Informatik Diplomarbeit

Verarbeitung digitaler Audiosignale mit Signalprozessoren und programmierbarer Logik

Ingmar Klippert

1 Vorwort

Das vorliegende Werk ist die Diplomarbeit von Ingmar Klippert.

Thema :

„Verarbeitung von Audiosignalen mit Signalprozessoren und programmierbarer Logik“.

betreuende Dozent :

Prof. Dr. rer. nat. Klaus Wüst des Fachbereiches Mathematik, Naturwissenschaften und Informatik im Fachgebiet Informatik.

Fach :

Signalverarbeitung

Ausführungszeitraum :

Sommersemester 96, Wintersemester 96/97

Ort:

Fachhochschule Gießen

Das Manuskript wurde auf einem IBM™ -kompatiblen PC mit dem Textverarbeitungsprogramm Microsoft Word™ erstellt. Die Grafiken wurden vorwiegend mit dem Programm CorelDraw™ und Adobe Photoshop™ angefertigt. Die in dieser Arbeit genannten Makennamen und Produktbezeichnungen unterliegen in der Regel dem patent- und warenrechtlichen Schutz, werden aber fortan nicht explizit mit ®, © oder ™ gekennzeichnet. Diese Diplomarbeit wurde im Rahmen einer Produktentwicklung der Firma Quasimidi angefertigt. Um den Marktvorsprung der Firma Quasimidi in Sachen digitaler Audioverarbeitung nicht zu schaden, ist die Ausgabe dieser Diplomarbeit an Personen, die nicht direkt mit dieser in Verbindung stehen bis zum 1.1.2000 untersagt.
Ich möchte es an dieser Stelle nicht unterlassen, all denen zu danken, die die Entstehung dieser Arbeit ermöglicht haben.

Kirchhain, Januar 1997 Ingmar Klippert

Adressen:

Ingmar Klippert
Kasseler Str. 22
34621 Frielendorf -Leimsfeld
Tel.: 06691-919309
Fax : 06691-919308
E-Mail: ingmar01(at)klipperts.de

Prof. Dr. rer. nat. Klaus Wüst
Fachhochschule Gießen-Friedberg
Wiesenstraße 14, 35390 Gießen
Tel.: 0641-309 527
Fax.: 0641-309 308

2 Inhalt

3 Einleitung

Computer werden heute nicht allein zum Rechnen, sondern für ganz unterschiedliche Aufgaben eingesetzt. Die Berührung zwischen Computer und Kunst gewinnt dabei immer mehr an Interesse: Computergrafik, Computeranimation und Computermusik sind zu alltäglichen Erscheinungen in unserer Medienwelt geworden. Unser Umgang mit technischen Hilfsmitteln wird mehr und mehr durch den Computereinsatz verändert und beeinflusst. Davon ist die Musik nicht verschont geblieben. Durch den raschen Fortschritt im Bereich Elektronik und Informationsverarbeitung werden fortlaufend Geräte mit neuen Bearbeitungsfunktionen und klanglichen Möglichkeiten entwickelt. In der Musikelektronik gibt es kaum einen Bereich, der nicht von der digitalen Technik betroffen ist und wo sich eine Ablösung der analogen Elektronik durch die digitale abzeichnet. Sei es Klangsynthese, Recording, Mixing, Filterung oder direktes Editieren der nun digital vorliegenden Daten, überall sind **d**igitale **S**ignal**p**rozessoren (fortan DSP) mit im Spiel welche manche Anwendungen erst ermöglicht haben. Elektronische Musikinstrumente, Mischpulte, Effektgeräte und digitale Tonaufzeichnungsgeräte verwenden DSPs und prägen heute schon den Alltag in professionellen Aufnahmestudios. Die voliegende Arbeit behandelt den Einsatz von DSPs und FPGAs (**F**ield-**P**rogrammable-**G**ate-**A**rray ; programmierbare Logiken) in digitalen Mischpulten an einem konkreten Beispiel.

4 Grundlagen

4.1 Studioalltag

Das Herz eines jeden Tonstudios ist das Mischpult. In ihm laufen alle Audiosignale zusammen, werden bearbeitet, gemischt, mit Effekten versehen und wieder ausgegeben. In typischen Tonstudios ist auch das Mischpult, welches am ehesten ins Auge fällt - 2 Meter breit und ca. 1000 Knöpfe darauf. Bei dieser Vielzahl an Bedienelementen ist es nicht verwunderlich, daß es eigene Studiengänge für diese Technik gibt (Toningenieur, Tonmeister). Zur Zeit findet man hauptsächlich Mischpulte mit vorwiegend analoger Technik in Studios . Dies hat mehrere Gründe:

- jeder Parameter des Mischpultes ist direkt erreichbar
- analoge Technik ist z.Zt. die kostengünstiere Lösung
- Zuverlässigkeit (kann nicht abstürzen)
- Toningeniuere sind diese Technik gewohnt

Jedoch hat diese Technik auch entscheidende Nachteile:

- Die Einstellung eines Pultes sind nicht abspeicherbar. Dies ist sehr hinderlich, wenn z.B. in einem Studio an mehreren Projekten gleichzeitig, aber mit nur einem Mischpult gearbeitet wird. Es müssen sämtlichen Einstellungen der Regler notiert werden, um den Zustand des Mischpultes zu einem späteren Zeitpunkt wieder herstellen zu können.
- Eine automatische Steuerung durch Computer ist nicht möglich. Wenn während eines Musikstückes ein Parameter geändert werden muß, z.B. die Hervorhebung eines Instrumentes beim Solo spielen, so wird dies z.Zt. per Hand gemacht und ist bei komplizierteren Parameteränderungen nicht mehr machbar.
- Verluste durch mehrere AD/DA Wandlungen. Die meisten Effektgeräte erzeugen ihre Effekte auf digitaler Ebene, d.h. das analoge Signal aus dem Mischpult muß erst digitalisiert werden, wird digital bearbeitet und wieder in analoge Signale umgewandelt um vom Mischpult weiterverarbeitet zu werden. Diese Wandlungen von analoge in digitale Ebene und zurück sind mit nicht unerheblichen Verlusten verbunden. Der Weg eines Signales kann durch mehrere Effektgeräte gehen, somit ist ein Klangverlust vorprogrammiert.

Diese Punkte lassen sich mit einem digitalen Mischpult lösen. Bei einem digitalen Mischpult werden die einkommenden Audiosignale sofort digitalisiert, sofern sie nicht schon digital vorliegen, und verlassen die digitale Ebene nur ganz am Ende der Verarbeitungskette, wenn die Musik hörbar gemacht werden muß. Im Zuge der immer weiter fortschreitenden Entwicklung von Prozessoren, ist es nur eine Frage der Zeit bis digitale Mischpulte die Analogen vom Markt verdrängen.

4.2 Digital Audio

4.2.1 AD/DA Wandler

Ein Schallereignis kann mit Hilfe eines Mikrophons in eine elektrische Spannung umgewandelt werden. Diese Spannung, welche den kontinuierlichen Verlauf der Schwingungen (Luftschwingungen) repräsentiert kann nicht direkt vom Computer verarbeitet werden, da er nur die diskreten Werte 0 und 1 akzeptiert. Um die Schwingungen für den Computer verständlich zu machen wird die Schwingung in kleine Zeitfenster aufgeteilt. Der Anfangszustand der Schwingung (Spannungswert) in diesem Zeitfenster wird erfaßt (Pulsamplitudenmodulation) und in die digitale Binärform gebracht(Amplitudenquantisierung), die nun der Computer od. der Signalprozessor einlesen kann. Man erhält aus einem kontinuierlichen Spannungsverlauf eine Folge von diskreten Werten, die jedoch für den Computer ausreichen, den Spannungsverlauf nachzubilden. Diese Art von Digitalisierung hängt von zwei Größen ab: der Abtastfrequenz und der Abtastauflösung. Die Abtastfrequenz bestimmt das zeitliche Intervall zwischen den Einzelabtastungen und damit die zeitliche Auflösung. Unter Abtastauflösung versteht man die Zerlegung des Wertes in Einzelstufen. Dabei spielt es eine Rolle mit welcher Anzahl Bits der Wert dargestellt wird: übliche Auflösung im Studiobereich sind 16 Bit ($2^{16} = 65535$ Abstufungen). Nach der Verarbeitung im Computer kann das umgerechnete Signal in ein analoges zurückgewandelt werden. Auch hier sin die Abtastfrequenz und die Abtastauflösung wieder die bestimmende Größen.

4.2.2 Codierung

Nach der Abtastung der Amplitudenwerte mittels der Pulsamplitudenmodulation, bei der das Signal zeitlich diskretisiert wurde, folgt die Amplitudenquantisierung. Die Amplituden werden als Zahlenwerte digital in Binärform abgelegt. Die als Liste hintereinander abgelegten Werte repräsentieren den Signalverlauf der Schallwelle über die Zeit hinweg und werden Wellenform (Waveform), Wellentabelle (Wavetable) oder Sample genannt. Der Begriff Sample hat also zwei Bedeutungen, erstens als einzelner Zahlenwert (Sample =Probe), der dem aktuellen Amplitudenwert entspricht, und zweitens als Liste aller Zahlenwerte eines Signals (z.B. Gitarren-Sample). Da bei der Quantisierung und Codierung eines akustischen Signals eine große Datenmenge anfällt, müssen bei Echzeitbearbeitungen schnelle und effektive Prozessoren angewandt werden. Hier kommen leistungsstarke Signalprozessoren zum Einsatz.

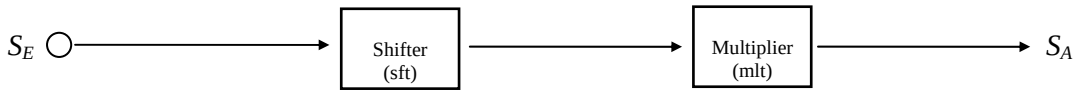
4.2.3 Digitale Klangbearbeitung

Nachdem die analogen Schallschwingungen in digitale Signale bzw. Zahlen umgewandelt worden sind, stellt sich die Frage wie man nun diese Zahlen verändern muß, um den Effekt zu erzielen, welchen analoge Geräte schon lange beherschen.

4.2.3.1 Verstärker (Amplifier)

Ganz einfach digital zu lösen ist die Verstärkung eines digitalen Audiosignals. Bei der Verstärkung handelt es sich im Prinzip um eine Multiplikation der einzelnen Samplewerte mit einem bestimmten Faktor. Dieser Wert liegt bei den meisten Anwendungen zwischen 0 und 2. Die Multiplikation mit 0 bedeutet eine Abschwächung des Signals bis auf Nullpegel. Eine Multiplikation mit 2 bedeutet eine Verdopplung der Pegel. Will man mehr Verstärkung erreichen, als nur eine Verdopplung, bedient man sich neben der Multiplikation dem Shiftens. Das digitale Wort geshiftet um 1 nach links bedeutet eine Multiplikation mit 2:

$0010\ 1010\ 0001\ 0111_B = 10775$
$0101\ 0100\ 0010\ 1110_B = 21550.$

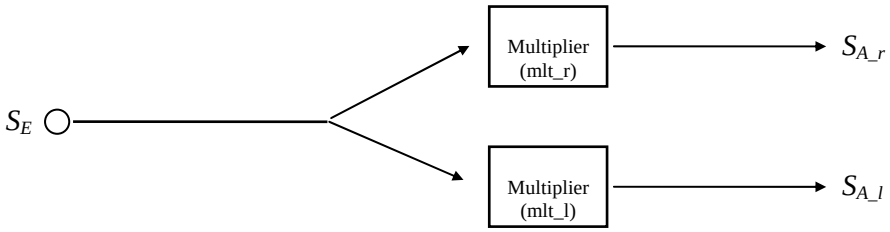


$$S_A = \left(S_E \cdot 2^{sft}\right) \cdot mlt$$

Diese Operationen kann z.B. ein DSP sehr schnell durchführen. Es besteht jedoch die Gefahr der Bereichsüberschreitung. Will man ein 16 bit Sample um den Faktor 4 verstärken, kann das Ausgangssample 18 bit breit sein. Um hier genügend reserven zu haben, werden 24 Bits Wortgrößen verwendet.

4.2.3.2 Panorama

Die Panorama eines Signals ist nah verwandt mit dem Balance-Regler auf unserer heimischen Hifi-Anlage. Je nach Stellung des Regler erklingt das Signal rechts oder links aus dem Lautsprecher.



Es handelt sich um ein Doppelverstärker, der das Signal an zwei Ausgänge weiterreicht. Die Summe der beiden Signale sollte konstant bleiben, d.h. die Stereoverteilung sollte keinen Einfluss auf die Gesamtlautstärke haben:

I. $S_{A\_r} = mlt\_r \cdot S_E$
II. $S_{A\_l} = mlt\_l \cdot S_E$
III. $mlt\_l + mlt\_r = 1$
IV. $S_E = S_{A\_l} + S_{A\_r}$

4.2.3.3 Digitale Filter

Die bekanntesten Filter in der Audio-Technik sind wahrscheinlich der Bass und der Höhenregler der Stereoanlage. Dreht man den Bassregler zu, werden die Bässe aus dem Signal bedämft oder anders gesagt nur die Höhen werden herausgefiltert und zum Hörer weitergeleitet. Genauso beim Höhenregler: Er regelt den Anteil der Höhen im Musiksignal.

Was sind nun genau Höhen und Bässe? Höhen sind die höherfrequenten Anteile der Musik (>3 kHz). Bässe sind die niederfrequenten Anteile (<300 Hz).Diese analogen Filter können sehr flexibel digital nachgeahmt und sogar noch verbessert werden.
Beim digitalten Filter handelt es sich um einen mathematischen Prozess. Ein Eingangszahlenfolge wird in eine Ausgangszahlenfolge umgerechnet.
Der einfachste digitale Filter ist der Mittelwertfilter. Gegeben sie ein Eingangssignal *x(n)* und ein Ausgangssignal *y(n),* wobei *n* als Index die einzelnen Samples bezeichnet. Ein „Vier-Sample-Mittelwertfilter“ kann durch Implementation folgender Gleichung realisiert werden:

$$y(n) = \frac{1}{4}\left[x(n) + x(n-1) + x(n-2) + x(n-3)\right]$$

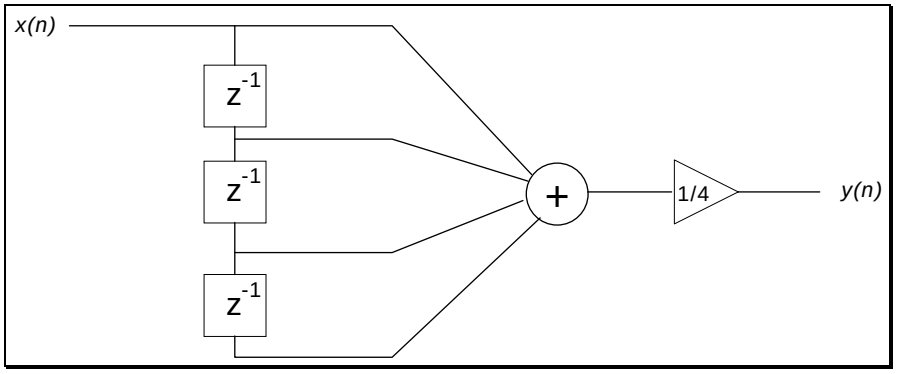


Abbildung 1 Mittelwert-Filter

Der Ausgang dieses Filters ist der Mittelwert aus dem aktuellen Sample x(n) und den drei vorhergehenden Samples. Abbildung 1 Mittelwert-Filter zeigt diese Gleichung als digitale Filterstruktur in Standardnotation mit Summierer als Kreis, Multiplizierer (0Verstärker) als Dreieck mit angegebenem Verstärkungsfaktor und Verzögerungsglieder als Rechteck. Die Bezeichnung z^{-1} bedeutet eine Verzögerung um ein Sample.
Die Funktion des Mittelwert-Filters ist leicht zu verstehen: Schnelle Veränderungen im Eingangssignal, d.h. hohe Frequenzen, werden durch die Mittelwertfunktioin geglättet; langsame Veränderungen oder tiefe Frequenzen werden kaum beeinflusst. Es handelt sich beim Mittelwert-Filter um einen Tiefpass-Filter. Tiefe Frequenzen können „passiern“, hohe Frequenzen werden gedämpft.
Ein sehr einfacher Hochpassfilter kann man mit dem sog. One-Zero-Filter realisieren.

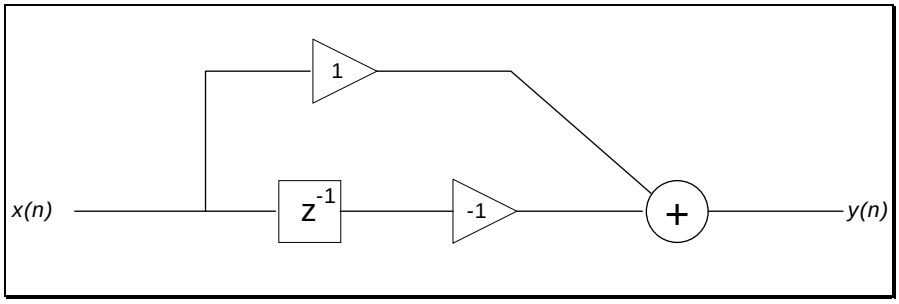


Abbildung 2 One-Zero-Filter

Hier wird anstatt der Summe die Differenz gebildet, so wirkt das Filter langsamen Signaländerungen, d.h. niedrigen Frequenzen entgegen. Hingegen werden schnelle Wertsprünge hoher Frequenzen hervorgehoben.

Die Wirkung dieser Filter ist jedoch höchst ineffizient, weswegen sie in der Audiotechnik nur selten eingesetzt werden.
Qualitätsmerkmale von digitalen Filtern sind:

- Flankensteilheit (od. Güte):
  Die Einheit der Flankensteilheit ist [1]dB/Oktave: d.h. Um wieviel dB das Signal bei einer Frequenzverdoppelung ([2]Oktave) abgeschwächt wird.Typische Werte bei digitalen Filtern sind 6 - 12 - 24dB/Oktave.
- Phasenverhalten:
  Bei analogen sowie bei digitalen Filtern kommt es in verschiedenen Fällen zu unerwünschten Phasenverschiebungen. Diese können zur Auslöschung ganzer Frequenzbereiche führen.
- Resourcenverbrauch:
  Ein digitales Filter verbraucht Rechenzeit. Diese Zeit muß so gering wie möglich gehalten sein, um es überhaupt in Echtzeit realisieren zu können oder um Kosten für ultraschnelle und ultrateure Signalprozessoren zu sparen.

Eine Weiterentwicklung des Mittelwert-Filters ist die Wichtung der in dei Rechnung einbezogenen Samples. Die Formel für dieses Filter lautet:

$$y(n) = \sum_{k=0}^{N-1} b(k)x(n-k)$$

mit *b(0)* bis *b(N-1)* als Filterkoeffizienten und *x(n)* als Eingangsdaten. Diese Art von Filter nennt man auch [3]FIR-Filter.

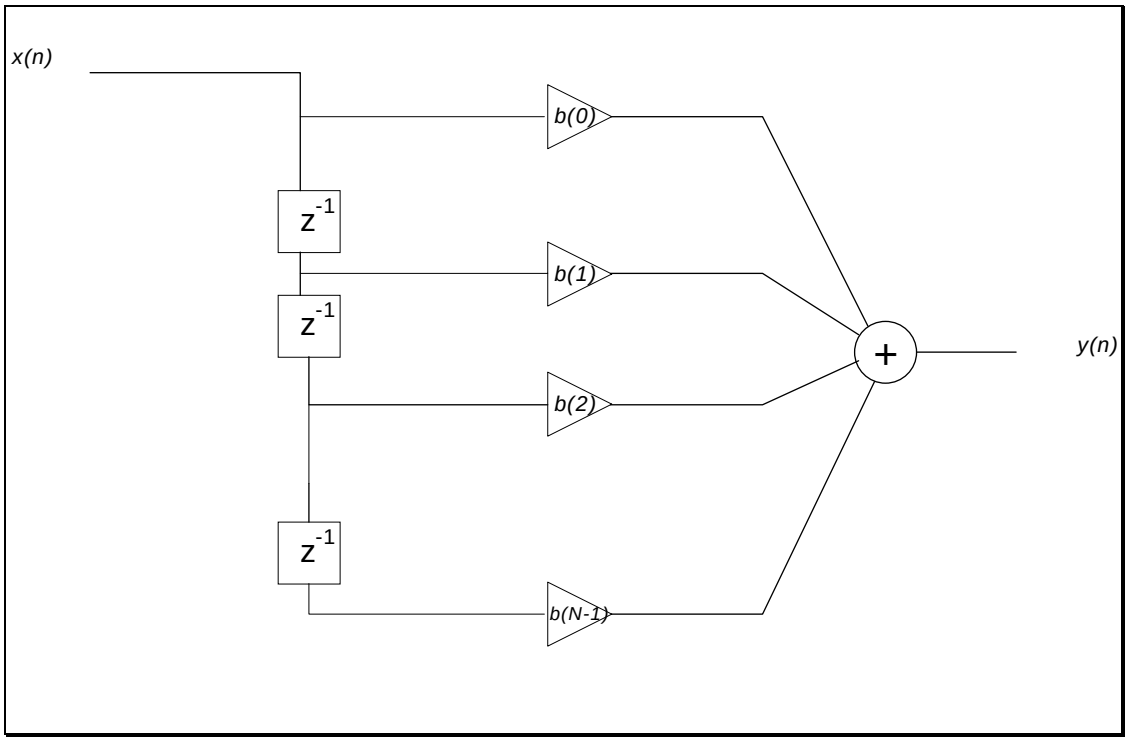


Abbildung 3 Nichtrekursiver FIR Filter

[1] dB (Dezibel) logarithmische dem Hörempfinden angepasste Einheit
[2] Oktave bezeichnet in der Musik ein Intervall von 12 Halbtönen; dies entspricht einer Frequenzverdoppelung
[3] Finite-Impulse-Response od. nichtrekursiver Filter

Bei bestimmter Auswahl der Koeffizienten *b(n)* lassen sich Filter mit genaubestimmbaren Frequenzverläufen herstellen. Bei steilflankigen Filtern geht die Anzahl der Koeffizienten über das heute technisch machbare hinaus, so daß auch diese Art von Filter noch nicht für den Bereich der Audio-Technik ausreicht.
Um steile Flanken in der Filtercharakteristik zu bekommen, ist es nötig viele vorhergehende Samples zu berücksichtigen. Es gibt einen in der Konstruktion recht einfachen, jedoch im mathematischen Sinne recht komplexe Lösungs: der [4]IIR-Filter. Durch Rückkopplung des Filterausgangs werden quasi alle vorhergehenden Samples berücksichtigt:

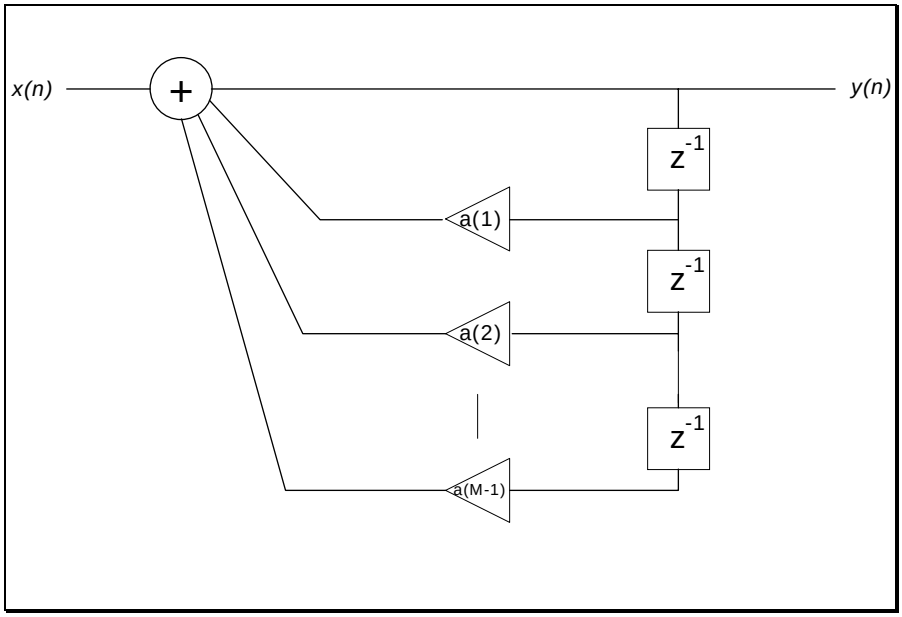


Abbildung 4 einfacher rekursiver IIR-Filter

Die allgemeine Formel des einfachen IIR-Filters lautet:

$$y(n) = x(n) + \sum_{k=1}^{M} a(k)y(n-k)$$

Die Schwierigkeit bei IIR-Filtern ist dessen Stabilitätsverhalten. Die Summe der Koeffizienten *a(k)* muss immer kleiner 1 sein, weil sonst das Filter in einen instabilen Zustand fällt. Bei einem Wert größer 1, würde jeder Wert im Betrag erhöht werden, wobei irgendwann der Wertebereich überschritten wäre. Ein IIR-Filter ist im Vergleich zu einem FIR-Filter wesentlich effizienter, weil sie weniger Speicher und Multiplikationen benötigen. In der Audiotechnik findet man vorwiegend Filter die sowohl einen rekursiven sowie einen nichtrekursiven Teil enthalten, so das eine Gleichung für digitale Filter geschrieben werden kann als

$$y(n) = \sum_{k=0}^{N-1}\left[b(k)x(n-k)\right] + \sum_{k=1}^{M}\left[a(k)y(n-k)\right]$$

mit *a(k)* als rekursive Rückkoppelungs-Koeffizienten und *b(k)* als nichtrekursive Koeffizienten.

[4] Infinite-Impulse-Response od. rekursiver Filter

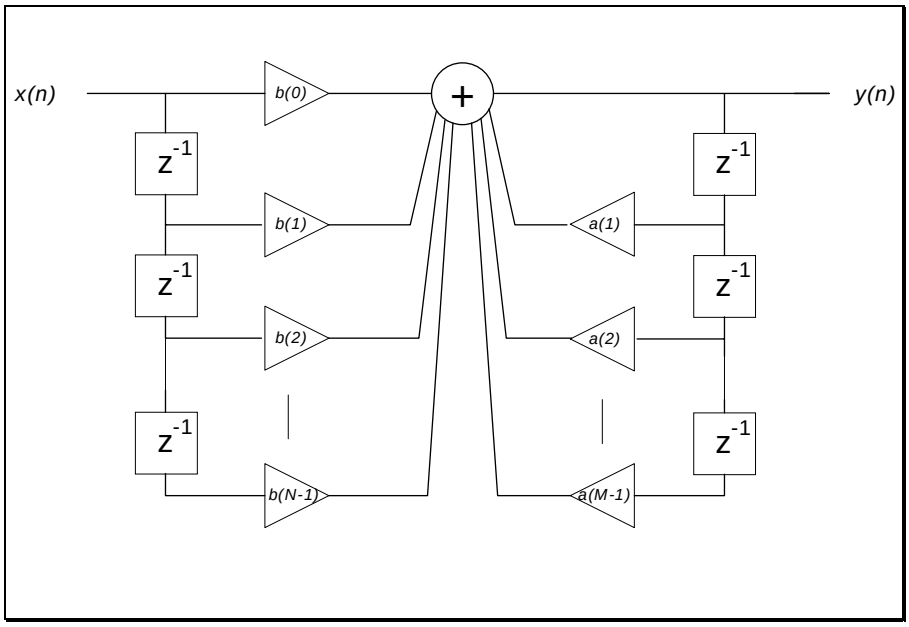


Abbildung 5 digitaler Filter mit nichtrekursiven und rekursiven Teilen

4.3 Signal Prozessoren

4.3.1 Architektur

Ein digitaler Signalprozessor unterscheidet sich in senem Aufbau nicht Grundlegend vom Microprozessor. Auch hier findet man die Grundeinheiten eines Prozessorsystems: Zentraleinheit, Speicher, Ein-/Ausgabeeinheit. Auch die prinzipiellen Funktionen sind vergleichbar: Instruktionen laden, Daten aus dem Speicher holen, Programm ausführen, Daten verarbeiten und gegebenfalls zurückschreiben oder ausgeben usw. Der wichtigste Unterschied sind Spezialfunktionen und das Einsatzgebiet. Der Mikroprozessor ist in seiner Architektur so konzipiert, daß er für eine breite Palette von Anwendungen geeignet ist. Signalprozessoren dagegen umfassen einen kleineren Befehlssatz und sind in ihrem Hardware-Aufbau für ein spezielles Aufgabengebiet zugeschnitten und optimiert.
Das herausragendste Merkmal von digitalen Signalprozessoren (DSP) ist deren Fähigkeit Multiplikationen und Akkumulationen, welche bei der digitalen Signalbearbeitung wie Filterung oder Fouriertransformation von zentraler Bedeutung sind, sehr schnell durchzuführen. Der in diesem Projekt verwendete Signalprozessor benötigt für eine Multiplikation mit paralleler Addition einen Maschinenzyklus von 40ns.
Die meisten DSPs sind nach der Harvard-Architektur aufgebaut. In diesem Schema sind Daten und Instruktionen in getrennten und unabhängigen Speichern gehalten. Dies ist nötig um mehrere Operationen parallel ausführen zu können. Durch das Vorhandensein mehrerer Adress- und Datenbusse für Daten- und Programmspeicher und durch sogenanntes Pipelining kann die parallele Verarbeitung der Daten und der Datenfluss realisiert werden.
Da Signalprozessoren einen grossen Fluß von Daten zu bewältigen haben, sind auch die Ein-/Ausgabe-Kanäle optimiert. Digitale Signale können über externe parallele Busse oder über schnelle serielle Schnittstellen übertragen werden.

4.3.2 Programmierung

Im allgemeinen werden DSPs als Co-Prozessoren eingesetzt. Für die Programmentwicklung werden Cross-Compiler (Assembler oder C), Linker und Simulatoren angeboten. Die Softwareentwicklung geschieht vorwiegend auf dem PC, d.h. Programme werden mit Werkzeugen (Editoren, Compiler, Emulatoren) auf dem PC erstellt und getestet und erst in

einem zweiten Schritt mit einem Crosscompiler in den Speicher des DSPs übertragen und ausgeführt. Weil DSP-Applikationen unter Echtzeitbedingungen ablaufen und daher zeitkritisch sind, müssen zumindest die wichtigsten in Assembler geschrieben und optimiert werden. Wegen der speziellen Architektur und der auftretenden Parallelität ist es nicht ganz trivial, Programme für DSP Chips zu erstellen. In der Audiotechnik muß die Applikation so implementiert sein, daß die Ausführungszeit unterhalb der Samplezeit liegt. In jedem Durchlauf muß ein neues Sample verarbeitet werden. Im Beispiel des Mischpultes würde sich der maximale Zeitbedarf folgendermaßen berechnen:

$$T = \frac{1}{fs} \Rightarrow 22{,}68ms \approx \frac{1}{44100Hz}$$

4.4 Programmierbare Logiken

Programmierbare Logikbausteine werden immer poulärer und sind nicht mehr aus der modernen Elektronik mehr wegzudenken. Früher war es, aufgrund der hohen Kosten der Entwicklungswerkzeuge, nur großen Firmen vorbehalten diese Technik zu verwenden. Heute gibt es schon relativ günstige Entwicklungstools auch für kleine Firmen. Der Hauptvorteil eines solchen Chips ist die Realisierung der Systemintegration auf kleinstem Raum. Ein einziger Chip kann zig PALs oder GALs ersetzen. Es gibt drei Arten von programmierbaren Bausteinen: CPLDs (Complex Programmable Logic Devices), FPGAs (Field Programmable Gate Arrays) und ASICs (Application Specified Integrated Circuits). Wobei der ASIC nicht programmierbar im eigentlichen Sinne ist. Der ASIC ist ein masken-programmierbares Gate Array. D.h. der rohe Chip besteht aus vielen kleinen Gattern, welche noch nicht verbunden sind. Es wird nun eine Maske entworfen, die applikationsspezifischen Verbindungen der Gatter darstellt. Der ASIC-Rohling wird durch Aufbringen dieser Verbindungen zum fertigen Chip. CPLDs besitzen eine sogenannte Segmented-Block-Architektur. Sie sind aus PAL-ähnlichen Blöcken aufgebaut, die über eine zentrale Verbindungsmatrix verschaltet werden.Durch die Anordnung der komplexen Module (Blöcke) um die Schaltmatrix sind CPLDs zwar weniger flexibel als z.B. ASICs, erzielen jedoch durch Optimierung in den Blöcken selber sehr schnelle Verarbeitungs-geschwindigkeiten. Die Programmierung von CPLDs basiert auf der von PALs und GALs bekannten EPROM-, EEPROM-, oder FLASH-Technik und zum anderen die SRAM-Technik, bei der die Logik über einen Speicherbaustein bei jedem Start neu in das IC geladen wird. FPGAs sind im Aufbau den ASICs ähnlicher, werden jedoch genauso programmiert wie CPLDs. Für die Charakterisierung von Logik-ICs wird oftmals der Begriff ´Körnung´. Er bezieht sich auf die Größe und der Komplexität der einzelnen Logikblöcke und meint eigentlich deren Aufnahmekapazität: je weniger Logik in ein Modul paßt, umso feinkörniger ist der Baustein. Beinhaltet ein Modul bespielsweise keine Flipsflops, muß man diese aus eunzelnen Blöcken aufbauen - es handelt sich um eine feinkörnige Architektur. Je mehr Blöcke untereinander verbunden werden, um so mehr Verdrahtungsresourcen sind erforderlich. Und viele Verbindungsleitungen benötigen natürlich Platz und sind daher teuer. Andererseits führen nicht vollständig ausgenutzte Logikblöcke bei grobkörnigen Strukturen ebenfalls zu einem erhöhten Platzbedarf.

4.5 Mikrocontroller

Mikrocomputer sind heute in fast allen elektrischen Geräten anzutreffen: Personenwaagen, Uhren, Steuerung von Robotern, Spülautomaten, Hifi-Anlage usw. Moderne Schaltungen werden nicht mehr wie früher aus TTL- oder CMOS Standardbausteinen aufgebaut, da der Microcomputer die selben Aufgaben viel flexibler und preiswerter lösen kann. Die Harwareentwicklung reduziert sich dabei auf ein Minimum und die Schwierigkeiten verlagern sich auf die Erstellung einer funktionierenden Software.

Der Unterschied zwischen einem Mikrocontroller und einem Mikroprozessor ist nicht klar definiert. Der Microprozessor, wie man ihn z.B. in der Datenverarbeitung in PC´s einsetzt ist für eine hohe Datendichte optimiert. D.h. er besitzt einen breiten Datenbus (32Bit,64Bit) und ist auf hohe Rechenleistung getrimmt. Der Mikrocontroller legt seinen Schwerpunkt auf die Einzelbitverarbeitung und einer hohen I/O Kapazität. Ein Microcontroller hat ein großes Aufgabengebiet: Tastaturen abfragen, serielle Schnittstellen bedienen, externe Co-Prozessoren bedienen, Signale abfragen, auf Signale reagieren (Interrupts), eine Anzeige ansteuern und eine Echtzeit zur Verfügung stellen. Um diese Aufgaben zu erfüllen hat er viele innere Baugruppen. Im PC befinden sich diese Baugruppen außerhalb des Prozessors. Ein Microcontroller ist somit ein Microprozessor inklusive o.g. Baugruppen.
Die Programmierung eines Microcontrollers findet meistens auf dem PC statt. Hier stehen verschiedene Entwicklungswerkzeuge zur Verfügung (Editor, Compiler, Linker, Simulator und Debugger). Zum großen Teil werden Microcontrollerprogramme in C oder Assembler geschrieben. Sogenannte Emulatoren sind dafür zuständig den Microcontroller in der konkreten Schaltung in der Entwicklungsphase zu ersetzen, um jederzeit Einblick über den Zustand des Mikrocontrollers zu haben oder ihn Steuern zu können (Breakpoints, Registereingriffe, Speichermanipulationen, Trace...).

5 Aufgabenstellung

5.1 Kurzbeschreibung

Die Hardware ist aufgeteilt in drei Sektionen:

1. Hostsektion
   Diese Sektion beinhaltet den Hostprozessor, der für die Gesamtsteuerung des Systems zuständig ist. Im einzelnen ist dies der HITACHI H8/3003 Prozessor, ein RAM,das Betriebsystem-EPROM, ein Resetcontroller und eine RS232-Schnittstelle.
2. Kanalsektion
   Die Kanalsektion ist für die Verarbeitung von 8 analogen Audioeingängen und 8 analogen Audioausgängen zuständig. Sie besteht aus vier Stereokanalzügen. Ein Kanalzug beinhaltet einen DSP , ein Stereo-AD/DA-Wandler und ein Stereogainkontroller, zur Einstellung der Eingangsempfindlichkeit des AD-Wandlers.
3. Mixsektion
   Die Mixsektion ist für die Mischung und Verteilung der gesamten Audiodaten zuständig. Diese Aufgabe übernehmen ein FPGA und ein weiterer DSP. Für zukünftige Erweiterungen gibt es einen Steckplatz wo weitere Kanalsektionen angeschlossen werden können.

Die Aufgabe ist die Entwicklung der einzelnen Softwarekomponenten:

- Hostprozessorprogramm
  - Steuerung der Kanalsektion
  - Steuerung der Mixsektion
  - Kommunikation mit dem PC
  - Parameterberechnung der digitalen Filter
  - Signalpegelüberwachung (Metering)
- Mixprozessorprogramm
  - Einlesen der Audiodaten
  - Berechung der Ausgangsdaten je nach Einstellung
  - Ausgabe der Daten an den FPGA
- FPGA-Layout
  - Parallele Schnittstelle zum MixDSP

- 16 serielle Schnittstellen zur Kanalsektion
- Wandlung von parallelen Daten in serielle
- Kanalsektionrouting

5.2 Die Aufgaben im Einzelnen

5.2.1 Hostprozessor

Der Hostprozessor übernimmt die Verwaltung des ganzen Boards.

- Booten und Steuern der DSPs der Kanalsektion (Kanal-DSPs)
  Über die I/O Ports des Hostprozessors werden die Kanal-DSPs resetet, initialisiert.und gesteuert. Es sind Routinen zu entwickeln, die die Kommunikation zwischen HostCPU und KanalDSP ermöglichen, Koeffizienten des KanalDSPs wie z.B. digitale Filterkoeffizienten berechnen und übertragen bzw. Daten aus dem KanalDSP lesen, um z.B. Metering durchzuführen.
- Initialisierung des Gainkontrollers (LM1972N). Der Gainkontroller ist ein digital programmierbares Potentiometer. Er besitzt eine serielle Schnittstelle zur HostCPU und regelt die Eingangsempfindlichkeit des analogen Eingangsverstärkers.
- Booten und Steuern des DSPs in der Mixsektion
  Der MixDSP hat im Anschaltzustand kein Programm im Speicher. Der Hostprozessor muß über seine I/O Ports das Programm übertragen und danach den DSP starten. Im weiteren Betrieb müssen Koeffizienten d.h. Parameter zur Steuerung der Audiosignale beeinflußt werden. Es müssen geeignete Routinen geschrieben werden welche die Kommunikation mit dem MixDSP ermöglichen.
- Programmieren und starten des FPGAs
  Im Anschaltzustand ist im FPGA noch keine Funktion implementiert. Das Programm des FPGAs liegt im ROM des Hostprozessors (ca. 3,9kB). Es muß beim Hochfahren des System über I/O-Ports seriell in den FPGA übertragen werden.
- Bedienung der seriellen Schnittstelle
  Ankommende Kommandos des PCs müssen an die DSPs oder Gainkontroller weitergegeben werden. Weiter müssen aktuelle Zustände der DSPs an den PC zurückgeleitet werden, um sie auf dem Bildschirm sichtbar zu machen.

5.2.2 Mixprozessor

Der Mixprozessor übernimmt einen Teil des Routings der Audiosignale auf dem Board. Es ist ein DSP-Programm zu schreiben welches in einem Samplezyklus (20µs) 10 Audiokanäle einließt, in einer Matrix miteinander mischt und sie auf 10 Kanälen wieder ausgibt. Die einzelnen Knotenpunkte der Matrix können vom Hostprozessor gesteuert werden.

5.2.3 FPGA

Es ist ein Design zu entwickeln, welches folgende Aufgaben erfüllt. Der FPGA übernimmt die Adaptierung zwischen der Datenbus/Adressbus-Schnittstelle des Mixprozessors und den seriellen Ein-/Ausgängen der 4 KanalDSPs bzw. 4 AD/DA-Wandler. Der FPGA wird von der Mixprozessorseite angesprochen wie ein dynamisches RAM. Die einzelnen Audiosamples haben eine Wortbreite von 16 Bit. In einem Samplezyklus muß der FPGA 8 seriell übertragene Audiosamples dem Mixprozessor als „DRAM Speicherzellen“ zu Verfügung stellen. Umgekehrt müssen die vom Mixprozessor geschriebenen „DRAM Speicherzellen“ seriell an die DSPs und die AD/DA weitergeben werden.
Weiterhin ist der FPGA verantwortlich für die synchronisation des gesamten digitalen Audiobereichs der Schaltung. Aus der Mainclock (FS512, 22,5792 MHz) sind folgende Taktsignale zu erzeugen:

- WCLK (FS1 ,Wordclock 44,1 kHz)
- BCLK (FS32, Bitclock 1,4112 MHz)
- FS256 (Systemclock für AD/DA-Wandler 11,2896 MHz)

6 Hardware

6.1 Blockschaltbild

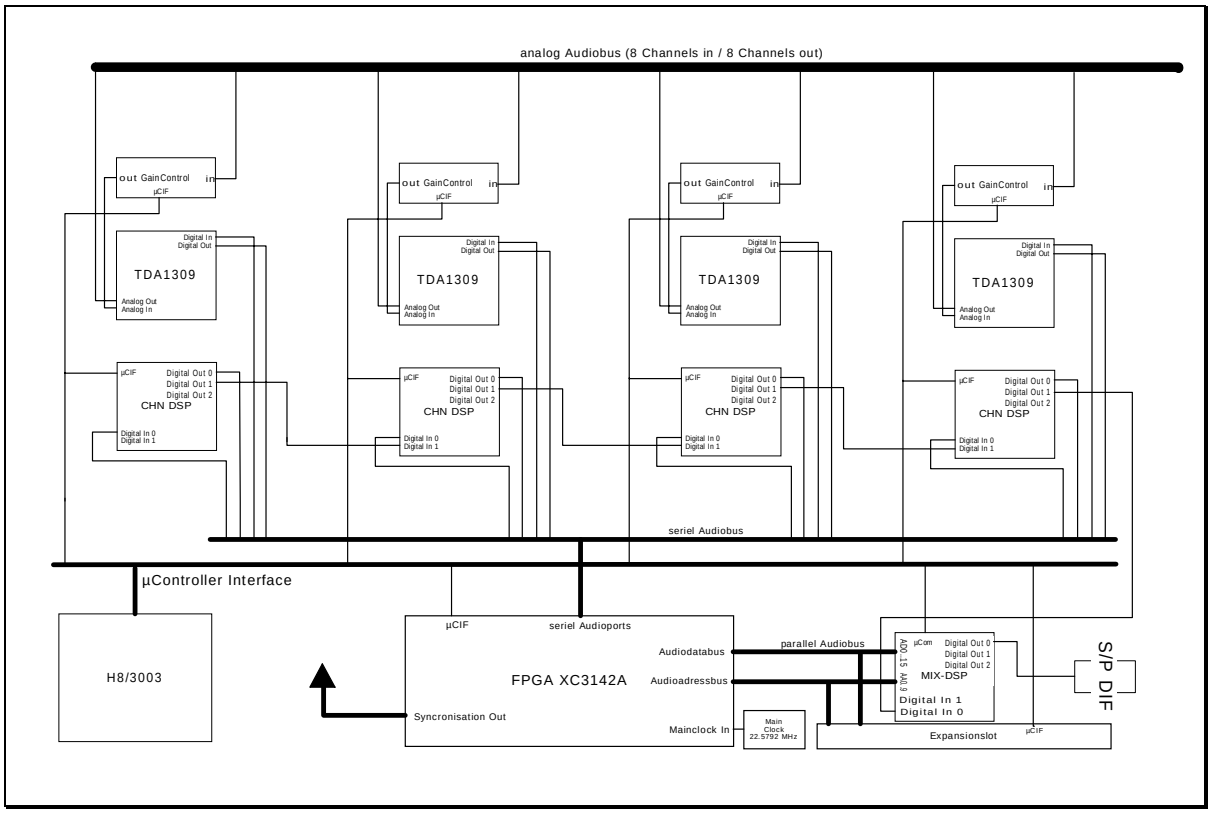


Abbildung 6 Blockschaltbild des Digimixers

6.2 Hostprozessor H8/3003

6.2.1 Einführung

Der H8/3003 ist ein zu sehr großen Leistungen fähiger Mikrocontroller. Er beinhaltet eine H8/300H CPU. Die H8/300H CPU hat intern eine 32-bit Architektur mit sechzehn 16-bit Registern und einem kleinen jedoch geschwindigkeitsoptimierten Befehlssatz. Es steht ein linearer Adressraum von 16-Mbyte zu Verfügung und ist ideal für Echtzeitkontrolle.
Der Mikrocontroller beherbergt u.a. RAM, ein 16-bit Timer-Einheit ([5]ITU), zwei serielle Interfaces, 8 A/D Wandler und 58(!) I/O Ports.
Die große Anzahl der I/O Ports war der ausschlaggebende Punkt bei der Wahl des Chips für dieses Projekt.

6.2.2 Aufbau

6.2.2.1 Blockdiagramm

(fehlt: Blockdiagramm Hitachi-Handbuch Seite: 5)

[5] integrated timer-pulse unit

6.2.2.2 Beschaltung

Der H8/3003 wird mit 16 MHz getaktet. Da fast alle Portpins benötigt werden, sind Sonderfunktion wie A/D-Wandler, DMA und die Time-pulse unit (ITU) abgeschaltet. Dem Prozessor steht ein EPROM von 128 kByte (27C010) und batteriegepufferter SRAM-Speicher (62256) von 32 kByte zu Verfügung. Die Pufferung des RAM soll ein verlorengehen der Daten nach dem Ausschalten verhindern. Die serielle Schnittstelle (SCI1) wird zur Kommunikation mit dem PC verwendet. Am seriellen Ausgang des Hitachi H8/3003 steht jedoch nur TTL-Pegel (0..5V) zu Verfügung. Die RS232-Norm schreibt jedoch vor, daß der Pegel für ein Low-Signal -12V und der Pegel für ein High-Signal +12V betragen muß, um eine höhere Übertragungssicherheit zu erhalten. Um aus TTL-Pegel in RS232-Pegel zu wandeln und umgekehrt, wird der Baustein MAX232 verwendet. Damit der Prozessor beim Hochfahren des Systems ein definiertes Resetsignal erhält, wird der Resetkontroller 7705 eingesetzt. Über die I/O-Ports sind die Hostschnittstellen folgender Baugruppen angeschlossen:

- FPGA XC3042A
- MixDSP
- 4 KanalDSPs
- 4 Gaincontroller

6.2.3 I/O Ports

Der H8/3003 stellt insgesamt 9 I/O Ports verschiedener Breite zu Verfügung. Verwendet man alle I/O Pins als Portleitungen erhält man 58 Anschlüsse, die zum großen Teil als Eingang, oder als Ausgang geschaltet werden. In folgender Tabelle sind die Belegungen der Ports aufgelistet:

Tabelle 1 Belegung der Portpins

Portpin	Signal	Beschreibung
PA0 ... PA5 (OUTPUT)	Hostadressleitung	Für zukünftige Erweiterungen am Expansionslot gedachte Adressleitung, um externe Hostschnittstellen über einen externen Adressdekodierer adressieren zu können.
PA6 (OUTPUT)	XOVLIRQ	Für zukünftige Erweiterung um die Übersteuerungssignale (Overload) des AD-Wandlers per Interrupt zu behandeln. Das vorangestellte ´X´ im Signalnamen bedeutet, das dieses Signal für externe, am Expansionsslot befindende Baugruppen gedacht ist.
PA7 (OUTPUT)	XOVLSEL	Da bei einer auftretenden Übersteuerung immer der gleiche Interrupt ausgelöst wird, muß der Hostprozessor herausfinden, in welchem Kanal die Übersteuerung stattgefunden hat. Mit dem Signal XOVLSEL selektiert er die durch die Hostadresse angewählten Kanäle und liest ihren Übersteuerungszustand aus.
PB0 (OUTPUT)	MIX_RES\	Dieses Signal setzt den MixDSP beim Hochfahren in einen definierten Zustand. (MixDSP-Reset)
PB1	Multifunktionspin: SSEL	Folgende Signale werden gesteuert:

(OUTPUT)		HXS (MixDSP): Frame Synchronisation Nach einem High-Puls sind die nächsten 8 Datenbits auf HX gültig. IFCD (KanalDSP): Unterscheidungssignal zwischen Daten und Kommandos
PB2 (OUTPUT)	FPGARST\	Low-Pegel leitet den Start der Programmierung des FPGAs ein.
PB3 (OUTPUT)	Multifunktionspin: HOSTCLK	Folgende Signale werden gesteuert: HRBCK (MixDSP) Fallende Flanke bewirkt Datenübernahme von Signal HR beim Empfangen von Hostdaten HXBCK (MixDSP) Fallende Flanke bewirkt Datenwechsel von Signal HX beim Senden zur HostCPU
PB4 (OUTPUT)	XDSPCS/	Für zukünftige Erweiterungen: Chipselect eines externen KanalDSPs adressiert durch die Hostadressleitungen
PB5 (OUTPUT)	XDSPRST/	Für zukünftige Erweiterungen: Resetsignal eines externen KanalDSPs adressiert durch die Hostadressleitungen
PB6 (OUTPUT)	XGAINCS	Für zukünftige Erweiterungen: Chipselectsignal für externe Gainkontrollers adressiert durch die Hostadressleitungen
PB7 (INPUT)	DONE-PG\	Eine fallende Flanke signalisiert die erfolgreiche Programmierung des FPGAs.
PC0	OVLSEL	Für zukünftige Erweiterungen: Der FPGA speichert eingetretene Übersteuerungen in einem Schieberegister. High-Pegel am OVLSEL selektiert das Schieberegister, welches dann synchronisiert mit der HOSTCLK ausgelesen wird.
PC1	MIX_CS\	Chipselectsignal des MixDSPs
PC2 ... PC5	reserviert	
PC6	DSP3_RS\	Resetsignal des 3. KanalDSP
PC7	DSP3_CS\	Chipselectsignal des 3. KanalDSP
P54	GAIN78	Chipselectsignal für den 4. Gaincontroller zuständig für Kanal 7 und 8.
P55	DSP4_RS	Resetsignal des 3. KanalDSP
P56	DSP4_CS	Chipselectsignal des 3. KanalDSP
P57	GAIN56	Chipselectsignal für den 3. Gaincontroller zuständig für Kanal 5 und 6.
P60	XFPGA_CS	Für zukünftige Erweiterungen: Chipselectsignal für externe FPGAs adressiert

		durch die Hostadressleitungen
P61		reserviert
P62	FPGA_CS	Chipselectsignal des FPGAs
P70	\ACK	Signaleingang zur Absicherung der Datenübertragung zwischen Hostprozessor und KanalDSP
P71	STATOUT	Datenleitung der Übersteuerungskontrolle Der FPGA sendet über diese Leitung welche Kanäle übersteuert sind.
P72	HX	Datenleitung zwischen MixDSP und Hostprozessor. Der Hostprozessor kann über diese Leitung Daten vom MixDSP lesen. Synchronisiert werden diese seriellen Daten mit der HOSTCLK.
P73	EMPTY\	Dieses Signal signalisiert einen leeren Empfangsbuffer des MixDSP.
P74	DRDY\	Dieses Signal signalisiert, daß sich im Sendebuffer des MixDSP ein abzuholendes Byte befindet.
P75	IFDO	Datenleitung vom KanalDSP zum Hostprozessor. Über diese Leitung kann die HostCPU sämtliche Daten aus dem RAM des KanalDSP lesen.
P76 u. P77		reserviert
P80 u. P81		reserviert
P82	OVLIRQ	Interrupteingang um die Übersteuerungssignale (Overload) des AD-Wandlers per zu behandeln.
P83	RAMCS\	Chipselect des RAM 62256
P84	ROMCS\	Chipselect des EPROMS 27C010

Zu jedem Port gibt es zwei Register; das *data direction register* und das *data register* .
Im *data direction register* wird festgelegt, ob es sich um ein Eingang oder um ein Ausgang handelt.

6.3 Analogsektion

6.3.1 Blockschaltbild

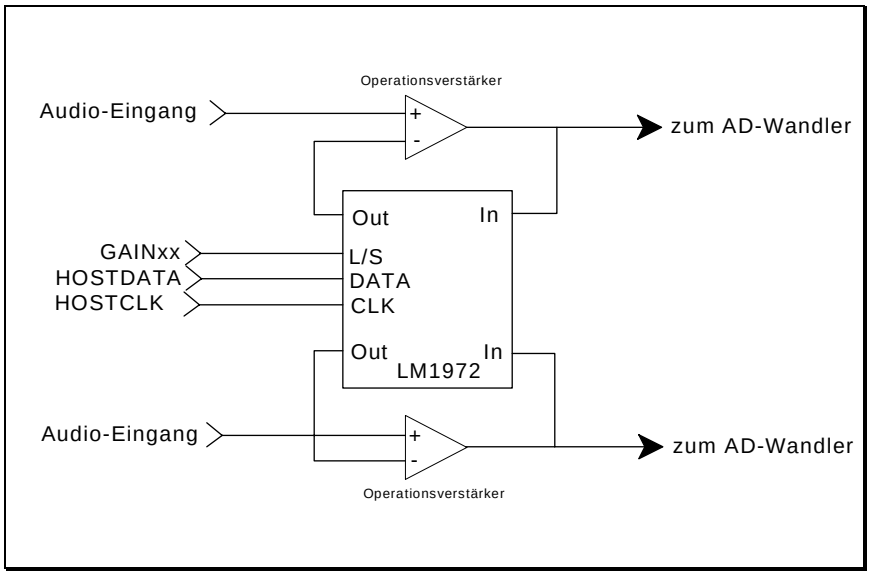


Abbildung 7 Eingangsschaltung

Das digitale Potentiometer LM1972 ist in die Rückkoppelschleife des Eingangsoperationsverstärker geschaltet. Um so höher die Dämpfung des LM1972 ist, desto höher ist die Verstärkung des Eingangssignals. Die Möglichkeit die Eingangsempfindlichkeit einstellen zu können ist aus verschiedenen Gründen nötig. An die Audioeingänge können sehr verschiedene Audioquellen angeschlossen werden:

- Mikrofone
- Gitarren
- Keyboards
- CD-Player ...usw

Jede dieser Quellen haben verschiedene maximale Ausgangsspannungen. Um die 16 Bit Auflösung des AD-Wandlers völlig ausnutzen zu können, ist es nötig den Eingangspegel so anzupassen das der AD-Wandler bei maximalpegel des Eingangssignal möglichst auch den maximalen Digitalwert ausgibt.

6.3.2 Gainkontroller

Der LM1972 ist ein digitales Potentiometer mit 128 (2^7) Rasterungen. Die hier verwendete Ausführung stellt 2 Kanäle zu Verfügung. Ein LM1972 bearbeitet einen Stereokanal. Das digitale Interface zum Hostprozessor besteht aus drei Leitungen: Chipselect, Data und Clock. Im gesendeten Datenwort ist jeweils der angesprochene Kanal und der einzustellende Wert enthalten. Der Baustein stellt anhand dieses Wertes den Dämpfungsfaktor mit Hilfe von sog. Tapswitsches ein. Der Eingang des Schieberegisters hängt permanent an der Datenleitung *HOSTDATA*. Durch einen Impulse an *LOAD/SHIFT* wird der Inhalt des Schieberegisters in das Latchregister übertragen und vom Decoder ausgewertet. In dem Modul Switch-Drive befinden sich Treiberstufen um den erhöhten Stromverbrauch der Tap-Switches decken zu können.

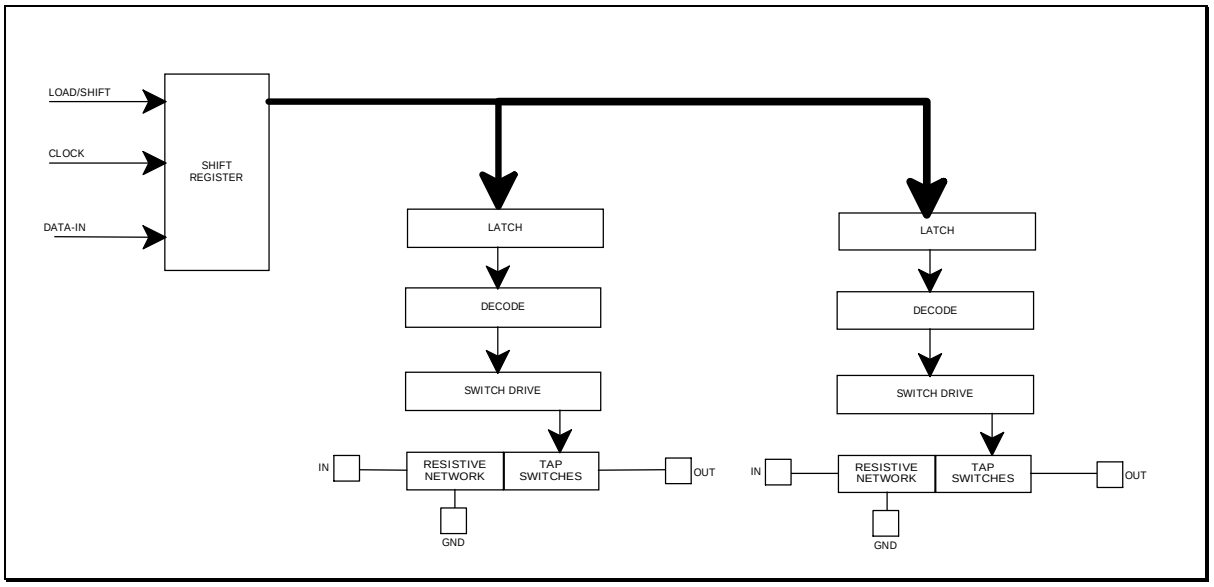


Abbildung 8 interne Struktur des LM1972

6.4 AD/DA-Wandler

6.4.1 Beschreibung

Der TDA1309H beinhaltet einen DA- und einen AD-Wandler. Der ADC (analog-digital-converter) ist als 1 Bit Sigma Delta Wandler ausgeführt. Er arbeitet mit einem Überabtastungsfaktor von 128 (≈ 5.6 MHz). In drei Filterstufen wird der 1 Bit-Datenstrom in 18 Bit Worte umgewandelt. Die 16 oberen Bits werden mit einer Frequenz von 44.1 kHz an das digital Interface der seriellen Schnittstelle weitergegeben.

6.4.2 Schnittstellen

Der AD/DA-Wandler besitzt eine serielle Schnittstelle, die mehrere Formate unterstützt. Die Schnittstelle besteht aus drei Anschlüssen: BITCLOCK, WORDCLOCK, DATA. Die Eingangsseite und die Ausgangsseite sind identisch. Die Synchronisationssignale WORDCLOCK und BITCLOCK müssen von einer externen Baugruppe geliefert werden. Eine WORDCLOCK hat die Länge von 32 BITCLOCKS. Jede steigende Flanke einer BITCLOCK synchronisiert ein Datenbit.

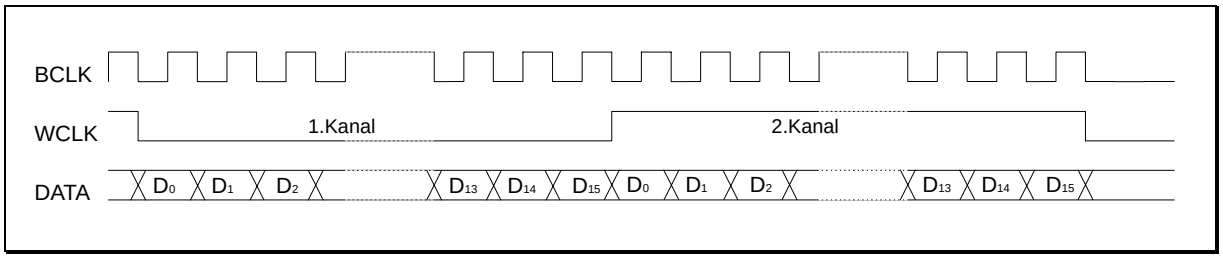


Abbildung 9 Datenformat der seriellen Audiodaten

(fehlt: könnte etwas mehr sein -> Application Note Phillips)

6.5 Kanal-Signalprozessor

6.5.1 Funktionsbeschreibung des Kanal-DSPs

Der KanalDSP ist für die rechenintensive Filterbearbeitung jedes Kanales zuständig. Er bekommt serielle Audiosignale vom AD-Wandler, bearbeitet sie mit 4 digitalen Filtern und gibt sie weiter an den FPGA. Der Hostprozessor kann während des Betriebes Filterkoeffizienten ändern, ohne das es im Audiodatenstrom zu Aussetzern kommt. Desweiteren ist der KanalDSP für die Mischung des Stereohauptbusses zuständig (Signale: CHAIN_12, CHAIN_23, CHAIN_34, CHAIN_4M). Jeder KanalDSP hat einen zweiten seriellen Eingang und einen zweiten seriellen Ausgang. Der zweite Ausgang eines DSPs ist jeweils an den zweiten Eingang des nächsten DSPs gekoppelt. Der letzte DSP ist an den MixDSP angekoppelt der diese Stereosumme (Summe der 8 Ausgänge der KanalDSPs auf zwei Kanäle je nach Einstellung verteilt) weiterverarbeitet (siehe Abbildung 6 Blockschaltbild des Digimixers).

6.5.2 Interne Struktur des Kanal-DSPs

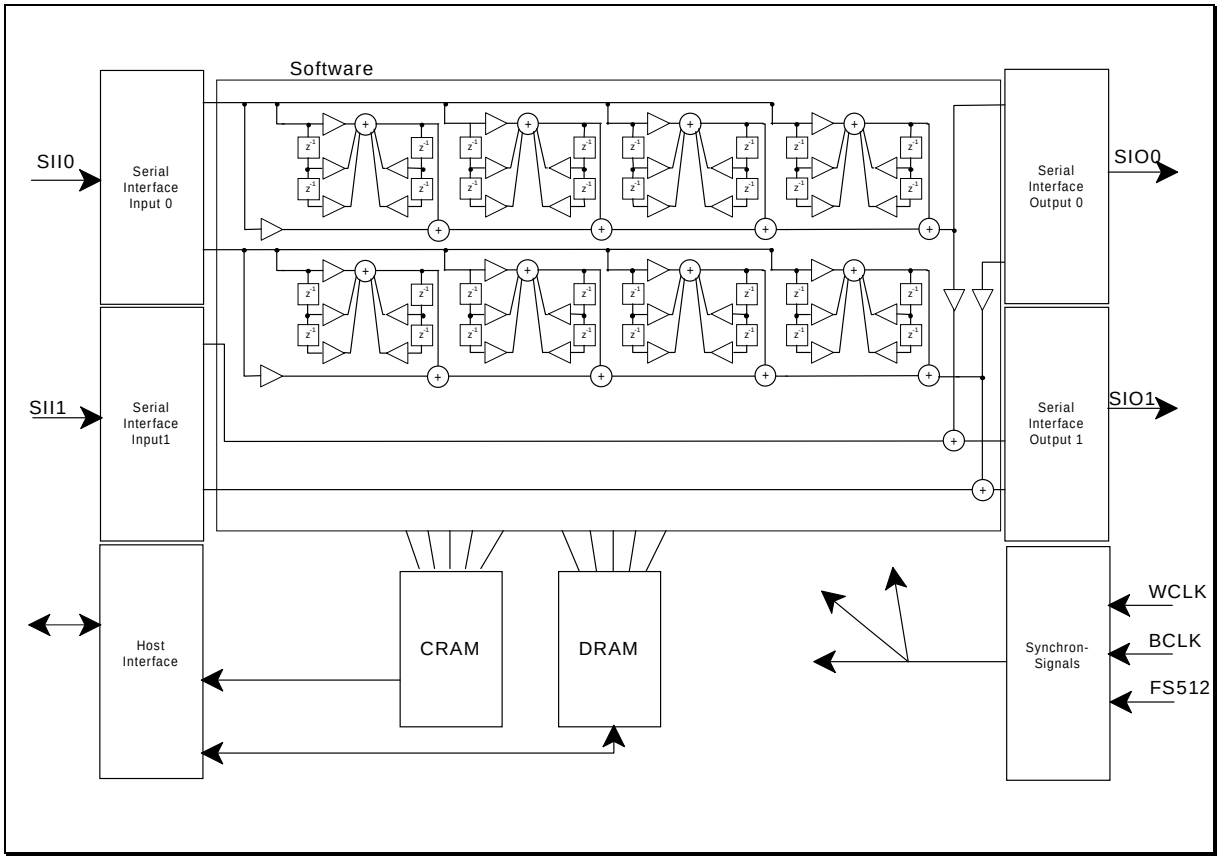


Abbildung 10 Interne Struktur des KanalDSP

Der KanalDSP ist ein speziell angefertigter DSP der Firma Quasimidi. Die Programme sind unveränderbar im Maskrom.des Chips. Es stehen mehrere Audioalgorithmen zu Verfügung. In diesem Projekt wird jedoch nur ein Algorithmus verwendet. Es handelt sich um acht sog. Biquat IIR-Filter. Da von einem DSP zwei Kanäle bearbeitet werden, sind pro Kanal vier Biquats verfügbar. Auf diese Art von Filter wird im Kapitel Software noch näher darauf eingegangen. Desweiteren beinhaltet der KanalDSP serielle Schnittstellen zur Audioübertragung, welche das gleiche Protokoll verwenden wie der AD/DA-Wandler und der FPGA, jedoch wird als Wortbreite im Stereobus 24 Bit verwendet. Wie in Abbildung 10

Interne Struktur des KanalDSP gezeigt gehen vier Audiokanäle in den DSP und vier Audiokanäle wieder hinaus. Hierzu werden jeweils zwei serielle Datenleitungen für Ein- und Ausgang gebraucht.
Der KanalDSP überträgt die Daten synchron zum Hostprozessor unter Verwendung der Signale: CS\, IFCD, IFCK, IFDI, IFDO and ACK\.

CS\ (input)	Chipselect	Low-Pegel auf diesem Signal aktiviert die Hostschnittstelle dieses DSPs.
RST\ (input)	Reset	Low-Pegel auf diesem Signal bringt den Kanal-DSP in einen definierten Anfangszustand.
IFDI (input)	Dateneingang	Auf dieser Leitung empfängt der KanalDSP Daten vom Hostprozessor.Sie werden durch die Clockleitung synchronisiert.
IFDO (output)	Datenausgang:	Auf dieser Leitung sendet der KanalDSP die angeforderten Daten zum Hostprozessor.Sie werden durch die Clockleitung synchrinosoert.
IFCK (input)	Clockleitung:	verbunden mit der Hostclock
IFCD (inout)	Commando/Data Select:	Diese Leitung signalisiert, ob es sich bei der Kommunikation zwischen Hostprozessor und KanalDSP um Daten oder um Commandos handelt.
ACK\ (output)	Acknowledge:	Bestätigungssignal zur Sicherung des Datenaustausches zwischen Hostprozessor und KanalDSP. Es wird das Ergebnis des Paritycheck ausgegeben.

Der Hostprozessor wird verwendet um Datenformate und Daten im Coeffizienten-RAM (CRAM) zu setzen und einzustellen. Ferner wird ein Modulsequenzer gesteuert, der den Programmablauf bestimmt. Der Hostprozessor kann auch Daten aus dem Speicher des DSPs lesen, was für das Metering benötigt wird.

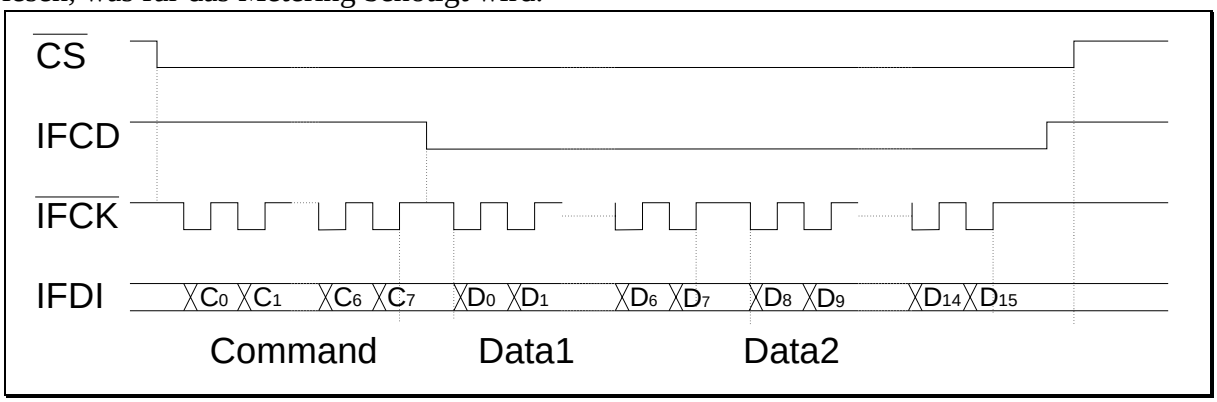


Abbildung 11 Timing der Hostschnittstelle des KanalDSPs

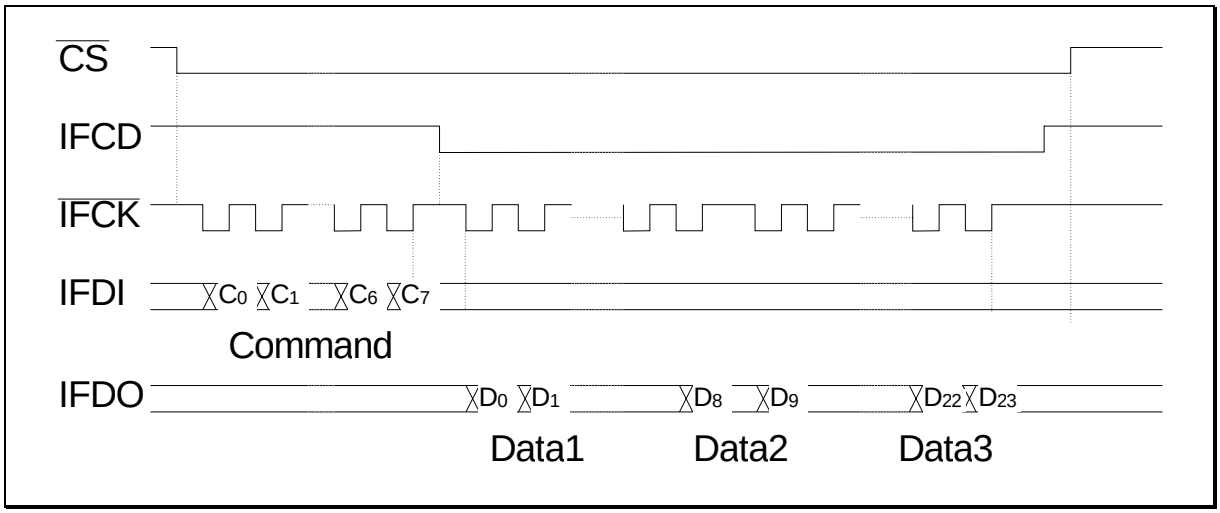


Abbildung 12 Timingbeispiel Datenübertragung von KanalDSP zur HostCPU

6.6 Mixsignalprozessor

6.6.1 Einführung

Der Mixsignalprozessor ist ein Fixed-Point DSP neuester Generation. Da durch ihn alle Audiosignale laufen, braucht dieser Chip eine extrem hohe Rechenleistung bei gleichzeitigem Datentransfer. Dieser Chip schafft es innerhalb von 160ns ein Datenwort aus einem externen Speicher in seine Register zu holen. Eine Multiplikation mit gleichzeitiger Addition (MAC-Befehl) und noch ein Datentransfer zwischen Registern benötigt 40ns. Sämtliche Operationen müssen in einem Zeitfenster von $1/44100\,s \approx 22676ns$ abgearbeitet sein.

6.6.2 Funktionsbeschreibung des MixDSPs

Der Mix-DSP liest über seinen 16 Bit breiten Datenbus die Signale der 8 Eingänge vom FPGA ein. Die 4 Stereokanäle (4·2 Kanäle) werden jeweils über die Adressleitungen angesprochen:

Kanal	Leseadresse
1 und 2	$00\ 0000\ 0000_2$
3 und 4	$00\ 0000\ 0001_2$
5 und 6	$00\ 0000\ 0010_2$
7 und 8	$00\ 0000\ 0011_2$

Diese 8 Eingänge sind über eine 8×8 Matrix mit 8 Ausgängen verbunden Die 8 Ausgänge werden über den Datenbus an den FPGA weitergegeben. Dieser Schreibvorgang geschieht auch über Adressen, die sich jedoch von den Leseadressen im MSB (**m**ost **s**ignificant **b**it) unterscheiden:

Kanal	Schreibadresse
1 und 2	$10\ 0000\ 0000_2$

3 und 4	$10\ 0000\ 0001_2$
5 und 6	$10\ 0000\ 0010_2$
7 und 8	$10\ 0000\ 0011_2$

Desweiteren wird über die eingangseitige serielle Schnittstelle der Stereobus der KanalDSPs eingelesen. Der Pegel dieses Busses wird durch zwei (rechts und links) Koeffizienten bestimmt. Nach der Bearbeitung wird diese Stereosumme an die ausgangsseitige serielle Schnittstelle zum optischen S/P-DIF-Wandler weitergegeben.

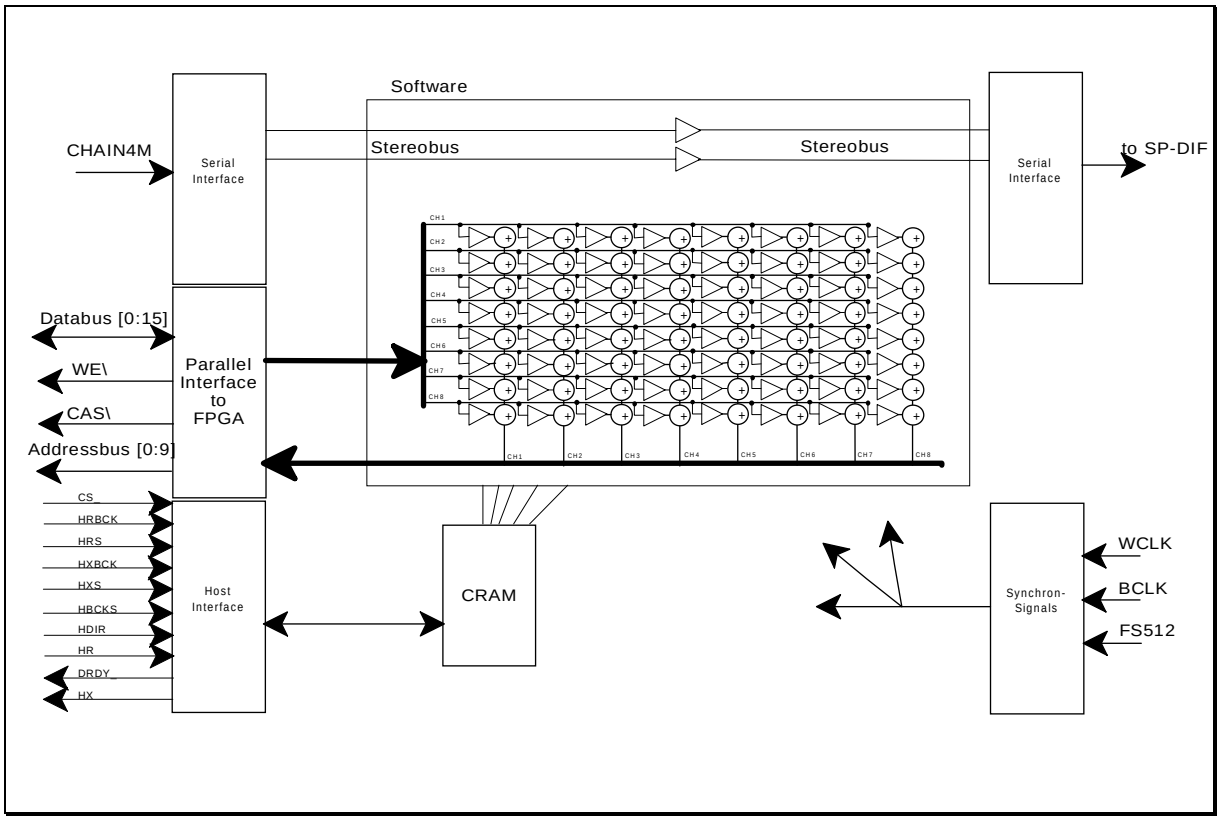


Abbildung 13 Interne Struktur des MixDSP

Das Datenformat der seriellen Audioschnittstelle entspricht dem Format des AD/DA-Wandlers, jedoch mit 24 Bit Wortbreite. Der serielle Audioausgang ist auf 16 Bit beschränkt, da es dem Standard Sony/Philips Format im Consumerbereich entspricht.

6.6.3 Die Hostschnittstelle des MixDSPs

Die Hostschnittstelle des Mix-Signalprozessor besteht aus 12 Signalleitungen:

RS\	Reset	Beim Start des Mixers wird der MixDSP durch einen Low-Impuls in einen definierten Zustand gebracht.
CS_	Chipselect Eingang	Dieses Signal aktiviert die Hostschnittstelle des DSPs.
HR	Hostdata receiver	Serielle Daten kommen über diese Leitung in das Empfangsschieberegister. Das serielle Format wird durch das HDIR-Signal bestimmt.
HRBCK	Receive clock input	Synchronisiert die Daten vom HR-Pin. Die aktive Flanke wird durch das HBCKS-Signal bestimmt.
HRS	Frame-Synchronisationssignal	Ein High-Impuls synchronisiert ein Frame (1 Frame = 8 bits) der HR-Daten.
HX	Hostdata transmitter	Auf dieser Leitung sendet der MixDSP die angeforderten

		Daten zum Hostprozessor
HXBCK	Transmit clock input	Synchronisiert die Daten vom HX-Pin. Die aktive Flanke wird durch das HBCKS-Signal bestimmt.
HDIR	Daten Format	Wenn dieses Signal auf Low-Pegel liegt gilt bei HR und HX : LSB first. (Dieses Signal ist beim Mixer fest auf Low gelegt)
HXS	Frame-Synchronisationssignal	Ein High-Impuls synchronisiert ein Frame (1 Frame = 8 bits) der HX-Daten.
EMPTY_	Buffer-Empty-Flag	Dieses Signal ist auf High wären der Übertragung von Koeffizienten oder Programmdaten. An diesem Signal kann der Hostprozessor abprüfen, ob er neue Koeffizientendaten senden darf.
DRDY_	Data ready output	Dieses Signal ist auf Low wenn der DSP fertig zum senden ist oder sich ein Datum im Sendebuffer befindet.
HRBCKS		Dieses Signal selektiert die aktive Flanke der Hostclocksignale HXBCK und HRBCK.

6.7 FPGA (Field Programmable Gate Array)

6.7.1 Einführung

In diesem Projekt wird ein FPGA der Firma XILINX mit der Bezeichnung XC3142A. Diesen Baustein gibt es in verschiedenen Gehäuse mit einer unterschiedlichen Pinanzahl, wobei hier die Ausführung mit 84 Pins im PLCC-Gehäuse genommen wird.

6.7.2 Aufgabe des FPGAs

Die Hauptaufgabe des FPGAs ist die Anpassung der DRAM-Schnittstelle des Mixprozessors an die seriellen Schnitstellen der KanalDSPs. In Abbildung xx sieht man das Format der DRAM-Schnittstelle des Mixprozessors. Die Mixprozessor kann die Daten in vier Register des FPGA schreiben. Durch Zeitmultiplexing bzw. durch Aufteilung der Sample-Fensters in zwei Teile ist es möglich 8 Kanäle zu übertragen. Umgekehrt kann der Mixprozessor aus 4 Registern des FPGA lesen.

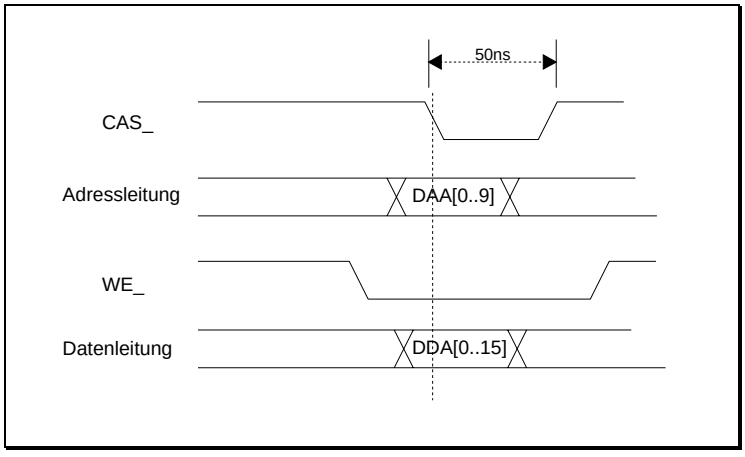


Abbildung 14 Schreibzugriff des MixDSPs

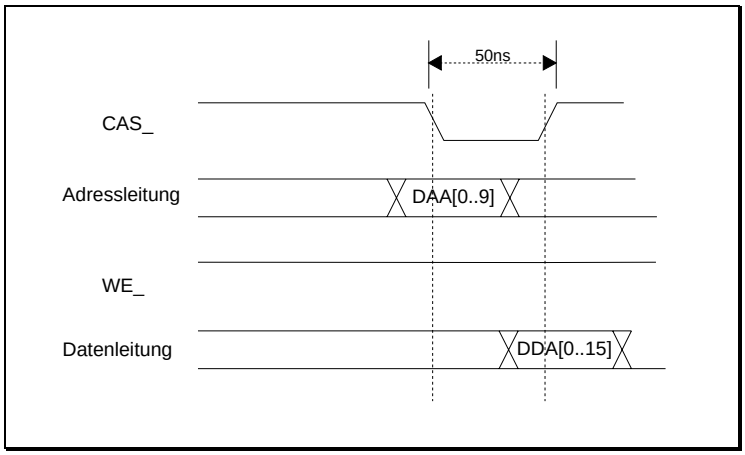


Abbildung 15 Lesezugriff des MixDSP

Eine weitere Aufgabe, die der FPGA zu erledigen hat ist die Erzeugung der Synchronisationssignale für den Mixer. Zu Verfügung steht das Signal FS512, welches von einem Quarzoszillator generiert wird.

6.7.3 Anschlüsse und Signale

FS512 Input	Mainclock Synchronisationseingang	Mit dieser Clock wird das ganze System synchronisiert. Durch Teilen werden sämtliche Clocksignale erzeugt: • $\frac{FS512}{2} = FS256$ • $\frac{FS512}{32} = BCLK$ • $\frac{FS512}{512} = WCLK$
FS256 Output	256fache Samplingfrequenz	Systemclock für AD-Wandler: Um eine 128fache Überabtastung (Oversampling) zu realisieren braucht der AD-DA Wandler diesen Takteingang.
WCLK Output	Wordclock	Eine fallende Flanke dieser Clock bewirkt ein Neustart der DSP-Algorithmen.
BCLK Output	Bitclock	Die Bitclock synchronisiert die Bitübertragung bei den seriellen Audioleitungen. steigende Flanke: Datenübernahme fallende Flanke: Datenwechsel
CAS_ Input	Column-Adress-Strobe	Synchronisiert die Kommunikation mit der DRAM-Schnittstelle des MixDSPs.
WE_	Write enable	Umschaltsignal zwischen Lese- und

Input		Schreibvorgang bei der Kommunikation des FPGAs mit der DRAM
AD_4 Input	Analog-Digital Wandler 4 Eingang	Serieller Dateneingang synchronisiert mit der Bitclock, zum Empfang der Audiodaten vom 4. AD-Wandler. Hinter diesem Pin verbirgt sich ein 16 Bit tiefes Schieberegister.
DA_4 Output	Digital-Analog Wandler 4 Eingang	Serieller Datenausgang synchronisiert mit der Bitclock, zum Senden der Audiodaten zum 4. AD-Wandler. Ein Schieberegister getaktet von der Bitclock schiebt die parallel vorliegenden Daten aus dem FPGA.
AD_3 Input	Analog-Digital Wandler 3 Eingang	s.o.
DA_3 Output	Digital-Analog Wandler 3 Eingang	s.o.
AD_2 Input	Analog-Digital Wandler 2 Eingang	s.o.
DA_2 Output	Digital-Analog Wandler 2 Eingang	s.o.
AD_1 Input	Analog-Digital Wandler 1 Eingang	s.o.
DA_1 Output	Digital-Analog Wandler 1 Eingang	s.o.
DDA[0:15] Input/Output	Digital Audio Databus (16 bit)	Mit dem CAS_-Signal synchronisierter Datenweg zwischen FPGA und MixDSP.
DAA[0:9] Input	Digital Audio Adressbus (10 bit)	Die Adressleitung werden von einer Adressdekoderlogik dekodiert. Die Adresse bestimmt welches der 8 Schieberegister im FPGA angesprochen wird.

6.8 S/P-DIF Ausgangsinterface

6.8.1 Einführung in das Sony/Philips Digital Interface Format

Die Firmen Sony und Phillips vereinbarten das *S*ony/*P*hilips-*D*igital - *I*nter*F*ace um eine digitale Audioschnittstelle dem Consumerbereich zur Verfügung zu stellen. Diese Schnittstelle gibt es in zwei verschiedenen Ausführungen: koaxial und optisch. Die zwei Formate unterscheiden sich nicht in der logischen Zusammensetzung, er werden jedoch unterschiedliche Übertragungsmedien verwendet. In der koaxialen Ausführung werden koaxiale Kabel verwendet, wogegen in der optische Ausführung die Daten über Lichtwellenleiter übertragen werden. In diesem Projekt wird die optische Variante verwendet. Die Kanalcodierung erfolgt im Bi-Phase-Mark-Code, wo zu Beginn und am Ende jedes Bits ein Flankenwechsel stattfindet, bei einer „1“ zusätzlich ein Flankenwechsel in der Bitmitte. Der Bus ist selbsttaktend und überträgt die Daten in 32-Bit grossen Sub-Frames. Zwei Sub-Frames bilden ein Frame, 192 Frames ein Datenblock. Die beiden Subframes eines Frames

enthalten jeweils das Sample des linken und rechten Stereokanals. Jedes Subframe ist 32 Bits gross. Bei einem Stereosignal mit einer Taktfrequenz von 44,1 kHz entsteht eine Datenfluss von rund 2,8 Mbit/sec ($2 \cdot 32 \cdot 44.100 = 2.822.400$).

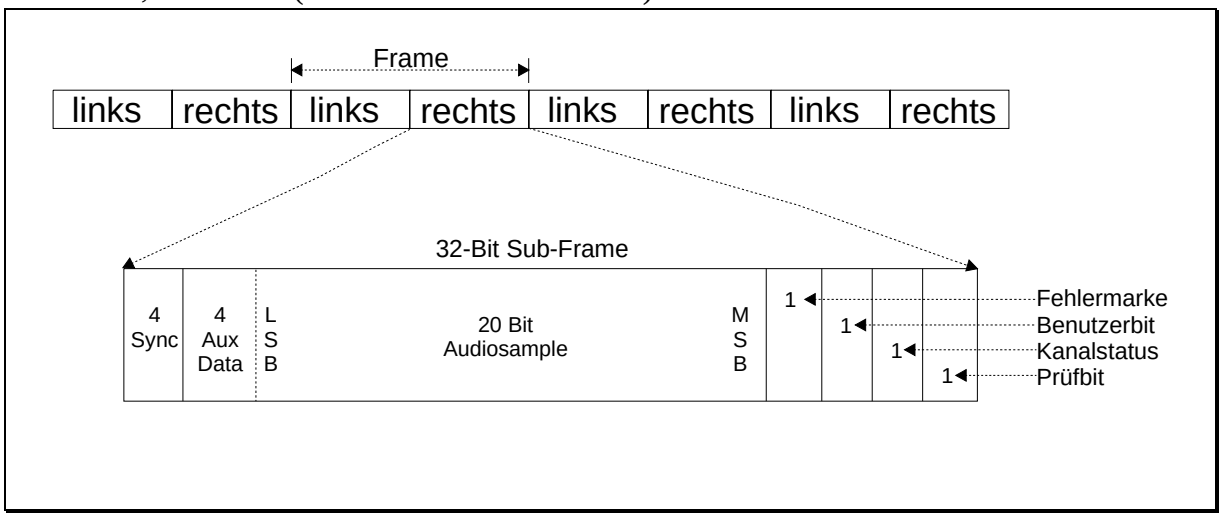


Abbildung 16 Datenformat eines S/PDIF-Sub-Frames

Zur Synchronisation werden die ersten 4 Bits im Sub-Frame verwendet. Danach folgen vie Zusatz-Bits, welche zur freien Verfügung stehen. Zusammen mit den 20 Bits für Audiodaten könnte man eine Auflösung von 24 Bits übertragen. Am Ende des Subframes folgen vier Statusbits. Über diese Statusbits werden standardisierte Daten übertragen, wie z.B. die momentan verwendete Abtastrate, oder den gerade übertragenden Titel. Das Ausgangsinterface beim Mixer besteht aus einem IC TC9231N und einem optischen Connenctor. Der IC wandelt das Audiodatenformat des Mixers in das S/P-DIF Format um.

6.8.2 Schnittstellen des S/P-DIF Wandlers

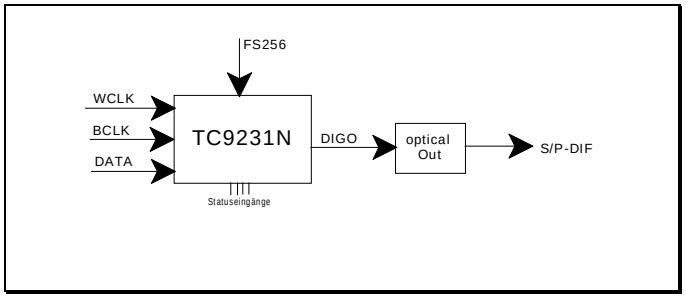


Abbildung 17 Anschlüsse des S/P-DIF Bausteins

Der TC9231N benötigt lediglich vier Signale um das S/P-DIF Protokoll zu erzeugen: FS256, WCLK, BCLK und die Audiodaten. Die Statuseingänge legen die Daten fest die über die Statusbits im Kanalprotokoll übertragen werden.

6.9 Erweiterungssteckplatz

6.9.1 Überlegungen der Erweiterungsfähigkeit

Auf dem Erweiterungssteckplatz sind der Audioadressbus, der Audiodatenbus und verschiedene Hostprozessorsignale vorhanden. Der Mixer ist durch diesen Erweiterungssteckplatz auf bis zu 2048 Audiokanäle erweiterbar. Die Zahl der Kanäle errechnet sich wie folgt:

$$2^{10(Adressleitungen)} \cdot 2(Kanäle / Adressleitung) = 2048.$$

Jedoch ist dies die logische Begrenzung der Kanalzahl. Da über den Audiodatenbus sämtliche Audiokanäle gehen, ist liegt das sinnvolle Maximum schon bei wesentlich weniger Kanälen. Im Rahmen der Diplomarbeit werden insgesamt nur 16 Kanäle verarbeitet. Später soll es jedoch möglich sein 24 bis 256 Kanäle zu verarbeiten. Jedoch ist der MixDSP nicht in der Lage $128 \times 128 = 16834$ Berechnungen pro Samplingperiode durchzuführen. Dies ist jedoch im normalen Gebrauch eines Mischpultes auch nicht nötig. Auch in größeren Studios werden zum selben Zeitpunkt selten mehr als 40 Eingangskanäle bei 24 Ausgangskanälen benutzt. Wenn jeder Eingangskanal im Durchschnitt auf zwei Ausgangskanälen geroutet wird, sind $40 \cdot 2 = 80$ Multiplikationen und $40 + 24 = 64$ Datentransfers nötig um diesen relativ komplexen Anwendungsfall zu decken. Dies ist mit dem MixDSP noch realisierbar. Schwierig ist jedoch die dynamische Anpassung des MixDSP-Algorithmus an das gewünschte Routing. Tonaussetzer bei einer Routingänderung wären fatal. Aus diesem Grund muß bei laufendem DSP der Algorithmus geändert werden, ohne das die bisherigen, Berechnungen im Ablauf gestört werden.

6.9.2 Funktionsabschnitte des Erweiterungssteckplatzes

Der Erweiterungsteckplatz besteht aus folgenden Funktionsabschnitten:

Audioschnittstelle	Die Audioschnittstelle ist die gleiche Schnittstelle wie die Schnittstelle zwischen MixDSP und FPGA. Es sind physikalisch die gleichen Leitungen: • DAD[0..15] • DAA[0:9] • WE\ • CAS\ • CHAIN_C1
Synchronisationssignale	Um die Erweiterungen mit dem Hauptboard zu synchronisieren, werden folgende Signal benötigt: • BITCLK • WORDCLK • FS256
Hostschnittstelle	Die Host schnittstelle besteht aus einem Hostadressbus und Hostsignalleitungen. Der Adressbus ist 5 Bits breit, so daß $2^5 = 32$ Erweiterungen adressiert werden können, denen jeweils folgende Signale zu Verfügung stehen: XFPGA_CS : FPGA-Chipselect XGAINCS : Gaincontroller-Chipsselect XDSPRS\ : KanalDSP-Reset XDSPCS\ : KanalDSP-Chipselect IFDO : Datenausgang KanalDSP ACK : Datensicherungssignal KanalDSP HOSTDATA : allgemeine Datenleitung HOSTCLK : Synchronisation von Hostdata SSEL : Synchronisationssignal RESET\ : lokale Resetleitung

7 Software

7.1 Steuerungs-Software (PC)

7.1.1 Gesamtüberblick

Um die vielfältigen Parameter des Mixers benutzerfreundlich steuern und testen zu können, wurde eine grafische Benutzeroberfläche auf dem PC entwickelt. Sie stellt neben üblicher Elemente eines analogen Mischpultes auch weiterreichende Einstellungshilfen dar, wie z.B. der Frequenzverlauf der Filter.

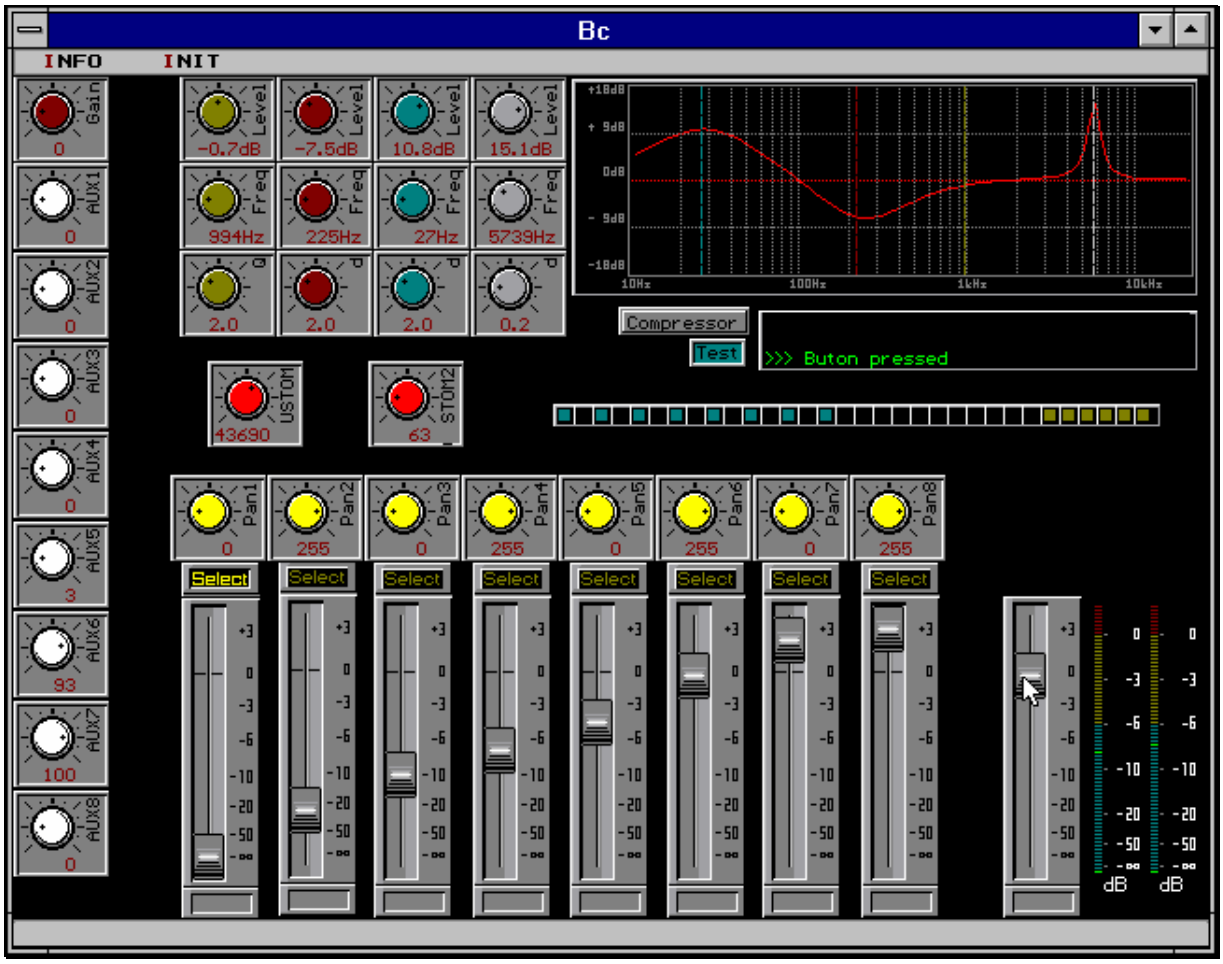
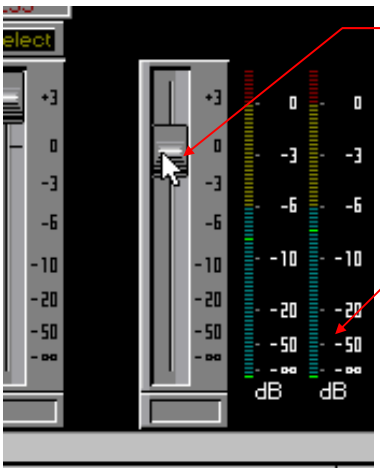


Abbildung 18 Screenshot der PC-Steuerungssoftware

Die Bedienung der Oberfläche ist eng angelehnt an die von Microsoft Windows. Die Parameter werden in Echtzeit über einen seriellen Port des PC´s an den Digitalmixer übertragen, d.h. jede Änderung wirkt sich sofort auf die Einstellungen in den DSPs aus.

7.1.2 Masterfader und Pegelanzeiger



Masterfader:
Der Masterfader wirkt sich direkt auf die Gesamtlautstärke des digitalen Ausgangs (S/P-DIF) aus.

Pegelanzeiger:
Der Pegelanzeiger (VU-Meter) zeigt den derzeitigen Pegel des digitalen Ausganges an. Der Spitzenwert wird zur besseren Ablesbarkeit ca. 1s lang konstant angezeigt.

Die Einheit des Masterfaders wird, wie in der Audiotechnik üblich in dB angegeben. Der maximale Dynamikumfang, den man mit 16 Bit Sampleauflösung darstellen kan, sind

$$20 \cdot \log(2^{16}) \approx 96{,}33dB.$$

Um bei der digitalen Verarbeitung Rechenzeit zu sparen werden die Werte gerundet und folgende Annäherung verwendet:

$2^{16} \rightarrow 96dB \quad Fehler: 0{,}3296dB$

$2^{15} \rightarrow 90dB \quad Fehler: 0{,}3089dB$

$2^{14} \rightarrow 84dB \quad Fehler: 0{,}2884dB$

M

$2^{1} \rightarrow 6dB \quad Fehler: 0{,}0206dB$

$2^{0} \rightarrow 0dB \quad Fehler: 0dB$

Der Multiplikationsbefehl des Masterfaders wird vom MixDSP ausgeführt. Die Quelle des Signales ist der Stereobus, welcher am seriellen Eingang des MixDSPs anliegt. Im MixDSP werden 24Bit breite Faktoren verwendet. Der 16Bit Samplewert muß dazu auf 24Bit ausgedehnt werden. Dazu werden die untersten 8Bit mit Nullen aufgefüllt. Das Ergebnis ist 48Bit breit, wobei die oberen 16Bit als Ergebnis verwendet werden. Um eine Verstärkung um 3dB zu erlauben, muß das Signal um 1Bit geshiftet werden. Dies führt jedoch zu einer Verstärkung von 6dB, was bei der Faktorenbestimmung berücksichtigt werden muß.
Beispiel:

Masterfader auf +3dB heißt, daß der Faktor eine Dämpfung von 3dB bewirken muß. Nach folgender Formel wird der Faktor x bei Dämpfung $a_{Dämpfung}$ berechnet:

$$x = 10^{\frac{(96dB - a_{Dämpfung}\,dB)}{20}}$$

Bei $a_{Dämpfung} = 3 \quad \Rightarrow \quad x = 10^{\frac{96-3}{20}} \approx 44668_{10} = AE7C_{16}$

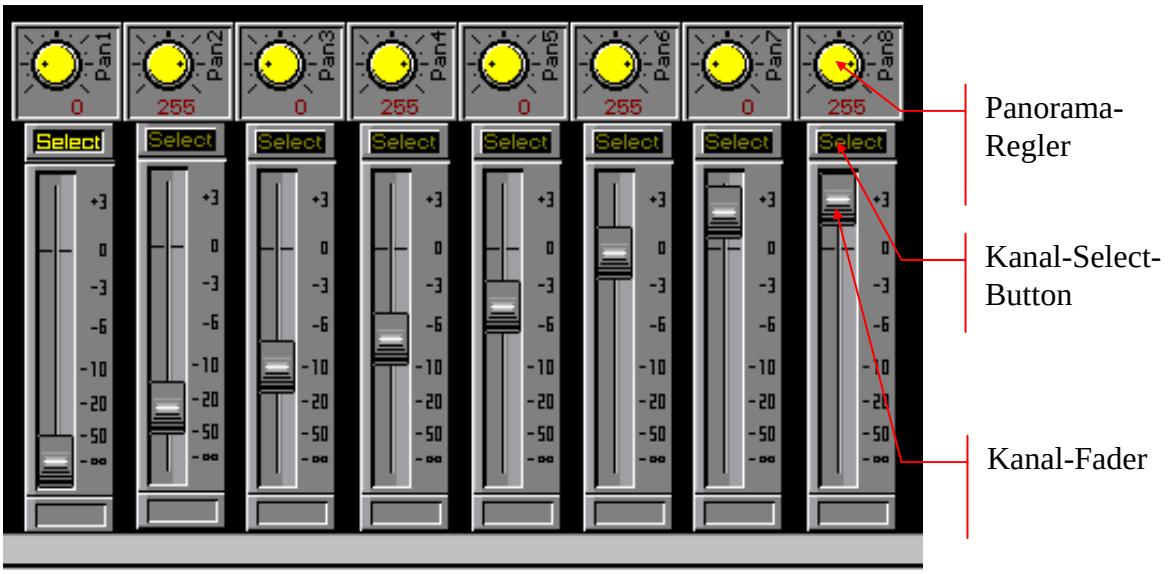
und 24Bit Faktorauflösung muß als Koeffizient $AE7C00_{16}$ übertragen werden.

Der Pegelanzeiger geht den umgekehrten Weg und verwendet folgende Formel:

$$x_{Pegel} = 20 \cdot \log(s_{Samplewert})$$

Würde nur diese Formel verwendet werden würde die Anzeige stark flackern und man würde kaum etwas erkennen. In sofern ist es nötig die Anzeige zu ´beruhigen´. Den Anwender interessiert wie hoch das Signal ausgesteuert ist. Es müssen die Maximalpegel angezeigt werden und das Abfallen des Pegels verlangsamt werden.

7.1.3 Kanallautstärken, Panorama und Select



Der Kanalfader ist zuständig für den Anteil des jeweiligen Kanals an der Stereosumme. Bei der Berechnung des Verstärkungsfaktors gelten die gleichen Regeln wie beim Masterfader. Es wird das Signal eines KanalDSPs auf die Stereosumme verteilt. Um die Verteilung auf die beiden Kanäle der Stereosumme regeln zu können, gibt es den Panoramaregler. Der Panoramaregler bestimmt die Position des Intrumentes bzw. des Kanals im Stereobild. Dafür wird eine lineare Funktion benutzt.Der Wertebereich des Panoramareglers ist von -100 (für links) bis +100 (für rechts). In der Mittelstellung ist der Wert des Panoramareglers 0 (die Signalanteile sind gleich):

Gesucht sind die Faktoren zur Verteilung des Signals: x_{links} und x_{rechts}.

Gegeben ist die Panoramastellung p (-100..+100) und

der Wert des Kanalfaders l (0..65535).

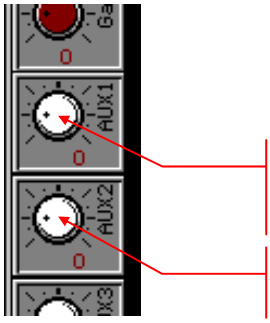
$$x_{links} = \frac{200 - (p + 100)}{200} \cdot l$$

$$x_{rechts} = \frac{(p + 100)}{200} \cdot l$$

Der Kanal-Select Button aktiviert den entsprechenden Kanal, um ihn mit dem Filter und den AUX-Reglern bedienen zu können. Es kann maximal nur ein Kanal aktiviert werden. Die Änderungen am Filter, am Gain und an den AUX-Reglern wirkt sich nur auf den aktivierten Kanal aus. Die Faktorenberechnung x_{AUX} aus der Reglerstellung p erfolgt linear:

$$x_{AUX} = p * 327.67$$

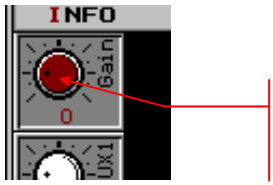
7.1.4 Aux-Wege



Die 8 Regler haben jeweils einen Wertebereich von p=0..100. Bei dem Wert p=100 wird das Signal ungedämpft auf den entsprechenden Ausgang ausgegeben. Dementsprechend wird bei der Einstellung p=0 das Signal nicht auf den Ausgang ausgegeben. Die Faktorenberechnung x_{AUX} aus der Reglerstellung p erfolgt linear:

$$x_{AUX} = p \cdot 327.67$$

7.1.5 Gain-Regler

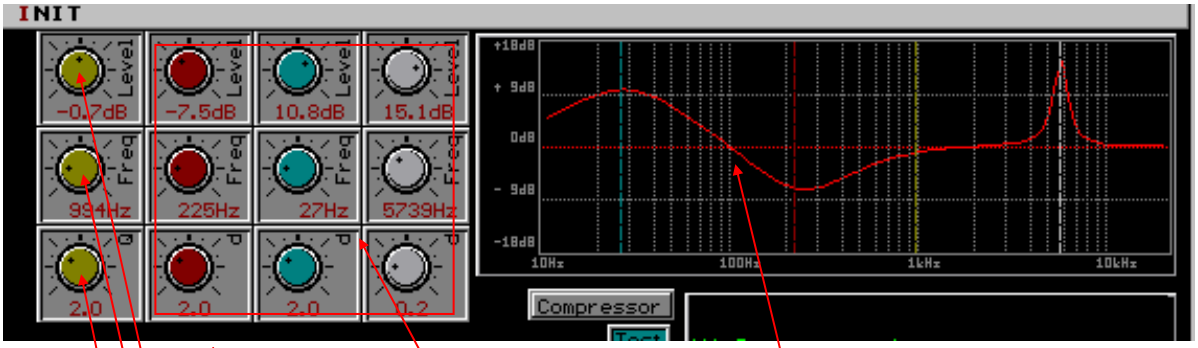


Der Gain-Regler steuert die digitalen Potis LM1972. Je nach Einstellung verstärkt der Gain-Controller das analoge Eingangssignal um bis zu 78dB. Die Schrittverteilung der Potis ist nicht über den gesamten Einstellbereich konstant:

Schritte	LM1972
0,5 dB	0 dB-47,5 dB
1 dB	48 dB-78 dB

An den Digimixer werden jedoch Werte von 0-126 übertragen, die direkt an den Gain-Controller weitergegeben werden.

7.1.6 Filtersektion



Pegelregler des 1.EQ.

Einstellung der Mittenfrequenz des 1.EQ.

Einstellung der Filtersteilheit des 1.EQ

Entsprechend den Reglern des 1.EQ.

Summe der Frequenzverläufe der 4 Filter. Die farbigen senkrechten Lienien stellen die Mittenfrequenzen der einzelnen EQs dar.

Die Filtersektion steuert die Filter in den KanalDSPs. Das PC-Steuerprogramm übergibt die eingestellten Werte direkt an den Digimixer. Erst der Digimixer selbst berechnet die Filterkoeffizienten aus den übertragenen Werten. Es stehen insgesamt 4 Filter (EQs) zur Verfügung. Jeder Filter ist durchstimmbar von 16Hz bis 20kHz(Mittenfrequenz). Weiter kann man bei jedem Filter die Flankensteilheit oder den sog. Q-Faktor von 0.1 (sehr steilflankig) bis 10 einstellen. Als dritter Parameter ist der Pegel der Filter einstellbar. Hiermit kann man bestimmen um wieviel ein Frequenzband gedämpft oder verstärkt werden kann.
Das PC-Steuerprogramm stellt den Frequenzverlauf der 4 Filter dar. Dazu wird aus dem Frequenz und Phasenverlauf aller 4 EQs die vektorielle Summe gebildet und den Frequenzverlauf dargestellt.

Gegeben sind die Mittenfrequenz θ_0 und den Q-Faktor Q.

Es wird eine Hilfsvariable d eingeführt, für die gilt $d = \frac{1}{Q}$.

Für einen Bandpass IIR-Filter 2.Ordnung gilt:

GAIN:

$$G(\theta) = \frac{d \cdot \sin\theta_0 \cdot \sin\theta}{\left[\left(d \cdot \sin\theta_0 \cdot \sin\theta\right)^2 + 4\left(\cos\theta - \cos\theta_0\right)^2\right]^{1/2}}$$

PHASE:

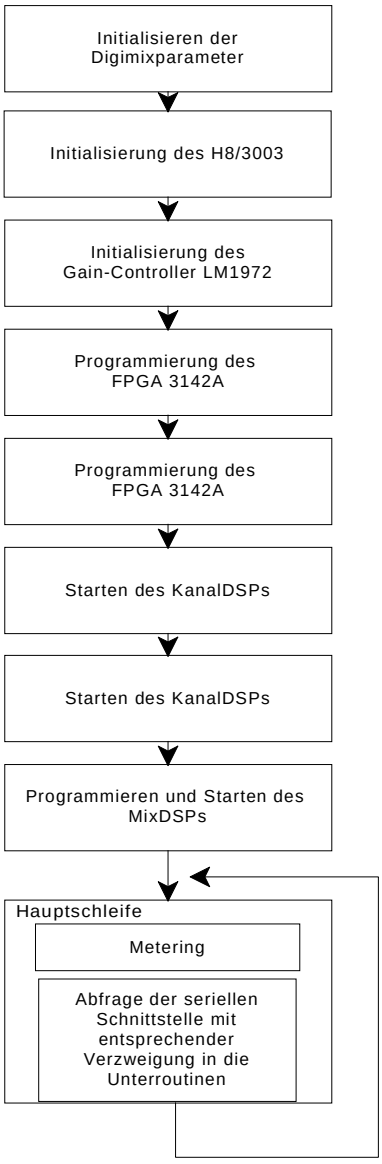
$$\Phi(\theta) = -\tan^{-1}\left[\frac{2\left(\cos\theta - \cos\theta_0\right)}{d \cdot \sin\theta \cdot \sin\theta_0}\right]$$

7.2 Mikrocontroller H8/3003

7.2.1 Entwicklungstools

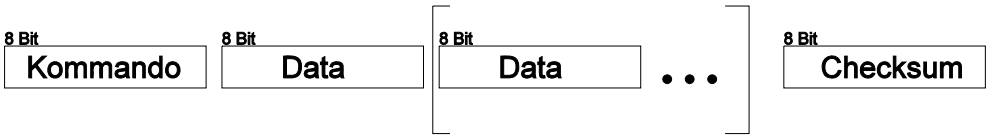
Zur Programmierung des H8/3003 steht ein IN-CUIRCUIT-EMULATOR bereit. Er ermöglicht Debugging im konkreten System. Zu jeder Zeit ist es möglich die Register der CPU einzusehen oder in den Programmablauf einzugreifen. Die Programmierung erfolgt ausschließlich in C, da in diesem Projekt keine zeitkritischen Algorithmen oder Programmabläufe nötig sind. Compiler und Linker sind als DOS-Kommandozeilen-Programme ausgeführt und wurden an die BorlandC 3.1 Entwicklungsumgebung angepaßt, um eine einheitliche Entwicklungsumgebung für alle drei (Hostprozessor, PC und MixDSP) Systeme zu erhalten. Der Sourcecode des Betriebsystems wurde auf zwei Module verteilt, ein Modul mit Low-Level Routinen und Interruptroutinen und ein zweites Modul mit High-Level Routinen.

7.2.2 Funktionsblockdiagramm



7.2.3 Kommunikation mit PC

Die Kommunikation zwischen Mixer und PC geschieht interruptgesteuert über RS232-Schnittstelle mit 9600Baud und folgendem Protokoll:



Je nach Kommando gibt es eine festgelegte Anzahl von Datenbytes. Das Packet ist dann gültig, wenn die Summe über alle übertragenen Bytes ein vielfaches von 256, ist d.h. die letzen 8 Bit sind 0. Tritt ein Übertragungsfehler ein, wird eine Message auf dem Bildschirm ausgegeben. Folgende Kommandos sind implementiert:

PC ⇒ Mixer Kommando	**Nr**	**Beschreibung**	**# Bytes**
P_CHANNELSELECT	1	Selektierung der Kanäle	1
P_VOLUME	2	Einstellung der Kanalpegel	1
P_GAIN	3	Einstellung der Eingangsempfindlichkeit	1
P_Eqlevel1	4	Equalizer 1 Level	1
P_Q_Faktor1	5	Equalizer 1 Q-Faktor	1
P_Frequenz1	6	Equalizer 1 Frequenz	2
P_Eqlevel2	7	Equalizer 2 Level	1
P_Q_Faktor2	8	Equalizer 2 Q-Faktor	1
P_Frequenz2	9	Equalizer 2 Frequenz	2
P_Eqlevel3	10	Equalizer 3 Level	1
P_Q_Faktor3	11	Equalizer 3 Q-Faktor	1
P_Frequenz3	12	Equalizer 3 Frequenz	2
P_Eqlevel4	13	Equalizer 4 Level	1
P_Q_Faktor4	14	Equalizer 4 Q-Faktor	1
P_Frequenz4	15	Equalizer 4 Frequenz	2
P_PANORAMA	16	Panorama des selektierten Kanals	1
P_AUX1	23	Pegelanteil auf Ausgang 1	1
P_AUX2	24	Pegelanteil auf Ausgang 2	1
P_AUX3	25	Pegelanteil auf Ausgang 3	1
P_AUX4	26	Pegelanteil auf Ausgang 4	1
P_AUX5	27	Pegelanteil auf Ausgang 5	1
P_AUX6	28	Pegelanteil auf Ausgang 6	1
P_AUX7	29	Pegelanteil auf Ausgang 7	1
P_AUX8	30	Pegelanteil auf Ausgang 8	1
P_MASTER	31	Einstellung des Pegels am SP-DIF Out	1

Mixer ⇒ PC Kommando	**Nr**	**Beschreibung**	**# Bytes**
TEXT_MESSAGE	1	Gibt eine Nachricht auf dem Bildschirm aus	Textlänge+1
VU_DATA	2	Sendet Metering-Werte	2

Beispiel einer Nachricht vom PC zum Mixer:

Die Frequenz des 2. Equalizers von Kanal 5 soll 4096Hz betragen.

- Setzen des aktiven Kanals. Bytefolge ⇒ 0x01; 0x05; 0xFA
- Setzen der Frequenz 4096Hz (0x1000). Bytefolge ⇒ 0x09; 0x10; 0x00; 0xE7

7.2.4 I/O-Ports

Der Befehlsatz des H8/3003 wurde durch einen Variablentyp „bit“ erweitert. Das wird durch den Compilerswitch „#pragma language=extendet“ erreicht. In der Variablendeklaration wird einer Variable ein Bit eines Registers oder eines I/O-Portes zugeordnet:

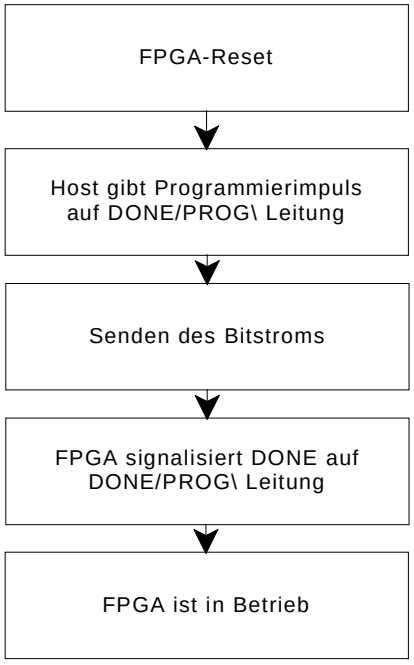
Beispiel: bit RxReady = scio_ssr.6;

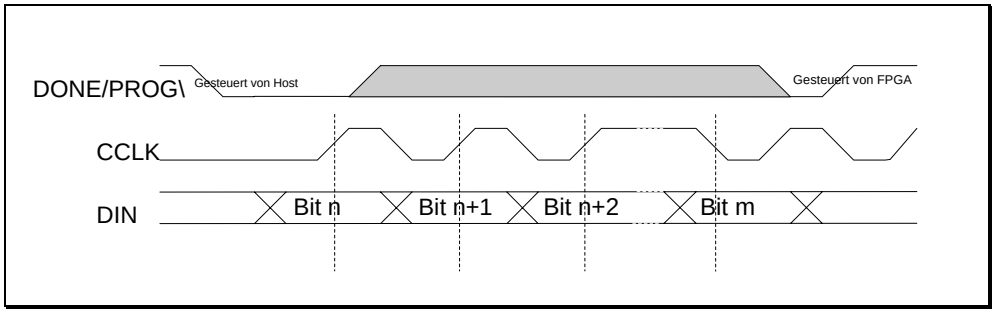
Mit der Variablen RxReady hat man direkten Zugriff auf das 6.Bit des SCIO_SSR-Register (Serial Communication Interface Serial Status Register).

Man kann jedem I/O-Port eine Variable zuordnen. Dadurch ist eine sehr übersichtliche Programmierung möglich. In der Headerdatei „PORT.H“ werden den I/O-Pins Variablennamen zugewiesen.

7.2.5 Programmierung des FPGA

Für die Programmierung des FPGAs werden die Leitungen FPGARST\, DONE-PG\, BOOTCLK, und HOSTDATA verwendet. Der Ablauf der Programmierung ist in folgendem Diagramm ersichtlich:





Die Programmdaten sind als konstantes Array im Eprom gespeichert. Die Programmierung wird durch einen Pulldown an DONE/PRG durch den Host gestartet. Nach diesem Impuls

werden ca. 30 kBit Daten in die internen Register des FPGAs geschoben. Der FPGA quittiert die Übertragung mit einem Pullup an der DONE/PROG Leitung. Danach ist der FPGA sofort in Betrieb.

7.2.6 Programmierung des MixDSP

Die Programmierung erfolgt über die im Hardware-Kapitel beschriebene Host-Schnittstelle. Es stehen 4 Programiermodi zur Verfügung:

- CMEM download
- PMEM download
- CMEM update
- PMEM update

Um ein komplettes Programm zu transferieren benutzt man den PMEM download Modus. Der DSP hat 5 Register (HR4-HR0) um insgesamt 40bit zwischen zu speichern. Nach aktiviertem Chipsselect werden die Programmdaten seriell auf der HOSTDATA Leitung ausgegeben. Die Bitsynchronisation erfolgt durch die HOSTCLK, die Framesynchronisation durch SSEL. Die Reihenfolge der Daten ist aus folgender Abbildung ersichtlich:

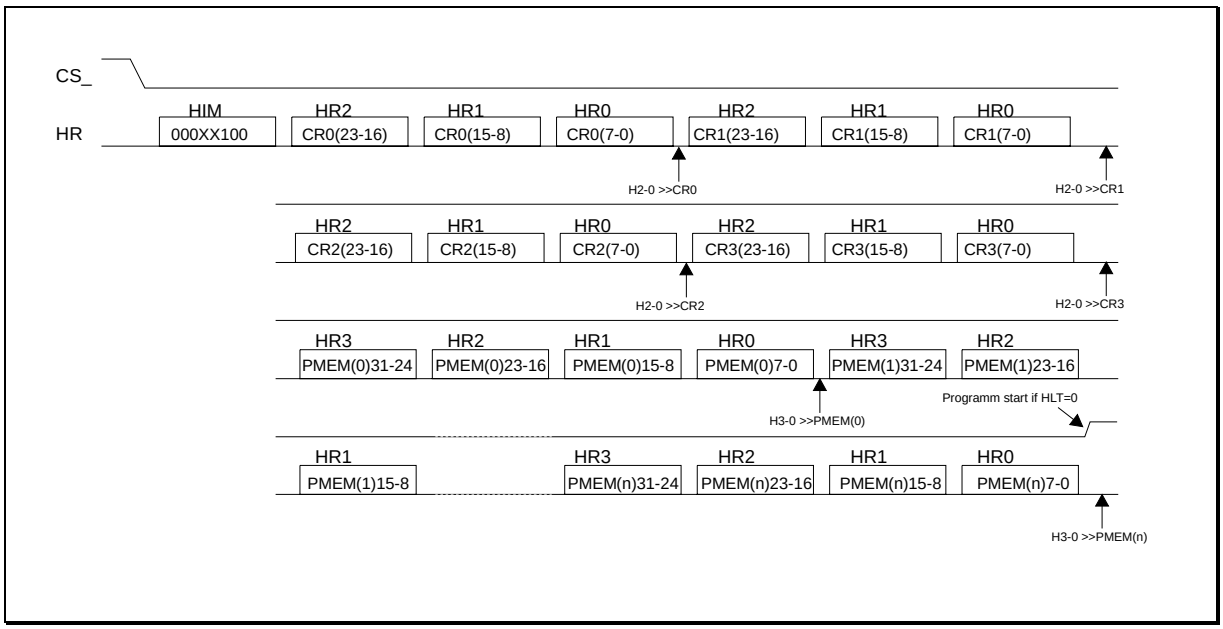


Abbildung 19 PMEM Download Protokoll

Der entprechende Programmiermodus wird durch das erste Byte, das HIM Register festgelegt. Die Bits in diesem Register haben folgende Belegung:

Bit	Name	Funktion
0	DST	Destination memory select: 0→PMEM ; 1→CMEM
1	UD	Update enable select: 1→update
2	DL	Download enable select: 1→download
3	HLT	Halt mode select: 0→DSP runs normally; 1→DSP stops
4	HINT	Host controller interupt request.
5		reserved
6		reserved
7		reserved

Nach gesendetem HIM Register werden zunächst 4 Controlregister gesendet. Sie dienen der Einstellung von Audiodatenformat, verwendetes RAM und diverse Einstelungen in der ALU und in der Multipliziereinheit. Danach wird das Programm selbst gesendet. Bei gelöschtem HLT-Bit und nach erfolgreicher Datenübertragung startet der DSP. Nun müssen die Koeffizienten initialisiert werden. Dies geschieht im CMEM Update Mode. Näheres dazu wird im Kapitel Steuern des MixDSPs erläutert.

7.2.7 Booten des KanalDSPs

Der KanalDSP beinhaltet Register, die beim Booten initialisiert und mit bestimmten Werten beschrieben werden müssen. Das wichtigste Register ist der Modulsequenzer (MSEQ). Er enthält bis zu 10 Adressen, die beim nächsten Reset nacheinander angesprungen werden. Mit Hilfe dieses Modulsequenzer wird beim Booten eine RAM-Clear Routine aufgerufen, bevor der Modulsequenzer auf das endgültige Programm zeigt. Desweiteren werden verschiedene Register zu Einstellung des Datenformates am seriellen Audiobus verwendet. Folgendes Diagramm beschreibt den Ablauf des Bootens:

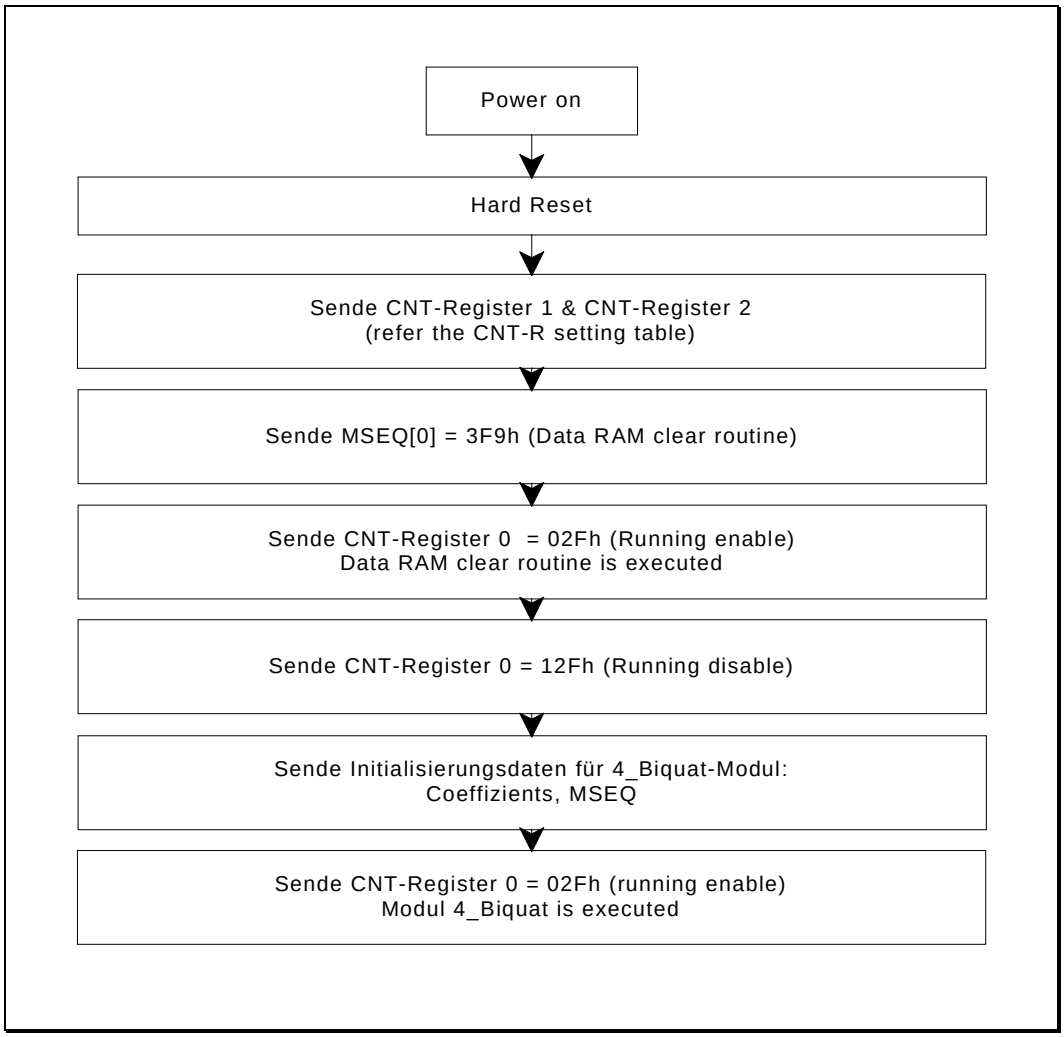


Abbildung 20 Bootablauf des KanalDSP

7.2.8 Steuern der Gain-Controller

7.2.9 Steuern der KanalDSPs

7.2.10 Steuern des MixDSPs

7.2.11 Erweiterungssteckplatz

7.3 FPGA- XC3142A-3

7.3.1 Programmierinterface

7.3.2 Erzeugung der Synchronisationssignale

7.3.3 Adressdekodierung

7.3.4 Parallel-Seriell Wandlung

7.3.5 Seriell-Parallel Wandlung

7.4 Misch-Signalprozessor

(kann man evtl gebrauchen)
Da jede Berechnung in einer 44100stel Sekunde durchgeführt sein muß, kann der DSP ca. : 113 Ein- od. Ausgabekanäle gleichzeitig verwalten:

$$\frac{1s}{44100 \cdot (160ns + 40ns)} \approx 113$$

Diese Rechnung ist jedoch noch nicht vollständig, da jedem Ausgangskanal mehrere Eingangskanäle zugeordnet werden können. Für jeden Eingangskanal braucht man einen MAC-Befehl. Will man das gesamte Routing des Mixers offenhalten, müßte man eine komplette Matrix mit beispielsweise $113 \cdot 113 = 12769$ Knotenpunkten implementieren. Für jeden Knotenpunkt würde ein MAC-Befehl benötigt, dies ist jedoch z.Zt. mit einem DSP in Echtzeit noch nicht machbar. Der hier verwendete DSP könnte maximal ca. 500 MAC-Befehle in einem Samplezyklus berechnen. Nun gilt es zu überlegen, wie man mit diesen Leistungsresourcen größtmögliche Flexibilität erhalten könnte

7.4.1 Einleitung

7.4.2 Funktionsdiagramm

7.4.3 Interruptbehandlung

7.4.4 Aux-Routing

7.4.5 Stereobuss-Berechnung

8 Resumeé

8.1 Probleme

8.1.1 FPGA→Entwicklung einer „Black-Box“

8.1.2 Timingschwierigkeiten

8.1.3 Ein Netz von Prozessoren

8.2 Erfahrungen

8.2.1 Es dauert immer länger als man denkt!

8.2.2 Motivation

8.3 Fazit

Jo - war net schlecht!

9 Anhang

9.1 Signalbeschreibung

WCLK	Wordclock (44100kHz)	Die Frequenz der Wordclock ist gleich der Abtastrate. Bei steigenden Flanke beginnt die Übertragung des linken Kanals. Bei fallender Flanke beginnt die Übertragung des Rechten Kanals. Innerhalb einer Wordclock müssen sämtliche Berechnungen und Übertragungen von den nächsten auszugebenden Samples abgeschlossen sein.
BCLK	Bitclock (FS32 ;1,4112 Mhz)	Die Bitclock syncronisiert die Bitübertragung bei den seriellen Audioleitungen. steigende Flanke: Datenübernahme fallende Flanke: Datenwechsel
FS512 MAIN_CLK	Mainclock (22,5792 Mhz)	Taktet direkt den Mixprozessor und den FPGA
FS256	11,2896 MHz	Wird als Systemclock für die AD/DA-Wandler benötigt
DSP1_IN	serielles Audiosignal:	Eingang des 1. KanalDSPs ist verbunden mit AD_1
DSP1_OUT	serielles Audiosignal:	1. Ausgang des 1. KanalDSPs ist verbunden mit dem seriellen FPGA-Eingang
DSP2_IN	serielles Audiosignal:	Eingang des 2. KanalDSPs ist verbunden mit AD_2
DSP2_OUT	serielles Audiosignal:	1. Ausgang des 2. KanalDSPs ist verbunden mit dem seriellen FPGA-Eingang
DSP3_IN	serielles Audiosignal:	Eingang des 3. KanalDSPs ist verbunden mit AD_3
DSP3_OUT	serielles Audiosignal:	1. Ausgang des 3. KanalDSPs ist verbunden mit dem

		seriellen FPGA-Eingang
DSP4_IN	serielles Audiosignal:	Eingang des 4. KanalDSPs ist verbunden mit AD_4
DSP4_OUT	serielles Audiosignal:	1. Ausgang des 4. KanalDSPs ist verbunden mit dem seriellen FPGA-Eingang
AD_1	serielles Audiosignal:	digitales Stereosignal (2 Kanal) vom 1. AD-Wandler kommend (Eingänge 1 und 2).
AD_2	serielles Audiosignal:	digitales Stereosignal (2 Kanal) vom 2. AD-Wandler kommend (Eingänge 3 und 4).
AD_3	serielles Audiosignal:	digitales Stereosignal (2 Kanal) vom 3. AD-Wandler kommend (Eingänge 5 und 6).
AD_4	serielles Audiosignal:	digitales Stereosignal (2 Kanal) vom 4. AD-Wandler kommend (Eingänge 7 und 8).
DA_1	serielles Audiosignal:	zur Ausgabe fertig berechnetes Stereosignal für die Ausgänge 1 und 2
DA_2	serielles Audiosignal:	zur Ausgabe fertig berechnetes Stereosignal für die Ausgänge 3 und 4
DA_3	serielles Audiosignal:	zur Ausgabe fertig berechnetes Stereosignal für die Ausgänge 5 und 6
DA_4	serielles Audiosignal:	zur Ausgabe fertig berechnetes Stereosignal für die Ausgänge 7 und 8
CHAIN_12	serielles Audiosignal:	2. Ausgang des 1. KanalDSPs (2 Kanal Audiobus) ist verbunden mit dem 2 KanalDSP
CHAIN_23	serielles Audiosignal:	2. Ausgang des 2. KanalDSPs (2 Kanal Audiobus) ist verbunden mit dem 3 KanalDSP
CHAIN_34	serielles Audiosignal:	2. Ausgang des 3. KanalDSPs (2 Kanal Audiobus) ist verbunden mit dem 4 KanalDSP
CHAIN_4M	serielles Audiosignal: (2 Kanal Audiobus)	2. Ausgang des 4. KanalDSPs ist verbunden mit dem seriellen Eingang des MixDSPs.
DOUT	serielles Audiosignal:	zur Ausgabe fertig berechnetes Stereosignal für die Ausgänge 7 und 8
GAIN12	Latchsignal:	steigende Flanke bewirkt die Datenübernahme des Gain-Controllers 1
GAIN34	Latchsignal:	steigende Flanke bewirkt die Datenübernahme des Gain-Controllers 2
GAIN56	Latchsignal:	steigende Flanke bewirkt die Datenübernahme des Gain-Controllers 3
GAIN78	Latchsignal:	steigende Flanke bewirkt die Datenübernahme des Gain-Controllers 4
DSP1_RS\	Resetleitung:	Bewirkt Neustart des KanalDSP; Um z.B. ein neues Programm zu starten.
DSP2_RS\	Resetleitung:	Bewirkt Neustart des KanalDSP; Um z.B. ein neues Programm zu starten.
DSP3_RS\	Resetleitung:	Bewirkt Neustart des KanalDSP; Um z.B. ein neues Programm zu starten.
DSP4_RS\	Resetleitung:	Bewirkt Neustart des KanalDSP; Um z.B. ein neues Programm zu starten.
DSP1_CS\	Chipselect:	Um Daten zum Hostport des 1. KanalDSPs zu senden wird diese Leitung Low.
DSP2_CS\	Chipselect:	Um Daten zum Hostport des 2. KanalDSPs zu senden wird diese Leitung Low.
DSP3_CS\	Chipselect:	Um Daten zum Hostport des 3. KanalDSPs zu senden wird diese Leitung Low.
DSP4_CS\	Chipselect:	Um Daten zum Hostport des 4. KanalDSPs zu senden wird diese Leitung Low.
AA[0:9]	Audioadressbus:	Wird zur Adressierung der Audiokanäle verwendet (max. 1024 Stereokanäle)

DA[0:15]	Audiodatenbus:	Wird zur schnellen Übertragung von Audiodaten verwendet. (max 7.9 MByte/s)
CAS\	Column adress strobe:	Dient zur syncronisation des Datentransfers zwischen MixDSP und FPGA
WE\	Write enable:	Low-Pegel des Signals signalisiert, daß der MixDSP den Audiodatenbus belegt
TXD	Transmitter:	serielle Leitung zwischen Hostprozessor und MAX232
RXD	Receiver:	serielle Leitung zwischen Hostprozessor und MAX232
HOSTCLK	Hostclock:	syncronisiert die Bitübertragung zwischen Hostprozessor und anderen Baugruppen
HOSTDATA	Hostdata:	Datenleitung zum seriellen Transport von Steuerbefehlen, Programmdaten oder Koeffizienten.
IFDO	KanalDSP-Datenausgang:	Auf dieser Leitung sendet der KanalDSP die angeforderten Daten zum Hostprozessor
IFCK	KanalDSP-Clockleitung:	verbunden mit der Hostclock
IFCD	Commando/Data Select:	Diese Leitung signalisiert, ob es sich bei der Kommunikation zwischen Hostprozessor und KanalDSP um Daten oder um Commandos handelt.
ACK\	Acknowledge:	Bestätigungssignal zur Sicherung des Datenaustausches zwischen Hostprozessor und KanalDSP.
MIX_CS\	Chipselect:	Um Daten zum Hostport des MixDSPs zu senden wird diese Leitung Low.
HX	Host Transmit:	Dieses Signal signalisiert dem Hostprozessor das der MixDSP daten senden will

10 Abbildungsverzeichnis

11 Tabellenverzeichnis

12 Literaturverzeichnis

13 Stichwortverzeichnis

14 Eidesstattliche Erklärung

Eidesstattliche Erklärung

Hiermit versichere ich, daß ich die vorliegende Diplomarbeit selbständig erarbeitet habe. Als Hilfsmittel diente mir die angegebene Literatur.

Neukirchen-Riebelsdorf, den 16. Dezember 1996

$\phi\beta\kappa\iota$OΛ◊ℑΔλη

-Ingmar Klippert-