q(k) = [010, 011, 111, 100, 000, 010, 010, 000, 101, 000, 010, 010, 111, 100, 000]$.
Nach dem die Zahlenfolge im Digitalrechner bearbeitet wurde, muss diese nun wieder analogisiert werden.



(a) Signal nach AAF

(b) Signal mit Abtastungen

(c) Abtastwerte

(d) Quantisierte Amplitudenwerte

Abbildung 3.6: Digitaliesierungsvorgang
Eigene Darstellung

3.3.4 Digital-Analog-Wandler

Sind die digitalen Daten im DSP verarbeitet, werden sie auch in analoger Form benötigt. Beispielsweise braucht man analoge Signale für hörbare Musik (um die Membran des Lautsprechers in Schwingung zu versetzen). In der Fachliteratur wird dabei häufig von einer Rekonstruktion gesprochen. Diese ist verlustfrei, wenn das Abtasttheorem eingehalten wurde. Die Umwandlung von digital nach analog, geht in so genannten Digital-Analog-Wandlern (DAW) oder Digital-Analog-Umsetzer (DAU) von statten. Die Grundgedanke eines solchen DAUs ist es, die digitalen Werte auf Impulse zu übertragen. Diese erzeugen mit einem, im DAU verbauten, Halteglied ein stufenförmiges Ausgangssignal, welches durch einen Tiefpass geglättet wird.[kurz]

3.3.5 Delta-Sigma-Wandler

Heute werden fast ausschließlich Delta-Sigma-Wandler für die Anwendung in DSPs verwendet. Es handelt sich dabei um einen 1-Bit-Wandler, der einen 1-Bit-Datenstrom erzeugt. Dabei steht eine 1 für eine Steigung des Signals und eine 0 für einen Abfall. Der Bitstrom ist also nichts anderes als eine Folge von Einsen und Nullen, die anzeigen ob das Signal zum Zeitpunkt steigt oder fällt. Daraus wird ersichtlich, dass bei einer höheren Betriebstaktrate auch die Genauigkeit der Umwandlung steigt, denn je öfter geprüft wird, wie sich die Funktion des Signals verhält, desto genauer sind die Informationen über sie. Man spricht dabei auch von der Technik des *Oversampling* (deutsch *Überabtasten*). Bleibt der

Signalverlauf konstant, so wechseln sich Einsen und Nullen ab, sodass eine Gerade durch eine Dreiecksverlauf approximiert wird. Intern passiert dies mit einem Differenzverstärker, welcher vor einem Integrator geschaltet ist. Dieser Ausgang wird dann zu einem Komparator geleitet, welcher entscheidet, ob eine 1 oder eine 0 ausgegeben wird. Rückwirkend wird der Ausgang des Komparators über den Differenzverstärker zum Integrator geführt. Das eigentliche Ausgangssignal wird mittels Tiefpass zum endgültigen analogen Signal.[iti]

3.4 Digitale Filter

Digitale Filter sind das klassische Anwendungsgebiet der digitalen Signalverarbeitung. Ihr Prinzip ist schon länger bekannt, wurde aber erst in den 80er Jahren mit dem ersten Aufkommen von DSP's populär. Nahezu alle Filterprobleme unter $100\,kHz$ werden heute mit digitalen Filtern gelöst, wobei sich diese Grenze in Zukunft nach oben verschieben wird, da sich die benötigte Hardware in stetiger Entwicklung befindet.
Ein Filter kann man als ein System definieren, welches Frequenzkomponenten vom Eingang gegenüber dem Ausgang verändert. Verändern schließt dabei Vorgänge wie Verstärken, Unterdrücken oder in der Phase verschieben ein. Systeme, die dabei ein kausales und stabiles LTI-System repräsentieren, spielen hierbei eine große Rolle.[grueningen]
Klassische digitale Filter sind Hochpass-, Tiefpass-, Bandpass- und Bandsperrfilter. Wie die Bezeichnungen schon andeuten, haben diese Filter die Funktionen, tiefe oder hohe Frequenzkomponenten zu unterdrücken oder nur bestimmten Frequenzanteilen Durchgang zu gewähren. Dabei spricht man allgemein vom Durchlass-, Übergangs- und Sperrbereich. In Abbildung 3.7 auf Seite 28 ist der Frequenzgang eines Hochpassfilters zu sehen. Zusätzlich wurden dort die eben genannten Bereiche eingezeichnet. Die Punkte auf der Amplitudenachse heißen Rippel (im Sperr- oder Durchlassbereich).
Da technisch gesehen eine vertikale Flankensteilheit, wie bei dem gezeigten Frequenzgang des Hochpassfilters, nicht erreicht werden kann, beziehungsweise nur dann erreicht wird, wenn man eine unendlich hohe Ordnung wählt, müssen Filterentwürfe approximiert werden. Beispielsweise besteht ein idealer Rechteckimpuls aus einer Folge von allen vorkommenden Frequenzen, womit das Spektrum dieses Rechteckpulses eine niemals vollständig abklingende *sinc*-Funktion darstellt (siehe Abbildung 2.5 und 2.6 auf Seite 10). So ähnlich verhält es sich mit der Flankensteilheit von idealen Filtern. Um sich trotz alledem an die unendlichen Spektren anzunähern, kann man diese fenstern. Um ein unendliches Spektrum zu begrenzen und somit auf eine endliche Impulsantwort zu kommen, multipliziert man die ideale Impulsantwort mit einer Funktion, die eine abklingende Eigenschaft besitzt. Darunter fallen unter anderem das Hamming- oder das Kaiser-Fenster. Die Fenstermethode ist unkompliziert in ihrer Umsetzung, da sie nur einer Multiplikation bedarf. Will man jedoch ein Filter entwerfen, der mit minimaler Ordnung auskommt, so greift man meist zu der Optimalmethode, welche das Standardverfahren symbolisiert. Bei dieser Approximation nutzt man beim Entwurf den gesamten Toleranzbereich des Filters aus. Dabei würde sich der Frequenzgang von Abbildung 3.7 auf Seite 28 über den gesamten schraffierten Bereich ziehen. Weitere Informationen dazu hält [grueningen] bereit.
Wie bereits in Kapitel 2.5.1 erwähnt, lässt sich das Verhalten eines Systems vollständig

mit der zugehörigen Übertragungsfunktion $H(z)$ beschreiben. Mit einigen Umformungen und einer Transformation in den Zeitbereich erhält man die Differenzengleichung in der Form von Formel 2.27 auf Seite 14. Mit dieser Gleichung kann das Signalflussdiagramm, oder auch Blockdiagramm, des Filters generiert werden.
Vorausgehend wurde erwähnt, dass Zähler und Nenner jeweils den nicht rekursiven beziehungsweise rekursiven Anteil des Filters beschreiben. So lassen sich zwei Arten von Filtern klassifizieren. Ist der rekursive Anteil 0, es gilt $\forall a_i = 0$, so ist die Rede von einem FIR-Filter (*finite impulse response*, deutsch endliche Impulsantwort). Dies wird deutlich, da alle a_i mit dem verzögertem Ausgang verknüpft sind. Nun ist der Ausgang nicht mehr von sich selbst und dem Eingangssignal, sondern ausschließlich von der Eingangsgröße abhängig. Ist dies nicht der Fall, so ist die Rede von IIR-Filtern (*infinite impulse response*, deutsch unendliche Impulsantwort), deren Impulsantworten unendliche Länge besitzen, da sie den rückgekoppelten Ausgang mit einbeziehen.

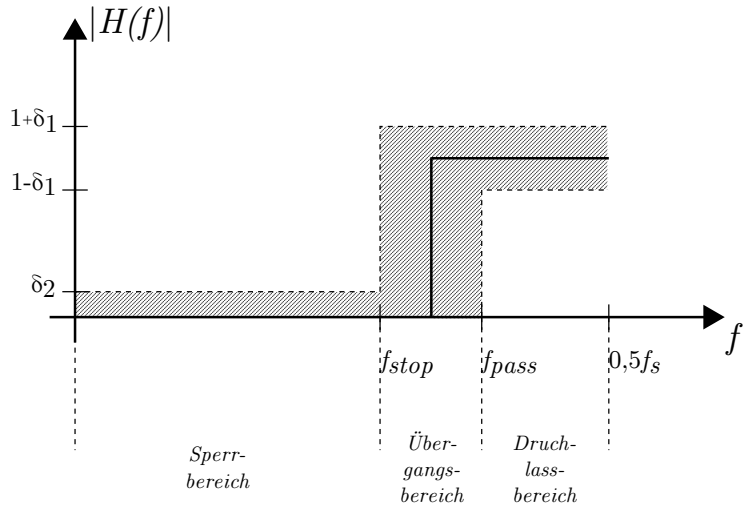


Abbildung 3.7: Idealer Frequenzgang Hochpass
Bild: Eigene Darstellung

3.4.1 Signalflussdiagramm

Systeme können zur Veranschaulichung in Signalflussdiagrammen dargestellt werden. Bei der Umsetzung hilft die Differenzengleichung. Es werden drei verschiedene Operationen dargestellt: Addition, Multiplikation und Verzögerung. Dabei wird eine Addition durch einen nicht ausgemalten Punkt, eine Multiplikation durch einen Faktor und eine Verzögerung mit einem z^{-1} am Signalflussgraph dargestellt. Eine Verzweigung, beziehungsweise Anfangs- und Endknoten werden durch geschwärzte Kreise repräsentiert.[han] Das Diagramm in Abbildung 3.8 auf Seite 29 stellt die Differenzengleichung:

$$y(k) = ax_1(k) + x_2(k-1) \tag{3.12}$$

dar.

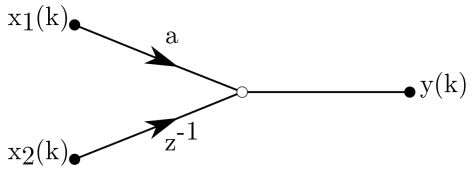


Abbildung 3.8: Signaflussdiagramm von Formel 3.12
Bild: eigene Darstellung

3.4.2 FIR-Filter

FIR-Filter besitzen eine endliche Impulsantwort, was bedeutet, dass deren Filterkoeffizienten direkt aus dem Zählerpolynom der Übertragungsfunktion abzulesen sind. Noch besser lässt sich die Struktur des Filters mit der Differenzengleichung aufbauen. Die Struktur ist das schon angesprochene Signalfluss- oder Blockdiagramm eines Filters. Nimmt man an, dass die Übertragungsfunktion eines Filter folgende Form hat:

$$H(z) = b_0 + b_1 z^{-1} + \cdots + b_N z^{-N}, \tag{3.13}$$

dann ist durch die Transformation in den Zeitbereich die Impulsantwort $h(k)$ gegeben durch:

$$h(k) = b_0\delta(k) + b_1\delta(k-1) + \cdots + b_N\delta(k-N) \tag{3.14}$$

oder in Sequenzschreibweise durch:

$$\{h(k)\} = \{b_0, b_1, \cdots, b_N\}. \tag{3.15}$$

Liegt ein spiegelsymmetrisches Filter vor, ist auch die Funktion im Diagramm spiegelsymmetrisch. Dann ist die Folge der Koeffizienten umkehrbar.[grueningen]
Es gibt verschiedene Strukturtypen von FIR-Filtern. Am gebräuchlichsten ist die so genannte Direktform oder Transversalform. Ein Beispiel dafür kann in Abbildung 3.9 auf Seite 30 betrachtet werden. Diese Struktur wurde genutzt um den praktischen Teil dieser Arbeit zu realisieren. Die Differenzengleichung der abgebildeten Struktur ist die folgende:

$$y(k) = b_0 x(k) + b_1 x(k-1) + \cdots + b_N x(k-N). \tag{3.16}$$

Man nehme an, dass ein Hochpass realisiert werden soll. Der zugehörige Frequenzgang ist analog zu dem des Hochpasses von Abbildung 3.7 auf Seite 28. Wie in vorherigen Kapiteln schon aufgezeigt wurde, ist die Fourier-Transformierte eines Rechtecksignals eine *sinc*-Funktion und umgekehrt hat die Fourier-Transformierte einer *sinc*-Funktion einen rechtecksignalförmigen Verlauf. Um nun mittels diskreter Faltung (FIR-Filter) einen Hochpass zu realisieren und die dafür benötigten Filterkoeffizienten zu ermitteln, muss vom Frequenzgang des Hochpasses, die inverse Fourier-Transformierte ermittelt werden und diese entsprechend in den positiven Zeitbereich verschoben werden. Die gegebene Funktion im Zeitbereich wird dann abgetastet - die Abtastpunkte stellen die Filterkoeffizienten dar.

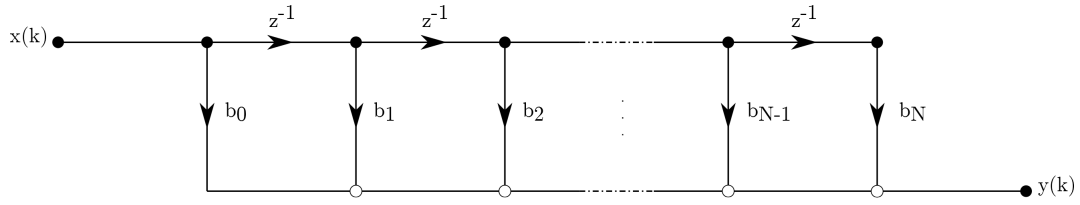


Abbildung 3.9: Signalflussdiagramm FIR-Filter
Bild: eigene Darstellung, Anlehnung an [grueningen]

3.4.3 IIR-Filter

IIR-Filter sind Filter mit einer unendlich langen Impulsantwort. Dies rührt daher, dass bei dieser Filterart das Nennerpolynom der Übertragungsfunktion nicht 0 ist. Auch bei IIR-Filtern überführt man die Übertragungsfunktion der Form von Formel 2.28 auf Seite 14 in eine Differenzengleichung der Gliederung von Formel 2.27 auf Seite 14. Hier unterteilt man diese Rechenvorschrift in ein Paar mit folgendem Aufbau[grueningen]:

$$w(k) = \sum_{i=0}^{N} b_i x(k-i) \tag{3.17}$$

und

$$y(k) = w(k) - \sum_{i=1}^{N} a_i y(k-i). \tag{3.18}$$

Dabei ist der erste Teil der nicht rekursive und der zweite der rekursive mit dem Eingang $w(k)$ und dem Ausgang $y(k)$. Man beschreibt nun die transformierte Differenzengleichung als eine Kaskade von zwei Systemen mit der Form:

$$Y(z) = \frac{1}{A(z)} \cdot B(z) \cdot X(z). \tag{3.19}$$

Daraus ist es möglich, analog zum FIR-Filter, die Struktur des Filters abzulesen. Diese Struktur von IIR-Filtern wird auch Direktform-I-Struktur genannt. Es existieren noch andere Formen von IIR-Filtern, wobei hier auf [grueningen] verwiesen wird.
Beim Entwurf von IIR-Filtern geht man meist mit der biliniearen Transformation vor, welches ebenfalls bei der Dimensionierung von analogen Filtern benutzt wird. Auch hier kommen Approximationsverfahren zum Einsatz, wie Butterworth, Tschebyschaeff oder Cauer. [grueningen]

3.4.4 Hallerzeugung

Da eines der Hauptziele dieser Arbeit darin besteht, mit dem LTI-System ein Eingangssignal in Echtzeit mit der Impulsantwort einer Kirche zu falten, ist es von Interesse, auf welche Weise Halleffekte erzeugt werden können.
Grob lassen sich dabei zwei Optionen wählen. Bei der ersten wird der Hall auf Grund eines FIR-Filters implementiert, bei der zweiten ist eine Implementierung eines algorithmischen Halls in Form eines IIR-Filters möglich.[jaeger]

Ein FIR-Filter bietet die Möglichkeit eine Zeitbereichsfaltung durch zuführen. Dabei wird eine Impulsantwort eines hallenden Raumes benötigt, die mit dem Eingangssignal in Echtzeit gefaltet werden kann. Eine andere Möglichkeit mittels FIR-Filter bietet die schnelle Faltung. Die schnelle Faltung nutzt das Faltungstheorem, um wesentlich effizienter als die erste Methode zu rechnen. Dazu wird das Eingangssignal und die Impulsantwort mittels FFT in den Frequenzbereich transformiert. Danach werden beide Spektren multipliziert, um abschließend eine Rücktransformation des gefalteten Signals zu errechnen.
Für die zweite Möglichkeit werden IIR-Filter benötigt, da für die Erzeugung von Hall unendlich viele Echos mit verschiedenen Laufzeiten generiert werden müssen. Dabei stellt eine Verzögerung, also eine Rückführung des Ausgangs (Eigenschaft von IIR-Filtern), einen Reflektionspunkt des Schalls im Raum dar. Steigt die Anzahl der Punkte, steigt auch die Komplexität des Filters und umso rechenaufwändiger wird er.[ahle]
Um ein einfaches Echo zu erzeugen addiert man zum Ausgang den Eingang plus einen gedämpften Teil des Ausgangs. Durch den Dämpfungsfaktor, der < 1 sein muss, klingt das Echo nach und nach ab. Ein auf diese Weise generiertes Echo wurde ebenfalls auf dem DSP implementiert.

3.4.5 Überlappungsmethoden

Es kann vorkommen, dass ein Signal mit endlicher Länge mit einem Signal unendlicher Länge gefaltet werden muss. Da man für die direkte Umsetzung einen unendlich großen Speicher benötigen würde und man unendlich lange Verzögerungen für das Ausgangssignal in Kauf nehmen müsste, sollte man die Berechnung in Teilrechnungen zerlegen. Diese durchaus hilfreiche Operation nennt sich Blockverarbeitung. Ein weiterer Grund für diese Methode ist, dass der Prozessor in der Zeit, wo kommende Daten aufgenommen werden, keiner großen Belastung ausgesetzt ist. Diese kann man nutzen um während der Entgegennahme der Daten, parallel die nötigen Berechnungen ausgeführt werden.
Außerdem ist die Tatsache, dass sich die diskrete Fourier-Transformation beziehungsweise die Fast-Fourier-Transformation nur auf endliche Signale abbilden lassen, ein weiterer Grund für eine Zerlegung des Eingangssignals. Dabei ist besonders darauf zu achten, die Teilergebnisse nach der Berechnung wieder fließend zusammen zu fügen. Es existieren zwei bekannte Methoden die sich mit dem Überlappungsverhalten befassen.

overlap-add-Methode

Bei der *overlap-add*-Methode teilt man das Eingangssignal in Blöcke der Länge L, wobei $L \geq N$. N definiert die Länge der Impulsantwort, mit der das Eingangsignal gefaltet werden soll. Aus dieser Faltung entsteht ein Teilsignal, welches die Länge $L + N - 1$ besitzt und somit $N - 1$ Stellen mit dem folgenden Teilsignal überlappen würde. Um die Übergange zu verbinden, werden jeweils die überstehenden Teile mit den ersten Werten des Folgeblocks addiert - daher rührt auch der Name *overlap-add*. Die letzten $N - 1$ Ergebnisse des endständigen Faltungsblockes werden dabei, insofern das Ausgangssignal die gleiche Länge wie das Eingangssignal haben soll, nicht berücksichtigt, da dies den Ausschwingvorgang beinhaltet. Möchte man jedoch ein Echo erzeugen, so müssen diese

Werte angehängt bleiben.[dobl]
In Abbildung 3.10 auf Seite 32 ist diese Methode grafisch dargestellt.

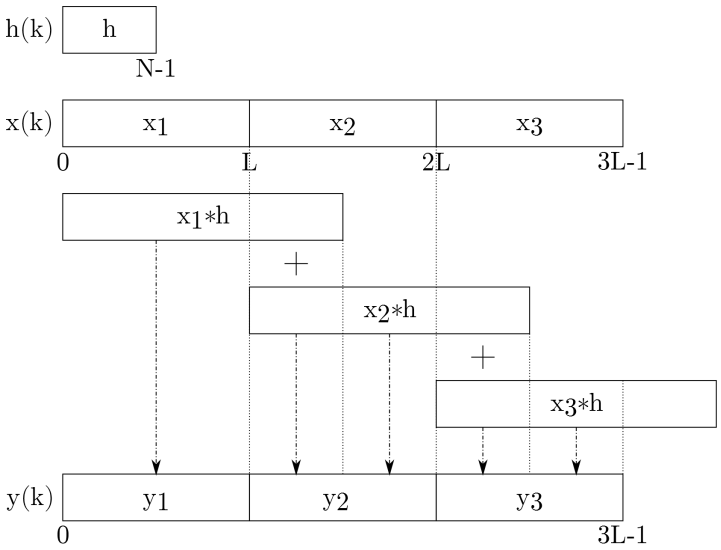


Abbildung 3.10: Schematische Darstellung der *overlap-add*-Methode
Bild: eigene Darstellung, Anlehnung an [dobl]

overlap-save-Methode

Bei der *overlap-save*-Methode wird das Eingangssignal ebenfalls in Teilblöcke zerlegt, jedoch werden die überlappenden Teile nicht addiert. Bei dieser Variante bilden jeweils die letzten M Werte des Vorgängerblockes die Anfangswerte des nächsten Blockes. Da diese Blöcke sozusagen gesichert werden, um sie dem nächsten Block zu übergeben, ist auch hier die Bezeichnung *overlap-save*-Methode schlüssig. Somit werden die ersten M Werte eines Datenblockes nicht berücksichtigt. Dies hat den Vorteil, dass die Addition gegenüber der *overlap-add*-Methode wegfällt. Dafür zieht dieses Verfahren einen Mehraufwand durch den Kopiervorgang mit sich.[dobl]
Eine schematische Darstellung dieses Verfahrens ist in Abbildung 3.11 auf Seite 33 illustriert.

3.5 Digitale Signalprozessoren

Zunächst einmal soll die Frage geklärt werden, worum es sich bei einem digitalen Signalprozessor (DSP) handelt. Ein DSP ist eine spezielle zentrale Recheneinheit (CPU), die insbesondere für die effiziente und zeitextensive Verarbeitung von digitalen Signalen entwickelt wurde und eine "*[...] effiziente Implementierung von Algorithmen der digitalen Signalverarbeitung [...]"([dobl], S.1)* ermöglicht.[mikrod]
Oft wird ein DSP auch mit Mikrocontrollern gleichgesetzt. Als Mikrocontroller bezeichnet man jedoch eine Recheneinheit (Prozessor) im Verbund mit Zusatzmodulen. Dies können Ein- und Ausgänge, Speicher oder Timer sein, die meist zusammen auf einem einzigen Chip angeordnet sind. Anwendung finden diese unter anderem in Ladegeräten, Motorsteuerun-

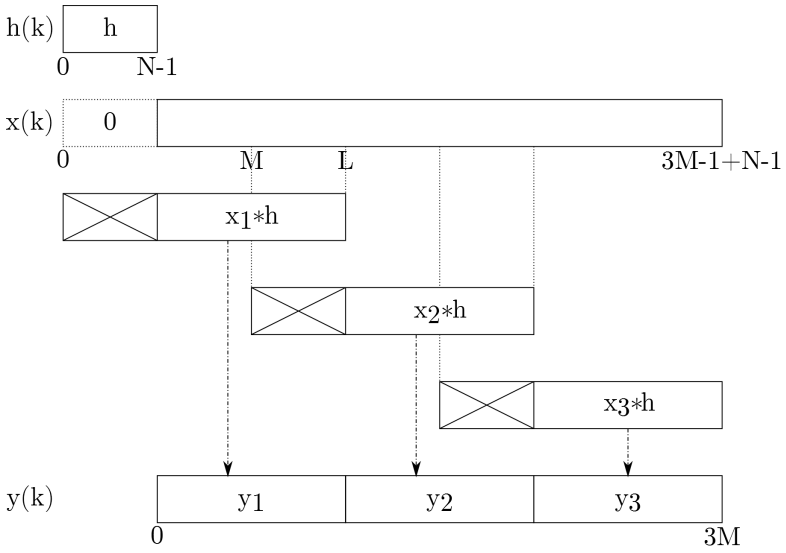


Abbildung 3.11: Schematische Darstellung der *overlap-save*-Methode
Bild: eigene Darstellung, Anlehnung an [dobl]

gen oder Messgeräten. Sie sind in ihrer Leistung nicht mit DSPs vergleichbar.[mikrom] Außerdem stiftet das *Field Programmable Gate Array*, besser bekannt als FPGA, oft zusätzlich Verwirrung in der Begriffswelt an. Diese programmierbare Logik ist, wie die beiden Namen schon sagen, ein Feld von programmierbaren Logikelementen (Flip-Flops). Diese können, in den neueren Varianten schon mit Hilfsprozessoren ausgestattet sein. Je größer das FPGA, desto größer die Rechenleistung. Durch ansteigende Größe, nimmt auch der Stromverbrauch, die Entwicklungszeit und die Produktionskosten zu, weswegen sich oft für effiziente DSPs entschieden wird.[mirkof]
DSPs besitzen ein weites Anwendungsspektrum, beginnend bei Kommunikation bis hin zum Verarbeiten von Sprache und Bildern. Das Hauptanwendungsgebiet ist jedoch die Kommunikationstechnik. Sie befinden sich in vielen alltäglichen Geräten, wie Telefonen, Modems, MP3-Player, Druckern, HiFi-Systemen oder Digitalkameras. Ein entscheidender Vorteil von DSPs ist, dass sie für jene Anwendungen programmiert werden können und somit ein einziger DSP verschiedene Aufgaben bewältigen kann, indem man die Software austauscht. Somit bieten DSPs eine kostengünstige Lösung für viele Anwendungsgebiete, die oftmals im Bereich der Echtzeitverarbeitung liegen.

3.5.1 Architektur

Bei digitalen Signalprozessoren handelt es sich um Skalarrechner. Dies bedeutet, dass im Unterschied zu Vektorrechnern keine Datenblöcke verarbeitet werden müssen um volle Durchsatzraten zu erhalten. Da DSPs meist dazu eingesetzt werden um kommende Signale, die sequentiell auftreten (zum Beispiel ein Audiosignal, bei dem digitale Abtastwerte mit einer bestimmten Rate nacheinander auftauchen), zu bearbeiten, ist es hier sinnvoll eine skalare Architektur zu wählen. Da DSPs hochspezialisierte Rechner repräsentieren, ist es wichtig schon beim inneren Aufbau auf Effizienz zu achten. Mit Architektur ist die Anordnung von Bussen, Speichern und Rechenwerken gemeint. Nahezu alle DSPs besitzen

eine modifizierte Harvard-Architektur. Zunächst soll jedoch erläutert werden, dass es zwei grundlegende Architekturen gibt, die von Neumann- und die Havard-Architektur. Die von Neumann-Architektur ist einfach aufgebaut, da sich bei der klassischen Variante der Speicher sowohl für Befehle, als auch für Daten aufteilt. In der einfachsten Option teilen sich die verschiedenen Datenströme sogar einen einzigen Bus. Ein Schema dieses Architekturtyps ist in Abbildung 3.12 auf Seite 34 zu sehen.
Eine klassische Harvard-Architektur bietet die Möglichkeit, separate Speicher und Busse für Befehls- und Datenströme zu benutzen. Die grundlegende Struktur ist in Abbildung 3.13 auf Seite 34 abgebildet. Ziel dieser Architektur ist es, die Rechenzeit zu minimieren, indem beispielsweise in einem Taktzyklus sowohl ein Befehlswort, als auch ein Datenwort transferiert werden können.[dobl]
Heute gibt es unzählig viele Modifizierungen dieser Architektur. Sie wird ihrem Anwendungsgebiet perfekt angepasst - so können optimale Ergebnisse in puncto Geschwindigkeit erreicht werden.
Der eingesetzte Prozessor ist ein skalarer Rechner mit Very-Long-Instruction-Word-Havard-Architektur[4]. Der Aufbau der Architektur des C6713^{TM}von Texas Instruments ist in Abbildung 3.14 auf Seite 35 dargestellt.

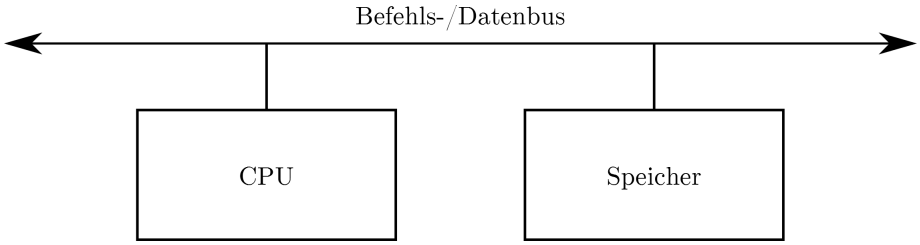


Abbildung 3.12: Allgemeine von-Neumann Architektur
Bild: eigene Darstellung

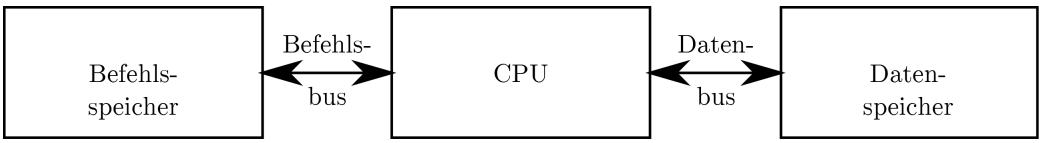


Abbildung 3.13: Allgemeine Havard-Architektur
Bild: eigene Darstellung

[4]VLIW sind lange Befehlswörter, die benötigt werden um alle Anweisungen der parallel rechnenden Einheiten unterbringen zu können

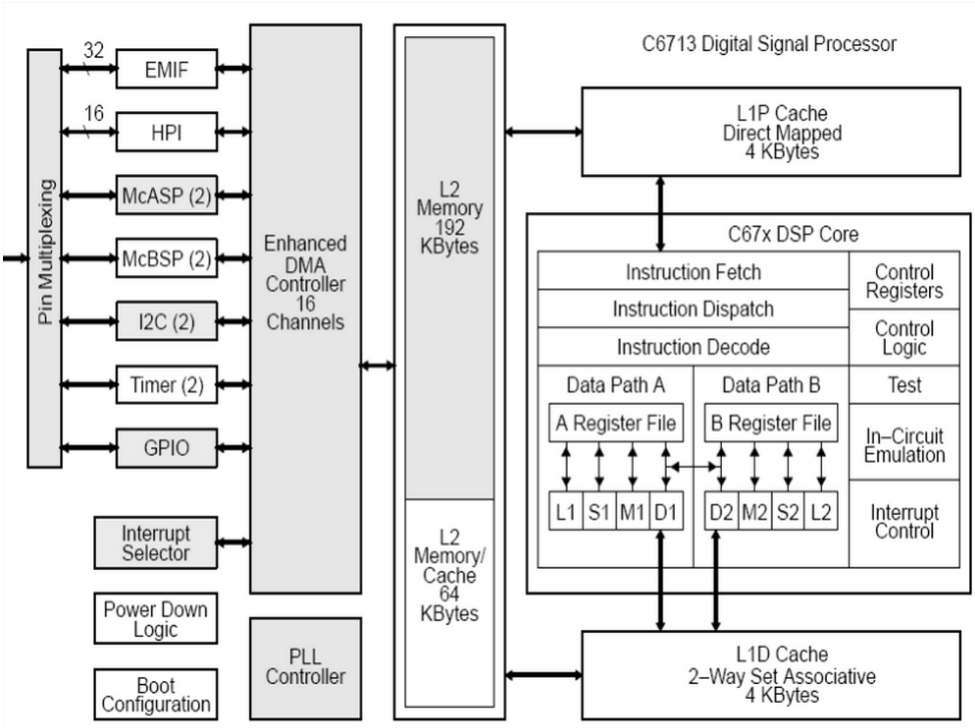


Abbildung 3.14: Erweiterte Havard-Architektur des C6713
Bild: [52lab]

4 Praktischer Teil

4.1 Aufgabenstellung

Die Aufgabe ist ein LTI-System in Form eines Hardwaremoduls zu entwerfen. Dies soll intern mittels eines zur Verfügung gestellten digitalen Signalprozessorboards umgesetzt werden. Zu den Teilaufgaben gehören auf der einen Seite der physikalische Aufbau der Black-Box und zum anderen der Entwurf des DSP-Programms in der Sprache C, einschließlich deren Test und Dokumentation.
Hardwarebetreffende Kriterien sind nur dadurch gegeben, dass die Black-Box mittels Eingangsklemmen ein Eingangssignal erhält und dieses nach der Bearbeitung im DSP über Ausgangsklemmen abgreifbar ist. Das Ausgangssignal soll dann an einem Oszilloskop visuell und verbunden mit einem Lautsprecher, akustisch wiedergegeben werden können. Zur Auswahl der verschiedenen Filtertypen, sollen von außen Schalter zur Verfügung stehen, mit denen die verschiedenen Funktionen gewählt werden können.
Seitens der Software wurden, neben den Designkriterien, drei verschiedene Varianten zur Umsetzung vorgeschlagen:

- Zeitbereichsfaltung ohne Latenz
- schnelle Faltung mit circa $100\,ms$ Latenz
- Entwurf eines IIR-Filters, mit Impulsantwortsannäherung mittels evolutionärer Optimierung.

Zu den Designkriterien zählt die Eingrenzung auf $16\,Bit\,Integer$ für Ein- und Ausgang, eine Nachhallzeit von $2\,s$ und die Begrenzung auf einen Kanal, also mono.
Um in die Thematik der DSV und von DSPs einzutauchen, bedurfte es erst einmal einer gründlichen Einarbeitung in den theoretischen Hintergrund.

4.2 Arbeitsumgebung

Zur Umsetzung der praktischen Aufgabe wurde das DSP Evaluation Board *TMS320C6713* der Firma Spectrum Digital^{TM} zur Verfügung gestellt. Des Weiteren wird ein Industrie-PC mit dem Betriebssystem Windows XP benutzt, auf welchem neben den nötigen Treibern, die Programme MATLAB, Audacity und Code Composer Studio v3.1 installiert sind. Das System wird mit einem $5\,V$ Spannungsgerät versorgt und ist über einem USB-Kabel mit dem PC verbunden. Das Board wird mit einem $3,5\,mm$ Klinke-Kabel mit Daten versorgt, welches am anderen Ende mit einer beliebigen Signalquelle verbunden werden kann, die ebenfalls eine Klinkenbuchse besitzt. Am Board selbst ist dieses Kabel mit dem *LINE*

IN-Anschluss verbunden. Außerdem ist am *LINE OUT*-Anschluss ebenfalls ein $3,5\,mm$ Klinke-Kabel, welches das Board mit dem PC verbindet, um dort die modifizierten Signale auswerten zu können. Dies ging meist mit der Aufnahme in MATLAB von statten, da damit aufwendige Verfahren, wie beispielsweise die Erstellung eines Spektrogramms, mit Hilfe von einer Kommandozeile angewendet werden können. Für andere Tests sind die Ein- und Ausgänge jeweils mit einem weiteren Anschluss überbrückt. Diese führen zu zwei Eingängen eines Oszilloskopes.

4.2.1 Kurzbeschreibung des *TMS320C6713 DSP*

Das Evaluation Board *TMS320C6713* ist ein rechenstarkes und vergleichsweise günstiges Paket, welches mit attraktiven Schnittstellen und nötiger Software geliefert wird. Das Board beinhaltet zum einen das Herzstück, den *C6713* floating-Point DSP, zum anderen wurde es mit einem *TLV320AIC23* Codec ausgestattet, welcher die Schnittstelle für Eingangs- und Ausgangssignale bietet. Die Abtastrate kann manuell eingestellt werden und reicht von minimal $8\,kHz$ bis zu maximal $96\,kHz$. Wenn gewünscht, kann das Board mit so genannten *Daughtercards* erweitert werden. Der Prozessor läuft mit einer Taktrate von $225\,MHz$. Das DSK wird mit $5\,V$ versorgt, wobei $1,26\,V$ für die Versorgung des Prozessors genutzt werden und die restliche Spannung für die Peripheriekomponenten, wie beispielsweise Speicher, verwendet wird. Es besitzt neben dem Versorgungsanschluss noch eine USB-Schnittstelle (TYP 2), womit das Board mit dem PC verbunden wird. Außerdem findet man vier $3,5\,mm$ Klinkebuchsen, die mit LINE IN, LINE OUT, MIC IN und HEADPHONE betitelt sind. Der DSP baut auf eine VLIW-Architektur. Somit ist es möglich, bei einer Taktrate von $225\,MHz$ jede $4,44\,ns$ acht Instruktionen ausführen zu können.
Als weitere Eckinformationen können die insgesamt acht Rechenwerke genannt werden, wobei diese aus sechs ALUs (*arithmetic-logic units*, deutsch *arithmetisch logische Einheit*) und zwei Multiplizierer bestehen. Auch zu erwähnen ist, dass der Prozessor sowohl Fest- als auch Gleitkommazahlen unterstützt.[wiley] [tref]
Der schematische Aufbau des Boards ist in Abbildung 4.1 auf Seite 39 zu sehen.

Speicher des *TMS320C6713*

Das Board besitzt unterschiedliche (schnelle) Speicher. Dabei handelt es sich um Flash-Speicher, SDRAM, Caches und Register, welche sich neben der Geschwindigkeit, auch in der Größe deutlich unterscheiden.
Der DSP verfügt über 32 $32\,Bit$ GP-Register (*General-Purpose Register*, deutsch *frei verfügbare Register*), was insgesamt einer Kapazität von $128\,Byte$ entspricht. Sie bilden die schnellsten frei verfügbaren Speicherelemente.
Der Cache des Baords teilt sich in zwei so genannte Levels auf. Der L1-Cache hat eine Größe von $8\,kByte$, wobei jeweils eine Hälfte davon den Programm-Cache beziehungsweise den Daten-Cache bilden. Der L2-Cache besitzt ein Fassungsvermögen von $256\,kByte$. Die bis hierhin beschriebenen Speicher bilden den internen Speicher des Prozessors. Deshalb spricht man dabei auch vom IRAM (*internal random access memory*, deutsch *interner*

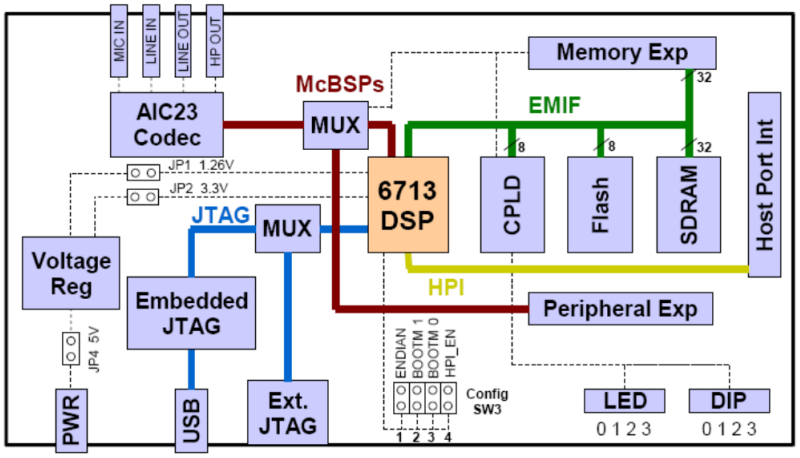


Abbildung 4.1: Aufbau des *TMS320C6713* DSK
Bild: [52lab]

Direktspeicher) (siehe Abbildung 3.14 auf Seite 35).
Speicher, der nicht direkt am DSP Anschluss findet, wird über den EMIF (*enhanced memory interface*, deutsch *erweiterter Speicherschnittstelle*) verwaltet. Dieser kann mit 32 *Bit*- beziehungsweise 8 *Bit*-Wörtern kommunizieren und steht mit dem SDRAM (16 *MByte*), dem Flash (256 *kByte*) und eventuellen Peripheriespeichern in Verbindung (siehe Abbildung 4.1 auf Seite 39). Dabei wird der Adressraum in 4 Bereiche geteilt. Der SDRAM liegt in $CE0$, der Flashspeicher und das CPLD in $CE1$. $CE2$ und $CE3$ sind für Daughter Cards reserviert.

4.2.2 Kurzbeschreibung des *TLV320AIC23* Codec

Das DSK ist mit einem hauseigenen Stereo-Codec mit der Typennummer *TLV320AIC23* bestückt und bildet die Schnittstelle für Eingangs- und Ausgangssignale. Der Codec tastet analoge Signale vom *MIC IN*- oder *LINE IN*-Eingang ab und wandelt diese in digitale Signale um, die vom DSP bearbeitet werden können. Nach der Bearbeitung wandelt der Codec das digitale in ein analoges Signal um und stellt es am *LINE OUT*- und *HEADPHONE*-Ausgang zur Verfügung. Der Unterschied der beiden Ein-/Ausgänge liegt in der Stärkerverschaltung. Bei den *MIC IN*- und *HEADPHONE*-Anschlüssen muss diese vorhanden sein, da diese Geräte meist keine eigene Stromversorgung besitzen.
Der Codec kommuniziert über zwei Wege. Der erste zum Konfigurieren (über McBSP0-Register), der zweite zum Senden und Empfangen von Daten (über McBSP1-Register). Auf ersterem spricht man mit einem 16-Bit-Wort, mit dem man verschiedene Einstellungen wie Lautstärke, Abtastrate oder Reset setzen kann. Dies muss vor der Benutzung durchgeführt werden. Der zweite Weg oder Kanal wird genutzt um die Nutzdaten zu übertragen. Das Schema ist auf Abbildung 4.2 auf Seite 40 dargestellt. Die verwendeten AD/DA-Wandler dieses Codecs arbeiten nach dem Delta-Sigma-Prinzip.[tref]
Der Codec transformiert[1] aus dem kontinuierliche System ein zeitdiskretes System. Hier

[1]Nicht gleichzusetzen mit der Transformation von Zeit- in Frequenzbereich und umgekehrt.

wird bestätigt, dass beide Systeme linear und zeitinvariant sind, denn das Signal passiert beide Systeme, wobei das zeitdiskrete eingebettet im kontinuierlichen liegt (Abbildung 4.3 auf Seite 40). Man kann demzufolge festhalten, dass LTI-Systeme in diesem Fall kombinierbar sind und ihre Bedingungen hinsichtlich der Zeitinvarianz und der Linearität erfüllen.

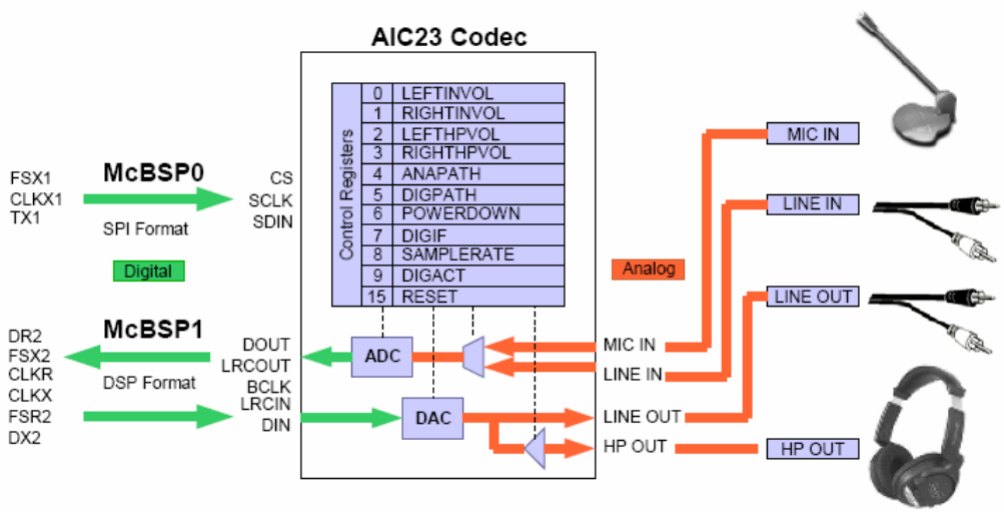


Abbildung 4.2: Aufbau des *TLV320AIC23*
Bild: [52lab]

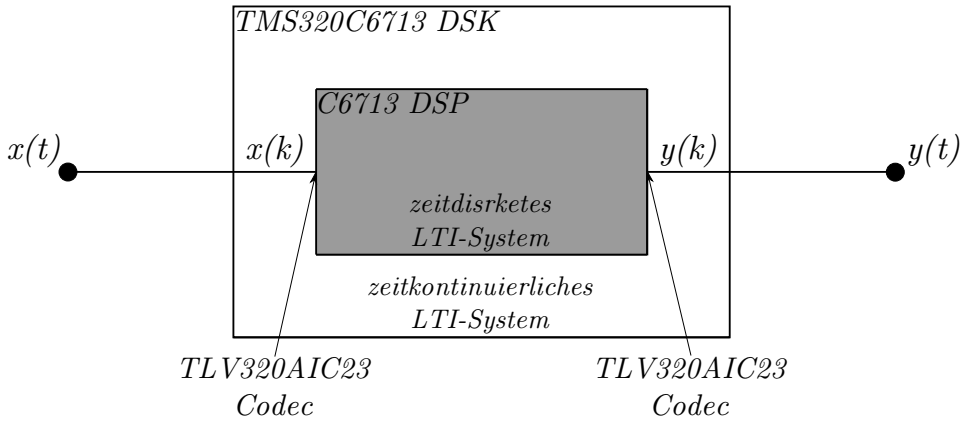


Abbildung 4.3: Schachtelung von Systemen
Bild: eigene Darstellung

4.3 Software

Wie bereits erwähnt wird mit dreierlei Software gearbeitet. Zum einen das Code Composer Studio v3.1, welches sowohl die Schnittstelle zwischen Board und PC, als auch zwischen Benutzer und Board, darstellt. Es bietet eine komfortable Bedienoberfläche, die einem handelsüblichen Programmiereditor gleicht. Dieses Programm unterstützt die Übertragung von dem Programmcode auf das Board, übernimmt Compiler- und Linkerarbeiten und

ermöglicht das aktuelle im Speicher liegende Programm zu starten oder anzuhalten. Außerdem lässt sich mit dieser Software der Debugging-Vorgang durchführen. Dazu zählt unter anderem die Messung an Takten, die eine bestimmte Funktion für ihre Ausführung benötigt.
MATLAB stellt ein mächtiges Rechenprogramm dar, welches obendrein eine immense Anzahl an Signal verarbeitenden Funktion beinhaltet. Hinzukommend können mittels eines Subprogramms, namens *fdatool*, Filter komfortabel und einfach designet und dimensioniert werden. Außerdem liefert MATLAB mit der Funktion *spectrogram()* die Möglichkeit, zu einem in einem Audioarray gespeicherten Signal, das spektrografische Diagramm berechnen zu lassen. Intern wird dies mit der FFT umgesetzt.[mathh] Auch zu Testfunktionen ist MATLAB eine große Hilfe, da Kommandobefehle wie *filter()* schnell und zuverlässig zwei Zahlenfolgen mit einander faltet.
Außerdem wurde zu Testzwecken das Audiobearbeitungsprogramm Audacity benutzt. Hier lassen sich Aufnahmen und die dazugehörige Frequenzanalyse noch einfacher und visuell ansprechender als mit MATLAB anzeigen. Ein weiterer Vorteil dieses Programms ist, dass man die Projektabtastfrequenz einstellen kann und somit alles auch mit 16000 Hz aufgenommen werden konnte. Einziger Nachteil ist, dass es zum Spektrogramm keine Farblegende bereit stellt, sodass sich nur intuitiv vermuten lässt, welche Intensitätsunterschiede zwischen den Frequenzen herrscht. Trotz alledem wurden die Spektrogramme aus dieser Software, für die Darstellung der Funktionsfähigkeiten der Filter benutzt.

4.3.1 *stand alone*-Funktion

Da die Black-Box auch im Vorlesungsbetrieb eingesetzt werden soll, ist es notwendig das Bootverhalten des Boards so zu konfigurieren, dass nach dem Anschluss der Stromversorgung direkt mit der Applikation gestartet werden kann.
Zunächst muss man jedoch wissen, was bei einem Reset des Boards passiert. Ein Hardware-Reset tritt durch das Betätigen des Reset-Tasters oder durch Trennen der Stromversorgung auf. Als erstes werden die *Configuration-Switches* des Boards gelesen. Diese Befinden sich direkt neben dem Reset-Taster. Danach werden alle Register auf einen Defaultwert gesetzt, der vom Entwickler festgelegt werden kann. Falls das *TMS320C6713* mit Peripheriegeräten verbunden ist, werden diese ebenfalls einem Reset unterzogen. Außerdem werden die globalen Interrupts ausgeschaltet. Danach kopiert der EDMA (*enhanced direct memory access*, deutsch *erweiterter Direktspeicher-Zugriff*) benötigte Daten von langsamen, nicht flüchtigen Speichern, zu schnellen flüchtigen Speichern. Dafür kann sogar Speicher über die HPI-Schnittstelle (*host port interface*[2]) in Frage kommen. Um diese Option zu aktivieren, nutzt man die *Configuration-Switches* 2 und 3 des Boards. Ist ihre Stellung beispielsweise 00, so bootet der Prozessor über den EMIF (*external memory interface*, deutsch *externe Speicher Schnittstelle*) vom Flash-Speicher.
Prinzipiell startet der *C6713*-Prozessor immer von der Speicheradresse $0x0$. Von dort ausgehend kann 1 $kByte$ vom $CE1$-Speicherbereich an diese Stelle kopiert werden. Benötigt

[2]Ein HPI ist ein paralleler Bus über welchen ein externer Hostprozessor direkt in den Speicherbereich einer CPU zugreifen kann.

man jedoch mehr Speicher für seine Bootroutine, ist es notwendig eine zweite Routine zu definieren. Diese muss zwingend in der Assemblersprache erfolgen, da zu diesem Zeitpunkt noch keine C-Umgebung initialisiert ist.
Sind alle erforderlichen Daten an der richtigen Speicheradresse, so ruft der DSP die von Texas Instruments™ bereit gestellte Funktion *_c_int*00() auf, welche die eben angesprochene C-Umgebung einbindet. Danach wird die *main*() des Programms aufrufen.[kaern]
Um die erforderlichen Daten auf den Flash-Speicher zu kopieren, wird das FlashBurn-Tool von Texas Instruments™ genutzt. Da dieser Speicher ausschließlich Daten im Hex-Format nutzt, müssen diese mit dem Code Composer Studio Plug-In hex6x umgewandelt werden.

4.4 Diskussion zur Umsetzung

4.4.1 Verwendeter Algorithmus

Für die Umsetzung der Aufgabe wurde die Implementierung der Zeitbereichsfaltung ausgewählt. Aus der theoretischen Arbeit geht hervor, dass der Grundalgorithmus für diese Faltung aus der Formel 2.34 auf Seite 16 abzulesen ist. Diese lautet wie folgt:

$$x(k) * h(k) = \sum_{n=-\infty}^{\infty} x(n)h(k-n) = y(k).$$

Der Vorteil bezüglich der Entwicklung dieser Umsetzung ist, dass es keinerlei Transformation in Frequenzbereich bedarf. Die eingehenden Abtastwerte werden vom Codec in einem Array gesichert und die berechneten Ausgangswerte in den Codec geschrieben.
Die oben gezeigte Formel wurde mittels einer *for*-Schleife programmiert.

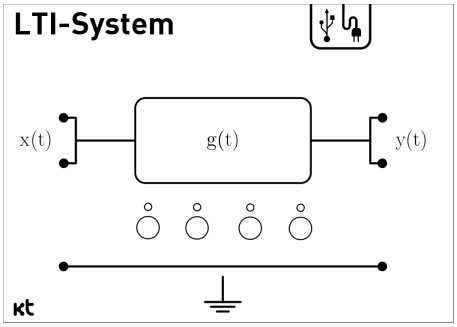
4.5 Dokumentation

4.5.1 Physikalischer Aufbau

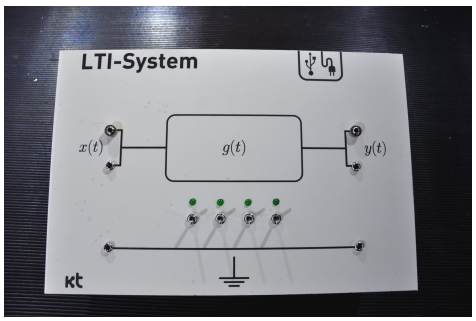
Für die Black-Box ist eine Laborkiste mit abnehmbaren Deckel zum Einsatz gekommen. Das DSP-Board wurde mit Abstandsbolzen an den Deckel geklebt. Die Bolzen selbst, sind an das Board geschraubt. Für die Versorgung mit Strom ist die Wahl auf das mitgelieferte Spannungsversorgungsgerät gefallen. Für den Anschluss wurde dafür ein Loch in die Box gebohrt (siehe Abbildung 4.4b auf Seite 43). Genauso verhält es sich mit der USB-Anbindung.
Um mit den Eingangsklemmen das Eingangssignal einzuspeisen beziehungsweise das Ausgangssignal abzugreifen, wurden die entsprechenden $3,5\,mm$-Klinkebuchsen des Boards mit einem einseitigen Klinkestecker versorgt, welche es erlauben die Signalleitungen zu trennen und deren Enden sauber auf die Box-Oberfläche zu montieren. Diese Kontakte können, mit Laborsteckern[3] oder mit $3,5\,mm$-Klinkesteckern, jeweils am Rande des Moduls abgegriffen werden. Durch den auf dem Board integrierten Codec-Baustein, können die Signale sowohl analog ein-, als auch ausgelesen werden.

[3] Umgangssprachlich ist hier auch von Bananen- oder Hirschmannsteckern die Rede.

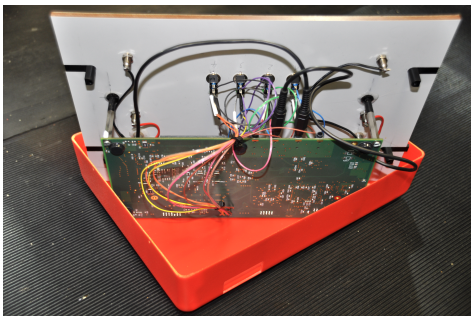
Um die verschieden Filtermodi auszuwählen, fiel das Hauptaugenmerk auf die vorhandenen Schalter des Boards. Es sind vier Stück an der Zahl und sie werden in der Dokumentation und im Quellcode auch als *Dips* (deutsch *Neigung*) bezeichnet. Da es möglich sein soll, mehr als nur vier Möglichkeiten auswählen zu können, wurden diese Schalter als Binärzahl betrachtet. Mit Ein-Aus-Kippschaltern aus dem Elektronikhandel, wurden die Kontakte durch Lötarbeiten überbrückt, um sie ebenfalls komfortabel auf der Box-Oberfläche zu montieren. Die Spannungen der LEDs wurden parallel abgegriffen. Die Leuchtdioden besitzen aus elektrotechnischen Gründen jeweils einen Vorwiderstand. Die Lötarbeiten mussten mit höchster Präzision erfolgen.
Das Layout der Box-Oberfläche kann in Abbildung 4.4a auf Seite 43 betrachtet werden, wobei die großen Kreise Platzhalter für die Schalter darstellen sollen und die kleinen den Ort der LEDs offenbaren. An den schwarz gefärbten Punkten werden die Buchsen eingelassen, wobei unten die Labor- und oben die Klinkebuchsen platziert sind. Um die Benutzeroberfläche auf die Box zu projizieren, dient eine Klebefolie, die mit einem Laserdrucker bedruckbar ist.
Das letztendliche Ergebnis der Hardware-Arbeit ist in den folgenden Abbildungen (4.4) dargestellt.



(a) Layout der Box-Oberfläche



(b) Box-Deckel



(c) Innenansicht

(d) Anschlussausparung

Abbildung 4.4: Bilder zur Black-Box
Abbildungen:Eigene Darstellungen

4.5.2 Funktionsweise des Programms

Beschreibung

Beim Starten des Programms werden die benötigten Buffer, Pointer und Variablen angelegt. Danach werden die Einstellungen für die Codec-Initialisierung gesetzt und der Codec gestartet. Es folgt eine Säuberung[4] der vorher angelegten Buffer, sowie das Anschalten der Interrupts. Danach wartet das Programm auf eingehende Daten. Das Programm kann vom Codec digitale Eingangswerte lesen und digitale Ausgangswerte an den Codec schreiben. Das Programm arbeitet interrupt-basiert. Ein Interrupt wird ausgelöst, sobald der Codec Eingangsdaten erhält. Ist dies der Fall so wird eine Interrupt-Service-Routine (ISR) ausgeführt. Diese liest die neuen Eingangsdaten und speichert sie in einem Buffer, währenddessen Daten vom Ausgangsbuffer auf den Output-Stream des Codecs geschrieben werden. Zwischen den eingehenden Daten, wenn das Programm nicht mit I/O[5] beschäftigt ist, führt es auf parallel zu den eben beschriebenen Buffern, die Filterfunktion aus und schreibt die Ergebnisse auf den Ergebnisbuffer. Dies kann ohne Veränderungen der Daten, und zwar, wenn kein Schalter betätigt wurde, passieren. Man nennt diesen Vorgang einfach Durchlass. Drückt man einen oder mehrere Schalter, so führt das Programm eine Faltung mit den Eingangsdaten und der gewählten Filter-Koeffizienten aus und schreibt es in einen Ausgangsbuffer. Wie das Handling der Buffer genau organisiert ist, wird im nächsten Unterkapitel (*Ping-Pong*-Buffer) beschrieben.
Die Schalterstellungen werden, da die *main()*-Funktion eine *while(1)*-Schleife enthält, kontinuierlich abgerufen und die dazugehörigen LEDs angeschalten.

Ping-Pong-Buffer

Um die Performanz des Programms zu steigern, wurde für die Speicherorganisation ein so genannter *Ping-Pong*-Buffer implementiert. Dieser ist eine Form des Multi-Buffering und ermöglicht die Parallelisierung der Berechnungen und der Schreibe- und Leseprozeduren des Ein- und Ausgangssignales.
Dabei wird, während auf dem einem Bufferpaar (werden von den Zeigern LEFT und AIC organisiert) Eingangsdaten gelesen beziehungsweise Ausgangsdaten geschrieben werden, auf dem anderen Bufferpaar (werden von den Zeigern FILLED und FIR organisiert) bereits aufgenommene Daten mit der $fir()$-Funktion bearbeitet und auf dem anderen Buffer dieses Bufferpaares geschrieben. Ist der Buffer bis zur letzten Stelle mit neuen Eingangswerten gefüllt, beziehungsweise wurden alle Ausgangsdaten vom Codec gelesen, erfolgt ein Bufferwechsel.
Nun können die Daten, die erst noch Eingangsdaten für die Berechnung darstellten, überschrieben und die Ergebnisse vom Codec gelesen werden. Parallel können, mit den eben eingelesenen Daten, neue Berechnungen durchgeführt und auf den Buffer abgelegt werden, der grade noch Daten für den Codec-Schreibvorgang beinhaltete.

[4] Dabei wird in jede Array-Stelle eine 0 geschrieben.
[5] Damit sind jegliche Prozesse gemeint, die das Lesen von Input und das schreiben von Output betreffen.

Um den Wechsel der Buffer nahtlos von statten gehen zu lassen, bedarf es einer Überlappung, da sonst bei jedem Bufferwechsel Schnitte zu hören wären. Dies passiert mittels eines Kopiervorganges, der die zuletzt benutzten Werte des alten Rechenbuffers (der Bereich von $SIZE$ bis $(SIZE + (TAPS - 1))$, mit einer Länge von $(TAPS - 1)$) an den Anfang (der Bereich von 0 bis $(TAPS - 1)$) des neuen Rechenbuffers dupliziert. Diese Überlappungsmethode folgt dem *overlap-save*-Verfahren. Abbildung 4.5 auf Seite 45 soll einen Überblick über die Organisation der Buffer geben, wobei hier der Zustand direkt nach der Initialisierung dargeboten wird.

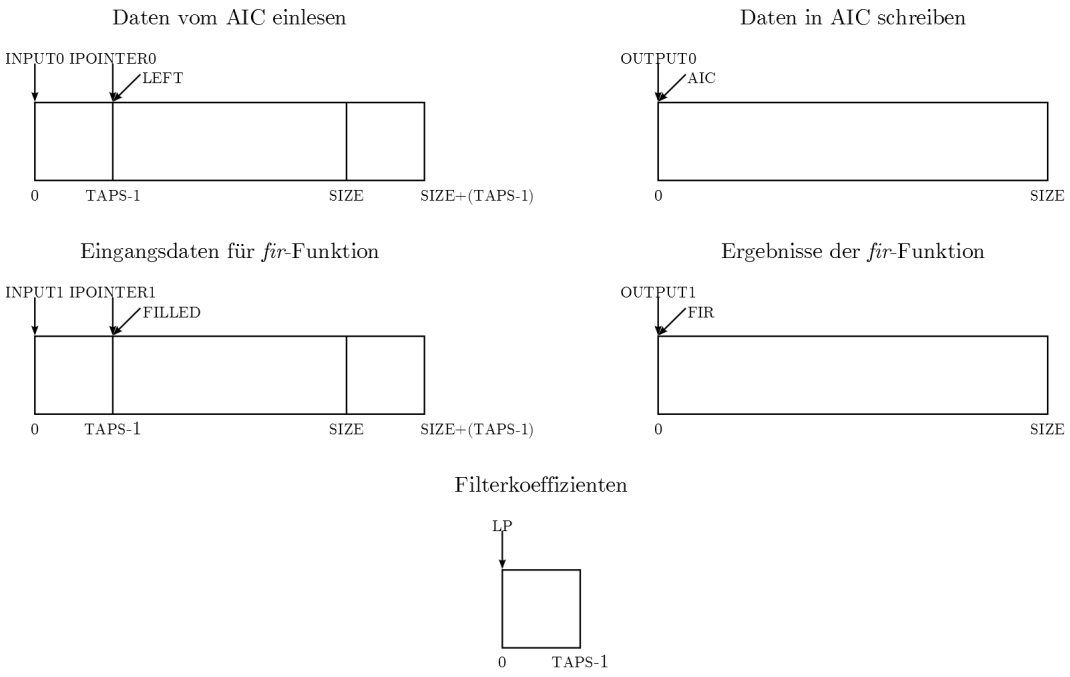


Abbildung 4.5: Buffer-Übersicht
Bild: eigene Darstellung

4.5.3 Quellcodedokumentation

Hier werden grundlegende Funktionen der Implementierung vorgestellt. Der komplette Quellcode, sowie eine Anleitung zum verbinden des Boards mit dem PC und der dazugehörigen Übertragung des Programms, befinden sich auf der beigelegten CD.

Initialisierungsparameter in *config*

Die in dieser Variable gesetzten Parameter konfigurieren mittels des *McBSP0*-Registers den Codec. Unter anderem ist hier einstellbar mit welcher Abtastrate der Codec betreiben werden soll oder ob die *MIC IN*- oder der *LINE IN*-Buchse als Schnittstelle beziehungsweise Aufnahmequelle der Audiodaten dienen soll. Die einzelnen Einstellungen werden mit Zahlen im Heximalsystem übergeben, wobei diese sich wiederum aus Binärzahlen zusammen setzen. Listing 4.1 zeigt die Parametereinstellung der Implementierung.

```
0x0011, /* Set-Up Reg 4        Analog audio path control */
        /* X          0        reserved */
        /* STA        00       sidetone attenuation: -6 dB */
        /* STE        0        sidetone: disabled */
        /* DAC        1        DAC: selected */
        /* BYP        0        bypass: off */
        /* INSEL      0        input select for ADC: line */
        /* MICM       0        microphone mute: disabled */
        /* MICB       1        microphone boost: enabled */
```

Listing 4.1: Beispiel Parameter Codec

void calc_none()

Ist kein Schalter betätigt, so lässt das Programm die eingehenden Samples ohne Bearbeitung passieren, dafür werden die Werte, die im Rechenbuffer liegen, ohne Modifikation in den Ergebnisbuffer kopiert.

void calc_echo

Hier wird das simulierte Echo erzeugt. Dafür werden dem Ergebnisbuffer Samples aus dem Rechenbuffer übergeben und ein Abbild dessen gedämpft dazu addiert.

void calc_LP()

In dieser Funktion wird die *overlap-save*-Methode implementiert. Dabei werden die letzten (TAPS-1) Werte des Arrays, der vor dem Bufferwechsel noch als Rechenbuffer agierte, an den Anfang des neuen Rechenbuffers kopiert und bilden deren Startsamples. Danach wird die $fir(in, w, out, TAPS, SIZE)$-Funktion mit den entsprechenden Koeffizienten aufgerufen. Speziell bei dieser Funktion werden die Filterkoeffizienten des Tiefpasses genutzt.

void fir(float *in, float *w, float *out, int ntap, int size)

In dieser Funktion wird der eigentliche Faltungsprozess durchgeführt. Dabei werden fünf Werte übergeben:

- float *in: Pointer auf den Buffer mit Eingangssignal-Werten
- float *w: Pointer auf den Buffer mit Koeffizienten
- float *out: Pointer auf den Buffer für Ausgangswerte
- int ntap: Länge der Impulsantwort (in Koeffizienten)
- int size: Anzahl der auf einem Buffer gespeicherten Eingangswerten

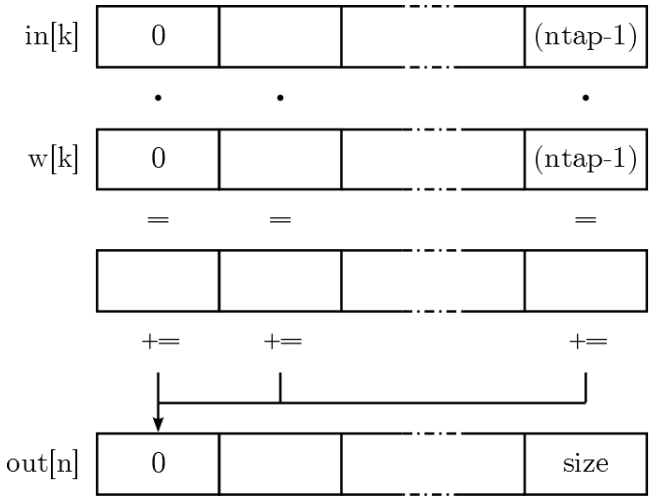


Abbildung 4.6: Visualisierung der inneren *for*-Schleife der $fir()$-Funktion
Bild: eigene Darstellung

Dabei besteht diese Funktion aus zwei *for*-Schleifen. Der Code der Funktion ist in Listing 4.2 zu sehen. Um diese Funktion besser zu verstehen, betrachtet man Abbildung 4.6 auf Seite 47. Im Prinzip wird jeder im Buffer liegende Wert, mit dem zugehörigen Koeffizient multipliziert und alle Ergebnisse der Multiplikationen zusammenaddiert und im Ausgangsarray gesichert. Außerdem wird, mit jedem Durchlauf der äußeren Schleife, das Datenfeld der Eingangswerte um eine Stelle verschoben, sodass nach jedem innerem Durchlauf der älteste Eingangswert nicht mehr betrachtet wird und dafür der nächste Wert aus dem Eingangsbuffer eingelesen werden kann. Die Funktion hat den Aufwand $O(((TAPS^2)+1)\cdot SIZE)$.

```
void fir(float *in, float *w, float *out, int ntaps, int size)
{
    int i,n;
    for(n=0;n<size;n++)
    {
        out[n]=0;
        for(i=0;i<ntap;i++)
        {
            out[n]+=w[i]*in[n-i]
        }
    }
}
```

Listing 4.2: Faltungsoperation

void switch_buffers()

Die $switch\_buffers()$-Funktion führt den Bufferwechsel aus, nachdem der Buffer mit neuen Eingangswerten voll beschrieben wurde. Dafür werden die Zeiger der jeweiligen Threads vertauscht, abhängig davon, welcher Zeiger auf welchem Buffer grade gearbeitet hat. Die-

ser Zustand wird durch die Variable *active* überprüft, welche nur die Zustände 0 und 1 besitzt.

void AIC_RX()

Diese Funktion repräsentiert die Schnittstelle zwischen Programm und Codec. Die Samples werden mittels der Funktion $DSK6713\_AIC23\_read(hCodec, *val)$ (aus $dsk6713\_aic23.h$) gelesen, der rechte Kanal herausgezogen und in den Eingangsbuffer geschrieben. Mit der selben Zählvariable werden mittels der $DSK6713\_AIC23\_write(hCodec, val)$-Funktion (aus $dsk6713\_aic23.h$) Daten zum Ausgang des Codecs geschrieben. Dies bedeutet, dass zu jedem eingelesenen Eingangssample ein Ausgangssample geschrieben wird. Nach jedem Lese-/Schreibvorgang wird überprüft, ob die letzte Stelle der Buffer erreicht worden ist. Ist dies der Fall, wird die $switch\_buffers()$-Funktion aufgerufen.

int readDips()

In dieser Funktion werden die Schalterstellungen ausgelesen und mit einer Integer-Variable im Dezimalsystem zurück gegeben. Dies passiert innerhalb einer *for*-Schleife, die genau vier mal durchlaufen wird. Dafür wird jedem Schalter eine Wertigkeit zugeteilt, sodass Schalter 1 der ersten Binärstelle entspricht, der zweite Schalter, der zweiten Binärstelle usw. Durch die logische *oder*-Verknüpfung, verbunden mit einem Bit-Shift, werden die einzelnen Wertigkeiten zur Rückgabevariable dezimal addiert.

4.5.4 Implementierte Filter

Die Filtereigenschaften wurden so dimensioniert, dass man akustisch eine deutliche Veränderung wahrnehmen kann. Die Impulsantworten, sowie die Frequenzgänge der Filter, sind im Anhang zu finden.

Tiefpass

Ein Tiefpass-Filter meint hier ein digitales Filter, welches Frequenzen unterhalb der Grenzfrequenz fast ungedämpft passieren lässt. Dieses Filter wird auch als Höhensperre, High-Cut-Filter oder Rauschfilter bezeichnet. Anwendungsgebiete des digitalen Tiefpassfilters sind unter Anderem der Anti-Aliasing-Filter, welches eben Frequenzen, die höher als die halbe Abtastfrequenz sind, unterdrückt oder ein Rauschfilter, der Rauschen unterdrücken soll[6]. Das Schaltzeichen des Tiefpassfilters ist in Abbildung 4.7a auf Seite 50 zu sehen.
Der in der Arbeit implementierte Tiefpass-Filter hat folgende Eigenschaften:

- Durchlassbereich: $0\,Hz$ bis $900\,Hz$
- Übergangsbereich: $900\,Hz$ bis $1000\,Hz$
- Sperrbereich: $1000\,Hz$ bis $8000\,Hz$
- Ordnung: 262

[6] Unter Rauschen versteht man in der DSV ein Signal, welches nahezu alle Frequenzen enthält. Akustisch hört es sich auch dementsprechend *rauschend* an.

Hochpass

Das digitale Hochpass-Filter lässt im Gegensatz zum Tiefpass-Filter nur die Frequenzen oberhalb der Grenzfrequenz nahezu ungedämpft passieren. Wenn die Rede von einer Tiefensperre, einem Low-Cut-Filter oder einem Trittschallfilter ist, wird damit ebenfalls das Hochpass-Filter gemeint. Anwendungsgebiete dieses Filtertypen sind beispielsweise das Herausziehen von Brummstörungen. Ebenfalls werden Hochpass-Filter genutzt um nur bestimmte hohe Frequenzen für einen Hochton-Lautsprecher passieren zu lassen. Das Schaltzeichen des Hochpassfilters ist in Abbildung 4.7b auf Seite 50 zu sehen.
Die Eigenschaften des implementierten Hochpass-Filters sind:

- Durchlassbereich: 4000 Hz bis 8000 Hz
- Übergangsbereich: 3900 Hz bis 4000 Hz
- Sperrbereich: 0 Hz bis 3900 Hz
- Ordnung: 262

Bandpass

Werden Frequenzen, die innerhalb eines Frequenzbandes liegen, durchgelassen, so ist die Rede von einem Bandpass-Filter. Diese Frequenzen sind allerdings, obwohl sie passieren können, in ihrer Amplitude gedämpft. Man kann einen solchen Filter auch als Kombination von hintereinander geschalteten Hoch- und Tiefpass-Filtern realisieren. Genutzt wird dieser Filter-Typ, um beispielsweise eine Bandbegrenzung durchzuführen und damit nur einen bestimmten Frequenzbereich übertragen zu müssen. Dies passiert in der Realität auch bei einem Telefon. Die Qualität des Sprachsignals nimmt dadurch erheblich ab, jedoch ist das durchgelassene Frequenzband ausreichend für Sprache, da dieses bei Frequenzen zwischen 200 Hz und 2000 Hz liegt. Hiervon wurde schon in der Einleitung Bezug genommen. Das Schaltzeichen des Bandpassfilters ist in Abbildung 4.7c auf Seite 50 zu sehen.
Der Bandpass-Filter wurde mit folgenden Eigenschaften dimensioniert:

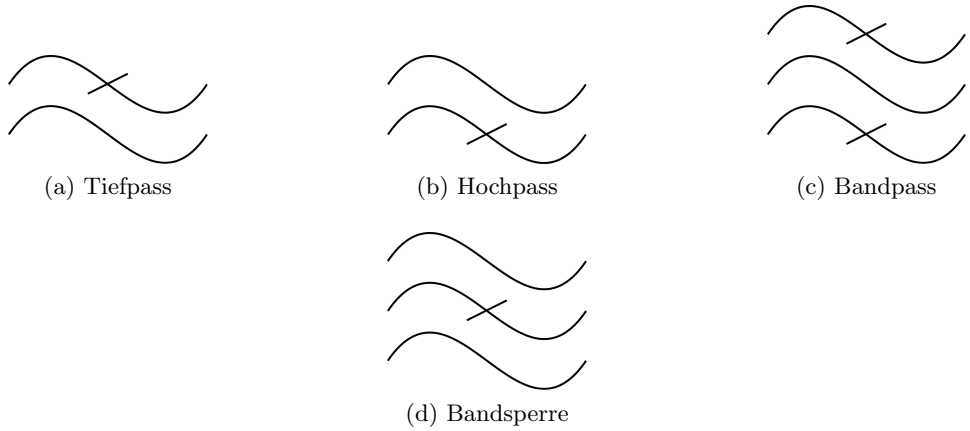
- Durchlassbereich: 1000 Hz bis 3000 Hz
- Übergangsbereich: 900 Hz bis 1000 Hz und 3000 Hz bis 3100 Hz
- Sperrbereich: 0 Hz bis 900 Hz und 3100 Hz bis 8000 Hz
- Ordnung: 262

Bandsperre

Das Bandsperr-Filter ist das Pendant zum Bandpass-Filter. Hier wird ein bestimmtes Frequenzband gesperrt, wobei die durchgelassenen Frequenzen in der Amplitude gedämpft werden. Häufig spricht man auch von einem Bandstopp-Filter, wobei sich dieser Name aus dem Englischen eingebürgert hat. Dieses Filter wird dazu eingesetzt, um Störungen eines

Signals zu unterdrücken. Dabei sollte die Störung in einem festen Frequenzband liegen und bekannt sein. Ein realitätsbezogenes Beispiel ist das herausfiltern der Frequenzen bei $50\,Hz$. Hier liegt die Frequenz der allgemeinen Netzspannung, die oft eine Störquelle darstellt. Das Schaltzeichen des Bandsperrfilters ist in Abbildung 4.7d auf Seite 50 zu sehen.
Die Größenordnungen dieses implementierten Filtertypen sind:

- Durchlassbereich: $0\,Hz$ bis $900\,Hz$ und $3100\,Hz$ bis $8000\,Hz$
- Übergangsbereich: $900\,Hz$ bis $1000\,Hz$ und $3000\,Hz$ bis $3100\,Hz$
- Sperrbereich: $1000\,Hz$ bis $3000\,Hz$
- Ordnung: 262



(a) Tiefpass (b) Hochpass (c) Bandpass

(d) Bandsperre

Abbildung 4.7: Filtersymbole
Abbildungen:Eigene Darstellungen

Echo

Das implementierte Echo hat eine Nachhallzeit von circa $1\,s$. Dafür wird dem Eingangsblock ein um die Hälfte gedämpfter Ausgangsblock addiert. Visuell kann dies auf dem Spektrogramm betrachtet werden. Dies befindet sich in Abbildung A11 auf Seite X im Anhang. Es ist zu erkennen, dass die Laute nach und nach abgeschwächt werden. Mit einem größeren Dämpfungsfaktor kann dieses Ausblenden in die Länge gezogen werden. Mit einer größeren Blocklänge verlängert sich die Zeit der Verzögerung zwischen Originalton und erstem Echo.

4.5.5 Tests

Um die implementierten Filter zu testen, werden die Spektren der Testtöne/-laute vor und nach dem Filtern verglichen. Dabei wurden die Filter jeweils mit $500\,Hz$, $2000\,Hz$, $4000\,Hz$ und einer *Wave*-Datei gespeist. Die Töne werden mittels einer Android-App namens *TestTone* generiert. In der *Wave*-Datei wurde das Wort *Test* aufgenommen. Die

Echofunktion wurde ausschließlich mit dieser Datei getestet, da eine zeitliche Verzögerung eines immer gleich bleibenden Tones keinen hörbaren Effekt mit sich zieht.

Ergebnisse

Die Testergebnisse sind auf den Abbildungen A7 bis A10 auf den Seiten VI bis IX im Anhang zu sehen. Anfangs müssen jedoch die Originalaufnahmen als Referenz betrachtet werden. Diese sind in Abbildung A6 auf Seite V im Anhang zu sehen. Die Spektrogramme sind dreidimensional, da man dort Zeit (Abszisse), Frequenz (Ordinate) und Intensität (Färbung) ablesen kann. Dabei erstreckt sich die Farbskala von grau (keine Intensität) bis weiß (maximal Intensität) über blau und rot[7]. Es ist deutlich an den Spektrogrammen zu erkennen, dass der Durchlassbetrieb ohne Einschränkungen funktioniert.
Die Erwartungen an die Filter sind, dass sie die jeweiligen Töne, die nicht in ihrem Durchlassbereich liegen, unterdrücken. Visuell ist dies auf dem Spektrogramm klar zu erkennen, da die Frequenzen, die nicht passieren dürfen, nach dem Filtern eine kältere Färbung gegenüber des Originalspektrums aufweisen.
Die gewünschten Anforderungen wurden erfüllt. Die reinen Sinustöne wurden, den Filter betreffend, in ihrer Intensität - in der Amplitude - gedämpft oder durchgelassen. Besonders gut ist das Ergebnis bei dem Spektrogrammen des gesprochenen Wortes zu erkennen.
Außerdem ist neben den grafischen Tests, auch akustisch auf Korrektheit überprüft worden. Auch hier wurden alle Erwartungen erfüllt.

4.5.6 Probleme

Ein Problem in der Umsetzung stellte die Länge der Impulsantwort der Kirche dar. Diese hat eine Dauer von $6\,s$, bei einer Abtastrate von $44,1\,kHz$. Daraus ergeben sich $6\,s \cdot 44100\,kHz = 2646000\,samples$. Nimmt man an, dass diese vom Typ *float* (entspricht $32\,Bit = 4\,Byte$) sind, so ergibt sich eine Dateigröße von rund $1,1\,MByte$. Diese übersteigt den IRAM des Boards. Alternativ wurde eine kürze Impulsantwort angeboten. Außerdem muss man, um das Abtasttheorem nicht zu verletzen, die Impulsantwort *downsamplen*. Dies bedeutet, dass die Abtastrate auf $16\,kHz$ vermindert wird, womit sich die Dateigröße auf $384\,kByte$ verringert.
Es besteht das Problem, dass die for-Schleife der $fir()$-Funktion einen erheblichen Rechenaufwand aufweist. Mit dem Code Composer Studio v3.1 wurde eine Taktzählung für diese Schleife mit 262 Koeffizienten durchgeführt. Daraus ergab sich, dass für einen Schleifendurchgang circa 12000 Takte benötigt werden. Bei einer größeren Koeffizientenanzahl, dauert der Rechenvorgang länger als die Aufnahme der Samples. Dadurch kommt das Ausgangssignal ins Stocken. Auch werden neben der Schleife noch andere Operationen wie das Lesen der Schalter, das Setzen der LEDs, das Kopieren der Daten oder das Aufrufen von Funktionen ausgeführt. Hier stößt der Signalprozessor an seine Grenzen.

[7] Es ergibt sich das Schema von tief zu hoch (grau-blau-rot-weiß)

5 Zusammenfassung & Aussichten

Nachdem ich mich ausgiebig mit dieser Thematik beschäftigt habe, komme ich zu folgenden Schlüssen:
Digitale Signalverarbeitung ist einer der aufsteigensten Äste in der Entwicklung von elektrischen Systemen. Sie löst heute mehr und mehr analoge Lösungsmöglichkeiten ab. Mit digitaler Signalverarbeitung lassen sich komplexe Probleme lösen, die viel Rechenaufwand benötigen. Es existiert ein großes Interesse für diese Thematik und der Trend dazu ist steigend. Das Experimentieren damit, beschäftigt viele sogar in ihrer Freizeit. Die erste Hürde für den Einstieg ist dabei die Einarbeitung in die theoretischen Grundlagen. Die zweite Hürde stellt aus Kostengründen die Anschaffung der Hardware dar. Infolgedessen sind kostengünstigere Mikrocontroller meist die erste Wahl.

Die ersten drei Kapitel bilden eine nützliche Zusammenfassung zu diesem Themengebiet. Das Endergebnis kann praxisbegleitend für den Vorlesungsbetrieb genutzt werden.
Der theoretische Teil dieser Arbeit umschließt, neben den mathematischen und signalverarbeitenden Ausführungen, einen umfassenden Einblick in die digitale Signalverarbeitung. Außerdem wurde das Thema der Umwandlung von analogen zu digitalen Signalen ausführlich erklärt.
Ein LTI-System konnte erfolgreich auf dem DSP programmiert werden. Der physikalische Aufbau ist funktional umgesetzt und visuell ansprechend. Zwischen dem Eingangs- und dem Ausgangssignal liegt keine hörbare Latenz vor, sodass die implementierten Filter repräsentativ eingesetzt werden können.

Durch folgende Erweiterungen kann die Umsetzung des Programms noch effizienter gestaltet:
Zunächst sollte überlegt werden, die Impulsantwort auf dem SDRAM zu initialisieren. Aufgrund dessen kann viel Speicherplatz auf dem schnellen IRAM eingespart werden. Es müsste jedoch überprüft werden, inwiefern die Schnelligkeit des SDRAMs zum Lesen der Koeffizienten geeignet ist. Eine weitere Möglichkeit wäre, Teilblöcke der Impulsantwort aus dem SDRAM in den IRAM zu laden. Zu Prüfen ist, ob der Kopiervorgang zu viel Rechenzeit in Anspruch nehmen würde.
Eine weitere Möglichkeit um hohe Performanz des DSPs betrifft die Designkriterien. Hier werden die Samples auf 16 *Bit* beschränkt. Der Signalprozessor besitzt jedoch 32 *Bit* Register. Man kann zwei Samples in einem Register speichern, womit man sich jeweils eine Lese- und Schreiboperation einspart. Dabei muss genauestens auf Überläufe geachtet werden, damit die Zahl, beginnend am LSB, (*last significant bit*, deutsch *Bit mit niedrigstem Stellenwert*) keine Behinderung der zweiten, im Register liegenden Zahl darstellt.

Außerdem müssen unbedingt alle verfügbaren Rechenwerke des Prozessors vollständig ausgenutzt werden. Damit kann eine echte Parallelität der Rechnungen erfolgen. Dies zieht eine enorme Zeiteinsparung mit sich. Wichtig hierbei ist, alle frei verfügbaren Register (32 32 *Bit* Register) zu verwenden, da sonst das hin und her Kopieren der Registereinträge eine Barriere für die Parallelität darstellt.
Hinzukommend kann die Performanz gesteigert werden, indem der C-Code von Hand in Assembler optimiert wird. So wird sichergestellt, dass prozessoroptimierte Operationen, wie beispielsweise die MAC-Operation, genutzt werden. Dies erfordert allerdings einen hohen Grad an Programmiererfahrung und kostet zusätzlich Entwicklungszeit.
Die optimale Implementierung wäre hier über die schnelle Faltung gegeben. Dabei wird das Eingangssignal in Blöcke der Länge L geteilt. Damit soll erreicht werden, dass aus einem aperiodischen Signal eine Folge von L-periodischen Teilsignalen wird. Es ist wichtig, dass die entnommenen Blöcke und die zu faltende Impulsantwort die gleiche Länge besitzen. Ist dies nicht der Fall, so muss das kürzere Signal mit Nullen verlängert werden. Dieser Vorgang wird *zero padding* bezeichnet.
Dann können beide Signale mittels der schnellen Fourier-Transformation in den Frequenzbereich transformiert werden. Hierzu kann der Algorithmus von Cooley und Tukey verwendet werden, der die Voraussetzung beansprucht, die Länge der Blöcke als eine Zweierpotenz zu wählen. Dies hat den Hintergrund, dass das Signal solange aufgeteilt wird, bis nur noch ein Sample übrig bleibt. Verletzt man die Vorgabe der Blocklänge, so geht die Rechnung nicht auf.
Nach der Transformation multipliziert man die beiden Spektren miteinander und wendet auf dem Ergebnis die inverse schnelle Fourier-Transformation an. Um die Datenblöcke im Überlappungsbereich zu korrigieren, kann eine der in der Arbeit vorgestellten Überlappungsverfahren verwendet werden.
Zusätzlich wird die schnelle Faltung noch effizienter, wenn die vorher genannten Optimierungen bezüglich Register- und Rechenwerkausnutzung angewendet werden.

Schlussendlich ist eine Grundlage für die Benutzung des *TMS320C6713* geschaffen, die die Möglichkeit bietet weitere Funktionen zu implementieren. Diesbezüglich können zusätzliche Anwendungen durch Verwendung von *Daughtercards* und anderen Peripheriegeräten geschaffen werden.

Anhang

Im Anhang befinden sich die Dokumentation zur Inbetriebnahme des Boards, sowie die in der Arbeit angesprochenen Impulsantworten, Frequenzgänge und Spektrogramme.

Inbetriebnahme des Boards

Das im Lieferumfang enthaltene Code Composer Studio v3.1 ist in der vorhandenen Version ausschließlich auf Windows XP fehlerfrei zu betreiben. Für andere Systeme wird keine Gewährleistung für die Funktion gegeben. Bevor das TMS320C6713 benutzt werden kann, müssen die nötigen Treiber und die Entwicklungsumgebung installiert werden. Dafür sind folgende Schritte nötig:
Zuerst muss die mit gelieferte Installations-CD einlegt werden. Durch Autorun erscheint Fenster von Abbildung A1.
Mit einem Klick auf *INSTALL PRODUCTS*, erscheint das Fenster, wie in Abbildung A2. Als erstes muss das Code Composer Studio installiert werden. Nach erfolgreicher Installation startet man nun die Installation der Treiber. Dafür klickt man im Fenster von Abbildung A2 auf *DSK6713 Drivers and Target Content.*

Abbildung A1: Startfenster Code Composer Studio

Nach dem auch dieser Vorgang abgeschlossen ist, kann jetzt das Board in Betrieb genommen werden. Dies ist damit verbunden die nötigen Kabel an das Board anzuschließen. Als erstes wird das USB-Kabel angesteckt. Danach erfolgt der Anschluss der Ein-

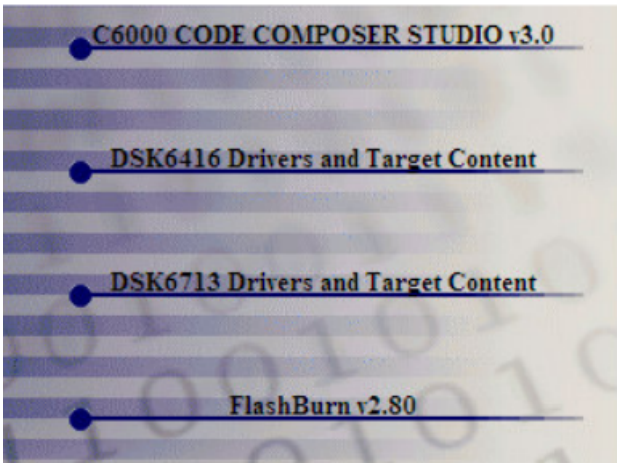


Abbildung A2: Untermenü Code Composer Studio

und Ausgabemedien an den Klinke-Buchsen. Jetzt erst wird das *TMS320C6713 DSK* mit dem Stromversorgungskabel verbunden. Es sollten einige LEDs blinken. Wenn keiner der Leuchtdioden mehr blinkt, kann das USB-Kabel mit dem Rechner verbunden werden. Nachdem Windows XP den vorher installierten Treiber gefunden hat, kann jetzt die Funktion überprüft werden. Dafür ist bei der Installation ein Desktop-Icon mit dem Namen *6713 DSK Diagnostic Utility v3.1* (zu sehen auf Abbildung A3a). Mit Klick auf diesen Button wird das Diagnose-Tool gestartet, in welchem man auf den Start-Button klicken muss. Die Überprüfung dauert etwa 30 Sekunden und es sollten alle acht Punkte grün aufleuchten.

(a) Diagnostic Utility (b) CCS v3.1

Abbildung A3: Icons Code Composer Studio

Jetzt kann mit einem Klick auf den Button vom Code Composer Studio (siehe Abbildung A3b) das Programm gestartet werden. Mit einem Klick auf *Debug* in der Befehlsleiste und wählen der Option *Connect* ist das Board jetzt mit der CCS-Umgebung verbunden. Ist ein Projekt erstellt worden, so kann mit F7 kompiliert werden. Alternativ kann unter *Project* → *Open* ein bestehendes Projekt geladen werden. Mit Strg+L öffnet sich ein Fenster, wo die *.out-Datei (meist im Debug Ordner innerhalb des Projektordners) des Projektes ausgewählt werden muss. Mit F5 wird das Programm auf dem DSP gestartet.

Impulsantworten

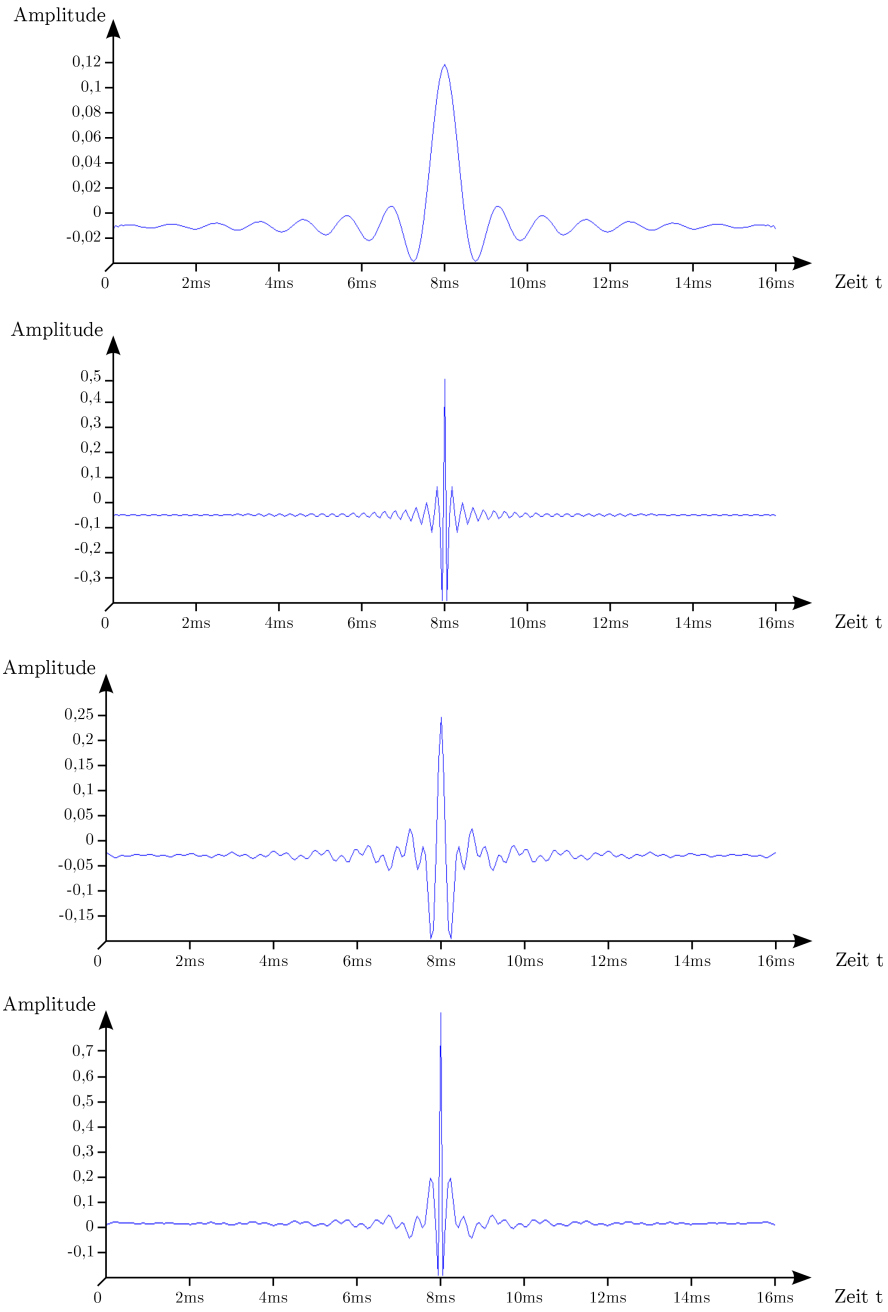


Abbildung A4: Impulsantworten
(1. Tiefpass, 2. Hochpass, 3. Bandpass, 4. Bandstopp)

Frequenzgänge

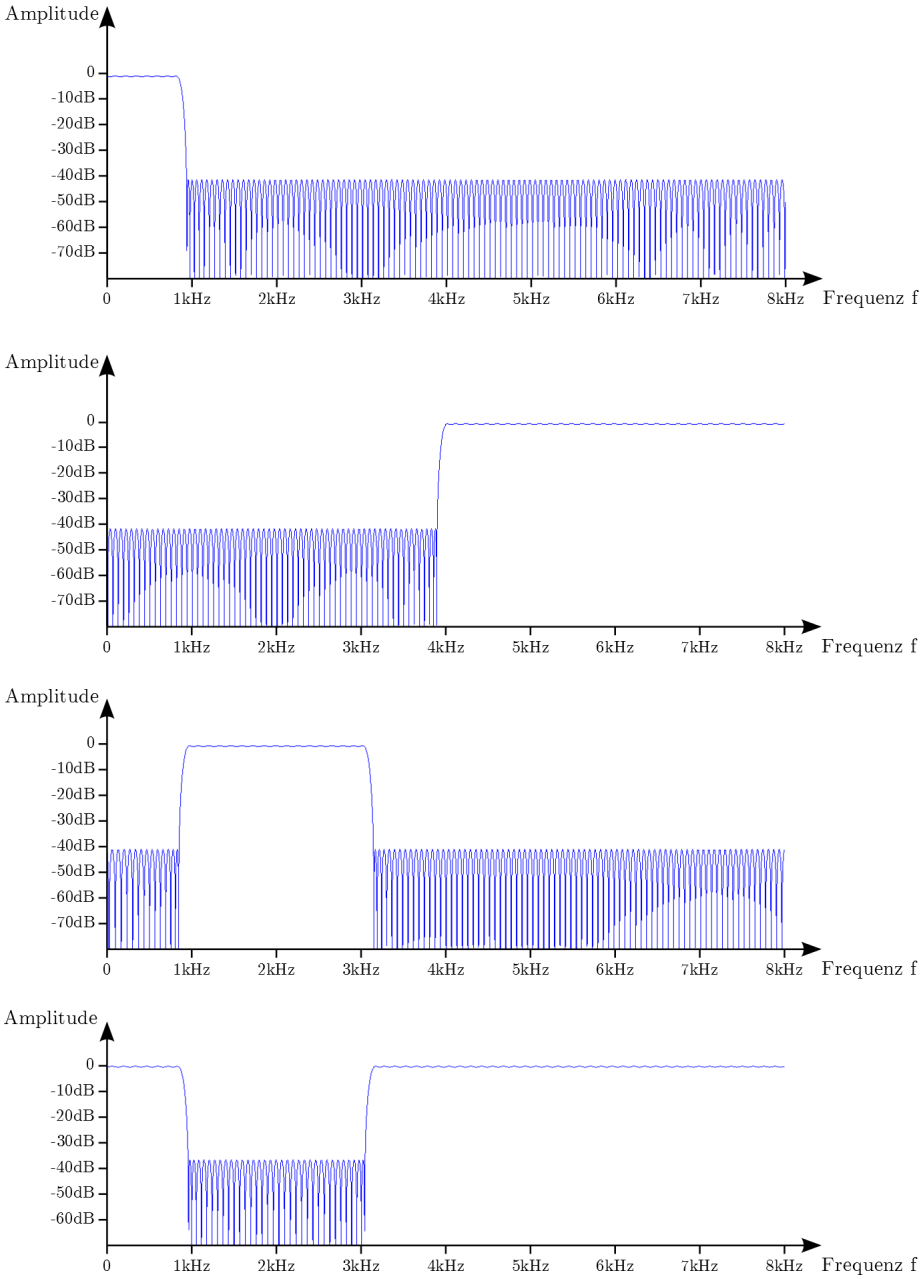


Abbildung A5: Frequenzgänge
(1. Tiefpass, 2. Hochpass, 3. Bandpass, 4. Bandstopp)

Spektrogramme

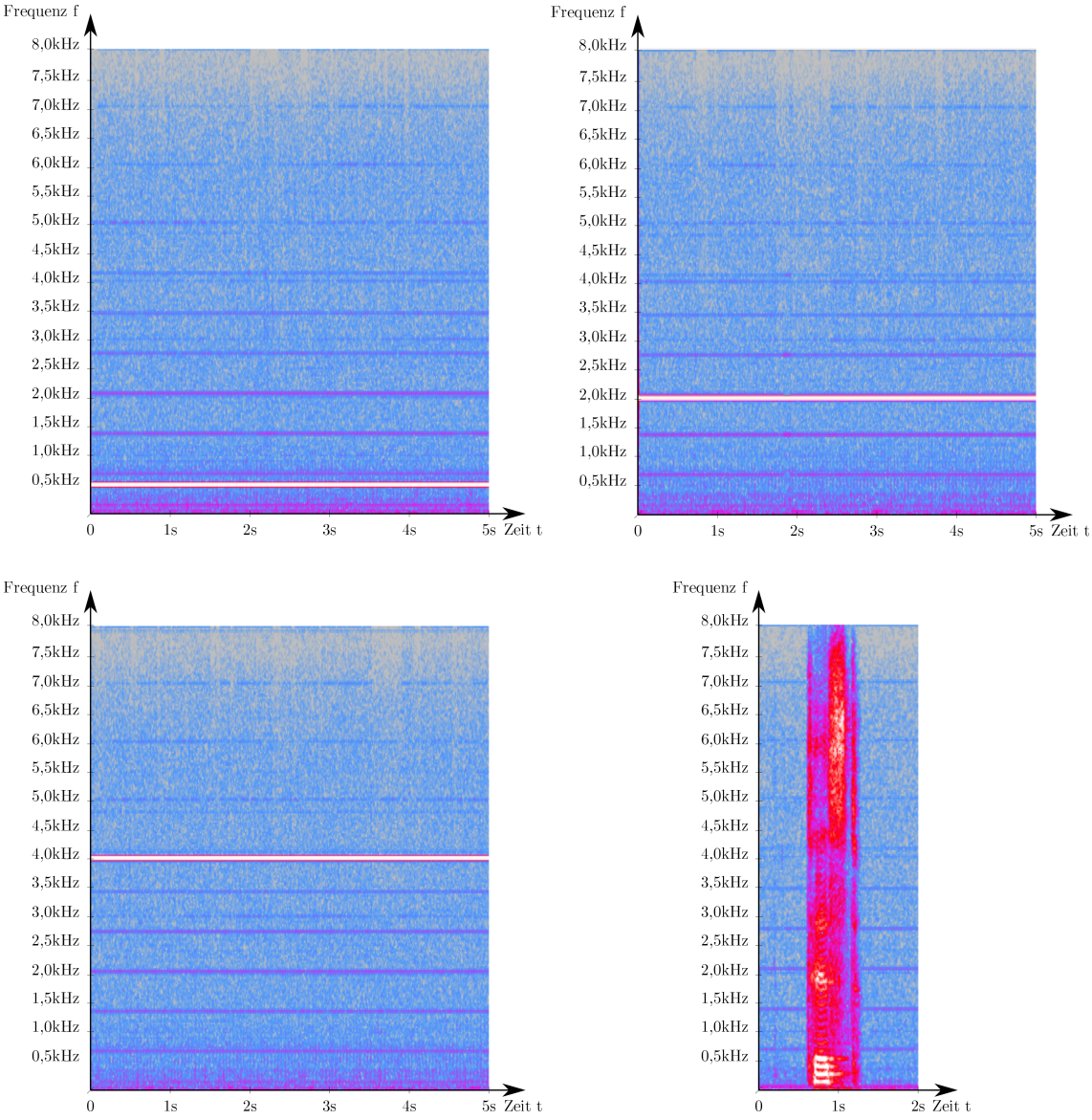


Abbildung A6: Spektrogramme ohne Filter
(links oben 500 Hz, rechts oben 2 kHz, links unten 4 kHz, rechts unten $Testlaut$)

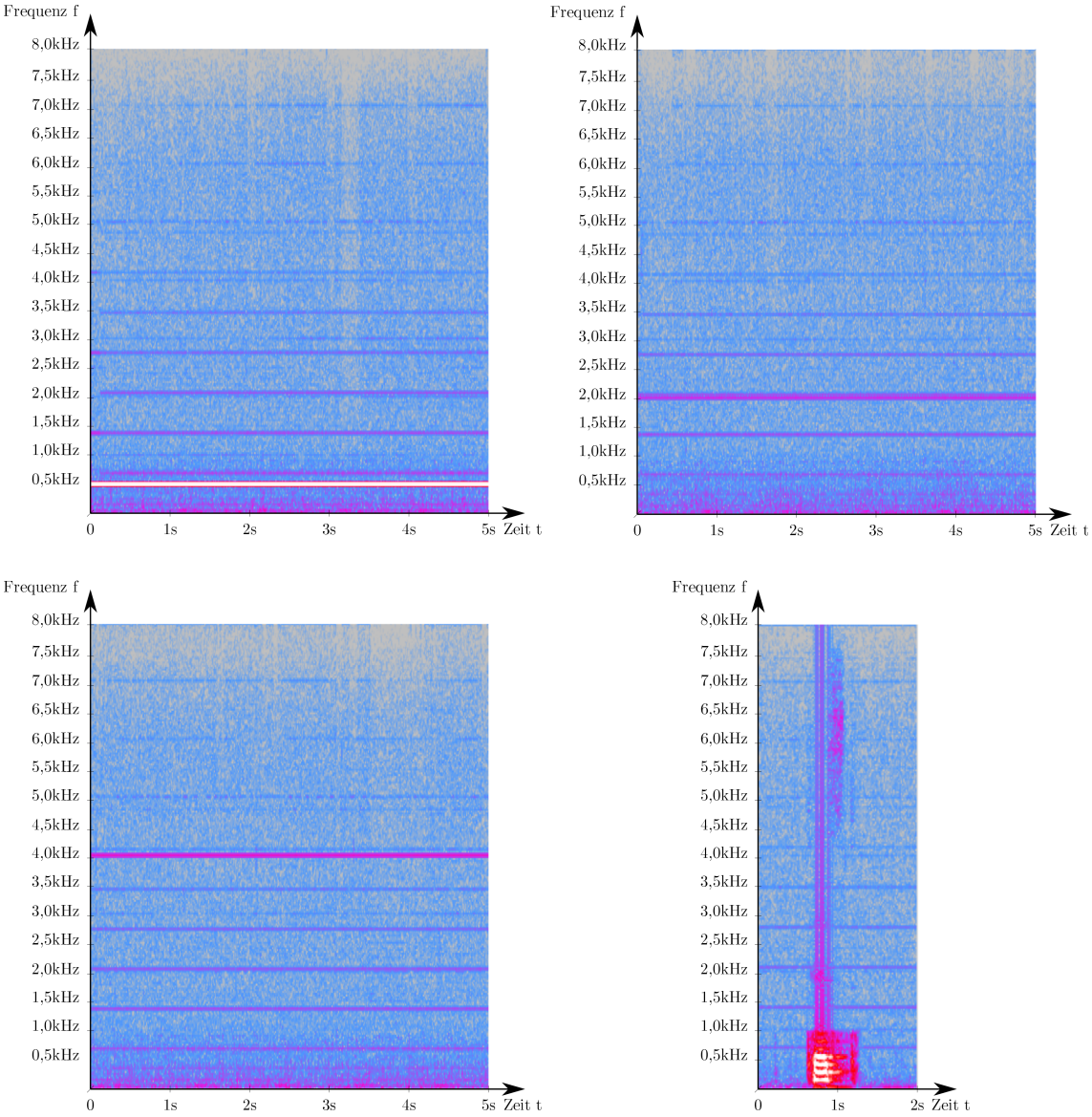


Abbildung A7: Spektrogramme Tiefpass gefiltert
(links oben 500 Hz, rechts oben 2 kHz, links unten 4 kHz, rechts unten $Testlaut$)

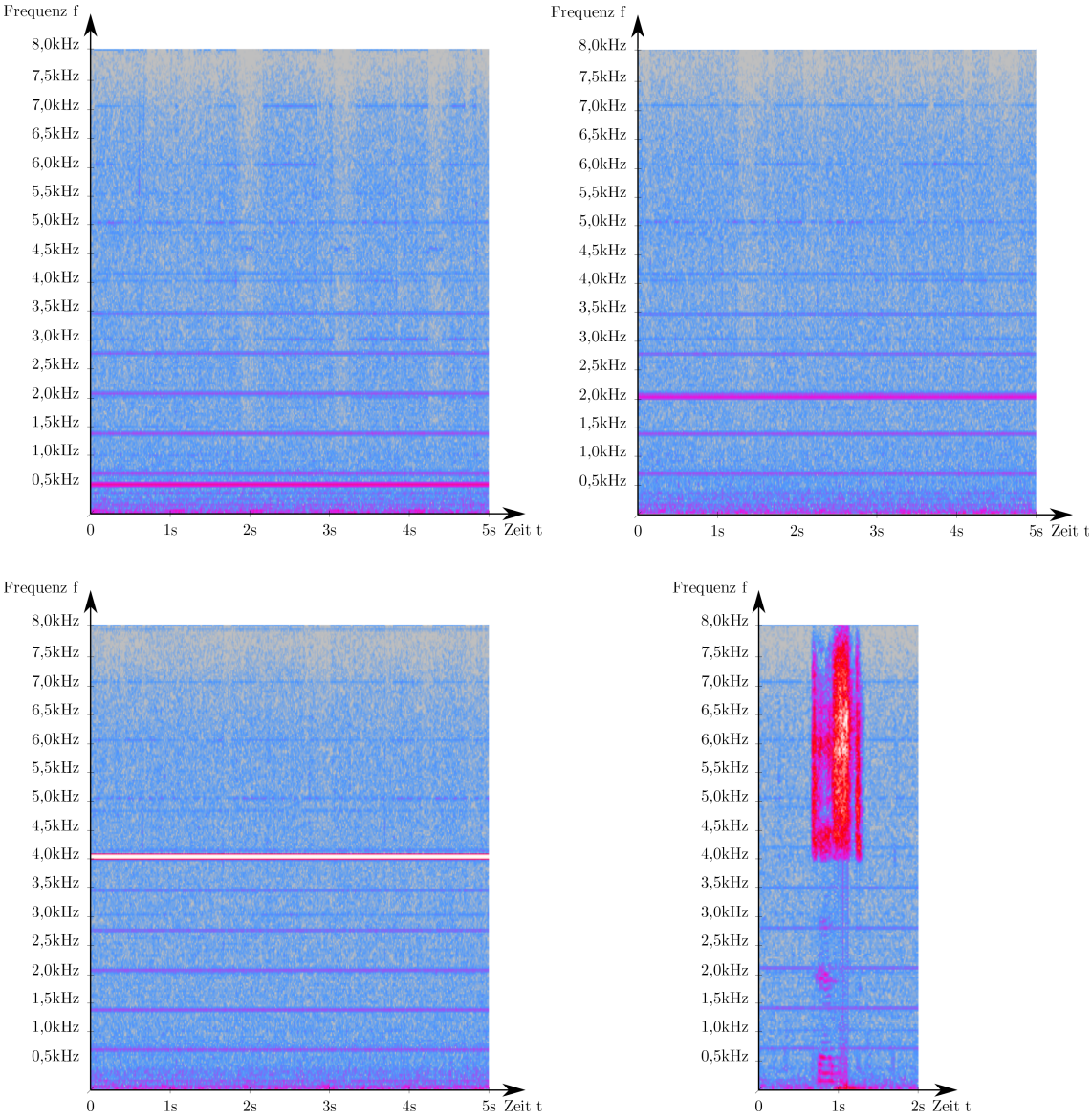


Abbildung A8: Spektrogramme Hochpass gefiltert
(links oben 500 Hz, rechts oben 2 kHz, links unten 4 kHz, rechts unten $Testlaut$)

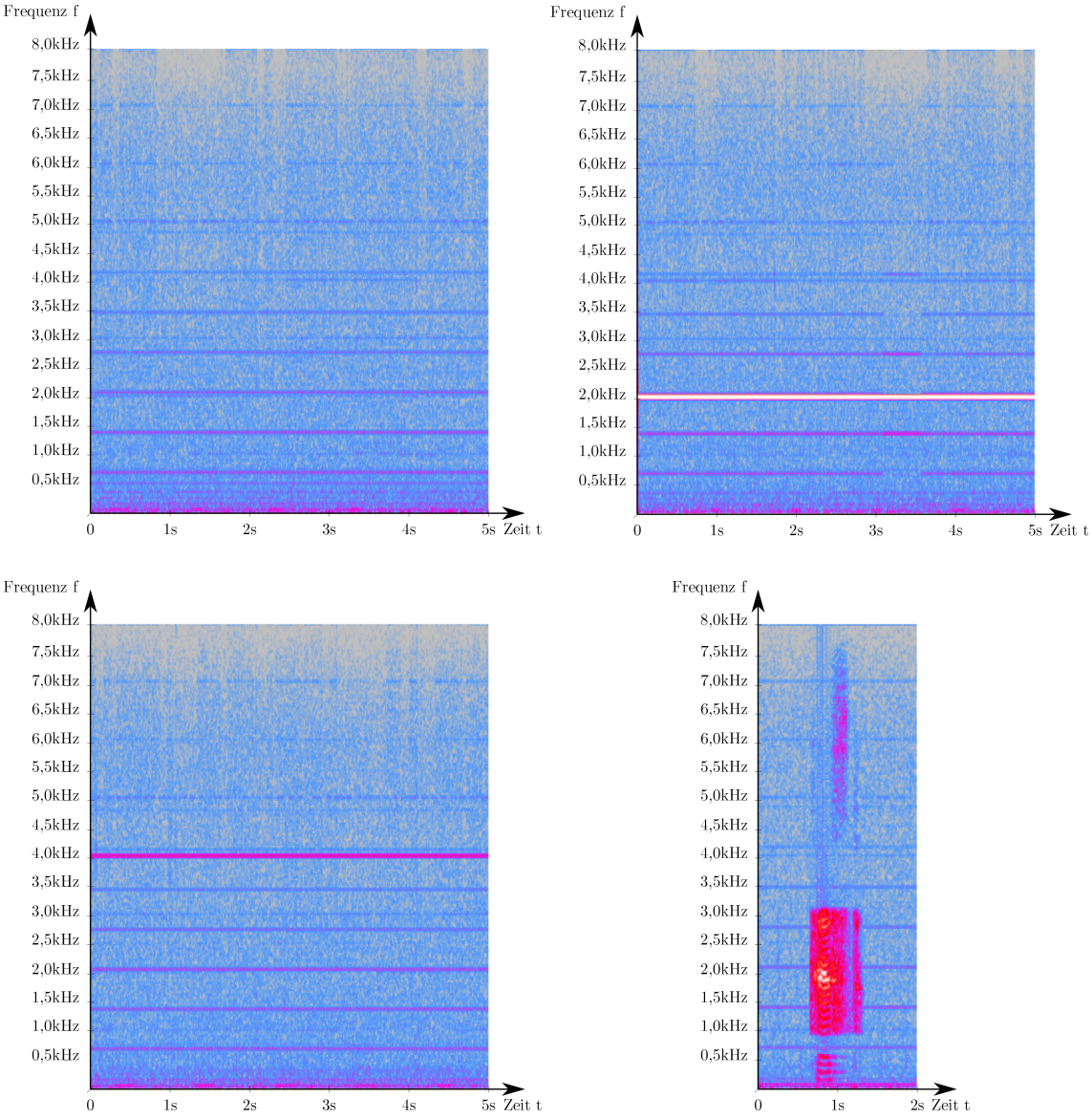


Abbildung A9: Spektrogramme Bandpass gefiltert
(links oben 500 Hz, rechts oben 2 kHz, links unten 4 kHz, rechts unten $Testlaut$)

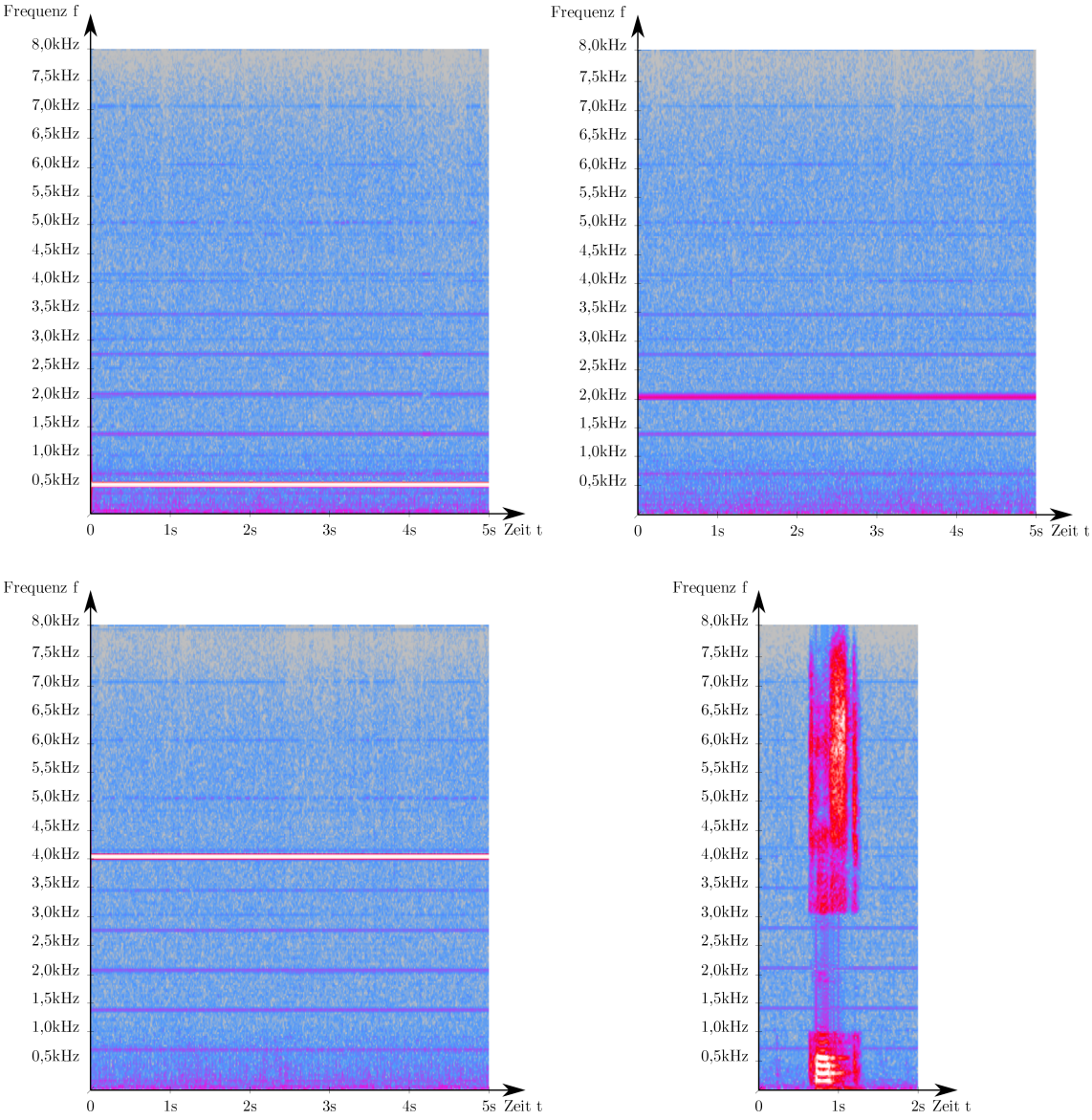


Abbildung A10: Spektrogramme Bandstopp gefiltert
(links oben 500 Hz, rechts oben 2 kHz, links unten 4 kHz, rechts unten $Testlaut$)

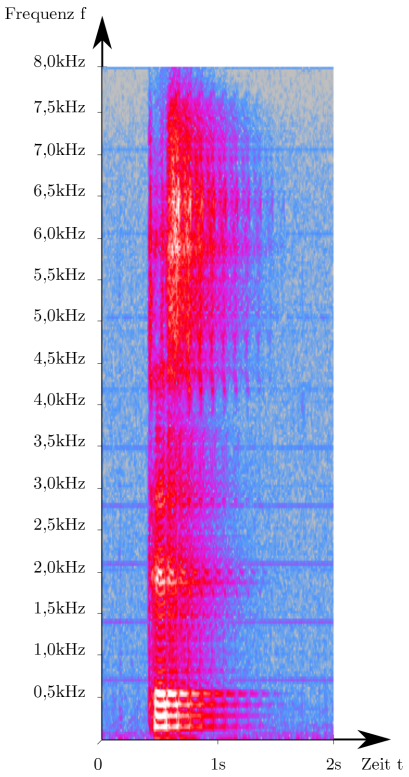


Abbildung A11: Spektrogramm Echo *Testlaut*

Literaturverzeichnis

Bücher & Skripte

[grueningen] **Daniel Ch. von Grüningen: Digitale Signalverarbeitung.** 2. Auflage, 2002, Carl Hanser Verlag München Wien.

[hoffmann] **Hoffmann, R.: Signalanalyse und -erkennung.** Springer Berlin, New York 1998. ISBN 3-540-63443-6

[enden] **Ad W.M. van den Enden und Niek A.M. Verhoeckx: Digitale Signalverarbeitung.** Friedr. Vieweg & Sohn Verlagsgesellschaft mbH, Braunschweig 1990. ISBN 3-528-03045-3

[dobl] **Gerhard Doblinger: Signalprozessoren (Architekturen, Algorithmen, Anwendungen) (2. Auflage).** J. Schlembach Fachverlag, Wilburgstetten 2004. ISBN 3-935340-43-5

[proch] **Prof. Dipl.-Ing. Ermenfried Prochaska und 7 Mitautoren: Digitale Signalprozessoren, Kontakt&Studium/Meß- und Prüftechnik Band 244.** expert verlag Böblingen 1988. ISBN 3-8168-0287-1

[kester] **Analog Devices, Inc./edited by Walt Kester: Mixed-Signal and DSP Design Techniques.** NEWNES Verlag USA 2003. ISBN 0-75067-611-6

[kurz] **Kurz/Wagner: Signalprozessoren-Praxis: Algorithmen, Bausteine, Systeme Applikationen.** Franzis-Verlag GmbH MÃŒnchen 1991. ISBN 3-7723-6502-7

[pearson] **John G. Proakis & Masoud Salehi: Grundlagen der Kommunikationstechnik.** 2. Auflage, Pearson Studium München 2004. ISBN 3-8273-7064-7

[tref] **Spectrum digital: TMS320C6713 DSK Technical Reference.** Stafford 2004.

[wolff] **Dr. Prof.-Ing. habil. Matthias Wolff: GrundzÃŒge der Kommunikationstechnik.** Skript 2 Signale und Systeme.

Internetseiten

26.08.2013

[diewelt] **dpa/lw: Rekord-Computer ist so groß wie ein Baseball-Feld.** Axel Springer AG 2013, Zugriff 17:34 Uhr.
http://www.welt.de/wirtschaft/webwelt/article110355984/Rekord-Computer-ist-so-gross-wie-ein-Baseball-Feld.html

[nvidia] **NVIDIA Corporation: Whitepaper NVIDIAs Next Generation CUDATM Compute Architecture: Kepler TM GK110.** 2012, Zugriff 19:44 Uhr.
http://www.nvidia.com/content/PDF/kepler/NVIDIA-Kepler-GK110-Architecture-Whitepaper.pdf

[mfk] **Annlis von Steiger& Matthias Vatter: Telegrafie.** Museum für Kommunikation, 2012, Bern, Zugriff 21:01 Uhr.
http://www.mfk.ch/magicnews/telegrafie.pdf

[mix] **MIX Professional Audio and Music Production: 1928 Harry Nyquist Theorem.** 2006, Zugriff 21:21 Uhr.
http://mixonline.com/TECnology-Hall-of-Fame/harry-nyquist-theorem-090106/

[bell] **C.E. Shannon: A Mathematical Theory of Communication.** Reprinted from The Bell System Technical Journal, Zugriff 22:15 Uhr.

02.09.2013

[wedel] **Oliver R. Ahlemann: Methoden der digitalen Audiobearbeitung.** Wedel 2002, Zugriff 14:13 Uhr.
http://www.fh-wedel.de/ si/seminare/ss02/Ausarbeitung/9.digitalaudio/ audio1.htm

05.09.2013

[neoberserker] **Cornelius Bradter: MTA-Klausurvorbereitung.** TFH Berlin, Zugriff 15:14 Uhr.
www.neoberserker.de/ftp/ftp2/ADDA_von_Steps.pdf

11.09.2013

[mikroq] **mikrocontroller.net: Quantisierung.** Zugriff 16:22 Uhr
http://www.mikrocontroller.net/articles/Quantisierung

26.9.2013

[mikrod] **mikrokontroller.net: Digitale Signalverarbeitung.** Zugriff 11:12 Uhr.
http://www.mikrocontroller.net/articles/Digitale_Signalverarbeitung#Hardware

[mikrom] **mikrocontroller.net: Mikrocontroller.** Zugriff 11:12 Uhr.
http://www.mikrocontroller.net/articles/Mikrocontroller#Was_ist_ein_Mikrocontroller

[mirkof] **mikrocontroller.net: FPGA.** Zugriff 11:12 Uhr.
http://www.mikrocontroller.net/articles/FPGA

01.10.2013

[parallax] **Parallax Inc: 4-Channel 12-Bit SPI Serial Interface A/D Converter.** Zugriff 09:37 Uhr.
http://www.parallax.com/sites/default/files/styles/full-size-product/public/604-00061.png?itok=nr7xAZl1

02.10.2013

[suche] **H. Suche: Modulpraktikum Messmethoden, Verscuh A/D- und D/A-Wandler.** paderborn 2000, Version 20-07-2010, Zugriff 18:30 Uhr.
http://groups.uni-paderborn.de/physik/studieninfos/praktika/versuche_anleitungen/pm01.pdf

[kobll] **Thomas Wilbert: Kenngrößen und Eigenschaften zeitdiskreter LTI-Systeme.** Universität Koblenz-Landau 2005, Zugriff 15:04 Uhr.
http://www.uni-koblenz.de/p̃hysik/informatik/DSV/LTI.pdf

[iti] **Klaus Lipinski, Dipl.-Ing.: SDW (Sigma-Delta-Wandler).** Zugriff 19:21 Uhr.
http://www.itwissen.info/definition/lexikon/Sigma-Delta-Wandler-sigma-delta-converter-SDW.html

11.11.2013

[koblf] **Pascal Berger: Fast Fourier Transformation Praktische Durchführung einer diskreten Fourier Transformation.** Universität Koblenz-Landau 2005, Zugriff 14:51 Uhr.

[han] **Bernd Edler:Lineare zeitinvariante Systeme.** Leibniz Universität Hannover, Zugriff 15:18 Uhr.
http://www.tnt.uni-hannover.de/edu/vorlesungen/DigSig/downloads/WS0708/digsig-06.pdf http://www.uni-koblenz.de/p̃hysik/informatik/DSV/FFT.pdf

[wiley] **Digitaler Auszug aus: Rulph Chassaing: Digital Signal Processing and Applications with the C6713 and C6416 DSK.** John Wiley & Sons Inc. 2005 New Jersey, Zugriff 15:42 Uhr.
http://www.echelonembedded.com/dsphwlab/files/ChassaingBook_Chap1.pdf

[werner] **Martin Werner: Skript Signale und Systeme.** Hochschule Fulda 2010, Zugriff 16:00 Uhr.
http://www2.hs-fulda.de/ werner/lehre/sus/SigSys_P3.pdf

15.11.2013

[jaeger] **Andreas Jäger: Bachelorarbeit Entwicklung eines hybriden Hallalgorithmus zur Integration in ein Surround-Raumsimulationsverfahren.** Stuttgart 2008, Zugriff 17:18 Uhr.
http://www.hdm-stuttgart.de/ curdt/Jaeger_Andreas.pdf

[ahle] **Oliver R. Ahlemann: Methoden der digitalen Audioverarbeitung.** Seminar Informatik FH Wedel, Wedel 2002, Zugriff 15:18 Uhr.
http://www.fh-wedel.de/ši/seminare/ss02/Ausarbeitung/9.digitalaudio/audio3.htm

20.11.2013

[52lab] **8A52labs: TMS320C6713-Pheripherals.** Zugriff 10:18 Uhr.
http://8a52labs.wordpress.com/2011/08/27/tms320c6713pheripherals/

[mathh] **R2013b Documentation: spectrogram.** Zugriff 15:07 Uhr.
http://www.mathworks.de/de/help/signal/ref/spectrogram.html

[kaern] **Creating a Stand-alone System.** FH Kaernten, Zugriff 18:41 Uhr.
http://ext02.fh-kaernten.at/rts/intern/downloads/DSP/iw6000%20Chapter%2014.pdf

Eidesstattliche Erklärung

Der Verfasser erklärt an Eides statt, dass er die vorliegende Arbeit selbständig, ohne fremde Hilfe und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt hat. Die aus fremden Quellen (einschließlich elektronischer Quellen) direkt oder indirekt übernommenen Gedanken sind ausnahmslos als solche kenntlich gemacht. Die Arbeit ist in gleicher oder ähnlicher Form oder auszugsweise im Rahmen einer anderen Prüfung noch nicht vorgelegt worden.

Ort, Datum Unterschrift des Verfassers