Diplomarbeit

Digitale aktive Geräuschunterdrückung mit geringer Latenz

durchgeführt von
Georg Teschinegg

Universität für Musik und darstellende Kunst Graz
Institut für Elektronische Musik und Akustik

Begutachter:
O.Univ.-Prof. Dipl.-Ing. Mag. Dr.techn. Robert Höldrich

Betreuer

Dipl.-Ing. Markus Guldenschuh
Dipl.-Ing. Dr.techn. Alois Sontacchi
Dipl.-Ing. Dr.techn. Werner Magnes
Dipl.-Ing. David Fischer

Oktober 2012, Graz

EIDESSTATTLICHE ERKLÄRUNG

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt und die den benutzten Quellen wörtlich und inhaltlich entnommenen Stellen als solche kenntlich gemacht habe.

Georg Teschineg
Graz, Oktober 2012

STATUTORY DECLARATION

I declare that I have authored this thesis independently, that I have not used other than the declared sources / resources, and that I have explicitly marked all material which has been quoted either literally or by content from the used sources.

Georg Teschinegg
Graz, October 2012

Zusammenfassung

Kopfhörer mit analoger aktiver Geräuschunterdrückung haben sich in verschiedenen Formen und Ausführungen kommerziell etabliert. Aufgabe dieser Kopfhörer ist es, den schwer zu dämpfenden, tieffrequenten Anteil des Lärms durch die Zuspielung gegenphasigen Anti-Lärms auszulöschen. Analogen Systemen sind in ihrer Wirkungsweise allerdings strukturelle Grenzen gesetzt, die durch digitale Signalverarbeitung möglicherweise erweitert werden können. Grundvoraussetzung dafür ist ein digitales System, welches eine möglichst vernachlässigbare Latenz aufweist. Dies macht die Verwendung eines leistungsfähigen Mikroprozessors und schneller Analog-Digital- bzw. Digital-Analog-Umsetzer unerlässlich.

Gegenstand dieser Diplomarbeit ist die Realisierung eines digitalen Systems zur Geräuschunterdrückung für die Verwendung in Kopfhörern. Dabei wurden leistungsfähige Analog-Digital-Umsetzer bzw. Digital-Analog-Umsetzer mit einem DSP-Entwicklungssystem verbunden und zwei verschiedene Methoden, welche auf dem LMS-Algorithmus basieren, zur Schallauslöschung getestet. Es soll untersucht werden, ob einerseits eine stärkere Auslöschung des Störschalls als bei den analogen Pendants erreicht werden kann und ob andererseits eine Auslöschung in höheren Frequenzbereichen möglich ist.

Abstract

Analog active noise cancellation headphones in their vast styles and designs are well established on the commercial market. These headphones are built in order to cancel low frequencies that are hard to absorb. The low frequency noise is cancelled with "anti-noise" of the same amplitude but opposite sign. Analog systems however are structurally limited in their effectiveness. These limits could possibly be amended by digital signal processing. The basic requirement for this solution is a digital system that has very low latency. For that reason it is necessary to use a powerful microprocessor and fast analog-digital as well as digital-analog converters. This thesis is concerned with the realization of a digital noise cancellation system for headphones. Powerful analog-digital and digital-analog converters are combined with a DSP-system and consequently two different noise cancellation methods that are based on an LMS algorithm are tested. The thesis analyzes the possibility of digital active noise cancellation and compares its performance with analog systems.

Inhaltsverzeichnis

1 Motivation

Motivation für die Durchführung dieser Diplomarbeit ist die Frage, ob durch ein digitales Geräuschunterdrückungssystem die vorhandenen Nachteile analoger Systeme ausgemerzt werden können. Prinzipbedingt funktioniert aktive Geräuschunterdrückung nur in tieferen Frequenzbereichen und teilweise nur für gewisse Schalleinfallsrichtungen. Es soll untersucht werden, ob in diesem Frequenzbereich eine bessere Auslöschung erreicht werden kann und ob durch den Einsatz digitaler Signalarbeitung eine Erweiterung des Frequenzgangs möglich ist.

Einer jener Faktoren, welcher am stärksten zur Verminderung der Leistungsfähigkeit beiträgt, ist die Laufzeitverzögerung im System. Praktisch jede Komponente, die ein Signal durchläuft verursacht eine solche Latenz. Der Einfluss zusätzlicher Komponenten, welche in einem digitalen System notwendig sind, muss so gering wie möglich gehalten werden.

Zunächst sollen die notwendigen Grundlagen bereitgestellt werden, die zum Verständnis des Prinzips der aktiven Geräuschunterdrückung notwendig sind. Dabei wird detailliert auf die angesprochenen Nachteile der analogen Systeme eingegangen. Durch die Beschreibung des digitalen Konzepts wird gezeigt, wie aus diesem Nutzen gezogen werden soll.

Eine Simulation des digitalen Systems soll eine Einschätzung über die Leistungsfähigkeit und den Einfluss diverser Variablen wie z.B. Latenz, Länge des verwendeten Filters oder Schrittweite des Least Means Square-Algorithmus bereitstellen. Getätigte Annahmen sollen dadurch verifiziert oder wiederlegt werden.

Die bereits angespochenen zusätzlichen Komponenten, die für ein Digitalsystem notwendig sind, werden anschließend kurz vorgestellt und deren Einfluss auch durch eine Messung verdeutlicht.

Abschließend wird die Leistungsfähigkeit des digitalen Systems durch eine Messung ermittelt und einigen kommerziell erhältlichen analogen Lösungen gegenübergestellt. Dadurch kann ein Resümee gezogen werden über die Sinnhaftigkeit der Verwendung digitaler Signalverarbeitung in einem Geräuschunterdrückungssystem für Kopfhörer.

2 Theoretische Grundlagen

2.1 Signaltheoretische Grundlagen

2.1.1 Gruppenlaufzeit

Passiert ein Signal ein lineares zeitinvariantes (LTI) System, kommt es üblicherweise zu einer frequenzabhängigen Änderung der Phase und somit einer Verzögerung. Diese Verzögerung entspricht der Ableitung der Phase gemäß

$$\tau(\omega) = -\frac{d}{d\omega}\left\{arg\left[H\left(e^{j\omega}\right)\right]\right\} \tag{1}$$

und wird Gruppenlaufzeit genannt. Von besonderer Bedeutung sind dabei zwei Sonderfälle.

Bei einem linearphasigen System steigt die Phasenänderung linear mit der Frequenz, daher ergibt sich im Phasendiagramm eine Gerade. Die Ableitung der Geraden ergibt eine Konstante, das bedeutet, dass alle Frequenzen das LTI-System mit der gleichen Verzögerung durchlaufen [1]. Man spricht in diesem Zusammenhang von einer konstanten Gruppenlaufzeit. Filter mit konstanter Gruppenlaufzeit weisen eine symmetrische Impulsantwort auf. Dies geht mit einer Verbreiterung der Impulsantwort einher und bewirkt daher eine recht große, aber frequenzunabhängige Verzögerung.

Minimalphasige Systeme weisen auf jeden Fall eine unsymmetrische Impulsantwort und keine lineare Phase auf. Dadurch ergibt sich eine frequenzabhängige Gruppenlaufzeit, welche aber der kleinstmöglichen Laufzeitverzögerung entspricht.

2.1.2 Filterklassen

In LTI-Filtersystemen lässt sich die Impulsantwort durch Differenzengleichungen gemäß der Form

$$y(n) = -\sum_{k=1}^{N} a_k y[n-k] + \sum_{k=0}^{M} b_k x[n-k] \tag{2}$$

beschreiben [2]. Sie stellt dabei eine Linearkombination aus unterschiedlich gewichteten Eingangs- und Ausgangswerten dar. Die Übertragungsfunktion ergibt sich daher nach der Z-Transformation zu

$$H(z) = \frac{Y(z)}{X(z)} = \frac{\sum\limits_{k=0}^{M} b_k z^{-k}}{\sum\limits_{k=0}^{N} a_k z^{-k}}. \tag{3}$$

Das Übertragungsverhalten wird von den Pol-und Nullstellen des Systems bestimmt. Solche Übertragungsfunktionen haben eine unendliche Impulsantwort und werden aus diesem Grund *Infinit Impulse Response* (IIR) -Filter genannt. Der Nennerterm stellt den Feedback-Zweig des Filters dar, das bedeutet, dass ein Teil des Ausgangssignals an den Filtereingang rückgekoppelt wird. Dies kann bei unzureichendem Filterdesign zur Instabilität führen. Vorteilhaft ist jedoch die Tatsache, dass mit nur wenigen Filterkoeffizienten lange Impulsantworten erreicht werden können [1].

Wird ein System zur Gänze über die Nullstellen beschrieben und weist keine Polstellen ungleich Null gemäß

$$H(z) = \sum_{k=0}^{M} b_k z^{-k} \tag{4}$$

auf, so spricht man von einem System mit endlicher Impulsantwort (*Finit Impulse Response*, FIR) [1]. Da keine Ausgangsdaten an den Eingang rückgekoppelt werden, sind FIR-Filter immer stabil. Hat ein FIR-Filter eine symmetrische Impulsantwort, so weist es außerdem einen linearen Phasengang auf und hat somit eine konstante Gruppenlaufzeit. Die Länge der Impulsantwort entspricht der Anzahl der Koeffizienten [2].

2.1.3 Least Mean Square Algorithmus (LMS)

Beim LMS-Algorithmus handelt es sich um eine zum Standard gewordene Rechenvorschrift zur Berechnung der Koeffizienten $\underline{w}(n)$ eines adaptiven Filters [3]. Das Filter verändert ein am Filtereingang anliegendes Referenzsignal x(n) derart, dass es bestmöglich einem nachzubildenden Signal d(n) entspricht.

$$d\,(n) \stackrel{!}{=} y(n) = \underline{w}^T(n)\underline{x}(n) \tag{5}$$

Das Filterausgangssignal y(n) stellt also eine Schätzung von d(n) dar. Der Unterschied der beiden entspricht dem Fehler e(n).

$$e(n) = d(n) - y(n) = d(n) - \underline{w}^T(n)\underline{x}(n) \tag{6}$$

Solche Algorithmen sind meist Optimierungsalgorithmen [4]. Als Optimierungskriterium wird nun die Minimierung des quadratischen Fehlers herangezogen, da ein möglichst kleiner Fehler impliziert, dass d(n) sehr gut angenähert wurde. Bei den betrachteten Signalen handelt es sich um statistische Größen, daher wird die Leistung des zu erwartenden Fehlers betrachtet [5]. Daraus folgt die Formulierung einer Kostenfunktion gemäß

$$J = E\left\{e^2(n)\right\}. \tag{7}$$

Durch Einsetzen der entsprechenden Größen ergibt sich die Kostenfunkton zu

$$J = E\left\{\left[d(n) - \underline{w}^T(n)\underline{x}(n)\right]^2\right\}. \tag{8}$$

Die Berechnung des minimalen Fehlers entspricht einer Extremwertaufgabe, die Kostenfunktion wird dementsprechend nach den Filterkoeffizienten abgeleitet und im Anschluss Null gesetzt. Die Ableitung der Kostenfunktion ergibt

$$\frac{\delta J}{\delta w} = -2\underline{p}(n) + 2\underline{w}(n)R_{\underline{xx}}(n), \tag{9}$$

wobei $\underline{p}$ dem Kreuzkorrelationsvektor zwischen d(n) und x(n), und $R_{\underline{xx}}$ der Autokorrelationsmatrix der Eingangsdaten entspricht [3]. Nach dem Nullsetzen ergibt sich die sogenannte "Wiener-Lösung" zu

$$\underline{w}_{opt} = \underline{p} * R_{\underline{xx}}^{-1}. \tag{10}$$

Die Wiener-Lösung setzt stationäre Eingangsgrößen voraus. Für die Berechnung der Autokorrelationsmatrix und des Kreuzkorrelationsvektors ist die Kenntnis der Eingangssignale bis zum Zeitpunkt t $=\infty$ erforderlich, was in der Praxis nicht möglich ist.

Wie in (10) zu erkennen ist, erfordert die Berechnung der optimalen Filterkoeffizienten eine Matrixinversion. Für den in der Praxis üblichen Fall, dass die Eingangssignale nicht stationär sind, müsste permanent eine Matrixinversion durchgeführt werden. Die Matrix weist eine Toeplitz-Struktur auf, kann daher durch spezielle Algorithmen relativ effizient berechnet werden. Trotzdem ergibt sich in Vergleich zu anderen Algorithmen ein hoher Rechenaufwand.

Außerdem setzt die Wiener-Lösung voraus, dass die Autokorrelationsmatrix regulär ist. Ist dies nicht der Fall, zum Beispiel bei korrelierten Eingangssignalen, existiert die inverse Matrix nicht und es kann keine optimale Lösung gefunden werden.

Ein alternatives Verfahren zur Wiener-Lösung ist die Methode des stärksten Abfalls (*Steepest Decent Algorithm*) [4]. Die optimalen Filterkoeffizienten werden dabei nicht in einem Durchgang, sondern Schritt für Schritt gemäß

$$\underline{w}(n+1) = \underline{w}(n) + \mu \left[\underline{p}(n) - R_{\underline{x}\underline{x}}(n)\underline{w}(n)\right] \tag{11}$$

berechnet [3]. Eine Matrixinversion ist nicht mehr nötig, die Kenntnis der Signalstatistik allerdings schon. Der Parameter μ wird Schrittweite genannt und ist für die Anzahl der Iterationen verantwortlich, die durchgeführt werden müssen, bis die optimalen Filterkoeffizienten erreicht worden sind.

Auch hier ergibt sich das Problem, dass die Berechnung der Autokorrelationsmatrix und des Kreuzkorrelationsvektors in der Praxis nicht genau möglich ist. Außerdem ist es in der Praxis schwierig, überhaupt Werte für $d(n)$ zu erhalten. Aus diesem Grund müssen die statistischen Größen geschätzt werden. Eine einfache Schätzung ergibt sich über die Momentanschätzungen [3, 4] zu

$$R_{\underline{x}\underline{x}}(n) = \underline{x}(n)\underline{x}^T(n) \tag{12}$$

und

$$\underline{p}(n) = \underline{x}(n)d(n). \tag{13}$$

Ein Einsetzen von (12) und (13) in (11) führt unter Berücksichtigung von (6) zum LMS-Algorithmus:

$$\underline{w}(n+1) = \underline{w}(n) + \mu\underline{x}(n)\underline{e}(n) \tag{14}$$

Er basiert auf der Methode des stochastischen Gradientenabstiegs. Augrund seiner Stabilität und Simplizität ist er der am häufigst angewendete adaptive Algorithmus [4].

Große Bedeutung kommt der Schrittweite μ zu. Auf der einen Seite ist sie verantwortlich für die Geschwindigkeit, mit welcher der Algorithmus zur optimalen Lösung konvergiert. Um möglichst schnell auf Änderungen des geschätzten Signals d(n) reagieren zu können, muss die Schrittweite so groß wie möglich gewählt werden. Dabei ergeben sich aber folgende Probleme:

Wie bereits beschrieben enthält der LMS-Algorithmus nicht die für eine perfekte Lösung notwendige Statistik der Eingangssignale, sondern bloß eine grobe Schätzung. Bei jedem Update wird entsprechend der negativen Steigung der Tangente entlang der Fehlerfunktion gesprungen, bis das Minimum der Fehlerfunktion erreicht ist. Bei zu großem μ werden sich die Filterkoeffizienten schrittweisen vom Minimum wegbewegen. Der Algorithmus divergiert und das System

wird instabil. Die maximal mögliche Schrittweite, bei welcher der Algorithmus noch stabil bleibt, hängt vom maximalen Eigenwert der Autokorrelationsmatrix des Eingangssignals λ_{max} ab und ergibt sich zu

$$0 < \mu < \frac{2}{\lambda_{max}}. \tag{15}$$

Außerdem hat die Schrittweite Einfluss auf die Genauigkeit des Algorithmus. Da die optimale Lösung des Algorithmus nie ganz erreicht, sondern nur angenähert werden kann, wird sich immer ein Restfehler einstellen. Dieser Restfehler wird auch als Fehleinstellung bezeichnet und hängt gemäß

$$M = \frac{\mu}{2}\sum_{k=1}^{M}\lambda_k \tag{16}$$

unter anderem von der Schrittweite ab. Je größer die Schrittweite also gewählt wird, umso schneller wird die optimale Lösung zwar angenähert, umso deutlicher wird sie aber auch verfehlt. Demzufolge bietet eine kleine Schrittweite zwar eine langsame Konvergenzgeschwindigkeit, ist jedoch am Ende näher an der optimalen Lösung.

Neben der Schrittweite hängt die Fehleinstellung noch von den Eigenwerten der Autokorrelationsmatrix als auch von der Filterlänge ab [3]. Es ergibt sich folgender Zusammenhang

$$M = \frac{\mu L \lambda_{av}}{2}, \tag{17}$$

wobei L die Filterlänge und λ_{av} den Durchschnittswert der Eigenwerte der Autokorrelationsmatrik darstellt.

Einen weiteren Einfluss auf die Konvergenzgeschwindigkeit hat neben der Schrittweite auch die Konditionszahl χ der Autokorrelationsmatrix [3, 6]. Sie gibt gemäß

$$\chi\left(R_{\underline{xx}}\right) = \frac{\lambda_{max}}{\lambda_{min}} \tag{18}$$

den Zusammenhang zwischen größtem und kleinstem Eigenwert an. Je größer die Konditionszahl, umso langsamer wird der Algorithmus konvergieren. Daher wird die Adaption bei gefärbtem Rauschen länger daueren als bei ungefärbtem.

2.2 Aktive Lärmunterdrückung

2.2.1 Grundlagen

Der Bekämpfung von Umgebungslärm wird in vielen Bereichen immer größere Bedeutung beigemessen. Der klassische Ansatz, Lärm mit Hilfe passiver Dämpfungsmaßnahmen (zum Beispiel mittels spezieller Schalldämpfer, absorbierend ausgestatteten Gehäusen von Quellen, raumakustischer Maßnahmen etc...) zu bekämpfen, ist sehr effizient für einen breiten Frequenzbereich, findet jedoch seine Grenzen bei stärker werdendem tieffrequenten Anteil des Umgebungslärms. Passive Dämpfung in diesem Bereich ist teuer, nimmt viel Platz in Anspruch und ist prinzipiell ineffizient [7].

Aktive Geräuschunterdrückung (im Englischen *Active Noise Cancellation*, ANC) basiert auf dem Prinzip der destruktiven Schallüberlagerung (siehe Abbildung 1). Einem über eine Primärstrecke einfallenden Störschall wird ein durch eine sogenannte Sekundärstrecke erzeugtes

Antischall-Signal mit gleichem Pegel, aber 180° gedrehter Phase zugespielt, wodurch es zur Auslöschung kommt.

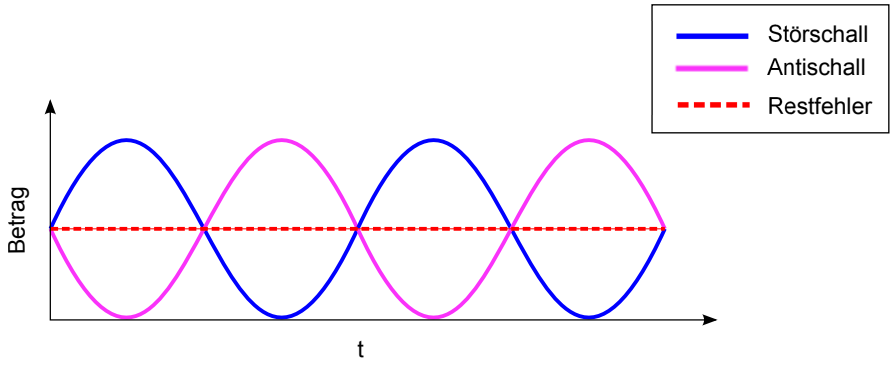


Abbildung 1: Prinzip ANC

Es ist leicht zu verstehen, warum die aktive Lärmunterdrückung speziell bei tieferen Frequenzen gut funktioniert: Bei tieffrequentem Schall ändert sich die Phase nur sehr langsam, wodurch genug Zeit bleibt, den einfallenden Störschall zu ermitteln und gegenphasig wieder auszuspielen. Je höher die Frequenz des Störschalls wird, umso weniger Zeit ist dafür vorhanden. Ab einer gewissen Grenzfrequenz ändert sich die Phase des Störschalls zu schnell und eine gezielte, gegenphasige Überlagerung kann nicht mehr gewährleistet werden. Dadurch kommt es oberhalb der Grenzfrequenz sogar zu konstruktiven Überlagerungen und damit zu einer Verstärkung des Störschalls.

Die Realisation eines solchen Systems benötigt zumindest ein Mikrofon zur Messung des Störschalls und einen Lautsprecher zur Aussendung des Antischalls und wurde erstmals 1936 in einem Patent von Paul Lueg [8] vorgestellt [7].

2.2.2 Anwendungen

Die Liste für potentielle Anwendungsmöglichkeiten ist lang und umfasst praktisch alle Bereiche, in welchen tieffrequente oder periodische Störungen auftreten. So verwendet zum Beispiel die Autoindustrie aktive Geräuschkompensation sowohl um den Geräuschpegel im Innenraum eines Automobils zu senken als auch den Lärm des Abgassystems zu reduzieren. Ebenso wird ANC verwendet, um in großen Industriehallen den Lärm lauter Maschinen zu dämpfen und somit das Arbeiten dort erträglicher zu gestalten.

Aktive Lärmunterdrückung ist selbstverständlich nicht limitiert auf akustische Einsatzgebiete. “Lärm” kann in seiner ursprünglichen Form ebenso als elektrisches Störsignal oder als Vibration verstanden werden. Aktive Geräuschkompensation kann daher in all jenen Bereichen zum Einsatz kommen, in welchen tieffrequente Störsignale den Anwender / die Anwenderin vor große Probleme stellen.

Die Verwendung von ANC in Kopfhörern gehört wohl zu den am häufigst umgesetzten Implementierungen. Da auch in dieser Arbeit die aktive Geräuschunterdrückung für Kopfhörer

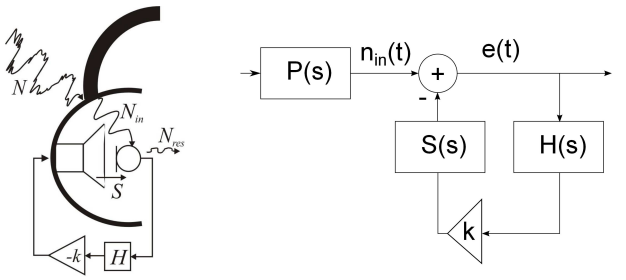
realisiert wurde, sollen alle weiteren Überlegungen auf diesen Anwendungszweck gerichtet sein.

2.3 ANC Kopfhörer

2.3.1 Analoges ANC

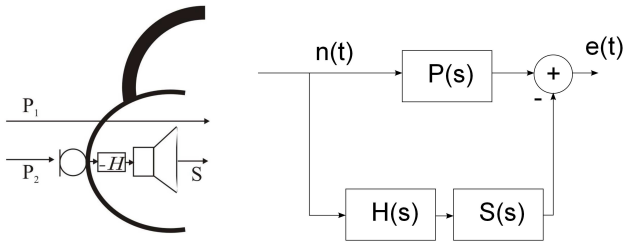
Der Großteil der kommerziell erhältlichen ANC Kopfhörer ist üblicherweise analog aufgebaut. Das bedeutet, dass das Antischall-Signal mit einem fixen Analogfilter *H(s)* geformt wird. Es gibt zwei Möglichkeiten, wie ein solches Filter eingesetzt wird, was in zwei häufig anzutreffenden Realisationen resultiert. Die erste ist die *Feedforward*-, die zweite die *Feedback*-Schaltung.

a.)



Übertragungsstrecke G(s) = $\frac{E}{N_{in}}$
Passive Kopfhörerdämpfung P(s)
Sekundärstrecke S(s)
Analoges Filter H(s)
Eingedrungener Störschall $n_{in}(t)$
Restfehler e(t)
Verstärkung k

b.)



Übertragungsstrecke P(s) = $\frac{P_1(s)}{P_2(s)}$
Sekundärstrecke S(s)
Analoges Filter H(s)
Einfallender Störschall n(t)
Restfehler e(t)

Abbildung 2: a.) Analoges Feedback ANC
b.) Analoges Feedforward ANC

Bei der Feedback-Schaltung soll das analoge Filter den kritischen Hochfrequenzbereich filtern, damit es hier nicht zu konstruktiven Überlagerungen kommt (siehe 2.2.1) Der Verstärkungsfaktor k soll bei Feedback-Schaltung für eine möglichst gute Unterdrückung sorgen. Der Nachteil beim analogen Feedback ANC ist die problematische Wahl von der Verstärkung k. Als Übertragungsfunktion G(s) des Gesamtsystems ergibt sich

$$G(s) = \frac{1}{1 + k * H(s) * S(s)}. \tag{19}$$

In einem solchen Szenario hängt die Leerlaufverstärkung prinzipiell nur von der Verstärkung k ab. Um eine hohe Auslöschung zu erreichen, sollte man einen möglichst großen Wert wählen um

G(s) klein zu halten. Kommt es innerhalb der geschlossenen Schleife zu Phasenverschiebungen, z.B durch die Verstärkung zu hoher Frequenzen im Rücklaufzweig, wird eine positive Rückkopplung erzeugt, das System gerät in Oszillation und wird instabil. Mathematisch betrachtet muss k*H(s)*S(s) ungleich -1 sein, da das System an diesem Punkt eine Polstelle aufweist und damit das Stabilitätskriterium nach Nyquist eingehalten wird. Die Polstelle entspricht also einer Phasendrehung um 180 Grad [9].

Beim Feedforward-System nimmt ein Mikrofon, welches sich außen an der Kopfhörermuschel befindet, den einfallenden Schall auf. Mit einem analogen Filter wir nun das Antischall-Signal geformt und über den Lautsprecher wiedergegeben. Das analoge Filter sollte dabei den Übergang des Schalls von außen am Kopfhörer nach innen nachbilden. Der Antischall überlagert sich mit dem einfallenden Störschall und sorgt für die Schallauslöschung. Ein Nachteil gegenüber dem Feedback-System ergibt sich aus der Starrheit des Systems und damit der Unfähigkeit, sich auf Änderungen der Störschallquellen einzustellen:

Mann muss Lärm, als auch die Umgebung, in welcher er sich ausbreitet und in welchem sich der Kopfhörer (bzw. dessen Träger/Trägerin) befindet, als höchst zeitvariant betrachten. Das bedeutet, dass sich das Phasenverhältnis zwischen Außenmikrofon und Überlagerungspunkt im inneren des Kopfhörers ständig ändert. So funktionieren analoge Feedforward-Systeme z.B. nur für eine bestimmte Schalleinfallsrichtung gut, ändert sich diese, muss eine Einschränkung der Leistungsfähigkeit in Kauf genommen werden. Um ein solches zeitvariantes Signal gut unterdrücken zu können, müsste sich das Filter auf die Änderungen einstellen können. Dies ist nur mit einem ANC-System zu bewerkstelligen, welches ein adaptives Filter aufweist.

Der Vorteil von Feedforward-Systemen gegenüber jenen mit Feedback-Struktur liegt darin, dass durch das außen befindliche Mikrofon die Referenz bereits ermittelt wurde, bevor der Schall das menschliche Ohr erreicht hat. Feedback-Systeme sind durch die Gruppenlaufzeiten prinzipiell immer zu spät und können auf den bereits entstandenen Fehler nur reagieren. Beim Feedforward-System erhält man die Referenz aber schon zu einem früheren Zeitpunkt. Die Zeit, welche die Schallwelle benötigt um vom Referenzmikrofon zum menschlichen Ohr zu wandern, steht bereits für die Erzeugung des Antischall-Signals zur Verfügung.

Der Vorteil bei analogen Systemen generell ergibt sich aus der relativ simplen Sekundärstrecke. Als Sekundärstrecke wird die Summe aller Komponenten bezeichnet, die das Signal durchlaufen muss. Sie umfasst im Wesentlichen das Mikrofon, Vorverstärker, Filter, Verstärker und Lautsprecher. Da für ein digitales System weitere Komponenten notwendig sind, welche eine zusätzliche Latenz verursachen, weisen analoge Systeme in der Regel kürzere Gruppenlaufzeiten auf. Eine große Gruppenlaufzeit hätte eine starke Verzögerung des Signals zur Folge, wodurch wiederum höhere Frequenzen nicht richtig ausgelöscht und unter Umständen (durch konstruktive Überlagerungen) sogar verstärkt werden.

2.3.2 Digitales ANC

2.3.2.1 Überblick

Die Verwendung eines digitalen ANC-Systems ermöglicht auf einfache Weise die Implementierung eines adaptiven Filters. Solche werden aus Stabilitätsgründen und aufgrund der einfachen Realisierung meist als FIR-Filter implementiert. Deren Änderungsfähigkeit erlaubt es digitalen ANC-Systeme sich auf den zeitvarianten Umgebungslärm einstellen und man erhofft sich aus

diesem Grund eine bessere Performance im Diffusfeld bzw. bei Quellen, deren Einfallsrichtung sich im laufenden Betrieb ändert.

Nachteilig wirkt sich die wesentlich komplexere Sekundärstrecke aus. Das Signal durchläuft neben den schon von der analogen Ausführung bekannten Komponenten wie Mikrofone, (Vor-) Verstärker und Lautsprecher ebenso Anti-Aliasing-Filter, Analog-Digital-Umsetzer, Digital-Analog-Umsetzer und Rekonstruktionsfilter. All diese Komponenten erzeugen eine zusätzliche Latenz des Ausgangssignals. Um eine eindeutige Verbesserung gegenüber analogen ANC-Systemen zu erreichen, ist es von fundamentaler Wichtigkeit, diese Latenz so gering wie möglich zu halten!

2.3.2.2 Feedforward ANC

Die generelle Funktionsweise des Feedforward-Algorithmus ist in Abbildung 3 zu sehen. Dabei dient ein Mikrofon an der Außenseite der Kopfhörermuschel als Referenz für den einfallenden Störschall. Ein weiteres Mikrofon befindet sich an der Innenseite der Kopfhörermuschel, hinter dem Lautsprecher. Dieses ermittelt den Fehler, also das noch vorhandene Restsignal, nachdem sich der Antischall mit dem einfallenden Primärschall überlagert hat.

Mathematisch betrachtet steht man vor dem Problem, ein unbekanntes System *P(z)* identifizieren zu müssen. Dieses System stellt die variable und daher zeitvariante Übertragungsstrecke zwischen Referenzmikrofon und Fehlermikrofon dar. Die Zeitvarianz ergibt sich durch die in Abschnitt 2.3.1 beschriebene Varianz der Quelle bzw. deren Übertragunsstrecke. Durch Kenntnis der beiden Eingangsgrößen kann sich das adaptive Filter *W(z)* mit Hilfe des LMS-Algorithmus (siehe 2.1.3) zu jedem Abtastzeitpunkt auf die Änderung von P(z) einstellen.

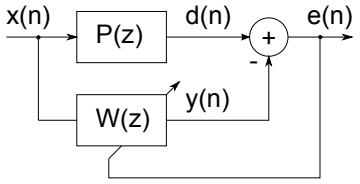


Abbildung 3: Feedforward ANC

Der Feedforward-Algorithmus stellt somit eine Systemidentifikation dar. Die bestmögliche Auslöschung wird erreicht, wenn (6) gleich Null ist, daher wenn

$$W(z) = P(z) \tag{20}$$

Durch die bereits im Abschnitt 2.3.1 erwähnte Sekundärstrecke $S(z)$ ändert sich das reale System entsprechend Abbildung 4. Durch diese wird der Fehler nicht Null werden, wenn (20) erreicht wird. Das reale Filter muss daher

$$W(z) = \frac{P(z)}{S(z)} \tag{21}$$

werden.

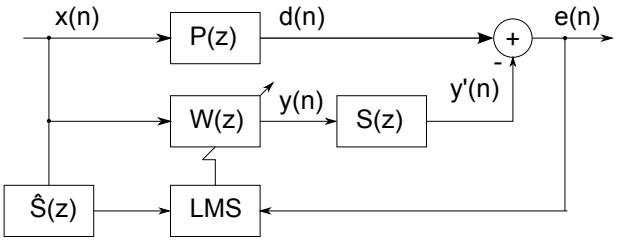


Abbildung 4: Feedforward ANC mit Sekundärstrecke

Die durch die Sekundärstrecke verursachte Verzögerung und Signaländerung muss bei der Implementierung eines solchen Systems berücksichtigt werden. Andernfalls würde es z.B. zu einem zeitlichen Versatz zwischen Referenzsignal und Fehlersignal im LMS-Algorithmus kommen. Dieser würde nicht mehr konvergieren, das somit erzeugte Antischallsignal y(n) würde eine Minimierung des Fehlers e(n) nicht erreichen können und zu Instabilität führen.

Der Einfluss der Sekundärstrecke muss demnach ermittelt werden (siehe Abschnitt 5). Wird das Referenzsignal nun mit der ermittelten Sekundärstrecke gefiltert, ist der zeitliche Versatz zwischen Referenzsignal und Fehlersignal beim LMS-Algorithmus aufgehoben und die Stabilität sichergestellt. Die Verwendung der geschätzten Sekundärstrecke entsprechend Abbildung 4 führt zum sogenannten *Filtered X-Algorithmus*, kurz FXLMS. Es sind somit, neben der Berechnung des LMS-Algorithmus, zwei FIR-Filterberechnungen notwendig.

Die Verwendung eines digitalen Filters führt zu einer besseren Schallauslöschung bei nichtstationären Signalen als bei der Verwendung eines analogen Filters, welcher nur für eine Einfallsrichtung ausgelegt ist.

2.3.2.3 Feedback ANC

Im Gegensatz zum Feedforward-System kommt das Feedback-System nur mit einem Mikrofon, dem Fehlermikrofon, aus. Mathematisch betrachtet handelt es sich nun nicht mehr um eine Systemidentifikation. Durch das Fehlen eines Referenzmikrofons gibt es kein System, welches identifiziert werden müsste. Stattdessen wird die Referenz vom Algorithmus selbst erzeugt. Der LMS übernimmt hierbei vielmehr die Aufgabe der Prädiktion. Aus den vorangegangenen Eingangsdaten wird ein neuer Ausgangswert geschätzt. Abbildung 5 verdeutlicht die Wirkungsweise des Feedback-Systems.

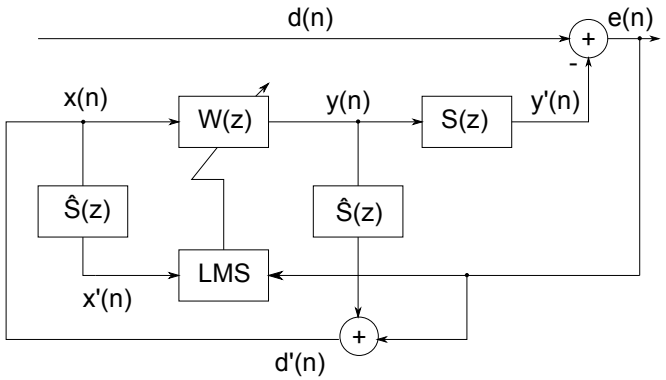


Abbildung 5: Feedback ANC mit Sekundärstrecke

Die im vorangegangenden Abschnitt beschriebenen Überlegungen zur Sekundärstrecke gelten selbstverständlich auch hier.

Der Feedback-Algorithmus benötigt drei FIR-Filterberechungen und stellt daher einen höheren Berechungsaufwand als der Feedforward-Algorithmus dar.

3 Simulation der ANC-Algorithmen

3.1 Einleitung

Vor der Realisierung des digitalen Antischall-Systems soll die Wirkungsweise der Feedforward- bzw. Feedback-Algorithmen in einem Matlab-Skript überprüft und getestet werden.

Zuerst sollen einige Abschätzungen bezüglich des Einflusses verschiedener Parameter getroffen werden. Zum Beispiel werden unterschiedliche Latenzen der Sekundärstrecke simuliert und deren Einfluss auf die Leistungsfähigkeit des LMS-Algorithmus ermittelt. Ebenso wird untersucht, ob sich eine bessere Auslöschung durch Erhöhung bzw. Senkung von Parametern wie Filterlänge oder Schrittweite des LMS-Algorithmus ergeben kann. Als Anregungssignal dafür wird stationäres Rosa-Rauschen verwendet.

Im Anschluss wird das System bei Anregung einer nicht stationären Quelle getestet. Als Geräuschquelle dient ein veränderliches Motorgeräusch. Dadurch soll untersucht werden, wie robust der LMS Algorithmus gegenüber nichtstationären Signalen ist.

Zum Abschluss wird die Leistungsfähigkeit der Feedforward- und Feedback-Algorithmen für eine einzelne Einfallsrichtung einander gegenübergestellt. Es soll herausgefunden werden, ob ein Algorithmus dem anderen gegenüber zu bevorzugen ist.

3.2 Vorbereitungen

Um ein realistisches Szenario simulieren zu können, müssen zuerst passende Geräuschquellen erzeugt und die Sekundärstrecke gemessen werden.

Geräuschquellen: Für eine realitätsnahe Simulation des ANC-Systems in einem realen Schallfeld müssen die verschiedenen Phasenverhältnisse zwischen Referenz- und Fehlermikrofon berücksichtigt werden. Daher wurden Impulsantworten aus verschiedenen Richtungen sowohl mit dem Referenzmikrofon als auch mit dem Fehlermikrofon gemessen. Der Messaufbau ist in Abbildung 6 zu sehen. Er besteht im Wesentlichen aus dem an einem Kunstkopf angebrachten ANC-Kopfhörer (siehe Abschnitt 4.2), welcher sich unter einem Lautsprecherarray, bestehend aus acht Lautsprechern, befindet. Über dieses Array wurde nacheinander ein Sweepsignal abgespielt und mit den in dem Kopfhörer verbauten Mikrofonen aufgenommen. Um alle Richtungen zu simulieren, wurde der Kunstkopf auf einem drehbaren Podest positioniert und bei unterschiedlichen Positionen gemessen. So war es möglich, nach der Entfaltung mit dem inversen Sweepsignal, Übertragungsfunktionen aus 85 Richtungen zu ermitteln. Für die Simulation wurde Rosa-Rauschen mit Impulsantworten einer ausgewählten Richtung gefaltet. Damit können, für diese Einfallsrichtung, realistische Phasenverhältnisse zwischen dem äußeren und inneren Mikrofon simuliert werden.

Abbildung 6: Messung der Impulsantworten

Sekundärstrecke: Für die im FXLMS benötigte Sekundärstrecke wurde ebenfalls auf eine bereits vorhandene Messung des ANC-Kopfhörers zurückgegriffen. Dabei wurde ein Sweepsignal über den Kopfhörerlautsprecher abgespielt und mit dem innen angebrachten Mikrofon wieder aufgezeichnet. Die dadurch erhaltene Impulsantwort wurde sowohl zur Simulation der echten als auch der geschätzten Sekundärstrecke verwendet. Dies entspricht im Algorithmus einer idealen Schätzung der Sekundärstrecke, wie sie in der Praxis nicht erreichbar sein wird.

3.3 Latenz

Im Abschnitt 2.2 wurde bereits das zentrale Problem der Gruppenlaufzeiten bzw. der Sigalverzögerungen in ANC-Systemen thematisiert. Um eine Einschätzung des Problems zu erhalten, werden drei Simulationen mit jeweils unterschiedlichen Laufzeitverzögerungen durchgeführt. Da die Auswirkungen unabhängig von der Wahl des Algorithmus sind, wird auf den Feedback-Algorithmus zurückgegriffen. Angeregt wurde das System mit stationärem Rosa-Rauschen. Bei der ersten Simulation wurde gänzlich auf Latenz verzichtet, sie enspricht also dem Idealfall. Die zweite Simulation weist eine Verzögerung von 6 Samples, die dritte eine mit 10 Samples auf. Die Abtastrate beträgt 22,05 kHz. In Abbildung 7 sind die Ergebnisse zu sehen.

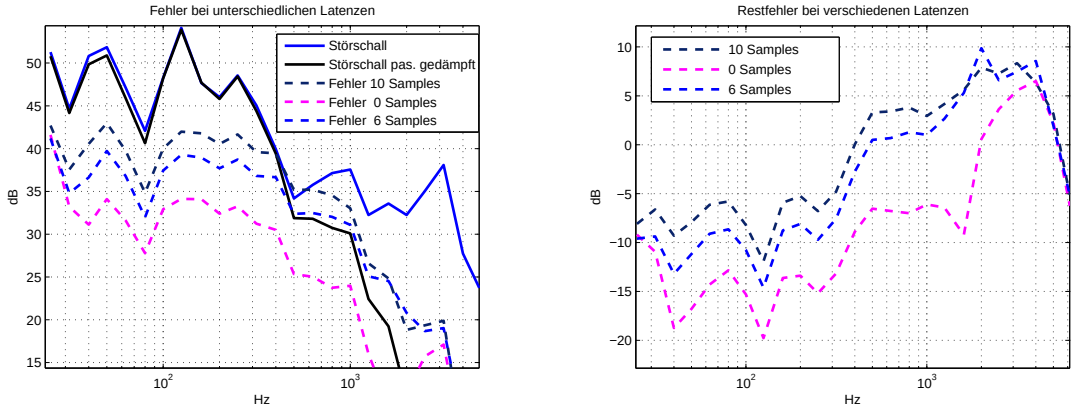


Abbildung 7: Einfluss der Latenz

Die blaue durchgängige Linie der linken Abbildung entspricht dem über die Zeit gemittelten Störschall am Außenmikrofon, die schwarze durchgängige Linie jenem am Innenmikrofon, betrachtet über die Frequenz. Deren Abfall ab ca. 400 Hz resultiert durch die passive Dämpfung der Kopfhörermuschel. Die Fehlersignale sind strichliert dargestellt. Man kann sehr gut erkennen, dass diese umso kleiner sind, je niedriger die Latenz gewählt wurde, was einer besseren Auslöschung entspricht.

Die Linien der rechten Abbildung zeigen den Unterschied der Fehlersignale zu den passiv gedämpften Innenmikrofonsignalen. Je weiter diese Kurven unter Null liegen, umso größer ist die Auslöschung im betrachteten Frequenzbereich. In höheren Frequenzbereichen überschreiten die Kurven die Nulllinie. Dort kommt es zu konstruktiven Überlagerungen und einer Störschallverstärkung. Je höher die Latenz gewählt wurde, umso früher tritt dieser Fall ein.

3.4 Schrittweite μ

Als nächstes wird der Einfluss der Schrittweite überprüft. Im System wurde dabei eine Latenz von zwei Samples bei 22,05 kHz eingestellt. Als Anregung dient wieder Rosa-Rauschen. In Abbildung 8 ist der zeitlich geglättete Verlauf der Energie des Fehlersignals für drei verschiedene Schrittweiten abgebildet. Links sind die Konvergenzgvorgänge zu sehen. Man kann erkennen, dass bei niedrigeren Schrittweiten ein längerer Zeitraum benötigt wird, bis die minimale Lösung erreicht wird.

Rechts sind die Restfehler nach dem Konvergenzvorgang abgebildet. Der größte Restfehler bleibt bei der höchsten Schrittweite übrig.

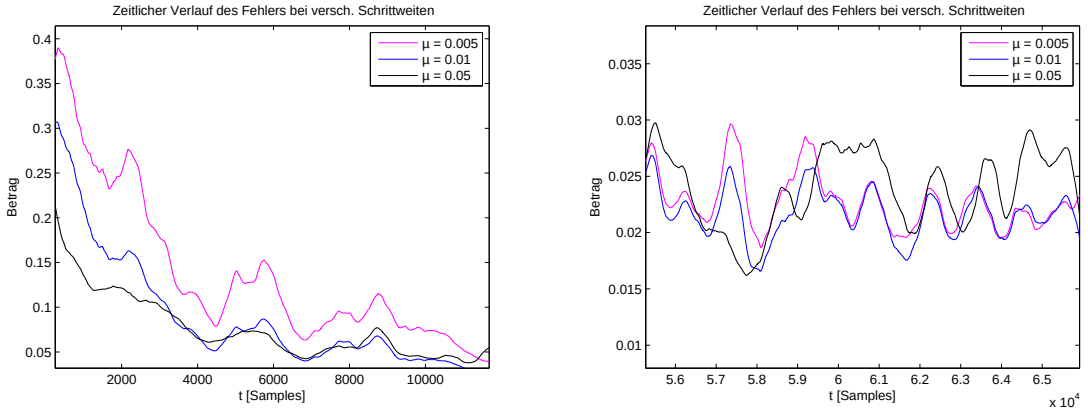


Abbildung 8: Einfluss der Schrittweite μ

Aufgrund der Messergebnissse kann man zu dem Schluss kommen, dass der nur geringfügig höhere Restfehler für eine schnellere Konvergenz in Kauf genommen werden kann, da in einer realen Situation stationäre Signale praktisch nicht vorkommen. In der Praxis auftretende Geräuschquellen sind höchst variabler Natur, deren Eigenschaften permanent großen Änderungen unterworfen sind. Um auf diese Änderungen schnell reagieren zu können, muss eine größere Schrittweite gewählt werden.

Der durch die größere Schrittweite verursachte Fehler des LMS Algorithmus wirkt sich nachteilig auf die Überlagerung höherer Frequenzen aus. Wie man in Abbildung 9 erkennen kann, kommt es bei zunehmender Schrittweite in diesem Bereich zu zunehmend konstruktiven Überlagerungen.

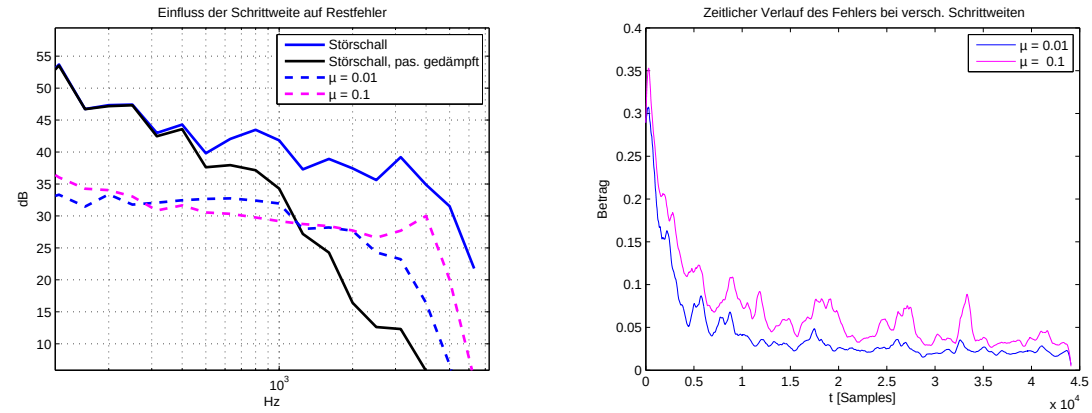


Abbildung 9: Restfehler bei größerer Schrittweite

Da das menschliche Gehör in diesem Frequenzbereich besonders empfindlich ist, können diese Überlagerungen störender sein als der entfernte tieffrequente Anteil. Mit Hilfe psychoakustischer Messgrößen wie beispielsweise der Lautheit kann eine geeignete Wahl von μ getroffen werden.

Wird die Schrittweite zu groß gewählt, wird der LMS instabil.

3.5 Filterlänge

Die gewählte Länge des FIR-Filters hat Einfluss auf die Faktoren Frequenzauflösung, Berechnungsaufwand und Fehleinstellung des LMS-Algorithmus.

Damit es zur Auslöschung im betrachteten Frequenzbereich kommen kann, muss das adaptive Filter die entsprechenden Frequenzkomponenten mit angemessener Genauigkeit abbilden können. Ist das nicht der Fall, wird eine Unterdrückung in diesem Bereich nicht funktionieren und die Leistungsfähigkeit des ANC-Systems beeinträchtigt werden. Höhere Filterlängen können aber auch größere Gruppenlaufzeiten bewirken, was sich wieder negativ auf die Latenz im System auswirkt. Wie in Abschnitt 2.1.3 bereits erwähnt wurde, erhöht sich mit steigender Filterlänge außerdem auch die Fehleinstellung des LMS-Algorithmus und damit der Restfehler nach der Adaption.

In Abbildung 10 ist die erreichte Auslöschung für unterschiedliche Filterlängen dargestellt.

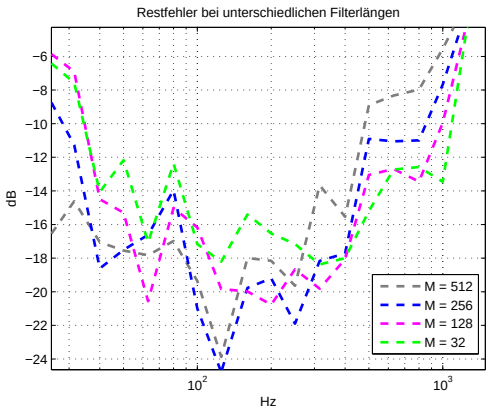


Abbildung 10: Einfluss der Filterlänge

Kleine Filterlängen resultieren zwar in niedrigen Gruppenlaufzeiten, eine bessere Auslöschung wird dadurch aber nicht unbedingt erreicht. Man kann erkennen, dass bei einer Filterlänge von 32 Punkten die Freqenzauflösung anscheinend nicht mehr ausreicht, um gewisse Frequenzkomponenten adäquat nachzubilden. Der Vorteil der geringeren Gruppenlaufzeit ist in den höheren Frequenzbereichen zu sehen, bei welchen es zu weniger konstruktiven Überlagerungen kommt.

In Abbildung 11 ist der zeitlich gemittelte Verlauf des Fehlersignals dargestellt. Wie zu sehen ist, stellt sich ein kleinerer Restfehler bei größeren Filterlängen ein.

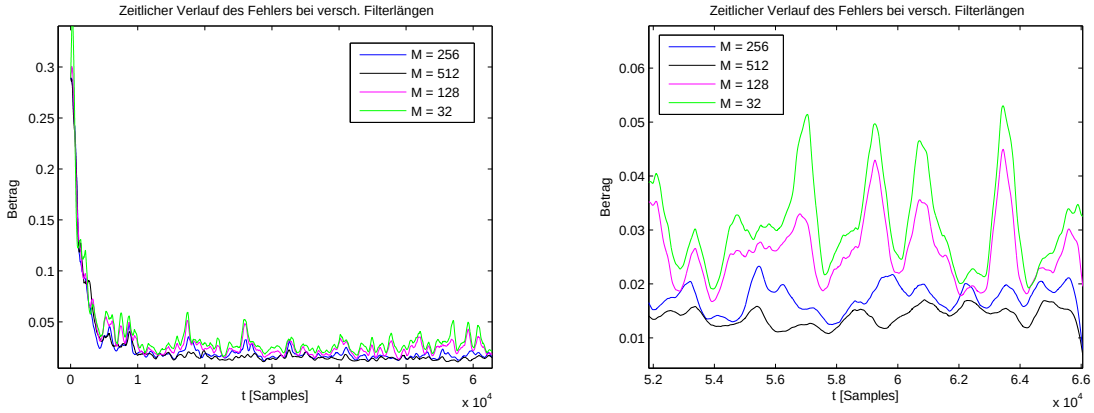


Abbildung 11: Restfehler bei unterschiedlichen Filterlängen

Nicht außer Acht zu lassen ist allerdings die Tatsache, dass mit der Filterlänge auch die Anzahl der Rechenoperationen, die in einer Faltung ausgeführt werden müssen, steigen.

3.6 Nichstationäres Anregungssignal

Bei Anregung des Feedforward-Systems mit einem nichtstationären Signal muss sich der LMS-Algorithmus möglichst schnell auf die Signaländerungen einstellen. Wie bereits in Abschnitt 3.4 erwähnt kann durch eine Erhöhung der Schrittweite eine schnellere Einstellung erreicht werden. Abbildung 12 zeigt links die Fehlerverläufe für unterschiedliche Schrittweiten bei einem Anregungssignal, bei welchem sich die Amplitude und der spektrale Inhalt mit der Zeit ändern. Zur besseren Vergleichbarkeit wird rechts der Fehlerverlauf dieses Signals mit jenem eines stationären Signals bei gleicher Schrittweite gegenübergestellt.

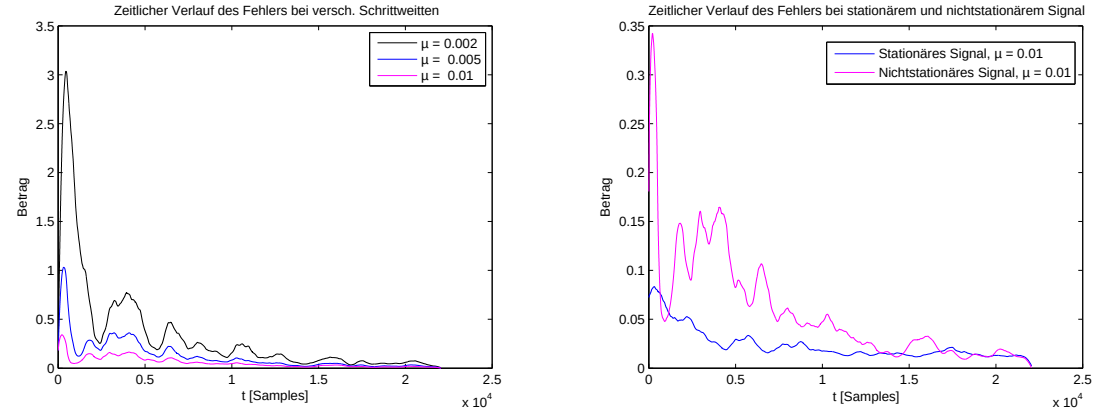


Abbildung 12: Fehlerverläufe für unterschiedliche Schrittweiten bei spektral nichtstationärem Anregungssignal

3.7 Vergleich zwischen Feedforward- und Feedback-Algorithmus

Es soll nun die Leistungsfähigkeit der unterschiedlichen Algorithmen gegenübergestellt werden. Um die Vergleichbarkeit gewährleisten zu können werden beide Algorithmen mit den selben Parametereinstellungen verwendet. Die Schrittweite beträgt einen Wert von 0.02, die Filterlänge ist 256 Punkte lang. In beiden Systemen wirde eine Latenz von zwei Samples bei 22.050 kHz eingeführt. Als Geräuschquelle dient stationäres Rosa-Rauschen.

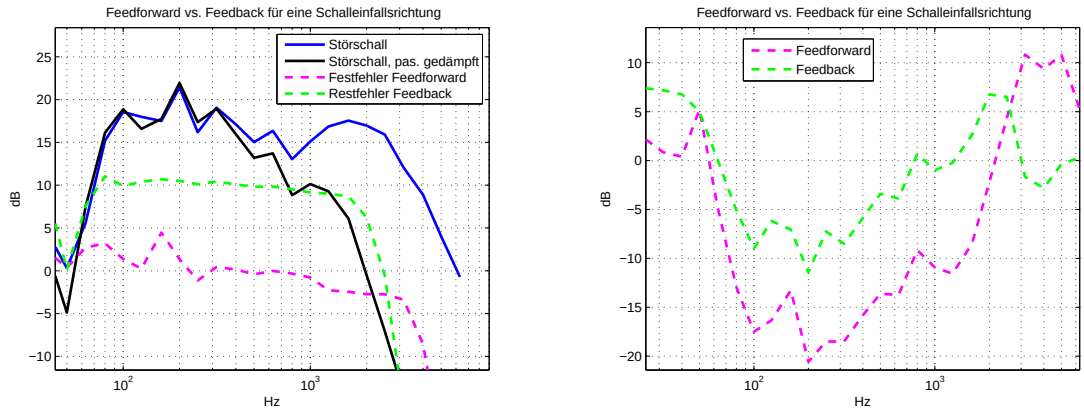


Abbildung 13: Feedforward vs. Feedback für eine Einfallsrichtung

Man sieht, dass sich der zeitliche Vorteil durch das Referenzmikrofon des Feedforward-Systems deutlich bemerkbar macht.

3.8 Zusammenfassung

Zusammenfassend kann behauptet werden, dass die größeren Fehler, die sich durch eine zu hohe Schrittweite und Filterlänge ergeben, keine allzu große Rolle spielen dürften. Sie werden durch die jeweiligen Vorteile wieder wettgemacht. Daher sollte ein möglichst hoher Wert für die Schrittweite gewählt werden, um sich auf die in der Praxis vorherschenden nichtstationären Signale rasch einstellen zu können. Man muss allerdings Abschätzungen treffen über den verstärkten Anteil der höheren Frequenz im System. Selbstverständlich muss die Stabilität im System auf jeden Fall gewährleistet sein.

Die Filterlänge sollte, solange der Rechenaufwand bewältigbar ist, nicht zu klein gewählt werden, um ausreichend Information über die Übertragungsfunktionen im System zu erhalten.

Ebenso konnte gezeigt werden, dass die Latenz im System so niedrig wie möglich gehalten werden muss, da die Leistungsfähigkeit des LMS-Algorithmus dadurch stark eingeschränkt wird.

Bei der Wahl des Algorithmus konnte kein Argument für das Feedback-System gefunden werden.

4 Sekundärstrecke - Beschreibung der Komponenten

4.1 Überblick

Die Realisierung eines funktionierenden ANC-Kopfhörerprototypen verlangt die Verknüpfung einer Reihe ausgewählter Komponenten. Zusammengenommen stellen sie die Sekundärstrecke dar. Wie in Abschnitt 2.3.2 bereits erwähnt, ist dies der kritische Punkt im digitalen System und für eine gute Performance hauptausschlaggebend. In diesem Abschnitt soll zunächst auf die verwendete Hardware und deren Funktion im Gesamtsystem eingegangen werden. Im Anschluss wird die Verknüpfung der einzelnen Komponenten beschrieben.

Abbildung 14 gibt einen Überblick über die benötigten Komponenten der Sekundärstrecke.

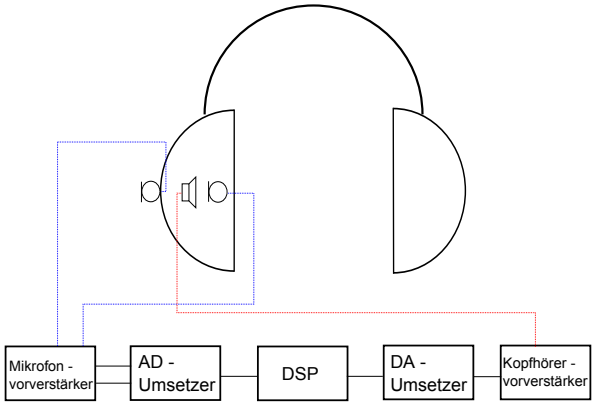


Abbildung 14: Prinzipschaltbild ANC Prototyp

Die Verwendung externer Umsetzer stellt keine rein technische Notwendigkeit dar. Ein konfigurierbarer Audiocodec ist auch auf dem verwendeten Evaluierungsboard des Signalprozessors (siehe Abschnitt 4.6) zu finden. Allerdings weisen die dort integrierten Anti-Aliasing-Filter bzw. Rekonstruktionsfilter eine Gruppenlaufzeit auf, welche die Performance des ANC-Systems erheblich einschränken würde. Durch die Verwendung ausgewählter Komponenten und eines speziell für diesen Anwendungszweck entworfenen Filters konnte das System für diese Anwendung optimiert und eine Sekundärstrecke mit niedriger Latenz gewährleistet werden.

4.2 Kopfhörer

Beim Kopfhörer handelt es sich um einen Prototypen, der dem Institut für Elektronische Musik und Akustik von der Firma AKG Acoustics GmbH zur Verfügung gestellt wurde. Auf jeder Seite wurde der Kopfhörer mit jeweils zwei Mikrofonen ausgestattet, damit die Verwendung der Feedforward-Methode möglich ist. Das Referenzmikrofon befindet sich auf der Außenseite und liefert Information über den in den Kopfhörer eindringenden Störschall. Des Fehlermikrofon ist an der Innenseite des Kopfhörers hinter dem Lautsprecher angebracht und ermittelt das Fehlersignal, welches nach der Überlagerung von Störschall und Antischall übrigbleibt.

4.3 Platine mit Mikrofonvorverstärker & Kopfhörervorverstärker

Ebenso von AKG Acoustics GmbH zur Verfügung gestellt wurde eine Platine, auf welche sich die Verstärkungseinheiten für alle benötigten Signale befinden.

Die Platine umfasst vier Vorverstärker für die Mikrofonsignale des Kopfhörers. Diese stellen eine fixe Verstärkung von ca. 20 dB bereit.

Die für die Digitalwandlung benötigten Anti-Aliasing-Filter befinden sich direkt nach den Verstärkern. Die Grenzfrequenz liegt dabei bei ca. 4000 Hz, danach erfolgt ein flacher Abfall von ca. 6 dB pro Oktave, was einem Filter erster Ordnung entspricht.

In der Praxis sind diese Werte ausreichend. Aktive Geräuschunterdrückung funktioniert in etwa bis 3 kHz. Signalanteile in höheren Frequenzregionen werden nicht benötigt, vor allem auch aus dem Grund, dass ab ca. 400 Hz die passive Dämpfung der Kopfhörermuscheln einsetzt. Viel eher profitiert man von der Tatsache, dass solch ein Filter erster Ordnung eine geringe Gruppenlaufzeit aufweist.

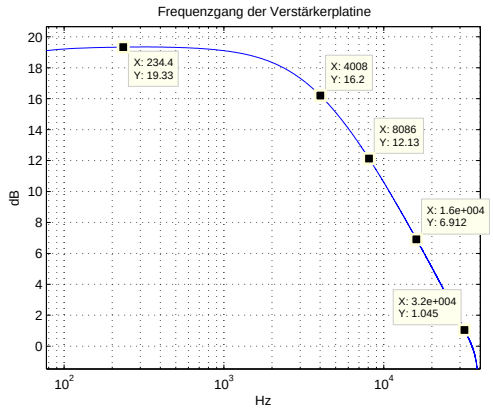


Abbildung 15: Verstärkung Mikrofonvorverstärker & Anti-Aliasing-Filter

Zusätzlich zur Vorverstärkung und Anti-Aliasing-Filterung verursacht der Mikrofoneingang noch eine Phasendrehung von 180°.

Der integrierte Kopfhörervorverstärker dient zur Rückführung des Antischall-Signals an den Lautsprecher des Kopfhörers. Zwei Trimmpotentiometer erlauben die Justierung der Lautstärke für den linken bzw. rechten Kanal.

4.4 Analog-Digital-Umsetzer

4.4.1 Umsetzer

Für die Analog-Digital-Umsetzung wurde auf den AD7655 von Analog Devices zurückgegriffen. Dabei handelt es sich um einen vierkanaligen Analog-Digital-Umsetzer (ADC) mit 16-Bit Auflö-

sung (entspricht einer Dynamik von ca. 96 dB) und einem maximalen Datendurchsatz von einer Million Samples pro Sekunde (1 MSPS).

Um gleichzeitig vier Kanäle verarbeiten zu können, kommen im Umsetzer zwei Abtast-Halte-Schaltungen (*Track-and-Hold*) sowie mehrere Multiplexer zum Einsatz. Die Wandlung erfolgt dabei immer simultan für zwei Kanäle. Dafür wird eine Zeit von nur zwei Mikrosekunden benötigt. Die unipolaren Eingänge erlauben einen Eingangsspannungbereich von null (Masse) bis 5 Volt. Bipolare Signale müssen daher im Vorfeld mit einem entsprechenden positiven Offset beaufschlagt werden. Die Wandlung der positiven Eingangsspannung liefert dementsprechend auch rein positive Binärwerte und erfolgt nach dem Wägeverfahren (sukzessive Approximation, SAR). Das Wandlungsergebnis kann im Anschluss sowohl seriell als auch parallel ausgelesen werden. Über den EXT/INT Pin ist es möglich den ADC als Master oder Slave zu definieren. Bei der Verwendung als Master wird ein intern generiertes Taktsignal als auch ein SYNC-Signal zum Auslesen der Daten bereitgestellt. Der Start der Wandlung erfolgt über eine negative Signalflanke am sogenannten *Conversion Start* (CNVST) Pin. Die Übertragung der Daten beginnt standardmäßig mit dem höchstwertigen Bit (MSB) des ersten Kanals. Wahlweise kann durch das High-Setzen des A/B-Pins auch der zweite Kanal zuerst übertragen werden.

Für den ANC-Prototypen wurden nur zwei der vier Eingänge benutzt. Die Datenübertragung erfolgt seriell mittels der Serial Peripheral Interface - Schnittstelle (SPI, siehe Abschnitt 4.9) .

Der Umsetzer ist von Haus aus mit keinem Anti-Aliasing-Filter ausgestattet, was die Implementierung eines eigenen Filters notwendig macht. Ein entsprechendes Filter ist bereits auf der Platine des Mikrofonvorverstärkers verbaut (siehe Abschnitt 4.3). Aufgrund der niedrigen Gruppenlaufzeit und der sehr hohen Geschwindigkeit des Umsetzers wird bei der Analog-Digital-Umsetzung kaum Latenz erzeugt.

Weitere Informationen zum ADC sind dem AD7655-Datenblatt [10] zu entnehmen.

4.4.2 ADC-Evaluierungsboard

Der AD7655 wird über ein eigenes Evaluierungsboard, dem EVAL-AD76XXCB von Analog Devices, angesprochen und in das System integriert. Die Eingänge sind, ebenso wie der Ausgang des davorgeschalteten Mikrofonvorverstärkers, als *Sub-Miniature-B* (SMB) Steckverbindung ausgeführt. Die vorhandenen Eingangsverstärker bereiten das Signal angemessen für die Wandlung auf. Zur Verwendung bipolarer Signale wird dabei ein 2.5 Volt Offset aufgeschlagen, wodurch diese rein positiv am Wandlereingang anliegen. Ebenso sind ein externes Taktsignal und die Anschlüsse für alle benötigten Referenzspannungen, das CNVST-Signal und der SPI- Datenleitungen vorhanden. Mittels einer speziellen Gatterschaltung kann ein CNVST Signal von 500 kHz erzeugt werden.

Da das Evaluierungsboard nur den Anschluss zweier Quellen erlaubt, können auch nur zwei Eingänge des ADCs genutzt werden. Ein beidseitiger Betrieb des ANC-Kopfhörers im Feedforward-Modus ist daher nicht möglich.

Der erste Eingang (AIN+) wird zum Anschluss des äußeren Referenzmikrofons genützt, der zweite (AIN-) für das innere Fehlermikrofon.

An den benötigten Anschlüssen wurden Drahtleitungen angelötet um die gewünschten Verbindungen realisieren zu können.

Weitere Informationen zum Evaluierungsboard sind dem entsprechenden Datenblatt [11] zu entnehmen.

4.5 Digitaler Signalprozessor

4.5.1 Überblick

Das Herzstück der digitalen Signalverarbeitung ist der ADSP-21489 SHARC Signalprozessor (DSP) von Analog Devices. Dabei handelt es sich um einen 32-bit-Gleitkomma-DSP, der eine Taktung bis 400 MHz erlaubt. Aufgrund ihres Aufbaus sind diese Prozessoren prädestiniert für Audioanwendungen. Es wird kurz auf jene Elemente eingegangen, die für die Umsetzung dieser Arbeit wichtig sind [12].

4.5.2 Signalprozessorkern

In Abbildung 16 sind die wichtigsten Komponenten des SHARC Prozessorkerns dargestellt. Im Wesentlichen umfasst dieser die Rechenelemente, Adressgeneratoren, den *Program Sequencer* und das Bussystem.

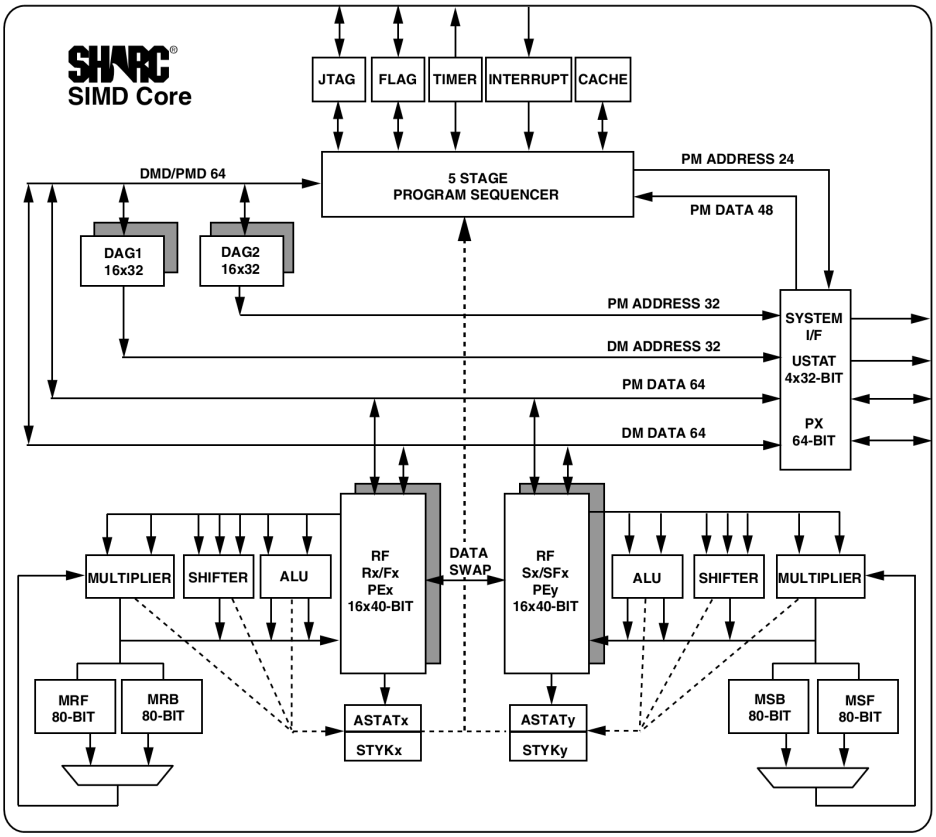


Abbildung 16: Blockdiagramm des SHARC Prozessorkerns

4.5.2.1 Super Harvard Architektur

Der ADSP-21489 SHARC Prozessor basiert auf der sogenannten Super Harvard Architektur. Wie bei der 'klassischen' Harvard Architektur sind Befehl- und Datenspeicher physisch voneinander getrennt und mit dem Prozessor über je einen eigenen Datenbus (DM-Bus) bzw. Befehlsbus (PM-Bus) verbunden, was den gleichzeitigen Zugriff auf die beiden Speicher ermöglicht. Dies erlaubt zwei Zugriffe (Befehl und Operant) innerhalb eines Zyklus. Die Erweiterung besteht nun darin, dass für jeden Speicher ein separater Datenbus als auch Befehlsbus existiert. Der Befehlsspeicher kann daher ebenfalls für Daten genützt werden. Zusätzlich können die Befehle in einem eigenen Cache-Speicher zwischengespeichert werden. Somit sind drei Zugriffe innerhalb eines Zyklus möglich (Befehl aus Cache und zwei Operanten aus dem Speicher). Die Übertragung der Daten vom Datenspeicher in den Befehlsspeicher erfolgt durch Speicherdirektzugriff (oder *Direct Memory Access*, DMA).

Aufgrund dieser Eigenschaft können zum Beispiel die für die Filterungen benötigten Faltungen der ANC-Algorithmen schneller ausgeführt werden, da in einem Schritt neben einer Berechnung ebenfalls ein altes Eingangssample als auch ein Filterkoeffizient geladen werden können.

4.5.2.2 Program Sequencer

Der *Program Sequencer* ist für den Ablauf eines Programms zuständig, indem er ständig die Adresse des folgenden Befehls zur Verfügung stellt. In einem linearen Programmablauf wird dabei immer die aktuelle Adresse inkrementiert und im Befehlspeicher abgelegt. Wird ein linearer Programmablauf unterbrochen, beispielsweise durch einen Funktionsaufruf, eine Schleife oder einen Interrupt, stellt der Program Sequencer die benötigte Adresse zur Verfügung. Per Multiplexer wählt er dazu aus verschiedenen ihm zur Verfügung stehenden Quellen die entsprechende Adresse aus.

4.5.2.3 Datenadressierungsgeneratoren

Ein Datenadressierungsgenerator (DAG) erzeugt Adressen für Daten, welche zwischen dem Befehl- und Datenspeicher und den Systemregistern transportiert werden. Aufgrund der doppelten Busstruktur verfügt der ADSP-21489 über zwei DAGs. Besonders relevant ist die Möglichkeit der zirkulären Adressierung. Dadurch ist die Implementierung von Ringpuffern für effiziente Berechnungen der Faltungsoperationen und des Filterupdates möglich.

4.5.2.4 Rechenelemente

Besonders an die Bedürfnisse von Audioanwendungen angepasst ist die Ausstattung des Prozessorkerns mit zwei Rechenelementen (PEx und PEy). Dadurch sind sogenannte SIMD-Operationen (*Single Instruction Multiple Data*) möglich, bei welchen in einem Zyklus ein Befehl doppelt, also mit zwei verschiedenen Datensets, durchgeführt werden kann. Dies kann z.B. bei der simultanen Berechnung des linken und rechten Kanals einer Stereoquelle genutzt werden und wäre auch bei einer Erweiterung des ANC-Prototypen auf zwei Kanäle sinnvoll. Für den einkanaligen Fall ist ein Betrieb im SISD-Modus (*Single Instruction Single Data*) ausreichend.

Jedes Rechenelement besitzt einen Multiplizierer, einen *Shifter* und eine *Arithmetic Logic Unit* (ALU).

4.5.3 Speicher

Der Prozessor besitzt fünf Mbit internen RAM und vier Mbit internen ROM Speicher, welcher in vier unabhängigen Speicherblöcken organisiert ist. Grundsätzlich kann jeder Block Kombinationen von Daten und Befehlen beinhalten. Für eine optimale Ausnutzung der Super Harvard Architektur ist es jedoch ratsam, wenn der erste Block nur mit Daten belegt ist, die mittels DM-Bus transportiert werden. Im zweiten Block können Daten und Befehle gespeichert sein, welche über den PM-Bus angesprochen werden. Dies erlaubt zwei Speicherzugriffe in einem Zyklus, solange der entsprechende Befehl im Cache liegt.

4.5.4 Digital Peripheral Interface

Das *Digital Peripheral Inteface* (DPI) ist Teil des *Input/Output Processors* und erlaubt die Kommunikation mit externen Komponenten. Es besteht im Wesentlichen aus den Anschlusspins, einer *Signal Routing Unit* (SRU), einem *Interrupt Controller* und Ports für verschiedene Schnittstellen, zum Beispiel einem SPI Port.

4.5.4.1 Signal Routing Unit

Die Aus-bzw. Eingänge externer Komponenten, welche an den Pins des DPI angeschlossen werden, sind nicht fix mit einem entsprechenden Eingang am DSP verbunden. Dazwischen gibt es die sogenannte *Signal Routing Unit*, eine Art "digitaler Patchpay", mit deren Hilfe Eingangssignale bequem an jeden beliebigen Pin des DSPs bzw. jedes Ausgangssignal an einen beliebigen DPI-Pin geroutet werden kann.

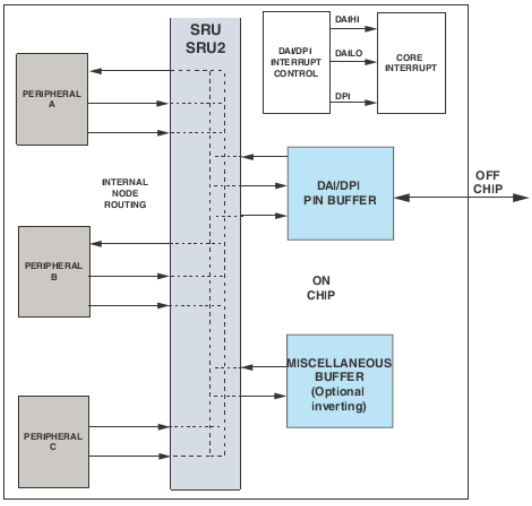


Abbildung 17: Signal Routing Unit

Über sogenannten Pin-Buffer definiert man einen Pin als Eingang oder Ausgang. Wird ein Pin nicht gebraucht, kann er über die Enable-Leitung deaktiviert werden.

4.5.4.2 SPI Ports

Speziell für die Verbindung mit externen Komponenten über die SPI-Schnittstelle verfügt der Prozessor über entsprechende Anschlüsse. Über die SRU müssen alle nötigen Daten- und Taktleitungen an die jeweiligen Ports geroutet werden. Durch spezielle Register können die einzelnen Anschlüsse ihrer gedachten Anwendung entsprechend konfiguriert werden.

Beim ANC-Prototypen wird ein Port zur Übertragung der gewandelten Audiodaten vom ADC zum DSP genutzt, der andere zur Übermittlung der Daten des Antischall-Signals vom DSP zum DAC.

4.5.4.3 Interrupt Controller

Werden Daten vom ADC zum DSP geschickt, muss dem Prozessor mitgeteilt werden, dass Daten vorhanden sind. Dies übernimmt der DPI *Interrupt Controller*. Der Prozessor unterbricht daraufhin seine gerade ausgeführte Tätigkeit und bearbeitet eine dafür vorgesehene Routine.

4.5.5 Precision Clock Generator

Der *Precision Clock Generator* (PCG) ermöglicht es, ein beliebiges Paar von Takt- und Synchronisationssignalen für externe Komponenten zu erzeugen. Dazu wird ein Eingangstakt benötigt, aus welchem sich die benötigten Ausgangstaktsignale ableiten lassen.

Als Eingang kann ein beliebiger, externer Oszillator ebenso gewählt werden, wie die prozessorinterne *Peripheral Clock* (PCLK). Die Ausgangsfrequenz ergibt sich über einen Divisor, der als Integerwert in einem speziellen Kontrollregister angegeben wird und entspricht dem Quotienten aus Eingangsfrequenz und Divisor.

Es ist ebenfalls möglich, die Phase der Ausgangssignale und die Pulsbreite (entspricht der Zeit, die das Taktsignal high ist) beliebig zu ändern.

Beim ANC-Prototyp werden zwei PCGs benötigt, um Startsignale für die externen Umsetzer zu erzeugen. Bei jeder fallenden Flanke wird eine Analog-Digital-Umsetzung gestartet bzw. ein Datenwert analoggewandelt. Die Abtastrate ergibt sich daher aus dem Divisor.

4.5.6 FIR-Hardwarebeschleuniger

Der Vollständigkeit halber soll hier nicht unerwähnt bleiben, dass der ADSP-21489 über einen integrierten Hardwarebeschleuniger für FIR-Filterberechnungen verfügt. Es handelt sich dabei um ein Hardware-Modul, welches FIR Filterberechnungen ohne Belastung des Prozessorkernes ermöglicht, sodass dieser sich um andere Aufgaben kümmern kann. Die speziell auf die Filterstruktur abgestimmte Recheneinheit ermöglicht eine effiziente und zeitschonende Filterberechnung, jedoch nur bei einer maximalen Taktfrequenz von maximal 200 MHz.

Da es im Prinzip für den Prozessor keine andere Aufgabe gibt als die Filterung von Audiosignalen und die Berechnungen auch mittels der prozessoreigenen Rechenwerke effizient und in ausreichender Geschwindigkeit zu erledigen war, wurde auf eine Einbindung des Hardwarebeschleunigers verzichtet. Sollte in Zukunft eine mehrkanalige Ausführung des ANC-Prototyps

gewünscht werden und die Rechenleistung des Prozessors reicht nicht mehr aus, so ist es empfehlenswert die Hardwarebeschleunigung zu nutzen.

4.6 DSP-Evaluierungsboard

4.6.1 Überblick

Beim *ADSP-21489 EZ Board* handelt es sich um ein geeignetes Starterkit für die Verwendung von SHARC Prozessoren. Es stellt eine Vielzahl an externer Hardware zur Verfügung, über welche der Prozessor angesprochen werden kann. In Kombination mit dem Prozessor ergibt sich also ein "Komplettpaket", welches für eine Vielzahl an Anwendungen genutzt werden kann. Für eine genaue Beschreibung sei auf die entsprechende Bedienungsanleitung verwiesen [13]. Es soll aber kurz auf die für diese Arbeit relevanten Komponenten eingegangen werden.

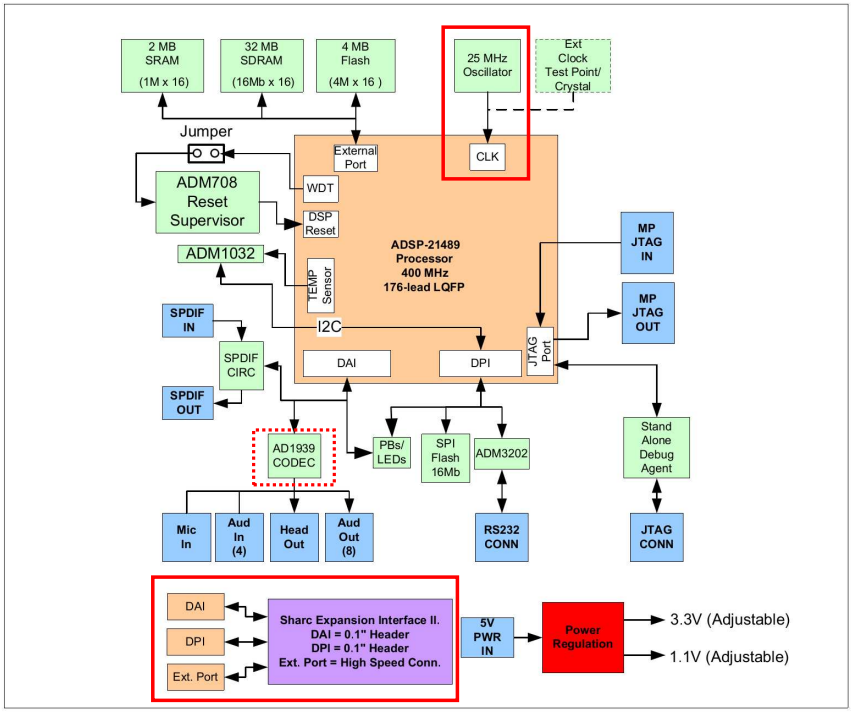


Abbildung 18: AD21489 Evaluierungsboard

4.6.2 Oszillatoren

Das Evaluierungsboard hat zwei Oszillatoren verbaut, die für diese Arbeit verwendet werden.

Zunächst sei der 25 MHz Oszillator erwähnt, über welchen die Taktung des Prozessors realisiert werden kann. Der Ausgang des Oszillators liegt von Werk aus bereits fix am CLK-Eingang des Prozessors an und muss daher nicht über die SRU verbunden werden. Über eine integrierte Phasenregelschleife (PLL) kann der Prozessor die maximale Taktfrequenz von 400 MHz erreichen. Ebenso wird dieser Oszillator für die Taktung des SDRAM Speichers genutzt, was in dieser Arbeit jedoch nicht notwendig ist. Die Startfrequenz, die sich nach dem Einschalten einstellt, wird über den Taktkonfigurationsschalter (SW5) erreicht.

Ein weiterer Oszillator, der speziell für Audioanwendungen vorgesehen ist, findet man beim boardeigenen Codec AD1939. Es handelt sich dabei um einen 24.576 MHz, der als Taktgeber der PCGs fungiert. Durch die gewählte Grundfrequenz kann somit durch einfache Integerdivision ein typische Audio-Abtastfrequenz erzeugt werden. Für die Verwendung des Audiooszillators muss die Drahtbrücke J5 am Board eingesetzt sein. Dadurch liegt das Taktsignal am DAI Pin 17 an und kann über die SRU beliebig geroutet werden.

4.6.3 Expansion Interface II

Das *Expansion Interface II* ist eine Schnittstelle, die Zugriff auf alle Eingänge des SHARC-Prozessors bietet. Über die *J2-Header* genannte Buchsenleiste des Evaluierungsboards ist der Prozessor mit den externen Umsetzern verbunden. Die Anschlüsse wurden als Crimpverbindungen mit Hilfe von entsprechenden Buchsengehäusen realisiert. Abbildung 19 gibt einen Überblick über die benutzten Pins.

Der Massepin wird mit dem digitalen Kreis des DACs verbunden um das selbe digitale Masseniveau auf DSP und DAC sicherzustellen.

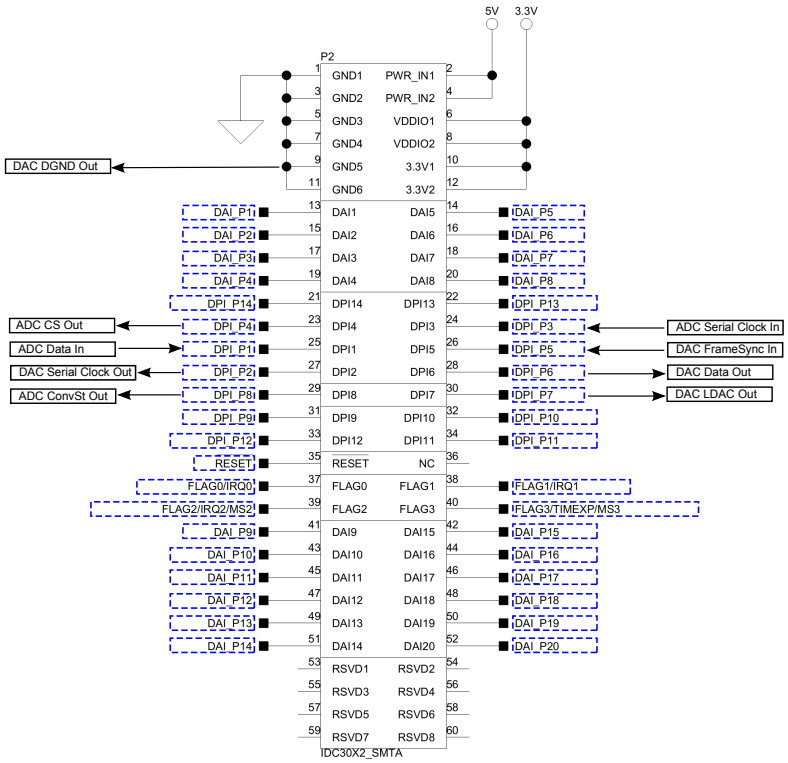


Abbildung 19: J2 Header des Expansion Interface II

4.7 Digital-Analog-Umsetzer

4.7.1 Umsetzer

Zur Digital-Analog-Umsetzung kommt ein AD5764R von Analog Devices zum Einsatz. Dabei handelt es sich um einen vierkanaligen Digital-Analog-Umsetzer (DAC) mit 16 Bit Auflösung, der bis zu einer maximalen Taktfrequenz von 30 MHz arbeitet. Die Daten müssen in einem 24 bit-Format an den Umsetzer geschickt werden, wobei das erste Byte der Adresse des jeweiligen DACs entspricht und die beiden anderen Bytes die eigentlichen Daten darstellen. Zur Wandlung wird dabei ein R2R-Netzwerk eingesetzt. Der Eingang ist seriell ausgeführt und ist kompatibel zur SPI-Schnittstelle. Pufferregister am Eingang erlauben die Zwischenspeicherung der Daten aller vier Kanäle zur gleichzeitigen Wandlung. Das Datenformat ist wählbar zwischen Zweierkomplement und Straight Binary.

Weitere Informationen zum DAC sind der AD5764R-Bedienungsanleitung [14] zu entnehmen.

4.7.2 Evaluierungsboard

Die Einbindung des AD5764R passiert über das Eval-AD5764R Evaluierungsboard. Alle Pins des Wandler-ICs sind direkt über das Board erreichbar. Für die Anbindung der SPI Daten- und Taktleitungen musste eine 14 Pin-Stiftleiste in ein entsprechendes Anschlussfeld (J21) gelötet werden. Ebenfalls kann der Umsetzer per PC über den vorhandenen USB-Port mittels der mitgelieferten AD5764R Evaluation Software gesteuert werden. Dadurch ist es zum Beispiel möglich diverse Register zu modifizieren bzw. auszulesen, die Kerntemperatur zu messen oder den analogen Ausgang zu kalibrieren.

Weitere Informationen zum Evaluierungsboard sind der entsprechenden Bedienungsanleitung [15] zu entnehmen.

4.8 Spannungsversorgung

Die Energieversorgung der externen Umsetzer und der Verstärkerplatine wird von einer eigens dafür angefertigten Spannungsversorgung übernommen. Dazu sind drei Spannungskreise notwendig; ein +12 Volt, ein -12 Volt Kreis und ein + 5 Volt Kreis.

+12 Volt werden einerseits für die Spannungsversorgung des positiven analogen Schaltkreises der beiden Umsetzer, andererseits auch für die Spannungsversorgung der Verstärkerplatine benötigt. Der -12 Volt Kreis ist für die negative analoge Spannung der Umsetzer zuständig. +5 Volt werden zur Versorgung der digitalen Wandlerschaltkreise benötigt.

Die +/- 12 Volt Spannungskreise werden von einem Gleichspannungswandler (DC-DC-Converter) bereitgestellt [16], welcher wiederum von einem +5 Volt Netzteil betrieben wird. Die vom DC-DC-Umsetzer bereitgestellten 125 Milliampere sind für die zu versorgenden Komponenten ausreichend.

4.9 Serial Peripheral Interface

Das *Serial Peripheral Interface* (SPI) ist ein von Motorola entwickeltes synchrones, serielles Bussystem für den Datenaustausch zwischen zwei oder mehreren Teilnehmern. Die Kommunikation ist in beiden Richtungen, also vollduplex, möglich. Es können sehr hohe Übertragungsraten durch die Verwendung von Taktraten bis zu einigen Megahertz erreicht werden. Die Verbindung der jeweiligen Komponenten erfolgt durch das Master-Slave-Prinzip. Pro Transfer sind mehrere Slaves, aber nur ein Master möglich. Der Master ist jener Teilnehmer, der das Taktsignal zur Verfügung stellt und den jeweiligen Slave selektiert.

Üblicherweise werden bei einer Übertragung über SPI vier Leitungen genutzt: eine Steuerleitung für den Takt, eine Datenleitung vom Master zum Slave (*Master Out Slave In*, kurz MOSI), eine weitere vom Slave zum Master (*Master In Slave Out*, kurz MISO) und ein *Slave-Select* bzw *Chip-Select* Leitung (CS). Letztere sind in der Regel *Low-Aktive*, das bedeutet, dass man die jeweiligen Eingänge auf Masse ziehen muss um den Slave zu aktivieren. In einem System mit nur einem Slave kann der Eingang permanent auf Masse gesetzt werden [17].

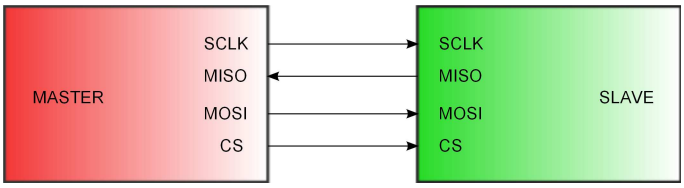


Abbildung 20: Einfache SPI-Verbindung mit zwei Teilnehmern

SPI wird in dieser Arbeit genutzt um die gewandelten Mikrofonsignale zum DSP bzw. das Antischallsignal zum DAC zu führen. Dabei wurden zwei unabhängige Bussysteme erstellt, eines zwischen ADC und DSP, ein weiterer zwischen DSP und DAC. Da die Daten nur in eine Richtung geschickt werden mussten, wurde auf die MISO-Leitungen verzichtet.

5 Identifikation der Sekundärstrecke

5.1 Überblick

Im Abschnitt 2.3.2 wurde bereits auf das Problem der Sekundärstrecke eingegangen. In Kapitel 4 wurden die einzelnen Komponenten genauer beschrieben. Für die Verwendung des FXLMS Algorithmus muss eine Schätzung der Sekundärstrecke vorliegen. Aus diesem Grund muss vor dem Betrieb eine Messung durchgeführt werden und ensprechend der Abbildungen 4 und 5 im Programmcode berücksichtigt werden. Im Folgenden wird das Messverfahren beschrieben.

5.2 Eigenschaften der Sweep-Messung

Die Messung der Sekundärstrecke erfolgte über das Sweep-Verfahren. Diese Methode verwendet ein sinusförmiges Anregungssignal, dessen Frequenz sich exponentiell über der Zeit erhöht, während der Pegel konstant bleibt. Die Antwort auf diese Anregung wird aufgezeichnet und die Impulsantwort zwischen Sende -und Empfangsposition wird mathematisch über eine Entfaltung mittels inversem Sweep ermittelt.

Sweeps weisen einige Eigenschaften auf, welche für die Identifikation von Übertragungsfunktionen besonders geeignet sind:

- Das Spektrum eines einzelnen Sweeps unterscheidet sich in einem nur sehr geringen Ausmaß von dessen periodischer Fortsetzung [18]. Es ist daher nicht notwendig mehrere Durchläufe zu starten und die Ergebnisse zu mitteln.
- Die harmonischen Verzerrungen liegen in der Impulsantwort getrennt von den linearen Anteilen, nämlich zu den negativen Zeitpunkten, vor. Die Impulsantwort bleibt frei von Verzerrungen. Der Grund: durch das Auftreten harmonischer Komponenten entstehen im Spektogramm entsprechende harmonische Kurven links des linearen Anteils. Durch das Entfalten wird die lineare Antwort auf den Zeitpunkt t=0 projiziert. Die harmonischen Komponenten verschieben sich dadurch in den negativen Zeitbereich [19, 20].

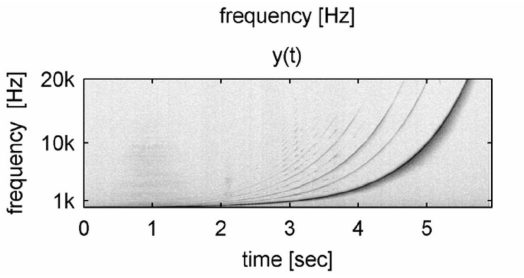


Abbildung 21: Spektogramm eines logarithmischen Sweeps mit Verzerrungen [19]

- Bei der Anregung verteilt sich die Energie auf die einzelnen Sinussignale. Die Messung ist daher energieeffizienter und es ist ein besserer Signal-Rausch-Abstand erreichbar als bei Messungen mit Rauschsignalen, da sich dort die Energie auf das ganze Spektrum aufteilen muss [20].

5.3 Messungen

5.3.1 Vorgangsweise

Das verwendete logarithmische Sweep-Signal umfasst einen Frequenzbereich von 20 Hz bis 20 kHz. Es wurde in Matlab zusammen mit einer invertierten Version zur Entfaltung erzeugt und mittels Pure Data (PD) parallel über zwei Ausgänge einer Soundkarte ausgespielt und wieder aufgenommen. Einer der beiden Ausgänge wird direkt wieder an einen Eingang angelegt und dient als Referenz, also als zeitlicher Bezugspunkt für die Ergebnisse. Der zweite Ausgang durchläuft die gesamte Sekundärstrecke. Der Kopfhörer wird an einem Kunstkopf angebracht um einen möglichst realistische Umgebung zu schaffen.

Beide Signale werden mit dem inversen Sweep entfaltet. Das erste Signal dient dazu, um den Einfluss der Soundkarte aus der Sekundärstrecke zu entfernen. Als Ergebnis liegt somit die reine Impulsantwort der Sekundärstrecke vor.

5.3.2 Latenzmessung der Wandlerkomponenten

Anhand einer Messung wurde die Latenz der Umsetzer und des Signalprozessors überprüft. Da es diese Komponenten sind, welche eine zusätzliche Latenz gegenüber dem analogen ANC verursachen, ist deren Kenntnis von besonderem Interesse.

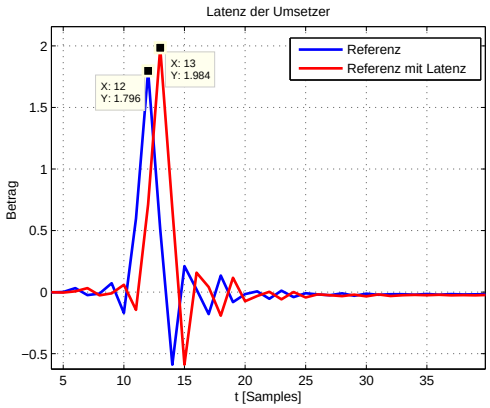


Abbildung 22: Latenz der Wandlerkomponenten

Die Ergebnisse der Messung sind in Abbildung 22 zu sehen. Die Latenz weist einen äußerst geringen Wert von einem Sample bei einer Abtastfrequenz von 48 kHz auf, das entspricht einer Zeit von ca. 20 Mikrosekunden. Prinzipbedingt ist ein besserer Wert nicht möglich. Die Messung bestätigt auf eindrucksvolle Weise die Tauglichkeit der Komponenten für ihren angedachten Zweck.

5.3.3 Impulsantwort und Frequenzgang der gesamten Sekundärstrecke

In Abbildung 23 ist links eine Messung der Impulsantwort der gesamten Sekundärstrecke dargestellt. Es ist möglich, eine grobe Abschätzung über die frequenzabhängige Gruppenlaufzeit zu treffen. Man erkennt, dass eine Zeit von ca. zwei Millisekunden (100 Samples bei 48 kHz) notwendig ist, bis der gesamte Anteil der Signalenergie übertragen worden ist. Im tieffrequenten Bereich kommt es zu größeren Laufzeitverzögerungen. Die ersten Signalanteile durchlaufen die Sekundärstrecke hingegen ab bereits ca. 0,16 Millisekunden (8 Samples).

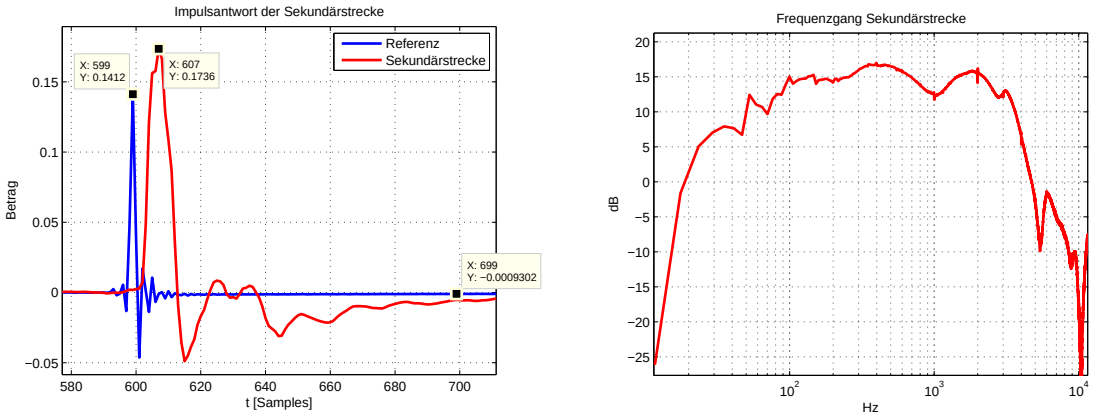


Abbildung 23: Impulsantwort und Frequenzgang der Sekundärstrecke

5.3.4 Einschränkungen

Es ist zu beachten, dass die Sekundärstrecke jedes Mal neu berechnet werden muss, wenn sich die Position des Kopfhörers am Kunstkopf ändert. Dies ist zum Beispiel der Fall, wenn der Kopfhörer entfernt und neu angebracht wird oder der Kopfhörer am Kunstkopf verrutscht. Durch die Positionsänderung schließt die Muschel anders ab, es ergeben sich unterschiedliche Schalldruckeigenschaften im Inneren des Kopfhörers, was einen Einfluss auf den Frequenzgang hat.

Der Vollständigkeit halber sollte nicht unerwähnt bleiben, dass die Messung mittels Kunstkopf nicht als ideal betrachtet werden darf. Unterschiedliche Kunstköpfe bzw. Ohr-Simulatoren weisen eine verschiedene Dichteeigenschaften auf, was zu einer Abweichung der Messergebnisse zwischen den Modellen führt [21].

6 Programmcode - Implementierung in einer Echtzeitumgebung

6.1 Entwicklungsumgebung

Der gesamte Programmcode wurde in Visual DSP++ 5.0 verfasst . Es handelt sich dabei um eine interaktive Entwicklungsumgebung (IDE) von Analog Devices speziell für SHARC-, TigerSHARC- und Blackfin-Prozessoren. Es bietet sämtliche Funktionen um vollständige DSP Projekte zu erstellen und debuggen:

- Ein C/C++ Compiler für die Verwendung von C/C++ Code inklusive Laufzeitbibliothek. Der Compiler übersetzt den in der jeweiligen Hochsprache verfassten Quelltext in Maschinencode. Aus den einzelnen C/C++ Datein werden verlinkbare *".doj"* - Objektdateien generiert. Die Laufzeitbibliothek erlaubt die Verwendung von Funktionen zur Zeit der Ausführung des Programms [22]. Detaillierte Informationen zum Compiler und zur Laufzeitbibliothek sind den "VisualDSP++ 5.0 C/C++ Compiler Manual for SHARC Processors" [23] und "VisualDSP++ 5.0 Run-Time Library Manual for SHARC Processors" [24] zu entnehmen.
- Ein Assembler zur Erstellung eines Programms in Maschinensprache. Dieser wandelt die prozessoreigene Maschinensprache in Maschinencode um und erzeugt ebenfalls verlinkbare *".doj"* - Objektdateien. Genauere Informationen sind dem "VisualDSP++ 5.0 Assembler and Preprocessor Manual " [25] zu entnehmen.
- Ein Linker zur Erstellung einer ausführbaren Datei (*Executable*): Der vom Compiler bzw. Assembler erzeugte Maschinencode ist nicht ausführbar, da er teilweise noch aus dem Sourcecode übernommene symbolische Namen (z.B. Funktionsaufrufe) enthält. Der Prozessor versteht aber nur Adressen. Der Linker ist für die sogenannte Referenzauflösung veantwortlich, also das Ersetzen von Objektnamen durch Adressen. Die vorhandenen *".doj"* - Objektdateien werden mit Hilfe der benötigten Bibliotheken zu einem tatsächlich vom Rechner ausführbaren Programm gebunden [22]. Genauere Informationen sind dem "VisualDSP++ 5.0 Linker and Utilities Manual " [26] zu entnehmen.
- Ein Debugger zur Fehlersuche durch z.B. schrittweise Ausführung des Programms, Benutzung von Anhaltepunkten oder Visualisierung von Speicherbereichen.
- Ein Simulator zum Arbeiten am Quellcode und Tests ohne angeschlossenes DSP-Board.

Für das Programm wurde sowohl C- als auch Assemblersprache verwendet. In C wurden die Initialisierungen der Hardware und der Datentransfer über die SPI-Schnittstelle verfasst. Die zeitkritischen Berechnungen der ANC-Algorithmen wurden in Assembler geschrieben. Ein Teil der Initialisierungen wurden aus dem in den Programmbeispielen mitgelieferten "21489 AD1939 C Sampled-Based Talkthru 48 or 96 kHz" Programm übernommen.

6.2 Prinzipieller Funktionsablauf

6.2.1 Taktung

Es soll nun ein Überblick über die Programmierung des Kommunikationssystems gegeben werden. Der Datenaustausch erfolgt über die beiden SPI Ports des SHARC Prozessors:

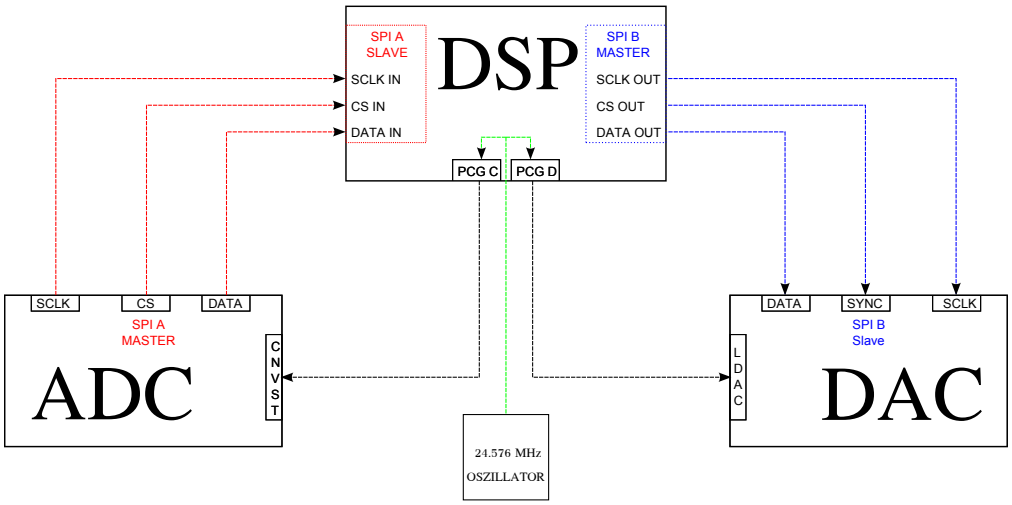


Abbildung 24: Prinzip des Kommunikationssystems

Der Start der Analog-Digital-Umsetzung wird vom Prozessor über ein vom PCG C generiertes Taktsignal erzeugt. Als Eingang dient dem PCG der 24.576 MHz Oszillator des SHARC Evaluierungsboards. Durch einen Divisor von 512 wird ein 48 kHz Taktsignal erzeugt, welches am CNVST Pin des Umsetzers anliegt. Über den Divisor ergibt sich also die Abtastrate. Jede negative Flanke des Taktsignals initiiert eine simultane Wandlung beider Eingangskanäle. Die Wahl der Abtastrate hat keinen Einfluss auf die Latenz, da die Daten unabhängig davon immer mit der gleichen Geschwindigkeit zum DSP geschickt werden. Es müssen jedoch Abschätzungen in Zusammenhang mit der verwendeten Filterlänge und dem Berechnungsaufwand getroffen werden. Alle Berechnungen des ANC-Algorithmus müssen zwischen zwei Abtastzeitpunkten geschehen.Wird die Abtastfrequenz erhöht, muss die Puffergröße für die jeweilige Faltung ebenfalls vergrößert werden um die gleiche Frequenzauflösung zu gewährleisten. Dadurch steigt wiederum die Berechnungszeit, welche von der Periodendauer des Abtastsignals eingeschränkt wird. Zu niedrig darf die Abtastfrequenz aber auch nicht gewählt werden, da aufgrund des flachen Anti-Aliasing-Filters die Nyquistfrequenz entsprechend hoch sein muss. Die gewählte Abtastrate von 48 kHz stellt einen guten Kompromiss dar.

Der SPI Port A wird zur Verbindung von ADC und DSP verwendet. Der Umsetzer übernimmt dabei die Rolle des Master. Ist die Wandlung abgeschlossen, werden die 32 Bit Daten beider Kanäle über den SDOUT Pin zum MOSI Dateneingang des Prozessors geschickt. Der Umsetzer erzeugt das notwendige Takt- und CS-Signal selbst und stellt diese über die SCLK bzw. CS Leitungen dem DSP zur Verfügung. Der Datentransport beginnt mit dem MSB des ersten Kanals.

Die Übertragung der DSP-Ausgangsdaten geschieht über den SPI Port B. Der DSP fungiert diesmal als Master und stellt neben den Daten auch sämtliche Steuerleitungen zur Verfügung. Analog zum CNVST Signal des AD-Umsetzers benötigt der DAC auch ein entsprechendes Signal, welches für den Start bzw. die Rate der digital-analog-Wandlung zuständig ist. Dieses sogenannte *Load DAC* (LDAC) Signal wird, ähnlich dem CNVST des ADCs, über die PCG D erzeugt. Der

Unterschied liegt in der Wahl des Divisorwerts, also in der Wandlungsrate: Um eine niedrige Latenz zu erreichen, muss die fallende Flanke des LDAC-Signals möglichst schnell nach der Übermittlung der Daten an den DAC erfolgen. Je größer die Zeit zwischen den fallenden Flanken, umso länger dauert es, bis die Wandlung stattfindet. Kommt eine Flanke, während der DAC keine gültigen Daten vorliegen hat, ignoriert dieser die Flanke. Der gewählte Divisorwert wurde auf 128 gesetzt und entspricht damit einer Wandlungsrate von 192 kHz, also einer Überabtastung um den Faktor vier.

6.2.2 Interrupt-gesteuerter Programmablauf

Unter Interrupt versteht man die Unterbrechung des sequenziellen Programmablaufs durch eine Unterbrechungsanforderung (*Interrupt Request*) und das Ausführen einer speziellen Unterbrechungsroutine (*Interrupt Service Routine*). Da der gesamte Ablauf über Interrupts gesteuert wird, befindet sich der Prozessor erstmal im Ruhezustand (*Idle-Mode*). Der Eingang der Daten vom ADC am Prozessor stellt die Unterbrechungsanforderung dar. Der Dateneingang wird vom I/O-Prozessor detektiert und dem Prozessor vermeldet. Dieser startet daraufhin die entsprechende Serviceroutine, welche aus dem Auslesen der Daten, der Berechnung des Antischalls und der Übertragung der Ausgangsdaten über den SPI B Port besteht.

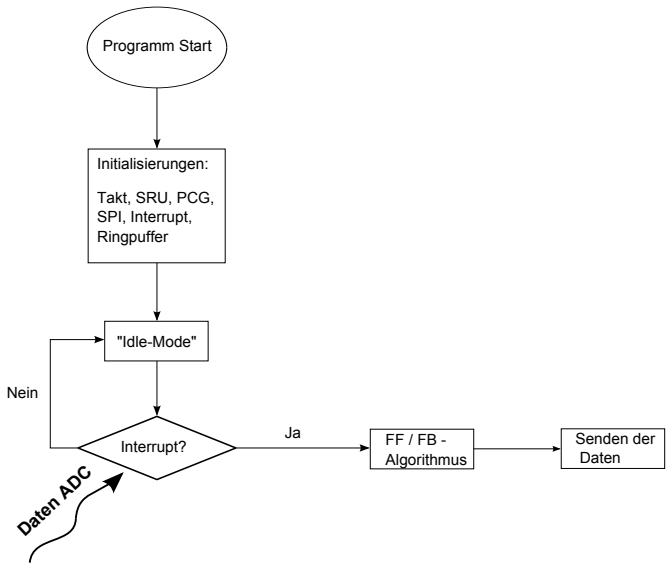


Abbildung 25: Programmablauf

Die Adresse der Serviceroutine steht in der sogenannten Interruptvektortabelle. Beim Auftreten eines Interrupts ist der Program Sequencer dafür zuständig die entsprechende Adresse in den Befehlsspeicher zu laden.

6.3 Beschreibung des Programmcodes

6.3.1 Initialisierungen

Um den Prozessor für die vorgesehenen Aufgaben vorzubereiten müssen einige Initialisierungsmaßnahmen getroffen werden. Diese werden allesamt beim Start von der "_main" Funktion aufgerufen, welche sich in der Datei 'main.asm' befindet. Sie stellt den Einsprungpunkt des Programms dar.

Phasenregelschleife: Die Funktion "_initPLL_SDRAM" der "init_PLL_SDRAM.c" Datei wurde vom Beispielprogramm "21489 AD1939 C Sampled-Based Talkthru 48 or 96 kHz" übernommen und dient in erster Linie der Initialisierung des internen Taktsignals. Die PLL nimmt als Eingangssigal den boardeigenen 25 MHz Oszillator und erzeugt daraus die maximale Taktfrequenz von 400 MHz. Ebenfalls wird das Taktsignal für die Kommunikation mit dem externen SDRAM-Speicher erzeugt. Dieser wird zwar nicht genutzt, soll aber aufgrund etwaiger Programmerweiterungen dennoch zur Verfügung stehen.

SRU: Die Initialisierungen der SRU dient dem Routing aller aus- und eingehenden Signale. Zunächst werden (des sauberen Programmierstils wegen) über die Funktion "clearpins" alle DAI und DPI Eingänge deaktiviert und als Ausgänge definiert. Das eigentliche Routing passiert in der Funktion "initSRU":

- Das am DAI Pin 17 anliegende Taktsignal des 24.576 MHz Audiooszillators wird zu den beiden verwendeten PCGs geschickt. Die Ausgänge der PCGs werden an die entsprchenden Eingänge des ADC bzw. DAC geschickt.

```
////////////////////////////////////////////////////////////////
// Create 192 kHz sampling puls
////////////////////////////////////////////////////////////////

SRU(DAI_PB17_O, PCG_EXTC_I);            // 1939-osc input 24.576 MHz
SRU(PCG_FSC_O,  DPI_PB07_I);        // PCG_C for LDAC OUT

SRU(DAI_PB17_O, PCG_EXTD_I);            // 1939-osc input 24.576 MHz
SRU(PCG_FSD_O,  DPI_PB08_I);        // PCG_C for CONVST OUT

SRU(HIGH, DPI_PBEN07_I);                  //define DPI Pin 7 is an output
SRU(HIGH, DPI_PBEN08_I);                        //define DPI Pin 8 is an output
```

- Die Datenleitung und Steuerleitungen vom ADC werden mit dem SPI Port A verbunden.

```
// Connect ADC
SRU(SPI_MOSI_O, DPI_PB01_I);     //Connect MOSI to DPI PB1.
SRU(SPI_CLK_O, DPI_PB03_I);      //Connect SPI CLK to DPI PB3.
```

- Die Datenleitung und Steuerleitungen vom DAC werden mit dem SPI Port B verbunden.

```
// Connect DAC
SRU(SPIB_CLK_O, DPI_PB02_I);     //Connect SPI CLK to DPI PB3.
SRU(SPIB_MOSI_O, DPI_PB06_I);    //Connect MOSI to DPI PB1.
SRU(SPIB_FLG0_O, DPI_PB05_I);    //Connect MOSI to DPI PB1.
SRU(HIGH, DPI_PBEN02_I);                 //define DPI Pin 2 is an output
SRU(HIGH, DPI_PBEN05_I);           //define DPI Pin 5 is an output
SRU(HIGH, DPI_PBEN06_I);                 //define DPI Pin 6 is an output
```

Beide Funktionen befinden sich in der "initSRU.c" Datei.

SPI: Über die Funktion "_initSPI" in der "initSPI.c" Datei werden beide SPI Ports konfiguriert. Dies geschieht im Wesentlichen über die entsprechenden SPI Kontrollregister (SPICTL & SPICTLB)

- Port A dient der Kommunikation mit dem ADC. Der DSP läuft im *Slave Receive*-Modus. Bei der Konfiguration des SPICTL Registers wird im Wesentlichen der SPI Port A dem Datenformat des ADCs angepasst (siehe Abbildung 26). Die gewählte Wortlänge von 32 Bit entspricht der Länge zweier Kanäle. Die gesetzten *MSBF*- und *CPhase*-Flags teilen dem Prozessor mit, dass das erhaltene Datenwort mit dem MSB beginnt und sich deren jeweiligen Bits mit der steigenden Flanke des Taktsignals ändern. Durch Setzen des *DMISO*-Flags wird der Datenausgang des Slaves deaktiviert, da keine Daten vom DSP zum ADC zurückgeschickt werden müssen. Zum Schluss wird der SPI Port über das *SPIEN*-Flag aktiviert.

```
// configuration of SPI port A as slave receiver
*pSPICTL = DMISO | SPIEN | WL32| MSBF | CPHASE;
```

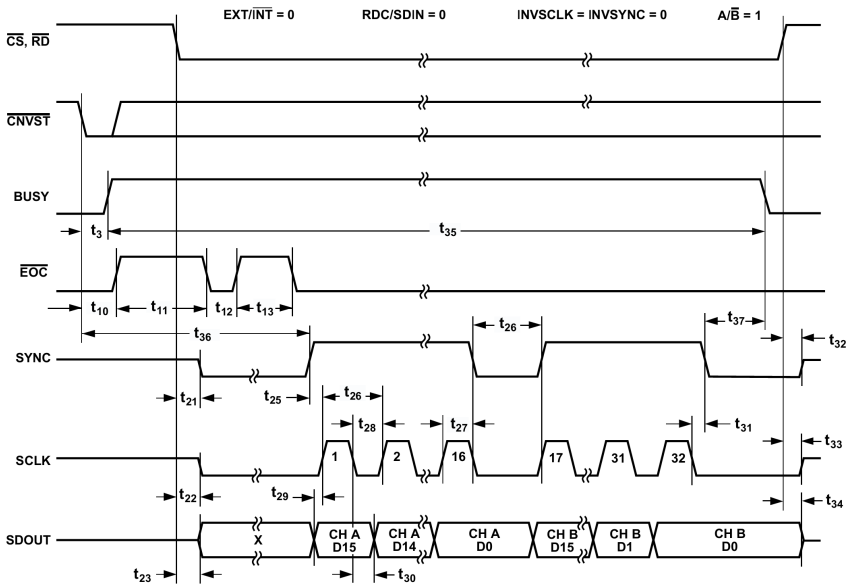


Abbildung 26: Spezifiktionen des ADC

- Port B dient der Kommunikation mit dem DAC. Der DSP läuft im *Master Transmit*-Modus. Die Symbolrate der gesendeten Datenwörter wird über das SPIBAUDB Register auf 25 MHz eingestellt.

```
*pSPIBAUDB = 0x2;       //SPICLK 25 MHZ
```

Das SPI Flag wird über das SPIFLGB Register konfiguriert und entspricht dem Status der SPI CS Leitung. Über die SRU wird die Leitung an den SYNC-Eingang des DACs gelegt. Durch das Setzen des Flags wird die Leitung high gesetzt, das angeschlossene Slave also deaktiviert. Über das *DS0EN* Flag wird der SPI CS-Pin für die Verwendung zur Geräteaktivierung eingeschaltet.

```
*pSPIFLGB = (0xF00|DS0EN);      //Disable SPI Flag0 Output
```

Um das Datenformat des DACs (siehe Abbildung 27) erreichen zu können, sind einige Überlegungen notwendig: Da es sich beim AD5764R um einen vierkanaligen DAC handelt, muss den 16 Bits an Daten noch eine 8 Bit Adresse für einen der vier DACs vorangestellt werden. Die sich ergebende Größe von 24 Bits ist mit dem SPI Port nicht kompatibel, da dieser nur Übertragungen von 8, 16 und 32 Bit-Wörtern zulässt. Aus diesem Grund wurde eine Wortlänge von 8 Bit gewählt. Durch Setzen der *CPhase*-, *TIMOD1*- und *SMLS*-Flags wird ein übergangsloser Transfer möglich. Das bedeutet, dass mehrere 8 Bit Datenwörter ohne Unterbrechung übertragen werden können. Das Datenwort für den DAC muss nun in zwei jeweils 8 Bit Wörter aufgespalten und zusammen mit dem Adressbyte mittels übergangslosen Transfers an den DAC geschickt werden (siehe Abschnitt 6.3.7). Über dem *CLKPL*-Flag wird die Polarität des Taktsignals entsprechend den DAC-Spezifikationen an das Datenwort angepasst. Das *SPIMS*-Flag definiert den Port als Master, der durch das *SPIEN*-Flag aktiviert wird.

```
//configuration of SPI port B as master transmitter
*pSPICTLB = (SPIEN | SPIMS | WL8 | MSBF | TIMOD1 | CPHASE | CLKPL | SMLS | GM);
```

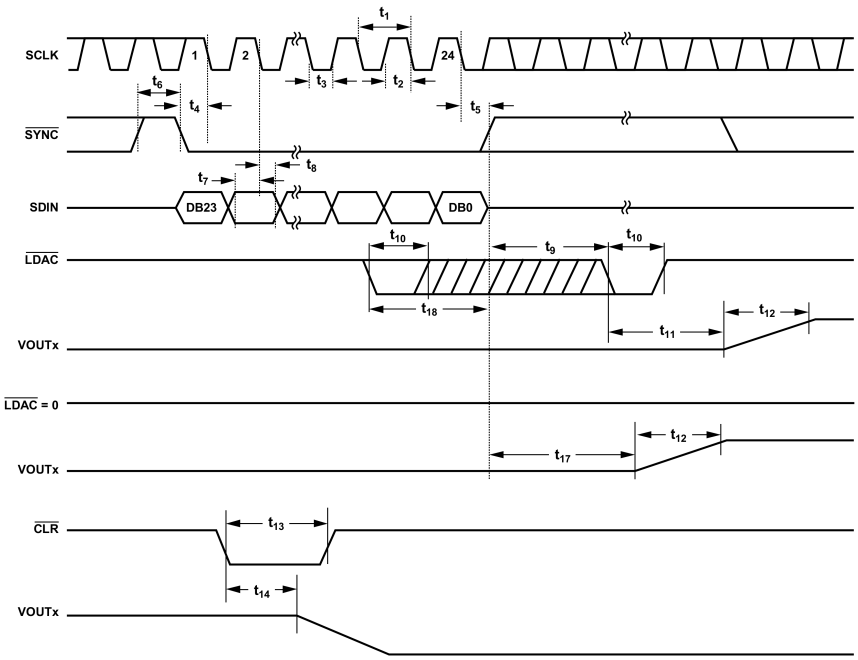


Abbildung 27: Spezifikationen des DAC

PCG: Über die Funktion "_initPCG" in der "initPCG.asm" Datei werden die geeigneten Startsignale zum Start der Wandlungen erzeugt. Über die beiden Kontrollregister PCG_CTLx0 und PCG_CTLx1 werden die entsprechenden PCGs konfiguriert. Im Ersteren wird über den Divisor die Taktfrequenz eingestellt und der Ausgang aktiviert. Das Setzen des *FSxSOURCE*-Flags erlaubt die Verwendung einer externen Quelle, in diesem Fall des 24.576 MHz Audiooszillators, welcher über die SRU an den Eingang der PCG geroutet werden kann. Anderenfalls liegt der Eingang fix am CLKIN-Pin, also am 25 MHz Oszillator.

```
// Configure Samplerate  / PCG_C for CONVST/LDAC

ustat1 = ENFSC | 128;   //192 kHz LDAC
dm(PCG_CTLC0) = ustat1;

        ustat1 = ENFSD |512;    //192 kHZ CONVST
        dm(PCG_CTLD0) = ustat1;

        // Assign 24.576 kHz Oscillator to PCG
        ustat1 = FSCSOURCE ;//| 0x1E00000;
                dm(PCG_CTLC1) = ustat1;
                nop;

                ustat1 = FSDSOURCE;
                dm(PCG_CTLD1) = ustat1;
                nop;
                // Change Pulswidth
                ustat1 = 0x1007D;       //for 192 kHz Samplerate
                //ustat1 = 0x100FE; //for 96 kHz Samplerat
                //ustat1 = 0x101FE; //for 48 kHz Samplerate
```

```
//ustat1 = 0x103FC; //for 24 kHz Samplerate
//ustat1 = 0x10FFE; //for 6 kHz Samplerate

dm(PCG_PW2) = ustat1;
```

Interrupt: Die Aktivierung des Interrupts geschieht in der "_initInterrupt" Funktion in der "main.asm" Datei. Dazu muss im IMASK-Register der SPI-Interrupt demaskiert und im MODE1-Register Interrupts global aktiviert werden.

```
bit set imask SPIHI;                          // Unmask SPI RX ISR Interrupt
bit set mode1 IRPTEN | CBUFEN;          // Unmask Interrupts globally
```

Ringpuffer: Für eine effiziente Filterberechnung wurden die Daten und Filterkoeffizienten in separaten Ringpuffern organisiert. Aufgrund der Busstruktur der Super Harvard Architektur können die Filterkoeffizienten im Programmspeicher und die Eingangsdaten im Datenspeicher abgelegt werden (siehe Abschnitt 4.5.2.3). Die unterschiedlichen Speicherbereiche werden mit Hilfe beider DAGs angesprochen. Jede DAG erlaubt die Nutzung von acht Ringpuffern, welche über folgende Systemregister konfiguriert werden:

- **Basisregister (B0 bis B7 für DAG1, B8 bis B14 für DAG2).** Sie stellen die Startadresse des jeweiligen Ringpuffers dar.
- **Indexregister (I0 bis I7 für DAG1, I8 bis I14 für DAG2).** Ein Indexregister hält eine Adresse und wirkt wie ein Pointer auf den Speicherplatz. Über diese Register werden Lese- bzw. Schreibzugriffe auf den Speicher durchgeführt. Wird das Indexregister nicht anders initialisiert, hält es zu Programmstart die Basisadresse.
- **Modifizierungsregister (M0 bis M7 für DAG1, M8 bis M14 für DAG2).** Sie stellen die Schrittweite dar, die der Pointer (sprich das Indexregister) nach dem Beschreiben oder Auslesen der Speicherzelle inkrementiert bzw. dekrementiert. Der Pointer muss daher nicht durch einen zusätzlichen Befehl auf eine andere Stelle gesetzt werden. Abbildung 28 zeigt ein Beispiel eines Ringpuffers mit einem Inkrement von vier.

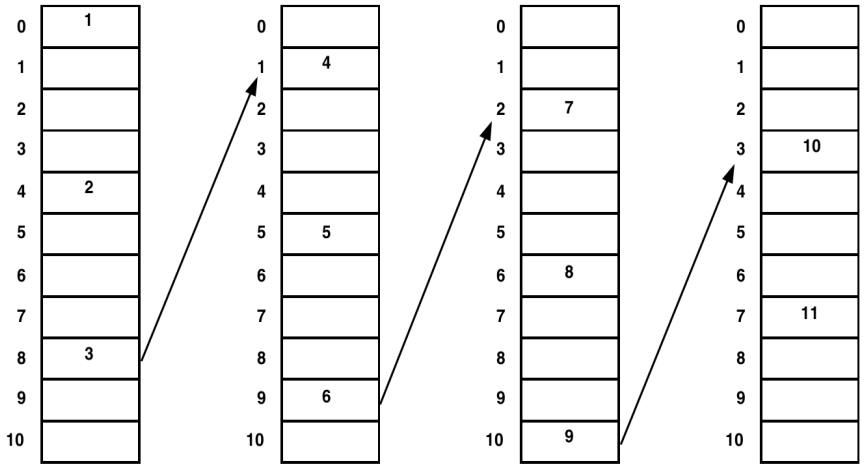


Abbildung 28: Ringpuffer mit M = 4

- **Längenregister (L0 bis L7 für DAG1, L8 bis L14 für DAG2).** Durch diese Register wird die Länge des jeweiligen Ringpuffers konfiguriert.

Über die Funktion "_init_ff_ringbuffer" in der "algorithm.asm" Datei werden die Ringpuffer entsprechend dem Feedforward-Algorithmus konfiguriert, die Funktion "_init_fb_ringbuffer" in der "FBalgorithm.asm" Datei ist für die Konfuguration des Feedback-Algorithmus verantwortlich.

Beispiel für die Konfigurierung eines Ringpuffers:

```
//Reference Mic Read Buffer
b0 = xout;
l0 = @xout;
m0 = 1;
```

Je nach dem, welcher Algorithmus berechnet werden soll, ist die entsprechende Funktion zu wählen. Die geschieht über die Definition der Konstante "USE_FF_ALGORITHM" in der "main.asm" Datei. Die Definition der Konstanten bewirkt die Verwendung des Feedforward-Algorithmus. Ein Fehlen dieser Definition entspricht der Wahl des Feedback-Algorithmus. Die If-Abfrage des Präprozessors überprüft nun, ob die Konstante definiert wurde und ruft die entsprechende Funktion auf.

Voraussetzung für die Nutzung von Ringpuffern ist die Aktivierung der zirkulären Adressierung durch Setzen des *CBUF*-Flags im MODE1-Register, was zugleich mit der globalen Aktivierung der Interrupts in der Funktion "_initInterrupt" passiert.

6.3.2 Interruptvektortabelle

In der Datei "21489_IVT" befindet sich die Interruptvektortabelle. Sie wurde ebenfalls aus dem "21489 AD1939 C Sampled-Based Talkthru 48 or 96 kHz" Beispielprogramm übernommen, jedoch auf die Verwendung von SPI-Interrupts angepasst.

Alle Interruptquellen des Prozessors sind in der Vektortabelle aufgelistet und entsprechend einer festgelegten Priorität geordnet. Die Priorität entscheidet darüber, welche Serviceroutine beim Auftreten mehrerer Interrupts zuerst ausgeführt wird. Aus diesem Grunde gibt es für SPI-Interrupts eine Option mit hoher (SPIHI) und eine mit niedriger Priorität (SPILI). Es wurde hier die Option mit der höheren Priorität, also der SPIHI-Interrupt gewählt.

Eine wesentliche Erweiterung stellen die Präprozessor-Anweisungen dar. Analog zur Konfiguration der Ringpuffer (siehe Abschnitt 6.3.1) wird durch die Definition der Konstante "USE_FF_ALGORITHM" die Verwendung des Feedforward-Algorithmus gewählt. Ein Fehlen dieser Definition entspricht der Wahl des Feedback-Algorithmus. Beide Algorithmen sind in unterschiedlichen Serviceroutinen geschrieben worden (siehe Abschnitt 6.3.5 und 6.3.6).

Die If-Abfrage des Präprozessors überprüft, ob die Konstante definiert wurde. Das Ergebnis der Abfrage entscheidet darüber, welche Adresse in der Vektortabelle aufscheint und welche Serviceroutine infolgedessen vom Compiler aufgerufen wird.

```
__SPIHI:
__P1I :          // 0x30: SPI transmit or receive (higher priority option) __SPII :

                #ifndef USE_FF_ALGORITHM
                jump _FB_algorithm;
                #else
                jump _FF_algorithm;
```

6.3.3 ADDS_21489_EzKit.h

Die ADDS_21489_EzKit.h Headerdatei enthält die Deklaration aller Funktionen und globalen Variablen, die im gesamten Quelltext Verwendung finden. Sie können sowohl im C- als auch im Assemblercode eingebunden werden. Bei der Verwendung in Assemblersprache muss den jeweiligen Namen ein Unterstrich vorangesetzt werden. Eine Variable namens "test" in C/C++ wird in Assembler also zu "_test".

Die Variablen bzw. Funktionen werden zur Verwendung in den verschiedenen Dateien mittels dem ".extern" Befehl eingebunden.

6.3.4 Das Memory Mapping

Unter *Memory Mapping* versteht man die Organisation des Arbeitsspeichers entsprechend der deklarierten Variablen, Funktionen und Routinen.

Einen Überblick über die Speicherarchitekur bekommt man über den *Expert Linker*. Dazu genügt ein Doppelklick auf die "ADSP-21489_ASM.ldf" Datei, das sogenannte *Linker Description File* (LDF), im VisualDSP++ Projektfenster. Hier kann man gut erkennen, wie der physische Speicherbereich innerhalb der einzelnen Blöcke in unterschiedlichen Segmente organisiert ist. Der gesamte Quellcode wird vom Linker auf diese Segmente aufgeteilt. Dazu muss in jeder Assemblerdatei das entsprechende Segment für den verfassten Quelltext vom Programmierer/von der Programmiererin angegeben werden. Für Code in C/C++ ist dafür der Compiler verantwortlich.

Wie in Abschnitt 4.5.3 bereits erwähnt wurde, müssen Daten und Befehle in unterschiedlichen Speicherblöcken abgelegt sein um Nutzen aus der Super Harvard Architekur zu ziehen. Daher wurde der Puffer für die Eingangsdaten im Segment "seg_dmda" des Block 1 RAM Speichers abgelegt. Da es sich dabei um einen Datenspeicher handelt, kann auf dieses Segment über den DM-Bus zugegriffen werden. Die Filterkoeffizienten werden im Segment "seg_pmda" des Block 2 RAM Speichers abgelegt, welcher einen Programmspeicher darstellt und über den PM-Bus

angesprochen werden kann.
Das verwendete LDF entstammt dem "21489 AD1939 C Sampled-Based Talkthru 48 or 96 kHz" Beispielprogramm, wurde aber durch das Hinzufügen von Quelltext entsprechend konfiguriert.

6.3.5 Implementierung des Feedforward-Algorithmus

Wurde in den "21489_IVT.c" und "main.asm" Dateien die Konstante "USE_FF_ALGORITHM" definiert, ist der Feedforward-Algorithmus ausgewählt. Jeder SPIHI-Interrupt führt nun zur Ausführung der "_FF_algorithm" Serviceroutine in der "algorithm.asm" Datei (siehe Abschnitt 6.3.2).

6.3.5.1 Definition der benötigten Variablen und Puffer

Zu Beginn der "algorithm.asm" Datei werden alle benötigten lokalen Variablen und Puffer gemäß Abbildung 29 definiert und damit in den jeweiligen Speichersegmenten angelegt.

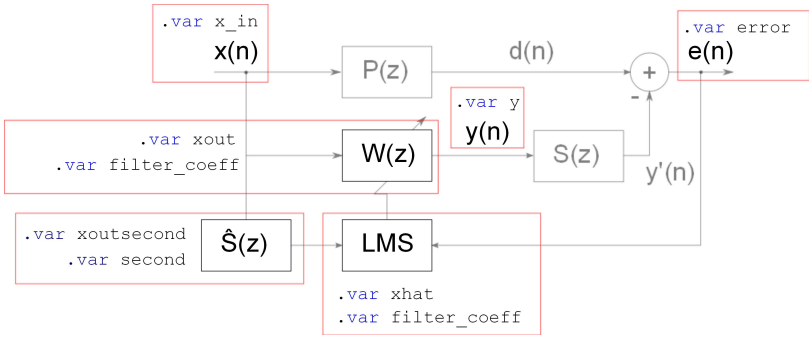


Abbildung 29: Variablen und Vektoren für Feedforward-Algorithmen

Für die Eingangsdaten werden insgesamt drei Puffer benötigt: "xout" hält die Eingangdaten für die Faltung des Referenzsignals mit dem adaptiven Filter, "xoutsecond" jene für die Faltung derselben mit der geschätzten Sekundärstrecke und "xhat" hält die mit der geschätzten Sekundärstrecke gefilterten Eingangsdaten für das Koeffizientenupdate. Sie werden im Segment "seg_dmda" angelegt und über den DM-Bus angesprochen.

Weiters werden zwei lokale Variablen dazu verwendet, um die Eingangsdaten in den jeweiligen Puffer zu übertragen. Für das Signal des Referenzmikrofons wurde dazu die Variable "x_in" erzeugt, für das Fehlermikrofon die Variable "error". Die Variablen "index", "index2" und "index3" halten Adressen für die Pointer und werden für die Faltungen bzw. das Koeffizientenupdate benötigt um den richtigen Einsprungpunkt im Puffer sicherzustellen.

```
section/dm seg_dmda;

.var x_in;
.var xout[BLKSZ];
.var xoutsecond[BLKSZ2]
.var error;
```

```
.var index;
.var index2;
.var index3;
.var xhat[BLKSZ];
```

Im Segment "seg_pmda" sind jene Puffer abgelegt, die die für die Faltungen bzw. das Filterupdate notwendigen Filterkoeffizienten halten. Im Puffer "filter_coeff" liegen die Koeffizienten des adaptiven Filters, "second" hält jene der geschätzten Sekundärstrecke. Beide werden über den PM-Bus angesprochen.

Die Initialisierung der Filter wird über eigene .dat-Dateien erledigt. Diese finden sich im VisualDSP++ Projektfenster: "IRsecond.dat" hält die Koeffizienten der geschätzten Sekundärstrecke, über die "coeffs.dat" Datei wird das adaptive Filter mit Nullen initialisiert.

```
.section/pm seg_pmda;

.var filter_coeff[BLKSZ] = "coeffs.dat";
.var second[BLKSZ2]= "IRsecond.dat";
```

Im Folgenden werden die einzelnen Schritte des Programmcodes ausführlich beschrieben.

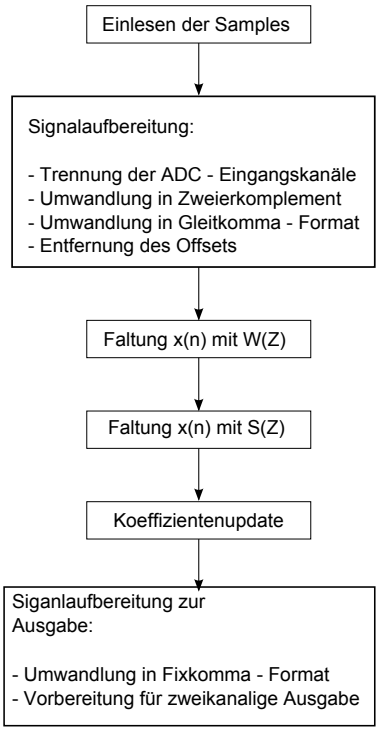


Abbildung 30: Programmschritte des Feedforward-Algorithmus

6.3.5.2 Einlesen und Aufbereitung der Eingangsdaten

Die 32 Bit Eingangsdaten beider Kanäle des AD-Umsetzers gelangen an den MOSI Pin des SPI Port A und werden mit Hilfe des bereitgestellten Taktsignals in das sogenannte Eingangsschieberegister (*Input Shift Register*) geschoben. Sobald ein ganzes Datenwort (entsprechend der definierten Länge im SPI Kontrollregister) empfangen wurde, wird es automatisch in ein auslesbares Empfangsregister, das RXSPI Register, transferiert. Über den Datenbus wird das Wort nun zur weiteren Verarbeitung in das prozessoreigene Systemregister *R1* geladen.

```
//Load Samples (2x16bit) out of Receive-Buffer
r1 = dm(RXSPI);
```

Die Daten beider Kanäle liegen nun in dem Register vor, und dies im entsprechendem Datenformat des AD-Umsetzers. An den 16 höchstwertigen Stellen liegt der erste, dahinter der zweite Kanal. Da der Umsetzer, wie in Abschnitt 4.4.1 bereits beschrieben, nur unipolare Signale wandelt, liegen die Eingangsdaten rein positiv im Straight Binary-Format vor. Daher ist eine Umwandlung in ein bipolares Signal, sprich Zweierkomplement, erforderlich, was ganz simpel erreicht werden kann. Wie in Abbildung 31 leicht für den 8 Bit Fall ersichtlich ist, unterscheiden sich die beiden Formate nur durch das MSB, welches beim Zweierkomplement dem Vorzeichenbit entspricht.

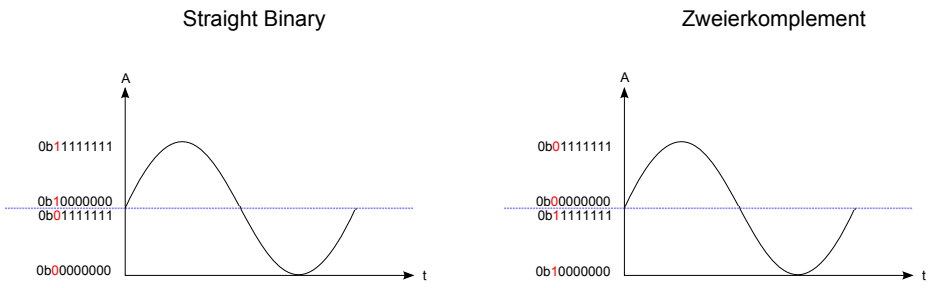


Abbildung 31: Straight Binary vs. Zweierkomplement

Es müssen also bloß die MSBs der beiden Kanäle umgedreht werden. Dies wird durch eine logische XOR-Verknüpfung mit der Bitmaske "0x80008000"erreicht.

```
//change data to 2's complement
r15 = 0x80008000;
r1 = r1 xor r15;
```

Im Anschluss müssen die beiden Eingangskanäle noch getrennt werden, also separat in die beiden höchstwertigen Bytes unterschiedlicher Systemregister geladen werden. Für den zweiten Kanal wird das Datenwort zunächst in das Systemregister *R2* kopiert und durch eine logische Verschiebung um 16 Stellen an die höchste Stelle geschoben. Der erste Kanal geht dadurch verloren, die hinteren Stellen werden mit Nullen aufgefüllt.

```
r2 = lshift r2 by 16;  //delete first channel
```

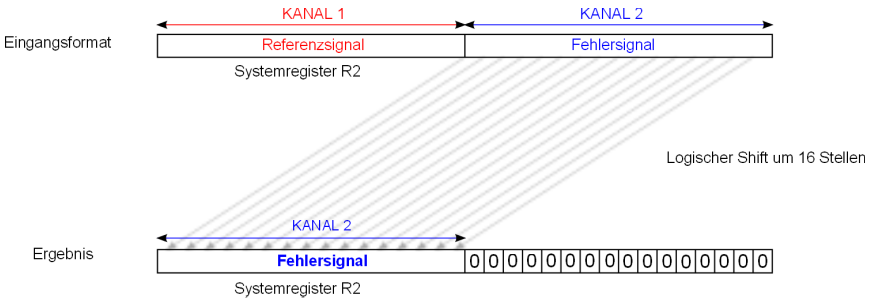


Abbildung 32: Extraktion des Fehlersignals

Für den ersten Kanal müssen die 16 hinteren Stellen, die dem zweiten Kanal entsprechen, gelöscht werden. Dies geschieht mit der logischen UND-Verknüpfung mit dem Bitmaske "0xFFFF0000".

```
r8 = 0xFFFF0000;
r1 = r1 AND r8;  // Extraction
```

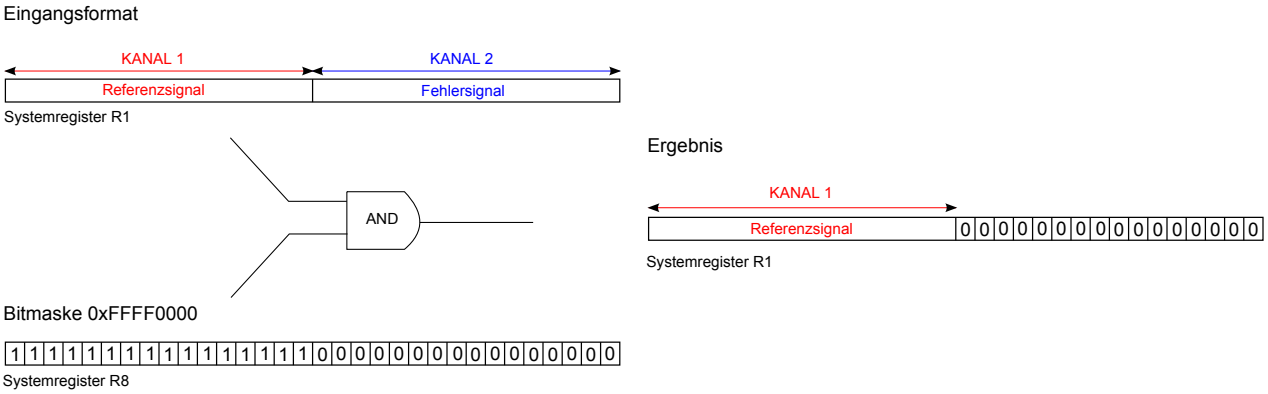


Abbildung 33: Extraktion des Referenzsignals

Im Anschluss werden die beiden Datenwörter von Festkomma in das 32 Bit Gleitkomma-Format umgewandelt.

```
r15 = -31; // Convert to floating point format
f0 = float r2 by r15;
f1 = float r1 by r15;
```

Leider musste festgestellt werden, dass der ADC nicht gleichspannungsfrei wandelt. Kanal eins hat einen positiven, Kanal zwei einen negativen Offset. Aus diesem Grund wird der gemessene Wert zu den Datenwörter addiert bzw. abgezogen. Die Werte wurden ermittelt, indem ein hochfrequenter Sinus an die Eingänge des ADCs angelegt und die Wandlungsergebnisse intern

aufgepuffert wurden. Diese Daten konnten über ein Debug-Fenster visualisiert und der jeweilige Offset im Plot direkt abgelesen werden.

```
f2 = 0.001; //offset
f0 = f0 - f2; //apply offset

f2 = 0.002455; //offset
f1 = f1 + f2; //apply offset
```

Die Daten liegen nun fertig aufbereitet zur Weiterverarbeitung bereit. Das Referenzsignal wird in der Variable "x_in", das Fehlersignal in der Variable "error" abgespeichert.

```
dm(x_in) = f1;
dm(error) = f0;
```

6.3.5.3 Faltung mit adaptivem Filter

Das adaptive Filter wird als FIR-Filter implementiert, da dieses einfach realisierbar ist und die Stabilität jederzeit gewährleistet ist. Wie bereits in Abschnitt 2.3.2 erläutert, müssen im FXLMS-Algorithmus mehrere FIR - Filterberechnungen durchgeführt werden. Dies geschieht durch die Anwendung der diskreten Faltung entsprechend

$$y[n] = \underline{x}[n] \star \underline{w}[n] = \sum_{k=1}^{K} x[k]w[n-k] \tag{22}$$

In dieser Formel stellt $y[n]$ den erzeugten Anti-Schall, $x[n]$ die Eingangsdaten und $w[k]$ die jeweiligen Filterkoeffizienten des adaptiven Filters dar. Zuerst müssen K Eingangssamples aufgepuffert werden. Dazu wird bei jedem Aufruf der Serviceroutine das gerade erhaltene Eingangssample im Ringpuffer 0 gespeichert.

```
//Save actual Sample in ringbuffer 0
f0 = dm(x_in);
dm(i0,m0) = f0;
```

Ist der Puffer voll, wird der älteste Eintrag vom neuen Sample überschrieben. Die Filterlänge definiert die Anzahl an Samples K, welche gespeichert werden können. Sie kann am Anfang der Datei über die Konstante *BLKSZ* eingestellt werden.

```
//Save actual Sample in ringbuffer 0
#define BLKSZ  256
```

Der Ringpuffer ist so organisiert, dass er von oben nach unten gefüllt und ausgelesen wird. Bei jedem neuen Speichereintrag springt der Pointer automatisch um M0=1, also auf die nächste Stelle. Dies entspricht jener Stelle mit dem ältesten Datenwort, welches daraufhin gleichzeitig mit einem Filterkoeffizienten ausgelesen wird. Das Auslesen der beiden Operanten bewirkt wieder eine Inkrementierung der Pointer um M0, also um eine Stelle.

```
//get oldest sample and last coef out of ringbuffer 3 & 10
f0 = dm(i0,m0), f4 = pm(i8,m8);
```

Durch die Super Harvard Architektur können jetzt mehrere Befehle gleichzeitig ausgeführt werden: die Multiplikation von Sample und Koeffizient und das Auslesen der beiden nächsten Operanten.

```
//multiplication sample * coef
//get newer sample and next coef
f8 = f0*f4, f0 = dm(i0,m0), f4 = pm(i8, m8);
```

Die gleiche Prozedur muss für alle Samples und Koeffizienten, also insgesamt K-mal abgearbeitet werden. Daher bietet sich die Verwendung einer Schleife an, in welcher jedes neue Multiplikationsergebnis mit dem vorherigen aufsummiert wird.

```
//Convolution loop
LCNTR=(BLKSZ2-2), DO convsec UNTIL LCE;
////////////////////////////////////////
//Debug code to check loop
//r14 = r14 +1;
//comp(f8,f15);
//if ne call _nop;
////////////////////////////////////////
convsec: f12=f0*f4, f8=f8+f12,f0=dm(i0,m0),f4=pm(i8,m8); //use f4 for dag2 and f0 for dag1!!

//missing calculation
f12=f0*f4, f8=f8+f12;
f8 = f8+f12;
//change signal polarity
//f6=-1.0;
//f8=f8*f6;
```

Aus programmiertechnischen Gründen werden die erste und die letzte Berechnung außerhalb der Schleife durchgeführt: Bei der ersten ist noch kein altes Multiplikationsergebnis vorhanden, welches mit dem Neuen aufsummiert werden muss. Die letzte Berechnung wurde aus der Schleife entfernt, da es dabei zu einer unerwünschten Inkrementierung der Pointer in den Ringpuffern kommen würde. Aus diesem Grund wird die Berechnung der letzten Puffereinträge ausserhalb der Schleife ohne Pointermanipulation durchgeführt. Das fertige Faltungsergebnis entspricht dem Antischall und wird in die globale Variable "_y" abgespeichert.

```
dm(_y) = f8;
```

Die Pointer stehen nach den Berechnungen bereits auf den richtigen Speicherbereichen für den nächsten Durchgang, eine Manipulation durch die "index"-Variablen ist im Prinzip überflüssig. Sie dienen nur der Sicherheit, da es bei einer eventuellen Erweiterung des Programms mit C/C++ Code zu unerwünschten Veränderungen der Indexregister kommen könnte.

Es ist zu beachten, dass während der Faltungsberechnung die ältesten Eingangssamples zuerst ausgelesen werden, diese mit Fortlauf der Schleifendurchgänge an Aktualität gewinnen und erst bei der letzten Berechnung jenes Datenwort genutzt wird, welches beim Start der Routine eingelesen wurde. Dies bedeutet, dass die Filterkoeffizienten im Puffer nach der Adaption dementsprechend "verkehrt" stehen müssen, also der erste Koeffizient an der letzten Pufferstelle stehen muss.

6.3.5.4 Faltung mit geschätzter Sekundärstrecke

Die Faltung mit der geschätzten Sekundärstrecke entspricht seitens der Programmierung genau jener mit dem adaptiven Filter, außer dass das Ergebnis der Faltung nicht in eine Variable, sondern wieder in einen Puffer geschrieben wird. Dieser wird im Anschluss für das Koeffizientenupdate gebraucht. Außerdem wird eine andere Filterlänge verwendet. Für die Abschätzung der Sekundärstrecke sind 100 Filterstellen ausreichend und dementsprechend wurde die Filterlänge über die *BLKSZ2* Konstante eingestellt. Aufgrund der Speicherorganisation der Ringpuffer ist darauf zu achten, dass die Koeffizienten der geschätzten Sekundärstrecke in umgekehrter Reihenfolge, also beginnend mit dem letzten Koeffizient, initialisiert werden.

6.3.5.5 Update der adaptiven Filterkoeffizienten

Die Änderung der Koeffizienten des adativen Filters entsprechend des LMS-Algorithmus erfolgt gemäß

$$\underline{w}[n+1] = \underline{w}[n] + \mu \cdot e[n] \cdot \underline{\hat{x}}[n] \tag{23}$$

Bei $\underline{\hat{x}}[n]$ handelt es sich um den Puffer der mit der Sekundärstrecke gefilterten Eingangsdaten, $\underline{w}[n]$ stellt den Puffer der Filterkoeffizienten des adaptiven Filters dar. Die Schrittweite μ und der Fehler *e[n]* sind in dieser Formel Skalarwerte, müssen also nur einmal für die Berechnung aller Filterkoeffizienten miteinander multipliziert werden. Dazu werden diese zunächst in die entsprechenden Systemregister *F2* und *F5* geladen. Während der Multiplikation kann bereits das erste Daten/Koeffizienten-Paar aus dem Speicher geladen werden. Im Anschluss daran ergibt sich ein Rechenaufwand von einer Multiplikation und einer Addition pro Filterkoeffizient. Die Berechnungen sind also sehr ähnlich umzusetzen wie die Faltungen.

Es folgt die Berechnung des ersten Subtrahenden in Register *F1*.

```
f2 = STPSZ;
f5 = dm(error);

f8 = f2*f5, f0 = dm(i5,m0), f4 = pm(i8,m8);

//f8=m*e ; f0=x'

//f1 = m*e*x = f8*f0,  //f1!=m*e*x' ; f4=w

f1 = f8*f0;
```

Ähnlich wie bei der Faltung können auch alle weiteren Berechnungen bequem in einer Scheife abgearbeitet werden. Der entscheidende Unterschied zur Faltung aus programmiertechnischer Sicht besteht darin, dass die Ergebnisse der einzelnen Schleifenergebnisse wieder an die selben Speicherstellen zurückgeschrieben werden müssen, aus denen sie geholt worden sind. Dies ist problematisch, da ein Speicherzugriff auf einen Koeffizienten bereits die Inkrementierung des entsprechenden Pointers zur Folge hat. Aus diesem Grund wird ein zweiter Ringpuffer über den gleichen Speicherbereich, also gleicher Länge und Basisadresse, benutzt: Über den Pointer *I8* werden die Koeffizienten ausgelesen, über *I11* wieder beschrieben. Die einzelnen Speicherzugriffe stehen sich damit nicht im Weg.

```
//Update loop
LCNTR=(BLKSZ-1), DO update UNTIL LCE;
f3=f4+f1, f0=dm(i5,m0), f4 = pm(i8,m8);
update: f1 = f8*f0, pm(i11,m8)=f3;
```

Bei jedem Schleifendurchlauf wird ein Koeffizient aktualisiert, die Anzahl an Berechungen entspricht also jener der Filterlänge bzw. der Anzahl an Filterkoeffizienten *BLKSZ*. Gleich wie bei der Faltung wird die letzte Berechnung außerhalb der Schleife durchgeführt, damit es hier zu keiner unerwünschten Pointer-Inkrementierung kommt.

```
f3 = f4+f1;
pm(i11,m8)=f3;
```

6.3.5.6 Aufbereitung des Ausgangssignals

Bevor die berechneten Ausgangsdaten an den DAC geschickt werden können, müssen sie wieder in das Festkomma-Format umgewandelt werden. Dies geschieht mit dem "trunc" Befehl.

```
f8 = dm(_y); //f8 = dm(error);
r15 = 31;
r1 = trunc f8 by r15;   // Convert to fix point format
```

Eine Umwandlung in das Straight Binary-Format ist nicht nötig, da der DAC für Wandlungen von Daten im Zweierkomplement ausgelegt ist. Dafür ist es notwendig, dass die Verbindung *LK9* auf dem DAC Evaluierungsboard auf Position B eingestellt worden ist.

Das Ergebnis der "trunc" Operation liefert ein Ergebnis mit einer höheren Auflösung als 16 Bit, welches vom DAC nicht gewandelt werden kann. Aus diesem Grund werden die untersten Stellen abgeschnitten. Dies geschieht durch eine Verschiebeoperation: mittels logischen Shifts um 16 Stellen nach rechts werden die untersten Bits verworfen. Im Anschluss müssen die restlichen Daten wieder nach links in die MSBs geschoben werden. Die Stellen der verworfenen Bits werden mit Nullen aufgefüllt. Zusätzlich wird das Antischall-Signal noch mit Hilfe einer logischen ODER-Funktion in die untersten Stellen kopiert. Dadurch kann es parallel über zwei DAC Kanäle gewandelt werden (siehe Abschnitt 6.3.7) und steht doppelt zur Verfügung. Dies kann zu debug-Zwecken sehr hilfreich sein.

```
r1 = lshift r1 by -16;
r2 = r1;
r1 = lshift r1 by 16;
r1 = r1 OR r2;
```

Das zur Wandlung bereite Datenwort wird wieder in die globale Variable "_y" geschrieben. Im Anschluss wird die Funktion "process_AD7655_samples" in der "process_audio.c" Datei aufgerufen, welche für die Datenübertragung zuständig ist.

```
dm(_y) = r1;

jump _process_AD7655_samples;
```

6.3.6 Implementierung des Feedback-Algorithmus

Wird die Konstante "USE_FF_ALGORITHM" in den Datein "main.asm" und "21489_IVT.asm" auskommentiert, ist der Feedback-Algorithmus ausgewählt. Bei Auftreten des SPIHI-Interrupts wird nun die Servicroutine "_fb_algorithm" in der "FBalgorithm.asm" Datei ausgeführt. Aus programmiertechnischer Sicht ist diese Routine in den wesentlichsten Punkten ident mit jener für die Berechnung des Feedforward-Algorithmus. Daher wird in diesem Abschnitt nur auf die Unterschiede eingegangen.

6.3.6.1 Definition der benötigten Variablen und Puffer

Wie in Abschnitt 2.3.2 bereits erwähnt stellt der Feedback-Algorithmus aufgrund einer zusätzlichen Faltung einen höheren Berechnungsaufwand dar.

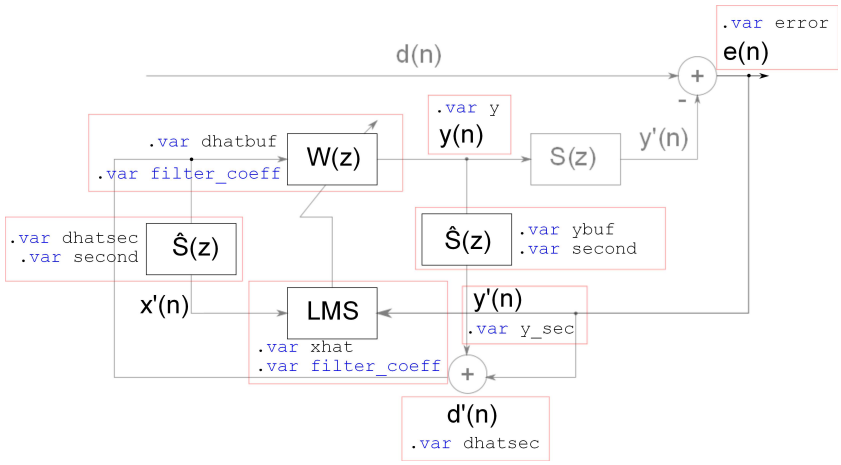


Abbildung 34: Variablen und Puffer für Feedback-Algorithmus

Daher werden nun insgesamt vier Puffer benötigt: "dhatbuf" für die Faltung mit dem adaptiven Filter, "dhatsecond" und "ybuf" für die Faltungen mit der geschätzten Sekundärstrecke und "xhat" für das Koeffizientenupdate. Sie werden im Segment "seg_dmda" angelegt und über den DM-Bus zugegriffen.

Zusätzlich benötigt der Feedback-Algorithmus drei lokale Variablen: "error" für die Zwischenspeicherung des Fehlersignals, "y_sec" um das Filterergebnis des Antischall-Signals mit der geschätzten Sekundärstrecke zu halten und "dhat" für die Summation von "error" und "y_sec".

```
.section/dm seg_dmda;

.var error;
.var dhatbuf[BLKSZ];
.var dhatsecond[BLKSZ2];
.var ybuf[BLKSZ2];
.var y_sec = 0;
.var dhat;
.var _xsec;
.var index;
.var index2;
.var index3;
.var index4;
.var xhat[BLKSZ];
```

Im Folgenden werden die einzelnen Schritte des Programmcodes ausführlich beschrieben.

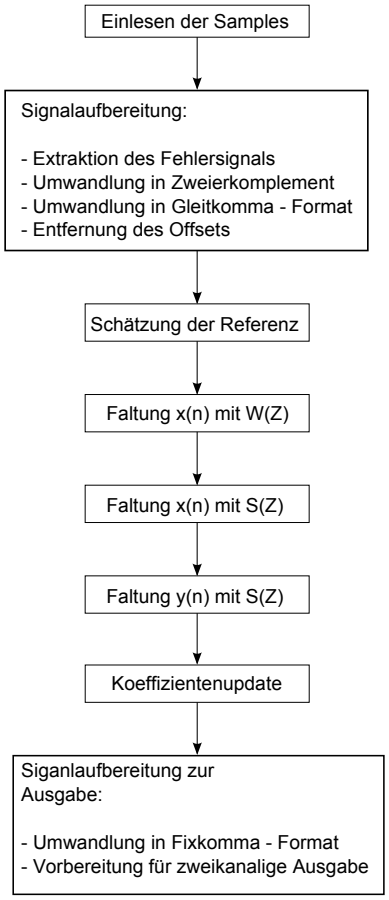


Abbildung 35: Programmschritte des Feedback-Algorithmus

6.3.6.2 Einlesen und Aufbereitung der Eingangsdaten

Da hier nur das Signals des Fehlermikrofons benötigt wird, muss auch nur dieses aus dem eingelesenen Datenwort extrahiert werden. Alle anderen Überlegungen entsprechen denen des Feedforward-Algorithmus.

6.3.6.3 Schätzung der Referenz

Der mit der geschätzten Sekundärstrecke gefilterte Ausgang des adaptiven Filters wird nun mit dem Fehlersignal aufsummiert und in der Variable "dhat" abgelegt. Das Ergebnis stellt die Schätzung der Referenz dar.

```
.section/dm seg_dmda;

f1 = dm(y_sec);
//f1 = 0;
```

```
f0 = f0 + f1;
dm(dhat) = f0;
```

6.3.6.4 Faltung mit adaptivem Filter

Die geschätzten Referenzsignale werden nun, analog zu den Eingangssignalen des Referenzmikrofons beim Feedforward-Algorithmus, aufgepuffert und mit dem adaptiven Filter gefaltet. Programmtechnisch ist die Faltung ident mit jener in Abschnitt 6.3.5.3.

6.3.6.5 Faltung des Referenzsignals mit geschätzter Sekundärstrecke

Auch diese Faltungsoperation entspricht genau jener beim Feedforward-Algorithmus in Abschnitt 6.3.5.3.

6.3.6.6 Faltung des Anti-Schalls mit geschätzter Sekundärstrecke

Diese Berechnung stellt eine Erweiterung gegenüber dem Feedforward-Algorithmus dar. Ebenso wie das geschätzte Referenzsignal muss auch das Antischall-Signal mit der geschätzten Sekundärstrecke gefaltet werden. Das Ergebnis wird in der Variable "y_sec" abgespeichert und zur Schätzung der Referenz mit dem Fehler addiert.

Programmtechnisch ist sie ident mit den vorherigen Faltungen und wird daher an dieser Stelle nicht ausführlich besprochen.

6.3.6.7 Update der adaptiven Filterkoeffizienten

Das Update der Filterkoeffizienten geschieht in gleicher Weise wie beim Feedforward-Algorithmus, zur Beschreibung wird daher auf den Abschnitt 6.3.5.5 verwiesen.

6.3.6.8 Aufbereitung des Ausgangssignals

Die Aufbereitung des Ausgangssignals geschieht in gleicher Weise wie beim Feedforward-Algorithmus, zur Beschreibung wird daher auf den Abschnitt 6.3.5.6 verwiesen.

6.3.7 Übertragung der Ausgangsdaten

Nachdem das Antischall-Signal entsprechend dem gewählten Algorithmus berechnet wurde, muss das fertige Datenwort an den DAC geschickt werden. Dafür ist die Funktion "process_AD7655_samples" in der Datei "process_audio.c" verantwortlich.

Wie bereits in Abschnitt 6.3.1 beschrieben, muss das 16 Bit Ausgangswort auf zwei 8 Bit Wörter aufgespalten und mit einem Adressbyte versehen werden. Das passiert für zwei Kanäle im "sbuffer" Array. Es handelt sich dabei um ein sechsstelliges Array des Datentyps Char, dessen einzelne Speicherbereiche aus jeweils 8 Bits bestehen. Für den zweikanaligen Fall müssen die sechs Speicherzellen entsprechend Abbildung 36 organisiert sein und sequenziell übertragen werden.

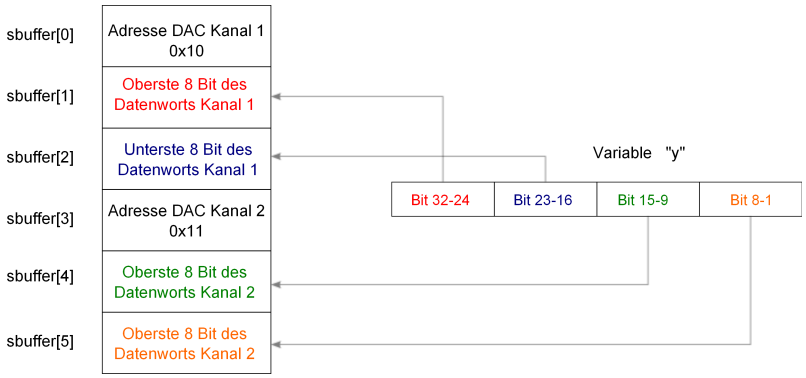


Abbildung 36: Speicherbelegung "sbuffer"

Die DAC-Adressen werden bereits bei der Initialisierung des Puffers in die erste bzw. vierte Speicherzelle geschrieben, alle anderen Felder mit Nullen initialisiert.

```
#define addrL  0x10  // 0b00010000      -> write L-sample to first DAC in data register
#define addrR  0x11  // ob00010001      -> write R-sample to second DAC in data register
.
.
.
char sbuffer[6] = {addrL,0,0,addrR,0,0};
```

Innerhalb der Funktion werden die Ausgangsdaten aus der globalen Variable "y" (ohne Unterstrich, da es sich um C-Code handelt) ausgelesen. Es ist zu beachten, dass es sich bei "y" um einen Integer handelt. Versucht man eine Variable dieses Datentyps in einen Char-Array zu kopieren, werden alle Bits über dem niedrigstwertigen Byte verworfen. Dies wird für das Aufspalten der Ausgangsdaten ausgenutzt: mittels Rechtsshift werden die benötigten Bits an die untersten 8 Stellen geschoben und in die jeweilige Pufferstelle geschrieben.

```
sbuffer[1] = (y) >> 24;
sbuffer[2] = (y) >> 16;
sbuffer[4] = (y) >> 8;
sbuffer[5] = (y);
```

Nun ist es möglich, die einzelnen Speicherzellen mittels übergangslosen Transfers zu übertragen. Das bedeutet, dass nach der Übertragung eines 8 Bit Wortes nahtlos die Übertragung des nächsten beginnt. Das Taktsignal reißt währenddessen nicht ab. Dies passiert für jeweils einen Kanal. Wie in Abbildung 27 zu sehen, muss aufgrund der Spezifikationen des DAC nach der Übertragung der 24 Kanalbits eine steigende Flanke des SYNC-Signals erfolgen. Zum Start der Übertragung des zweiten Kanals ist zeitgleich mit dem ersten Taktsignal eine fallende Flanke erforderlich.

Programmtechnisch wird dies mit zwei verschachtelten FOR-Schleifen gelöst. In der ersten wird zunächst das entsprechende Flag im SPIFLGB Register null gesetzt. Wie in Abschnitt 6.3.1 bereits erklärt wird über dieses Flag die CS-Leitung des SPI-Ports konfiguriert. Durch das Nullsetzen wird die Leitung low, der angeschlossene Slave also ausgewählt. In der anschließenden Schleife werden die drei obersten Einträge des "sbuffer" in das TXSPI-Register geladen. Analog zum Einlesen der Daten über RXSPI handelt es sich beim TXSPI um ein beschreibbares

Ausgangsregister. Die Daten werden im Anschluss an den Kopiervorgang in ein Ausgabeschieberegister kopiert und über SPI automatisch gesendet. Da ein Schleifendurchgang schneller erfolgt als die Übertragung eines Puffereintrags, muss gewartet werden, bis das jeweilige Wort am Empfänger angekommen ist, da es sonst vom nächsten Kopiervorgang überschrieben werden würde. Dadurch würde nur das letzte Datenwort tatsächlich übertragen werden. Aus diesem Grund wurde eine Wartefunktion mittels While-Schleife implementiert. In dieser wird das *TXS* Statusflag überprüft und damit festgestellt, ob sich noch Daten im Ausgangsschieberegister befinden. Ist es gesetzt, ist dies der Fall. Wurde das Datenwort vollständig übertragen, wird das Flag gelöscht und die Schleife beginnt von vorne mit der Übertragung des nächsten Puffereintrags.

Nach drei Durchgängen ist der erste Kanal vollständig übertragen worden. Das Setzen des entsprechenden Flags im SPIFLGB Register bewirkt eine steigende Flanke an der CS-Leitung bzw. am SYNC-Eingang des DAC. Bis zu diesem Zeitpunkt wurde der erste Kanal inklusive DAC-Adressbyte in 24 Taktzyklen transferiert. Der CS-Ausgang war genau während dieser 24 Taktzyklen low. Im Anschluss startet die äußere Schleife von neuem und der zweite Kanal wird nach der gleichen Vorgangsweise übertragen.

Die *"rti;"* Instruktion beendet die Interruptserviceroutine und der Prozessor springt in den *Idle-Mode*, bis die entsprechende Serviceroutine durch ein neues Datenwort am SPI-MOSI-Eingang erneut aufgerufen wird.

```
i = 0;

//k...run 1st loop 2x for first/second Channel
for(k=0;k<2;k++)
{

// Enable Slave / SPI Flag0 Output

                *pSPIFLGB &= (0xEff);

                //2nd loop to transmit first channel
                        //i...run loop 3 times for address byte and 2x8 bit data
                    for(i=0;i<3;i++)
                        {         j=0;
                                        *pTXSPIB = sbuffer[i+(k*3)];

                //Wait for the SPI to indicate that it has finished.
                        while ((*pSPISTATB & TXS))
                                {j++;}
                        }

                 j=0;
                        //Wait for the SPI to indicate that it has finished.
                        while (!(*pSPISTATB & SPIFE))
                        {j++;}

      //  Delay(10);

//Disable Slave / SPI Flag0 Output

        *pSPIFLGB |= (0xF00);
// wait N clock cycles between L and R channel (DS going high!)
        //      Delay(0);
        asm("nop;");
i = 0;
}
k=0;

asm("rti;");
```

7 Test des ANC-Systems

7.1 Einführung

Es sollen nun die für beide Algorithmen ermittelten Messergebnisse präsentiert werden. Dabei werden sie auch jenen Messergebnissen gegenübergestellt, die mit kommerziell erhältlichen ANC-Kopfhöreren zu einem früheren Zeitpunkt von AKG auf ähnliche Weise mit einem typgleichen Kunstkopf ermittelt wurden.

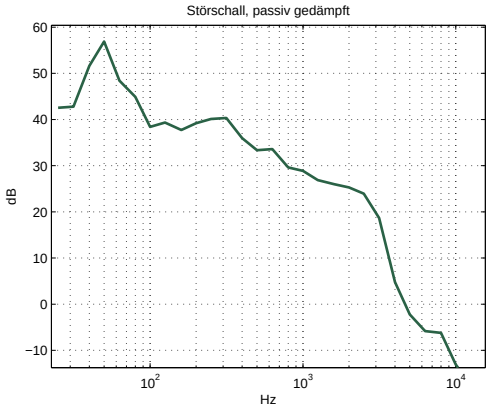


Abbildung 37: Passiv gedämpftes Anregungssignal

7.2 Messung digitales Feedforward ANC

7.2.1 Feedforward ANC bei einer Schalleinfallsrichtung

Zur Messung der Leistungsfähigkeit der Feedforward-Methode wurde ein in PD generiertes Rosa-Rauschen in einer reflexionsarmen Versuchskabine über einen Lautsprecher abgespielt. In kurzer Entfernung davon befand sich ein Kunstkopf mit angebrachten ANC-Kopfhörer.

Abbildung 38: Versuchaufbau für eine Einfallsrichtung

Nach der Überlagerung von eingedrungenem Schall und Antischall wurde über das im Kunstkopf angebrachte Mikrofon das Fehlersignal gemessen.

Abbildung 39 stellt die Messergebnisse des digitalen Feedforward-Systems jenen der analogen ANC-Kopfhörer gegenüber. Dabei wurde nur die Leistung des aktiven Teils der Geräuschunterdrückung untersucht, indem das passiv gedämpfte Rauschsignal im Logarithmischen vom Restfehler abgezogen wurde.

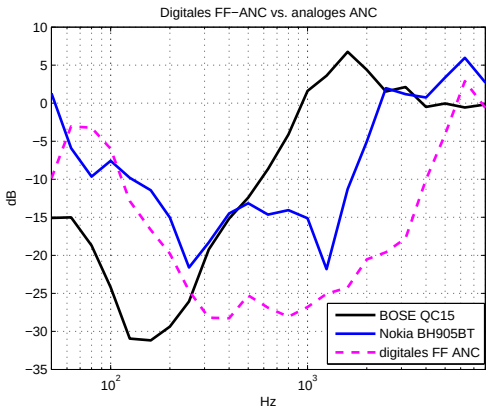


Abbildung 39: Leistungsfähigkeit digitales Feedforward- vs Analoges ANC

Man kann erkennen, dass sich ab ca 200 Hz beim digitalen System eine Verbesserung gegenüber den analogen Kopfhörern um ca. 10 dB ergibt. Der Einfluss der analogen Filter wird hier

deutlich. Es muss bei großer Auslöschung (entspricht hohes k) in den höheren Frequenzbereichen stärker gefiltert werden, um Stabilität gewährleisten zu können. Im digitalen System sind Auslöschungen bis zu ca. 5 kHz möglich.

Es sei jedoch erwähnt, dass analoge ANC-Kopfhörer in der Gesamtabstimmung mit der passiven Dämpfung entworfen werden. Die hier abgebildeten Ergebnisse geben nur die Leistungsfähigkeit des aktiven Teils analoger Systeme wieder. Wird der passive Teil optimal auf diesen abgestimmt, kann in Summe eine bessere Schallauslöschung speziell in höheren Frequenzbereichen erreicht werden.

Die geringere Leistungsfähigkeit im tieffrequenten Bereich des digitalen System findet ihre Begründung im Messverfahren. Aufgrund der Dimensionierungen des ANC-Kopfhörers und des Kunstkopfes konnte die Kopfhörermuschel nicht dicht am Kunstkopf angebracht werden. Dies resultiert in einer speziell im tiefen Frequenzbereich nicht optimalen Schallabstrahlung. Außerdem erhöht sich dadurch die Gruppenlaufzeit jener Frequenzen. Abbildung 40 zeigt die Abhängigkeit der Gruppenlaufzeit von der Dichtheit des Kopfhörers.

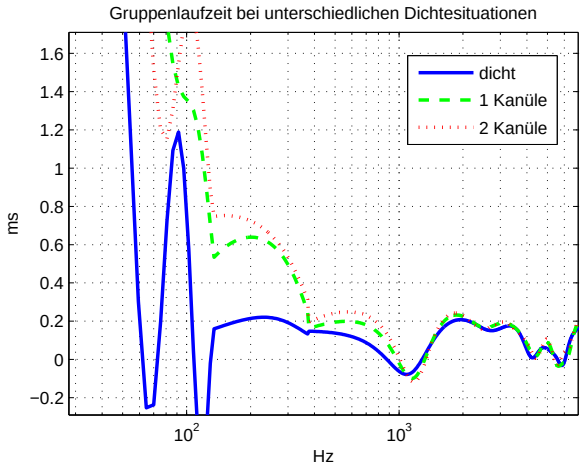


Abbildung 40: Abhängigkeit der Gruppenlaufzeiten von der Dichte des Kopfhörers

Die blaue Kurve entspricht dem Ergebnis bei jener Messung, bei welcher der Kopfhörer normal am Kunstkopf angebracht worden ist. Bei den übrigen Messergebnissen wurde die Dichte durch das Einfügen von Kanülen reduziert. Wie man gut erkennen kann, führt das zu einer weitern Signalverzögerung bei tiefen Frequenzen.

Aus diesem Grund wurde in einer weiteren Messung der Kopfhörer dichter am Kunstkopf angebracht. Wie man in Abbildung 41 sehen kann, ergibt sich eine Verbesserung der Auslöschung speziell bei tiefen Frequenzen.

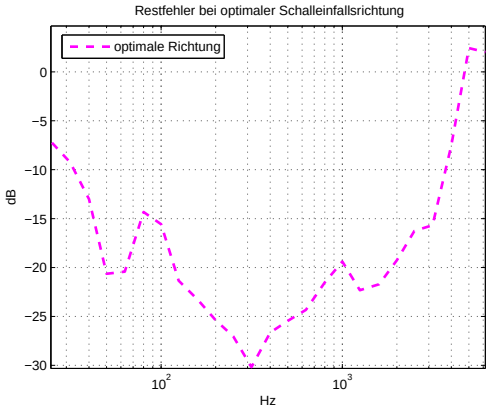


Abbildung 41: Leistungsfähigkeit digitales FF-ANC bei dichtem Kopfhörer

Die bisherigen Messergebnisse stellen das Ergebnis bei der optimalen Schalleinfallsrichtung dar. Diese Richtung ergibt sich durch die Verlängerung einer gedachte Linie zwischen Fehler- und Referenzmikrofon und stellt bei diesem Messaufbau die linke Seite direkt neben dem linken Kopfhörerkanal dar. Hierbei ist der zeitliche Vorteil am größten und eine Auslöschung des Störschalls ist bis zu höheren Frequenzbereichen möglich. In Abbildung 42 sind die Restfehler für verschiedene Einfallsrichtungen zu sehen. Durch den Verlust des zeitlichen Vorteils ergibt sich ein größerer Restfehler bei höheren Frequenzen, wodurch sich das Frequenzband, in welchem ANC funktioniert, verkleinert. Für die Messung wurde der Lautsprecher hinter dem Kunstkopf als auch rechts daneben angebracht.

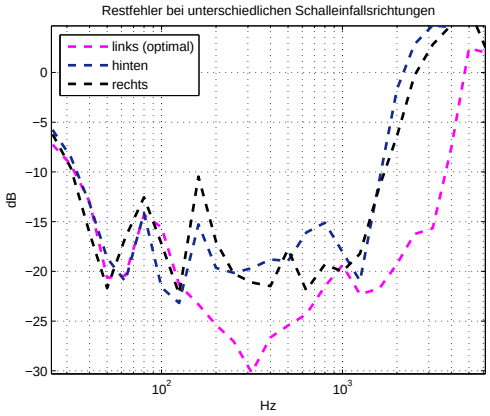


Abbildung 42: Leistungsfähigkeit digitales FF-ANC bei verschiedenen Schalleinfallsrichtungen

In Abbildung 43 ist rechts der zeitliche Verlauf des Restfehlers des digitales Systems für drei Schrittweiten dargestellt. Die verwendete Abtastrate beträgt 48 kHz. Bei einer Schrittweite von 0.008 ist nach einer Zeit von ca. 15 Sekunden der minimale Restfehler immer noch nicht erreicht worden. Daher ist diese Schrittweite in der Praxis völlig unbrauchbar. Die Lösung im

Anwendungsfall wird irgendwo zwischen den gezeigten Schrittweiten von 0.8 und 0.08 liegen.

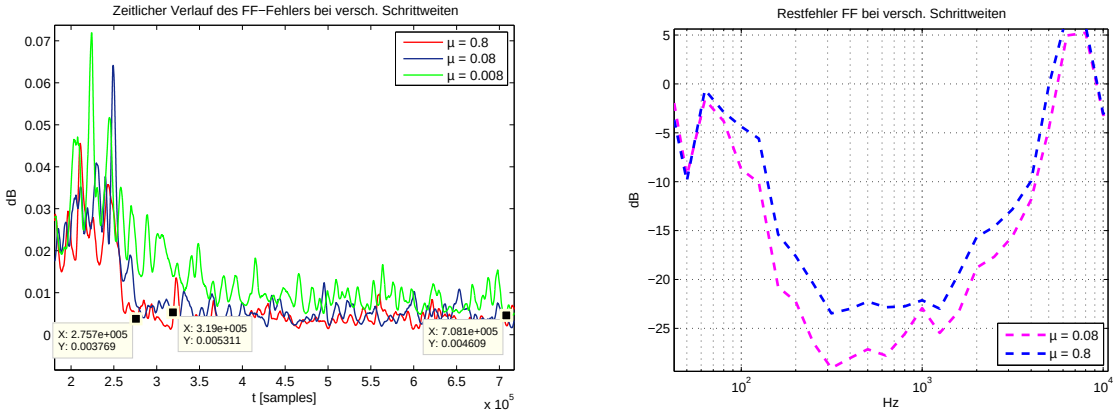


Abbildung 43: Links: Verlauf des Restfehlers für verschiedene Schrittweiten. Rechts: Fehler für verschiedene Schrittweiten im Frequenzbereich

Rechts wird verdeutlicht, wie sehr sich die kleinere Schrittweite auf den Grad der Auslöschung auswirkt. Man kann erkennen, dass doch noch einige Dezibel an Auslöschung durch das Senken der Schrittweite erreichbar ist. Dies ist allerdings nur bei stationären Umgebungen ratsam, da die Schallauslöschung bei sich schnell ändernden Signalen schlechter funktionieren wird. Das Messergebnis stimmt somit mit dem Simulationsergebnis überein.
Ebenso wurde der Einfluss der Filterlänge überprüft. Abbildung 44 zeigt das gemessene Ergebnis für drei verschiedene Filterlängen.

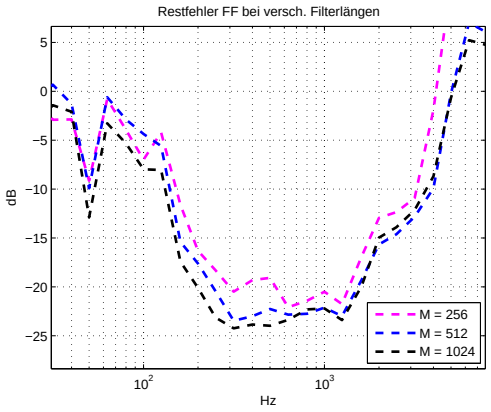


Abbildung 44: Fehler bei verschiedenen Schrittweiten im Frequenzbereich

Auch hier stimmt die Messung mit dem Simulationsergebnis überein. Offensichtlich wird durch die bessere Frequenzauflösung die Übertragungsfunktion des Kopfhörers und der Sekundärstrecke besser identifiziert und dementsprechend das Ausgangssignal genauer nachgebildet.

Daher ist die Verwendung eines längeren Filters vorteilhaft. Voraussetzung dafür ist ein Prozessor, der die aufwendigeren Rechenoperationen zeitgerecht durchführen kann.

7.2.2 Feedforward ANC im Diffusfeld

Um eine diffusere Anregung zu erhalten, wurde der Kunstkopf aus der Versuchskabine entfernt und zwischen reflektierenden Wänden positioniert. Das Rosa-Rauschen wurde in dieser Anordnung über zwei gegen die Wände gerichtete Lautsprecher wiedergegeben.

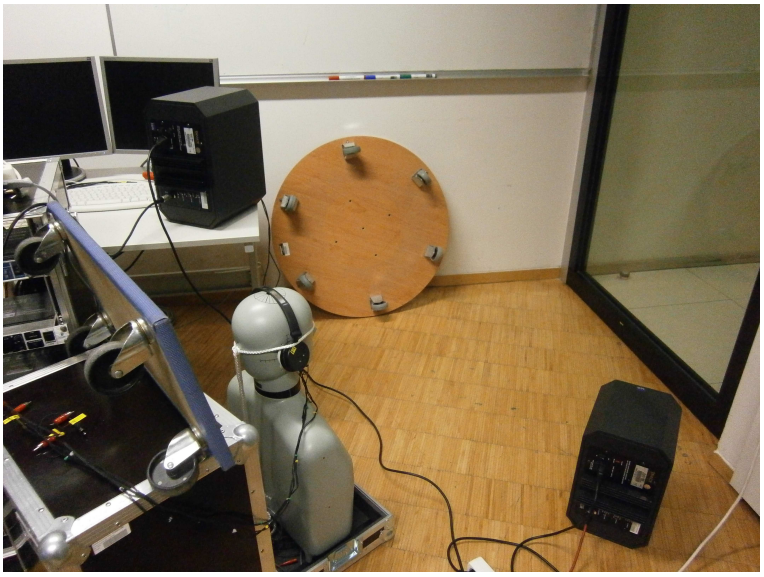


Abbildung 45: Versuchaufbau Diffusfeld

Es ist anzumerken, dass ein optimales Diffusfeld nicht erzeugt werden konnte, da dazu ein Hallraum erforderlich wäre. Ein solcher stand für die Messungen nicht zur Verfügung. Die Ergebnisse in Abbildung 46 entsprechen daher dem durch den in Abbildung 45 gezeigten Messaufbau erreichten Schallfeld.

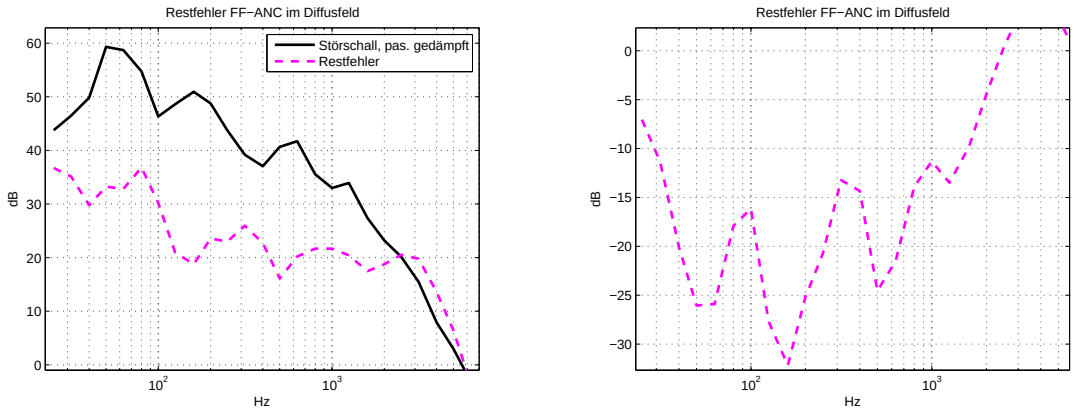


Abbildung 46: Leistungsfähigkeit digitales FF-ANC im Diffusfeld

Man sieht, dass, ähnlich wie bei den unterschiedlichen Einfallsrichtungen, Auslöschung von ca. 25 dB bis ca. 3 kHz möglich ist. Der Verlauf der Fehlerkurve ist dem kammfilterartigen Spektrum des diffusen Anregungssignals geschuldet.

7.3 Feedback ANC

Abbildung 47 stellt die Messergebnisse des digitalen Feedforward-Systems dar. Der Messaufbau entspricht genau jedem der Feedforward-Messung für eine Einfallsrichtung.

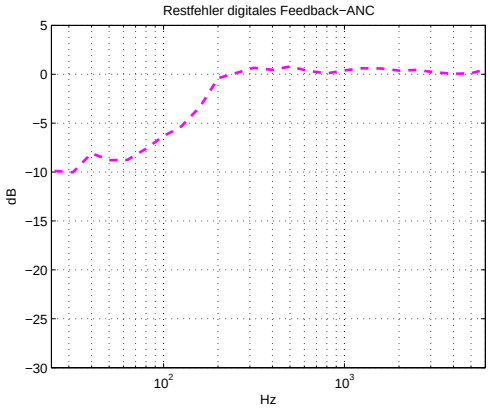


Abbildung 47: Leistungsfähigkeit digitales Feedback ANC

Wie hier gezeigt werden kann, ist die Verwendung eines digitalen Feedback ANC-Systems nicht sinnvoll. Der fehlende zeitliche Vorteil gegenüber dem Feedforward-Algorithmus wirkt sich offenbar in sehr starkem Ausmaß aus. Da keine besseren Ergebnisse für unterschiedliche Einfallsrichtungen bzw. im Diffusfeld zu erwarten sind, wurde auf weitere Untersuchungen verzichtet.

8 Fazit & Ausblick

Im Zuge des praktischen Teils dieser Diplomarbeit wurde ein digitales ANC-System simuliert und ein funktionierender Prototyp für einen Kanal realisiert. Dabei wurden Umsetzer hoher Schnelligkeit mit einem leistungsfähigen DSP-System über SPI-Schnittstelle miteinander verbunden. Sie sorgen, zusammen mit Filtern geringer Gruppenlaufzeit, für eine niedrige Latenz der Sekundärstrecke.

Diese Sekundärstrecke wurde für die Verwendung im Filtered-X-LMS-Algorithmus mittels Sweepmessung geschätzt. Es wurden zwei Varianten des Algorithmus, der Feedforward- und der Feedback-Algroithmus, implementiert und getestet. Die Messergebnisse wurde auch jenen von kommerziell erhältlichen analogen Antischall-Kopfhörern gegenübergestellt.
In der Diplomarbeit wurden die nötigen theoretischen Grundlagen zum Verständnis aktiver Geräuschunterdrückung bereitgestellt. Ebenso wurde auf die Nachteile derartiger analoger Systeme eingegangen und daraus die Motivation für ein digitales System abgeleitet. Des Weiteren wurden die dafür verwendeten Komponenten ausführlich beschrieben.

Wie gezeigt werden konnte, ist durch digitales Feedforward ANC eine sehr starke Auslöschung des einfallenden Störschalls möglich. Für die optimale Einfallsrichtung, also jene Richtung, die auf einer gedachten Linie mit dem Innen- und Außenmikrofon liegt, kann dies auch für sehr hohe Frequenzbereiche bis ca. 5 kHz erreicht werden, aber auch bei unterschiedlichen Schalleinfallsrichtungen bzw. in diffusen Schallfeldern ist ein hoher Auslöschungsgrad in einem Frequenzbereich von ca. 3 kHz erreichbar. Bei analogen Systemen wird maximal eine Auslöschung bis ca. 2 kHz erreicht. Dadurch ergibt sich eine zusätzliche Auslöschung von ca. 10 dB zwischen 2 und 3 bzw. 5 kHz. Durch die Verwendung größerer Filterlängen kann die Leistungsfähigkeit zudem noch etwas gesteigert werden. Dies gilt auch für die Senkung der Schrittweite, was allerdings nur in stationären Umgebungen sinnvoll ist.

Für die Verwendung des Feedback-Algorithmus konnte kein Argument gefunden werden.

Das hier realisierte ANC-System ist trotz penibler Arbeitsweise nicht frei von Fehlern und in seinem Umfang ausbaufähig. Es soll in diesem Zusammenhang nur der Überprüfung eines Konzepts dienen. So ist es beispielsweise leicht zu verstehen, dass ein einkanaliges Kopfhörersystem in der Praxis völlig untauglich ist. Durch die hier bereitgestellte Anleitung und Beschreibung sollte aber eine Erweiterung auf einen zweiten Kanal, die Verwendung entsprechender Hardware vorausgesetzt, keine großen Schwierigkeiten darstellen. Die Rechenleistung des DSPs für zwei Kanäle ist auf jeden Fall gegeben, auch wenn möglicherweise weitere Prozessorkomponenten eingebunden werden müssen. Der Betrieb im SIMD-Modus würde sich dafür anbieten, da die Berechnungen durch Einbindung des zweiten Rechenelements parallelisiert werden können. Ebenso steht noch zusätzlich das FIR-Beschleunigungsmodul zur Verfügung.

Eine weitere potentielle Erweiterung wäre die Implementierung einer Messung der Sekundärstrecke. Dadurch könnte die Leistungsfähigkeit auf unterschiedliche Tragesituationen angepasst werden.

Außerdem ist leicht zu erkennen, dass das realisierte System eine weitere fundamentale Eigenschaft, nämlich jene der Portabilität, keinesfalls gewährleistet. Es wäre ratsam, die einzelnen Komponenten der Sekundärstrecke losgelöst von deren Evaluierungsboards auf einer Platine unterzubringen und miteinander zu verbinden. Dies würde auch die störanfälligen Kabelverbindungen obsolet machen, die das System mit gelegentlichen Störgeräuschen in Form von Knacksern oder Rauschen beeinträchtigen.

Literatur

[1] Oppenheim, A.V.; Schafer, R.W.; Buck J.R.: *Zeitdiskrete Signalverarbeitung.* Elektrotechnik : Signalverarbeitung. Pearson Studium, 2004.

[2] Grünigen, Daniel Ch. von: *"Digitale Signalverarbeitung"*, 2004. 3. Auflage.

[3] Haykin, Simon: *"Adaptive Filter Theory"*, 2002. 4. Auflage.

[4] Rupp, Markus: *"Vorlesung Vertiefung Signalverarbeitung Adaptive Filter"*. Skript, 2007. Institut für Nachrichtentechnik und Hochfrequenztechnik TU Wien.

[5] Keiler, Florian: *"Beiträge zur Audiocodierung mit kurzer Latenzzeit"*, 2006. 1. Auflage.

[6] Benesty, Jabob; Sondhi, M. M.; Huang Yiteng: *"Springer Handbook of Speech Processing"*, 2007. 1. Auflage.

[7] Kuo, Sen M.;Morgan, Dennis R.: *"Active Noise Control: A Tutorial Review"*. Proceedings of the IEEE, Vol. 87, No. 6, Juni 1999.

[8] Lueg, Paul: *"Process of silencing sound oscillations"*, 1936.

[9] Bartels, Volker: *"Headset with Active Noise-Reduction System for Mobile Applications"*. J. Audio Eng. Soc., Vol. 40, No.4, April 1992.

[10] Analog Devices: *"AD7655: Low Cost, 4-Channel, 1 MSPS 16-Bit PulSAR® A/D Converter"*, 2005. Datenblatt.

[11] Analog Devices: *"EVAL-AD76XXCB: Evaluation Board For AD761x,762x/AD763x/AD764x/AD765x/AD766x/AD767x/AD795x"*, 2009. Datenblatt.

[12] Analog Devices: *"ADSP-214xx SHARC Processor Hardware Reference, Rev. 1.0, February 2012"*, 2010. Bedienungsanleitung.

[13] Analog Devices: *"ADSP-21489 EZ-Board Manual"*, 2010. Bedienungsanleitung.

[14] Analog Devices: *"AD5764R: Complete Quad, 16-Bit, High Accuracy, Serial Input, Bipolar Voltage Output DAC"*, 2008. Datenblatt.

[15] Analog Devices: *"EVAL-AD5764R: Evaluation Board for Quad, 16-Bit, High Accuracy, Serial Input, Bipolar Voltage"*, 2009. Datenblatt.

[16] TRACOPOWER: *"TEN 3 Series"*, 2009. Datenblatt.

[17] Motorola, Inc.: *"SPI Block Guide V03.06"*, 2012. Revision 4.

[18] Müller, Swen; Massarani, Paulo: *Transfer-Function Measurement with Sweeps.* J. Audio Eng. Soc, 49(6):443–471, 2001.

[19] Weinzierl, Stefan; Giese, André; Lindau Alexander: *Generalized Multiple Sweep Measurement.* In: *Audio Engineering Society Convention 126*, 5 2009.

[20] Farina, Angelo: *Simultaneous Measurement of Impulse Response and Distortion with a Swept-Sine Technique*. In: *Audio Engineering Society Convention 108*, 2 2000.

[21] Guldenschuh, Markus; Sontacchi, Alois; Perkmann Michael; Opitz Martin: *"Assessment of Active Noise Cancelling Headphones"*. 2012.

[22] Schmaranz, Klaus: *"Softwareentwicklung in C"*. Springer Verlag, 2005.

[23] Analog Devices: *"VisualDSP++ 5.0 C/C++ Compiler Manual for SHARC Processors"*, Januar 2011. Revision 1.5. Bedienungsanleitung.

[24] Analog Devices: *"VisualDSP++ 5.0 Run-Time Library Manual for SHARC Processors"*, Januar 2011. Revision 1.5. Bedienungsanleitung.

[25] Analog Devices: *"VisualDSP++ 5.0 Assembler and Preprocessor Manual"*, Mai 2010. Revision 3.4. Bedienungsanleitung.

[26] Analog Devices: *"VisualDSP++ 5.0 Linker and Utilities Manual "*, Januar 2011. Revision 3.5. Bedienungsanleitung.

Abbildungsverzeichnis

Abkürzungsverzeichnis

LTI	Linear, zeitinvariant
IIR	Infinit Impulse Response
FIR	Finit Impulse Response
LMS	Least Mean Square
ANC	Active Noise Cancellation
FXLMS	Filtered-X Least Mean Square
dB	Dezibel
kHz	Kilohertz
Hz	Hertz
ADC	Analog-to-Digital-Converter
MSPS	Million Samples pro Sekunde
SAR	Sukzessive Approximation Register
MSB	Most Significant Bit
SPI	Serial Peripheral Interface
SMB	Sub-Miniature-B
SHARC	Super Harvard Architecture
DSP	Digital Signal Processor
DM	Data Memory
PM	Program Memory
DMA	Direct Memory Access
DAG	Datenadressierungsgenerator
SIMD	Single Instruction Multiple Data
SISD	Single Instruction Single Data
ALU	Arithmetisch-Logische Einheit
RAM	Random-Access-Memory
ROM	Read Only Memory
DPI	Digital Peripheral Interface
I/O	Input/Output
SRU	Signal Routing Unit
DAC	Digital-to-Analog-Converter
PCG	Precision Clock Generator
PCLK	Peripheral Clock
PLL	Phase -locked Loop, Phasenregelschleife
MHz	Megahertz
IC	Integrated Circuit
PC	Personal Computer
USB	Universal Serial Bus
DC-DC-Converter	Gleichspannungswandler
MOSI	Master Out Slave In
MISO	Master In Slave Out
CS	Chip Select
PD	Pure Data
IDE	Interaktive Entwicklungsumgebung
SDRAM	Synchronous Dynamic Random Access Memory
LDF	Linker Description File