VLSI - Entwurf für ein digitales Perzeptionsmodell des menschlichen Hörens

Dissertation

zur Erlangung des akademischen Grades
Doktor der Naturwissenschaften (Dr. rer. nat.)

am Fachbereich Informatik der Universität Hamburg

vorgelegt von

Alexander Schwarz

aus Dresden

Hamburg, März 2002

Genehmigt vom Fachbereich Informatik der Universität Hamburg
auf Antrag von:

Prof. Dr.-Ing. Bärbel Mertsching (Universität Hamburg)
Prof. Dr.-Ing. Wolfgang Nebel (Universität Oldenburg)
Prof. Dr.-Ing. Rolf-Rainer Grigat (TU Hamburg-Harburg)

Einreichung: Hamburg, 26. November 2001

Disputation: Hamburg, 15. März 2002

Prof. Dr.-Ing. H. Siegfried Stiehl
Dekan

Kurzfassung

Mit der Einbeziehung psychoakustischer Modelle in technische Systeme für die Sprachverarbeitung, wie automatische Spracherkennung, Sprachgütebewertung in Kommunikationssystemen und digitale Hörgeräte kann eine signifikante Verbesserung der Verarbeitungsqualität erreicht werden. Die vorliegende Dissertation beschreibt sowohl eine Konzeption zur geeigneten Optimierung audiosignalverarbeitender Algorithmen, als auch einen Systementwurf für die Verifikation, den Test und den direkten Einsatz der dafür entwickelten hochintegrierten anwendungsspezifischen Schaltungen. Für die nichtlinearen adaptiven Nachregelschleifen des eingesetzten Oldenburger Perzeptionsmodells wurde mit Hilfe eines C/C++ Prototypen eine zweigeteilte Architektur mit Fest- und Fließkomma-Einheiten entwickelt, die in ihrer Präzision an die lokale Signaldynamik angepaßt worden ist (Operandenpräzision 14 Bit Mantisse für Addition, Subtraktion und Multiplikation und 6 Bit für Division bei jeweils 6 Bit exzeß-kodierter Charakteristik). Im Zusammenwirken mit einem dafür angepaßten Verfahren zur Sprachgütebeurteilung durch das Perzeptionsmodell selbst konnte die optimale Architektur bestimmt und validiert werden, indem nur die perzeptiv relevanten Signalverzerrungen am Ausgang des Perzeptionsmodells zur Berechnung eines Gütemaßes herangezogen wurden.

Für die Synthese der Algorithmen wurde kein konventioneller struktureller Ansatz gewählt. Vielmehr wurde ein durchgängiger Entwurfsfluß für die direkte Datenpfadsynthese mit einem speziellen CAD-Werkzeug (*Synopsys Behavioral Compiler*) entwickelt. Nach der Erarbeitung einer synthetisierbaren VHDL-Beschreibung des Algorithmus' wurden die mit dem C/C++ Prototypen spezifizierten VHDL-Fließkomma-Operatoren so erstellt, daß sie durch Operatorüberladung inferiert werden können. Dabei mußten insbesondere die Einflüsse und Rückwirkungen aus Architekturvarianten, Randbedingungen und Beschränkungen des Design-Systems, der untersuchten FPGA- (*Xilinx Virtex* und *Altera Flex*) und der ASIC-Technologien (*ES2-07*) auf die Optimierung des Entwurfsflusses und des Algorithmus' selbst berücksichtigt werden. Der Entwurfsfluß ist durch die Abläufe des abschnittsweise automatischen CAD-Werkzeugs nur grob vorgegeben und mußte durch eine skript-basierte Steuerung zum Erreichen der gewünschten Ergebnisse und zur Umgehung von Bibliotheks- und Werkzeuginkonsistenzen individuell angepaßt und optimiert werden. Das Verifikationsverfahren für Optimierungen am Algorithmus und dem Datenformat wurde so ausgebaut, daß die Simulation mit dem ursprünglichen C-Kode, dem optimal quantisierenden C++ Prototypen und den VHDL-Modellen in jeweils derselben Umgebung möglich ist. Das hier erarbeitete Hardware-Design konnte auf Verhaltens-, Register-Transfer- und Layoutebene korrekt simuliert werden; es benötigt in einem *Xilinx Virtex*-FPGA *XCV800* nur 19 % der Gatterressourcen (äquivalente Gatterzahl: 80.000) bzw. 10 % des Onchip-RAM-Speichers und erzielt eine Taktfrequenz von 40 MHz. Mit einer Gesamtverzögerung von 3,5 µs je Abtastwert werden die für eine Echtzeit-Datenverarbeitung maximal zulässigen 4,096 µs deutlich unterschritten. Mit der Integration aller Bestandteile des entwickelten Chipsatzes in ein kommerzielles Prototypensystem und der Einbindung in eine PC-basierte Arbeitsumgebung steht damit ein vollständiges und echtzeitfähiges Audiosignalverarbeitungssystem den gewünschten Anwendungen zur Verfügung.

Abstract

A significant processing quality improvement can be achieved for technical speech processing systems such as automatic speech recognition, speech quality estimation in communication systems and digital hearing aids by the integration of psychoacoustic models. The presented Ph. D. thesis describes a conception for a qualified optimization of audioprocessing algorithms as well as a system approach for verification, test and direct application of dedicated integrated circuits. A mixed fix point/floating point architecture was developed for the nonlinear adaptation loops of the implemented Oldenburger Perception Model. The resolution of these arithmetical units which can be scaled due to the local signal dynamic throughout the model (optimal operand precision 14 bit mantissa for addition, subtraction and multiplication and 6 bit for division each with a 6 bit excess-coded characteristic) has been specified using a C/C++ prototype. The optimized architecture has been estimated and validated in conjunction with an adapted method for the speech quality estimation by the perception model itself, since the perceptual relevant signal distortions at the model output are used for the calculation of a quality measure.

A comprehensive design flow for the direct datapath synthesis was chosen for the algorithms using a special CAD tool (*Synopsys Behavioral Compiler*) rather than a conventional structural synthesis approach. After establishing a synthesizable VHDL description of the algorithm itself the arithmetical VHDL operators specified with the C/C++ prototype have been developed for direct operator overloading and inference. Especially the influence and feedback of architectural options, boundary conditions, and limitations of the design tool and the tested FPGA (*Xilinx Virtex, Altera Flex*) and ASIC (*ES2-07*) technologies had to be taken into consideration for the optimization of the design flow and the algorithm itself. The design flow is roughly given by the partially automatic design tool and had to be adapted by script based controlling in detail to achieve the desired results and to avoid library and tool inconsistencies. The verification method for the algorithm and data type optimizations were extended for the simulation and validation of the original C-code, the optimally quantisizing C/C++ prototype and the VHDL-models in the same environment. The presented hardware design has been correctly simulated on the behavioral, register-transfer and layout level. The overall usage of a *Xilinx Virtex* FPGA *XCV 800* is about 19 % for logic resources (equivalent gate count 80.000), 10 % for the internal RAM cells and the maximum clock rate is about 40 MHz. With an overall delay of 3,5 µs for the processing of one sample the real-time demands of at minimum 4,096 µs were clearly met. A real-time capable audio-signal processing system is available for the desired applications by the integration of all components of the developed chip set in a commercial hardware prototyping board and the implementation in a PC-based working environment.

Danksagung

Herzlich gedankt sei all denjenigen, die über mehrere Jahre hinweg auf vielfältige Art und Weise zum Gelingen und zum Abschluß der vorliegenden Arbeit beigetragen haben. Das gilt vor allem Professor Dr. Bärbel Mertsching und der IMA-Arbeitsgruppe und den Gutachtern Professor Dr. Wolfgang Nebel und Professor Dr. Rolf-Rainer Grigat. Besonderen Dank möchte ich den ehemaligen Kollegen Dr. Maik Bollmann, Rainer Hoischen, Dr. Steffen Schmalz und Gerriet Backer sowie den am Projekt beteiligten studentischen Mitarbeitern Johannes Bitterling, Michael Scholz und Nikolaus Voss aussprechen, ohne deren Unterstützung keine solche Effizienz möglich gewesen wäre.

Den Mitstreitern im Projekt „Silicon Ear“ an der Universität Oldenburg in den Arbeitsgruppen Medizinische Physik mit Professor Dr. Dr. Birger Kollmeier, Dr. Martin Hansen, Michael Kleinschmidt und Dr. Jürgen Tchorz ebenso wie den Beteiligten aus der Arbeitsgruppe „Entwurf Integrierter Schaltungen“ Professor Dr. Wolfgang Nebel und insbesondere Matthias Brucke danke ich für die gedeihliche, sehr angenehme und freundschaftliche Kooperation.

Gern erinnere ich mich an die früheren Kollegen vom Fraunhofer Institut für mikroelektronische Schaltungen und Systeme in Dresden, Dr. Günther Despang, Dr. Eberhard Oberst, Ljudmilla Kleinmann und Wolfram Kluge, denen ich dafür danken möchte, mich ursprünglich für die hier behandelte Thematik interessiert und zahlreiche Vorarbeiten ermöglicht und intensiv unterstützt zu haben.

Allen, die nicht die Mühe scheuten, den vorliegenden Text zu prüfen, sei ebenfalls herzlich gedankt, insbesondere meinem Vater für die profunde Durchsicht.

Inhaltsverzeichnis

Kapitel 1

Einleitung

Die Gestaltung und Optimierung technischer Systeme für die Audiosignalverarbeitung hat in den letzten Jahren durch die Anwendung psychoakustischer Erkenntnisse und den Einsatz der gehörgerechten Signalvorverarbeitung zu großen Fortschritten geführt. So ermöglicht die perzeptive, an der Funktionsweise des menschlichen Ohres orientierte, Kodierung von Audiodaten eine signifikante Datenreduktion in kommerziellen Audiosystemen (*Sony Minidisc*, *MPEG-Layer3*). Vor allem aber profitieren Systeme zur Sprachverarbeitung, wie automatische Spracherkennung, Sprachgütebewertung in Kommunikationssystemen und digitale Hörgeräte von der Einbeziehung psychoakustischer Modelle. In diesen Modellen können dabei unterschiedliche Abstraktionsebenen für die Abbildung der psychoakustischen Prozesse berücksichtigt werden. Aber auch optimierte Ansätze sind hoch komplex und mit einem großen Rechenaufwand verbunden, womit insbesondere ein Einsatz der technisch und wirtschaftlich attraktiven digitalen Realisierungen solcher Perzeptionsmodelle erschwert wird.

Die vorliegende Dissertation beschreibt sowohl eine Konzeption zur geeigneten Optimierung audiosignalverarbeitender Algorithmen und deren praktische Umsetzung in anwendungsspezifische Hardware, als auch einen Systementwurf für die Verifikation, den Test und den direkten Einsatz der dafür entworfenen integrierten Schaltungen. Die Voraussetzung hierfür, ein binaurales psychoakustisches Modell, wurde im Hinblick auf die oben genannten Zielapplikationen für die Sprachvorverarbeitung in der Arbeitsgruppe „Medizinische Physik“ der Universität Oldenburg entwickelt. Dieses Modell findet unter Berücksichtigung wesentlicher Funktionseinheiten der effektiven auditorischen Signalverarbeitung einen Kompromiß zwischen notwendiger Modellierungstiefe und möglichst geringem Berechnungsumfang. Die Hauptbestandteile des Perzeptionsmodells sind eine Gammatone-Filterbank zur spektralen Zerlegung des akustischen Signals, ein Modell der inneren Haarzellen zur neuronalen Wandlung und ein nichtlinearer Teil zur Nachbildung zeitlicher Kontrastierungs- und Verdeckungseffekte in Form von fünfstufigen Adaptationsschleifen. Da die Echtzeiteigenschaft der zugrundeliegenden Algorithmen die Vorbedingung für den praktischen Einsatz in den Zielapplikationen ist, muß die trotz des Zuschnitts verbleibende Komplexität der Signalverarbeitung durch eine geeignete Hardware-Beschleunigung bewältigt werden. Die Dissertation war eingebettet in das durch die Deutsche Forschungsgemeinschaft geförderte Projekt „*System- und Schaltungstechnik einer integrierten Cochlea für Sprachanalyse, Spracherkennung und Sprachcodierung*“[1]. Das Projektziel sollte durch die Entwicklung einer integrierten Lösung für ein programmierbares Gatter-Array (FPGA) bzw. einen anwendungsspezifischen integrierten Schaltkreis (ASIC) untergliedert in zwei Teildesigns unterstützt werden. Die Arbeitsteilung im zugrundeliegenden Projekt ermöglichte, daß sich die vorliegende Dissertation auf die Voruntersuchungen und die Implementierung der nichtlinearen Adaptationsschleifen des „*Silicon Ear*" und die System- und

Integrationsaspekte konzentrieren konnte. Bei der Umsetzung und der Transformation der digitalen Algorithmen aus der ursprünglichen C/C++ Notation sind die primären Design-Entscheidungen für den Hardware-Entwurf von hoher Bedeutung. Voruntersuchungen zu Abtastfrequenz, Dynamikumfang, Präzision der Wertedarstellung und typischen Signalformen beabsichtigter Anwendungen bestimmen Zahlendarstellung, Architektur und Speicherbedarf. Besondere Bedeutung kommt der Bewertung von Verzerrungen durch Wortlängenbegrenzung des Hardware-Algorithmus zu. Eine 1:1 Übertragung des IEEE-Formats für Fließkommazahlen (*single precision)* aus der C-Notation ist infolge der Komplexität der benötigten Operatoren (Addition, Subtraktion, Multiplikation, Division) und des Ressourcenverbrauchs (Fläche, Leistung) nicht durchführbar. Es wird demnach eine geringer auflösende Darstellung sowohl für ein Festkomma-Format als auch ein angepaßtes Fließkomma-Format untersucht und bewertet. Dabei soll weniger das dabei entstehende allgemeine Quantisierungsrauschen, sondern vor allem der perzeptiv relevante Anteil minimiert werden. Für das „*Silicon Ear*“ soll eine Systemlösung entstehen, d. h. die Hardware muß so spezifiziert und ausgestattet werden, daß die Integration in ein Signalverarbeitungssystem (PC und oder DSP) für Weiterentwicklungen von Algorithmen etc., aber auch zu Testzwecken ermöglicht wird. Dabei muß die Funktionalität des Chip-Satzes und der externen Geräte z. B. hinsichtlich der Schnittstellen aber auch der allgemeinen technischen Anforderung (Bauform etc.) bereits im Vorfeld festgelegt und hinsichtlich der technischen Realisierbarkeit überprüft werden. Der klassische Kompromiß zwischen Größe und Verzögerungszeit der zu synthetisierenden Schaltungen muß unter diesen äußeren Bedingungen gefunden werden. Weitere Bedingungen ergeben sich aus der verwendeten Halbleitertechnologie und den CAD-Werkzeugen. Ein zeiteffizienter Hardware-Entwurf ist in hohem Maße von der Konsistenz der Synthese- und Technologiebibliotheken des Entwurfswerkzeuges, vor allem aber von der Durchgängigkeit des Entwurfsflusses abhängig. Dieser ist durch die Abläufe des abschnittsweise automatischen CAD-Werkzeuges grob vorgegeben, muß aber zum Erreichen der erforderlichen Ergebnisse und zur Vermeidung von Inkonsistenzen individuell angepaßt und optimiert werden.

Die folgenden Kapitel beschreiben den Weg von der Systemanalyse bis hin zur Realisierung einer integrierten Schaltung für ein FPGA bzw. ASIC, wobei die Schwerpunkte auf den speziell für diese Art der Sprachsignalverarbeitung entwickelten arithmetischen Strukturen, den angewandten Entwurfsverfahren, dem Entwurfsfluß für das CAD-Werkzeug, der Verifikation des „*Silicon Ear*" in einer Zielanwendung und der Systemintegration liegen. Damit beschränkt sich diese Arbeit nicht nur auf eine konzeptionelle Beschreibung zur Hardware-Umsetzung von audiosignalverarbeitenden Algorithmen, sondern erläutert auch die Lösung der praktischen Probleme beim Umgang mit den verwendeten CAD-Werkzeugen für den VLSI-Entwurf.

[1] Dieses von der Deutschen Forschungsgemeinschaft (DFG) unter den Leitzeichen Ko 942/12 und Me 1289/4 geförderte Projekt mit dem Titel „*System- und Schaltungstechnik einer integrierten Cochlea für Sprachanalyse, Spracherkennung und Sprachcodierung“* (zunächst unter dem Kurztitel „*Integrierte Cochlea“*, später unter „*Silicon Ear“* firmierend) wurde in enger Kooperation zwischen den Arbeitsgruppen „Medizinische Physik“ (Fachbereich Physik) und „Entwurf integrierter Schaltungen“ (Fachbereich Informatik) der Carl-von-Ossietzky-Universität Oldenburg einerseits und der Arbeitsgruppe „Informatikmethoden für Mikroelektronikanwendungen“ der Universität Hamburg andererseits durchgeführt, wobei die vorliegende Dissertation in der zuletzt genannten Gruppe entstand.

Kapitel 2

Physiologische und psychoakustische Aspekte des menschlichen Gehörs

Dieses Kapitel gibt einen Überblick über physiologische und psychoakustische Eigenschaften des auditorischen Systems des Menschen. Hierbei werden jene Aspekte des Gehörs behandelt, die zur Motivation und zum Verständnis der auditorischen Modelle beitragen. Ausgehend vom anatomischen Aufbau und der Erläuterung einzelner Funktionseinheiten werden die für die Ton- und Sprachperzeption wichtigen Empfindungsgrößen und Zusammenhänge zwischen der Frequenz- und Intensitätswahrnehmung bzw. dem zeitlichen Verlauf (Modulation) des einfallenden Schallsignals vorgestellt.

2.1 Das auditorische System im Überblick

Das menschliche Gehör (siehe Abb. 2.1) besteht wie bei Säugetieren allgemein aus Außen-, Mittel- und Innenohr, wobei die aktiven Prozesse der Schallübertragung im Mittel- und Innenohr stattfinden.

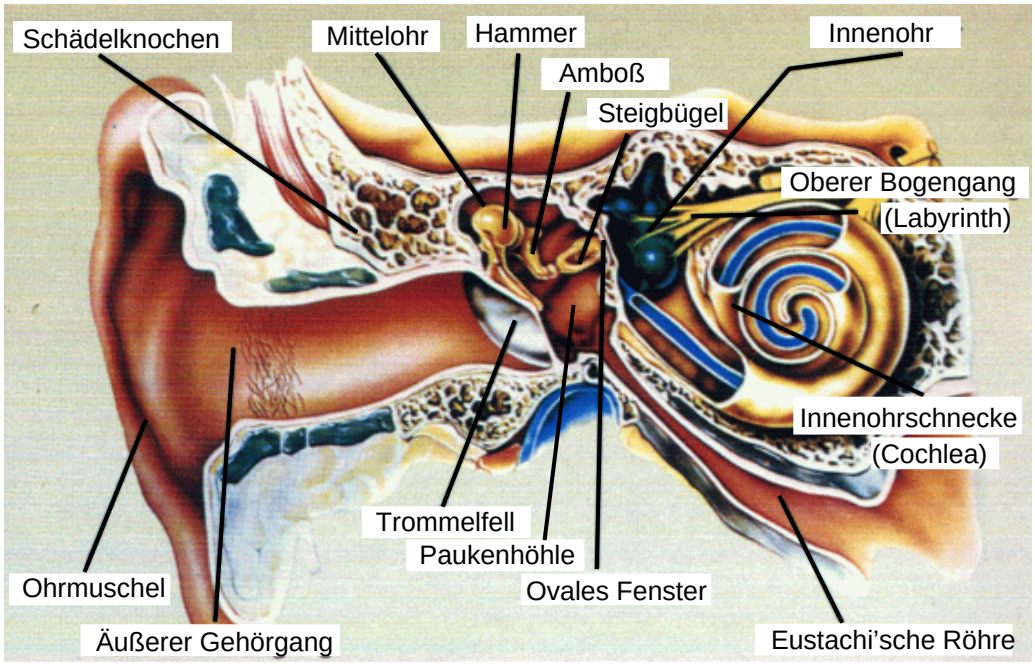


Abb. 2.1: Schnittdarstellung und Aufbau der inneren und äußeren Teile des menschlichen Ohres. [Philips Hearing Instruments 2001].

Das Außenohr, insbesondere die Ohrmuschel, verursacht eine richtungsabhängige Filterung des Schallsignals, die eine wesentliche Voraussetzung zur Schallortung ist, d. h. zur Unterscheidung, ob Signale von vorne oder hinten eintreffen. Das Schallsignal wird für hohe Frequenzen durch die Trichterform gebündelt, je nach Eintrittswinkel gefiltert und durch die Form (Länge) des Gehörganges im Bereich einer sich daraus ergebenden Resonanzfrequenz bei etwa 2-3 kHz um 10 bis 15 dB verstärkt.

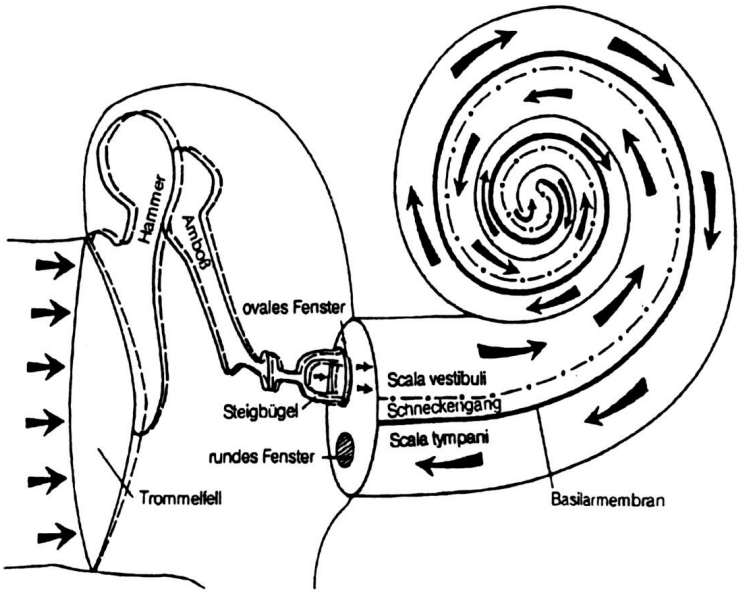
Das Trommelfell trennt und schützt das Mittelohr vor der Außenwelt und wird durch die eintreffenden Schalldruckwellen ausgelenkt. Für eine korrekte Auslenkung um eine Mittenlage müssen ausgeglichene Druckverhältnisse zwischen der Paukenhöhle des Mittelohres und dem Umgebungsluftdruck bestehen, was über die *Eustachi'sche* Röhre hergestellt wird.

Die Hauptfunktion des Mittelohres besteht in der Impedanzanpassung zwischen dem vom schallweichen Medium Luft umgebenen Trommelfell und den, unten näher beschriebenen, mit einem „schallharten" flüssigen Medium gefüllten Gängen des Innenohres, der Cochlea (siehe auch Abb. 2.2). Es gibt insgesamt drei Gehörknöchelchen: der Hammer ist mit dem Trommelfell verwachsen und überträgt dessen Schwingungen auf den Amboß, der den Steigbügel auslenkt. Dessen Fußplatte bewegt daraufhin die Membran im ovalen Fenster des Innenohres. Das Flächenverhältnis zwischen Trommelfell und Größe der Steigbügel-Fußplatte - vor allem aber das Hebelverhältnis der Gehörknöchelchen - erzeugt eine um den Faktor 50 höhere Auslenkkraft je Flächeneinheit. Im Falle nicht angepaßter Impedanzen bzw. einer direkten Auslenkung der Membran im ovalen Fenster würde die meiste Schallenergie dort wieder reflektiert. Im Mittelohr existieren außerdem zwei Muskeln (*musculus stapedius*, *musculus tensor tympani*) die zugleich die kleinsten Skelettmuskeln des Menschen sind. Die Effizienz der Übertragung ist vor allem durch den am Steigbügel (*stapes*) ansetzenden *Stapedius*-Muskel beeinflußbar. Efferent (absteigend) innerviert reagiert dieser bei hohen Schalldrücken, verschlechtert durch Kontraktion die Impedanzanpassung zum ovalen Fenster und übt damit eine Schutzfunktion für das Innenohr aus (Stapediusreflex).

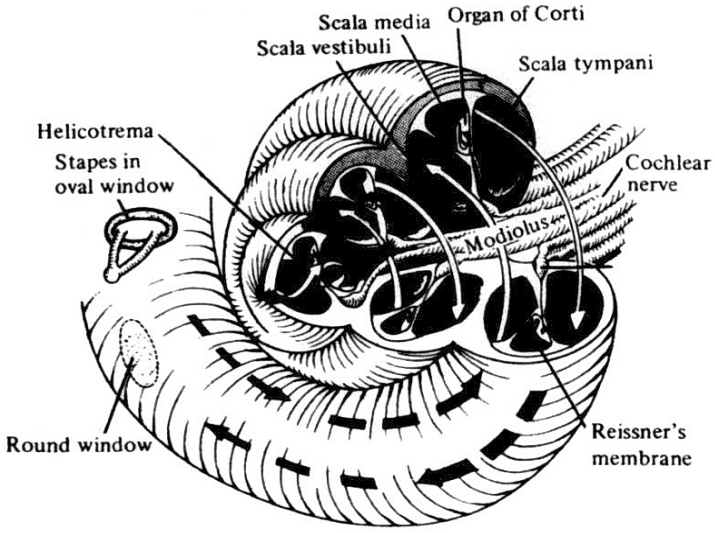
Im Innenohr findet die eigentliche Transformation mechanischer Schallwellen in neuronale Erregungsmuster statt. Wie in Abb. 2.2 gezeigt, breiten sich die Schalldruckwellen in der mit Perilymphflüssigkeit gefüllten *scala vestibuli* vom ovalen Fenster her aus, laufen bis zum Helicotrema an der Spitze der Innenohrschnecke, wo eine Verbindung zur *scala tympani* besteht, und in dieser zurück bis ans Mittelohr. Hier kann der notwendige Druckausgleich durch das Auslenken der Membran am runden Fenster geschehen.

Die *scala vestibuli* wird an der inneren Seite von der Reissner'schen Membran abgeschlossen, hinter der sich eine dritte Skale, die *scala media*, erstreckt. Zwischen der *scala media* und der *scala tympani* verläuft die cochleäre Trennwand, welche die für die Innenohrfunktion wichtigsten Bestandteile der Cochlea beherbergt. Das sich an der cochleären Trennwand befindliche mit neuronalen Wandlern ausgestattete mikromechanische Resonanzsystem umfaßt die Tekto-

rialmembran (oder Dachmembran), das Corti'sche Organ und die Basilarmembran (siehe auch Abb. 2.4).



(a)



(b)

Abb. 2.2: Schematische Darstellung des Innenohres und der Übertragung der Auslenkungsbewegung am Trommelfell in Druckschwankungen entlang der Innenohr-Gänge (*scala vestibuli, scala media, scala tympani*). Das Innenohr besitzt „aufgerollt" eine Länge von 3,2 cm und beschreibt 2,5 Windungen bis zur Spitze (Helicotrema) (a) [Hellbrück 1993] und (b) [Evans 1982].

Durch die Druckdifferenz der sich in der Perilymphe ausbreitenden Schalldruckwellen zwischen den Skalen kommt es zur Ausprägung einer Wanderwelle auf der Basilarmembran (Abb. 2.3) und damit zu einer Schwingungseinkopplung auf die o. g. Bestandteile der cochleären Trennwand. Da die Steifigkeit und die Dicke der Basilarmembran entlang der Längsachse

abnehmen, während ihre Breite zunimmt, kommt es bereits durch passive Resonanz zu einer Frequenzdispersion des Schallsignals, sofern dieses aus einem Frequenzgemisch besteht.

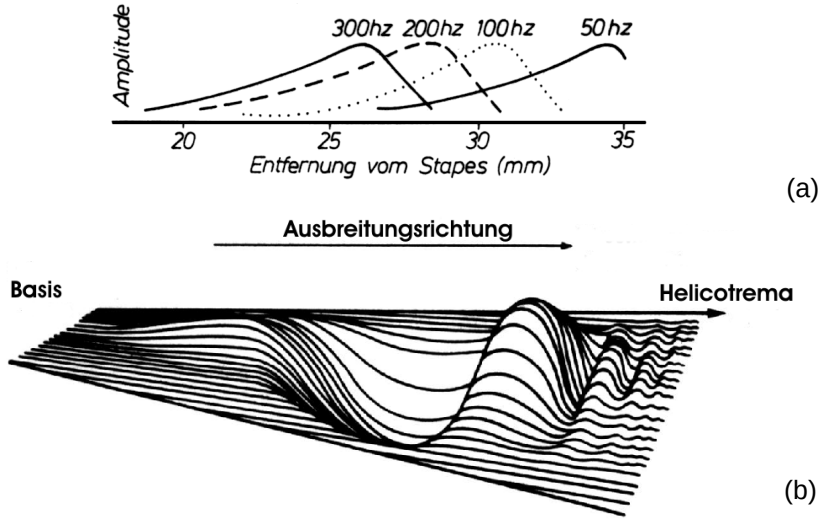


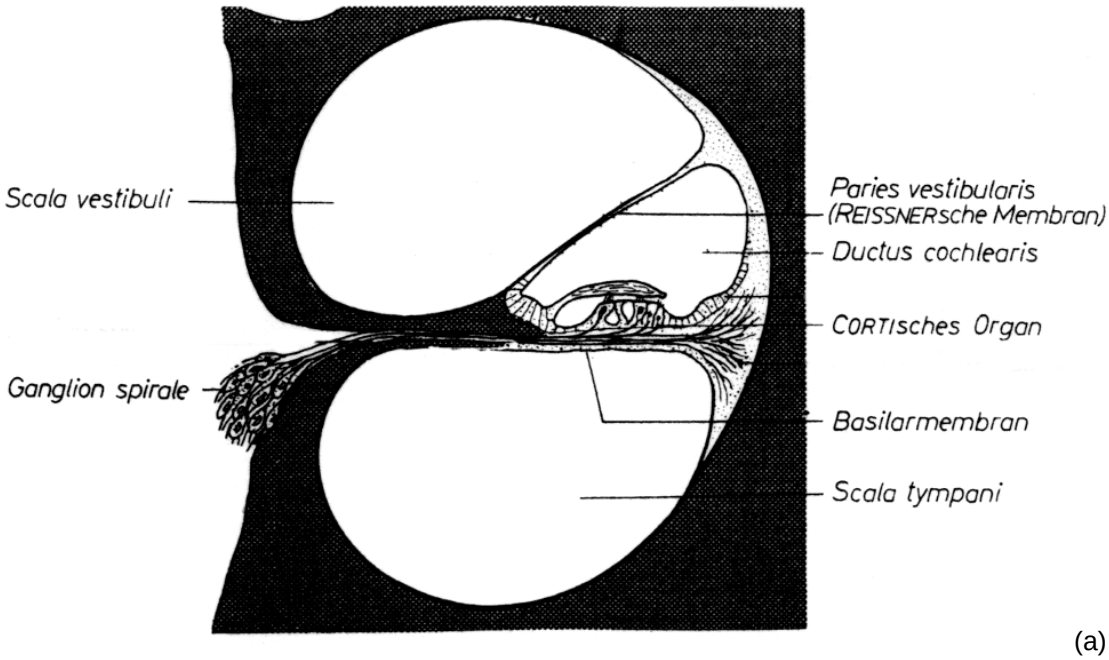
Abb. 2.3: Auslenkungsmaxima einer Wanderwelle bei 300 Hz, 200 Hz, 100 Hz und 50 Hz (a) [Oeken 1984] und die Ausbreitung der Wellen entlang der Basilarmembran zwischen Basis und Helicotrema (b) nach [Evans 1982].

Für eine Frequenz zwischen etwa 20 und 20.000 Hz (maximaler Hörbereich) kann somit an einem bestimmten Ort der Basilarmembran ein Auslenkungsmaximum beobachtet werden: hohe Frequenzen bilden sich in der Nähe des ovalen Fensters (Basis der Basilarmembran) ab, während in Richtung Helicotrema die tieferen Frequenzen Maxima erzeugen (Frequenz-Orts-Transformation).

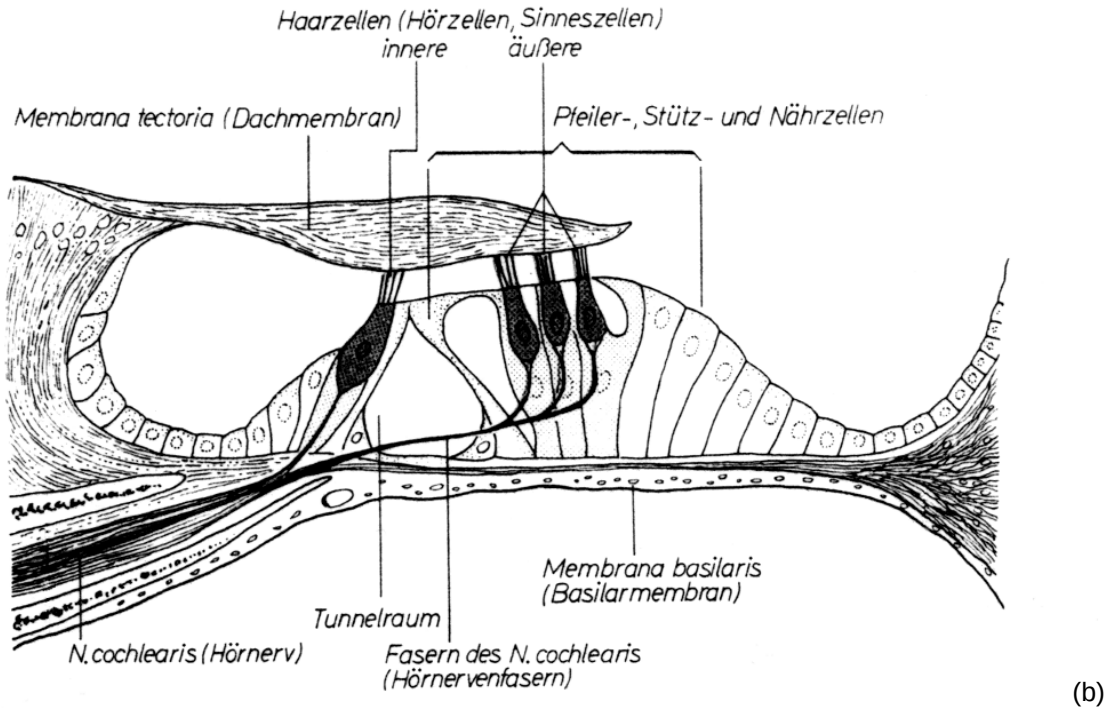
Im Corti'schen Organ, verursachen die Wanderwellen eine Lageverschiebung zwischen Tektorial- und Basilarmembran und damit eine Scherkraft an den Haarbündeln (Stereozilien) der Haarzellen. Dieser mechanischer Reiz löst im Inneren der Haarzellen eine Konzentrationsfluktuation von Ladungsträgern (Ionen) aus, womit ein Spannungsimpuls in den Enden der hier ansetzenden Spiral-Ganglion Neurone generiert wird. Die inneren Haarzellen sind in ihrer Funktion als Sensorzellen mit 90 - 95 % der insgesamt etwa 30.000-40.000 aufsteigenden (*afferenten*) Nervenfasern innerviert. Dabei ist jede Faser mit nur einer inneren Haarzelle, die Haarzelle aber mit bis zu 20 Fasern verbunden.

Neben den afferenten Fasern enden auch etwa 1.000 absteigende (*efferente*) Fasern im Corti'schen Organ, jedoch mehrheitlich (95 %) an den äußeren Haarzellen. Deren Anzahl übersteigt die der etwa 3.500 inneren Haarzellen mehrfach: einer Reihe innerer Haarzellen stehen insgesamt etwa 13.000 äußere Haarzellen in drei bis vier Reihen gegenüber (siehe Abb. 2.4 und

Abb. 2.5). Hierbei ist jede efferente Faser mit bis zu 50 äußeren Haarzellen, diese jedoch mit bis zu sechs Fasern gleichzeitig verbunden.



(a)



(b)

Abb. 2.4: Schnitt durch eine Windung der Cochlea (a), eine detaillierte Darstellung der Anatomie des Corti'schen Organs (b) [Oeken 1984].

Für die äußeren Haarzellen wurden motile Eigenschaften nachgewiesen, d. h. sie können nach Anregung kontrahieren und, da deren Stereozilien fest mit der Tektorialmembran verbunden sind, die Schwingungseigenschaften der Basilarmembran beeinflussen.

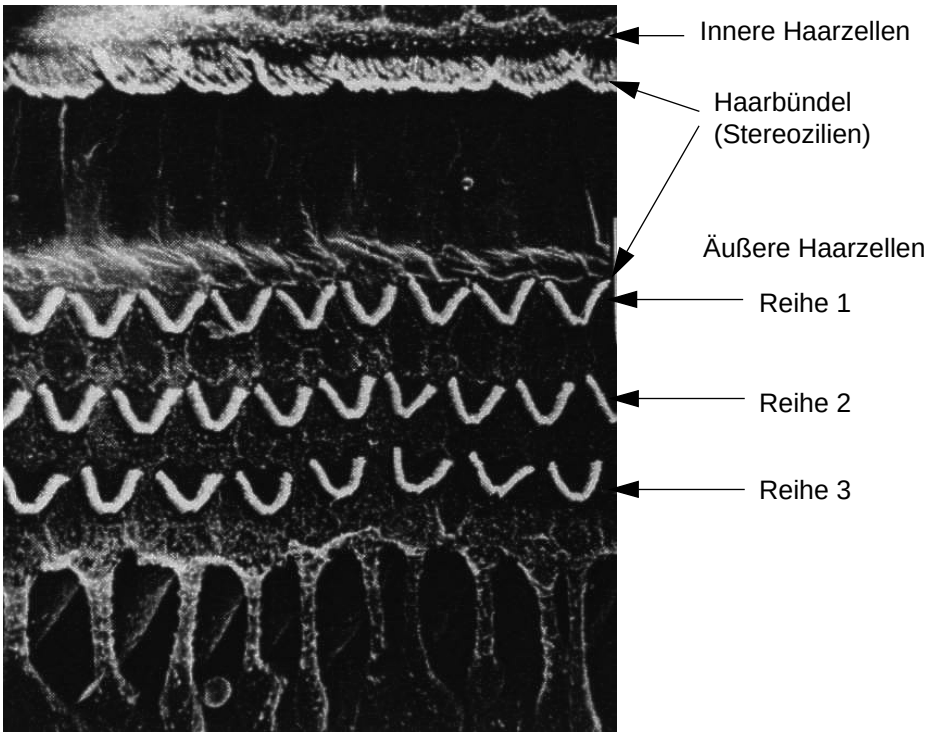


Abb. 2.5: Eine mikroskopische Aufnahme der Haarzellen auf deren Stützgewebe [Evans 1982]; im oberen Bereich befindet sich die Reihe der inneren Haarzellen darunter die drei Reihen äußerer Haarzellen.

Mit Testsignalen, wie Impulsen oder kurzen Dauertönen (*bursts*) können die Eigenschaften der Kodierung mechanischer Auslenkungsbewegung in neuronale Erregungsmuster - für diese exemplarischen Fälle - dargestellt werden. Der Potentialaufbau in den inneren Haarzellen wird nur bei beschleunigter Auslenkung der Stereozilien in einer Richtung ausgelöst; übertragen wird demnach die gleichgerichtete Zeitableitung der Basilarmembranschwingung. Ein kurzer Dauerton erzeugt eine wie in Abb. 2.6 schematisch dargestellte Impulsfolge auf einer an der Haarzelle angeschlossenen Nervenfaser.

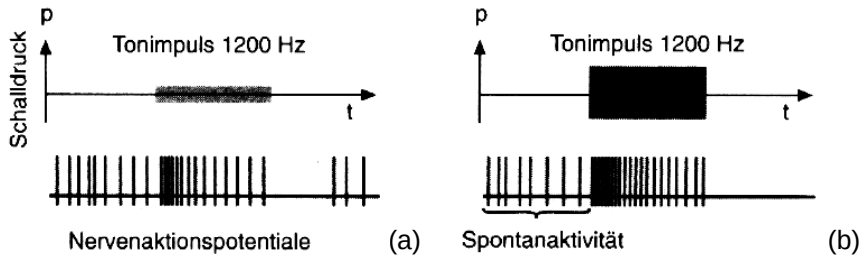


Abb. 2.6: Kodierungsprinzip für einen kurzen Dauerton in eine pulsratenmodulierte Folge von Entladungsimpulsen auf einer Nervenfaser bei niedrigem (a) und höherem Schalldruck (b) nach [Birbaumer und Schmidt 1990].

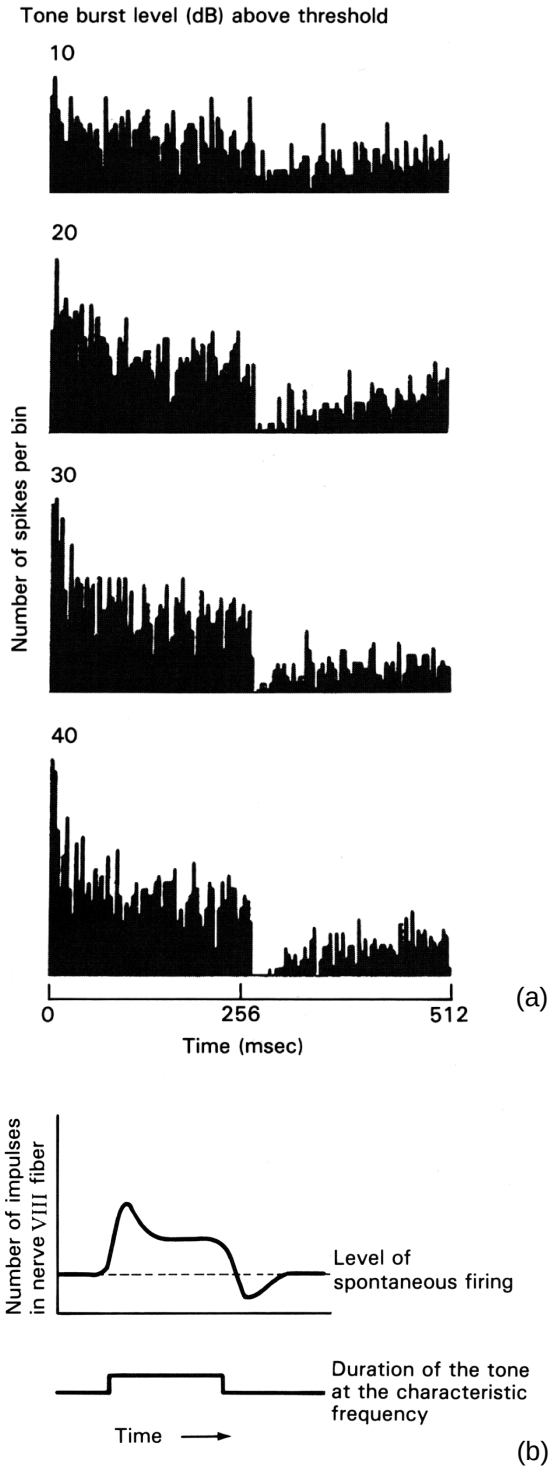


Abb. 2.7: Histogrammdarstellung für neuronale Entladungsimpulse (*spikes*) in einer Nervenfaser als Antwort auf einen kurzen Dauerton *(burst)*; bei verschiedenen Schallpegeln (a) und schematisch (b) [Kelly 1991].

Einem einzelnen neuronalen Impuls muß eine Zeitspanne der Erholung bzw. der Wiederherstellung des elektrischen Ruhepotentials folgen (Refraktärzeit), d. h. Impulse sind nicht beliebig oft auslösbar. Allerdings können die Nervenfasern bis zu einer bestimmten Frequenz der Erregung folgen und „feuern“ dann mehrheitlich synchron zur Auslenkungsphase des anregenden Schallsignals (Synchronizität bei einer Vorzugsphase). Für Frequenzen oberhalb von 1 kHz geht die Phasensynchronizität unter zunehmender Auslassung einiger Vorzugsphasen verloren. Erhöht sich die Intensität des anregenden Schalls, steigt einerseits die Phasensynchronizität als auch die Impulsrate in den Fasern. Die in Abb. 2.7 (a) dargestellten Histogramme veranschaulichen die Feuerrate einer Nervenfaser abhängig vom Lautstärkepegel über der Hörschwelle und der zeitlichen Dauer des anregenden Tones. Die Frequenz des Tones ist in diesem Beispiel so hoch, daß dessen Periode anhand des zeitlichen Verlaufs der Feuerraten nicht mehr erkennbar ist. Außerdem kann neben einer sehr kurzen Verzögerungszeit auch eine zeitliche Adaptation der Impulsrate beobachtet werden, d. h. die Antwort auf einen amplitudenkonstanten kurzen Dauerton geht sehr schnell nach der Einschaltspitze mit einer kleinen Zeitkonstante (2 bis 5 ms) und danach langsamer mit einer größeren Zeitkonstante von (30 bis 50 ms) zurück (leichter Abfall des Sockels). Das adaptive Verhalten der Nervenfaser, stark auf Änderungen zu reagieren und danach auf eine geringere stationäre Antwort einzuschwingen, bewirkt eine Dynamikkompression bzw. eine zeitliche Kontrastierung der neuronalen Antwort. Die Abb. 2.7 (b) zeigt schematisch die Hüllkurve einer Impulsraten-Antwort; nach dem Ausschalten des Reizes sinkt die Feuerrate infolge des Erholungseffektes sogar unter die Spontanrate (spontane Aktionspotentiale im Ruhezustand).

Während das auditorische System insgesamt 120 dB Dynamik verarbeiten kann, können einzelne Fasern nur etwa 20 - 40 dB in der Feuerrate kodieren. Damit ist die rezeptorische Effizienz einzelner Haarzellen sehr beschränkt; erst ganze Populationen innerer und äußerer Haarzellen und entsprechender Nervenbahnen im Verbund sind in der Lage, die Frequenz-, Dynamik- und Zeitauflösung zu bewältigen. Durch die Staffelung von inneren Haarzellen und Nervenfasern unterschiedlich hoher Schwellwerte und Spontanentladungsraten und der bei noch höheren Pegeln stattfindenden Erregung ganzer Bereiche des Innenohres kann der hohe Dynamikbereich von 120 dB abgebildet werden. Der in Abb. 2.8 dargestellte Kurvenverlauf beschreibt die neuronale Empfindlichkeit eines Ortes auf der Basilarmembran.

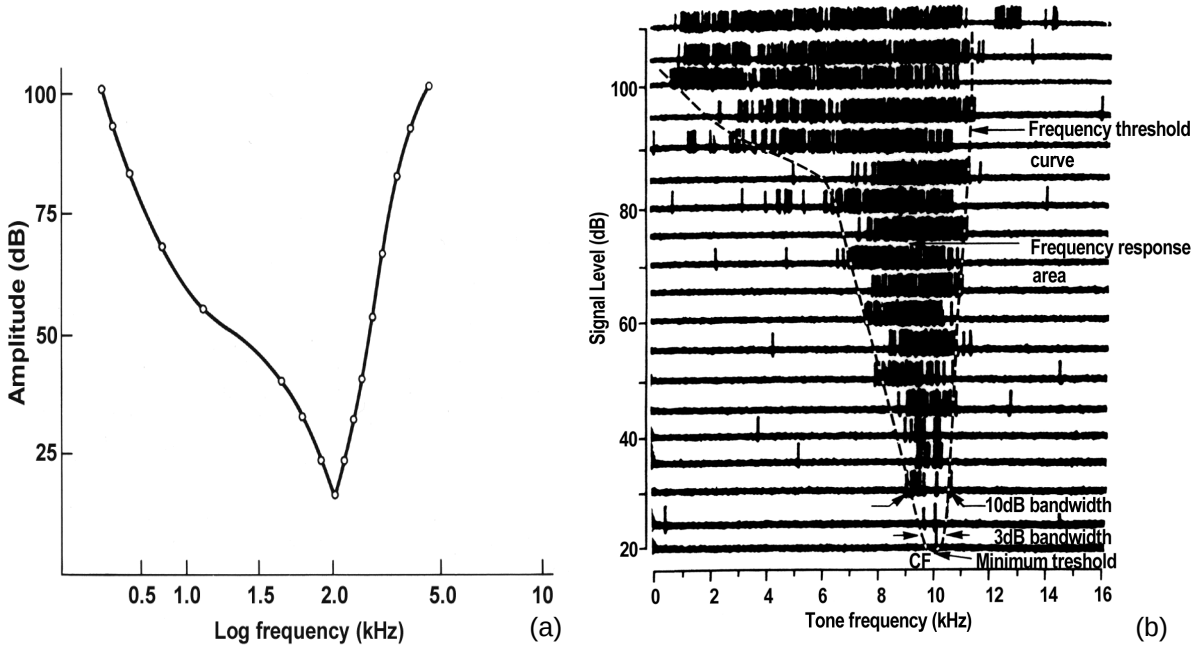


Abb. 2.8: Neuronale Abstimmkurven (*tuning curves*) des Innenohres. Der Empfindlichkeitsabfall zu höheren Frequenzen ist wesentlich stärker als zu tiefen Frequenzen hin. Die Bestfrequenzen (*Charakteristische Frequenz - cf*), in (a) bei 2 kHz [Kelly 1991] und in (b) bei 10 kHz [Evans 1982] bezeichnen die minimale frequenzabhängige Schwelle für den zugehörigen Ort auf der Basilarmembran. Im von der Abstimmkurve eingeschlossenen Bereich feuern die assoziierten Nervenfasern. Außerhalb der Kurve liegende Impulse zeigen den statistischen Charakter der Spontanentladungen.

Diese Abstimmkurven können für alle Orte der Basilarmembran bestimmt werden, wobei die Flankensteilheit und die Überhöhung an der Resonanzstelle mit zunehmender charakteristischer Frequenz anwachsen. Die Überhöhung bei *cf* bzw. die scharfe Frequenzauflösung insgesamt wird aber erst durch die Motilität und efferente Innervierung der äußeren Haarzellen erreicht. In der Literatur als „aktiver Prozeß“ bezeichnet, entsteht durch eine lokale Rückkopplung eine Art Verstärkung der Basilarmembranauslenkung, was auch als „cochleärer Verstärker“ bezeichnet wird. Dieser wirkt allerdings nur bei kleinen Pegeln bis etwa 50 dB SPL und geht schnell in die Sättigung. Die äußeren Haarzellen werden dabei durch die Scherkraft an den Stereozilien zu aktiver Kontraktion bzw. Elongation angeregt. Damit wird die Schwingung der Basilarmembran an dieser Stelle verstärkt, was infolgedessen auch die Aktivität der inneren

Haarzellen steigert. Ist dieser aktive Prozeß jedoch gestört oder unterbrochen, liegt ein typischer Innenohrhörschaden vor. Statt der scharfen Abstimmung verbleibt die rein mechanische Resonanz der Basilarmembran an diesem Punkt. Die als *Recruitment* bezeichnete Form der Schallempfindungsschwerhörigkeit beschreibt diese Einschränkung des nutzbaren Dynamikbereichs: die mechanischen Auslenkungen der Basilarmembran bei kleinen Pegeln liegen nun unterhalb der Wahrnehmungsschwelle, während mittlere Schallpegel bereits zu einer Schmerz- bzw. Lärmempfindung führen. Darüber hinaus ist nun die Resonanz an der Bestfrequenz soweit abgeflacht, daß damit die hohe Frequenzauflösung des gesunden Ohres verloren geht. Hörschäden sind jedoch meist eine Kombination aus verschlechterten Hörschwellen (durch Schädigung der inneren Haarzellen), verringerter Dynamik- und Frequenzauflösung und Modulations- bzw. Zeitauflösung. Letzteres ist insbesondere für die Sprachperzeption von großem Nachteil, da die ursprüngliche Modulation und damit das Zeitmuster der Sprache verloren geht.

Sprachsignale erzeugen ein komplexes Erregungsmuster in den etwa 30.000 - 40.000 Nervenfasern des *nervus cochlearis*. Bis zum Eintritt der akustischen Sinneswahrnehmung in die primäre Hörrinde des Großhirns durchläuft die afferente *Hörbahn* verschiedene Stationen, in denen die Signale immer weiter verarbeitet, mit denen der anderen Kopfseite kombiniert und in wesentlichen Eigenschaften bereits ausgewertet werden (Abb. 2.9).

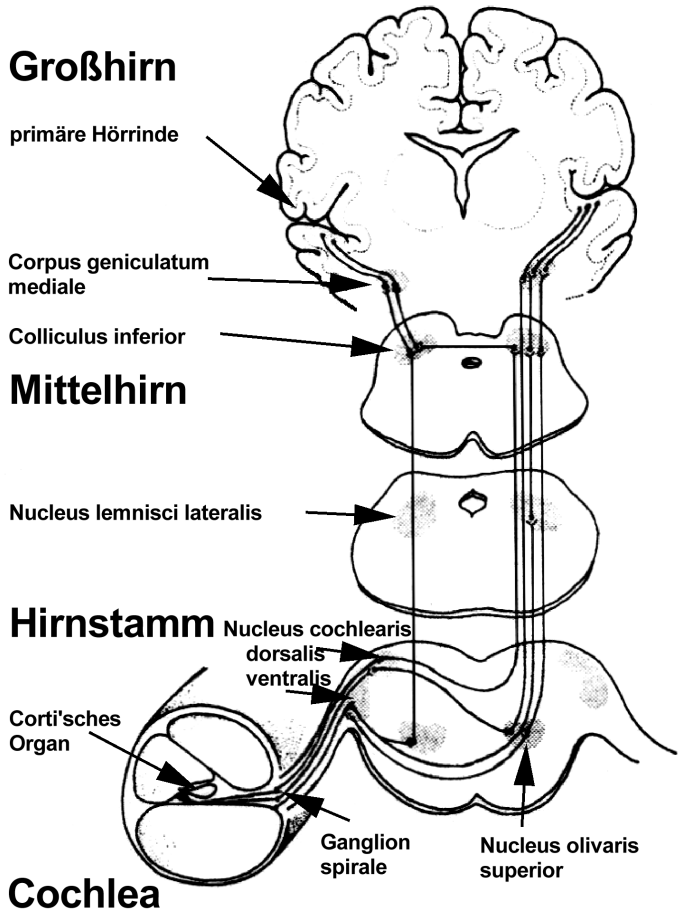


Abb. 2.9: Die *Hörbahn* umfaßt alle an der auditorischen Perzeption beteiligten neuronalen Funktionseinheiten zwischen Hörnerv und Großhirn [Oeken 1984].

Zunächst beginnt der Signalpfad in den Haarzellen und den Spiralganglien, führt dann über den *nervus cochlearis* am Ausgang des Innenohres schließlich in die höheren neuronalen Verarbeitungsstufen der Hörbahn. Der *nucleus cochlearis* beherbergt eine erste Modulationsfrequenzanalyse, d. h. Detektoren für die Amplitudenmodulationen in den Frequenzgruppen (Frequenzbänder, Verarbeitungseinheiten des Innenohres, siehe Kap. 2.2) erlauben eine separate Weiterleitung und eigenständige Auswertung der Modulationsfrequenz. Ein Abzweig in die obere Olive *nucleus olivaris superior* ermöglicht hier eine erste Bewertung der interauralen Pegel- und Phasendifferenzen, während der Hauptstrang kontralateral durch *colliculus inferior* und *colliculus geniculatum mediale* führt. Auf der Ebene des *colliculus inferior* existiert durch einen weiteren kontralateralen Abzweig eine zweite Stufe zur Schallquellenlokalisation. Der auditorische Reiz ist damit nach dem Durchlaufen von Hirnstamm und Mittelhirn am *colliculus inferior* bereits so stark vorverarbeitet, so daß eine komplexe interne Repräsentation ergänzt um Informationen zum binauralen Vergleich bzw. der Schallquellenlokalisation und Signalmodulation in den primären Hörkortex, die Hörrinde im Großhirn, geleitet werden kann. Entlang der beschriebenen Signalpfade bleibt die im Innenohr erfolgte Frequenz-Orts-Zuordnung im Prinzip erhalten, d. h. auf allen Ebenen bis hin zur Hörrinde kann eine jeweils unterschiedlich verzerrte tonotope Abbildung der Signalfrequenzen beobachtet werden: benachbarte Frequenzgruppen bleiben benachbart. Analog verhält es sich mit der Darstellung der Modulationsfrequenzen. Ausgehend vom phasensynchronen Feuern der Spiralganglien werden die Modulationsfrequenzen im Hirnstamm ausgewertet und periodotop weitergeleitet. Ähnliches gilt für die Darstellung der Einfallsrichtung bzw. das räumliche Abbild komplexer Schallsignale. Möglich durch die binaurale Schallaufnahme und richtungsabhängige Filterung des Außenohres und ergänzt um eine binaurale Frequenzgangauswertung werden auch hier die räumlichen Beziehungen verschiedener Schallquellen zueinander entlang der Hörbahn beibehalten. Die Komponenten der Reizverarbeitung sind funktional also klar verteilt und erst ihr Zusammenwirken verbunden mit der Gewinnung und Auswertung vieler Signaleigenschaften ermöglicht die Perzeption komplexer Signale, insbesondere der Sprache. Damit wird auch deutlich, daß die individuelle abstrakte Interpretation dieses mehrdimensionalen Merkmalraumes im Großhirn getrennt von der heute weitgehend beschreibbaren Vorverarbeitung erfolgt.

2.2 Die perzeptiven Eigenschaften des Gehörs

Im Laufe der Signalgewinnung und Verarbeitung nimmt das menschliche Gehör die meßbaren physikalischen Größen wie die Amplitude, die Frequenz, die Modulationsfrequenz und die interaurale Phasendifferenz (Einfallswinkel) auf nichtlinearen Skalen wahr. Man spricht in diesem Zusammenhang von Empfindungsgrößen, deren Abbildungsfunktion zu einer Verzerrung des Merkmalraumes gegenüber der physikalischen Umwelt führt.

Eine wichtige physikalische Meßgröße ist der Schalldruck p [Pa] bzw. die Schallintensität I [W/m^2]. Gleichung Gl. (2.1) beschreibt den quadratischen Zusammenhang zwischen dem Effektivwert des Schalldrucks p und der Schallintensität I.

$$I = \frac{p^2}{\rho \cdot c} \qquad \text{Gl. (2.1)}$$

Das Produkt aus Mediendichte ρ und Schallgeschwindigkeit c steht für den Schallwellenwiderstand bzw. die Schallkennimpedanz. Die Pegellautstärke L (*sound pressure level - SPL*) wird in Dezibel (dB) gemessen. 0 dB SPL entspricht einem Schalldruck von 20 µPa am Trommelfell, also etwa dem 2 * 10^{-10} fachen des normalen atmosphärischen Drucks (101,3 kPa).

Bereits frühzeitig wurden die Kurven gleicher Pegellautstärken, die *Isophonen*, bestimmt [Fletcher und Munson 1933] (siehe Abb. 2.10).

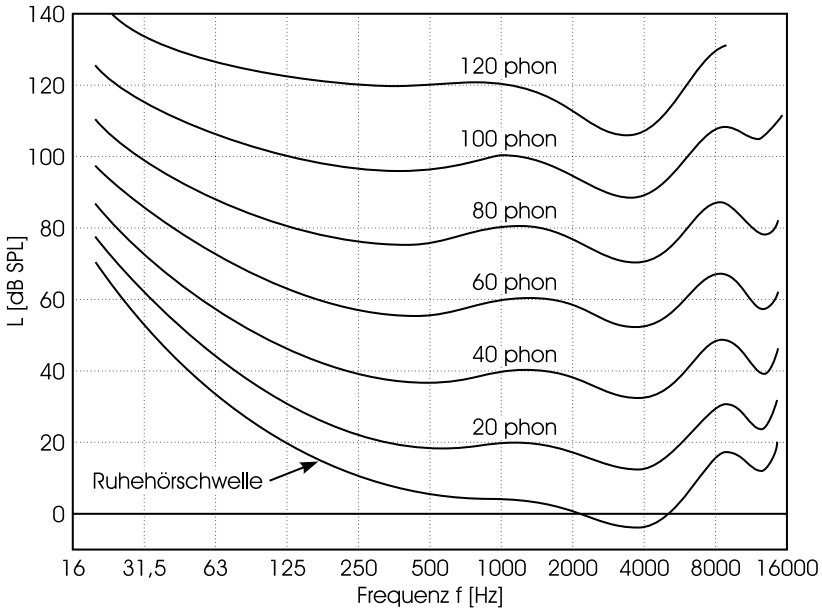


Abb. 2.10: *Isophonen*, Kurven gleicher Lautheit. Die Referenzfrequenz liegt dafür bei 1.000 Hz, d. h. der Pegel eines 31,5 Hz-Signals müßte für eine gleiche Lautstärke von z. B. 60 phon (bei 1.000 Hz) um etwa 30 dB verstärkt werden [Kollmeier 1999].

Die *phon*-Skala entspricht dem Schalldruckpegel (SPL) bei 1.000 Hz. Entlang einer Isophone wird subjektiv die gleiche Lautheit verspürt.

Das Darstellung in Abb. 2.11 zeigt, daß die Empfindlichkeit des Ohres stark frequenzabhängig und im Bereich von 500 - 4.000 Hz am höchsten ist. Hier besitzt das Ohr auch den größten Dynamikumfang, der im übrigen ebenfalls frequenzabhängig ist.

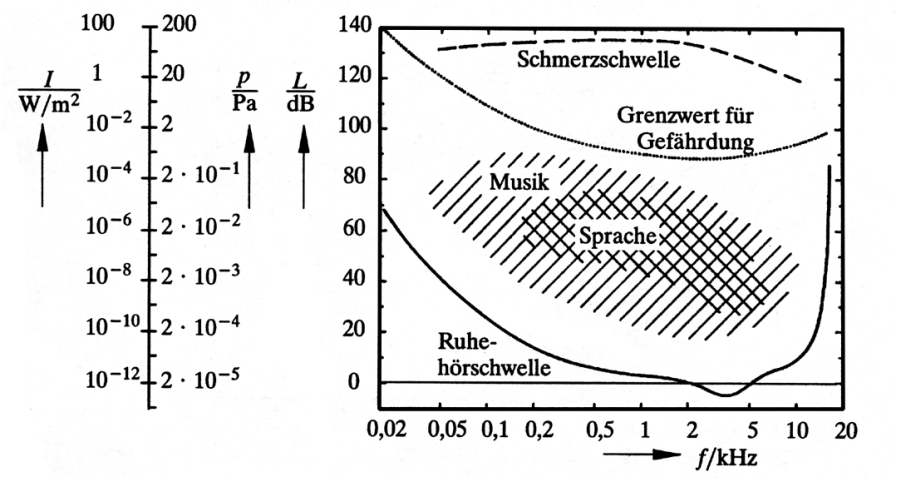


Abb. 2.11: Dynamik- und Frequenzumfang des menschlichen Gehörs. Die Schallintensität *I* [W/m^2] steht mit dem Quadrat des Schalldruckes *p* [Pa] und dem Schalldruckpegel *L* [dB] in Beziehung [Zwicker 1982].

Zwischen Ruhehörschwelle (0 dB SPL) und Schmerzgrenze (etwa 120 dB SPL) liegen sechs Zehnerpotenzen des Schalldruckes *p* (siehe Abb. 2.11), was 12 Zehnerpotenzen der Schallintensität *I* entspricht. Der für die Laute der Sprache interessante Bereich liegt zwischen etwa 150 und 8.000 Hz. Die Hauptenergie für stimmhafte Laute (Vokale) ist insbesondere in drei Formanten[2] enthalten, die (individuell unterschiedlich) etwa im Bereich zwischen 200 und 3.000 Hz liegen. Stimmlose[3] oder transiente[4] Laute der Konsonanten besitzen dagegen auch Anteile in höheren Frequenzbereichen. Sie haben für die Sprachperzeption eine bedeutende Funktion, da sie die Silben- und Wörtertrennung erkennbar machen.

Die Empfindungsgröße Lautheit, gemessen in *sone*, ist auf den Isophonen also konstant und folgt dem psychophysischen Potenzgesetz für Intensitätsempfindungen Gl. (2.2) von Stevens [Stevens 1975]. Die Bezugsschallintensität I_0 liegt bei 0 dB SPL (10^{-12}W/m^2).

$$N[sone] = \left(\frac{I}{I_0}\right)^n \qquad \text{Gl. (2.2)}$$

[2] Formanten: Hauptresonanzfrequenzen des Nasen-Rachenraumes bei der Erzeugung einzelner Vokale, z. B. können die Mittenfrequenzen F1=600 Hz, F2=1.050 Hz und F3=2.600 Hz die Formanten für den Vokal „a“ eines männlichen Sprechers sein (siehe auch Abb. 3.14).

[3] Frikative: entstehen durch Reibe- oder Friktionsgeräusche (Rauschen) hervorgerufen durch turbulente Luftströmungen im Spracherzeugungstrakt.

[4] Plosive: Geräusch beim plötzlichen Abbau von Über- oder Unterdrücken nach temporärer Verschlußbildung im Spracherzeugungstrakt.

Eine Verdopplung der Lautheit bedarf demnach einer Erhöhung der Schallintensität *I* um 10 dB, wobei ein Exponent von n ≈ 0,3 gilt. Da die Schallintensität *I* vom Quadrat des Schalldruckes abhängt, gilt für das ebenfalls mögliche Verhältnis von Schalldrücken der Exponent n ≈ 0,6. Ein *sone* entspricht bei einen 1.000 Hz Ton einem Pegel von 40 dB SPL; 50 dB entsprechen 2 *sone*, etc. Den ursprünglich in [Zwicker 1982] dargestellten Zusammenhang zwischen Schalldruck L [dB] und Lautheit N [sone] zeigt Abb. 2.12.

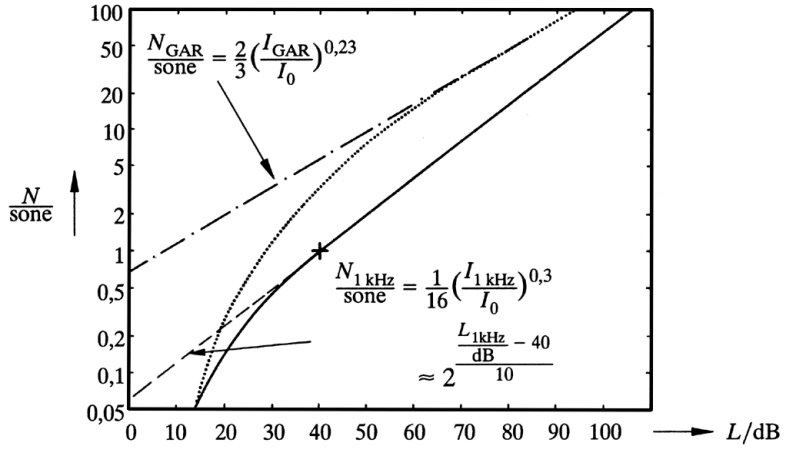


Abb. 2.12: Lautheit N_{1kHz} für einen 1.000 Hz Ton und N_{GAR} für gleichmäßig anregendes Rauschen, nach [Zwicker 1982] und [Vary et al. 1998].

Auch die Tonheit *z* bzw. die empfundene Verhältnistonhöhe H_V (Abb. 2.13) hängen nichtlinear von der tatsächlichen Frequenz ab.

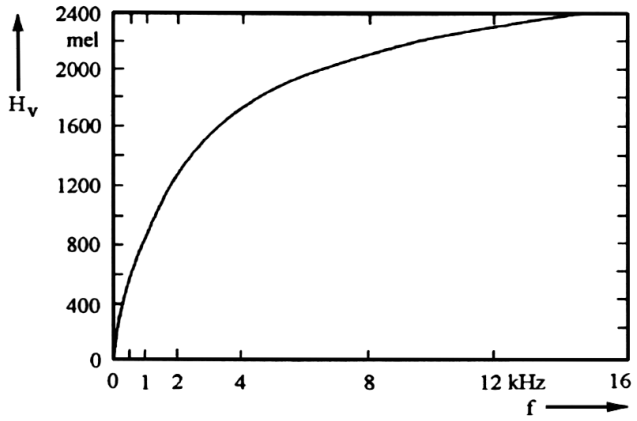


Abb. 2.13: Die Verhältnistonhöhe H_V wird in *mel (melody)* gemessen. Ihre lineare Skala wird hier gegen die logarithmische Tonfrequenz-Skala abgetragen [Zwicker 1982].

Eine verdoppelte Verhältnistonhöhe entspricht einer verdoppelten Frequenz nur bis etwa 500 Hz. Für höhere Frequenzen geht diese Proportionalität verloren. Die Verhältnistonhöhe

steht in engem Zusammenhang zur Tonheit *z* gemessen in *Bark*. Eine Verhältnistonhöhe von 100 mel entspricht 1 Bark. Damit ist der Frequenzbereich bis 16 kHz in 24 Bark unterteilt. Abbildung 2.14 zeigt die Zusammenhänge zwischen der Frequenz *f*, der Tonheit *z* und der Frequenzgruppenbreite Δf_G.

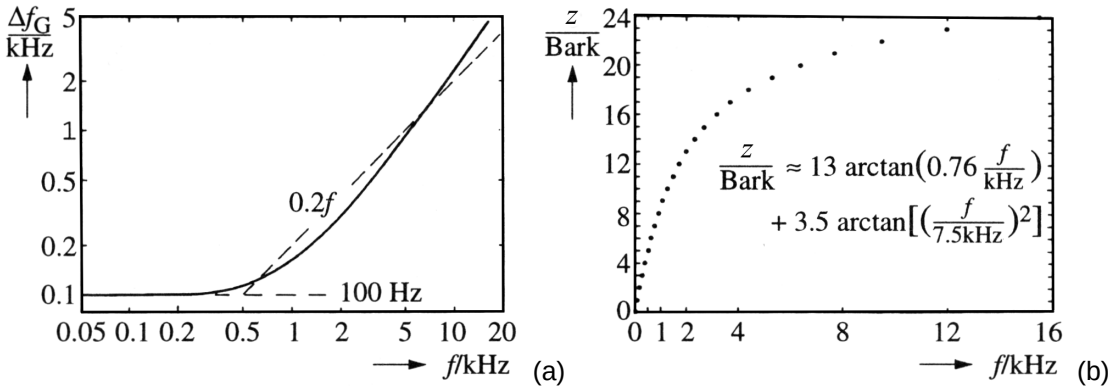


Abb. 2.14: Die Frequenzgruppenbreite Δf_G (in Hertz) bleibt bis 500 Hz etwa konstant bei 100 Hz und nimmt dann entlang der logarithmischen Achsen annähernd konstant mit 0,2 der Mittenfrequenz zu (a) und die Zuordnung zwischen Frequenz und Tonheit *z* (b), nach [Zwicker 1982] und [Vary et al. 1998].

Diese 24 Abschnitte des hörbaren Bereichs entsprechen 24 Frequenzgruppen des auditorischen Systems. Die psychoakustische Relevanz dieser Frequenzbänder ist in der frequenzgruppenweisen Zerlegung, Verarbeitung und Abbildung des Schallsignals (insbesondere der hervorgerufenen Lautheit) begründet, d. h. die wahrgenommene Lautheit ist von der Bandbreite des Reizes abhängig. Zum Beispiel stellt sich die Summenlautheit zweier gleich lauter Schallsignale je nach Frequenzgruppendistanz anders ein: liegen die Töne nicht in der gleichen Gruppe, addiert sich deren (Teil-)Lautheit; andernfalls addiert sich nur die Intensität, d. h. beide Töne zugleich angeboten entsprechen einem in seiner Schallintensität um 3 dB verstärkten Einzelton. Verwendet man Rauschsignale variabler Bandbreite, aber gleicher Leistung, kann damit die Breite der Frequenzgruppen bestimmt werden: sobald bei zunehmender Bandbreite des Rauschens eine höhere Lautheit wahrgenommen wird, ist die Grenze der lokalen Frequenzgruppe erreicht.

Das Lautheitsmodell nach [Zwicker 1982] beschreibt, wie einzelne Spektralanteile eines komplexen stationären Schallsignals zur Summenlautheit beitragen, bzw. wie der Erregungspegel in einer Frequenzgruppe die Wahrnehmung eines Teilsignals in einer benachbarten Frequenzgruppe beeinflußt. Ist der Erregungspegel nämlich ausreichend hoch, können Signale der Nachbargruppe für die Wahrnehmung unterdrückt bzw. ganz ausmaskiert werden. Diese Frequenzmaskierung wird z. B. für Datenreduktionsverfahren (*MPEG*[5] *Layer 3* - Komprimierungsstandard) ausgenutzt.

[5] MPEG - *Moving Pictures Expert Group*.

Da die Spektralzerlegung entlang der Basilarmembran nach der Frequenz-Orts-Theorie erfolgt, können an dieser Strecke die Skalen der Tonheit *z*, der Verhältnistonhöhe H_V und der Frequenz *f* abgetragen werden (Abb. 2.15).

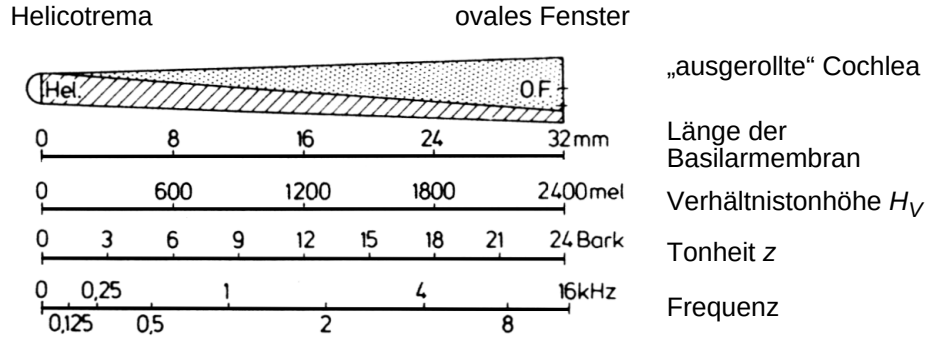


Abb. 2.15: Zuordnung zwischen der Frequenz und dem Ort auf der Basilarmembran bzw. den Tonskalen des Innenohres, nach [Zwicker 1982].

Die Frequenzauflösung ist jedoch wesentlich feiner möglich, als es die Verarbeitung in 24 Frequenzbändern erlaubt. Unterhalb von 500 Hz können Töne im Abstand von 3 Hz unterschieden werden; über 1.000 Hz verringert sich die Auflösung zunehmend mit 0,6 % der jeweils betrachteten Frequenz. Dabei nutzt das auditorische System aus, daß ein einzelner Ton zwar ein Erregungsmaximum an einem Ort erzeugt, entsprechend der Abstimmkurve aber eine ganze Region anregt. Für die Frequenzdiskrimination wird dabei die Translation dieser Region insgesamt um einen verschobenen Ton ausgewertet.

Eine für die Sprachwahrnehmung wesentliche Eigenschaft ist das in den neuronalen Erregungsmustern sichtbare Zeitverhalten des auditorischen Systems. Dadurch bedingt existiert ein zeitliches Auflösungsvermögen, was um die Fähigkeit der Auswertung von Signalmodulationen erweitert ist. Wird z. B. ein rechteckförmiges Rauschsignal (Maskierer, siehe Abb. 2.16) angeboten, verdeckt es in diesem Fall während der Simultanverdeckung einen Ton bis zu einem Pegel von 50 dB.

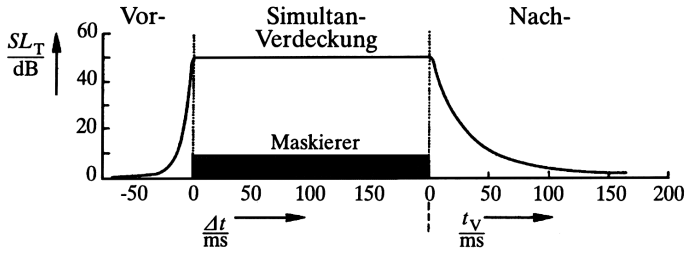


Abb. 2.16: Zeitliche Vor-, Simultan- und Nachverdeckung (Nachverdeckungszeit t_V) von Testtönen mit einem Pegel SL_T über der Ruhehörschwelle durch einen Rauschmaskierer der Länge Δt, nach [Vary et al. 1998, Zwicker 1982].

Aber danach und bemerkenswerterweise auch davor würde dieser Ton ebenfalls maskiert bzw. zeitlich verschmiert, solange sein Pegel unter der Verdeckungskurve bleibt. Wird also ein Testton angeboten, benötigt das Ohr eine kurze Zeit, um eine Hörempfindung aufzubauen. Die Mit-

hörschwelle wird deswegen von einer zeitlich später einsetzenden aber stärkeren Hörempfindung (verursacht durch den Maskierer) erhöht bzw. zeitlich verschmiert. Die Vorverdeckungsphase ist (individuell sehr unterschiedlich) etwa 10 bis 20 ms lang. Sehr viel ausgeprägter ist dagegen die Nachverdeckung: hier wird die Ruhehörschwelle erst nach etwa 200 ms wieder erreicht. Die Steilheit der Nachverdeckungskurve ist dabei stark abhängig von der Maskiererdauer. Diese Vorgänge verdeutlichen die zeitlichen Adaptationsvorgänge im auditorischen System, und die Trägheit des Auf- und Abbaus von Erregungsmustern. Damit im Zusammenhang steht die Detektion von Modulationsfrequenzen des Schallsignals. Hohe Modulationsfrequenzen sind infolge des Nachverdeckungseffektes schwerer wahrzunehmen; das Minimum der psychoakustischen Modulationstransfer-Funktion, welche die Abhängigkeit des ebenmerklichen Modulationsgrades von der Modulationsfrequenz beschreibt, liegt bei 4 Hz. Hier können die kleinsten Schwankungen wahrgenommen werden, was ideal der Sprachsilbenfrequenz angepaßt ist. Bis etwa 10 Hz werden deutliche Intensitätsschwankungen, bei höheren Modulationsfrequenzen die Rauhigkeit des Schalles wahrgenommen.

Mit Modellen des auditorischen Systems können heute wesentliche Eigenschaften der Ton- und Sprachverarbeitung beschrieben und vorhergesagt werden. Die Abb. 2.17 zeigt eine modellhafte Zerlegung des Sprachsignals „Elektroakustik“, realisiert für 23 Tonheiten *z* (Frequenzgruppen 1 bis 23).

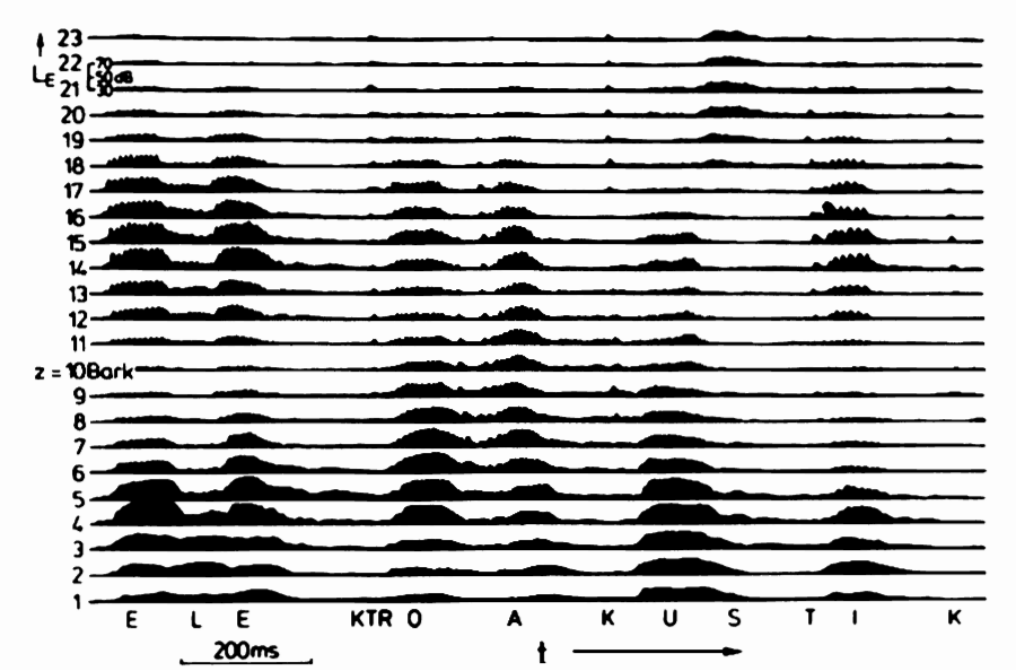


Abb. 2.17: „Gehörgerechtes" Spektrogramm für das Wort Elektroakustik aus [Zwicker 1982]. Dargestellt sind die Erregungspegel L_E des Modells, womit die Aktivität in den entsprechenden Frequenzgruppen veranschaulicht wird. Gut zu erkennen ist der bis zu drei Formanten umfassende Spektralgehalt der Vokale. Konsonanten enthalten dagegen nur eine geringe Signalenergie, erlauben aber die Trennung von Silben und Wörtern. Die Grundfrequenz des männlichen Sprechers von etwa 100 Hz überträgt sich in Form der gezackten Hüllkurve einzelner Erregungsmaxima und demonstriert die Phasensynchronisation der Feuerrate.

Kapitel 3

Modelle des Auditorischen Systems

Um Modelle des auditorischen Systems bemüht sich die Forschung bereits seit den 20iger Jahren [von Bekesy 1928, von Bekesy 1953]. Dabei bestimmten die jeweils technischen Möglichkeiten Vorgehensweise und Modellierungstechnik. Diese Modelle unterscheiden sich nach den Aspekten des auditorischen Systems, die nachgebildet werden sollen. Die Abbildung 3.1 zeigt mögliche Abstraktionsebenen und deren (physiologische) Zuordnung. Ein umfassendes Modell der auditorischen Perzeption, d. h. der vollständigen Sensorik, Geräusch-, Ton- und Sprachperzeption einschließlich darauf aufsetzender Leistungen, wie Lokalisation, Orientierung und Sprachverarbeitung, existiert aufgrund der Komplexität und der noch nicht abgeschlossenen Erforschung nicht. Statt dessen beschränkt man sich auf Teilmodelle und -aspekte.

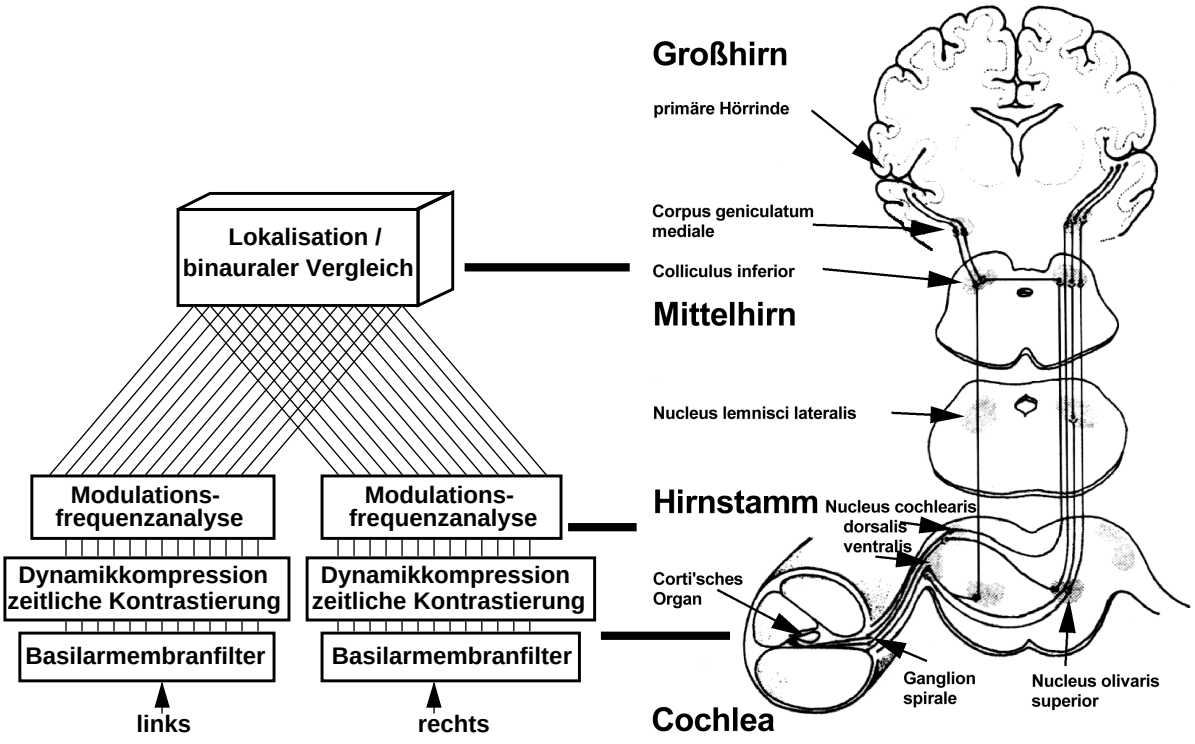


Abb. 3.1: (Physiologische) Zuordnung der Hörbahn nach [Oeken 1984] zu wesentlichen Modellebenen [Kollmeier 1999].

In einigen Modellen wird die Generierung von Erregungsmustern in auditorischen Nerven und verschiedenen Stationen der Hörbahn detailliert beschrieben, während andere Ansätze die effektive Signalverarbeitung bevorzugen und deren Umfang der Funktion des Gesamtsystems im gewünschten Anwendungskontext unterordnen. Die in technischen Systemen realisierten Prinzipien der auditorischen Signalverarbeitung orientieren sich an Modellen des Hörvorgangs, die

größtenteils auf physiologischen und psychoakustischen Ansätzen und Daten beruhen. Die bislang erarbeiteten Entwicklungen verfolgen je nach Modellierungstechnologie, Modellierungstiefe und Anwendungen sehr unterschiedliche Strategien. Mechanische, mechano-optische und mikromechanische Modelle nehmen dabei eher eine Außenseiterrolle ein, während es eine Vielzahl von (digitalen) Computer-Simulationsprogrammen aber auch analogen Schaltungsentwürfen gibt. Letztere werden meist direkt in einer Analog-VLSI[6]-Technologie realisiert bzw. angewendet und haben aufgrund ihres Prinzips[7] sehr gute Modellierungseigenschaften für die abgebildeten akustisch-neuronalen Wandler und neuronalen Verarbeitungsstrukturen. Da gewöhnlich alle Zweige des Modells als Schaltungsteile realisiert werden und demnach ein hochgradig paralleles System entsteht, sind sie apriori echtzeitfähig. Andererseits besteht aber auch eine signifikante Anfälligkeit für allgemeine Schwächen analoger Schaltungen, wie Parameterfluktuationen, Temperaturabhängigkeiten etc. Mit der breiten Verfügbarkeit von programmierbarer digitaler Rechentechnik, insbesondere digitaler Signalprozessoren (DSP[8]), sind eine ganze Reihe verschiedener Ansätze zur digitalen Modellierung erarbeitet worden. Durch die Komplexität der abgebildeten Algorithmen aber auch die Flexibilität der Programme werden diese Modelle meist als Laboraufbauten im PC-Maßstab realisiert (Filterbänke bzw. Transformationen, wie FFT[9]). Kostspielige digitale Hardware-Realisierungen sind eher selten und rechtfertigen sich nur für bestimmte und je nach Anwendung möglichst einfach gehaltene Varianten. Die Breite möglicher Applikationen reicht allgemein von einer detaillierten und möglichst genauen Modellierung für Grundlagenuntersuchungen, bis hin zum technischen Einsatz angepaßter und minimierter Teilmodelle. Für die Simulationen realer Vorgänge im auditorischen System, wie Sprach- und Tonperzeption, wird ganz allgemein eine möglichst exakte Modellierung der psychoakustischen Prinzipien für Signalwandlung und -aufbereitung gesucht. Für den Einsatz in technischen Anwendungen, welche die Verarbeitung im menschlichen auditorischen System zur maschinellen Spracherkennung, Szenenanalyse und akustischen Orientierung, Sprachdatenkomprimierung und instrumentellen Qualitätsbewertung sprachverarbeitender Systeme nachbilden oder im Falle pathologischer Veränderungen korrigieren oder ersetzen sollen (Hörgeräte und Cochlea-Implantate), muß die Komplexität des Modells der Applikation also der Realisierbarkeit untergeordnet werden. Wird eine Kombination aus Echtzeitverarbeitung und hohem Modellierungsgrad benötigt, kommen die Analog-VLSI-Systeme in Betracht. Spielen allerdings Flexibilität, Stabilität und Verfügbarkeit eine große Rolle, und ist ein Zuschnitt bzw. eine Vereinfachung des Modells möglich, empfiehlt sich die Software-Lösung. Da digitale Rechnermodelle auch dann noch sehr lange Programmlaufzeiten verursachen können, ist mit einer Hardware-Umsetzung der Algorithmen eine weitere Beschleunigung, Miniaturisierung und letztendlich industrieller Einsatz möglich. Im folgenden sollen einige aktuelle und spezifische Ansätze bzw. Applikationen vorgestellt und bewertet werden.

---

[6] VLSI - *Very Large Scale Integration*.

[7] Aktive analoge Filter und direktes Abbilden von Zeitfunktionen durch Auf- und- Umladen von Speichern (Widerstands-Kondensator-Netzwerke mit aktiven Verstärkern).

[8] DSP - *Digital Signal Processor.*

[9] FFT - *Fast Fourier Transformatio*n.

3.1 Mikromechanische Modelle

Bei den mechanischen Modellen wird insbesondere das Schwingungs- und Resonanzverhalten und die Frequenzdispersion an der Basilarmembran bzw. des Corti'schen Organs untersucht, indem dünne, elastische Membranen (gekoppelte Feder-Masse-Systeme) in Spannvorrichtungen angeregt und dabei abgetastet werden. In [Zhou et al. 1993] wird ein solcher Aufbau für Laser-Interferenzmessungen an einer künstlichen Cochlea (zwei gekoppelte flüssigkeitsgefüllte Skalen mit einer Polymermembran) vorgestellt. Ein mikromechanisches Modell aus einer 7 µm starken Silizium-Schicht wird in [Tanaka et al. 1998] veröffentlicht. Hier wird das Schwingungsverhalten der Basilarmembran mit einer Fischgrätenstruktur aus unterschiedlich langen Resonatoren näherungsweise nachgebildet, um ein „künstliches Cochlea-Mikrofon" zur Verfügung zu stellen (siehe Abb. 3.2 a). Das elektrische Ersatzschaltbild (Abb. 3.2 b) stellt den Bezug zu den *Transmission-Line* Modellen von [Zwislocki 1948] und [von Bekesy 1960] her, mit denen sowohl die Resonanzbildung als auch die Wanderwellenausbreitung entlang der Cochlea berücksichtigt wurde. Setzt man diese Fischgrätenstruktur einem Schallfeld aus, kann sowohl dessen Spektralgehalt als auch die Erregungsausbreitung an den Schwingungsamplituden der Resonatoren modelliert werden. Aufgrund der sehr aufwendigen Herstellungsverfahren erlangten derartige mechanische Modelle bislang keine große Verbreitung.

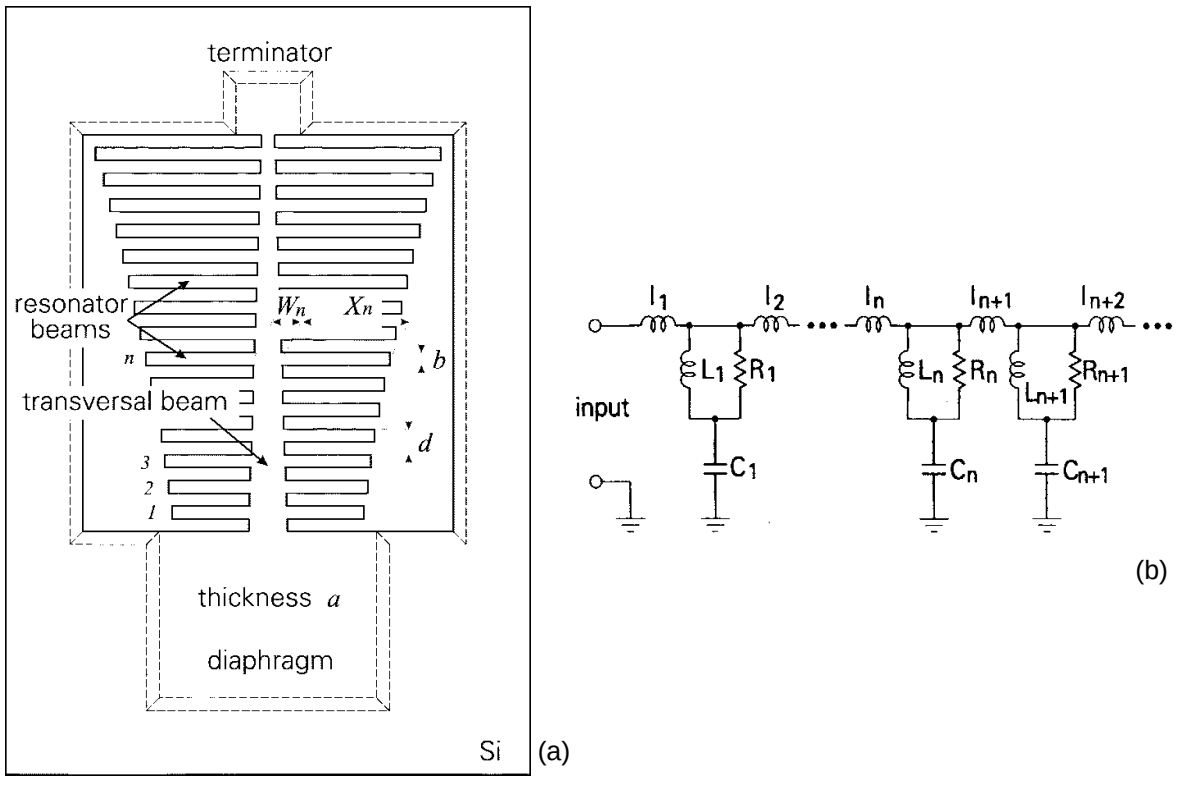
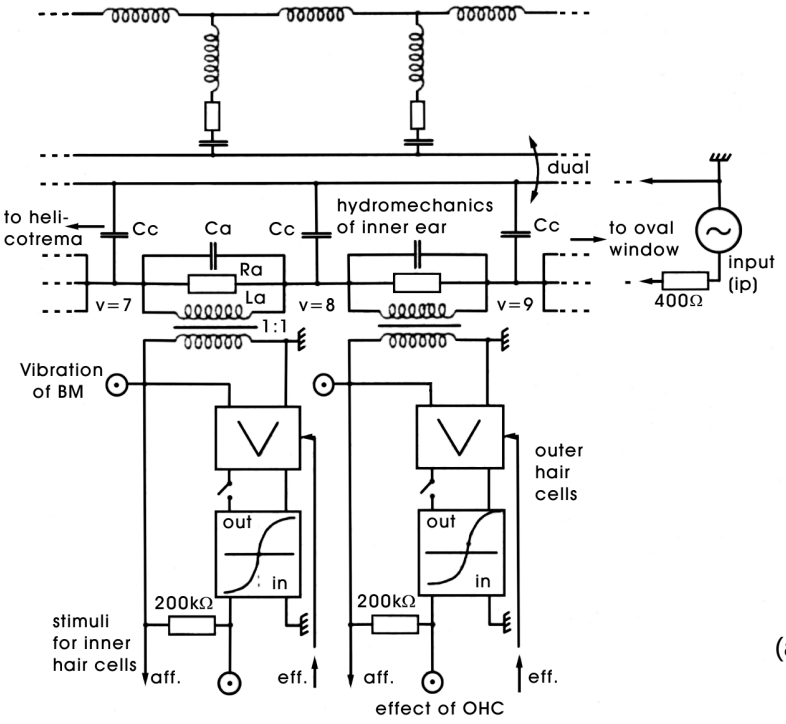


Abb. 3.2: Silizium-Resonator in Fischgrätenstruktur (a) und elektrisches Ersatzschaltbild (b) nach [Tanaka et al. 1998]. Die Mikroresonatoren wurden mit Hilfe von FEM-Berechnungen[10] dimensioniert.

[10] FEM - *Finite Element Modeling.*

3.2 Analoge elektronische Modelle

Das bereits angedeutete analoge Ersatzschaltbild der „Fischgräten-Cochlea“ legt es nahe, direkte analoge Modelle des peripheren auditorischen Systems zu entwerfen. Ein umfangreicher Filter-Ansatz mit nichtlinearen Verstärkern wird bei [Zwicker 1986] als eine diskrete Schaltung beschrieben. Dabei wurde eine Kaskade von 90 Sektionen bestehend aus passiven resonanten RCL-Kreisen mit nachgeschalteten gesteuerten Verstärkern aufgebaut (siehe Abb. 3.3).



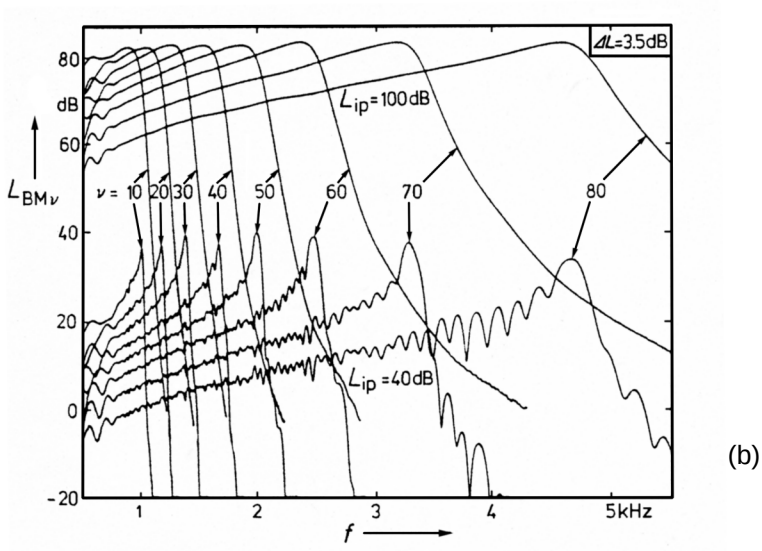


Abb. 3.3: Teile des Kaskadenmodell des Innenohres mit analogen Filtern (a) und damit aufgenommene Abstimmkurven (b) der Sektionen ν = 10...80 bei zwei Eingangspegeln L_{ip} (40 u. 100 dB SPL) nach [Zwicker 1986].

Die RCL-Komponenten sind so dimensioniert, daß sich eine mit der Sektion wachsende Resonanzfrequenz ergibt und so die Frequenzzerlegung des Basilarmembransystems abgebildet werden kann. Der davon abgeleitete afferente Pfad zu den sensorischen inneren Haarzellen (nicht Bestandteil des Modells) wird durch nichtlineare Rückkopplung (gesteuerter Verstärker) beeinflußt, womit das Wirken der äußeren Haarzellen, also die pegelabhängige Hemmung der Basilarmembranschwingung, im Modellausgang berücksichtigt werden kann.

Unter Beibehaltung des analogen Ansatzes wurde vor allem mit den Arbeiten von [Mead 1989] zur Abbildung neuronaler Systeme in hochintegrierte analoge Schaltungen (Analog-VLSI) eine qualitativ neue Stufe erreicht. Die erste Analog-VLSI-Cochlea wurde von [Lyon und Mead 1988] (siehe Abb. 3.4) beschrieben. In [Lazarro und Mead 1989] wurden weiterführende Arbeiten zum Ausbau des Modells veröffentlicht.

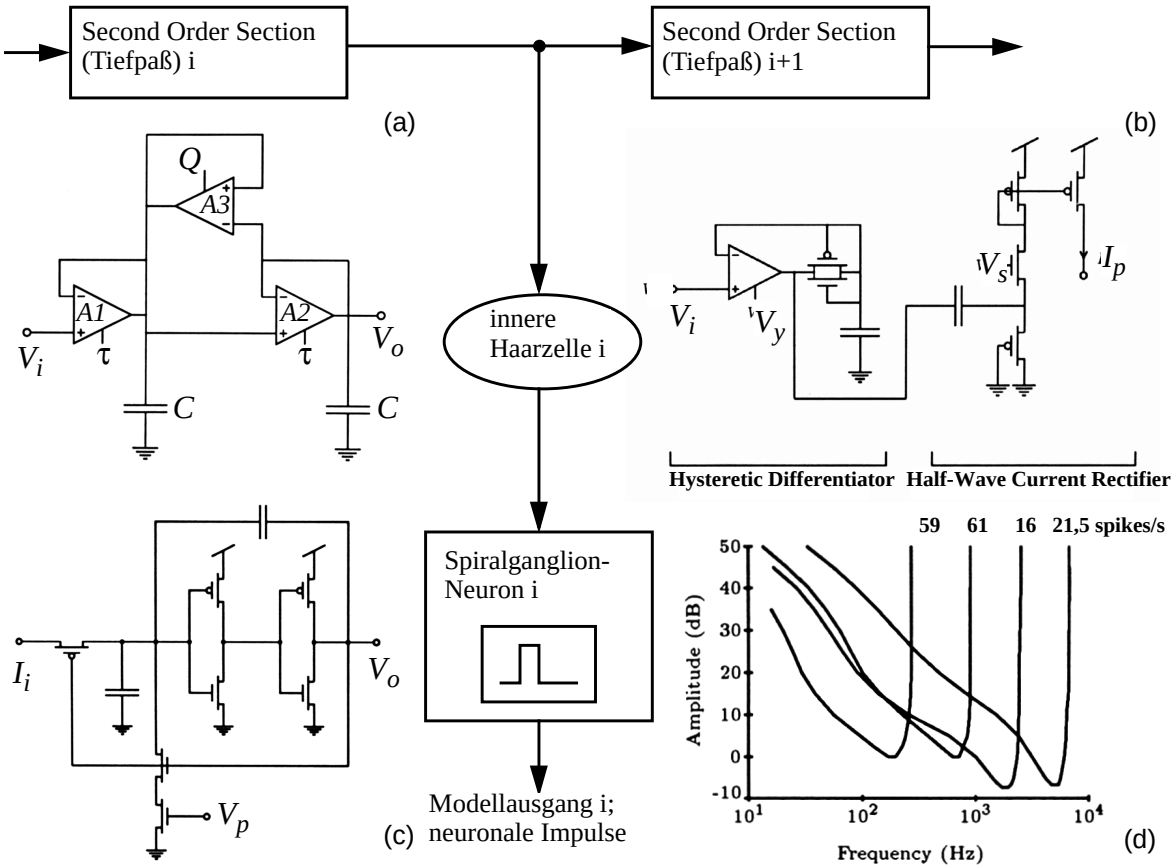


Abb. 3.4: *Silicon Cochlea* nach [Lyon und Mead 1988] und [Lazarro und Mead 1989] Innenschaltung des zweistufigen Tiefpasses der Filterkette (a); der inneren Haarzelle (Differentiation und Halbwellengleichrichtung) (b); des Spiralganglion-Neurons (c) zur Generierung einer stromproportionalen Impulsfolge. Die Intensitätskurven in (d) bilden für vier Modellausgänge (Frequenzkanäle) die Pegel ab, die zur Erzeugung der jeweilig konstanten Impulsrate *(spikes per second)* erforderlich sind.

Das *Transmission-Line*-Modell der Basilarmembran wurde mit 480 Tiefpaß-Filtern 2ter Ordnung (Abb. 3.4 a) realisiert, deren Knickfrequenzen äquidistant auf einer logarithmischen Skala bezüglich der Position in der Kette eingestellt sind. Grundbausteine sind justierbare Trans-

konduktanzverstärker in Spannungsfolgerschaltung. Mit Bias-Strömen am Anschluß τ können die Zeitkonstanten der Integrierglieder (*A1* und *A2*, Kondensator C ist identisch) und damit die Knickfrequenzen dieser Tiefpässe erster Ordnung eingestellt werden. Der Bias-Strom *Q* in *A3* bestimmt die Güte des zweistufigen Filters. Die Ausgangsspannung jeder Stufe wird sowohl an die folgenden Filter als auch an ein Modell der inneren Haarzellen (Abb. 3.4 b) weitergegeben. Das Differenzierglied berechnet (justiert durch V_y) die zeitliche Ableitung der Eingangsspannung und erzeugt eine zusätzlich logarithmisch komprimierte Ausgangsspannung. Diese wird nachfolgend in einen (halbwellen-)gleichgerichteten Strom I_p umgewandelt. Diese Signaltransformation entspricht der Abbildung der eindirektionalen Beschleunigung der Basilarmembran-Bewegung. Den Ausgang der Haarzelle wandelt ein Modell des Spiral-Ganglions (erste Ebene des Hörnervs) in eine Folge von neuronalen Entladungsimpulsen einstellbarer Breite. Diese Schaltung (Abb. 3.4 c) erzeugt aus dem Ausgangsstrom des Gleichrichters Pulse mit maximaler Amplitude, deren Breite mit der Steuerspannung V_P eingestellt wird. Damit erscheint am Modellausgang eine aktivitätsproportionale neuronale Impulsentladungsrate, welche die gewünschte Form der Abstimmkurven näherungsweise wiedergibt (Abb. 3.4 d).

Mit der Kopplung zweier 56-stufiger analoger Cochlea-Modelle wurde von [Mead et al. 1991] der Aufbau eines echtzeitfähigen binauralen Verarbeitungsmodells möglich. Hierbei wird eine Matrix von 3.136 Vier-Quadrant-Korrelatoren angesteuert. Die Abbildung 3.5 zeigt die damit mögliche Darstellung von Frequenz-Orts-Zeitmustern binauraler akustischer Erregungen.

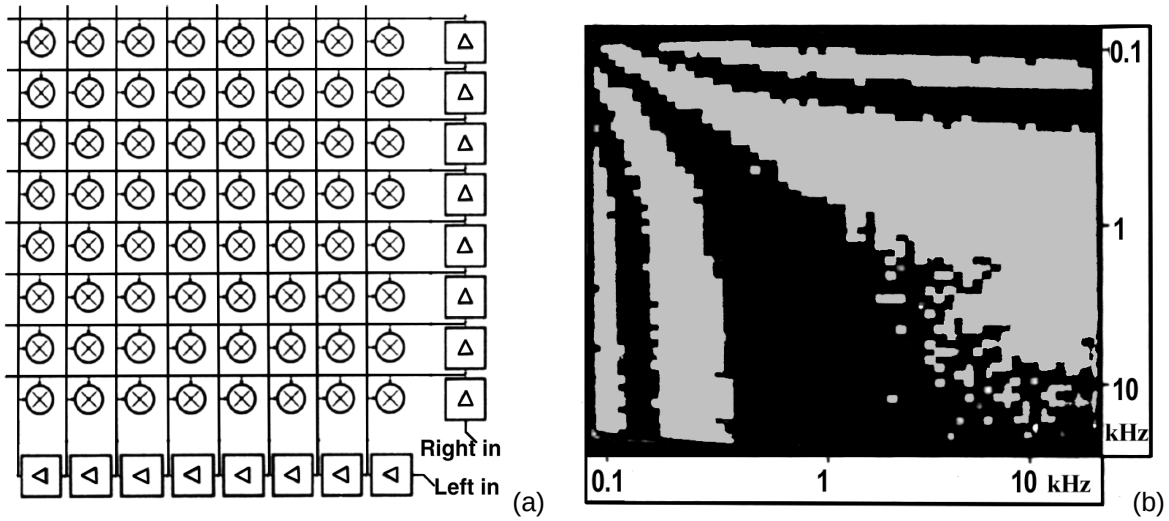


Abb. 3.5: „Stereusis"-Chip (a) mit zwei *Silicon Cochleas* (Verstärkerketten) und Korrelatoren (⊗) und die binaurale Ausbreitung des Erregungsmusters (b) für einen 100 Hz Ton bei einer interauralen Phasendifferenz von π/2 nach [Mead et al. 1991].

Für die Analog-VLSI Modelltechnik ist das physiologieorientierte Prinzip der Übertragungskette aus Tiefpässen mit „seitlichen" Abzweigungen für Haarzell- und Adaptationsschaltungen bestimmend geblieben. Die dargestellten Grundlagen wurden hinsichtlich praktischer Anwendungsfälle wie Spracherkennung, auditorische Szenenanalyse und Cochlea-Implantate angepaßt und ausgebaut.

Die *Silicon Cochlea* von [van Schaik et al. 1996a, van Schaik et al. 1997b] ist eine verbesserte Version der Entwürfe von [Lyon und Mead 1988] und [Watts et al. 1992], da sie die Abstimmung der Tiefpaßkette mit einem speziellen CMOS[11]-kompatiblen CLBT[12] Transistor ermöglicht. Diese als Stromquelle geschalteten Transistoren steuern über eine Widerstandskette die Bias-Ströme in den Filtern und skalieren damit deren Frequenzgänge entsprechend den gewünschten neuronalen Abstimmkurven (Abb. 3.6).

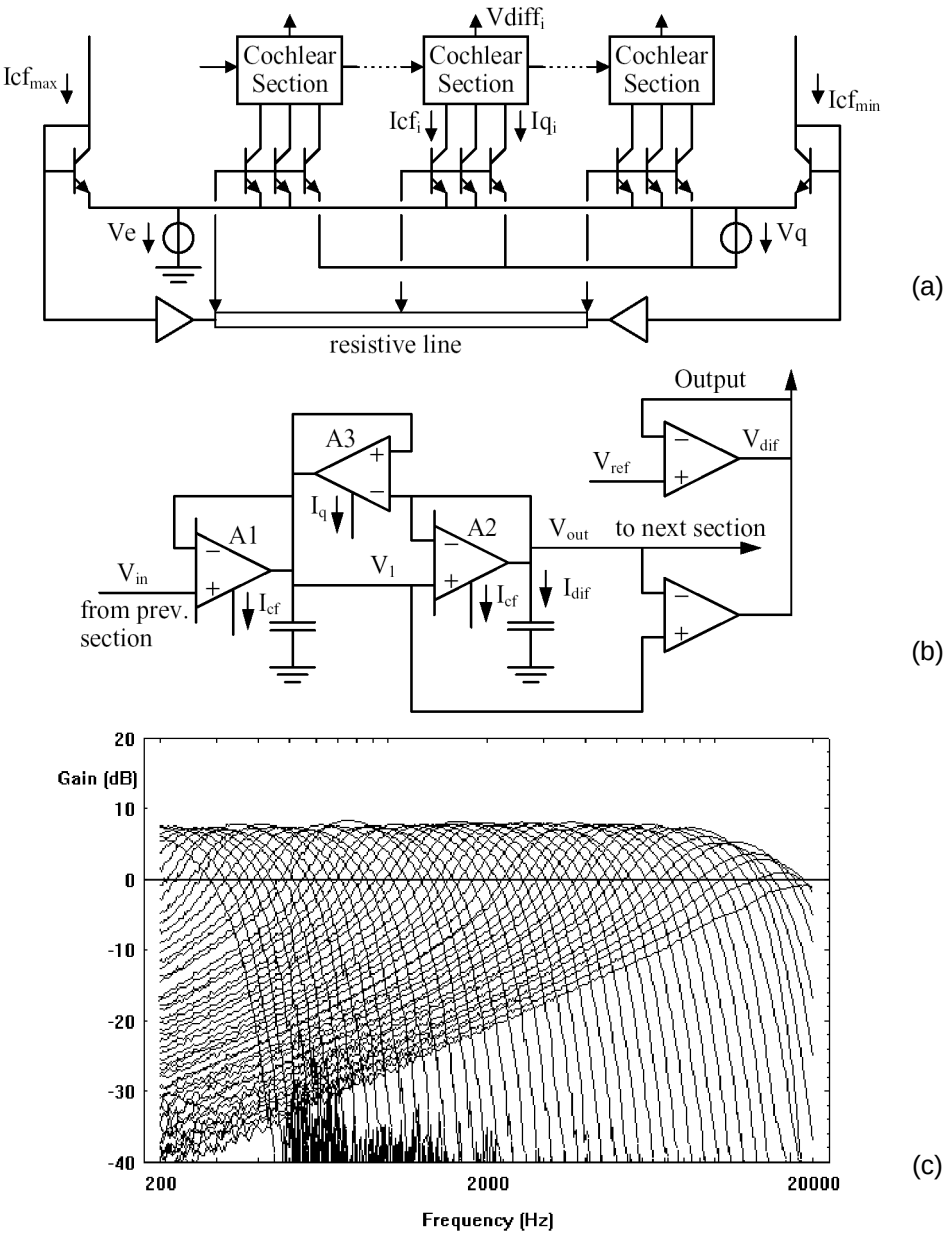


Abb. 3.6: Analoge Cochlea von [van Schaik et al. 1996a]. Das Frequenzverhalten der *Second Order Sections* (b) bestimmen Bias-Ströme, die durch CLBT-Transistoren an einer Widerstandskette für die gesamte Filterbank erzeugt werden (a). Die Abstimmkurven der *Silicon Cochlea* erhalten damit eine höhere Regularität (c) und Temperaturstabilität.

[11]CMOS - *Complementary Metal Oxide Semiconductor.*

[12]CLBT - *Compatible Lateral Bipolar Transistor.*

Der in ECPD15[13] (1,5 µm CMOS) gefertigte Chip erreicht eine Fläche von 4,77 x 3,21 mm^2 und enthält 104 Stufen, die einen Frequenzbereich von 200 bis 20.000 Hz abdecken. Diesen Arbeiten folgten weitere Entwürfe zu Schaltungen der inneren Haarzellen mit der Modellierung von Adaptationsvorgängen [van Schaik et al. 1997b] und einem *Spiking Neuron*-Chip, der den Haarzellenausgang in eine proportionale Impulsentladungsrate umsetzt [van Schaik et al. 1996b]. Damit sind wesentlicheVerarbeitungsstufen vorhanden, um beispielsweise das Amplituden-Detektionsmodell von [Hewitt und Meddis 1994] zu realisieren [van Schaik et al. 1997a]. Zum praktischen Einsatz gelangte das Modell bei der auditorischen Szenenanalyse, bzw. Vorverarbeitung zur Spracherkennung für die Steuerung mobiler Roboter [Shamma et al. 1998, Cohen et al. 1999].

Der im Vergleich mit einer digitalen Implementierung unschlagbar geringe Stromverbrauch der Analog-VLSI-Systeme führt direkt zur Anwendung in batteriebetriebenen oder aus elektromagnetischen Feldern gespeisten Geräten (Transponder[14]), wie z. B. Cochlea-Implantate. Ein *low-power* optimierter Entwurf wird in [Sarpeshkar et al. 1998] und eine für ein Cochlea-Implantat angepaßte Variante bereits bei [Wang et al. 1997] vorgestellt (Abb. 3.7).

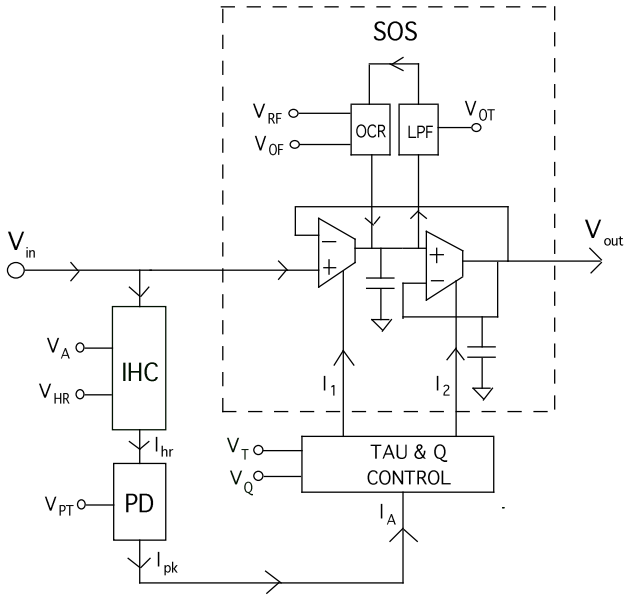


Abb. 3.7: Eine Stufe des optimierten Cochlea-Modells nach [Sarpeshkar et al. 1998]. Zeitkonstante (Resonanzfrequenz der *Second-Order-Section*, SOS) und Filtergüte *Q* sind einstell- und regelbar und zur Vermeidung einer Akkumulation der Operationsverstärker-Offsets über die gesamte Filterkette werden alle Stufen individuell offset-kompensiert (*low pass filter* - LPF und *offset correction block* - OCR). Ein Modell der inneren Haarzelle (*inner hair cell* - IHC) berechnet nach Gleichrichtung und Differenzierung einen der Eingangsspannung proportionalen Steuerstrom für den Peak-Detektor (PD). Dessen Ausgang speist eine Feed-forward Steuerung (TAU & Q CONTROL), welche die Filtergüte *Q* und Zeitkonstante τ der SOS über die Ströme I_1 und I_2 einstellt.

[13] ECPD - ES2 (*European Silicon Structures*)-Multi-Wafer-CMOS-Prozess mit zwei Metallebenen.

[14] Transponder empfangen sowohl Energie als auch Information aus einem elektromagnetischen Feld.

Auch dieses VLSI-Modell besteht aus einer charakteristischen steuerbaren Tiefpaßkaskade mit Haarzellenmodell und Spitzenwertdetektion: die Abb. 3.7 zeigt eine Stufe der Kaskade. Die gesamte Schaltung umfaßt 117 solcher Stufen, überdeckt damit einen Frequenzbereich von 100-10.000 Hz und kann etwa 60 dB Dynamik verarbeiten. Realisiert wurde das Modell in einem 1,2 µm CMOS Prozeß, benötigt eine Fläche von 7,7 mm^2 und bei 5 V Versorgungsspannung etwa 100 µA Betriebsstrom.

In Abb. 3.8 sind Übertragungskurven des Modells dargestellt; sie zeigen das erwartete Verhalten für die pegelabhängige Filtersteilheit einer Stufe (a) und die Frequenz-Orts-Beziehung der Basilarmembran (b).

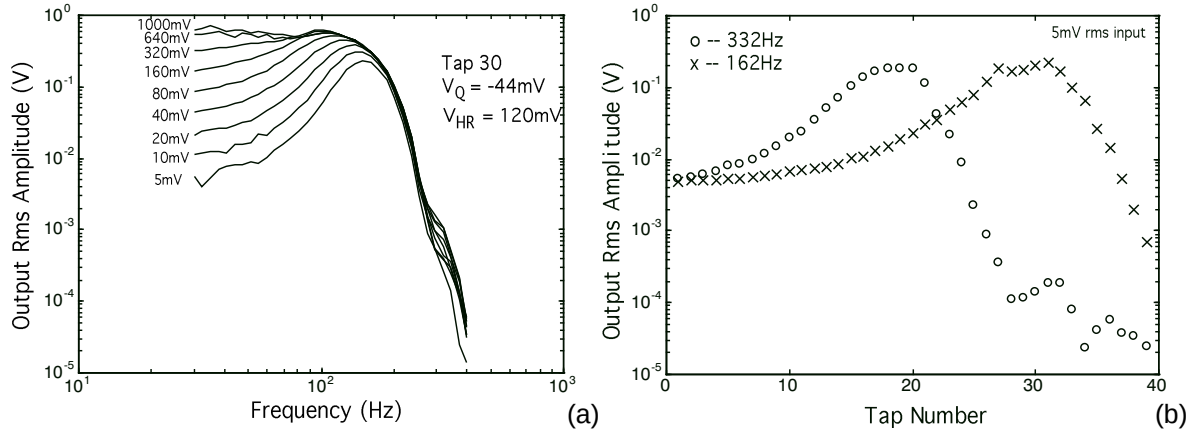


Abb. 3.8: Ausgangspegel an Stufe 30 bei verschiedenen Eingangsspannungen, die Filtergüte nimmt mit wachsenden Eingangspegeln ab und ein größerer Bereich der Basilarmembran wird erregt (a). Die Resonanzen in den Stufen (*Taps*) der Filterkaskade bei Eingangssignalen von 332 Hz und 182 Hz bei 5 mV Eingangspegel (b) zeigen die erwartete Frequenzzerlegung entlang der Basilarmembran [Sarpeshkar et al. 1998].

Cochlea-Implantate werden dann eingesetzt, wenn die normale Schallübertragung auf den Hörnerv im Innenohr prinzipiell nicht (mehr) möglich, der Hörnerv selbst aber noch elektrisch stimulierbar ist. Energie und Information werden dabei oft mit einer elektromagnetischen Kopplung durch die Haut (transkutan) übertragen. Mit außen auf der Cochlea aufsitzenden extracochleären Elektroden oder intracochleären Elektrodenschläuchen im Skalengang der Schnekke kann mit angepaßten elektrischen Impulsen dem Nerv ein Muster übertragen werden. Dieses Reizmuster wird in geeigneter Form aus dem externen Schalldruckverlauf (Mikrofonsignal) generiert, wobei ein Cochlea-Modell, das annähernd die zeitliche Erregungsausbreitung auf der Basilarmembran simuliert, die beste Korrelation zwischen technischen und natürlichen Signalen erreichen kann.

Die modellbasierte Vorverarbeitung kann dahingehend optimiert werden, daß eine auf den Ausgangssignalen aufsetzende automatische Spracherkennung (ASR) deutlich höhere Erkennungsraten erlangt. Eine Beispielapplikation mit Basilarmembranfilterbank, kanalbezogener Signalenergieberechnung bzw. Analyse der Nulldurchgänge (Abb. 3.9 a, b) wird in [Kumar et al. 1997] und [Kumar und Andreou 1997] gegeben.

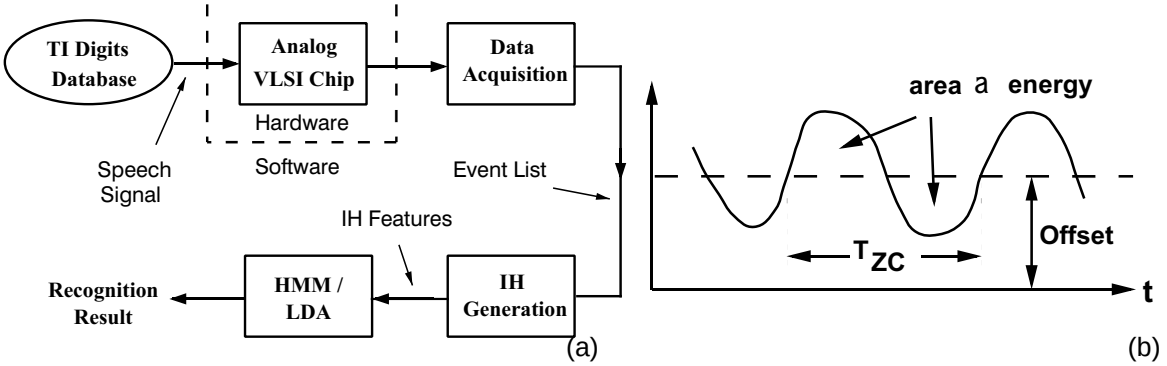


Abb. 3.9: Prinzip der Spracherkennung mit auditorisch motivierter Vorverarbeitung in einem Analog-VLSI-Schaltkreis. Intervallhistogramme (IH) werden von einem *Hidden-Markov-Model* (HMM) und einer *Linear Diskriminant Analysis* (LDA) ausgewertet. Die Schallenergie wird aus der Fläche unter der Kurve bezüglich eines Offsets berechnet (Integral nach Vollwellengleichrichtung) und gleichzeitig die Zeit zwischen zwei Schnittpunkten einer Vollwelle mit der Offsetspannung bestimmt (T_{ZC})(b) [Kumar et al. 1997].

Die Testsprachsignale waren im vorliegenden Fall gesprochene Zahlen (Ziffern, *Texas Instruments Digits Database*), die von der analogen Vorverarbeitung analysiert werden. Jede Sektion der Basilarmembran-Filterbank besteht aus einem mehrfachen Tiefpaß erster Ordnung und zwei nachgeschalteten Bandpässen. Die Filterbandbreiten sind auf einer logarithmischen Skala etwa gleich groß und erreichen dabei näherungsweise die Charakteristik der Frequenzgruppen. Hoher Aufwand ist bei der Eliminierung von Offset-Spannungen und inhärentem 1/f-Rauschen erforderlich, was durch einen kanalspezifisch dimensionierten Hochpaß erreicht wird. Mit der Detektion eines Nulldurchganges (Komparator) wird ein Histogramm aktualisiert und die ermittelte Signalenergie in die Intervalle (*bins*) der jeweiligen Nulldurchgangszeit T_{zc} sortiert. Das Intervallhistogramm (IH) wird direkt als Merkmalsvektor verwendet und an ein fünf-stufiges HMM[15] in Kombination mit einer LDA[16] übergeben. Die erreichte Erkennungsrate ist nach den dokumentierten Ergebnissen eher eine Frage der Dimensionsierung des HMM/LDA-Erkenners. Für einzelne 20 bin-IH–Merkmalsvektoren (alle 100 Hz aktualisiert) werden Raten zwischen 88 % und 98 % erreicht.

Den Hardware-Teil des Erkenners zeigt Abb. 3.10, wobei eine Kombination aus Analog- und Digitalteil ersichtlich wird. Unter Verwendung eines 1,2 µm CMOS Prozesses (5 V) konnte

[15] HMM - *Hidden Markov Model.*

[16] LDA - *Linear Discriminant Analysis*.

diese Schaltung auf 4 mm^2 mit einer 15-kanaligen Filterbank zwischen 100 und 8.000 Hz realisiert werden.

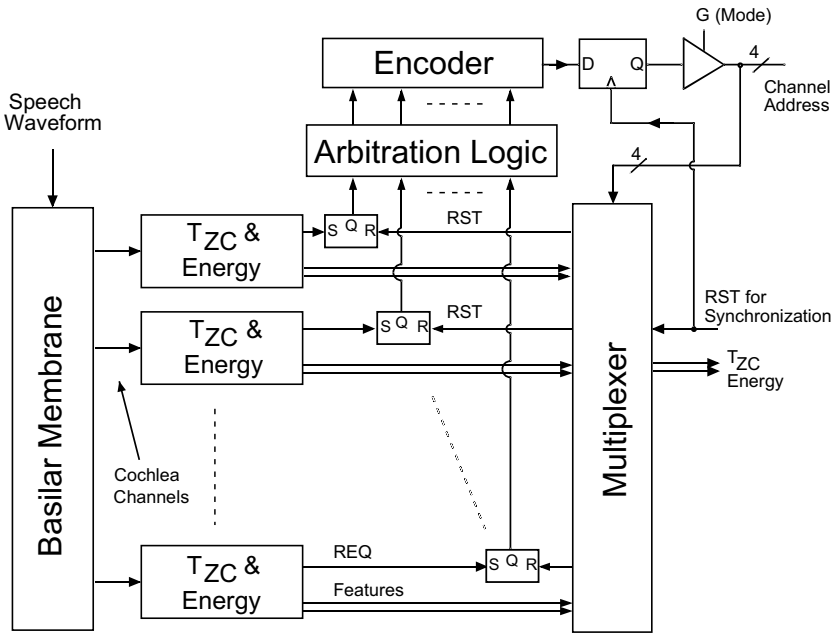


Abb. 3.10: Das Innenleben des in Analog-VLSI realisierten Spracherkenners. Nach Basilarmembran-Filterung und kanalweiser Perioden- und Energiebestimmung übergibt eine Kombination aus Registern und Multiplexer/Encoder die Daten sequentiell an den Hostrechner [Kumar et al. 1997].

Der große Vorteil der analog-VLSI Technik ist die inhärente massive Parallelität und der geringe Stromverbrauch. Dennoch konnten sich diese Ansätze abgesehen von Fallbeispielen keiner umfangreichen Verwendung erfreuen. Sowohl Stabilitätsprobleme, aber auch die Schwierigkeit der Einbindung in besser beherrschbare digitale technische Systeme begründen die geringe Verbreitung zugunsten der zahlreicheren Ansätze und Systeme mit Mikroprozessoren.

3.3 Digitale Modelle

Mit der breiten Verfügbarkeit von programmierbarer digitaler Rechentechnik, insbesondere digitaler Signalprozessoren, sind eine ganze Reihe verschiedener Ansätze zur digitalen Modellierung erarbeitet worden. Digitale adaptive Tiefpaßketten zur Basilarmembran-Filterung werden zunehmend durch parallel implementierbare Filterbänke ersetzt. Durch die Komplexität der abgebildeten Algorithmen sind diese Modelle bislang nur als Laboraufbauten im PC-Maßstab realisierbar. Es existieren bereits einige digitale Hardware-Implementierungen von Teilmodellen, die jedoch mehr an der spezifischen Modellierung einzelner Funktionseinheiten des auditorischen Systems orientiert sind. Im folgenden sollen einige Beispiele vorgestellt werden, welche die Unterschiedlichkeit der jeweils verfolgten Modellierungsstrategien und die Wech-

selbeziehungen zwischen Algorithmus und Modellarchitektur illustrieren. Abgesehen von der geeigneten mathematischen bzw. systemtheoretischen Beschreibung müssen i. d. R. Einschränkungen und Vereinfachungen bezüglich des Modellumfangs hingenommen werden. Modellierungstiefe und Leistungsfähigkeit werden dabei von der beabsichtigten Applikation, hauptsächlich aber durch eventuelle Echtzeitanforderungen bzw. Ressourcenbeschränkungen der zugrundeliegenden Rechner-Hardware beeinflußt.

Spektral-/Synchronizitätsanalyse zur auditorischen Vorverarbeitung für Spracherkenner.
Zur recheneffizienten Gewinnung von auditorisch vorverarbeiteten Merkmalsvektoren für die automatische Spracherkennung wurde bereits von [Seneff 1986] [Seneff 1988] ein mehrstufiges Zeitbereichsmodell mit Bandpaßfilterbank, Haarzellenmodell und abschließender Synchronizitätsdetektion entwickelt (Abb. 3.11).

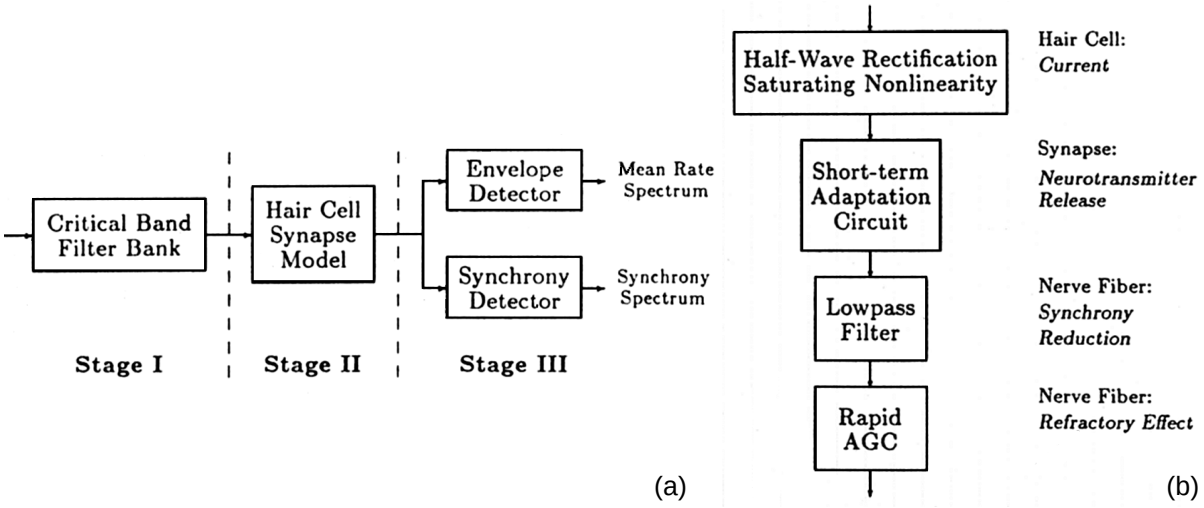


Abb. 3.11: Dreistufiger Aufbau des Cochlea-Modells (a) und die innere Struktur des Haarzellen-Modells (b) aus [Seneff 1988]. Nach Basilarmembranfilterung und adaptivem Haarzellenmodell werden sowohl der gemittelte spektrale Energiegehalt als auch ein synchron (mit der Kanalmittenfrequenz) antwortender Ausgang bereitgestellt. Das Haarzellenmodell besteht aus Halbwellengleichrichtung, zweistufiger Pegel-Adaptation und einem Tiefpaßfilter zur Modellierung des Phasensynchronizitätsverlusts.

Zunächst berechnet eine lineare Bandpaßfilterbank 40 Kanäle zwischen 120 und 6.500 Hz. Die Filterbandbreite entspricht der doppelten Frequenzgruppenauflösung, also 0,5 Bark (siehe Abb. 3.12). Das anschließende Haarzellenmodell besteht im wesentlichen aus Halbwellengleichrichtung, zweistufiger Pegelregelung (Adaptation) und Tiefpaßfilterung. Die letzte Stufe berechnet zwei Ausgangswerte je Kanal: erstens erhält man eine Energie-proportionale mittlere neuronale Feuerrate (Einhüllendenextraktion des Haarzellenmodells), die eine Aktivitätsdetektion und Segmentierung in phonetische Kategorien ermöglicht; zweitens wird mit Hilfe eines Synchronizitätsdetektors ein stark frequenzselektiver Spektralanteil (für die Bandpaß-Mittenfrequenz) extrahiert.

Für die auf diese Weise analysierten Sprachsignale konnten daraufhin die zeitlich/spektralen Muster, d. h. sowohl die bandspezifischen Energieanteile als auch klare Resonanzen für Formanten und Frikative, der Merkmalsextraktion zur Verfügung gestellt werden. Auf einer Sun-SPARC-Station 2-Workstation benötigt das Modell die 40-fache Echtzeit.

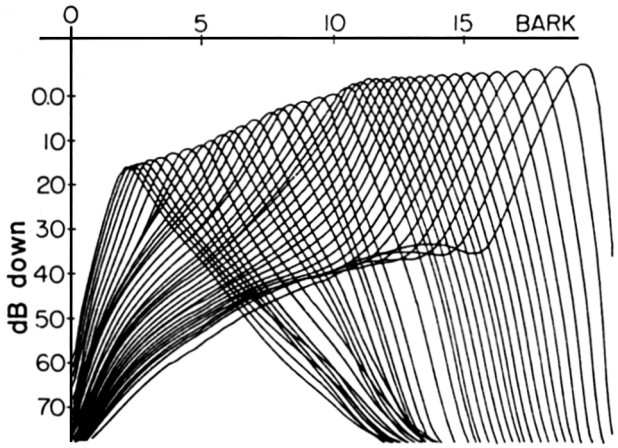


Abb. 3.12: Abstimmkurven aller 40 Kanäle des Modells, gemessen an den Ausgängen der Basilarmembranfilterbank [Seneff 1988].

Eine vereinfachte Assembler-Implementierung für einen digitalen Signalprozessor (Motorola 56000) mit Festkomma-Architektur wurde von [Ahn und Westerkamp 1990] vorgenommen. Hierfür werden vor allem angepaßte Filterdimensionen (Zweistufige Bandpaßfilter) verwendet, für die die erweiterte Harvard-Architektur[17] des DSP ausgelegt ist. Mit zweistufigen Filtern allein können die notwendigen Frequenz- und Phasengänge jedoch nur in erster Näherung modelliert werden. Außerdem muß sich aus Komplexitätsgründen die Basilarmembranfilterbank auf eine lineare Struktur beschränken und muß damit von der für die Frequenz- und Dynamikselektivität bedeutenden Wirkung der äußeren Haarzellen absehen. Trotz vereinfachter Haarzellenmodelle und geringerer Frequenzauflösung wird für die Berechnung von 20 Frequenzkanälen zwischen 200 und 3.500 Hz bei einer Abtastfrequenz von 16 kHz nur für einen einzigen Kanal Echtzeit erreicht.

Gegenüber den Tiefpaßketten-Modellen wird die zeitliche Aktivitätsausbreitung entlang der Basilarmembran weniger gut dargestellt. Andererseits ist das Prinzip der Bandpaßfilterbank rechentechnisch wesentlich kostengünstiger zu implementieren, was mit den Gammatone-Filterbänken (s. u.) wieder aufgenommen wird.

Zeitbereichsmodell mit Tiefpaß-Filterkaskade zur psychoakustischen Modellierung (auch von Innenohr-Hörstörungen).
Ein detailliertes Zeitbereichsmodell der menschlichen Cochlea wird von [Kates 1991] beschrieben. Es modelliert die Wanderwellenausbreitung auf der Basilarmembran mit einer

[17] Drei unabhängige Busse und Speicher für Programm-Kode und zwei separate Datenbereiche.

112-stufigen Tiefpaß-Kette mit abfallenden Knickfrequenzen und exponentieller Frequenz-Orts-Beziehung zwischen 16 kHz und 100 Hz (Abb. 3.13).

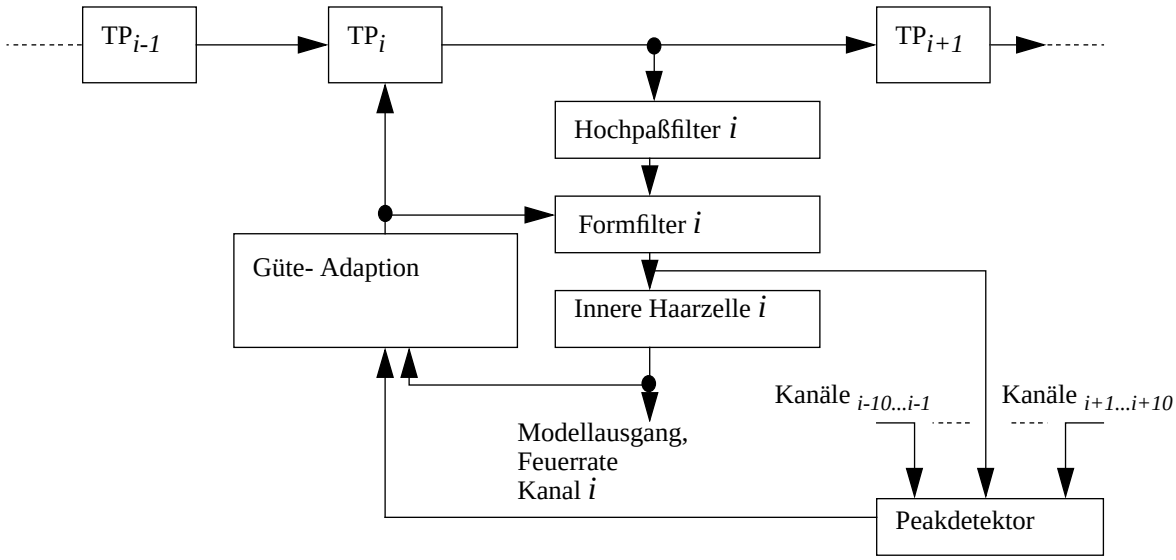


Abb. 3.13: Struktur eines Kanals des 112-stufigen Zeitbereichsmodells von [Kates 1991]. Nach jedem i-ten Tiefpaß berechnet ein Hochpaßfilter die zeitliche Ableitung, während das „Form"-Bandpaßfilter den charakteristischen Frequenzgang der neuronalen Abstimmkurven approximiert. Die Filtergüte von Tief- und Bandpaß sind dynamisch parametrisierbar, womit eine nichtlineare Resonanzanpassung/Kompression (an der Bestfrequenz des Kanals) realisiert wird. Das Haarzellenmodell enthält niedrig- und hochspontane Fasern, und bezieht die temporale Adaptation und dem Synchronizitätsverlust der Feuerrate ein. Für die Güteadaptation (Resonanzschärfe der Filter) wertet der Peakdetektor die Aktivität in den umgebenden 10 Kanälen aus.

Ähnlich den analogen Modellen wird damit sowohl die Spektralzerlegung, als auch der zeitliche Verlauf der Wanderwellenausbreitung berücksichtigt. Um die typische Form der neuronalen Abstimmkurven für jeden Frequenzkanal zu erhalten, zweigen Hoch- und Bandpaßfilter ab, die für die notwendige Resonanzüberhöhung an der Bestfrequenz sorgen und die zeitliche Ableitung der Basilarmembranschwingung berechnen. Das sich anschließende Modell für die innere Haarzelle und das Spiral-Ganglion-Neuron berechnet die neuronale Feuerrate am Ausgang. Dabei wird mit verschieden parametrisierten, parallelen Neuronen berücksichtigt, daß jede Haarzelle Neuronen mit jeweils unterschiedlichen Aktivitätsschwellwerten und Spontanraten innerviert. Die temporale Adaptation der summierten Feuerraten wird in zwei Stufen vorgenommen. Mit $\tau = 2{,}5$ ms wird die „schnelle" Adaptation und mit $\tau = 50$ ms die Kurzzeit-Adaptation mit Hilfe einer Pegelregelung (AGC[18]) vorgenommen. Die Eigenschaft der Spiral-Ganglion-Neurone, welche die Synchronizität von Anregungsphase und Feuerrate ab etwa 1 kHz verlieren, wird mit einem einstufigen Tiefpaß approximiert. Eine wesentliche Eigenschaft des Modells ist die nichtlineare Rückkopplung auf die Filter eines Kanals, welche die

[18] AGC - *Automatic Gain Control.*

Wirkung der äußeren Haarzellen widerspiegelt. Dazu wird sowohl die Feuerrate der betreffenden inneren Haarzelle, als auch die Aktivität der gesamten Umgebung (Ausgänge der Formfilter der benachbarten unteren und oberen 10 Frequenzkanäle) berücksichtigt. Die Filtergüte wird nichtlinear geregelt und damit das Resonanzverhalten an der lokalen Bestfrequenz nachgeführt. Mit einer Änderung bzw. Abschwächung der Rückkopplung konnten außerdem die für Innenohr-Hörschäden typischen Rekruitment-Effekte (siehe Kap. 2.1) simuliert werden. Die Abbildung 3.14 zeigt einen Ausschnitt (25 - 50 ms) des neuronalen Musters an den Ausgängen des Modells für die gesprochene Silbe „da“.

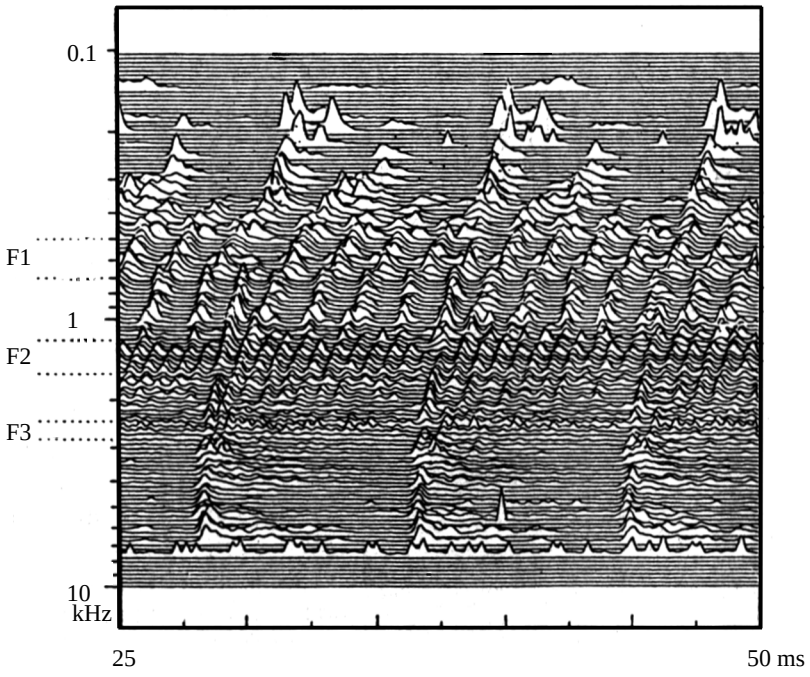


Abb. 3.14: Neuronales Erregungsmuster (Feuerraten) an den 112 Ausgängen des Modells von [Kates 1991] auf die gesprochene Silbe „da“.

In dieser Frequenz-Zeitdarstellung im Bereich zwischen 25 und 50 ms sind wesentliche psychoakustisch nachgewiesene Eigenschaften vor allem des Vokals „a“ qualitativ erkennbar. Die Wiederholung der Feuerraten-Maxima etwa alle 7,5 ms spiegeln die Grundfrequenz der Stimmbänder wider (130 Hz). Die Formanten, d. h. die Hauptfrequenzanteile, die den Vokal „a“ formen, heben sich deutlich durch eine Daueraktivität bei F1, F2 und F3 heraus. Dabei ist die Phasensynchronizität der Feuerrate bis zum zweiten Formanten gegeben, während sie sich bei F3 verliert. Die zeitliche Ausbreitung der Wanderwellen zwischen den hohen Frequenzbändern am ovalen Fenster und den tiefen Frequenzkanälen in der Nähe des Umlenkpunktes der Innenohrschnecke (Helicotrema) zeigt sich an der Verzögerung der Maxima der Grundfrequenz. Wie von [Kates 1993] festgestellt wird, ist trotz der relativ feinen Unterteilung in 112 Frequenzkanäle die Form und Bandbreite der Abstimmkurven im ursprünglichen Modell noch nicht schmalbandig genug, um physiologischen Daten zu genügen. Anstelle der Tiefpaß-Stufen wurde ein modifiziertes eindimensionales Übertragungsketten-Modell (*modified transmission line - MTL*) eingesetzt, das die geforderten Filtersteilheiten aufweist.

Tiefpaßfilterkette für ein Cochlea-Implantat-System.
In [Schwarz 1993] und [Bollerott et al. 1996] wird ein auf 16 Frequenzkanäle limitiertes Tiefpaßkettenmodell vorgestellt, das auf dem Entwurf von [Kates 1991] beruht (Abb. 3.15).

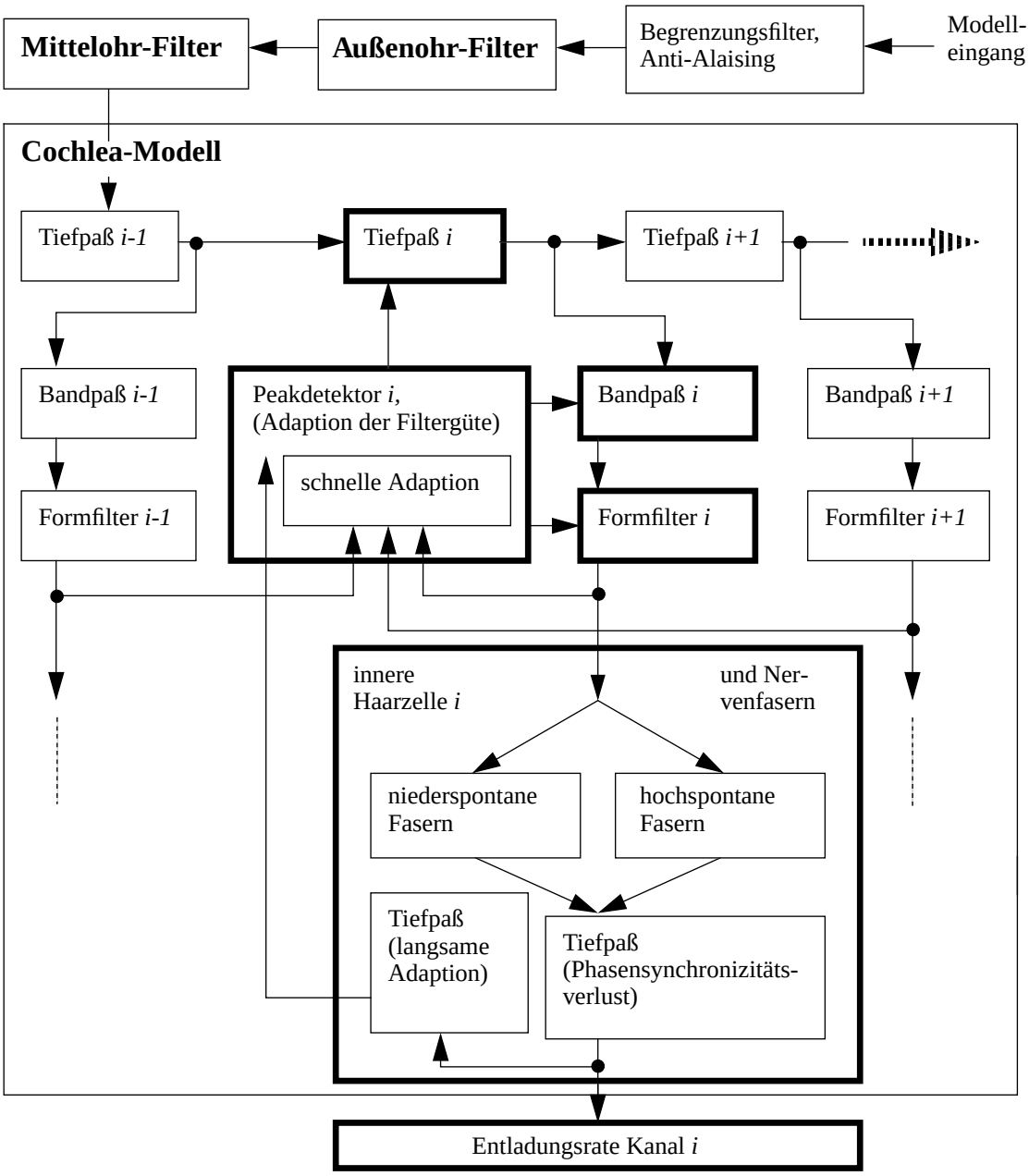


Abb. 3.15: Blockdiagramm des 16-kanaligen Cochlea-Modells von [Schwarz 1993] zur Signalanalyse für ein Cochlea-Implantat-System.

Die noch zu breitbandigen Abstimmkurven des Modells von [Kates 1991] kommen hier einer Überarbeitung und Reduzierung des Systems auf wenige Frequenzkanäle entgegen. Für digital arbeitende Cochlea-Implantat-Systeme sind die o. g. umfangreichen Modelle eher ungeeignet, da der Berechnungsumfang zu hoch ist, um Echtzeitanforderungen zu erfüllen oder den notwendigen geringen Stromverbrauch für den Batteriebetrieb zu gewährleisten. Die für das Implantat zum Einsatz kommenden Signalprozessoren haben dabei nicht nur die Signalanalyse

und -aufbereitung, sondern auch die Wandlung in Reizmuster für den Hörnerv durchzuführen. Um die Übertragungskette von der natürlichen Umwelt auf den Hörnerv zu ergänzen, wird in diesem Ansatz zusätzlich zu dem Mittelohrhochpaß und dem Cochlea-Modell ein passives lineares Außenohrfilter vorgesehen. Dieser Bandpaß zweiter Ordnung besitzt eine für den Gehörgang typische Resonanz bei 2 - 3 kHz. Die wesentlichen Bestandteile zur Regelung der Filtergüte in den jeweils beteiligten Filtern eines Kanals werden beibehalten. Die Anregung einer ganzen Region der Basilarmembran wird aufgrund der geringeren Kanalzahl nur mit den unmittelbar benachbarten Kanälen approximiert. Die Reduktion der Tiefpaßkette bewirkte darüber hinaus, daß die statische Verstärkung der parallelen Filter und die nichtlineare Filtergüte-Adaptation nunmehr zu gering und deswegen anzupassen waren. Trotz der starken Reduzierung auf die 16 Frequenzkanäle zwischen 250 und 7.400 Hz kann ein dem 112-stufigen Modell von [Kates 1991] vergleichbares Aktivitätsmuster erzeugt werden (vgl. Abb. 3.16). Diese und weitere Simulationen des Modells auf Pegelsprünge, sinusiodale Eingangssignale und Sprachabschnitte bestätigten die ausreichende Abbildung der gewünschten psychoakustischen Eigenschaften für die Generierung gehörgerecht vorverarbeiteter Reizmuster. Das Modell wurde bereits zur Vorbereitung einer direkten Hardware-Synthese des Algorithmus als Verhaltensbeschreibung (mit *Verilog-XL*) implementiert und konnte korrekt simuliert werden.

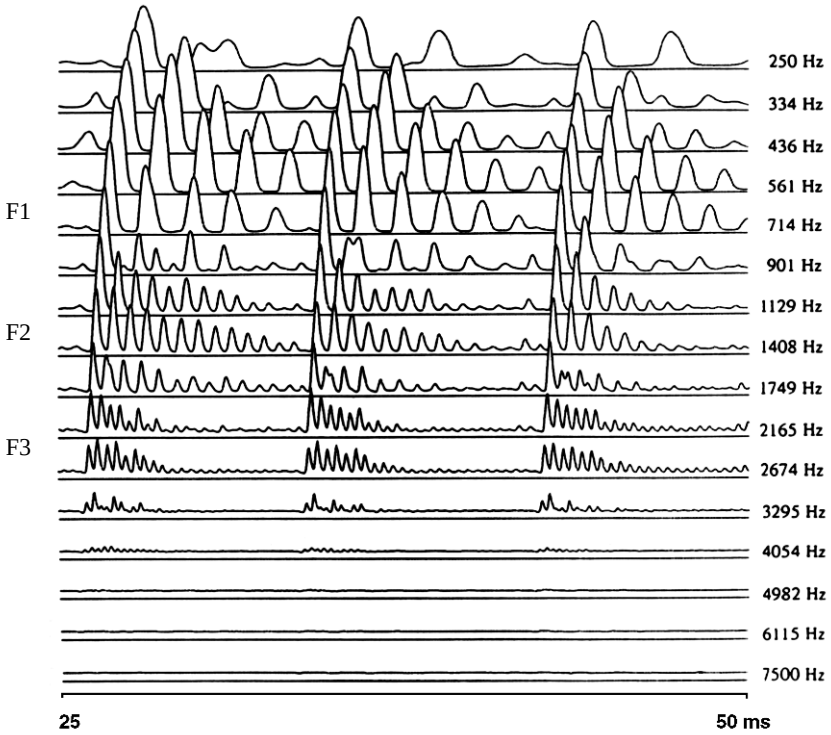


Abb. 3.16: Modellausgang des zum Einsatz in einem Cochlea-Implantat-System vorgesehenen nichtlinearen Modells [Schwarz 1993]. Der spektrale Verlauf und die Hüllkurven der neuronalen Aktionspotentiale für eine gesprochene Silbe „da“ entsprechen qualitativ dem natürlichen System. Trotz der begrenzten Kanalanzahl kann die Grundfrequenz der Stimmbänder (ca. 120 Hz), die Lage der Formanten F1, F2, und F3 für den Vokal „a“ (700, 1200 und 2400 Hz) und die zeitliche Ausbreitung der Erregung entlang der Basilarmembran deutlich dargestellt werden. Keiner der drei Formanten fällt genau mit einer Kanalbestfrequenz zusammen, so daß eine erhöhte Aktivität in den jeweils benachbarten Kanälen zu beobachten ist.

Eine alternative Implementation des Modells auf einem Motorola-56002-DSP (Taktrate 40 MHz, mindestens 2 Taktzyklen je Befehl) erwies sich als zu komplex, um bei einer Abtastfrequenz von 31.250 Hz in Echtzeit ablaufen zu können. Obwohl die Tief- und Bandpässe auf optimal von der DSP-Architektur unterstützten zweistufigen Filtern beruhen, liegt der Hauptaufwand der Berechnungen in der lokalen und frequenzlateralen adaptiven Rückkopplung der Filter- und Haarzellenausgänge.

Von [Kim et al. 1999] existiert eine weitere Implementierung des Modells nach [Kates 1991], die als Vorverarbeitung für die Spracherkennung optimiert ist. Dabei wird die Struktur der Tiefpaßkette für die Basilarmembranfilterung beibehalten, aber auf das Haarzellen- und Spiralganglion-Modell verzichtet. Als Ersatz wird eine nichtlineare Funktionseinheit (siehe Abb. 3.17) zur Erzeugung einer Zeit-Frequenzdarstellung von Nulldurchgängen und Amplitudenmaxima vorgeschlagen. Diese Vorverarbeitung zur Gewinnung von Merkmalsvektoren zur automatischen Einzelworterkennung zeichnet sich insbesondere unter Störgeräusch durch signifikant höhere Erkennungsraten aus als LPC[19]-basierte Verfahren, die Cepstral-Koeffizienten als Merkmalsrepräsentation generieren.

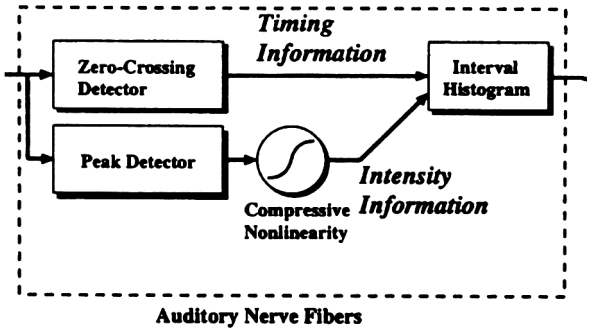


Abb. 3.17: *Zero-crossings with peak amplitude*-Einheit des Zeitbereichsmodells zur Sprachvorverarbeitung nach [Kim et al. 1999]. Ein Intervall-Histogramm (von Feuerraten) wird dabei aus der nichtlinear verarbeiteten Zeit- und Amplitudeninformation des basilamembrangefilterten Signals gewonnen. Die zeitlichen Parameter der Signale ermittelt ein pegelunabhängiger Nulldurchgangszähler, wobei die Spitzenwerte der Filterbankausgänge noch einer nichtlinearen Kompression (Pegelregelung) unterworfen werden.

Bei Experimenten wurde eine angepaßte Version des Modells mit 20 Kanälen im Bark-Abstand zwischen 0 und 5.000 Hz (bei 16 kHz Abtastfrequenz) eingesetzt und verschiedene Vorverarbeitungsverfahren an einem HMM-Einzelwort-Erkenner getestet. Dabei wurden 87,4 % bei 30 dB SNR und 34,1 % bei 0 dB SNR erreicht (73,9 % bzw. 3,3 % bei LPC-basiertem Verfahren). Obwohl keine Hardware-Realisierung im Mittelpunkt steht, wird hierbei eine anwendungsspezifische und recheneffiziente Optimierung angestrebt (begrenzter Frequenzbereich, geringe Kanalanzahl, vereinfachtes Haarzellenmodell).

[19] LPC - *Linear Predictive Coding* ist ein datenkomprimierendes Sprachübertragungsverfahren (z. B. LPC-Vocoder, bei dem von der Senderseite nach einer Sprachanalyse nur Filterkoeffizienten und Anregungsparameter für ein Sprachsynthesefilter zum Empfänger übertragen werden).

Frequenzabtastfilterbank und Haarzellenmodell mit FPGA[20] für Cochlea-Implantate.
Ein in Hardware realisierter Entwurf eines Cochlea-Models ebenfalls für Anwendungen in Cochlea-Implantaten findet sich in [Meyer-Bäse et al. 1997a]. Die beschriebene Abtastfilterbank wird unter Anwendung eines exakten Residuen-Zahlensystems (RNS[21]) implementiert. Das dafür entworfene residuenarithmetische Rechenwerk erweist sich - für die Berechnung der Filterkaskade - als wesentlich effizienter, als eine vergleichbare Version mit FIR-Filtern[22] und Zweierkomplement-Arithmetik. Ein Frequenzabtastfilter (FSF[23]) kann aus einer Resonator-Kammfilter-Kombination bestehen, wobei die Nullstellen des nachgeschalteten Kammfilters die Polstellen der Resonatoren exakt auslöschen. Die Resonatoren sind durch die (Winkel-) Lage ihrer Polstellen auf dem Einheitskreis definiert, die Kammfilter durch die Anzahl der Verzögerungsstufen (Nullstellen). Zu Lasten eines hohen theoretischen Entwurfsaufwands und unter Hinnahme einer unflexiblen Filteranordnung und bezüglich Frequenzverlauf und Genauigkeit handoptimierter Filterstrukturen (siehe Abb. 3.18) werden für eine 8-kanalige Filterbank zwischen 900 und 8.000 Hz (Abtastfrequenz 16.624 Hz) nur insgesamt 1.572 CLBs[24] eines *Xilinx*[25]-FPGA benötigt. Ein vergleichbares Design mit FIR-Filtern verbraucht dagegen 11.292 CLBs.

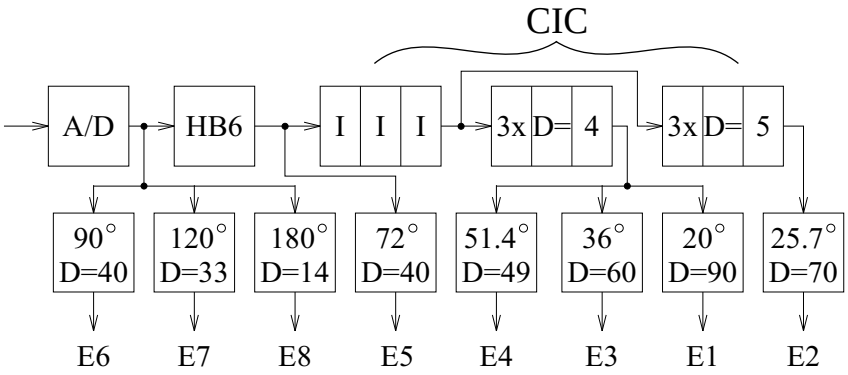


Abb. 3.18: Abtastfilterbank zur Spektralanalyse für ein Cochlea-Implantat-System nach [Meyer-Bäse et al. 1997a]. Nach A/D Wandlung erfolgt eine Vorfilterung mit einem Anti-Alias Filter (*HB6*) und einem kombinierten Integrator-Kamm-Filter (*CIC*). Die Filterbank-Ausgänge E1 bis E8 werden aus Frequenzabtastfiltern mit Polstellenwinkeln zwischen 20 und 180 Grad und Kammfiltern mit einer Verzögerung von D = 14...90 gebildet. Die unteren Filterbankkanäle benötigen eine zusätzliche dreifache Verzögerung von D = 4 bzw. D = 5. Die lokal angepaßten Wortbreiten der Filter liegen zwischen 14 und 22 Bit.

[20] FPGA - *Field Programmable Gate Array* sind konfigurierbare digitale Bausteine, in denen die zu realisierende Schaltung aus einer Vielzahl frei verknüpfbarer Logikzellen gebildet wird. Programmiert werden sowohl die lokalen Schaltfunktionen der Logikzellen (d.h. das Beschreiben von statischen und wiederbeschreibbaren Speichern), als auch deren Vernetzung auf dem FPGA (vgl. Kap. 4.2).

[21] RNS - *Residue Number System.*

[22] FIR - *Finite Impulse Response*; nichtrekursives Filter mit endlicher Impulsantwort.

[23] FSF - *Frequency Sampling Filter.*

[24] CLB - *Complex Logic Blocks* eines *Xilinx* XC4000-FPGA.

[25] *Xilinx, Inc.* - FPGA-Hersteller.

Diese Basilarmembranfilterbank wird von [Meyer-Bäse et al. 1997b] um das FPGA-Design eines Modells der inneren Haarzellen ergänzt. Für die Realisierung des Transmitter-Reservoire-Ansatzes nach [Meddis 1986] werden zunächst 276 CLBs eines *Xilinx* XC3090-FPGA benötigt (entspricht 70 % der Gesamtkapazität dieses FPGA).

Ein zweiter Ansatz (siehe Abb. 3.19) orientiert sich dagegen an einer funktionalen Version, d. h. an der Implementierung gekoppelter Übertragungsfunktionen, die einzelnen Eigenschaften des Modells entsprechen. Das wesentliche Element ist ein aus drei Anteilen bestehender Adaptationsalgorithmus nach [Westerman und Smith 1984]: danach hat das adaptierte Ausgangssignal zwei unterschiedlich schnell abklingende Komponenten (mit den Zeitkonstanten $\tau = 3$ ms und $\tau = 60$ ms) und einen Daueranteil.

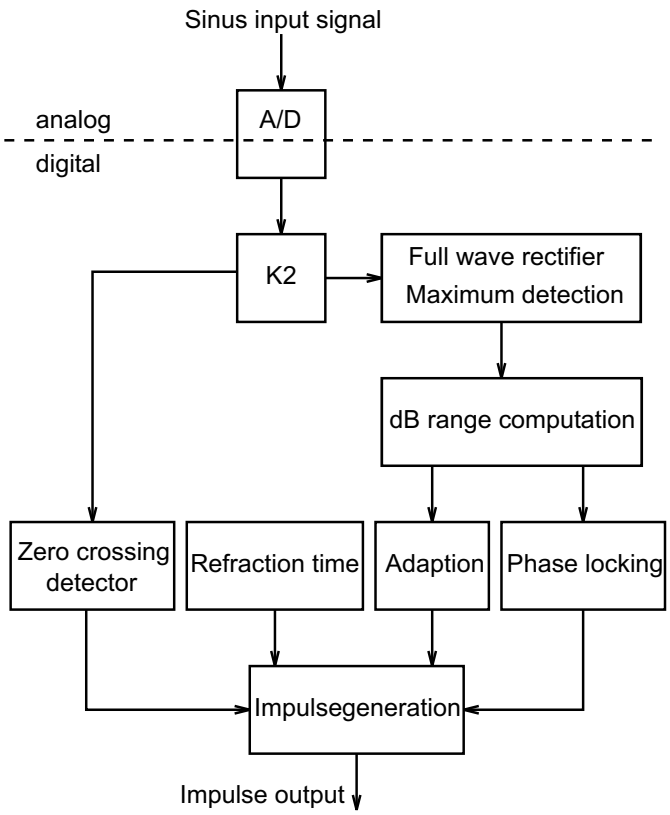


Abb. 3.19: Blockdiagramm des FPGA-Designs für ein funktionales Modell der inneren Haarzellen nach [Meyer-Bäse et al. 1997b]. Nach 12 Bit-A/D Wandlung ins Zweierkomplement-System (*K2*) werden zunächst die Einhüllenden nach Vollwellengleichrichtung auf Maxima untersucht und mit Hilfe einer ROM[26]-Tabelle vier Pegelbereichen in dB zugeordnet. Zwei weitere I/O-Tabellen ordnen den Pegeln die entsprechenden Werte der Blöcke für Adaptation und Phasensynchronizität (*Phase locking*) zu. Die abschließende Stufe zur Erzeugung neuronaler Impulse wird von einer Erholungszeit (*Refraction time*) und einem Nulldurchgangsdetektor (*Zero crossing detektor*) beeinflußt. Erholungszeit und Ausgang der Adaptationsstufe bestimmen primär die Impulserzeugung, Phasensynchronizität und Nulldurchgang die Phasenlage des Impulses.

[26]ROM - *Read Only Memory.*

Das funktionale Haarzellenmodell benötigt 227 CLBs und damit nur unwesentlich weniger als das Transmitter-Reservoire-Modell nach [Meddis 1986]. Insbesondere die grobe Zuordnung mit 4-stufigen Wertetabellen sorgt für ein qualitativ mäßiges Gesamtverhalten des Modells. Immerhin wird es auf einem Multi-FPGA-Board realisiert und kann zusammen mit der o. g. Filterbank in Echtzeit betrieben werden.

Wavelet-Dekomposition für Hörgeräte**.**
Die wesentliche Stufe aller Modelle des auditorischen Systems ist die Dekomposition des Sprachspektrums in einzelne Frequenzbänder. Neben den Filterbank-Prototypen werden dafür auch Zeit-Frequenz-Analysealgorithmen wie z. B. FFT oder aber Wavelet-Filterbänke eingesetzt. Im Gegensatz zur gleichmäßigen Unterteilung des Zeit-Frequenzdarstellung bei einer FFT über einer gegebenen Anzahl von Abtastwerten (bei FFT ganzzahlige Potenzen von 2) berechnet eine Wavelet-Analyse durch iteratives Falten und Unterabtasten lokal unterschiedlich aufgelöste Spektren[27]. Da eine solche Bandunterteilung der Auflösung nach Frequenzgruppen des Ohres anpaßbar ist, liegt der Versuch nahe, hiermit die Vorteile eines rechentechnisch effizienten Wavelet-Algorithmus [Mallat 1989] mit einer annähernd idealen Zeit-Frequenzauflösung kombinieren zu können. Die Anwendung eines solchen Analyse-Modells zur Recruitment-Kompensation, also einem bandspezifischen Dynamikausgleich für Hörgeschädigte (siehe Kap. 2.2), wird in [Drake et al. 1993] beschrieben. Bei einer Unterteilung des Hörspektrums in drei Teilbänder (0 - 1.000 Hz, 1.000 - 2.000 Hz, 2.000 - 4.000 Hz) wurden bei subjektiven Tests gegenüber einem parallelen Filterbankmodell jedoch keine Verbesserungen erreicht. Das kann darauf zurückzuführen sein, daß bei der Berechnung der Wavelet-Koeffizienten (Gewichtungsfaktoren vor der Resynthese) nur statische Isophonen, aber keine wirklichen Lautheitsmodelle (siehe Kap. 2.2) Verwendung fanden. Vorteilhaft aus Sicht einer dedizierten Hardware ist der nur linear mit der Eingangsdatenlänge ansteigende Rechenaufwand der Wavelet-Analyse/ Resynthese (bei FFT dagegen $n * log(n)$, n = Eingangsdatenlänge). Auch im Vergleich zur Dekomposition mit parallelen Filterbänken ist der Ressourcenbedarf geringer, da sich der Datendurchsatz und die Wavelet-Filterlänge bei jeder weiteren Iteration jeweils auf die Hälfte reduzieren: im *i-ten* Teilband kann bei $1/2^i$ weniger Speicher auch $1/2^i$ langsamer gerechnet werden.

In Arbeiten von [Solbach et al. 1998] wird ein ähnlicher Ansatz zur gehörgerechten Signalanalyse beschrieben. Hierbei wird die Wavelet-Dekomposition mit einem Gammatone-Motherwavelet auf der Basis des optimierten Gammatone-Filters von [Patterson et al. 1992] durchgeführt. Damit gelingt die Anpassung an die auditorische Filtercharakteristik noch we-

[27] Im Prinzip wird bei jeder Iteration das Teilband *w* in zwei Teilbänder *w1* und *w2* geteilt. Danach wird nur das niederfrequente Teilband *w2* weiterverarbeitet, nach der Faltung mit der zugehörigen Skalierungsfunktion unterabgetastet und in der nächsten Iteration wiederum geteilt. Die Filter selbst werden durch Faltung mit (durch äquivalente Unterabtastung) auseinander hervorgehenden Gewichtsfunktionen (Wavelets) realisiert. Das Ergebnis ist eine hohe Frequenzauflösung bei geringer Zeitauflösung für tiefe Frequenzen und vice versa für hohe Frequenzen (hingegen muß man sich bei der FFT entweder für ausreichend hohe aber äquidistante Frequenz- *oder* Zeitauflösung entscheiden). Diese Transformation ist vollständig umkehrbar, d. h. es existiert ein zur Analyse inverser Resynthese-Filteralgorithmus für frequenzbandbeschränkte Signale.

sentlich besser, als im vorigen Beispiel. Dieses Modell wurde für die auditorische Quellentrennung und Transkription polyphoner Musik verwendet.

Transputermodell für die Tonanalyse.
Ein weiterer Versuch, parallele Standard-Hardware für die auditorische Vorverarbeitung einzusetzen, findet sich bei [Tyrrell et al. 1992]. Hier wird zur Basilarmembranfilterung bereits eine Gammatone-Filterbank in verschiedenen Konfigurationen mit T800 Transputern realisiert (Abb. 3.20).

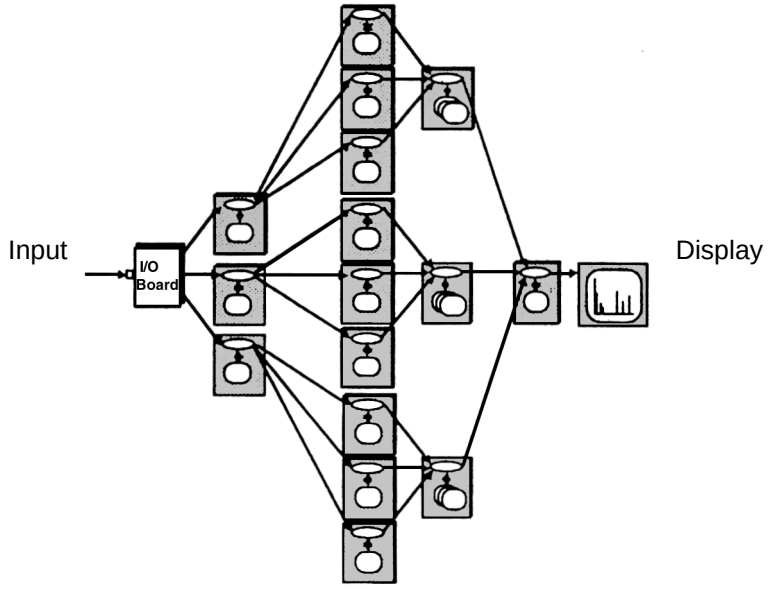


Abb. 3.20: Transputer-System zur auditorischen Gammatone-Filterung mit 16 Transputern T800 aus [Tyrrell et al. 1992].

Unter Ausnutzung der 4 Transputer-Links werden 20 Filterkanäle in 16 Transputern untergebracht, wobei die maximale Abtastfrequenz bei 9,65 kHz liegt.

Ein aktualisierter Entwurf arbeitet mit einem T9000 Transputer-Subsystem zur Implementierung von 64 parallelen Bandpaßfiltern bei einer Abtastrate von 16 kHz [Howard et al. 1998, Tyrrell et al. 2000]. Weitere Modellstufen, wie z. B. Haarzellenmodelle bzw. temporale Adaptationsmechanismen werden nicht berücksichtigt; das System wird zur gehörgerechten Echtzeit-Spektralanalyse von Musik- und Sprachsignalen eingesetzt. Auch dieser Ansatz illustriert den hohen Hardware-Aufwand, den die Implementierung selbst eines Cochlea-Teilmodells unter Echtzeitbedingungen erfordert.

Auditory Image Model (AIM) für Spracherkennung und psychoakustische Modellierung. Das „Auditory Image Model“ von [Patterson et al. 1995] erlaubt den direkten Vergleich eines funktionalen psychoakustischen Modells mit einer weitaus aufwendigeren physiologisch motivierten Implementation. Die Abbildung 3.21 gibt dazu einen Überblick.

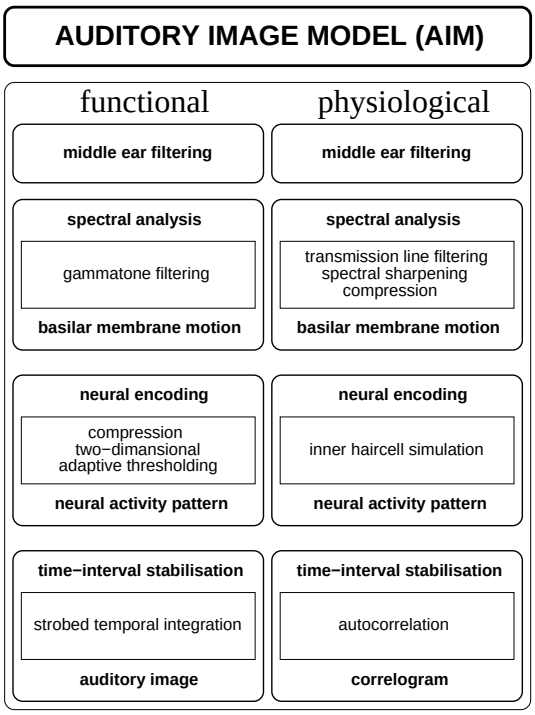


Abb. 3.21: Das „Auditory Image Modell“ nach [Patterson et al. 1995] kann wahlweise als funktionales (recheneffizientes) oder physiologisch detailliertes Modell bzw. abschnittsweise gemischt betrieben werden.

Die Verarbeitung erfolgt in drei Stufen: Spektralanalyse, neuronale Mustergenerierung und zeitliche Integration. Voraus geht jedoch eine Mittelohrfilterung, deren Güte optional nachgeregelt werden kann. In der ersten Modellstufe kann die lineare Gammatone-Filterbank der funktionalen Variante durch eine nichtlineare „Wellen-Digital-Filterbank“ der physiologischen Variante ersetzt werden. Letztere modelliert die Auslenkungen der Basilarmembran durch die bekannte Tiefpaßübertragungskette, wobei die Nichtlinearitäten die lokalen Rückkopplungseffekte durch die äußeren Haarzellen simulieren. In der zweiten Modellstufe erfolgt die Umwandlung der Filterbankausgänge in neuronale Muster, wobei zunächst eine kanalweise Gleichrichtung und eine Dynamik-Kompression durchgeführt werden. Das Haarzellenmodell berücksichtigt den zeitlichen Verlauf der Mustergenerierung in jedem Kanal mit Hilfe unterschiedlich schnell antwortender und adaptierender Zelltypen. Außerdem erfolgt die Einbeziehung benachbarter Frequenzkanäle, so daß im Ausgangssignal die Kontrastierung im Zeit- und Frequenzbereich erreicht werden kann. Die physiologisch motivierte Alternative arbeitet mit dem Haarzellenmodell von [Meddis 1988], das die neuronale Kodierung auf dem Prinzip von Neurotransmitterreservoiren detailliert beschreibt. Die dritte Modellstufe berechnet die zeitliche Integration in jedem Frequenzkanal. Funktional werden dafür kanalweise Verzögerungsketten implementiert, die das Signal zunächst linear mit der Verweildauer in der Kette dämp-

fen. Dabei wird im Ausgangssignal der Haarzelle nach Aktivitätsmaxima gesucht, und ein mit einem Maximum beginnendes Zeitfenster wird auf den Inhalt der Verzögerungskette addiert. Damit ist das AIM in der Lage, die Feinstruktur (Zeitintervall-Information) trotz Zeitintegration zu erhalten. Alternativ wird ein Autokorrelator eingesetzt. Die Abbildung 3.22 zeigt die Ausgänge des funktionalen Modells an den drei Stufen für das gesprochene Wort „hat“ (*engl.*).

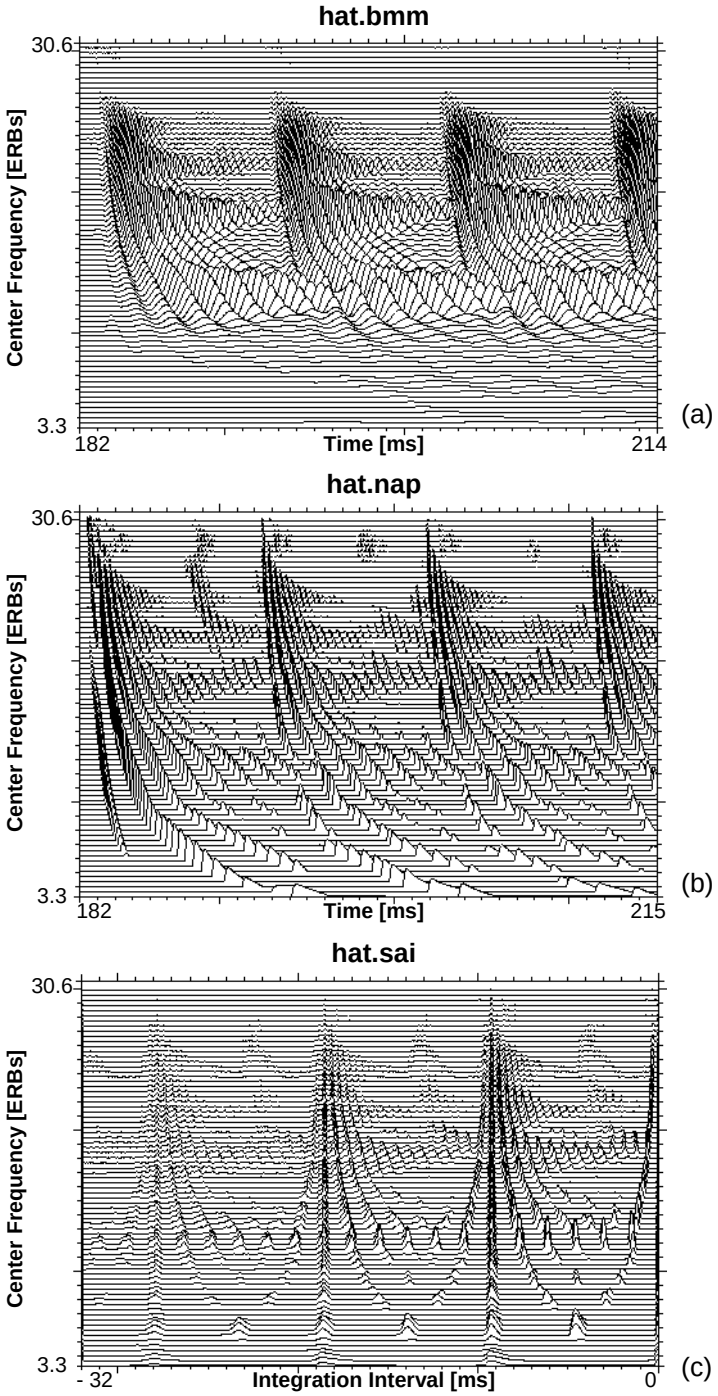


Abb. 3.22: Antwort des funktionalen „AIM“ auf das gesprochene Wort „hat“ (*engl.*) bei 60 dB SPL Eingangspegel. Diagramm (a) zeigt den Signalverlauf am Ausgang der Gammatone-Filterbank in den 75 Frequenzkanälen zwischen 100 und 6.000 Hz (3.3 und 30.6 ERB[28]), (b) den Ausgang der inneren Haarzellen und (c) der zeitlich integrierenden Stufe [Patterson et al. 1995].

Dieses digitale Zeitbereichsmodell wurde ursprünglich als reine Software-Realisierung für verschiedene (nicht echtzeitfähige) UNIX-Systeme konzipiert und stellt eine Plattform zur funktionalen psychoakustischen Simulation der Tonperzeption und der Effekte bei Innenohrschwerhörigkeit zur Verfügung. Außerdem konnte das „AIM“ als Vorverarbeitung für die Spracherkennung eingesetzt werden. Eine recheneffiziente Implementierung des AIM für eine „general-purpose“- CPU wurde von [Slaney 1993] veröffentlicht.

Eine Erweiterung der hier eingesetzten linearen Gammatone-Filterbank, die eine pegelunabhängige Frequenzdekomposition vornimmt, erfolgte durch [Irino und Patterson 1997]. Die „Gammachirp“-Filterbank realisiert eine Rückkopplung zur Regelung der Filterverstärkung.

Haarzellenmodell und Pitch-Perzeption zur psychoakustischen Echtzeitsimulation.
Ein Modellansatz, der über eine einfache Bandpaßanordnung zur Basilarmembranfilterung und die Stufe der zugehörigen Haarzellen weit hinausgeht und die weitere neuronale Verarbeitung in Stellate- und Koinzidenz-Zellen einbezieht, wird von [Jones et al. 1998, Temple et al. 1996, Temple et al. 1997, Lim et al. 1997a] und [Lim et al. 1997b] beschrieben.

Der Schwerpunkt der Arbeiten lag auf der detaillierten Abbildung der neuronalen Detektion der Sprecher-Grundfrequenz (*pitch*). Das Eingangssignal wird mit IIR[29]-Bandpässen 2-ter Ordnung in 30 Frequenzkanäle auf einer logarithmischen Skala zwischen 80 und 5.000 Hz zerlegt. Jedem Basilarmembranfilter folgt eine Haarzelle nach [Meddis 1986, Meddis 1988], welche die Halbwellengleichrichtung, die nichtlineare Sättigung und eine zweistufige Adaptation vornimmt. Danach folgt die Erzeugung von neuronalen Entladungsimpulse *(spikes)* in unterschiedlich parametrisierten Bänken aus Stellate-Zellen, deren Ausgänge in Koinzidenzzellen zusammengefaßt werden, und neuronale Muster für die Grundfrequenz im Sprachsignal (Sprecherfrequenz, *pitch*), erzeugen. Die Abbildung 3.23 zeigt den vierstufigen Aufbau des Modells.

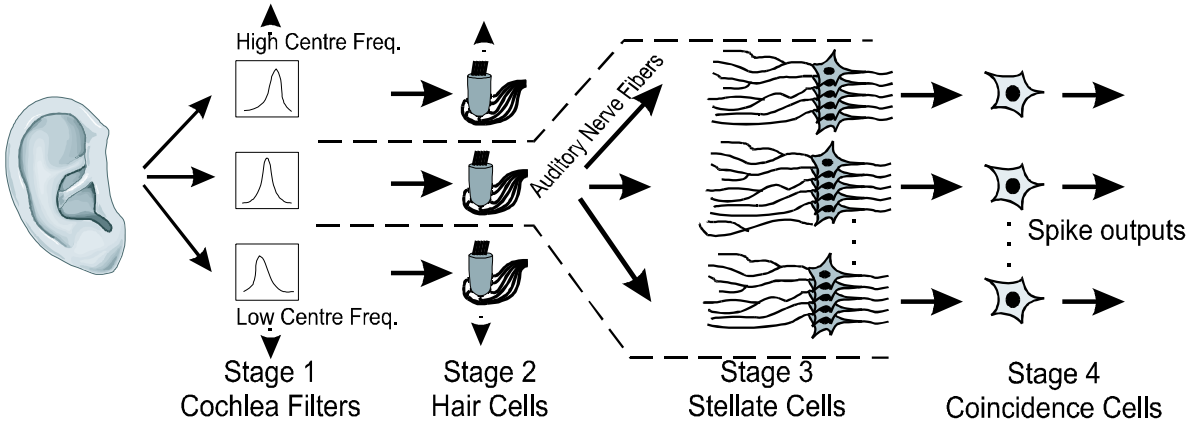


Abb. 3.23: Struktur des vierstufigen Pitch-Detektionssystems [Jones et al. 2000].

[28]ERB - *Equivalent rectangular bandwidth*. Die entsprechende ERB-Skala bezeichnet eine der Bark-Skala vergleichbare Unterteilung nach Frequenzgruppen. Siehe hierzu [Moore und Glasberg 1987].

[29]IIR - *Infinite Impulse Response*; rekursives Filter mit unendlicher Impulsantwort.

Das Modell wurde als strukturelles und mit VHDL[30] (vgl. Kap. 4.2) beschriebenes Hardware-Design für *Xilinx*-FPGA realisiert. Der gesamte Entwurf arbeitet bei 20 kHz Abtastrate mit vorzeichenbehafteten 16 Bit-Festkommazahlen, obwohl - wie weitere Tests zeigten - auch 14 Bit ausreichend sind. Der Aufbau des Modells aus bitseriell arbeitenden Funktionseinheiten erleichtert die Schnittstellengestaltung, muß aber durch erhöhten Steuerungsaufwand erkauft werden. Die Abbildung 3.24 zeigt beispielhaft den internen Aufbau des Basilarmembranfilters.

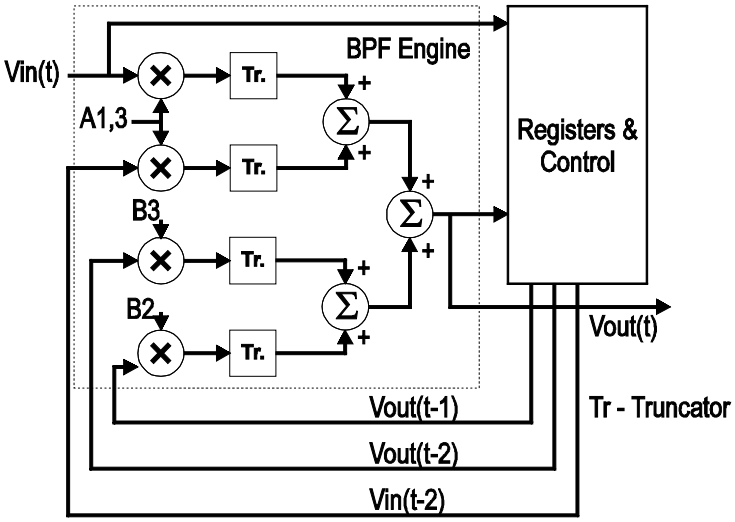


Abb. 3.24: Bitserielle Arithmetik für das zweistufige IIR-Bandpaßfilter (BPF): dargestellt ist ein Kanal der Basilarmembranfilterbank. Konstantenmultiplikation (⊗), Stellenreduktion (Abschneiden, *truncate*) und Addition (Σ) bilden ein Schaltnetz; der korrekte Datentransport wird von Registerbänken und dem Steuerautomaten sichergestellt [Jones et al. 1998].

Der Hardware-Entwurf des gesamten Pitch-Detektions-Systems besitzt eine ausgedehnte Verarbeitungsstruktur. Der Ausgang der Haarzelle beschreibt die Konzentration von Neurotransmittern im synaptischen Spalt, d. h. ein der Impulsentladungs-Wahrscheinlichkeit proportionaler Wert. Dieser wird von den nachfolgenden auditorischen Nervenfasern zunächst mit skaliertem Rauschen versehen. Die Stellate-Zell-Dendriten geben eine tiefpaßgefilterte Version (einstufiges IIR-Filter mit $f_g = 300$ Hz) an den Zellkörper weiter. Dieser generiert, die Phasensynchronizität berücksichtigend, einen neuronalen Entladungsimpuls. Insgesamt arbeiten 20 Bänke zu je 40 solcher Stellate-Zellen in jedem Frequenzkanal parallel, wobei die 40 Zellen einer Bank auf eine Vorzugs-Modulationsfrequenz zwischen 80 und 300 Hz eingestellt sind (*best modulation frequency - bmf*). Die Ausgänge der Bänke werden in Koinzidenz-Zellen zusammengefaßt. Existiert eine dominierende Modulationsfrequenz, wird in diesen Zellen ein weiterer Impuls generiert, und dadurch auf eine hohe Koinzidenz in dem Impuls-Muster der jeweiligen Bank hingewiesen. Abbildung 3.25 veranschaulicht den Aufbau des kompletten Hardware-Entwurfs. Für Echtzeit-Tests wurde ein einzelner Frequenzkanal komplett in einem

[30] VHDL - *Very Large Scale Integrated Hardware Description Language* (Hardware-Beschreibungssprache).

Xilinx XC4000-FPGA implementiert. Eine Echtzeitverarbeitung wird tatsächlich erreicht, obwohl eine einzige Stellate-Zelle 40-fach multiplex betrieben wird (sequentielles Design). Die Parameter der Zellen müssen in einem externen RAM[31] gespeichert werden. Der Frequenzkanal benötigt 44.095 FPGA-Gatter und erreicht eine Taktrate von 25,6 MHz. Die Verarbeitung eines Abtastwertes benötigt 1.280 Taktzyklen, d. h. 50 µs, was den geforderten 20 kHz Abtastfrequenz entspricht. Die vollständige Implementierung des 30-kanaligen Designs ist somit nicht durch fortgesetztes Multiplexen der bereits vorhanden Hardware möglich und verlangt eine (hier nicht verifizierte) parallele Implementierung von insgesamt 1,32 Millionen XC4000-Gattern.

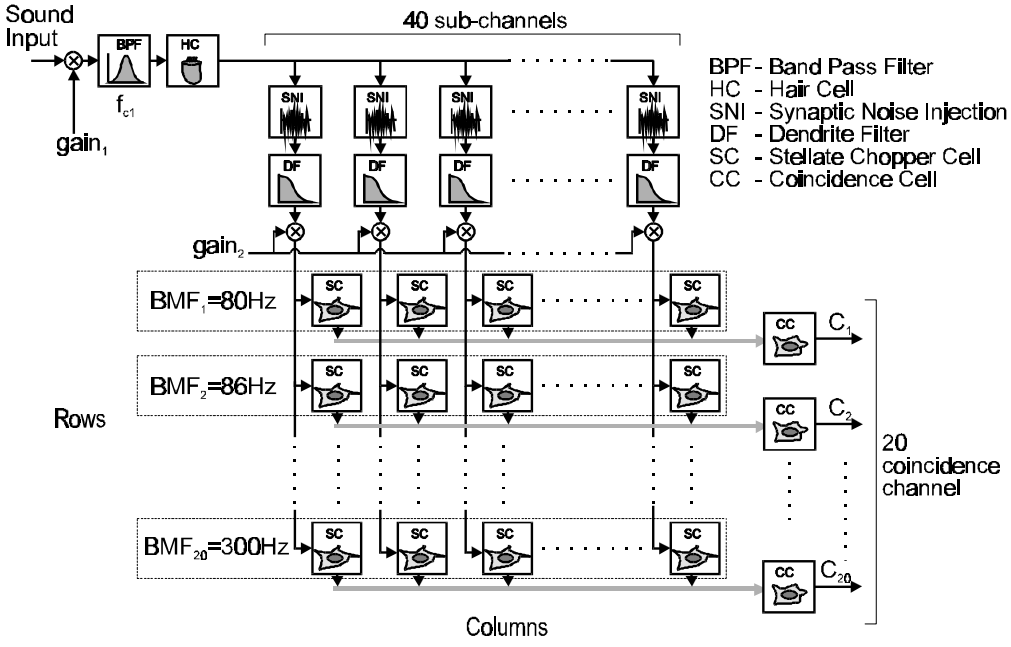


Abb. 3.25: Hardware-Struktur des Pitch-Detektions-Modells nach [Jones et al. 2000].

Modell zur Untersuchung auditiver Wahrnehmungschwellen.
Die im menschlichen Gehör existierenden Wahrnehmungsschwellen, die durch die spektralen und zeitlichen Verdeckungseffekte bei der Abbildung komplexer Signale entstehen, sind Gegenstand einer in [Baumgarte 2000] dargelegten Arbeit. Dafür wird ein physiologisch und psychoakustisch motiviertes Gehörmodell erstellt und in einer Anwendung zur Sprachkodierung validiert. Hierbei wurden die Beziehungen zwischen Wahrnehmungsschwellen und der Perzeption der bei der Kodierung auftretenden Signalverzerrungen untersucht. Der sehr umfangreiche Ansatz wurde in Form von RCL-Netzwerken modelliert und mit Wellendigitalfiltern (für ein zeitdiskretes System) simuliert. Insgesamt wurden 251 Kanäle bei einem Abstand der Kanalresonanzfrequenzen von 0,1 Bark vorgesehen. Zur Herausbildung der typischen Abstimmkurven wird ein aktives Modell der äußeren Haarzellen verwendet, wobei eine laterale Kopplung der Aktivität mit insgesamt 32 benachbarten Frequenzkanälen existiert. Das Modell der inneren Haarzellen enthält ein einstufiges Tiefpaßfilter zur Nachbildung der hier stattfin-

[31] RAM - *Random Access Memory.*

denden Ausgleichsvorgänge. Vor der Detektion der Wahrnehmungschwellen werden weitere Stufen und Funktionseinheiten im Nachverarbeitungsblock zusammengefaßt. Mit der Addition von weißem Rauschen wird zunächst die Ruhehörschwelle des Gehörs berücksichtigt. Durch eine zeitliche Spreizung wird anschließend die Vorverdeckung und mit weiteren Tiefpässen die Nachverdeckung simuliert.

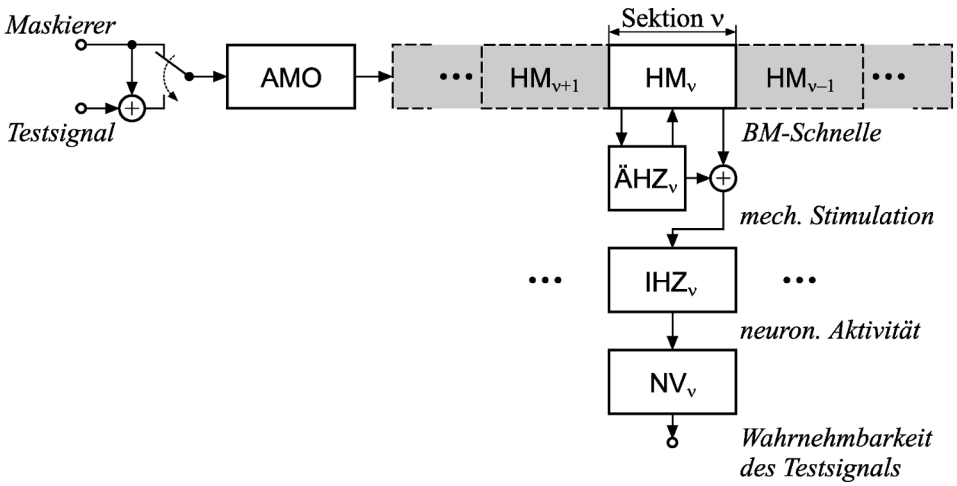


Abb. 3.26: Genereller Aufbau des Modells nach [Baumgarte 2000]. Dem Außen- und Mittelohrmodell (AMO) folgt die 251-stufige spektrale Signalzerlegung im Filterkettenmodell für die Hydromechanik bzw. die Basilarmembranschwingung im Innenohr (HM). Anschließend werden die aktiven Prozesse für jede Stufe ν mit einem rückgekoppelten Filter zur Nachbildung der äußeren Haarzellen (ÄHZ) dargestellt. Die neuronale Wandlung findet in den inneren Haarzellen (IHZ) statt, wobei nach einer weiteren Stufe neuronaler Verarbeitung (NV) der Modellausgang für den Kanal ν folgt. Strukturell besteht damit eine große Ähnlichkeit zu früheren Arbeiten von [Zwicker 1986] oder [Kates 1991].

Gegeben durch die feine Frequenzauflösung und die Berechnung der Wahrnehmungsschwellen bewirkt der sehr hohe Rechenaufwand, daß die Modellsimulation auf einem PC mit 400 MHz bei einer Abtastfrequenz von 100 kHz um etwa den Faktor 40 langsamer als Echtzeit ist.

Binaurale Modelle

Alle beschriebenen Modellvarianten waren zunächst monaurale Ansätze. Soll die Interaktion und zentrale Verarbeitung zwischen linkem und rechtem Ohr und insbesondere die Kodierung und Auswertung der Richtungsinformation berücksichtigt werden, müssen zwei monaurale Modelle in geeigneter Weise zu einem binauralen Gesamtmodell verkoppelt werden. Diese erfordern wesentlich mehr als den doppelten Rechenaufwand, da die interauralen Pegel- und Zeitdifferenzen auszuwerten sind. Ein Beispiel für die Anwendung eines binauralen Software-Modells zur Sprachqualitätsschätzung findet sich bei [Hauenstein 1997]. Hier werden beide Signalvarianten von einer Polyphasen-Allpaß-Filterbank frequenzgruppengerecht spektral zerlegt. Nach einer Hüllkurvenextraktion erfolgt eine Lautheitsdichteberechnung, die sowohl eine Überlagerung (Frequenzverschmierung) zwischen den Frequenzkanälen als auch eine Kom-

pression und Nachverdeckung (Zeitverschmierung) leistet. Nachfolgende Stufen berechnen aus diesen Signalverläufen ein Maß für die objektive Verständlichkeit eines Spracheingangssignals. Darüber hinaus sei hier noch auf die binaurale Variante „BAIM“ des Auditory Image Models von [Francis und Anderson 1997] und der Cocktail-Party-Prozessor von [Bodden 1993] als Vorverarbeitung zur Spracherkennung verwiesen.

3.4 Das Oldenburger Perzeptionsmodell

3.4.1 Motivation und Struktur

Wie sich anhand der bisher beschriebenen Modellansätze bereits andeutet, ist die Abbildung der vollständigen expliziten neuronalen Verarbeitung viel zu komplex, um einen experimentellen und praktischen Einsatz zu ermöglichen. Genau hier setzt ein auf physiologischen und psychoakustischen Daten beruhendes binaurales Modell der effektiven Signalverarbeitung an [Dau et al. 1996a, Dau et al. 1996b] (Abb. 3.27).

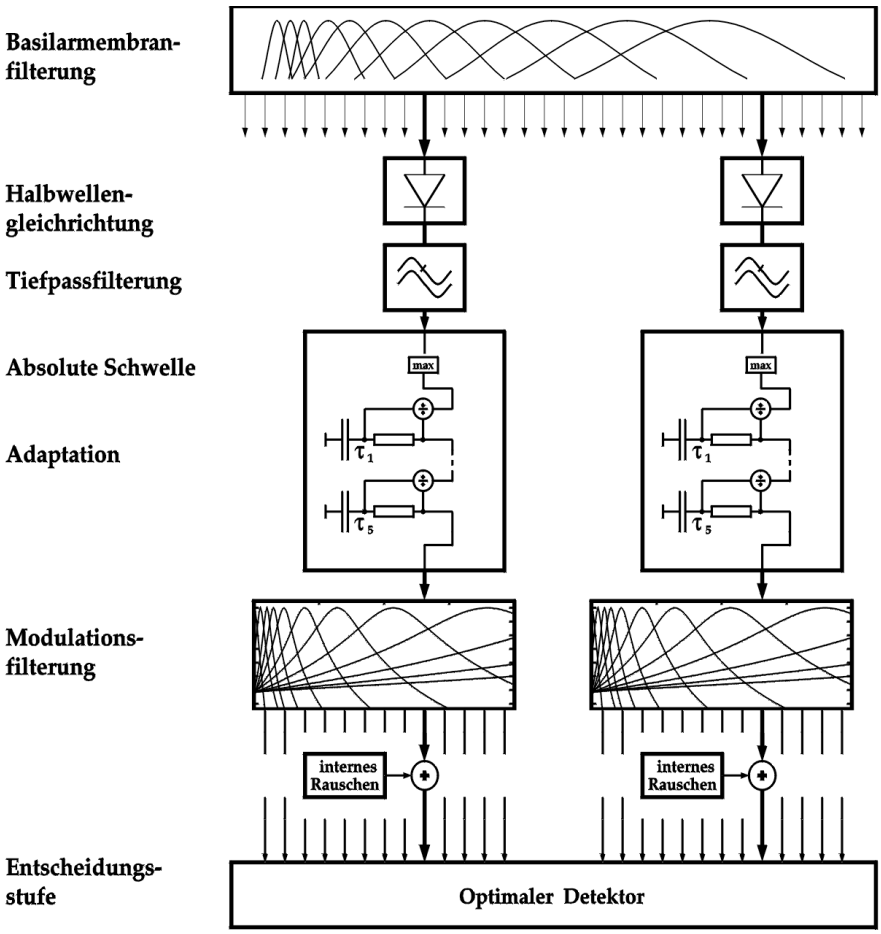


Abb. 3.27: Binaurales Perzeptionsmodell nach [Dau et al. 1996a, Dau et al. 1996b].

Für einzelne Modellstufen besteht eine physiologische Motivation, der Schwerpunkt liegt aber auf der effektiven Berechnung einer „internen Repräsentation“: ein von einem (akustischen) Zeitsignal ausgelöstes Aktivitätsmuster auf einer abstrakten hohen Stufe des auditorischen Systems. Modellumfang und -parameter wurden anhand psychoakustischer Experimente bestimmt. Die entstandene Struktur (siehe Abb. 3.27) wurde innerhalb der letzten Jahre in der Arbeitsgruppe „Medizinische Physik“ an der Universität Oldenburg entwickelt und mit unterschiedlichen Ausbaustufen in Anwendungen wie Sprachgütemodellierung, Spracherkennung und Hörgerätealgorithmen validiert. Nachfolgend sollen die Bestandteile des Modells kurz vorgestellt werden, die für die Hardware-Implementierung von Bedeutung sind.

Die Basilarmembran-Filterung und damit die Unterteilung in frequenzgruppenbreite Bänder wird mit einer linearen Gammatone-Filterbank modelliert. Diese enthält 30 Bandpaß-Kanäle, wobei die Mittenfrequenzen zwischen 73 und 6.681 Hz äquidistant auf der ERB-Skala verteilt sind. Auf der linearen Frequenzachse haben die Filter einen symmetrischen Frequenzgang, der den Mithörschwellen (neuronalen Abstimmkurven) angepaßt wurde. Eine ursprünglich von [Patterson et al. 1987] vorgeschlagene Implementierung wurde von [Hohmann und Hansen 1997] als Vierfach-Iteration eines komplexen Tiefpaßfilters erster Ordnung in der allgemeinen Form nach Gl. (3.1) angenähert[32].

$$\begin{aligned} y_{FB}[k] &= \tilde{a} \cdot y_{FB}[k-1] + b \cdot x[k] \qquad \tilde{a}, x[k], y_{FB}[k] \in C, b \in R \\ b &= 1 - |\tilde{a}| \end{aligned} \qquad \text{Gl. (3.1)}$$

Die aktiven (Verstärkungs-)Prozesse im Corti’schen Organ, also das Wirken der äußeren Haarzellen, werden an dieser Stelle nicht berücksichtigt. Entsprechende Arbeiten mit einer nichtlinearen Gammatone-Filterbank finden sich z. B. bei [Kollmeier et al. 1998]. Hier wird das Perzeptionsmodell zur Modellierung von Hörschäden adaptiert. Die Funktionsweise der inneren Haarzelle, die Umwandlung der mechanischen Auslenkung (in nur einer Richtung) in Rezeptorpotentiale, wird dagegen mit der nachfolgenden kanalweisen Halbwellen-Gleichrichtung abgebildet. Aus dem Ausgang eines Filterbankkanals $y_{FB}\,[k]$ entsteht nach Gl. (3.2) der gleichgerichtete Wert $y_{HG}\,[k]$:

$$y_{HG}[k] = \begin{cases} y_{FB}[k] & y_{FB} > 0 \\ 0 & sonst \end{cases} \qquad \text{Gl. (3.2)}$$

Da die Haarzellen und Spiralganglien nur bis zu einer Grenzfrequenz der Erregung synchron folgen können (Phasensynchronizität) und danach gewissermaßen die Hüllkurve übertragen,

[32] Die nachfolgenden Gleichungen zur Beschreibung des Oldenburger Perzeptionsmodells, Gl. (3.1) bis Gl. (3.11), sind in Form allgemeiner Filter-Differenzengleichungen notiert, d. h. *x[k]* ist die Folge des zu diskreten Zeitpunkten *k* abgetasteten Eingangssignals, *y[k]* die entsprechende Ausgangsfolge und *a* und *b* sind die Filterkoeffizienten. Die tatsächlichen Werte für Konstanten und Parameter werden in Kap. 4.1.2 im Zusammenhang mit einer Ressourcenanalyse aufgeführt.

wird ein Tiefpaßfilter erster Ordnung, Gl. (3.3), mit einer Grenzfrequenz von 1 kHz und den Filterkonstanten a_h und b_h vorgesehen.

$$y_{TP}[k] = a_h \cdot y_{HG}[k] - b_h \cdot y_{TP}[k-1] \qquad \text{Gl. (3.3)}$$

$$y_{DS}[k] = \begin{cases} y_{min} & y_{TP} \le y_{min} \\ y_{TP}[k] & sonst \end{cases} \qquad \text{Gl. (3.4)}$$

Der Filterausgang $y_{TP}\,[k]$ wird anschließend nur für Werte größer als die untere Schranke y_{min} in den Ausgang $y_{DS}\,[k]$ übernommen (Gl. (3.4)). Dieses Vermeiden kleiner Werte dient der Implementierung einer absoluten Detektionsschwelle und andererseits der Beruhigung der nachfolgenden Regelkreise im Falle eines sprungförmigen Eingangssignals. Die entlang des gesamten auditorischen Systems, insbesondere aber in den Hörnervenfasern, meßbaren zeitlichen Adaptations- und Kontrastierungsvorgänge, werden durch eine Hintereinanderschaltung von fünf nichtlinearen Adaptationsschleifen [Püschel 1988] realisiert. Eine physiologische Entsprechung gibt es hierfür nicht; vielmehr ist es die dem gesamten Hörsystem innewohnende Betonung der zeitlichen Signaländerungen, die auf eine solche Regelkreisstruktur führt. Die Abbildung 3.28 zeigt die Reaktion des Modells auf ein Rauschsignal mit einer rechteckförmigen Einhüllenden. Während Signaländerungen nahezu linear übertragen werden, erfolgt bei stationären Signalen eine logarithmische Dynamikkompression. Die nichtlineare Regelung wird dadurch erreicht, daß der Eingang eines Kreises durch den tiefpaßgefilterten Ausgang dividiert wird. Der einstufige Tiefpaß bestimmt mit seiner Zeitkonstante das zeitliche Antwortverhalten des Regelkreises. Die Form des rückgekoppelten Dividierers entstand aus der Näherung für die logarithmische Transformation, die bei der (Pegel-)Wahrnehmung (Weber-Fechnersches Gesetz) beobachtet wird. Angenähert durch eine Potenzfunktion und in fünffacher Iteration bei steigenden Zeitkonstanten erfüllen die Nachregelschleifen sowohl diese logarithmische Transformation stationärer Pegel, als auch die Abweichung für kurze Testtöne bzw. Ein- und Ausschaltvorgänge: hierbei werden sich nicht alle Kreise im eingeschwungenen Zustand, sondern zum Teil noch im Über- bzw. Unterschwingen befinden, also eine nichtlogarithmische, nahezu lineare Abbildung verursachen.

Für eine Stufe gilt allgemein (bei der Frequenz $f = 0$):

$$y[k] = \sqrt{x[k]} \qquad \text{Gl. (3.5)}$$

und für die Reihenschaltung aus fünf Stufen:

$$\text{[illegible}[k] = \sqrt[2^5]{x[k]} \approx \log(x[k]) \qquad \text{Gl. (3.6)}$$

Die Adaptationsvorgänge verursachen die dem Gehör innewohnenden zeitlichen Verdekkungseffekte, wobei im Modell nur die Nachverdeckung berücksichtigt wird. Mit Hilfe von Nachverdeckungsexperimenten wurden auch die Zeitkonstanten τ_i der Tiefpässe in den fünf Kreisen bestimmt. Ein weiterer Parameter ist der Schwellwert thr_i für den Divisor, der eine Division durch Null bei noch gänzlich ungeladenen Tiefpässen verhindern soll. Entsprechend der Potenzfunktions-Beziehung wird er für den stationären Fall nach Gl. (3.7) berechnet.

$$thr_i = \sqrt[i]{y_{min}} \qquad i = \{1, 2, 3, 4, 5\} \qquad \text{Gl. (3.7)}$$

Damit lautet die Gesamtübertragungsfunktion für die Nachregelschleifen für *i={1, 2, 3, 4, 5}* zur Berechnung des Ausgangs der letzten Schleife y_{NR} Gl. (3.9), wobei der Eingang der ersten Schleife $y_0[k]=y_{DS}[k]$ ist:

$$y_{NR}[k] = \prod_{i=1}^{5} \frac{y_{i-1}[k]}{tmp_i[k]} \qquad \text{Gl. (3.8)}$$

Der Divisor $tmp_i[k]$ berechnet sich nach Gl. (3.9) entweder aus der tiefpaßgefilterten Version des Quotienten der vorherigen Abtastschritte oder ist gleich dem Schwellwert thr_i ist. Die o. g. Zeitkonstanten τ_i bestimmen dabei die Filterkonstanten a_i und b_i.

$$tmp_i[k] = \begin{cases} a_i y_i[k-1] + b_i y_i[k-2] & (a_i y_i[k-1] + b_i y_i[k-2]) > thr \\ thr_i & sonst \end{cases} \qquad \text{Gl. (3.9)}$$

Die Abbildung 3.28 zeigt anschaulich, wie sich die fünf Nachregelschleifen beim Eintreffen eines (Rausch-)Signals verhalten.

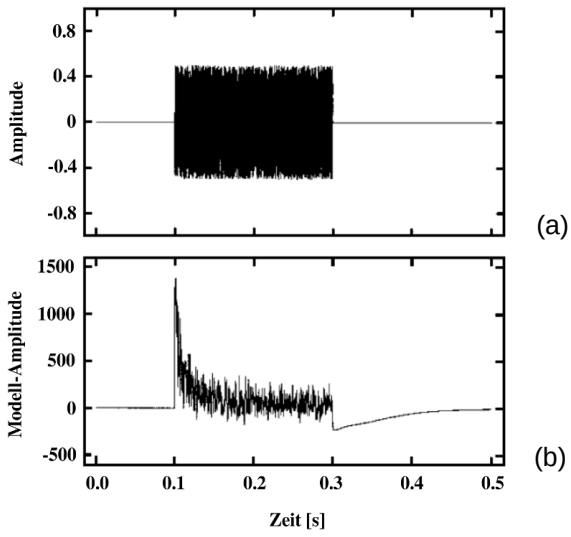


Abb. 3.28: Ausgang der Nachregelschleifen in Modelleinheiten (b) auf ein 200 ms langes Rauschsignal (a) [Dau 1999].

Das plötzliche Einsetzen des Signals auf die ungeladenen Schleifen verursacht die Maximalantwort (Division durch die Schwellwerte), wie sie auch bei phasensynchronen instantanen Rezeptorpotentialen in Hörnervenfasern beobachtet werden. Danach erfolgt die Einschwingphase entsprechend den Zeitkonstanten der Tiefpässe und die Ausgangsamplitude nimmt exponentiell ab. Endet der Reizton, befinden sich die Schleifen noch in einem aktiven Zustand; die Tiefpässe entladen sich mit einer auf der logarithmischen Skala linearen Charakteristik. Gleichzeitig wird verhindert, daß neu einsetzende Signale zu einer Aktivität am Ausgang beitragen; sie werden „nachverdeckt“.

Die in Abb. 3.27 dargestellte nachfolgende Modulationsfilterbank berechnet für jeden Frequenzkanal das Spektrum der hier auftretenden Modulationsfrequenzen. Das danach zusätzlich aufgebrachte „interne Rauschen“ modelliert statistische Übertragungsfehler des Nervensystems und die Auswirkungen der Ruhehörschwelle. Mit Hilfe der abschließenden Entscheidungsstufe (Kreuzkorrelations-Detektor) kann die aktuell berechnete interne Repräsentation sofort mit Referenzmustern verglichen werden. Um das Modell aber in mehreren verschiedenen Applikationen einsetzen zu können (siehe Kap. 3.4.2), werden in der für die Hardware-Implementierung vorgesehenen Modellversion nur noch zwei weitere Funktionseinheiten nach den Adaptationsschleifen vorgesehen.

Der Ausgang y_{NR} der Nachregelschleifen wird nun linear auf Modellamplituden y_{MOD} mit der allgemeinen Gl. (3.10) umgerechnet, wobei für das minimale Eingangssignal y_{min} am Ausgang 0 Modelleinheiten berechnet werden.

$$y_{MOD}[k] = m \cdot y_{NR}[k] + n \qquad \text{Gl. (3.10)}$$

Die Sensitivität der Neurone für Modulationsfrequenzen nimmt in Richtung höherer Verarbeitungsstufen ab; ihr Tiefpaßcharakter nimmt damit zu, was sich in einer abnehmenden Grenzfrequenz äußert. Im Bereich der Hörrinde werden noch Modulationen von etwa 4 Hz wahrgenommen, was der Silbenfrequenz entspricht. Deswegen folgt eine einstufige, mit einer Zeitkonstante von $\tau = 20\ ms$ empirisch parametrisierte, Tiefpaßfilterung (Filterkonstanten a_o und b_o) nach Gl. (3.11), die diese Begrenzung des Auflösungsvermögens für zeitliche Einhüllendenfluktuationen nachbilden soll. Der Ausgang y_{OUT} des Modells an dieser Stelle lautet demnach:

$$y_{OUT}[k] = a_o \cdot y_{MOD}[k] + b_o \cdot y_{OUT}[k-1] \qquad \text{Gl. (3.11)}$$

Der Modellausgang y_{OUT} bezeichnet nun eine interne Repräsentation akustischer Signale ähnlich einem Spektrogramm, berücksichtigt allerdings durch die Komprimierung und Kontrastierung in den Adaptationsschleifen den zeitlichen Verlauf der Intensitätsabbildung im peripheren

auditorischen System. Die Abbildung 3.29 stellt eine typische Zeit-Frequenzdarstellung des Modellausgangs dar.

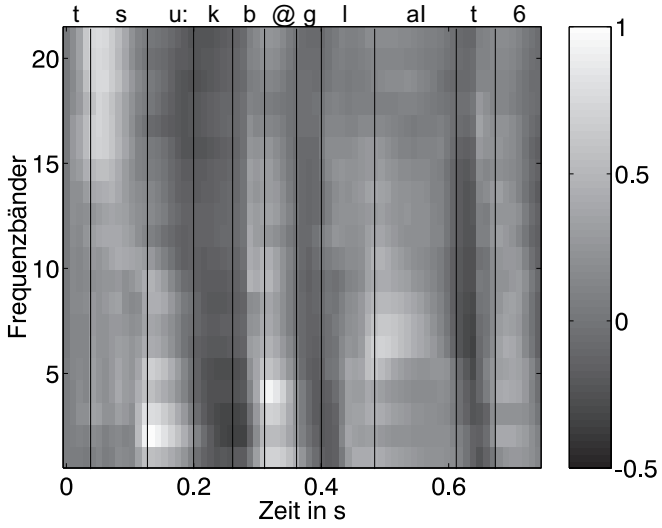


Abb. 3.29: Interne Repräsentation des Wortes „Zugbegleiter“ in den Frequenzbändern 1 bis 21. Die Farbabstufungen bezeichnen normierte Beträge am Ausgang der Nachregelschleifen. Oben ist die Unterteilung der Phonemfolge abgetragen, wie sie z. B. für die Spracherkennung von Bedeutung ist [Hartmann 2000].

Damit steht der Gesamtumfang des für die Hardware-Realisierung vorgesehenen Algorithmus’ fest (Abb. 3.30). Die Filterbank wird von vornherein für eine binaurale Verarbeitung implementiert, während die Adaptationsschleifen aufgrund des auf den ersten Blick hohen Verarbeitungsumfanges zunächst zweifach vorgesehen werden.

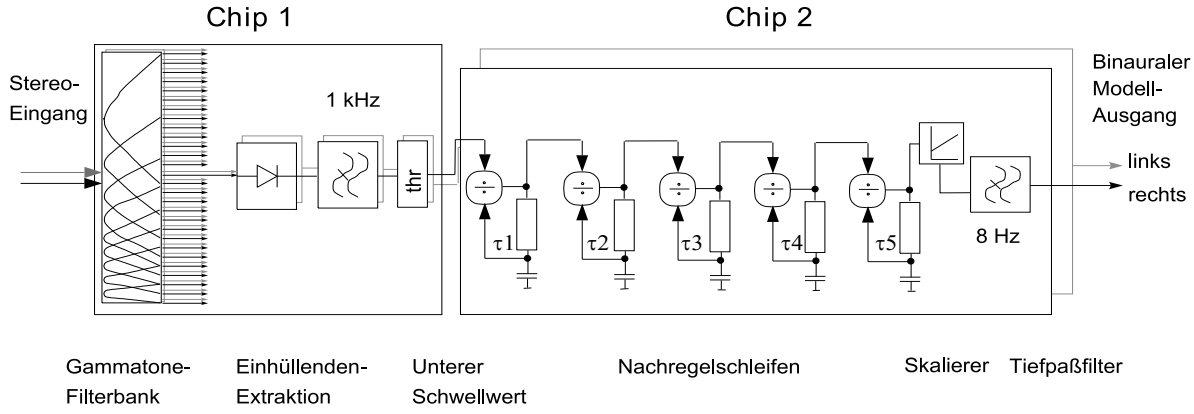


Abb. 3.30: Umfang des Perzeptionsmodells für die Hardware-Realisierung. Hierfür wird eine 30-kanalige Version in binauraler Ausführung spezifiziert.

Mit diesem Modell konnte eine Vielzahl verschiedener psychoakustischer Experimente quantitativ modelliert werden (z. B. Simultan-, und Nachverdeckung, Maskierung im Frequenzbe-

reich, Testtonintegration, Amplitudenmodulationsdetektion und -maskierung, Sprachwahrnehmung bei Normal- und Schwerhörigen unter Störgeräusch) [Kollmeier 1998]. Die meisten in der Literatur beschriebenen Modelle zur Signalverarbeitung im auditorischen System beschränken sich auf eine kleinere Zahl von zu beschreibenden Phänomenen bzw. auf spezielle physiologische Effekte, weisen jedoch strukturelle Ähnlichkeiten mit dem o. g. Modell auf.

3.4.2 Anwendungen

Im folgenden sollen die vorrangigen Zielanwendungen der Hardware-Version des Oldenburger Perzeptionsmodells: Spracherkennung, Sprachgütemodellierung und digitales Hörgerät dargestellt werden. In allen Fällen läßt sich die gehörgerechte Vorverarbeitung als Signalverarbeitungs-Modul ausgliedern und kann in einem entsprechend konfigurierten gemischten Hardware-Software-System als Koprozessor eingesetzt werden. Dabei liefert das Perzeptionsmodell im normalen Fall Ausgangsdaten an die entsprechenden Anwendungen, kann aber auch Bestandteil einer umfangreicheren Vorverarbeitung werden.

Robuste Spracherkennung

Die Ausnutzung der „internen Repräsentation“ des Perzeptionsmodells als Merkmalsvektor zur Spracherkennung wird von [Tchorz et al. 1996, Tchorz et al. 1997] und in Verbindung mit verschiedenen Störgeräuschunterdrückungsalgorithmen bei [Kleinschmidt et al. 1998, Kleinschmidt 1999] beschrieben (Abb. 3.31).

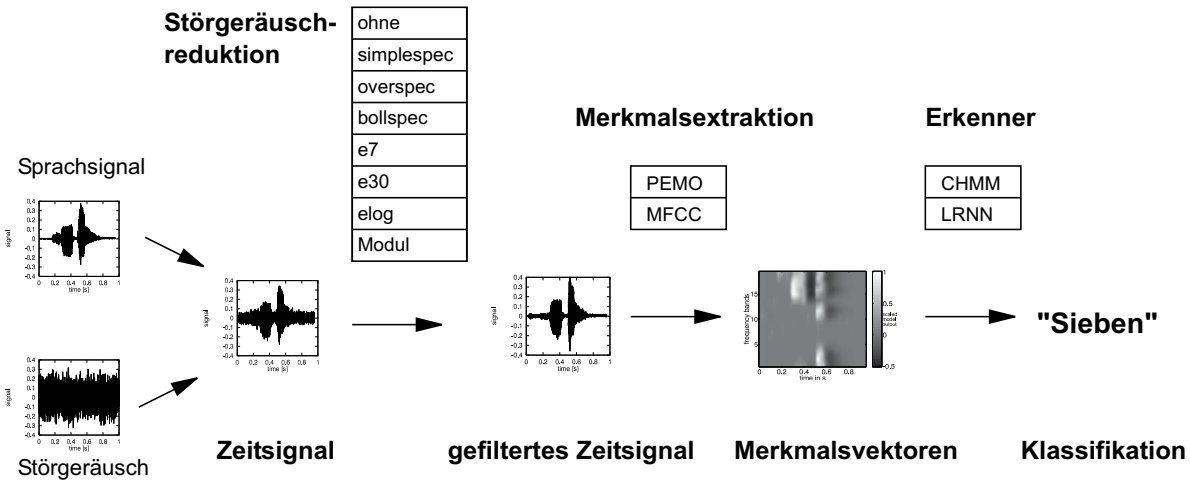


Abb. 3.31: Aufbau des Spracherkenner-Testsets mit unterschiedlichen Störgeräuschunterdrückungsverfahren: unverarbeitet (*ohne*), mit spektraler Subtraktion (*simplespec, overspec, bollspec*), nach Ephraim/Malah (*e7, e30, elog*) und bei Modulationsfilterung (*modul*); zur Anwendung in den Spracherkennungsstrategien: Lokales Rekurrentes Neuronales Netz (*LRNN*) und Kontinuierliche Hidden-Markov Modelle (*CHMM*) bei unterschiedlicher Vorverarbeitung mit dem Perzeptionsmodell (*PEMO*) oder Mel-Cepstralkoeffizienten (*MFCC*) [Kleinschmidt 1999].

Dazu wird das Modell um ein Vorbetonungsfilter (Präemphase), einen Hochpaß erster Ordnung, vor der Gammatone-Filterbank ergänzt. Bei der durch Abtastung und Filterbank begrenzten Maximalübertragungsfrequenz des Modells ergibt sich mit dem Hochpaß eine dem Gehörgang des Menschen ähnliche Betonung mittlerer Frequenzbänder (300 - 4.000 Hz). Zusätzlich wird eine 10 ms-Mittelwertbildung zur Gewinnung des Merkmalsvektors eingesetzt.

Grundsätzlich besitzt das Perzeptionsmodell die Eigenschaft, auch bei Rauschen eine von der Merkmalsextraktion verwertbare „interne Repräsentation“ des Sprachsignals bereitstellen zu können. Wird die Vorverarbeitung mit dem Perzeptionsmodell zusätzlich durch eine Störgeräuschunterdrückung ergänzt, kann die Stabilität der Merkmalsgewinnung noch weiter erhöht werden.

Abbildung 3.32 zeigt beispielhaft die Zeit-Frequenzdarstellungen des Perzeptionsmodells bei unterschiedlich verrauschtem Eingangssignal.

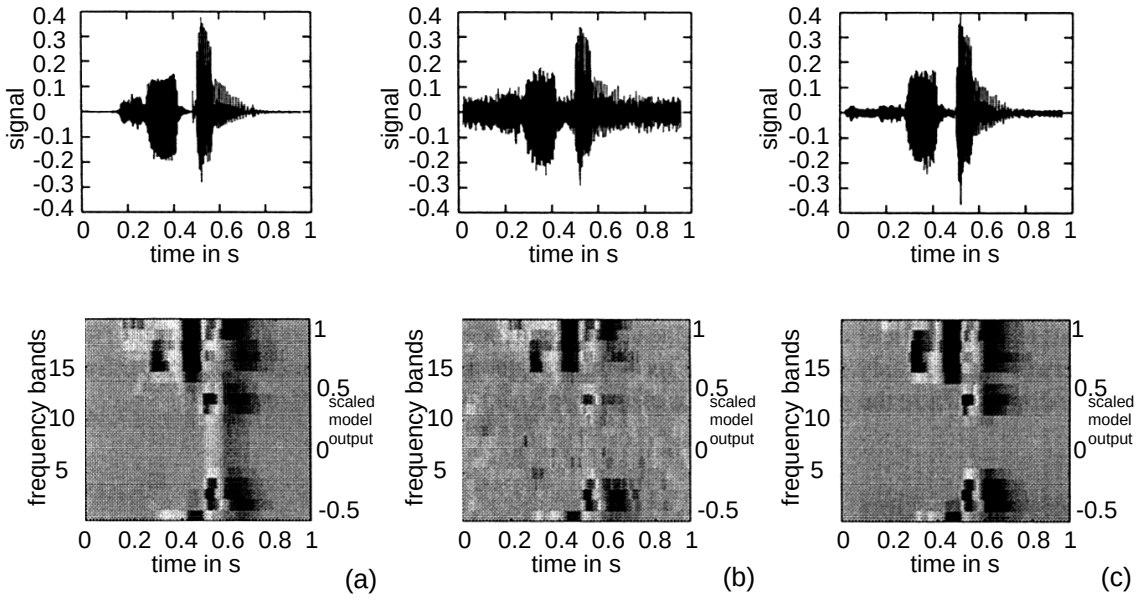


Abb. 3.32: Zeitsignal und Merkmalsvektoren des Wortes „Sieben“ (männlicher Sprecher) unverrauscht (a), 5 dB SNR bei CCITT-Rauschen (b) und (c) mit einer Störgeräuschunterdrückung nach Ephraim und Malah (Gleichung 7) [Kleinschmidt 1999].

Aus Abbildung 3.33 geht die signifikante Überlegenheit einer Einzelwort-Spracherkennung mit einem Lokalen Rekurrenten Neuronalen Netzwerk (LRNN) bei Vorverarbeitung mit dem Perzeptionsmodell (PEMO) und CCITT-Normrauschen[33] gegenüber der alternativen Erkennung mit kontinuierlichen Hidden-Markov-Modellen hervor.

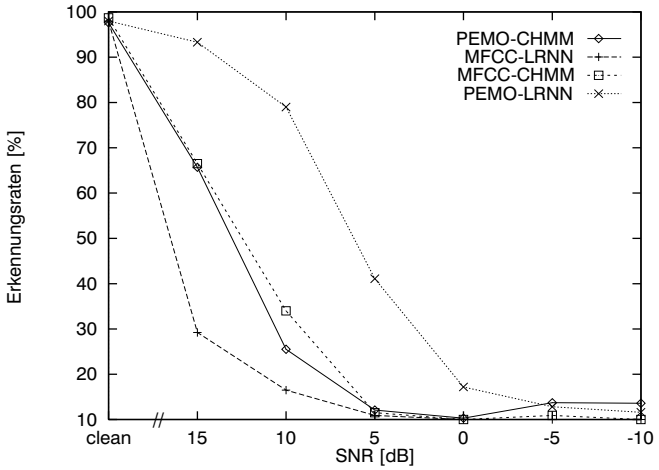


Abb. 3.33: Erkennungsleistung der beiden Spracherkenner LRNN (Lokales Rekurrentes Neuronales Netz) und CHMM (Kontinuierliche Hidden-Markov-Modelle) bei unterschiedlicher Vorverarbeitung mit PEMO (Perzeptionsmodell) oder MFCC (Mel-Cepstralkoeffizienten) [Kleinschmidt 1999].

Sprachqualitätsmessung

Die instrumentelle Sprachqualitätsmessung und -vorhersage liefert ein objektives Gütekriterium zur Beurteilung der Übertragungsqualität von Kommunikationssystemen. Dabei wird der Unterschied zwischen einem Referenzsignal und der durch die Übertragung verzerrten Version dieses Referenzsignals mit Hilfe physikalischer Parameter quantifiziert.

Insbesondere bei nichtlinearen Sprach-Kodier-Dekodier-Systemen (Codec[34]) haben sich klassische Signal-Rausch-Abstandskriterien als dafür ungeeignet erwiesen [Hansen und Kollmeier 1995, Berends und Stemerdink 1994]. Wird jedoch die Qualitätsbestimmung einschließlich einer gehörgerechten Vorverarbeitung beider Signale mit einem Perzeptionsmodell kombiniert, kann eine valide Aussage über den Grad der Verzerrungen und deren Auswirkung auf die Sprachverständlichkeit getroffen werden [Vary et al. 1998][35].

[33] CCITT - *Comitee Consultativ Internationale de Telegraphique et Telephonique*, sprachsimulierendes Rauschen, unmodelliert mit sprachähnlichem Spektrum, Norm G.227.

[34] Ein Codec ist ein kombinierter **Co**dierer/**Dec**odierer. Elektronische Bausteine, die sowohl AD- als auch DA-Wandler enthalten, werden ebenfalls als Codecs bezeichnet.

[35] Siehe hierzu auch vergleichbare Arbeiten zur modellbasierten Sprachqualitätsbewertung bei [Hauenstein 1997].

Für instrumentelle Sprachqualitätsvorhersage wird das Perzeptionsmodell, wie in Abb. 3.34 dargestellt, zur Berechnung der internen Repräsentation der beiden zu vergleichenden Signale eingesetzt.

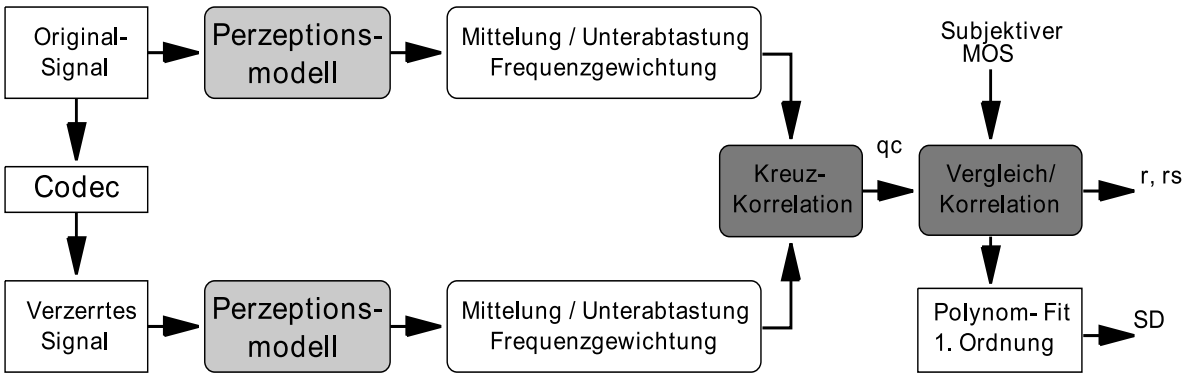


Abb. 3.34: Perzeptionsmodell bei der Verarbeitung und Vergleich der originalen und der verzerrten Version von Sprachsignalen.

Nach dem 8 Hz Tiefpaßfilter des Perzeptionsmodells erfolgt eine Unterabtastung, realisiert durch eine zeitliche Integration über 20 ms Abschnitte ohne Überlappung. Die anschließende Frequenzbandgewichtung mit näherungsweise einer 40 Phon-Isophone auf der ERB-Skala trägt der perzeptuellen Bedeutung einzelner Bänder für die Sprachqualität Rechnung. Von der durch den Codec verzerrten und der unverzerrten Version wird mit Gl. (3.12) der Kreuzkorrelationskoeffizient q_c als Qualitätsmaß bestimmt. Dieses wird aus allen korrespondierenden Abtastwerten der zweidimensionalen Zeit-Frequenzdarstellung berechnet; $X_{i,j}$ und $Y_{i,j}$ sind die Signalrepräsentationen zum Zeitpunkt $t = i*\Delta t$, $i = 1...N$ und bei der Frequenz $f = f_0 + j*\Delta f$, $j = 1...M$ (entspricht dem Filterbankkanal) mit dem jeweiligen Gewicht w_j.
$\overline{X_v}$ und $\overline{Y_v}$ sind die Mittelwerte von $w_j X_{i,j}$ und $w_j Y_{i,j}$.

$$q_C = \frac{\sum_{i,j=1}^{N,M} (w_i X_{i,j} - \overline{X}_v) \cdot (w_i Y_{i,j} - \overline{Y}_v)}{\sqrt{\sum_{i,j} (w_i X_{i,j} - \overline{X}_v)^2} \sqrt{\sum_{i,j} (w_i Y_{i,j} - \overline{Y}_v)^2}} \qquad \text{Gl. (3.12)}$$

Das Sprachmaterial wurde dem ETSI[36] Halfrate Selection Test [ETSI 1991, ETSI 1992] entnommen und besteht aus 4 Sätzen, gesprochen von zwei weiblichen und zwei männlichen Personen. Zu den durch 6 verschiedene Codecs in 6 unterschiedlichen Umgebungsbedingungen verzerrten Varianten wurden weitere 36 mit einer MNRU[37] verzerrte Sätze in die Bewertung einbezogen. Darüber hinaus wurden erfolgreiche Tests mit Material verschiedener ITU[38]-Normen (ITU-8 kHz, ADPCM[39]) durchgeführt. Um zeigen zu können, wie gut das berechnete ob-

[36] ETSI - *European Telecommunications Standards Institute.*

[37] MNRU - *Modulated Noise Reference Unit.*

[38] ITU - *International Telecommunications Union.*

[39] ADPCM - *Adaptive Differential Pulse Code Modulation.*

jektive Qualitätsmaß mit einer subjektiven Bewertung des Testmaterials übereinstimmt, wurde in einem zweiten Schritt das objektive Qualitätsmaß für jedes Testpaar mit einem subjektiven Urteil (MOS[40]) korreliert. Der subjektive MOS geht auf ACR[41]-Testreihen mit 12 männlichen und 12 weiblichen Personen zurück, die in einem einfachen Hörtest über einen Telefon-Apparat die Sprachqualität der Sprachdatensätze bei 79 dB SPL einschätzen mußten [Hansen 1998]. Das Ergebnis einer solchen zweistufigen Bewertung zeigt Abb. 3.35.

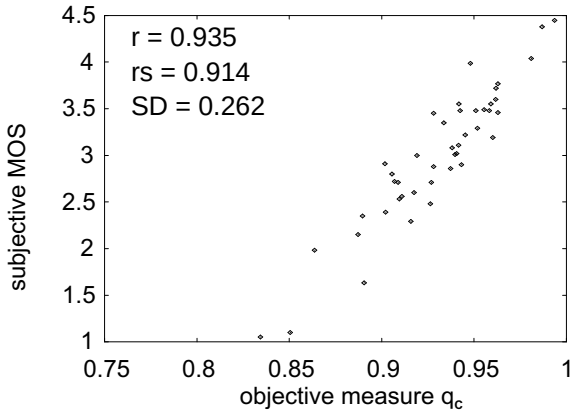


Abb. 3.35: ETSI Halfrate Selection Test. Subjektive Sprachqualität (ausgedrückt als MOS - *Mean Opinion Score*) gegen die objektive Sprachqualität q_c für die ETSI-Testdatenbasis.

Hierbei wird die hohe Korrelation des objektiven und subjektiven Qualitätsmaßes der Werteverteilung entlang einer näherungsweisen Proportionalitätsgeraden zwischen q_c und dem MOS deutlich. Der lineare Korrelationskoeffizient erreicht $r = 0.935$, der Spearman-Rangkorrelationskoeffizient $r_s = 0.914$. Die Standardabweichung der Punktwolke von einer (für jeden Test individuell durchgeführten) Polynom-Näherung ersten Grades liegt bei $SD = 0.262$ MOS-Einheiten.

Das beschriebene Verfahren eignet sich darüber hinaus auch für eine fortlaufende, kontinuierliche Beurteilung der Sprachübertragungsqualität, indem durch Skalen-Transformation und Tiefpaßfilterung ein $q_c(t)$ bestimmt wird [Hansen und Kollmeier 1998a, Hansen und Kollmeier 1998b]. Somit ist man in der Lage, eine ständige Qualitätsüberwachung in Sprach-Übertragungssystemen bereitzustellen.

[40] MOS - *Mean Opinion Score.*

[41] ACR - *Absolute Category Rating.*

Digitales Hörgerät

Wie [Kollmeier 1998, Kollmeier et al. 1998] zeigt, ist das Perzeptionsmodell in der Lage, die Ursachen typischer Auswirkungen von Innenohrschäden, wie verminderte Frequenz- und Zeitauflösung bzw. Modulationsdetektion und krankhaft veränderte Dynamikabbildung durch lokale Parametervariationen nachzubilden. Damit besteht die in Abb. 3.36 skizzierte Möglichkeit, die Ausgänge der Gammatone-Filterbank mit einer Synthese-Filterbank zu rekombinieren. Ein noch zu spezifizierender Kompensationsalgorithmus kann, auf den Ausgängen sowohl der Filterbank als auch der Nachregelschleifen beruhend, eine Gewichtungsvorschrift für die zu resynthetisierenden Kanäle errechnen. Dabei spielt die statische und dynamische frequenzgruppenbezogene Pegelregelung auf der Basis eines Lautheitsmodells eine zentrale Rolle.

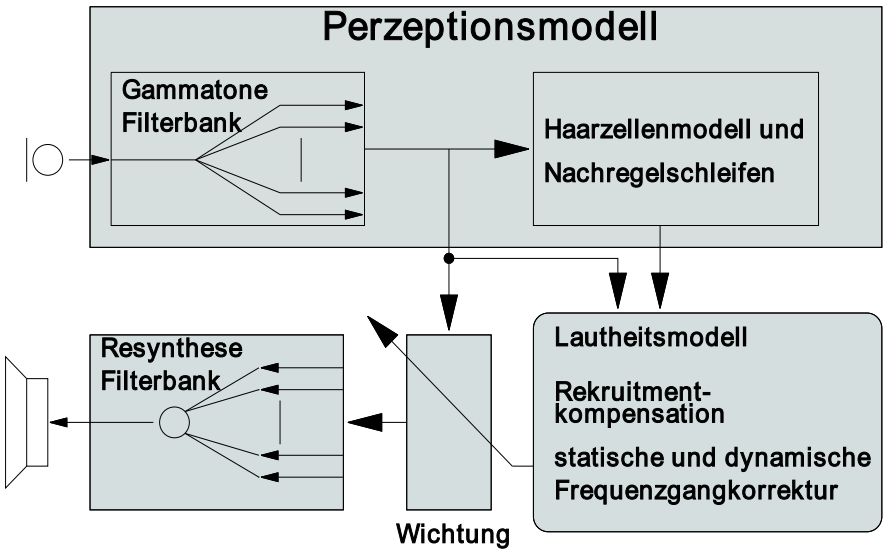


Abb. 3.36: Mögliches Prinzip für den Aufbau eines monauralen digitalen Hörgeräts durch Erweiterung des Perzeptionsmodells mit einem Lautheitsmodell, eine statische und dynamische Frequenzgangkorrektur und eine resynthetisierende Filterbank.

Diese dritte potentielle Hauptanwendung des Perzeptionsmodells erfordert damit eine Erweiterung des Verarbeitungsalgorithmus. Stellt das Modell bisher ein reines Analysesystem dar, das ein eintreffendes Mikrofonsignal auf verschiedenen Abstraktionsstufen gehörgerecht präsentiert, benötigt das Hörgerät zunächst einen resynthetisierenden Block zur Ausgabe des akustischen Signals auf einem Ohrhörer, eine Resynthesefilterbank. Zusätzlich bedarf es zwischen Analyse und Resynthese einer Verarbeitungssequenz, die eine dem individuellen Hörschaden entsprechende Anpassung bzw. Korrektur des zu resynthetisierenden Signals vornimmt. Eine wesentliche Ausdehnung im Kompensationssalgorithmus bedeutet die Erweiterung auf ein binaurales Hörgerät, denn hierfür wird ein binaurales Lautheitsmodell benötigt.

Mit der Beschreibung der Funktionsweise und der benötigten Anwendungen des Perzeptionsmodells kann nun eine gezielt auf die Hardware-Realisierung abgestimmte Systemspezifikation erarbeitet werden.

Kapitel 4

Das Perzeptionsmodell: Analyse, Adaption und Verifikation

Dieses Kapitel analysiert die Anforderungen an das aufzubauende Signalverarbeitungssystem und grenzt unter Realisierbarkeitsaspekten den Entwurfsraum so weit ein, daß eine vollständige Spezifikation als Ausgangspunkt für die Hardware-Implementation gegeben werden kann. Dabei wird der benötigte Kompromiß zwischen der Verarbeitungsleistung, dem Flächenbedarf und den Voraussetzungen des zur Schaltungssynthese verwendeten CAD-Systems gefunden. Bei dieser Anforderungsanalyse muß bereits auf den bezüglich Designvarianten und Synthetisierbarkeit ohnehin stark iterativen Hardware-Entwurfsprozeß stellenweise weit vorgegriffen werden. Andernfalls ist es möglich, daß in späteren Stadien des Entwurfs durch die Limitationen des Designsystems unüberwindliche Hindernisse entstehen.

Ausgehend von einer Spezifikation und Partitionierung des Algorithmus', einer Analyse der benötigten Ressourcen, der zu verarbeitenden Daten, den Anforderungen des Gesamtsystems und der Leistungsfähigkeit der Entwurfswerkzeuge wird eine Abschätzung und Auswahl der Operatoren und Datenformate getroffen. Zur Bewertung der Signalverzerrungen durch Quantisierung und Operationen mit begrenzten Wortlängen bzw. Vereinfachungen und Optimierungen des nichtlinearen Algorithmus werden Tests mit einer Sprachqualitätsbewertung durchgeführt.

4.1 Systemspezifikation

4.1.1 Partitionierung des Perzeptionsmodells

Die Unterteilung des Entwurfs (siehe Abb. 4.1) und der jeweiligen Untersuchungen liegt sowohl in der Arbeitsteiligkeit des zugrundeliegenden Projektes als auch in ersten Abschätzungen zum Ressourcen- und damit Flächenbedarf einzelner Module begründet. Der Chip 1 enthält neben dem linearen Teil des Modells (Filterbank) die Halbwellengleichrichtung und die Einhüllendenextraktion. Begründet durch die innere Struktur sind hier andere Methoden für den funktionalen Test und vor allem zur Beurteilung der Quantisierungsfehler erforderlich, als für die nichtlineare Verarbeitung im Chip 2. Die Abbildung 4.1 zeigt die bereits erwähnte Darstellung der Signalverarbeitung in einem der 30 Frequenzkanäle; die Schattierung deutet die

binaurale (stereophone) Verarbeitung an. Im Chip 2 erfolgt eine kombinierte Verarbeitung sowohl mit linearen einstufigen Filtern als auch mit nichtlinearen Pegelregelungen.

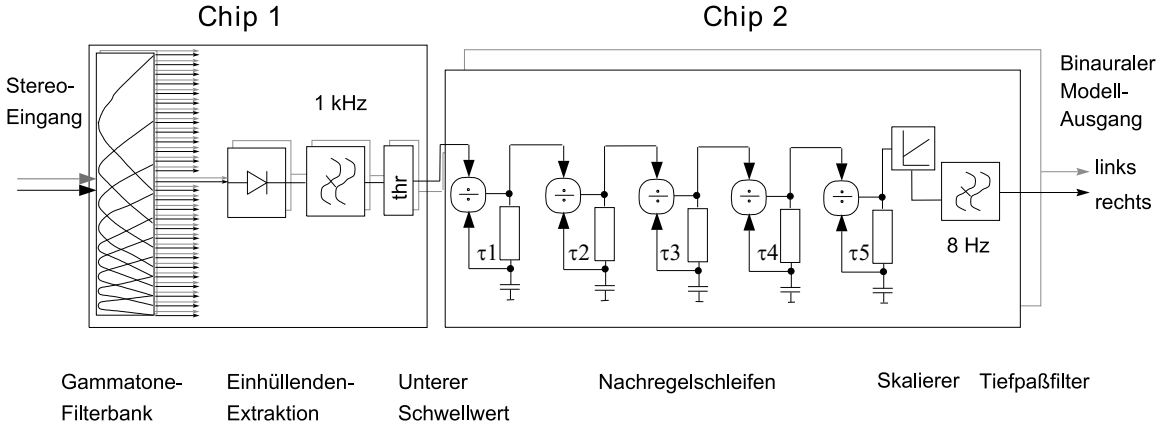


Abb. 4.1: Aufteilung des binauralen Perzeptionsmodells für den Schaltungsentwurf in einen Chip-Satz, wobei Chip 1 und Chip 2 jeweils unterschiedliche Anforderungen an die Datenpfadsynthese und die Verifikation stellen.

Darüber hinaus wird eine in Hardware-Entwürfen unmittelbar als problematisch einzuschätzende echte Division benötigt. Bedingt durch die Rückkopplung ist hier mit besonderen Problemen für die Signaldynamik zu rechnen.

Da das Chip 1-Teildesign Gegenstand des parallelen Teilprojekts ist (siehe Kap. 1), wird im weiteren nur am Rande darauf eingegangen.

4.1.2 Ressourcenanalyse

Die Struktur des originalen Perzeptionsmodells in der C-Implementation besteht aus insgesamt 30 Frequenzkanälen für jeweils den linken und rechten Kanal. Eine vollständig parallele binaurale Struktur umfaßt damit 2 x 30 Gammatone-Filter, 2 x 30 Haarzellenblöcke (Halbwellengleichrichtung, Tiefpaßfilter, unterer Schwellwert), 2 x 30 x 5 = 300 Nachregelschleifen und 2 x 30 Skalierer und 8 Hz Tiefpaßfilter. Alle Frequenzkanäle sind gleichzeitig (im Takt der Abtastfrequenz) zu berechnen. Damit besteht ein Rechenaufkommen von 42.968.640 arithmetischen Operationen je Sekunde mit IEEE[42]-Fließkomma-Operanden (*single precision*) [IEEE 1985]. Dabei ist für das Perzeptionsmodell und im speziellen für die Filterbank im Chip 1 eine Abtastfrequenz von f_A = 16.276 Hz festgelegt. Da die adaptiven Nachregelschleifen in der Regel nur langsam veränderliche Energiewerte an den Analyseausgängen liefern und für die Anwendungen eine (experimentell bestimmte) Unterabtastung von 1:4 zulässig ist, wurde für den

[42] IEEE - *Institute of Electrical and Electronics Engineers.*

Chip 2 eine Abtastfrequenz von $f_{A,Chip\ 2} = f_{A,Chip\ 1}/4 = 4.069$ Hz vorgesehen. Hierdurch summiert sich der Rechenaufwand für Chip 2 auf den in der Tab. 4.1 dargestellten Umfang.

Block	Variablen-addition (-subtraktion)	Konstanten-multiplikation	Variablen-division	Relation
Nachregelschleifen	300	600	300	300
Skalierer	60	60	0	0
Tiefpaß	60	120	0	0
gesamt	420	780	300	300
gesamt je Sekunde	1.708.980	3.173.820	1.220.700	1.220.700

Tab. 4.1: Umfang der Berechnungen im gesamten Perzeptionsmodell. Die Anzahl der Operationen je Sekunde bezieht sich auf eine Abtastfrequenz von 16.276 Hz für die Gammatone-Filterbank und den Haarzellenblock und 4.069 Hz für die Nachregelschleifen, den Skalierer und den Tiefpaß.

Die Original-Implementierung als C-Programm für eine General-Purpose CPU ist streng sequentiell und erreicht entsprechend lange Programmlaufzeiten[43]. Von einer exklusiven Hardware-Version wird erwartet, daß sie an sich bereits eine höhere Geschwindigkeit besitzt. Sie kann außerdem durch Parallelisierung die Berechnungsdauer weiter verkürzen. Folgende Varianten sind dabei denkbar:

1. vollständige sequentielle Abarbeitung: alle Operatoren sind nur einmal vorhanden und werden im Zeitmultiplex-Betrieb verwendet (minimaler Ressourcenbedarf, hohe Timing-Anforderungen an die Operatoren und das Steuerwerk, alle Operatoren sind Variablenrechenwerke)
2. vollständig parallele Abarbeitung: diese Variante kann sofort ausgeschlossen werden, denn die Summe aller Rechenwerke entspräche einer Anzahl von ALUs[44] bzw. FPUs[45] handelsüblicher CPUs: ein derartiger Flächenbedarf ist nicht zu realisieren. Die fünf Nachregelschleifen eines Kanals sind infolge der Datenabhängigkeiten ohnehin nicht parallelisierbar, sie müssen sequentiell angeordnet bleiben.
3. teilweise Parallelisierung: Filterbankkanal, Haarzellenblock, fünf Nachregelschleifen und Skalierer/Tiefpaß werden je nach Platzangebot in der gewählten Zieltechnologie mehrfach instanziert und berechnen im Zeitmultiplexverfahren jeweils eine Gruppe von Frequenzkanälen. Die Hardware einer einzelnen Nachregelschleife wird in jedem Falle im Zeitmultiplex-Verfahren von allen fünf Stufen verwendet.

[43] etwa Faktor 5 langsamer als Echtzeit mit einer Intel-Pentium-CPU (120 MHz Takt)

[44] ALU - *Arithmetic and Logic Unit.*

[45] FPU - *Floating Point Unit.*

Die Synthetisierbarkeit mit einer konkreten Zieltechnologie und der jeweils erreichbare Durchsatz in der Verarbeitung sind für die Wahl der Variante 1.) oder 3.) ausschlaggebend.

In der Abb. 4.2 ist das interne Verarbeitungsschema einer einzelnen der fünf für jeden Frequenzkanal in Reihe geschalteten Adaptationsschleifen und des Skalierers bzw. Tiefpaßfilters dargestellt, so wie es in der ursprünglichen C-notierten Version verwendet wurde. Wie daraus zu ersehen ist, kommen alle vier Grundrechenoperationen und eine arithmetische Relation vor. Die Multiplikation erfolgt ausschließlich mit Konstanten und erst ab der Subtraktion im Skalierer treten negative Werte auf, d. h. die Adaptationsschleifen allein benötigen keine vorzeichenbehaftete Arithmetik.

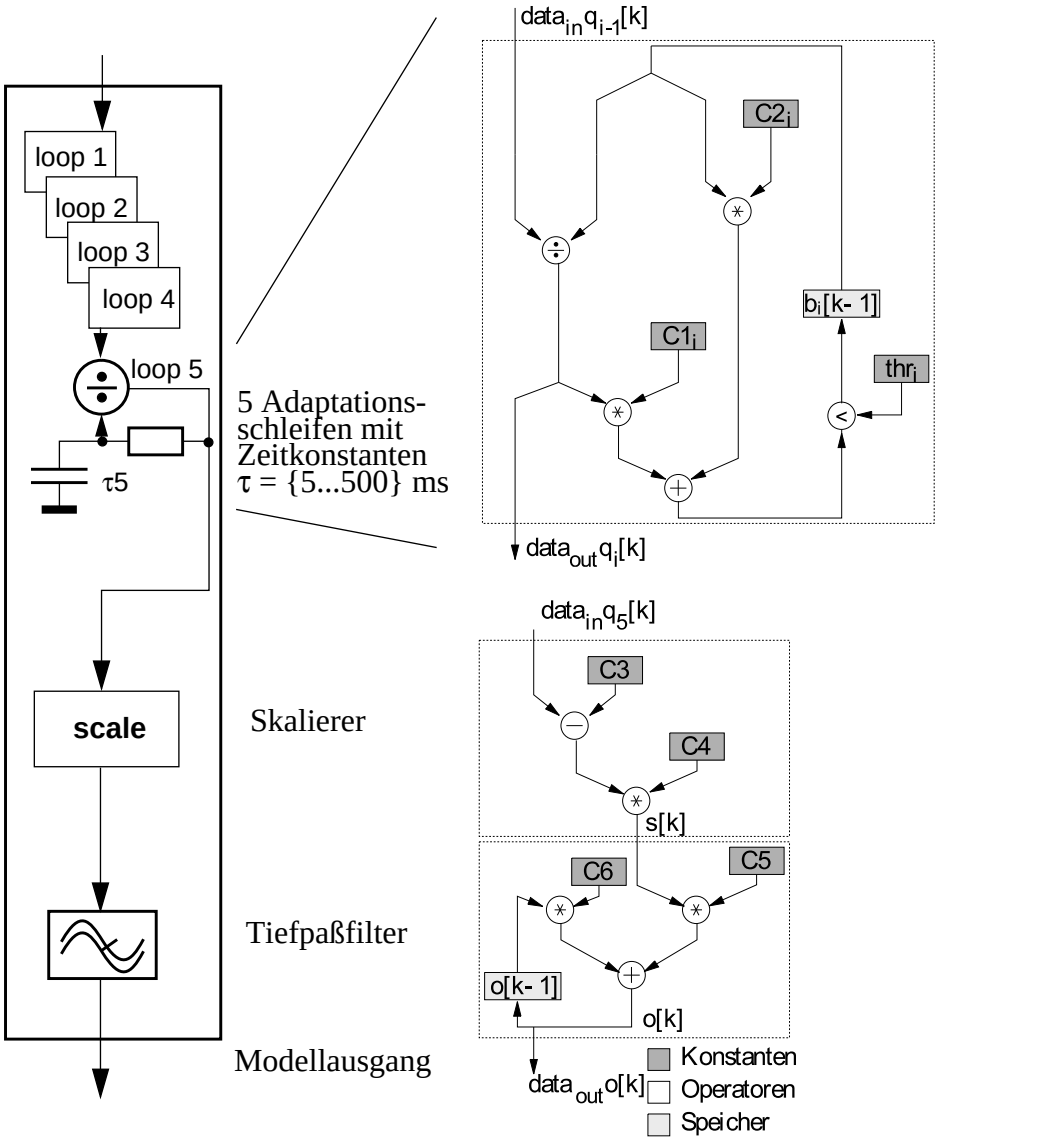


Abb. 4.2: Originaler Nachregelschleifen-Algorithmus (a) mit einstufigen IIR-Tiefpässen, einer Umskalierung und abschließendem Tiefpaß; (b) als Operationsschema.

Die Verarbeitung für die *i*-te Stufe wird mit den Differenzenfunktionen Gl. (4.1) bis Gl. (4.5) beschrieben. Demnach wird der Quotient aus dem aktuellen Dividenden und dem tiefpaßgefilterten Quotienten des letzten Systemtaktes berechnet. Der Tiefpaß ist ein einstufiges IIR-Filter mit zwei Konstantenmultiplikationen, dem die Schwellwertabfrage zur Begrenzung des Quotienten folgt. Diese Schleifenberechnung wird fünffach (aufeinander aufbauend) iteriert, wobei die Rückkopplung in den Schleifen mit dem Index zunehmend träger wird (wachsende Zeitkonstanten, siehe Kap. 3.4). Der nachfolgende lineare Skalierer dient der Berechnung sog. Modelleinheiten, die vor allem für die Spracherkennung und instrumentelle Sprachqualitätsmessung [Hansen 1998] ein direkt verwendbares Datenformat darstellen. Jede Nachverarbeitung in den Anwendungen wird damit weiter entlastet. Es gilt die einfache lineare Gleichung Gl. (4.4) und für das lineare einstufige Tiefpaßfilter die Differenzengleichung Gl. (4.5).

Die Berechnungsvorschriften und Parameter sind in der Tab. 4.2 zusammengestellt.

Operation	**Originale Berechnung**
Adapt.-Division *i* *i= [0, 1, 2, 3, 4]*	$q_0[k] = \frac{x[k]}{b_0[k-1]}$ (1. Schleife) Gl. (4.1) $q_i[k] = \frac{q_{i-1}[k]}{b_i[k-1]}$ (andere) Gl. (4.2)
Adapt.-Tiefpaß *i*	$b_i[k] = \begin{cases} C1_i \cdot q_i[k] + C2_i \cdot b_i[k-1] & bei \quad b_i[k] > thr_i \\ thr_i & sonst \end{cases}$ Gl. (4.3) wobei gilt: $C2_i = 1 - C1_i$
Skalierer	$s[k] = C3 \cdot (q_4[k] - C4)$ Gl. (4.4)
abschließender Tiefpaß	$o[k] = C5 \cdot s[k] + C6 \cdot o[k-1]$ Gl. (4.5) wobei gilt: $C6 = 1 - C5$

Tab. 4.2: Operationen in den Teilblöcken und Berechnungsvorschriften für:
x[k]: Eingangswert eines Frequenzkanals
q_i[k]: Quotient in der Adaptationsschleife *i*
b_i[k]: Ausgang des Tiefpasses in der Adaptationsschleife *i*
s[k]: Ausgang des Skalierers
o[k]: Ausgang eines Frequenzkanals nach Tiefpaßfilterung
k: Zeitinkrement, Takt
C: Konstanten (siehe Tab. 4.3)

Die nachfolgende Tabelle (Tab. 4.3) enthält die Berechnung der Filterkonstanten und Parameter für die Gleichungen in Tab. 4.2.

Parameter / Konstante	**Originale Berechnung**
Filterkonstanten im Adaptations-Tiefpaß *i* *i= [0, 1, 2, 3, 4]*	$C = 1 - e^{\frac{(-1)}{f_A \cdot \tau}}$ Gl. (4.6) bei τ_i = {0,005; 0,05; 0,129; 0,253; 0,5} [*s*] und der Abtastfrequenz f_A = 4.069 *Hz* berechnen sich die Filterkonstanten nach der allgemeinen Gleichung Gl. (4.6): $C1_0$ = 0,047963711 $C1_1$ = 0,004903153 $C1_2$ = 0,001903308 $C1_3$ = 0,001027759 $C1_4$ = 0,000491400
Divisor-Schwellwerte	$thr_i = \sqrt[2^{i+1}]{y_{min}}$ Gl. (4.7) Mit Gl. (4.7) und $y_{min}=10^{-5}$ lauten die Schwellwerte thr_i: thr_0 = 0,003162278 thr_1 = 0,056234133 thr_2 = 0,237137371 thr_3 = 0,486967525 thr_4 = 0,697830585
Konstanten im Skalierer	Der Ausgangswertebereich soll K_{range} = 100 Modelleinheiten für stationäre Modelleingangssignale (siehe Gl. (3.5) und Gl. (3.6)) betragen: $C3 = \frac{K_{range}}{(1 - \sqrt[32]{y_{min}})} = 330,940177892$ Gl. (4.8) $C4 = \sqrt[32]{y_{min}} = 0,697830585$ Gl. (4.9)
Konstanten für den abschließenden Tiefpaß	ebenfalls nach Gleichung Gl. (4.6) und mit τ = 0.02 *s* und f_A = 4.069 *Hz* lautet C5 = 0.012212842

Tab. 4.3: Berechnungsvorschriften und Werte für die Filterkonstanten und Parameter in Tab. 4.2.

Für den minimalen Eingangswert $x[k]=y_{min}$ beträgt damit der Ausgang der Nachregelschleifen $q_4[k]=C4$, womit nach Gl. (4.4) $s[k]=o[k]=0$ Modelleinheiten berechnet werden.

Jedes Filter benötigt außerdem eine Speicheradresse. Der Speicherbedarf liegt somit bei 2 x 30 x (5+1) Adressen, insgesamt also 360 Speicherworten des verwendeten Datenformats. Bei 32 Bit-IEEE-Fließkommawerten (*single precision*) entspricht das 1.440 Byte.

4.1.3 Signaldynamik

Von den zu verarbeitenden Signalen ist insbesondere die Dynamik von Interesse, da eine applikationspezifische integrierte Signalverarbeitungsschaltung die Qualitätsanforderungen der Anwendungen erfüllen muß, aber aus Kostengründen nur die minimale Datenbreite verwenden soll. Um diese bestimmen zu können, müssen Maxima und Grenzen in der Werteverteilung in typischen Anwendungsdaten untersucht werden.

Natürliche Signale weisen in vielen Fällen näherungsweise eine Gauß- oder Normalverteilung auf. Der Verteilungs(dichte)funktion (VDF) von Sprachsignalen hingegen genügen eher solchen Dichtefunktionen, die eine ausgeprägt hohe Häufigkeit nahe Null und einen exponentiellen Abfall zu größeren Amplituden hin aufweisen (siehe Abb. 4.3).

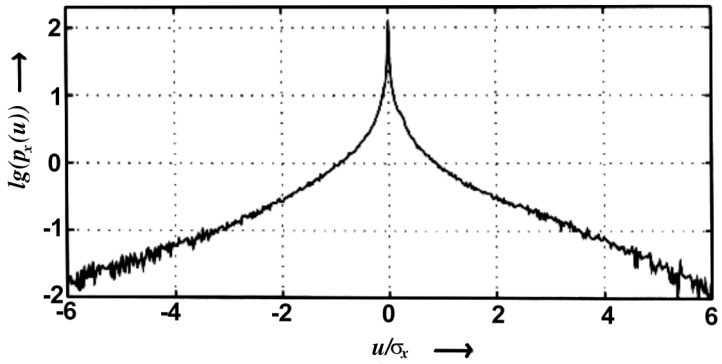


Abb. 4.3: Histogramm von Sprachsignalen nach [Vary et al. 1998] für 489.872 Werte in 1.024 Intervallen.

So findet sich in der Laplace-Verteilung Gl. (4.10), noch besser in der Gamma-Verteilung nach Gl. (4.11), eine passende Verteilungsdichtefunktion, da sie die große Häufigkeit kleiner Signale bzw. sogar Signalpausen in der Sprache modelliert (die Gamma-VDF besitzt sogar für das erste Moment, dem Mittelwert μ_x , eine Singularität) [Vary et al. 1998].

$$p_x(u) = \frac{1}{\sqrt{2\sigma_x}} e^{-\sqrt{2}\frac{|u-\mu_x|}{\sigma_x}} \qquad \text{Gl. (4.10)}$$

$$p_x(u) = \frac{\sqrt[4]{3}}{2\sqrt{2\pi\sigma_x}} \frac{1}{\sqrt{|u-\mu_x|}} e^{\left(-\frac{\sqrt{3}}{2}\right)\frac{|u-\mu_x|}{\sigma_x}} \qquad \text{Gl. (4.11)}$$

Im Histogramm nach Abb. 4.4 kann man erkennen, daß die Werteverteilung der Quotienten in allen Schleifen eine ähnliche Verteilung aufweist und außerdem Grenzwerte besitzt. Verarbeitet wurde der komplette ETSI-Sprachdatensatz (44 Datensätze, [ETSI 1992]), der als typisches Signal für alle Anwendungen des Perzeptionsmodells gelten kann und so eine hohe Validität der dadurch begründeten Dynamikbegrenzungen absichert. Die Histogramme wurden mit dem Originalperzeptionsmodell in IEEE-Fließkomma-Format (*single precision*) berechnet und zeigen, daß die Quotienten eine schleifenabhängige Obergrenze nicht überschreiten und ihre ma-

ximale Häufigkeit nahe Null haben. Bei den Divisoren zeigt sich außerdem der Einfluß des unteren Schwellwertes thr_i.

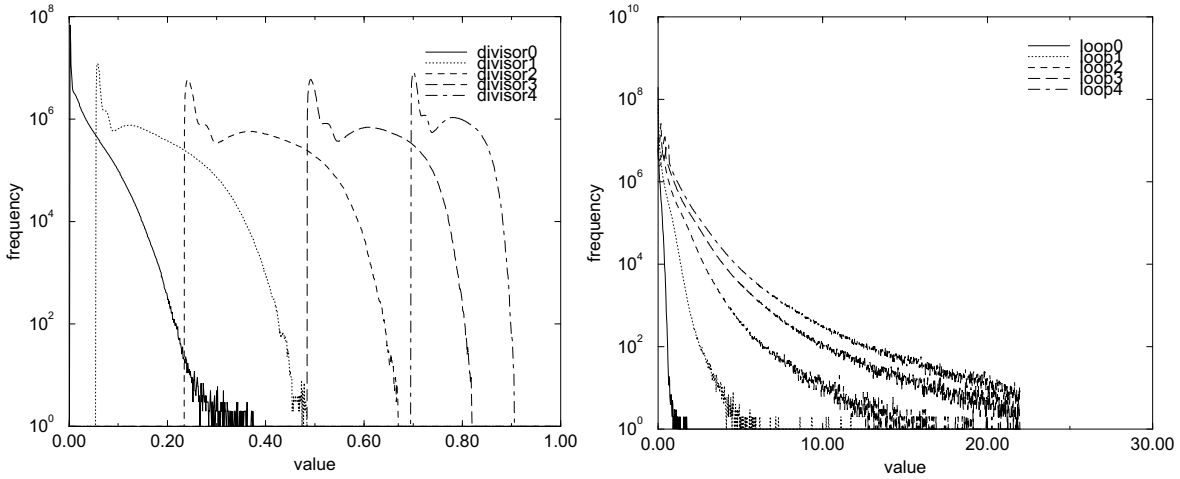


Abb. 4.4: Histogramme der in der C-Implementation der Nachregelschleifen (0 bis 4) auftretenden Werte für Quotient und Divisor für den ETSI-Testdatensatz [ETSI 1992] (vollausgesteuert). Am starken Abfall der Signalhäufigkeit an der Obergrenze bei einem Wert von 22 ist zu erkennen, daß der Testdatensatz auch von vornherein dynamikbegrenzt ist.

Zur Untersuchung von Extremwerten wurden mit einer Mischung aus acht Störgeräuschen folgende Maximalwerte für Quotienten und Divisoren in allen fünf Schleifen ermittelt (Tab. 4.4):

Schleife	**1**	**2**	**3**	**4**	**5**
Divisor	0.31	0.48	0.67	0.83	0.91
Quotient	0,93	3,81	14,73	29,3	41,2

Tab. 4.4: Maximalwerte für Störgeräusche (weißes Rauschen, Restaurant, Chirp-Signale, Industrielärm, Bohrmaschinenlärm, Straßenlärm und Druckmaschine) bei drei verschiedenen Pegeln (voll ausgesteuert, -15 dB und -30 dB) und jeweils 2 s Länge.

Bei der Verarbeitung von (auch verrauschten) Sprachsignalen und bei Vollaussteuerung weisen alle fünf Stufen jeweils mit dem Stufenindex zunehmende Maximalwerte auf. Die Division in der letzten Stufe muß die größte Dynamik bewältigen.

4.1.4 Anforderungen aus Systemsicht

Neben den bisher beschriebenen Modellkomponenten wird der Chip-Satz des Perzeptionsmodells intern um weitere Einheiten erweitert (Abb. 4.5). So enthält der Chip 1 neben der 30-kanaligen binauralen Gammatone-Filterbank und dem Modell der inneren Haarzellen, d. h. der Halbwellen-Gleichrichtung und der Tiefpaßfilterung, eine Einheit zur binauralen Lateralitäts-

bewertung. Hier werden für jeden Frequenzkanal die Amplitudenquotienten und Phasendifferenzen zwischen linkem und rechtem Kanal berechnet und über 128 Abtastwerte gemittelt [Brucke et al. 1998, Schwarz et al. 1999b].

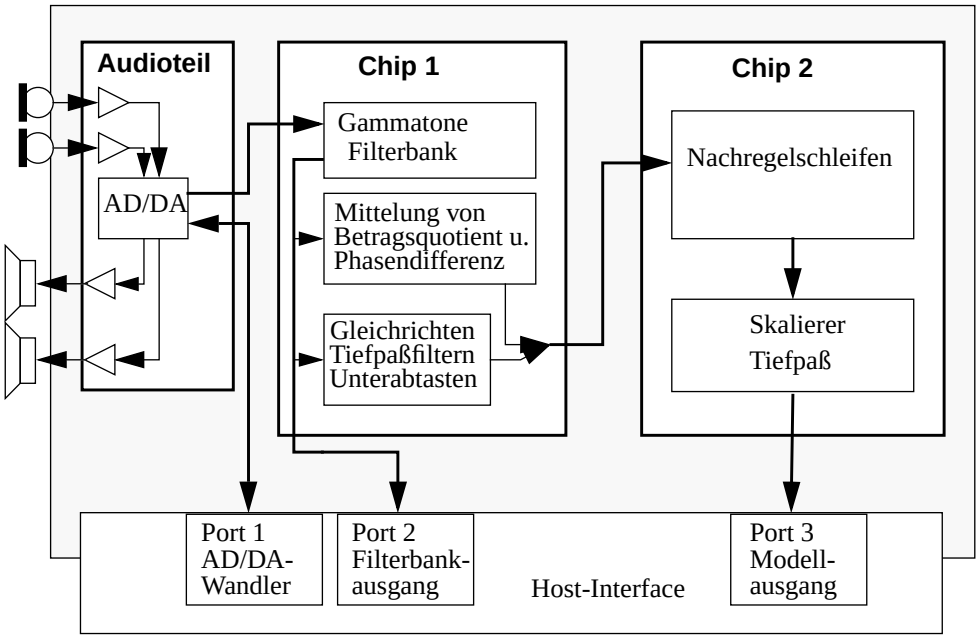


Abb. 4.5: Systemspezifikation, Schnittstellen und Anbindung an ein Host-Interface (z. B. PCI-Bus eines PC).

Damit jedoch in einem Systemaufbau der Chip-Satz für die gewünschten Anwendungen eingesetzt werden kann, müssen neben der eigentlichen Signalverarbeitung entsprechende Schnittstellen implementiert werden. Serielle Protokolle ermöglichen kleine (und billige) Chip-Gehäuse, die auch der Hörgeräteanwendung angepaßt sind, erfordern aber einen hohen Bit-Takt. Die Ein- und Ausgangsdatenformate sind an allen Schnittstellen identisch auf 16 Bit Wortbreite festgelegt. Die Datenwortbreite ergibt sich primär aus der 16 Bit-Quantisierung des A/D-Wandlers (Port 1). Damit sind 90 dB Dynamikumfang[46] abbildbar, mehr als ausreichend für Sprachsignale. Der Port 2 enthält alle 30 Ausgänge der Filterbank, d. h. jeweils den linken und rechten Stereokanal sowie den Real- und Imaginärteil. Die Schnittstelle Port 3 gibt mit vierfacher Unterabtastung die Ein- und Ausgänge der Nachregelschleifen aus und leitet die im Chip 1 berechneten Amplitudenquotienten und Phasendifferenzen durch. Das Datenaufkommen an allen drei Ports zeigt die Tab. 4.5. Ein serielles Protokoll bedeutet gegeben durch den Bittakt also einen hohen Systemtakt von mindestens etwa 32 MHz, während parallele Schnittstellen breite Datenpfade erzeugen. Andererseits hängt die Festlegung des Systemtaktes von der Architektur des arithmetischen Kerns ab und wird um so höher, je sequentieller dessen Struktur ist. Da aber das Zielsystem die in Abb. 4.5 dargestellte Struktur hat, zunächst keine

[46] Mit jeder Binärstelle wird der darstellbare Signalbetrag verdoppelt, gemessen in Dezibel ergibt sich deswegen $20 \log(2) \cong 6{,}02$ dB ≈ 6 dB. Da der AD/DA-Wandler das Zweierkomplementformat für die Darstellung der Abtastwerte nutzt und somit ein Bit die Signalpolarität kodiert, verbleiben 15 Bit zur Darstellung der Amplitude, was 90 dB entspricht (siehe auch Kap. 4.2.3).

Gehäuseminimierung im Vordergrund steht und die Schnittstelle (Hostinterface) zum PC-System ohnehin parallel ausgeführt ist, werden für alle Ports wortbreite, parallele Schnittstellen spezifiziert[47].

Interface	**Port 1**	**Port 2**	**Port 3**
parallel	nicht möglich, AD/DA-Wandler hat serielles Interface	2 x 30 x 2 x f_A = 1.953.120 Worte/s (stereo, 30 Frequenzkanäle, Real- und Imaginärteil)	(2+2+2) x 30 x f_A/4 = 732.420 Worte/s (Stereo-Eingang Chip 2, Amplitude und Phase, Stereo-Ausgang Chip 2, 30 Frequenzkanäle)
seriell	2 x 16 x f_A = 520.832 Bit/s	2 x 30 x 2 x 16 x f_A = 31.249.920 Bit/s (16 Bit Datenwortbreite)	(2+2+2) x 30 x f_A/4 x 16 = 11.718.720 Bit/s (16 Bit Datenwortbreite)

Tab. 4.5: Maximal geplanter Datendurchsatz an den Ports des Chip-Satzes bei einer Abtastfrequenz von $f_A = 16.276\ Hz$.

Das erfordert weiterhin eine Organisation der von Chip 1 produzierten sequentielle Reihenfolge der Daten. Der Aufbau eines solchen Frames aus sortierten Einzeldaten (Slots) kann an dieser Stelle schon festgelegt werden (Tab. 4.6). In der realisierten Version sollen jedoch die Ein- und Ausgangsdaten von Chip 2, die Betragsquotienten der Amplituden und die Phasendifferenzen nicht alle gleichzeitig an Port 3 verfügbar sein. Der Datenstrom besteht hierbei aus den Eingangsdaten, die Chip 2 durch seine Ausgangsdaten ersetzt. Somit enthält ein Frame nur die 60 Werte, die von der Gammatone-Filterbank aus einem Stereoabtastwert des Eingangs für die Nachregelschleifen generiert werden, d. h. mit aufsteigender Frequenzkanal-Nummer erst für den linken und danach für den rechten Kanal.

links			**rechts**		
Slot 0	. . .	Slot 29	Slot 30	. . .	Slot 59
Abtastwert Kanal 0	. . .	Abtastwert Kanal 29	Abtastwert Kanal 0	. . .	Abtastwert Kanal 29

Tab. 4.6: Unterteilung in einem Frame an Ein- und Ausgang von Chip 2.

Mit der Abtastfrequenz von f_A=16.276 Hz sind auch die Durchsatz-Anforderungen für die Signalverarbeitung gegeben. Der Chip 1 hat $1/f_A$ = 61,44 µs Zeit, um aus einem Stereoabtastwert alle 120 Ausgangswerte zu berechnen. Im Chip 2 erfolgt keine Datenvervielfachung und 60 Werte (links/rechts) müssen in $4/f_A$ = 245,761 µs berechnet werden, d. h. es verbleiben maximal 4,096 µs Zeit zur Bearbeitung eines Abtastwertes.

[47] Der bidirektionale Port1 benötigt dadurch einen seriell/parallel Umsetzer im Hostinterface.

4.2 Entwurfsentscheidungen und Designsystem

Neben der technischen und inhaltlichen Analyse der Aufgabe besteht ein wesentliches Problem darin, anhand der Fähigkeiten der zur Verfügung stehenden Werkzeuge zur Schaltungssynthese die Realisierbarkeit des Designs relativ frühzeitig einschätzen zu müssen. Das heißt, eine Analyse der unterstützten Datenformate und der im CAD-Werkzeug umsetzbaren Operationen und Konstrukte, der erforderlichen Übergänge zwischen verschiedenen Werkzeugen und den Voraussetzungen der gewählten Zieltechnologie haben wesentliche Rückwirkungen auf eine gegebenfalls notwendige Umformulierung bzw. Adaptierung der Entwurfsaufgabe. Im folgenden sollen deshalb das praktische Vorgehen beim CAD-gestützten Schaltungsdesign für den Chip 2 und die Voraussetzungen im Designsystem *Synopsys*[48] hinsichtlich der Datenformate, Operationen und der generellen Alternativen und Prinzipien dargestellt werden.

4.2.1 Designsystem und Zieltechnologie

Die Realisierung derartiger Signalverarbeitungsschaltungen unterliegt in der Praxis einer ganzen Reihe von Nebenbedingungen, die das Optimum aus Funktionalität, Architektur, Zieltechnologie, Entwurfswerkzeug und möglichem Zeitaufwand entsprechend den eingegangenen Kompromissen sehr unterschiedlich beeinflussen können.

Das Gajski-Walker-Diagramm [Walker und Thomas 1985, Lehmann et al. 1994] stellt die prinzipiell möglichen Entwurfssichten und Zwischenebenen beim Schaltkreisentwurf zusammen (Abb. 4.6).

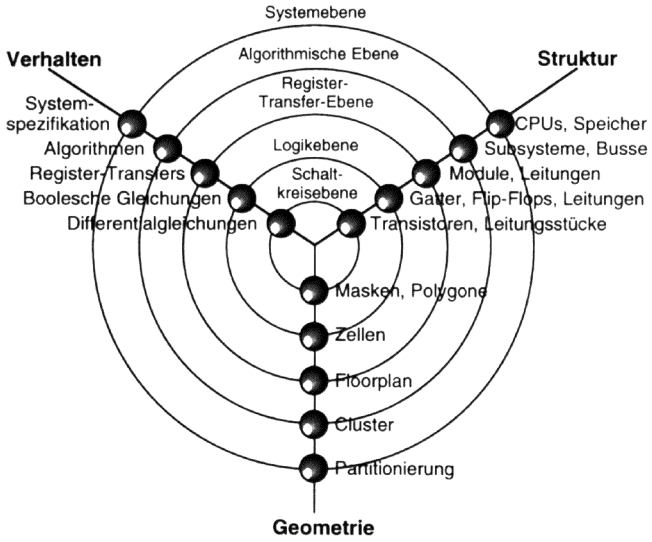


Abb. 4.6: Y-Diagramm nach Gajski-Walker [Lehmann et al. 1994, Walker und Thomas 1985].

[48] *Synopsys, Inc.*- Hersteller eines CAD-Systems für Schaltungssynthese und -simulation.

Derzeit verfügbare CAD-Systeme decken die meisten dargestellten Bereiche und Zweige ab, wobei allerdings in allen Fällen werkzeugspezifische Aspekte zu berücksichtigen sind. So ist bereits die Kodierung in der standardisierten Hardware-Beschreibungssprache VHDL vom jeweiligen Entwurfswerkzeug nicht unabhängig, d. h. alle verfügbaren CAD-Systeme unterstützen einerseits nur Subsets von VHDL, ergänzen andererseits den Entwurf durch eigene optimierte (und proprietäre) Bibliotheken (*packages*) mit Datentypen und synthetischen Operatoren oder Funktionen. Prinzipiell eignet sich VHDL zur Beschreibung verschiedener Ebenen im Designprozeß und ist i. d. R. mit einer Verhaltens- oder Strukturbeschreibung der Ausgangspunkt des CAD-gestützten Entwurfs. Auch für die Implementation des Perzeptionsmodells wurde mit der textuellen Beschreibung des Verhaltens des Algorithmus' begonnen und die gewünschte Schaltung in VHDL spezifiziert. Das Designsystem *Synopsys* ist in der Lage, den abstrakten VHDL- oder Verilog-Kode[49] in eine optimierte Netzlistenbeschreibung einer digitalen Schaltung umzuwandeln, d. h. zu synthetisieren. VHDL, vom Ursprung her eine Programmiersprache zur Verhaltensbeschreibung und Simulation von Hardware, steht nicht mit der vollen Spezifikation für die Synthese zur Verfügung. Der eigentlich kreative Prozeß besteht somit in der Formulierung einer *synthetisierbaren* VHDL-Beschreibung. An dieser Stelle soll darauf verzichtet werden, VHDL und das mit *Synopsys* umsetzbare Subset im Detail zu erläutern; vielmehr sollen die für die Umsetzung des Perzeptionsmodells bedeutsamen programmtechnischen und sprachlichen Mittel bzw. die Designalternativen für arithmetische Einheiten und der entwickelte Entwurfsfluß vorgestellt werden. Die Schaltungssynthese für den Chip 2 wurde sowohl für ASIC-[50] bzw. Standardzell-Technologien, als auch für programmierbare Logikbausteine (FPGA) durchgeführt. Hierbei wurde mit einem strukturellen Schaltungsdesign begonnen. Da jedoch die Entwurfsaufgabe in der Implementierung eines Algorithmus' bestand, wurde bevorzugt die Möglichkeit der direkten Datenpfadsynthese aus einer VHDL-Beschreibung des Perzeptionsmodells untersucht und realisiert. Für Standardzell-Technologien wurde der *Synopsys Design Compiler* verwendet, für den Entwurf von FPGA-Designs der darauf angepaßte *Synopsys FPGA Compiler* und der *Synopsys Behavioral Compiler* zur direkten Verhaltenssynthese (alle in der Version 99.10, Patch-level 2).

4.2.2 VHDL-Hardware-Beschreibung und FPGA-Zielplattform

Eine synthetisierbare Verhaltensbeschreibung in VHDL zu erzeugen, bedeutet ganz allgemein die sprachlich strukturierte Modellierung von Werteberechnungen, -zuweisungen und -umwandlungen auf den Datentypen, die vom Synthesewerkzeug korrekt umgesetzt werden können. Das mögliche Ergebnis der Synthese, d. h. der Zuordnung bestimmter Programmkonstrukte auf die Hardware einer Zieltechnologie sowie deren Optimierung durch das Werkzeug, ist z. T. werkzeug- und technologiespezifisch und erfordert entsprechendes Vorwissen. Nur vom Umfang her überschaubare Schaltungen können anhand des generierten Schaltplans manuell überprüft werden; im allgemeinen prüft man die korrekte Beschreibung und Synthese stufenweise und iterativ mit intensiven Simulationen.

[49] *Verilog (XL)* ist neben VHDL eine zweite, sehr verbreitete Hardware-Beschreibungssprache.

[50] ASIC - *Application Specific Integrated Circuit.*

Ein Hauptprinzip von VHDL ist die Möglichkeit, Operationen zeitlich steuern zu können:

```
if clock'event AND clock='1' then
      out <= in;
end if;
```

Hier wird bei einer steigenden Flanke des Taktsignals *clock* das Signal *in* auf das Signal *out* zugewiesen. Der primären Ausrichtung auf die Beschreibung von Hardware-Einheiten wird durch die starke Modularität und Strukturierung des VHDL-Kodes Rechnung getragen. Eine *entity* ist ein Strukturobjekt, das unterschiedliche Architekturen (*architecture*) haben kann, in denen das Verhalten steckt. Hier werden gewöhnlich weitere *entity*-Strukturobjekte als Komponenten *(component*) eingebunden und über deren Signalports miteinander im VHDL-Kode vernetzt. Die gegebenenfalls vorhandenen alternativen Architekturen von Komponenten werden mit einem *configuration*-Konstrukt ausgewählt. Die internen und weitere externe Signale sind nun Gegenstand der eigentlichen Verhaltensbeschreibung in einer *architecture*. Um bestimmte Abläufe (Zuweisungen etc.) zu kapseln und von externen Ereignissen (Signalaktivitäten) abhängig zu machen, definiert man hier Prozesse (*process*). Die Architekturen haben gewöhnlich mehrere miteinander vernetzte Prozesse. In Prozessen können nun weitere lokale Signale und Variablen und daraus zusammengesetzte Typen definiert und verwendet werden. Wichtig in diesem Zusammenhang ist der Unterschied von Signal- und Variablenzuweisungen: Signale werden immer an der nächsten im- oder explizit vorhandenen Taktflanke zugewiesen, egal, an welcher Stelle sie im VHDL-Kode einer zeitlichen Kapsel (Prozeß) stehen. Variablen dagegen werden sofort zugewiesen. Das hat großen Einfluß auf die Umsetzung bestimmter Datenabhängigkeiten. VHDL erlaubt außerdem die Verwendung eigener Funktionen (*function)* und Prozeduren (*procedure*). Die Gesamtheit eigener VHDL-Einheiten (*entity*, *function* etc.) kann in einer Bibliothek (*package)* zusammengefaßt werden und läßt sich beispielsweise als Paket in andere Entwürfe einbinden. Mit Konstrukten wie *case* und *if-then-else* Anweisungen, wird Select- und Multiplex-Logik beschrieben. Konstrukte wie *while*, *for*- und *do*- Loops erlauben die Implementierung von Iterationen und Schleifen, was vor allem im Kontext der direkten Datenpfadsynthese mit dem *Synopsys Behavioral Compiler* Bedeutung hat (siehe Kap. 5.1). Für die Verifikation von VHDL-Beschreibungen werden VHDL-*Testbenches* erstellt, die dem zu testenden Design eine Simulationsumgebung für die Stimuligenerierung und die notwendigen Datei-Schnittstellen verschaffen. In einer *testbench* können z. B. die verschiedenen Konfigurationen einer *entity* simuliert werden.

Eine weitere wichtige Eigenschaft beim praktischen Schaltungsentwurf ist die Verflechtung des eigentlich unabhängigen VHDL-Kodes mit den darin gegebenenfalls einzubettenden Anweisungen (*pragma*) für das Synthesewerkzeug.[51] Die gesteuerte Interpretation des eigenen VHDL-Kodes, bzw. die kontrollierte Steuerung der Synthese, ist für deren Erfolg unverzichtbar. Außerdem verteilt sich die Steuerung auf den VHDL-Kode und die in der Regel umfangreichen metasprachlichen Direktanweisungen für das Synthesewerkzeug in Form eines Syntheseskripts. Diese Einflußnahme erfolgt gewöhnlich auf sehr hohem Abstraktionsniveau, wobei

[51] z. B. -- *pragma set_dont_touch* vergibt z. B. das „*dont_touch*“ Attribut an die nachfolgend deklarierten Designbestandteile.

der Schwerpunkt hierbei weniger auf der Umsetzung einzelner VHDL-Konstrukte liegt, sondern mehr auf dem optimierten Einsatz gewünschter synthetischer Blöcke und der kompatiblen Benennung von Designbestandteilen. Mit Hilfe der Syntheseskripts wird der ursprüngliche VHDL-Kode übersetzt. Damit wird erreicht, daß in bestimmten Situationen überhaupt ein Ergebnis entsteht bzw. suboptimale Syntheseergebnisse und Inkompatibilitäten mit Designbibliotheken oder nachfolgenden Werkzeugen (z. B. für das Plazieren und Verdrahten) vermieden werden können. Es existiert eine Vielzahl von Skript-Kommandos einschließlich diverser Kommando-Optionen, die alle Schritte zwischen Analyse, VHDL-Interpretation, Synthese, Optimierung (Komponentenauswahl, Umwandlung logischer Ausdrücke etc.) und Dateitransfer in und aus dem Entwurfswerkzeug steuern. Die für Chip 2 durchgeführte Steuerung der Synthese mit dem *Synopsys Behavioral Compiler* benötigte das in Anhang A befindliche Skript und enthält praktische Beispiele für o. g. Anweisungen. Deutlich muß darauf hingewiesen werden, daß es mit einem simulier- und synthetisierbaren VHDL-Kode für ein beliebiges Entwurfsproblem nicht getan ist, sondern eine enge Verknüpfung mit dem gewählten Entwurfsfluß und dem CAD-Werkzeug besteht. Das äußert sich in der Praxis, wie in der hier beschriebenen Arbeit, dadurch, daß schätzungsweise 60 - 80 % der Projekt-Zeit vom iterativen Optimieren der Syntheseschritte, also von der Werkzeugsteuerung, in Anspruch genommen werden und der eigentliche VHDL-Entwurf fast zur Nebensache wird (siehe auch Kap. 5.1).

Die zur Verfügung stehenden CAD-Werkzeuge unterstützen sowohl den Standardzell-Entwurf für ASIC, als auch die Erstellung von FPGA-Designs. Für den Chip 2 wurden sowohl die Standardzell-Bibliothek ES2-07 (0,7 µm CMOS) als auch die FPGA-Technologien der *Altera Flex10K*-Familie[52] und der *Xilinx Virtex* Bausteinfamilie verwendet. Den schematischen Aufbau eines *Xilinx Virtex*-FPGA zeigt Abb. 4.7.

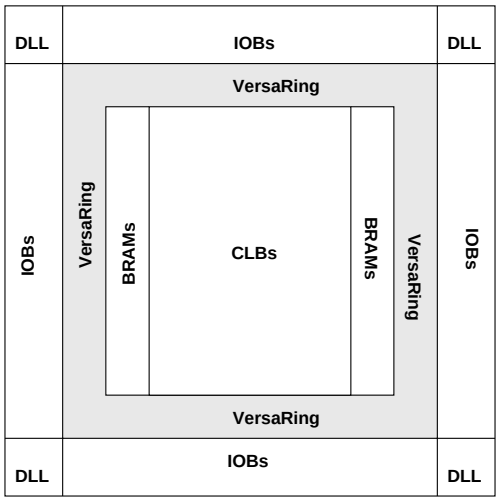


Abb. 4.7: Die Aufteilung der Chip-Fläche eines *Xilinx*-Virtex-FPGA. Die konfigurierbaren Logikblocks (*CLBs*) sind die Hauptbestandteile des programmierbaren Bausteins. Sie werden ergänzt durch statischen Block-RAM (*BRAM*), einen Verdrahtungskanal (*VersaRing*), konfigurierbare I/O-Zellen (*IOB*) und vier unabhängige Taktgeneratoren, bzw. -umformer (*delay locked loops - DLL*) [Xilinx Inc. 1999].

[52] *Altera, Inc.* - FPGA-Hersteller.

Die ursprünglich direkt angestrebte ASIC-Technologie für das Perzeptionsmodell besitzt a priori das größte Potential hinsichtlich hohem Datendurchsatz (geringe Gatter-Laufzeiten, hohe Taktrate) und minimalem Flächenbedarf bei den z. Z. verfügbaren Technologien für das Prototypendesign mit 0,35 bzw. 0,5 µm Strukturbreite. Sie verursacht aber die höchsten Kosten, die im Extremfall auch für einen möglicherweise nicht funktionsfähigen Chip getragen werden müssen. Da ein FPGA dagegen beliebig oft reprogrammierbar ist, kann ein ASIC-Design prototypisch als FPGA-Entwurf entstehen und hier beliebig ausgetestet werden, bevor ein Technologiewechsel erfolgt.

Den inneren Aufbau einer FPGA-Zelle zeigt beispielhaft die Abb. 4.8. Ein CLB wird aus zwei Slices gebildet. Ein Slice besteht wiederum aus zwei Logikzellen. In den (hier nicht dargestellten) internen RAM-Zellen der Look-Up-Tables (LUT) einer Logikzelle wird beim Programmieren des FPGA die logische Verknüpfung der bis zu vier Eingänge in Form einer Booleschen Gleichung gespeichert. Dem Schaltnetz folgt eine Carry-Logik (insbesondere für arithmetische Funktionen von Bedeutung), die dem Aufbau schneller Carry-Querverbindungen zwischen den Slices dient. Die ausgangsseitigen Register komplettieren das Schaltnetz der Look-Up-Tables zu einem getakteten und somit synchronisierbaren Schaltwerk.

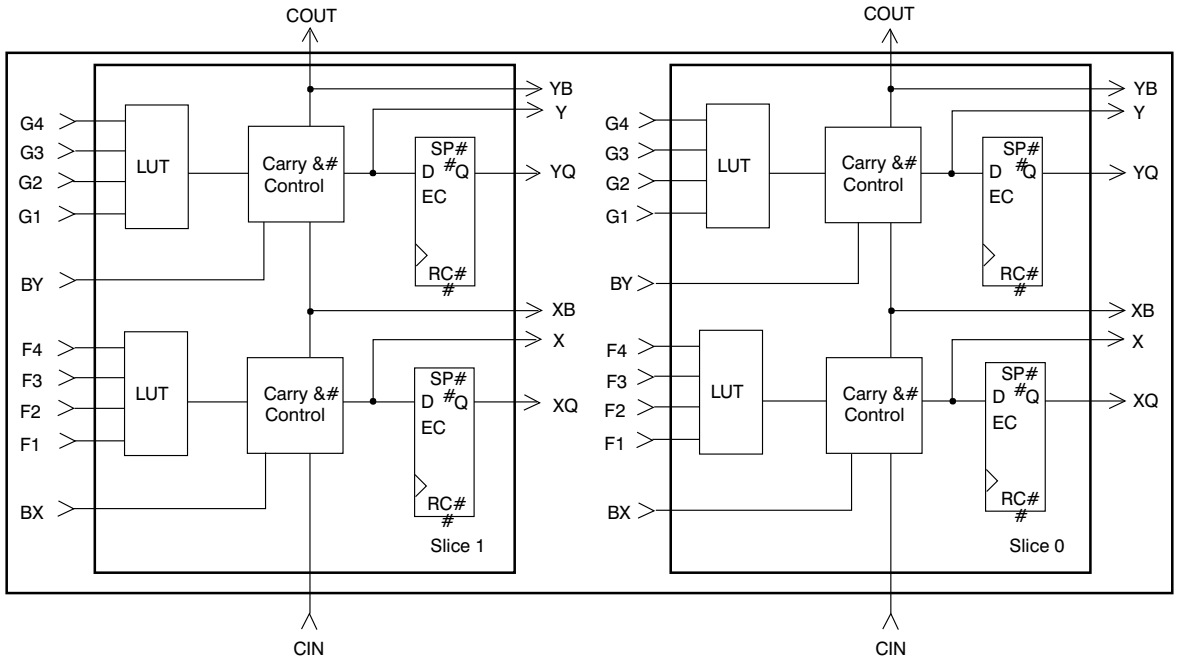


Abb. 4.8: Aufbau eines CLB eines *Xilinx Virtex*-Bausteins bestehend aus zwei identischen Slices. Die Logikzellen eines Slices besitzen Look-Up-Tables (LUT), welche die logische Verknüpfung der Eingänge vornehmen, zwei Carry-Blöcke zum Aufbau schneller Carry-Ketten und zwei D-Flip-Flops zur Synchronisation der Ausgangssignale [Xilinx Inc. 1999].

Komplexe Designs nutzen nun zumindest Teile der CLBs und entstehen ansonsten durch eine Vernetzung mit anderen CLBs und Slices, RAM-Blöcken und Pins. Neben den Logikzellen (deren Look-Up-Tables auch als verteilte Speicher nutzbar sind) können die RAM-Blöcke di-

rekt als statischer Onchip-RAM oder -FIFO[53] mit einfacher Ansteuerung genutzt werden. Die Pad-Zellen der Pins können sowohl Ein- als auch Ausgang sein und unterstützen mehrere Logik-Normen (z. B. TTL[54], CMOS, LVDS[55]).

Zwischen den Slices und zu den Chip-Pins verlaufen globale und lokale Verdrahtungskanäle, die eine kritische Ressource darstellen. Die Signallaufzeiten auf den nicht immer direkten Wegen werden oft zum begrenzenden Faktor für die maximal erreichbare Taktrate. Hierin liegt auch die Hauptursache dafür, daß ein FPGA-Design die Performanz eines optimierten Standardzell-ASIC-Designs nicht erreichen kann. Die Kreuzungspunkte im Gitter der Verdrahtungskanäle werden von lokalen programmierbaren Verdrahtungsmatrizen verschaltet. Jedes CLB besitzt einen exklusiven Zugang zu einer solchen Matrix (siehe Abb. 4.9).

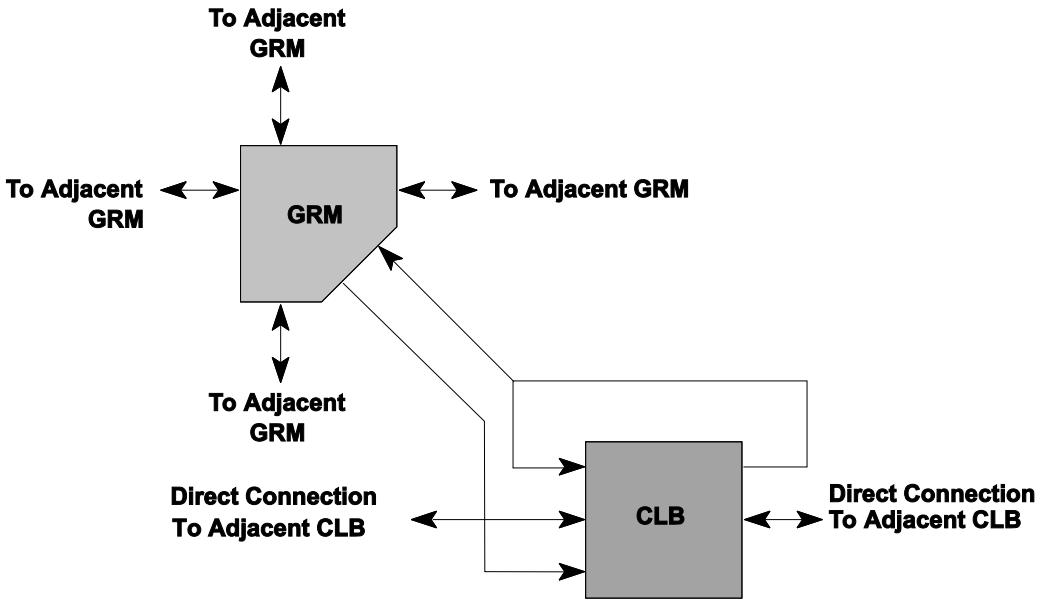


Abb. 4.9: Anbindung eines CLBs an die Umgebung. Mit einer dedizierten Schaltmatrix, der *Global Routing Matrix* (GRM), wird die Verbindung in das globale Verdrahtungsnetzwerk hergestellt. Daneben existieren zusätzliche Querverbindungen zu benachbarten CLBs [Xilinx Inc. 1999].

Da der Designspeicher im FPGA, der die Belegung der Logikzellen und der Verdrahtungsmatrizen enthält, flüchtig ist, muß nach jedem Abschalten der Betriebsspannung das Design neu geladen werden.

Aufgrund der Reprogrammierbarkeit und der mittlerweile angekündigten hochdichten FPGA-Technologien[56] ist ein „schnelles“ Prototyping auch für größere ASIC-Designs die kostengün-

[53] FIFO - *First In First Out*; spezieller i. a. wortbreiter Speicher einer bestimmten Tiefe, wobei die zuerst eingelesenen Daten auch zuerst am Ausgang verfügbar werden (Warteschlangen-Prinzip).

[54] TTL - *Transistor-Transistor Logic*.

[55] LVDS - *Low-Voltage Differential Signaling.*

stigere, flexiblere und erfolgversprechendere Methode. Für die komplexen CAD-Systeme ist ein Technologiewechsel prinzipiell kein Problem, so daß im Idealfall ausgehend von einem FPGA-Prototypen in einem zweiten Schritt ein ASIC-Design aus der gleichen VHDL-Beschreibung erzeugt werden kann. Infolge der Vielzahl der Nebenbedingungen aber, etwa beim Bereitstellen und Anpassen der Entwurfs- und Simulationsbibliotheken, ist damit trotzdem ein u. U. hoher manueller Arbeitsaufwand verbunden.

Die Parameter der im einzelnen untersuchten FPGA-Bausteine von *Altera* und *Xilinx* sind in der Tab. 4.7 gegenübergestellt.

Xilinx Virtex XCV 800-4	***Altera Flex 10k100A-1***
56x84 CLB-Array (Reihen x Spalten)	624 Logic Array Blocks
888.439 Systemgatter	158.000 Systemgatter
21.168 Logikzellen	4.992 Logikzellen
114.688 Block-RAM-Bits in 8 Blöcken	24.576 RAM-Bits in 6 Blöcken
301.056 Distributed-RAM-Bits	-
512 Nutzbare I/O-Pins	406 Nutzbare I/O-Pins
4 Taktgeneratoren (*Delay Locked Loops* - DLLs)	2 Taktgeneratoren (PLL)

Tab. 4.7: Technische Parameter, Logik-, Speicher- und Taktressourcen eines *Altera Flex* 10K100 FPGA und eines *Xilinx Virtex* XCV-800 FPGA.

Dabei kamen als Backend-Werkzeuge zur Plazierung und Verdrahtung der Designs *MaxPlusII* (Version 9.2) für *Altera*-FPGA und der *Xilinx Designmanager* (Version 2.1i, Patch-Level 4) für *Xilinx*-FPGA zur Anwendung.

4.2.3 Zahlendarstellung

Der erste und augenscheinlichste Konflikt zwischen dem realen Algorithmus und der gewünschten Hardware-Realisierung des Perzeptionsmodells wird bei der Betrachtung der verwendbaren Datenformate deutlich. Der ursprünglichen Fließkomma-Verarbeitung im Perzeptionsmodell (IEEE-32 Bit-*single precision*) stehen im Synthesewerkzeug nur Festkomma-Blöcke direkt synthetisierbar gegenüber. Deswegen wurde untersucht, welche Eigenschaften die beiden Formate besitzen und welche Konsequenzen daraus für den Algorithmus und die Handhabung des CAD-Systems erwachsen.

[56] für *Xilinx-Virtex-II* Bausteine wird bereits die 10 Millionen-Gatter-Marke angekündigt

Festkommazahlen

Für Festkommazahlen werden Stellenwertsysteme verwendet, wobei die mit *n* Vorkommastellen und *m* Nachkommastellen quantisierte Zahl x_{fix} als endliche Summe dargestellt wird:

$$x_{fix} = \sum_{i=-m}^{n-1} a_i b^i \text{ , wobei } 0 \le a_i \le b-1 \qquad \text{Gl. (4.12)}$$

Im binären Zahlensystem ist die Basis b=2 und [0, 1] die Menge der Werte für a. Die Stellen einer *(n,m)*-Festkommazahl x_{fix} lauten demnach:

$$x_{fix} = (2^{n-1} \ldots 2^0, 2^{-1} \ldots 2^{-m}) \qquad \text{Gl. (4.13)}$$

Das Komma liegt dabei zwischen *n=0* und *m=-1*. Reelle Zahlen können durch eine solche Summe angenähert werden, wobei entsprechend der Stellenzahl und -interpretation unterschiedliche Wertebereiche erreicht werden:

- vorzeichenlose (*n,m*)-Festkommazahl $0 \le x_{fix} \le 2^n - 2^{-m}$
- 1er Komplement $2^{-m} - 2^{n-1} \le x_{fix} \le 2^{n-1} - 2^{-m}$
  (*n,m*)-Festkommazahl mit Vorzeichen im MSB[57] des Vorkommateils, wobei die Null redundant kodiert ist.
- 2er Komplement $-2^{n-1} \le x_{fix} \le 2^{n-1} - 2^{-m}$
  (*n,m*)-Festkommazahl mit nicht redundanter Kodierung der Null, aber unsymmetrischen Randwerten.

In allen Fällen ist die Auflösung des Wertebereichs einer Festkommazahl zwischen Minimal- und Maximalwert äquidistant, d. h. ein Signalwert hat entsprechend der Wortlänge *n+m* mögliche Stufen. Der Dynamikumfang im Sinne abbildbarer Signalpegel (Betrag des Signals) liegt zwischen dem kleinsten

$$x_{min} = 2^{-m} \qquad \text{Gl. (4.14)}$$

und dem größten darstellbaren Betrag

$$x_{max} = 2^{n-1} - 2^{-m} \qquad \text{Gl. (4.15)}$$

[57] MSB - *Most Significant Bit*; Bit mit dem höchsten Stellenwert.

und lautet für die bei der Signalverarbeitung übliche Zweierkomplementdarstellung:

$$D_{Fix} = 20lg\left(\frac{x_{max}}{x_{min}}\right) = 20lg(2^{(n+m-1)} - 1) \ [dB] \qquad \text{Gl. (4.16)}$$

Für die Quantisierung, aber auch die Anpassung höher aufgelöster Ergebnisse arithmetischer Operationen (Multiplikation, Überlauf bei der Addition) an das ursprüngliche Datenformat hat man die Wahl zwischen verschiedenen Rundungsverfahren. *Round to/away from zero* beschreibt permanentes Abrunden (Abschneiden) bzw. Aufrunden und erzeugt einen absoluten Fehler von maximal dem Wert der ersten nicht mehr darstellbaren Stelle, bestimmt durch die Quantisierung $Q_{FIX} = 2^{-(n+m-1)}$. *Round to nearest* rundet auf den nächsten darstellbaren Wert, womit sich der maximale Fehler auf $Q_{FIX}/2$ gegenüber dem Abschneiden halbiert (Abb. 4.10):

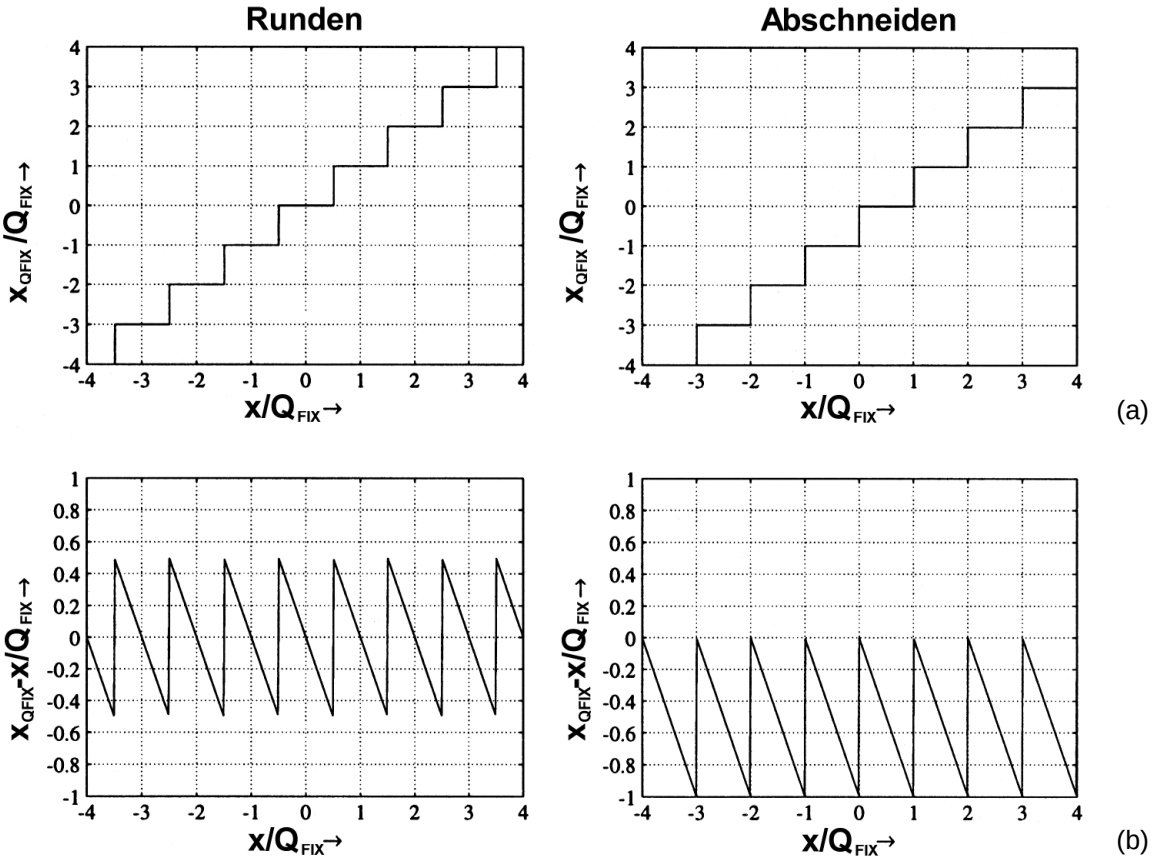


Abb. 4.10: Quantisierungskennlinien für die auf eine Quantisierungsstufe Q_{FIX} normierten Werte x mit Festkommazahlen x_{QFIX} (a) und der entstehende absolute Fehler (x_{QFIX}-x) (b) durch Abschneiden (*round to zero*) bzw. Runden (*round to nearest*) nach [Zölzer 1996].

Solange höher quantisierte Zahlen im Wertebereich eines geringer auflösenden Festkomma-Formates darstellbar sind, ist der maximale Quantisierungsfehler konstant.

Fließkommazahlen

Fließkommazahlen entstanden als Folge der bei Festkommazahlen immer wiederkehrenden Skalierungsprobleme, um Zahlenrepräsentationen im darstellbaren Bereich zu halten. Allgemein gilt für eine Fließkommazahl x_{float} mit dem Vorzeichenbit *s*, der Mantisse *M* (*m* Stellen) und dem Exponenten *e* (*n* Stellen) bei einer Basis *b*:

$$x_{float} = (-1)^s \left(\sum_{i=1}^{m} M_i b^{-i} \right) \cdot b^e \text{, wobei } 0 \le |M| \le b^0 = 1 \qquad \text{Gl. (4.17)}$$

Die heute gebräuchlichste Notation wird in der IEEE-754 Spezifikation [IEEE 1985] festgelegt und beschreibt neben den Datenformaten die Rechenoperationen, standardisierte Rundungsregeln und die Behandlung von Sonderfällen. Das IEEE-Fließkomma-Format (*single precision)* z. B. hat dementsprechend bei 32 Bit Wortlänge folgende Struktur (Abb. 4.11):

Abb. 4.11: 32 Bit-IEEE-Fließkomma-Format (*single precision*).

Mit *s*=0 bezeichnet das Vorzeichenbit eine positive, mit *s*=1 eine negative Zahl. Die Mantisse wird als Signifikant bezeichnet und umfaßt dabei nicht nur die 23 Bit breite Fraktion (Nachkommastellen), sondern eine nicht mitgespeicherte Vorkommastelle („*hidden bit*“). Der ebenfalls nicht dargestellte Dezimalpunkt befindet sich vor dem MSB der Fraktion. Der Signifikant besteht somit aus *m*=24 Bit. Um nicht mit negativen Exponenten rechnen zu müssen, wird der Exponent *Exzeß*-kodiert und Charakteristik *E* (Stellenanzahl *n*=8) genannt. Die Exzeß-Kodierung erfolgt mit einer Verschiebung (*bias*) von 2^{n-1}-1=127. Der Wertebereich der Charakteristik liegt unsymmetrisch im Intervall [-126,127]. Damit können deren Randwerte Spezialfälle, wie *NaN*[58] ($E = 255, f \neq 0$), *Infinity* ($E = 255, f = 0$), *Zero* ($E = f = 0$) und nicht normalisierte Zahlen ($E = 0, f \neq 0$, „*hidden bit*“ nicht gesetzt) kodieren. Mit einer Normalisierung („*hidden bit*“ gesetzt*)* wird eine Vergleichbarkeit und Eindeutigkeit der Zahlendarstellung erreicht.

Die interessanteste Eigenschaft im Hinblick auf die Darstellung von Abtastwerten (von Sprachsignalen) ist die „halblogarithmische“ Auflösung des Wertebereichs (siehe Abb. 4.12). Sie entsteht mit der für jeden Exponenten bereichsweise äquidistanten Auflösung durch die (Festkomma-)Mantisse. Mit einem ausreichend groß dimensionierten Exponenten kann auch die „Lük-

[58] NaN - *Not a Number.*

ke“ zwischen der Null und den vom Betrag her kleinsten darstellbaren Werten minimiert werden (kleinster positiver Wert für das IEEE-Format *single precision* = $2^{-126} \approx 1{,}2*10^{-38}$).

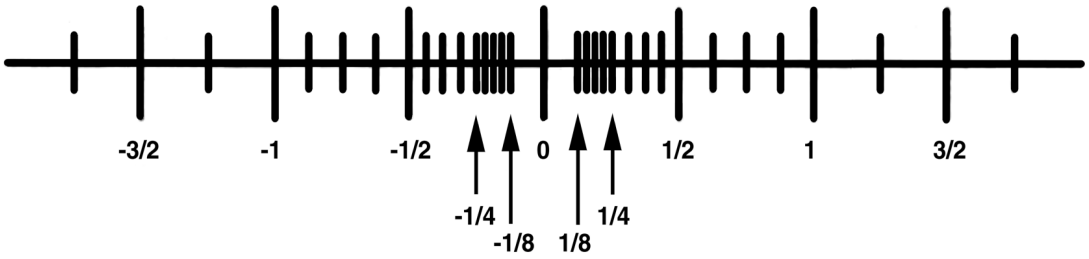


Abb. 4.12: Auflösung einer 6 Bit-Fließkommazahl: 1 Bit Vorzeichen, 3 Bit Signifikant 2 Bit Exponent ([-2, 1], Exzeß = 2), Basis 2, nach [Murdocca 1997].

Allgemein existiert für eine normalisierte Fließkommazahl nach Gl. (4.17) bei einer rein fraktionalen *m*-stelligen Mantisse (kein „*hidden bit*“) mit dem Wertebereich [0,5...1,0) und einem *n*-stelligen Exponenten mit Exzeß-Kodierung (bei einer Verschiebung von 2^{n-1}-1) zwischen dem größten darstellbaren positiven Wert x_{max}:

$$x_{max} = (1 - 2^{-m})2^{(2^{n-1}-1)} \qquad \text{Gl. (4.18)}$$

und dem kleinsten darstellbaren positiven Wert x_{min}:

$$x_{min} = \frac{1}{2}2^{(-2^{n-1}+2)} \qquad \text{Gl. (4.19)}$$

ein Dynamikbereich von:

$$D_{Float} = 20lg\left(\frac{x_{max}}{x_{min}}\right) = 20lg(1 - 2^{-m})2^{2^n-2} \quad [dB] \qquad \text{Gl. (4.20)}$$

Damit überstreichen Fließkommazahlen einen wesentlich größeren Wertebereich als Festkommazahlen in der zur Verfügung stehenden Wortbreite. Die Quantisierungsstufe Q_{FLOAT} ist nicht mehr wie bei Festkommazahlen konstant, sondern wächst in einem durch einen höheren Exponenten bestimmten Quantisierungsintervall (womit sich auch der absolute Fehler erhöht). Für die zu quantisierende Zahl *x* im Bereich $2^{eq} \le x \le 2^{eq+1}$ gilt bei diesem Exponenten *eq* eine Quantisierung von $Q_{FLOAT} = 2^{-m}*2^{eq}$, d. h. die Quantisierungsstufe ist nur abschnittsweise konstant. Das kann auch anhand der Quantisierungs- und Fehlerkennlinien in Abb. 4.13 nachvollzogen werden. Das höchstwertige Bit der hier verwendeten Mantisse kodiert allerdings das

Vorzeichen, womit die Bereiche für einen bestimmten Exponenten nur zweistufig äquidistant quantisiert sind.

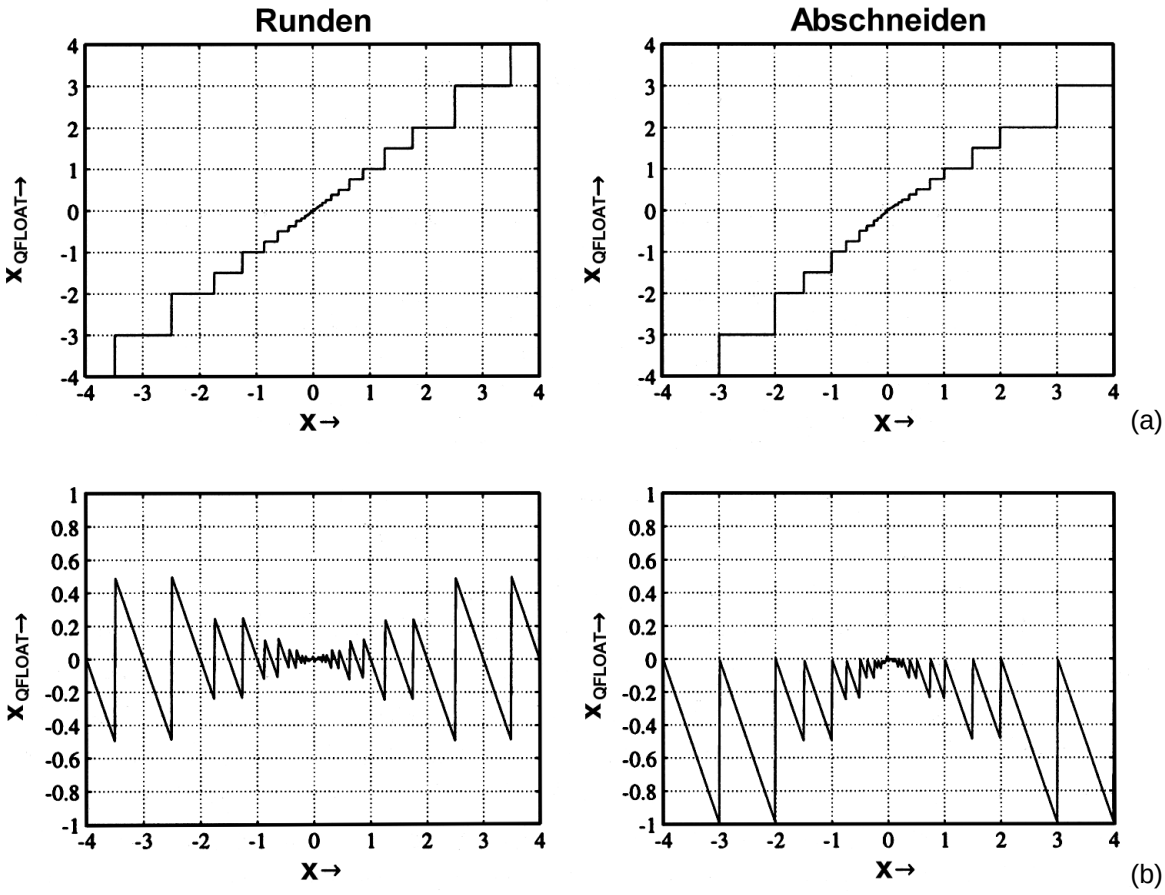


Abb. 4.13: Kennlinie für die Quantisierung einer höher aufgelösten Zahl *x* auf eine Fließkommazahl x_{QFLOAT} (a) und der absolute Fehler (b) durch Runden und Abschneiden, Mantissenbreite *m = 3* (inkl. Vorzeichenbit) und Exponentenbreite *n = 3* nach [Zölzer 1996].

Unter Berücksichtigung der Verteilungsdichtefunktion von Sprachsignalen (siehe Kap. 4.1.3) ist damit ein Fließkomma-Format zur Abbildung dieser Signale offensichtlich besser geeignet, als ein in der Wortlänge vergleichbares Festkomma-Format. Durch den höher aufgelösten Bereich nahe Null, in dem Sprachsignale auch die höchste Häufigkeit aufweisen, sind insgesamt geringere Quantisierungsverzerrungen zu erwarten.

4.2.4 Datenformate und Operatoren im Designsystem

Mit *Synopsys* kann auf Variablen symbolisch gerechnet werden, d. h. eine Addition kann z. B. durch eine strukturelle Addierer-Instanz (als *component*) instanziert, mit einem Funktionsaufruf (*DW_add*) eingebunden oder mit einem intuitiven „+“ „inferiert“ werden. Die Berechnung *a := b + c;* ist somit direkt synthetisierbar. Das Designsystem stellt alle Grundrechenoperationen mit den Festkomma-Datentypen *signed* (Zweierkomplement mit Vorzeichen) und

unsigned (vorzeichenlose Dualzahl) und generischen Verarbeitungsbreiten bereit. Eigene (zusammengesetzte) Datentypen können definiert, die arithmetischen Einheiten jedoch nur auf *signed*, *unsigned* oder Subtypen rechnend synthetisiert werden. Die Hardware-Umsetzung von Operationen auf dem Fließkomma-Format ist ungleich aufwendiger und deswegen auch nicht als synthetisierbare Operation im Designsystem enthalten. Der Fließkomma-Typ *real* kann nur für Simulationszwecke verwendet werden.[59] Die Tabelle 4.8 zeigt einige Beispiele für synthetisierbare arithmetische Funktionen der *Synopsys DesignWare*-Bibliothek.

DW01_absval	Absolute Value
DW01_add	Adder
DW02_sum	Vector Adder
DW01_sub	Subtractor
DW02_mult	Multiplier
DW02_mac	Multiplier-Accumulator
DW02_mult_2_stage	Two-Stage Pipelined Multiplier
DW02_divide	Combinatorial Divider
DW_div_rem	Combinatorial Divider with Quotient and Remainder
DW01_inc	Incrementer
DW01_dec	Decrementer
DW_square	Integer Squarer
DW02_sqrt	Combinatorial Square Root
DW02_cos	Combinatorial Cosine
DW02_sin	Combinatorial Sine

Tab. 4.8: Auszug aus den in der *Synopsys DesignWare* verfügbaren arithmetischen Funktionen [Synopsys Inc. 1999].

Architekturvarianten

Neben den arithmetischen Einheiten enthält die *DesignWare* Entwurfsbibliothek weitere synthetische Blöcke (z. B. Zähler, Spezialregister, synthetische Speicher und FIFOs, etc.). Dabei gibt es zu den meisten Operatoren unterschiedliche Implementierungen (flächenoptimal oder zeitoptimal), die je nach Synthese-Anforderungen ausgewählt werden können (siehe Entwurfsfluß-Beschreibung im Kap. 5.1) bzw. vom Synthesewerkzeug automatisch als optimal erkannt

[59] Ab der Version (*Synopsys* 2000.05) sind generische Bibliothekselemente von Fließkomma-Einheiten mit einem neuen Werkzeug (*Modul Compiler*) erstellbar. Dieser Entwurfsfluß ist jedoch mit FPGA-Technologien noch nicht kompatibel.

und eingesetzt werden. Wird beispielsweise eine Addition verlangt, hat *Synopsys* je nach Nebenbedingungen die Auswahl aus den sechs Addierer-Varianten, dargestellt in Abb. 4.14:

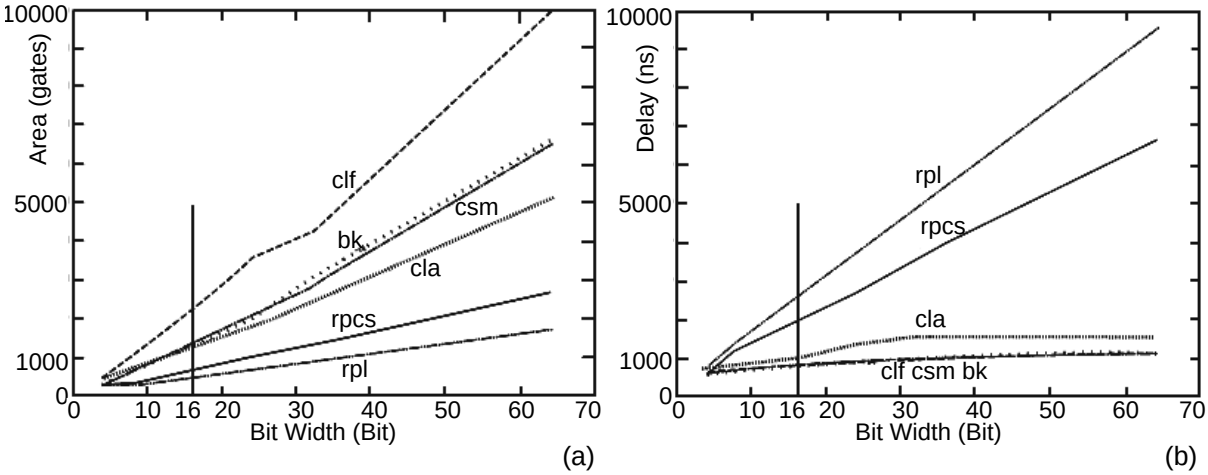


Abb. 4.14: *Synopsys DesignWare*-Addiererstrukturen im Vergleich. Der Zuwachs an Fläche (a) und Verzögerungszeit (b) in Abhängigkeit von der Operandenbreite bei
ripple carry (rpl),
ripple carry select (rpcs),
Brent Kung (bk),
carry look ahead (cla),
fast carry look ahead (clf) und
conditional sum (csm) Addierern [Tong 1999].

Die Eigenschaften dieser Architekturvarianten dienen nicht nur dem CAD-System, sondern auch im Falle einer notwendig werdenden manuellen Auswahl als Richtlinie. Nicht immer werden vom Synthesewerkzeug die optimalen Architekturen eingesetzt, und es muß mit dem expliziten Setzen von *constraints*[60] im VHDL-Kode bzw. im Syntheseskript eingegriffen werden (siehe auch hierzu Anhang A).

Division

Im folgenden sollen die Konsequenzen bzw. praktischen Alternativen bei der Architekturwahl anhand des kritischen Divisionsoperators dargelegt werden. Der kombinatorische Divisionsoperator *DW02_divide* beruht auf fortgesetzten Subtraktions- und Konkatenierungs-Iterationen, bis ein nicht weiter um den Divisor reduzierbarer Rest vom Dividenden übrigbleibt. Implementiert ist ein Radix-2 Digit-Recurrence Algorithmus nach [Ercegovac und Lang 1994], wobei in jedem Iterationsschritt eine Quotientenstelle konkurrent mit einer Zwischenergebnisstelle, dem aktuellen Rest, berechnet wird. Danach wird zunächst der Divisor vom Stellenwert des MSB des Dividenden subtrahiert.

[60] *Constraints* sind Parameter, Forderungen oder Einschränkungen an den Synthese-Prozeß. Ein globales *constraint* ist z. B. die geforderte Taktrate einer Schaltung. Nach deren Festlegung sucht das Syntheseswerkzeug primär nach ausreichend schnellen Architektur- und Schaltungsvarianten.

Gilt Minuend ≥ Subtrahend, ergibt sich eine „1“ für den Quotienten und der Rest ist gleich dem ersten Zwischenergebnis; andernfalls ist der Quotient „0“ und das Zwischenergebnis entspricht dem Stellenwert des MSB des Dividenden. Nun wird das nächste (MSB-1) Bit des Dividenden an der (LSB[61]-1)-ten Stelle mit dem Zwischenergebnis verkettet und der Divisor davon subtrahiert. Ist die Subtraktion möglich, wird eine „1“ an die (LSB-1)-te Stelle des Quotienten gebracht und das Zwischenergebnis ist der Rest aus der Subtraktion. Ansonsten erhält der Quotient an der (LSB-1)-ten Stelle eine „0“ und das Zwischenergebnis bleibt für die nächste Iteration erhalten. Ist also ein Quotient Q zu berechnen (Gl. (4.21)),

$$Q = \frac{A}{B} \qquad \text{Gl. (4.21)}$$

ist das Zwischenergebnis Z_i nach Gl. (4.22) gleich dem Rest der Subtraktion in der *i-ten* Iteration

$$Z_i = A - BQ_i \qquad \text{Gl. (4.22)}$$

und es gilt Gl. (4.23), wenn s_i die i-te Stelle des aktuellen Quotienten Q_i ist,

$$BQ_i = BQ_{(i-1)} + Bs_i \qquad \text{Gl. (4.23)}$$

Die Iteration zur Berechnung des aktuellen Rests (Zwischenergebnis Z) besteht mithin aus

$$Z_i = Z_{(i-1)} - Bs_i \qquad \text{Gl. (4.24)}$$

und der Quotient Q_i ist nach der i-ten Iteration (Gl. (4.25), *LEN(A)* = Wortbreite von A)

$$Q_i = \sum_{j=1}^{j=i} s_i 2^{(LEN(A)-j)} \qquad \text{Gl. (4.25)}$$

Diese Iteration, eigentlich ein klassisches sequentielles Design mit einem Schaltnetz (Subtraktion, Konkatenierung) und einem Steuerungsautomaten, der die Schleife zur Berechnung der Quotienten- und Zwischenergebnisstellen organisiert, wird im *DW02_divide* „ausgerollt“, d. h. alle Iterationen existieren als reale Hardware. Darüber hinaus besitzt der Dividierer einen Parameter-Pin, über den angegeben werden kann, ob Zweier-Komplementzahlen oder vorzeichenlose Operanden dividiert werden sollen (wird beim „Inferieren“ automatisch belegt) und

[61] LSB - *Lowest Significant Bit*; Bit mit dem geringsten Stellenwert.

einen Ausgangs-Pin, der eine stattgefundene Division durch „0“ anzeigt. Obwohl nach dem Datenbuch die Wortbreiten der Operatoren unterschiedlich sein können, wird bei einer praktisch durchgeführten Synthese vom CAD-Werkzeug eine Angleichung der Wortbreite angemahnt. Die Abbildung 4.15 zeigt eine Flächen- und Verzögerungsabschätzung des Divisionsoperators für die beiden einzigen in *Synopsys* zur Verfügung stehenden Architekturvarianten mit *ripple carry* (rpl) und *borrow-look-ahead* (bla) Subtrahierern.

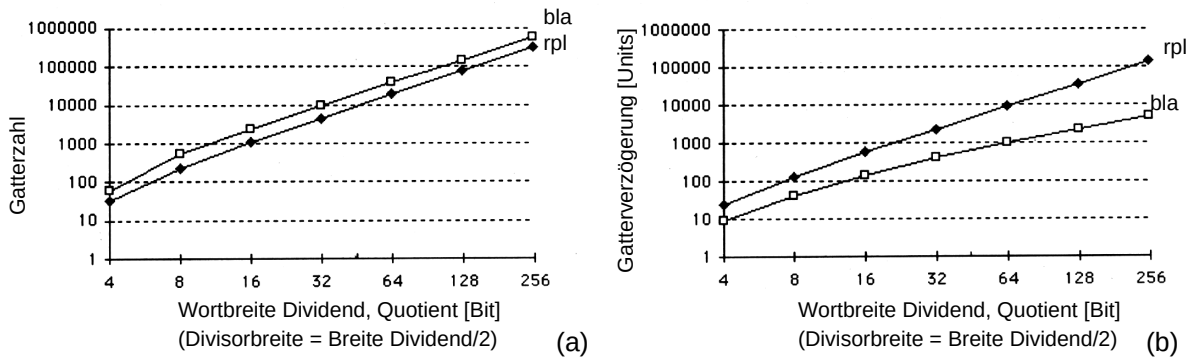


Abb. 4.15: Flächen- (a) und Verzögerungswerte (b) für *Synopsys DesignWare*-Dividierer [Synopsys Inc. 1996]. Wenn die Wortbreiten nur mit der zweiten Potenz anwachsen, steigen Flächen- und Zeitbedarf mit der zehnten Potenz. Architekturvarianten - Division mit *ripple borrow* Subtrahierer (rpl) und *borrow look ahead* (bla) Subtrahierern zeigen dabei die typischen Ergebnisse für flächenoptimale, aber langsame *ripple carry* bzw. schnelle, aber flächenextensive *carry look ahead* Strukturen. Gatterzahl und Schaltnetzverzögerung sind nur als Verhältnisskalen zu verstehen, denn durch die starke Technologieabhängigkeit kann ein Vergleich erst nach der Synthese und Optimierung stattfinden.

Diese Schätzungen des Flächenbedarfs und der Verzögerungszeit beziehen sich auf Standardzell-Designs. Beide Schaltnetz-Varianten entstehen durch Ausrollen der fortgesetzten Subtraktion auf der Basis einer *ripple carry* (rpl) (Abb. 4.16) bzw. *carry look ahead* (Abb. 4.17) Addierer/Subtrahiererstruktur.

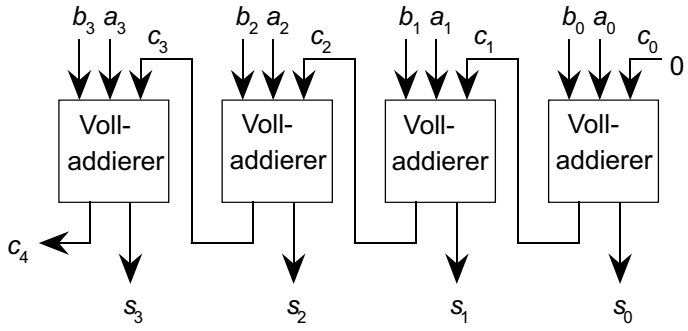


Abb. 4.16: Prinzip eines 4 Bit *ripple carry* Addierers, der die 4 Stellen *i = [0,3]* zweier Binärzahlen a_i, b_i zu einer Summe s_i addiert. Ein möglicher Übertrag (*carry* - c_i) pflanzt sich von Stufe 0 bis in die höchste Additionstufe (für die MSBs der Operanden) fort und produziert eine maximale Verzögerung, den kritischen Pfad durch das Design, nach [Murdocca 1997].

Für die *carry look ahead* Architektur werden die Überträge zwischen einzelnen Stellen im voraus in separater Logik berechnet, womit auf Kosten eines höheren Hardware-Aufwandes der lange Carry-Pfad durch alle Addiererstufen zu vermeiden ist. Im allgemeinen werden damit die Laufzeiten im Design signifikant verkürzt (Abb. 4.17).

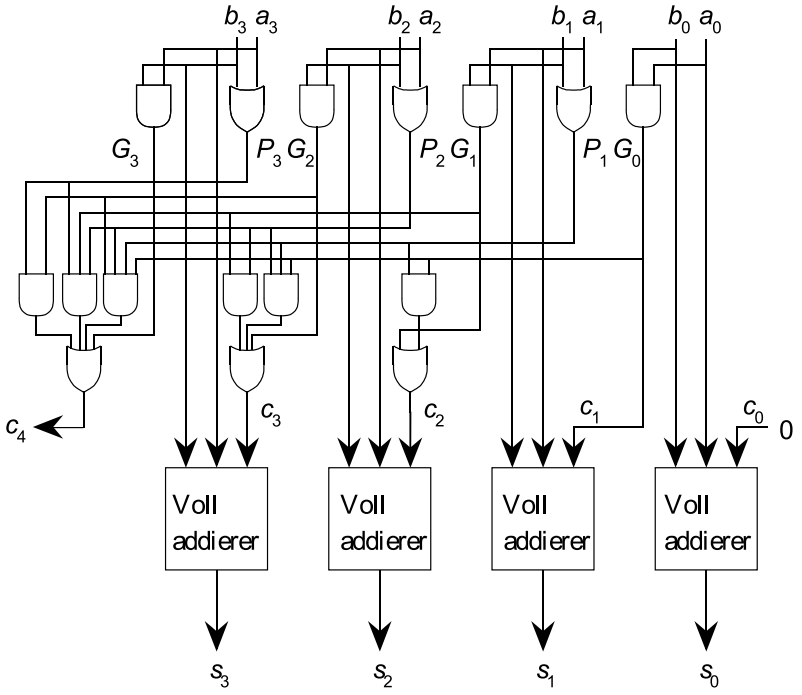


Abb. 4.17: Beim *carry look ahead* Addierer wird der Übertrag nicht sequentiell vom Addierer der LSBs zum MSB Volladierer weitergereicht, sondern aus den Operanden parallel zur eigentlichen Summe berechnet. Bei der Carry-Bit Behandlung wird die Übertrag-Generierung G_i und Propagierung P_i unterschieden, nach [Murdocca 1997].

Bei fast allen heute üblichen FPGA [Andraka 2000] stellt sich diese Betrachtung etwas anders dar. Zunächst sind hier komplexe Zellen die kleinste Einheit. Durch dedizierte schnelle Carry-Logik sind einfache *ripple carry* Strukturen zwischen den FPGA-Zellen sowohl kleiner als auch schneller als die synthetischen allgemeinen Architektur-Vorschläge des *Synopsys Design Analyzers*. Ähnliches gilt für die in der *Xilinx*-Struktur vorteilhaft implementierbaren *binary-tree* Multiplizierer. Diese für *Xilinx*-FPGA optimalen Architekturvarianten werden, da sie als Bibliotheksfunktionen für den *Synopsys Design Analyzer* aufbereitet sind, auch automatisch als effektivste Variante gefunden und eingesetzt.

Im wesentlichen sind mit verschiedenen Schaltnetzvarianten für arithmetische Operatoren die Implementierungsvorräte der *Synopsys DesignWare* erschöpft. Prinzipiell andere Lösungen wie serielle, getaktete Varianten (z. B. serieller Dividierer mit fortgesetzter gesteuerter Subtraktion) werden nicht angeboten. Aus der Sicht der automatischen Synthese ist das jedoch verständlich, denn die Einbettung von Schaltwerken erweitert die Entwurfsaufgabe des Werkzeugs enorm. Eine Ausnahme bilden die Pipeline-Multiplizierer der *Synopsys DesignWare*, die

allerdings auch kein komplexes getaktetes Verhalten haben, sondern mit jedem Takt ein Ergebnis liefern können. Das wird mit der Einführung von Registerbänken erreicht, die das Schaltnetz in Abschnitte mit vergleichbar langen Signallaufzeiten unterteilen.

Festkomma-Operationen

Im Hinblick auf die Wahl der Zahlendarstellung kann in *Synopsys* nur mit dem voll unterstützten Festkomma-Format komfortabel gearbeitet werden. Die Position des Kommas in Operanden und Operationsergebnis und mögliche Überläufe sind auch beim „Inferieren“ von Operationen manuell zu berücksichtigen. Dazu sind beim Aufruf der Operation im VHDL-Kode zusätzlich die Input-Output-Konventionen der *Synopsys DesignWare*-Operatoren einzuhalten (siehe auch [Synopsys Inc. 1999]).

Addition und Subtraktion: Zunächst sind die Operanden bezüglich der Kommaposition auszurichten. Da das Ergebnis einen Übertrag enthalten kann, sind die verarbeiteten Variablen (Operanden und Ergebnis) von vornherein eine MSB-Stelle breiter anzulegen. Danach ist das Ergebnis auf Überlauf zu testen und gegebenenfalls der maximal darstellbare Wert einzusetzen. Die Kommastelle der Operanden bleibt im Ergebnis erhalten.
Eine $(n.m)_1 \pm (n.m)_2$ Operation erfordert also einen $(n + 1 + m)$ breiten Addierer/Subtrahierer.

Multiplikation: Hierbei addieren sich die Vor- und Nachkommaanteile der Operanden. Vom Ergebnis müssen, um im Produkt die Kommaposition der Faktoren weitergeben zu können (und das Datenformat der Operation zwischen Ein- und Ausgang zu erhalten), $m+n$ Stellen abgeschnitten werden. Die $(n.m)_1 * (n.m)_2$ Multiplikation ergibt also einen $2*(n + m)$ breiten Multiplizierer.

Division: Die Wortbreitenrelationen bei der Division von Festkommazahlen kann umgekehrt zur Multiplikation erklärt werden. Damit ein möglicher großer Quotient infolge Division mit einem Divisor < 1,0 dargestellt werden kann, sind hierfür ebensoviele Vorkomma-Bits zusätzlich vorzusehen, wie das Format des Divisors gültige Nachkommastellen aufweist. Für den inferierten *Synopsys*-Dividierer (bei dem die Eingänge und der Ausgang gleichbreit anzulegen sind) ist deswegen der $(n.m)$ Dividend mit m Nullen hinter dem LSB und der $(n.m)$ Divisor um m (mit Null initialisierte) Stellen vor dem ursprünglichen MSB zu erweitern. Das Format des Quotienten ist danach $(n+m.m)$, denn es gilt:

$$\text{Operandenbreite}_{\text{Dividend}} = \text{Operandenbreite}_{\text{Quotient}} + \text{Operandenbreite}_{\text{Divisor}} .$$

Sollen den Eingangsvariablen gleich formatierte Quotienten an nachfolgende Operationen weitergegeben werden, müssen im Falle eines Überlaufs (eines der zusätzlichen m Vorkomma-Bits des Quotienten ist gesetzt) gegebenenfalls Maximalwerte für den Quotienten substituiert werden.

Bitserielle Architektur

Bisher wurde stets von der Verarbeitung eines angepaßten (bitparallelen) Datenformats ausgegangen. Hätte vor allem ein geringer Flächenverbrauch höchste Priorität, käme eine weitere Architekturvariante für arithmetische Einheiten in Betracht: eine bitserielle Realisierung des Algorithmus', in der die Daten bitweise seriell miteinander verrechnet werden. Die notwendige Fläche für die Rechenwerke kann dabei auf Kosten der wachsenden Fläche für Steuerwerke und zu Lasten des Datendurchsatzes minimiert werden. Die Verarbeitung benötigt damit eine Verzögerung von mindestens einem Taktzyklus je zu verarbeitendem Bit. Ein praktischer Ansatz zur Implementierung bitserieller Rechenwerke und FIR-Filter in FPGA findet sich in [Andraka 1996] und [Andraka 1993]. Der große Vorteil besteht hierbei in der Ausnutzung der begrenzten FPGA-Hardware-Ressourcen. Schaltwerke sind aufgrund der inneren Struktur in FPGA wesentlicher einfacher und effizienter zu implementieren als große arithmetische Schaltnetze. Eine wesentliche Einschränkung sind die begrenzten Verdrahtungsressourcen des FPGA, so daß ein großflächiges Netz viel Verschnitt bei der Belegung der Logikzellen und hohe Verzögerungszeiten über lange Verdrahtungswege hervorruft.

Die Abbildung 4.18 zeigt beispielhaft das Prinzip eines seriellen *carry save* Addierers nach [Andraka 1993]. Der Volladdierer, bestehend aus zwei Halbaddierern und dem OR-Gatter, summiert die an X und Y anliegenden Operanden beginnend mit dem LSB. Die serielle Addition wird dadurch ermöglicht, daß Summenausgang und Übertrag in einem Register taktsynchron gespeichert und das Carry-Bit rückgekoppelt werden. Vor der Berechnung einer neuen Summe muß die Schaltung zurückgesetzt werden, um die Verfälschung durch gespeicherte Carry- und Summenbits auszuschließen.

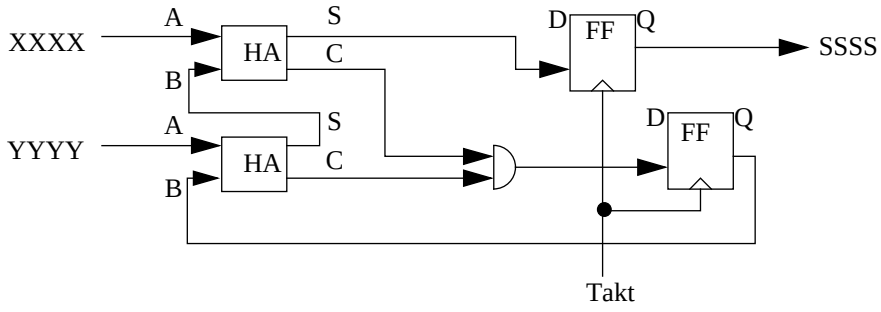


Abb. 4.18: Bitserieller *carry save* Addierer durch bitweise Taktung und Speicherung der Summen- und Carry-Bits [nach Andraka 1993].

Ein Nachteil dieser reinen Schaltwerkrealisierung besteht sicherlich in der mehrschichtigen und komplexen Automatenstruktur zur Steuerung der bitbreiten Arithmetikeinheiten. Aber das wesentliche Problem besteht im konzeptionellen Widerspruch mit dem Entwurfsfluß und dem Synthesewerkzeug (siehe Kap. 5), d. h. bitserielle Strukturen werden nicht unterstützt. Trotzdem zeigen praktische realisierte Designs (Kap. 3.3), daß durch manuelle Optimierungen und Abschätzungen geeignete Applikationen für eine bitserielle Arithmetik gefunden werden können, die ein vorteilhaftes Verhältnis aus Logikbedarf und Datendurchsatz aufweisen.

4.3 Psychoakustisch motivierte und synthesegerechte Architekturoptimierung

Die Feststellung, ob eine bestimmte Signalverarbeitungsstruktur den beabsichtigten Anwendungen gerecht werden kann, und die Bestimmung eines Fehlermaßes für die in Kauf zu nehmenden Abweichungen, ist ein immer wiederkehrendes Problem und auch für die vorliegende Arbeit von wesentlicher Bedeutung. Da es aus Ressourcengründen unmöglich ist, die IEEE-Fließkomma-Implementierung einer CPU zu übernehmen, tritt an deren Stelle eine den Möglichkeiten entsprechend realisierbare, aber geringer auflösende Implementierung. Bereits auf einer Prototypen-Ebene (C/C++) wird damit eine Bewertung der auftretenden Quantisierungsverzerrungen und Dynamikeinschränkungen durchgeführt, denn eine Simulation von VHDL-Kode ist aufgrund der langen Simulationszeiten hierbei vollkommen ungeeignet.

4.3.1 Verifikation des Prototypen

Die Wortbreitenreduktion verursacht in dem vorliegenden nichtlinearen, rückgekoppelten System der Nachregelschleifen - wie i. a. bei rekursiven digitalen Filtern - einen Rauschanteil im Ausgangssignal und kann außerdem zur Ausbildung von Grenzzyklen führen. Diese Systeminstabilitäten und Schwingungen entstehen einerseits direkt durch die Quantisierung und andererseits durch Dynamikbegrenzung und Überlauf. Das breitbandige Filterrauschen, welches das Signalrauschen des Eingangsquantisierers des digitalen Systems überlagert, wird vor allem durch Rundungsfehler nach arithmetischen Operationen erzeugt. Die Quantisierungsfehler in Filterkoeffizienten bedingen dagegen eine Frequenzgangverzerrung.

Für lineare Filter beispielsweise existieren analytische Beschreibungen der entstehenden Fehlersignale (Rauschen); Fehlerbeschreibungen und Stabilitätskriterien können auch für einige nichtlineare Systeme angegeben werden. Im Falle der mehrfachen rückgekoppelten Division im Perzeptionsmodell erweist sich ein solches Vorgehen aufgrund der Komplexität als nicht mehr praktikabel. Hierfür wurde begleitend zur schrittweisen Minimierung der internen Wortbreiten des Algorithmus' eine Systemsimulation mit typischen Sprachsignalen aus Standardtestdaten durchgeführt. Die Referenzimplementierung des ursprünglichen Algorithmus' mit IEEE-Fließkomma-Arithmetik wird dabei mit dem geringer quantisierenden Prototypen verglichen. Wie die nachfolgenden Abschnitte zeigen, werden auf diese Weise für die Nachregelschleifen des Perzeptionsmodells spezifische Datenformate und Operatoren sowohl im Fest- als auch Fließkomma-Format bestimmt.

Die Verarbeitungsunterschiede zwischen Original und Hardware-Prototyp könnten anhand mittlerer quadratischer Fehler bzw. anhand des Signal-Rauschabstandes beider Systeme bewertet werden. Dieses Vorgehen ließe allerdings außer acht, daß es sich hierbei ausschließlich um die Verarbeitung von Audiosignalen handelt, deren Verzerrungen und Rauschanteile nicht durch eine technische SNR-Bewertung, sondern letztendlich vom menschlichen auditorischen

System selbst beurteilt werden. Wie Arbeiten z. B. von [Gosse et al. 1997] für die Optimierung von Audio-Coder/Decoder Systemen zeigen[62], ist mit der Einführung eines perzeptuellen Fehlerkriteriums zur Anpassung der beteiligten Filterbänke eine deutlich verbesserte Übertragungsqualität möglich, als es bisher verwendete Fehlerkriterien vermögen, die das erreichte Maß an (perfekter) Rekonstruktion nach Rückumwandlung auswerten.

Da im Falle des Perzeptionsmodells bereits eine solche Applikation zur perzeptuellen Bewertung der Sprachübertragungsgüte existiert (siehe Kap. 3.4.2), wurde diese als eine Testbench für die Algorithmusoptimierung eingesetzt und die Auswirkungen der fortgesetzten Reduktion der internen Wortbreiten auf die Sprachqualität des ETSI-Testdatensatzes [ETSI 1992] solange durchgeführt, bis die Unterschiede in den Berechnungsergebnissen zum 32 Bit-IEEE-Fließkomma-Format des Originalmodells gerade noch akzeptabel waren. (Abb. 4.19)

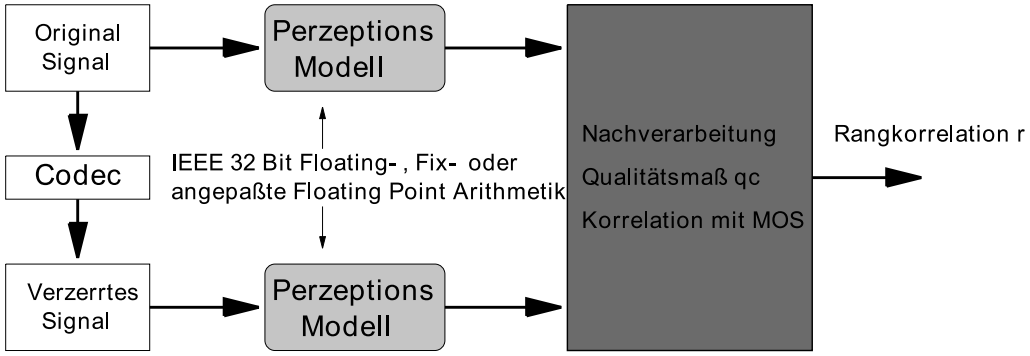


Abb. 4.19: Prinzip der Wortbreitenbestimmung mit der Anwendung Sprachgütebeurteilung.

Nach der Verarbeitung des unverzerrten und des verzerrten Signals erfolgt eine Frequenzgewichtung der Kanäle (ansteigendes Gewicht mit zunehmender Frequenz) und eine Kreuzkorrelation zwischen beiden Ausgangssignalen. Daraus wird das Qualitätsmaß q_c berechnet und in einem zweiten Schritt der Rangkorrelationskoeffizient r zwischen subjektiven (*MOS*) und objektiven (q_c) Sprachqualitätsmaß bestimmt. Die Größe des Korrelationskoeffizienten ist ein Maß für die Güte der Sprachqualitätsvorhersage. Der geringer quantisierende Prototyp wird nun aufgrund der internen Wortlängenbegrenzung diese Qualitätsvorhersage verschlechtern und damit Aufschluß darüber geben, wie stark die so berechnete interne Repräsentation des akustischen Signals bei Verwendung einer bestimmten (minimalen) Dimensionierung der arithmetischen Einheiten vom Original abweicht. Somit wird über die Differenzen im Qualitätsmaß indirekt die perzeptuelle Relevanz der Quantisierungsverzerrungen bestimmt und ein sicheres Maß zur Bestimmung einer minimalen Wortbreite der arithmetischen Strukturen bereitgestellt. Wie experimentelle Untersuchungen zur Sprachqualitätsbewertung zeigten (siehe

[62] Hierfür wurde ein auf dem MPEG1-Audio Standard beruhendes auditorisches Modell angepaßt, das im Kern aus einer 49-kanaligen FIR-Bandpaßfilterbank besteht, deren Mittenfrequenzen im Abstand von 0,5 Bark den Bereich von 0,5 bis 24,5 Bark überdecken.

Kap. 3.4.2), können perzeptuell relevante Verzerrungen im Perzeptionsmodell bis zu einer Verringerung des Korrelationskoeffizienten *r* auf etwa 0,9 hingenommen werden.

Obwohl dieses Vorgehen einen hohen Rechenaufwand zur Qualitätsbewertung aller 176 Doppelsätze des Testdatensatzes bedeutet, kann von der hohen Validität der korrekten Sprachqualitätsvorhersage unter Verwendung des Perzeptionsmodells partizipiert werden. Aus diesem Grunde wurde auch auf einen (alternativ möglichen und rechentechnisch vorteilhaften) direkten Vergleich der Ausgangssignale des originalen und des verzerrenden Perzeptionsmodells auf gleiche Eingangssignale verzichtet. Zur Bewertung der Abweichungen der objektiv berechneten Sprachqualität existiert keine äquivalente subjektive Bewertung (wie sie der MOS liefert) und damit kein unabhängiges Kriterium. Die Anwendung eines SNR-Kriteriums wäre zwar möglich, ist aber eben nicht das Ziel der Untersuchungen (s. o.).

Um trotzdem die beschriebene Methode anhand eines SNR-Kriteriums zusätzlich überprüfen zu können, wurden die oben beschriebenen Versuchsreihen mit künstlich auf die Testsätze des ETSI-Datensatzes aufaddiertem Rauschen durchgeführt. Dafür wurde auf die unverzerrten und (codec-)verzerrten Testsätze telefonbandgefiltertes weißes Gauß'sches Rauschen mit Signal-Rausch-Abständen von 0 - 40 dB addiert. Für jeden SNR wurde der Korrelationskoeffizient *r* zwischen objektivem Qualitätsmaß und MOS-Werten bestimmt, womit die in Abb. 4.20 gezeigte Kennlinie entsteht.

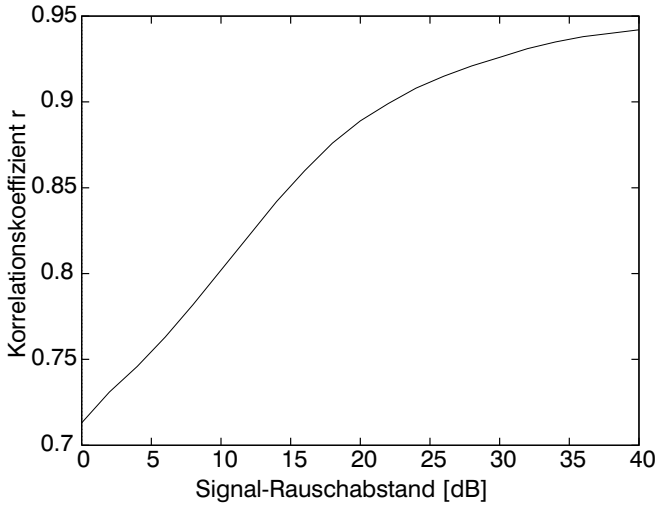


Abb. 4.20: Korrelationskoeffizient *r* (Gütemaß der erreichten Sprachqualitätsvorhersage) in Abhängigkeit des Signal-Rauschabstandes von additiv verrauschtem Sprachmaterial des ETSI-Datensatzes.

Durch das zusätzlich aufgebrachte Rauschen verringert sich die Korrelation der berechneten objektiven Sprachqualität mit der subjektiven Bewertung anhand des MOS. Auf diese Art und Weise kann beschrieben werden, inwieweit ein bestimmter Signal-Rausch-Abstand in beiden Eingangssignalen bei Verarbeitung mit dem originalen Perzeptionsmodell einer Verringerung der Korrelationskoeffizienten *r* durch Verarbeitung mit dem Hardware-Prototypen äquivalent ist. Bei hohen Signal-Rausch-Abständen konvergiert der Korrelationskoeffizient erwartungs-

gemäß zu dem bei der Verarbeitung unverrauschter Signale ermittelten Wert. Unter ca. 20 dB SNR verringert sich die Korrelation deutlich. Wenn als akzeptables Qualitätsniveau ein SNR von 30 dB angesetzt wird, dürfen sich äquivalent auswirkende Wortlängenverkürzungen im Algorithmus nur zu einer minimalen Verringerung der Korrelation mit den subjektiven MOS-Urteilen auf etwa 0,92 führen, womit auch die früher bestimmte Schranke (s. o.) bestätigt wird.

4.3.2 Festkomma- und Fließkomma-Arithmetik

Festkomma-Design

Da das Designsystem die direkte Implementierung von Festkomma-Daten -und -Operatoren unterstützt, wurde zunächst auch der Festkomma-Ansatz untersucht. Dazu wurde der originale Fließkomma-Algorithmus auf *integer* Operatoren umgestellt und empirisch mit dem Sprachqualitätskriterium auf kleinstmögliche Operandenbreiten optimiert [Schwarz et al. 1999a, Brucke et al. 1998] und [Brucke et al. 1999].

Die Abbildung 4.21 zeigt die Verarbeitungsstruktur der Nachregelstufe mit Festkomma-Arithmetik, wobei nur die Verarbeitung der *(n, m)* Festkommazahlen im Dividierer detailliert dargestellt ist. Diese Struktur wird im Zeitmultiplexbetrieb für alle fünf Nachregelschleifen sequentiell durchlaufen.

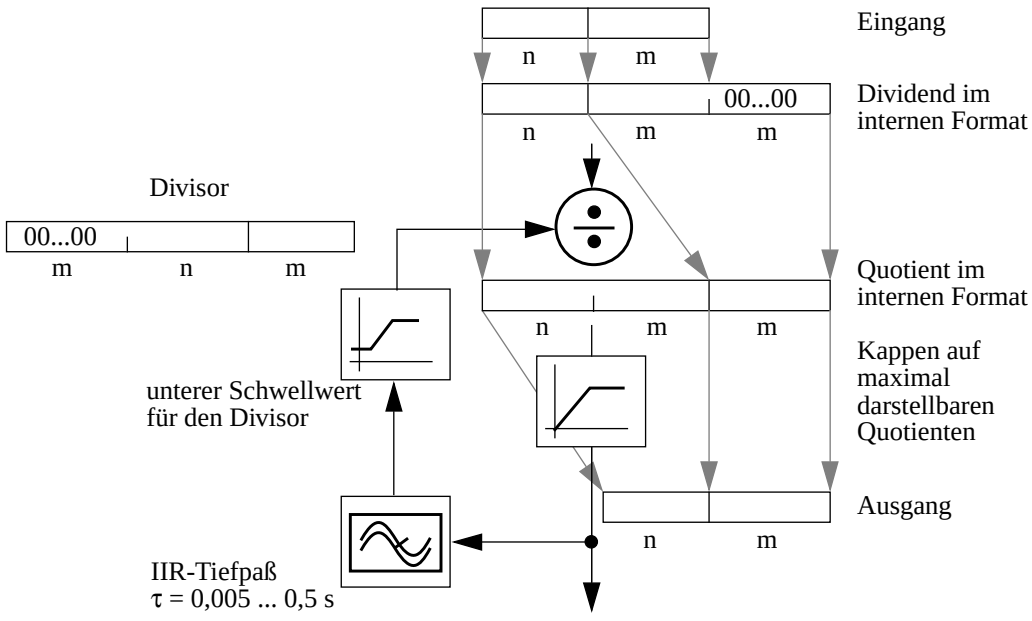


Abb. 4.21: Einsatz des *Synopsys DesignWare*-Dividierers für ein Festkomma-Design. Um zusätzliche Genauigkeits- und Dynamikverluste zu vermeiden, muß die interne Verarbeitungsbreite den einfachen Vorkommaanteil (n), aber die doppelte Nachkommastellenzahl (2 m) umfassen.

Das Eingangssignal der ersten Nachregelschleife ist zwar vorzeichenlos (*unsigned*) und enthält nur 16 binäre Nachkommastellen, die Division jedoch kann Werte erzeugen, die deutlich größer als 1,0 sind. Die Eigenschaften der Division von *(n,m)* Festkommazahlen erfordert deswegen die Einführung eines divisionsinternen erweiterten Formats, um ein Überschreiten des zulässigen Wertebereichs des Quotienten kontrolliert abfangen zu können. Da für den Multiplexbetrieb der Ausgang einer Nachregelschleife (gleichzeitig der Quotient) in der darauffolgenden Schleife nicht größer als der eigene Dividend sein darf, muß der Quotient trotzdem in der Regel gekappt werden.

Die Abb. 4.22 veranschaulicht zwei konträre Ergebnisse der Wortbreiten-Dimensionierung. Hierbei ist zu erkennen, wie aufgrund unzureichender bzw. ausreichender Rechengenauigkeit ein signifikanter Unterschied im Sprachqualitätsmaß entsteht. Getestet wurde dabei mit dem ETSI-Sprachdatensatz bei mittleren Pegeln. Bereits an der Verteilung der einzelnen Einträge (Meßpunkte), ist zu erkennen, daß aufgrund der unzureichenden Quantisierung in Abb. 4.22 (b) die Korrelation zwischen objektivem und subjektivem Qualitätsmaß stark verringert ist: die Punktwolke, die sich idealerweise um eine Gerade konzentriert, fällt deutlich erkennbar auseinander.

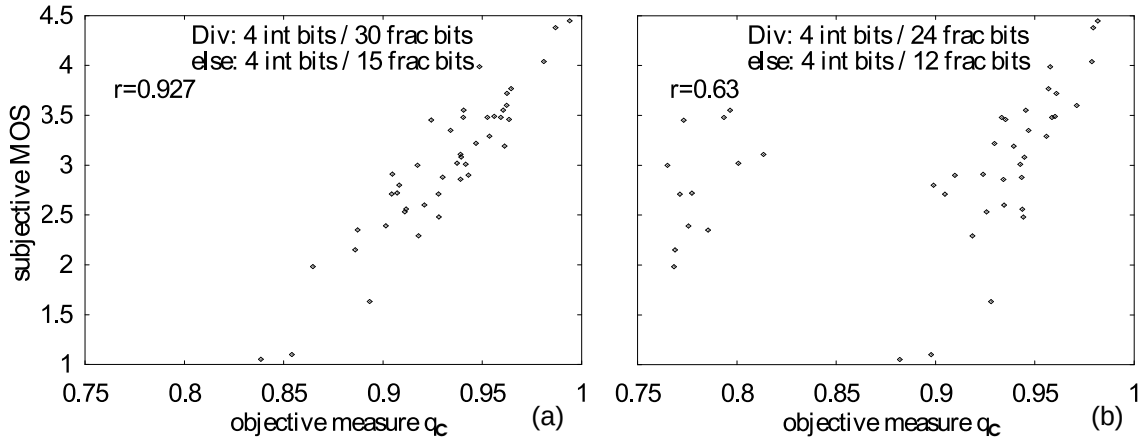


Abb. 4.22: Die Bestimmung des Sprachqualitätsmaßes für offensichtlich ausreichend dimensionierte Nachregelschleifen (a) bei 34 Bit Wortlänge in der Division und 19 Bit bei allen anderen arithmetischen Operationen. In (b) dagegen wird bereits bei einer Reduktion um 6 Bits Nachkommastellen in der Division und 3 Bits in den anderen Operationen nur noch ein Rangkorrelationskoeffizient *r* von 0,63 erreicht.

Damit konnte durch systematisches Verringern der Wortbreite ein Festkomma-Format gefunden werden, daß minimale Quantisierungsverzerrungen und Dynamikbegrenzungen bei minimalem Hardware-Aufwand (Wortbreite) aufweist. Die Schwellwertbildung und die Tiefpässe im Chip 2 sind im Vergleich zur Division insgesamt unkritisch, da der Dividierer aufgrund seiner Größe bereits bestimmt, ob das Design überhaupt realisierbar ist.

Die bei der Schaltungssynthese für den Festkomma-Entwurf erzielten Ergebnisse hinsichtlich des Flächenverbrauchs und der Schaltnetzverzögerung zeigt Tab. 4.9.

Design	**ES2-0,7 µm**	
	Fläche	**Schaltnetzverzögerung auf dem längsten (kritischen) Pfad**
Nachregelschleife (mit 34 Bit-Division) **nach Flächenoptimierung**	6,6 mm^2	760 ns
Nachregelschleife (mit 34 Bit-Division) **nach Taktratenoptimierung**	12,8 mm^2	250 ns
	Altera Flex10K100A (100.000 Gatteräquivalente)	
Festkomma-Dividierer (34 Bit) **flächenoptimal**	1.186 Logikzellen, (24 % Füllungsgrad für das gesamte FPGA)	1.527 ns

Tab. 4.9: Entwurfsergebnisse mit einem Festkomma-Design bei 34 Bit Wortbreite für eine Standardzell-Technologie (ES2-07) und für ein *Altera*-FPGA. Allein 24 % aller Gatterressourcen in dem *Flex10K100*-Baustein werden von einem Dividierer verbraucht.

Der Chip 2 hat 4,096 µs Zeit, um einen Eingangswert zu verarbeiten, wobei in dieser Zeit u. a. fünf Divisionen berechnet werden müssen. Damit ist das *Altera*-FPGA für Sequentialisierung eindeutig zu langsam und angesichts des Füllungsgrades für einen einzelnen Dividierer auch zu klein. Das Problem des Festkomma-Entwurfs, sein immenser Flächenbedarf gekoppelt mit langen Verzögerungszeiten, macht eine Implementierung in dieser Technologie unmöglich; es erzwingt einen ASIC-Entwurf und erreicht hier dennoch eine beträchtliche Chip-Fläche.

Fließkomma-Design

Die Festkomma-Division ist somit das entscheidende Problem und legt zwingend die Suche nach Alternativen nahe. Da der Algorithmus keine andere Operation an dieser Stelle zuläßt, kann nur versucht werden, eine platzsparende Architekturvariante bzw. ein alternatives Datenformat einzusetzen.

Eine logarithmische bzw. Fließkomma-Darstellung besitzt den offensichtlichen Vorteil, daß die eigentliche Division (Multiplikation) im Dividierer (Multiplizierer) nur mit den vergleichsweise kleinen Operandenlängen der Mantissen rechnen muß. Die Mantissenlänge, die ja die Präzision der Quantisierung bestimmt, ist selbst im 32 Bit-IEEE-Fließkomma-Format mit 23 Bit deutlich kleiner, als der Festkomma-Entwurf (s. o.) verlangt. Da der Flächenbedarf exponentiell mit der Wortlänge zusammenhängt, erscheint die Realisierbarkeit dieser Operatio-

nen nun wesentlich aussichtsreicher. Darüber hinaus zeigen die Ausführungen in Kap. 4.1.3, daß für die Verarbeitung von Sprachsignalen mit den dabei auftretenden Verteilungsdichtefunktionen ein Fließkomma-Format die ohnehin besser angepaßte Darstellung ist. Andererseits ergibt sich aus der fehlenden Unterstützung im CAD-Werkzeug und den notwendigen Konvertierungen zwischen Fest- und Fließkommaformat ein deutlicher Anstieg des Implementierungsaufwandes.

Für die Entwicklung eines Neuroprozessors konnte in [Wüst et al. 1998] allerdings anhand der Implementierung eines *lokalen rekurrenten neuronalen Netzes* (LRNN) zur Spracherkennung gezeigt werden, daß eine Kombination aus Fest- und Fließkomma-Arithmetik durchaus auf einen vorteilhaften Kompromiß zwischen minimalem Hardware-Aufwand und ausreichender Verarbeitungsbreite führen kann[63]. Begünstigt durch die Gauß-Verteilung der Netzgewichte wurde hier anstatt der sonst üblichen Quantisierung mit 8 Bit Festkommazahlen eine Fließkomma-Multiplikation vorgesehen. Unter Beibehaltung einer höheren Festkomma-Wortbreite für die Addition wurde die Fließkomma-Multiplikation ohne signifikanten Qualitätsverlust für 5 Bit-Operanden (1 Bit Vorzeichen, 1 Bit Mantisse und 3 Bit Exponent) reduziert.

Da im Chip 2-Algorithmus wesentlich höhere Wortbreiten erforderlich sind, und damit der Aufwand zur Umwandlung von Fest- in Fließkommazahlen und umgekehrt steigt, ist eine ähnliche Trennung in die zwei Formate so vorzunehmen, daß nur wenige Umkonvertierungen notwendig werden. Für die Nachregelschleifen ist eine reine Fließkomma-Implementierung sinnvoll; der Skalierer und der Tiefpaß verbleiben im Festkomma-Teil, da hier nur addiert und multipliziert wird. Diese Unterteilung erfordert nur eine Umkonvertierung des 16 Bit-Festkomma-Eingangs in das Fließkomma-Format, eine Rückkonvertierung nach der letzten Nachregelschleife und eine Wortlängenanpassung an das wiederum 16 Bit breite Festkomma-Ausgangsformat (siehe Abb. 4.23).

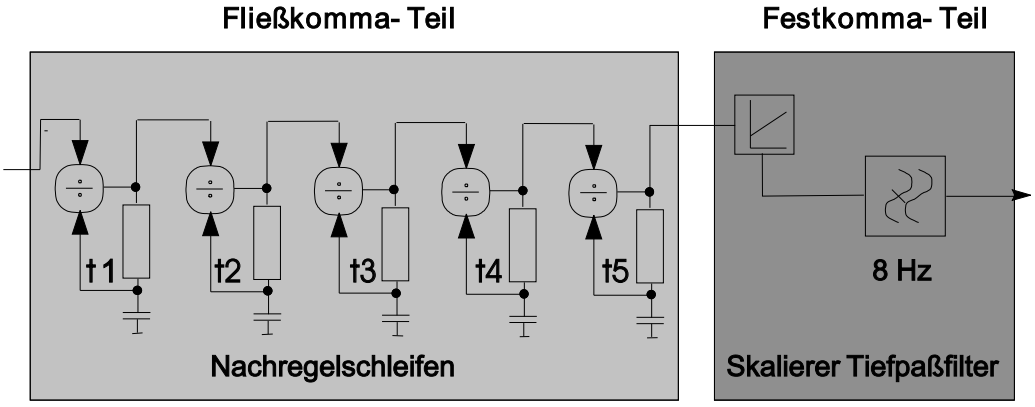


Abb. 4.23: Aufteilung des Algorithmus in einen Fließ- und Festkomma-Teil.

[63] Auf einem *Xilinx* XC4020-FPGA konnten bei einer Logikressourcen-Auslastung von 82 % 255 Neuronen (externer Speicher für die Gewichte, interner Aktivitätsspeicher, 4 parallele Multiply-Accumulate Einheiten) einschließlich der benötigten Multiplex-Logik und einer 8 Bit-Schnittstelle zur Hostkommunikation mit einem PC realisiert werden.

Die Implementierung des C/C++ Prototypen sollte der komfortablen Bestimmung der Wortbreiten und -effekte dienen. Dazu wurde die C++ Klasse *xfloat* für einen skalierbaren Fließkomma-Datentyp einschließlich der benötigten arithmetischen Operationen und Hilfsfunktionen entwickelt, die eine bitgenaue Simulation ermöglicht [Bitterling 1999].

Die Festkomma-Operationen im Skalierer und im Tiefpaß des Prototypen wurden direkt umgesetzt und arbeiten mit normalen Integer-Operationen der CPU direkt auf Long-Integer-Variablen.

Die Fließkomma-Klasse *xfloat* hat folgende Eigenschaften:

- Die *xfloat*-Klasse ist ein beliebig skalierbarer generischer Datentyp aus Mantisse und Exponent.
- Ein n-stelliger Exponent ist mit einer Verschiebung von 2^{n-1} Exzeß-kodiert.
- Operanden und Operationen besitzen kein Vorzeichen, nur nichtnegative Werte sind gültig.
- Enthalten sind alle Grundoperationen (Addition, Subtraktion, Multiplikation, Division) und eine Vergleichsoperation („<“).
- Berechnungsergebnisse sind immer normalisierte Zahlen und haben dafür eine führende „1“ im MSB.
- Zur Vereinfachung und besseren Überprüfbarkeit der Operationen ist kein „hidden“ Bit vorgesehen.
- Die Ergebnisse der Operationen sind immer gerundet (*round to nearest*).
- Bei Über- oder Unterlauf werden die maximal bzw. minimal darstellbaren Werte substituiert.
- Enthalten sind Konvertierungsfunktionen von und nach Festkomma und zwischen unterschiedlichen Fließkomma-Formaten.
- Sie besitzt zusätzlich Hilfs- und Konvertierungsroutinen zur Darstellung und Verifikation interner Zahlenrepräsentationen.

Vom vorgeschlagenen Fließkomma-Format profitieren vor allem die Division und die Multiplikation, da deren zentrale Operationen nur noch in Mantissenbreite ausgeführt werden. Damit wird eine drastische Flächenbedarfsreduktion, insbesondere bei der Hardware-Division, erreicht.

Nachteile des Fließkomma-Formats bestehen bei der Addition und der Subtraktion. Hier müssen vor der eigentlichen Operation zunächst die Exponenten der Operanden durch Verschiebung einer Mantisse angeglichen werden. Dabei kann es zur Auslöschung einer Mantisse und

dem Verlust aller signifikanten Stellen kommen: jeweils ein Operand kann komplett verschwinden. Im Falle der Subtraktion ähnlicher Werte sind sogar stark fehlerhafte Ergebnisse möglich (‚dirty zero" Problem [Spaniol 1976]).

Die Tabelle 4.10 gibt einen Überblick über die Bestandteile der Fließkomma-Klasse *xfloat*. Dieser Klasse ist noch eine weitere Klasse (*xint*) untergeordnet, die der *xfloat*-Klasse skalierbare und generische Integer-Typen für Mantissen- und Exponentenverarbeitung zur Verfügung stellt.

Name	**return Typ**	**Parameter**	**Verwendung**
xfloat	-	-	Konstruktor für statische Elemente
xfloat	-	unsigned,unsigned	Konstruktor mit Breitenangabe
exc	base_t[a]	-	liefert aktive Verschiebung für Exponenten
fval	double	-	aktiver Wert als *double float*
maxquant	double	-	größter darstellbarer Wert
minquant	double	-	kleinster darstellbarer Wert ≠ 0
str	char*	-	aktiver Wert als Klartext
norm	-	-	Normalisierung
conv	xfloat	unsigned,unsigned	Konvertierung
resize	-	unsigned,unsigned	Breiten nachträglich ändern
=	xfloat&	double|xfloat	Zuweisung auf xfloat
+	xfloat	xfloat	Addition
-	xfloat	xfloat	Subtraktion
*	xfloat	xfloat	Multiplikation
/	xfloat	xfloat	Division
==	bool	xfloat,xfloat	Test auf Gleichheit
<	bool	xfloat,xfloat	Test auf „kleiner als“ (Relation)
directvalue	base_t	-	interne Repräsentation lesen
directassign	-	base_t	interne Repräsentation schreiben

a. Der Return-Typ *base_t* wird an zentraler Stelle auf einen Basisdatentyp zugewiesen. Um bei Bedarf größere Wortbreiten verwenden zu können, wird hier mit *long long int* gearbeitet.

Tab. 4.10: Bestandteile und Struktur der generischen Fließkomma-Klasse *xfloat*.

Die Tabelle 4.11 stellt die nach [Hennessy und Patterson 1996, Shirazi et al. 1995] implementierten allgemeinen Fließkomma-Operatoren für die Grundrechenarten zusammen:

Operator	**Name**	**Operation für Basis 2, M = Mantisse, e = Exponent**
+	*xfloat_add*	$(M_1,e_1) + (M_2,e_2) = \begin{cases} (M_1 + M_2 \cdot 2^{-(e_1 - e_2)}, e_1) & \text{falls } e_1 \geq e_2 \\ (M_1 \cdot 2^{-(e_2 - e_1)} + M_2, e_2) & \text{falls } e_1 \leq e_2 \end{cases}$
-	*xfloat_sub*	$(M_1,e_1) - (M_2,e_2) = \begin{cases} (M_1 - M_2 \cdot 2^{-(e_1 - e_2)}, e_1) & \text{falls } e_1 \geq e_2 \\ (M_1 \cdot 2^{-(e_2 - e_1)} - M_2, e_2) & \text{falls } e_1 \leq e_2 \end{cases}$
*	*xfloat_mul*	$(M_1,e_1) \cdot (M_2,e_2) = (M_1 \cdot M_2, e_1 + e_1)$
/	*xfloat_div*	$\frac{(M_1,e_1)}{(M_2,e_2)} = \left(\frac{M_1}{M_2}, e_1 - e_1\right)$

Tab. 4.11: Die vier Grundrechenoperatoren für *xfloat*-Fließkommazahlen.

Im Vorgriff auf nachfolgende Kapitel ist zu erwähnen, daß bereits nach dem ersten Entwurf der Fließkomma-Klasse Iterationen über den kompletten Hardware-Syntheseprozeß erfolgen mußten. Insbesondere anhand der Division wurde der entscheidende Kompromiß zwischen Datenformat und Realisierbarkeit erarbeitet.

Nach entsprechenden Simulationszyklen mit dem Prototypen und unter Berücksichtigung der Gegebenheiten des Synthesewerkzeugs (Festkomma-Operatoren der *Synopsys-DesignWare*), der Randbedingungen (Flächenbedarf, Schaltnetzverzögerung) für das Design und eines durchgängigen und übersichtlichen Entwurfsflusses (siehe Kap. 5) wurden die in der Tab. 4.12 aufgeführten Operandenbreiten als beste Kompromißlösung bestimmt.

Diese Übersicht gibt die mit Sprachqualitätstests ermittelten minimalen Wortbreiten für den Fließkomma- und den Festkomma-Teil an. Die Fließkomma-Operanden und -Konstanten haben das gleiche Format. Nicht so im Skalierer und im Tiefpaß; hier mußten, da die *Xilinx*-FPGA-Zieltechnologie nur begrenzte Multipliziererbreiten zuläßt (der rechte Operand darf

nicht größer als 18 Bit sein), unsymmetrische Multiplikationen mit 18 Bit-Festkomma-Konstanten vorgesehen werden.

Fließkomma-Operatoren	Dividierer (6.6 / 6.6) (interne Wortbreite für Mantissendivision: 12 Bit)	Mantisse (m = 6), Exponent (n = 6) Max-Wert 111111 . 111111 Min-Wert 100000 . 000000 Null 000000 . 100000 Verschiebung 100000
	Multiplizierer (14.6 * 14.6), Addierer (14.6 + 14.6), Subtrahierer(14.6 - 14.6)	Mantisse (m = 14), Exponent (n = 6) Max-Wert 11111111111111 . 111111 Min-Wert 10000000000000 . 000000 Null 00000000000000 . 100000 Verschiebung 100000
Festkomma-Operatoren (2er Komplement)	Multiplizierer (5,15 x 18)	18 Bit Konstanten (nur Nachkomma) x 5 Bit Vor-, 15 Bit Nachkomma
	Subtrahierer (5,15 - 5,15)	5 Bit Vor-, 15 Bit Nachkomma - 5 Bit Vor-, 15 Bit Nachkomma
	Addierer (7,13 + 7,13)	7 Bit Vor-, 13 Bit Nachkomma + 7 Bit Vor-, 13 Bit Nachkomma

Tab. 4.12: Minimal mögliche Wortbreiten und Operatoren im Chip 2 in der Schreibweise (Mantisse.Exponent) für Fließkomma-Operationen und (Vorkomma,Nachkommastellen) bei Festkomma-Operationen.

Algorithmusoptimierung

Zur Ermittlung der Quantisierungsparameter war es darüber hinaus notwendig, den Algorithmus so umzustellen, daß Zwischenergebnisse und Konstanten in gut quantisierbaren Wertebereichen abgebildet werden. Hinsichtlich der Zahlenrepräsentation erwies sich die originale Tiefpaßgleichung als ungeeignet (nach Gl. (4.3)). $C1_i$ liegt sehr nahe bei 0,0 und wegen $C2_i$ = 1,0 - $C1_i$ ist $C2_i$ nahe bei 1,0. Für das Fließkomma-Format ist naturgemäß die Quantisierung um 0,0 genauer als bei 1,0 und da die Filtergleichung umgeformt werden kann, wird nur $C1_i$ verwendet. Dabei sind die zwei Varianten A und B möglich (siehe Tab. 4.13), wobei die Variante B nur noch eine Multiplikation enthält. Außerdem können der Skalierer und der Tiefpaß zusammengefaßt und damit eine Multiplikation eingespart werden.

Die reellwertigen Konstanten in Tab. 4.13 wurden nach Abbildung der ursprünglichen Konstanten (Tab. 4.3) in die spezifischen Fest- und Fließkomma-Formate daraus wieder ins Dezimalsystem rückgerechnet und gerundet. Die aus *C5* und *C3* zusammengefaßte Konstante *C7* wird skaliert und um drei Bits nach rechts verschoben, damit die Operanden der Addition in Gl. (4.30) gleiche Kommapositionen haben. Diese Skalierung um 1/8 verschiebt die zu inter-

pretierende Kommaposition im 16 Bit-Ausgangssignal von Chip 2 um drei Stellen nach rechts. Nach Abschneiden der vier niederwertigsten Bits im 20-Bit-Vektor *o[k]* besitzt das Ausgangssignal des Algorithmus' 10 Vorkomma- und 6 Nachkommastellen (2er Komplement). Durch die Abbildung der Berechnungen und Konstanten auf die lokalen Datenformate verschiebt sich auch der Ausgangswert der Nachregelschleifen auf rückgerechnet 1,03125 und 1,34375 am Modellausgang für den minimalen Eingang (LSB gesetzt) $y_{min} = 2^{-16} \cong 1{,}5*10^{-5}$.

<table>
<tr><th>Operation</th><th colspan="3">Optimierte Berechnung</th></tr>
<tr><td>Adapt.-Division i
i = [0, 1, 2, 3, 4]</td><td colspan="3">$q_0[k] = \frac{x[k]}{b_0[k-1]}$ (1. Schleife) Gl. (4.26)

$q_i[k] = \frac{q_{i-1}[k]}{b_i[k-1]}$ (andere) Gl. (4.27)</td></tr>
<tr><td>Adapt.-Tiefpaß i
Variante A

oder

Variante B</td><td colspan="3">$b_i[k] = \begin{cases} C1_i \cdot q_i[k] - C1_i \cdot b_i[k-1] + b_i[k-1] & bei \quad b_i[k] > thr_i \\ thr_i & sonst \end{cases}$ Gl. (4.28)

$b_i[k] = \begin{cases} C1_i \cdot (q_i[k] - b_i[k-1]) - b_i[k-1] & bei \quad b_i[k] > thr_i \\ thr_i & sonst \end{cases}$ Gl. (4.29)</td></tr>
<tr><td>Skalierer und abschließender Tiefpaß zusammengefaßt</td><td colspan="3">$\frac{o[k]}{8} = C7 \cdot (q_4[k] - C4) + C6 \cdot \frac{o[k-1]}{8}$ Gl. (4.30)</td></tr>
<tr><td>optimierte Konstanten</td><td>$C1_1$ = 0,04877090454
$C1_2$ = 0,00498771667
$C1_3$ = 0,00193607807
$C1_4$ = 0,00098764896
$C1_5$ = 0,00049990415</td><td>thr_0 = 0,00316238403
thr_1 = 0,05623245239
thr_2 = 0,23713684082
thr_3 = 0,48696899414
thr_4 = 0,69781494140</td><td>C4 = 0,697814941
C6 = 0,987575531
C7 = 0,513874054
(C7= C5*C3/8 aus Gl. (4.4) und Gl. (4.5))</td></tr>
</table>

Tab. 4.13: Optimierte Operationen in den Teilblöcken.
x[k]: Eingangswert eines Frequenzkanals
$q_i[k]$: Quotient in der Adaptationsschleife i
$b_i[k]$: Ausgang des Tiefpasses in der Adaptationsschleife i
s[k]: Ausgang Skalierer
o[k]: Ausgang eines Frequenzkanals
thr_i: Schwellwerte für die Divisoren
$Cj_{(i)}$: Konstanten
k: Zeitinkrement, Takt

Im Kap. 5.2 wird gezeigt werden, daß algebraische Vereinfachungen im Tiefpaß der Regelschleifen nicht notwendig zu Vorteilen beim Ressourcen-Sharing und Scheduling führen. Vorteilhaft ist die Zusammenfassung von Skalierer und Tiefpaß, weil die Multiplikationskonstante in einen ähnlichen Wertebereich wie die gefilterten Signale transformiert wird.

Verifikation des Fließkomma-Algorithmus

Die Berechnung des Fehlers der vier Grundoperationen ist systematisch nur für die Multiplikation und Division möglich, weil bei der Addition und der Subtraktion durch die erwähnten Effekte bei der Mantissenanpassung ein Operand vollständig „verschwinden“ kann. Zusätzlich besteht bei der Subtraktion das „dirty zero“ Problem, wenn ähnlich große Fließkomma-Operanden $M * b^e$ (mit m Mantissenstellen) voneinander subtrahiert werden. Vom Ergebnis kann dann nur gesagt werden, daß es im Bereich $-b^{e-m} ... b^{e-m}$ liegt; daher sollte es nicht ohne weiteres als „Null“ interpretiert werden [Spaniol 1976].

Bei einer Basis b und der jeweiligen Präzision von m Mantissenstellen ergibt sich ein größter relativer Quantisierungsfehler $\varepsilon_m = b/2 * b^{-m}$ (Maschinen-epsilon ε_m [Goldberg 1991]). Das entspricht bei einer (*round to nearest*) Rundung dem Wert der höchstwertigen nicht mehr darstellbaren Mantissenstelle und ergibt:

$\varepsilon_{m\_6} = 0{,}015625$ bei einer (6.6)-Fließkommazahl

$\varepsilon_{m\_14} = 0{,}000061035$ bei einer (14.6)-Fließkommazahl

Es ist zu erwarten, daß die Ergebnisse einer Multiplikation bzw. der Division entsprechend der Fehlerfortpflanzung den jeweiligen Summenfehler der relativen Eingangsfehler der Operanden aufweisen werden. Die Addition und insbesondere die Subtraktion können darüber hinaus durch die Teilauslöschung bei der Anpassung der Exponenten und Mantissen des kleineren Operanden einen höheren und nicht systematischen Fehler generieren.

Unter den gegebenen Projekt-Bedingungen war es jedoch weder möglich noch Absicht, vollständig implementierte und verifizierte IEEE-konforme Fließkomma-Einheiten erneut zu entwerfen. Aufgrund der Komplexität einer formalen Verifikation wird zur Überprüfung der korrekten Funktion ein Black-Box-Testverfahren mit einer hohen Anzahl Testmuster genutzt und eine Evaluierung gegen 64 Bit-IEEE-Fließkommawerte *(double precision)* vorgenommen[64]. Die Testmuster wurden dabei so erzeugt, daß auf den gesamten *(M.e)*-Bitvektor der *xfloat*-Zahl eine gleichverteilte Zufalls-Integerzahl (*rand*()) der Länge $m+n$ kopiert wurde, die dadurch auch nicht normalisiert ist. So erreicht man eine Gleichverteilung über dem Logarithmus der

[64] In [O’Leary et al. 1999] wird eine Methode für die formale Verifikation einer *Intel* Floating Point Unit vorgestellt. Während für die Bewertung der *fmul* und *fdiv* Funktionen eine IEEE-gemäße komplexe Verifikation gegen eine höherpräzise Berechnung unter Verwendung der zur Verfügung stehenden Rundungsverfahren vorgenommen wird, geht man im Falle der Addition und Subtraktion auch auf einen Black-Box-Test zurück.

Operanden im darstellbaren Wertebereich. Diese Testbench wurde später so erweitert, daß mit ihr auch die VHDL-Prototypen gegen die C++ Implementierung getestet werden konnten. In Abb. 4.24 sind die relativen Fehler der mit 10.000 Pseudozufallszahlen gegen 64 Bit-IEEE-Werte (*double precision)* getesteten Operatoren dargestellt.

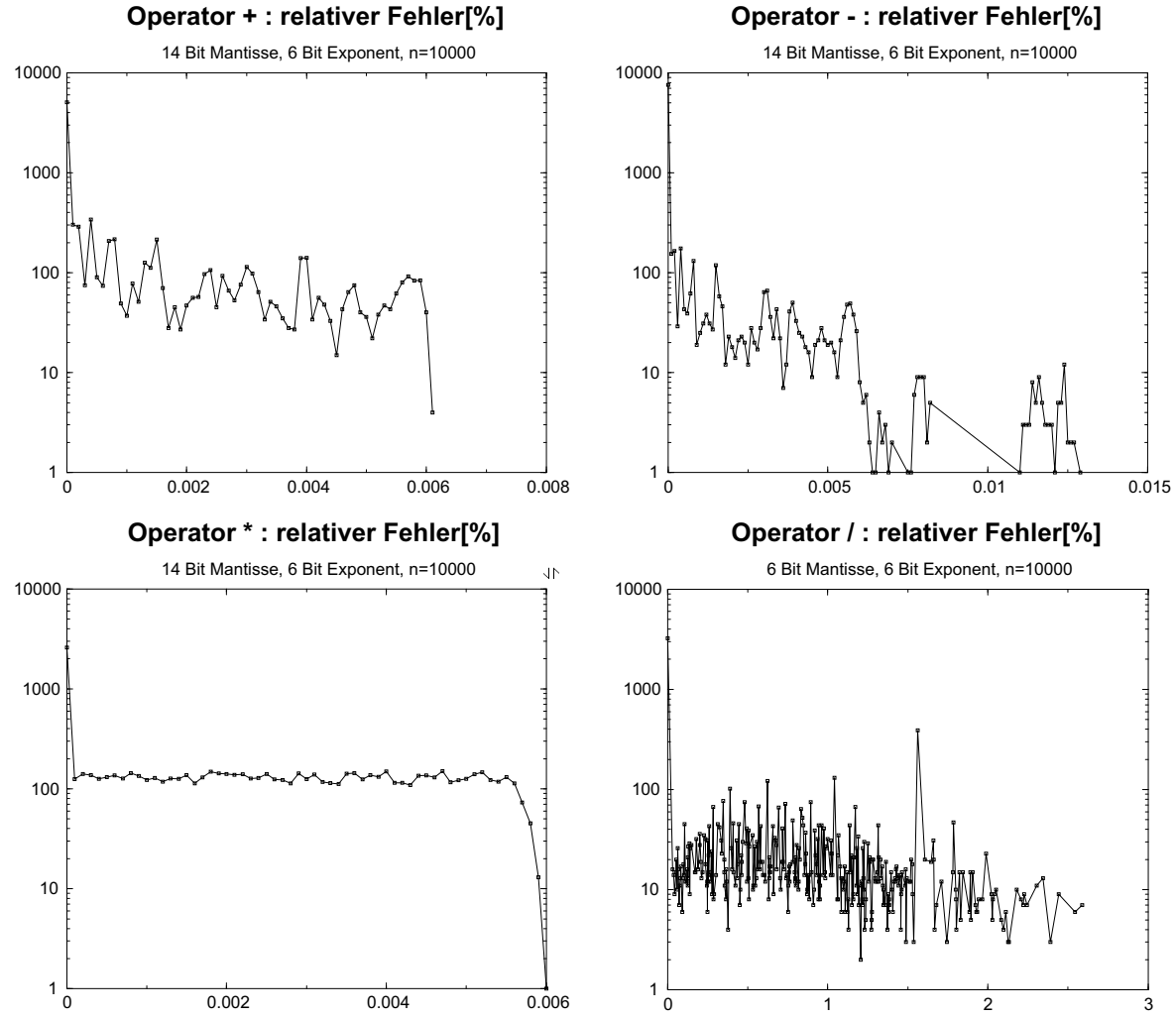


Abb. 4.24: Verifikation der C++ *xfloat* Fließkomma-Operatoren. Histogramme des relativen Fehlers gegenüber 64 Bit-IEEE-Operationen (*double float*).

Dabei zeigt sich, daß sowohl die Addition als auch die Multiplikation den erwarteten relativen Fehler als Summe der relativen Quantisierungsfehler der Operanden aufweist, der durch die *round-to-nearest* Rundung wieder halbiert wird ($\varepsilon \leq \varepsilon_{m\_14}$). Die Division deutet auf einen eigentlich unerwarteten Präzisionsverlust von etwa einer Mantissenstelle hin ($\varepsilon \leq 2*\varepsilon_{m\_6}$). Die Ursache hierfür ist in der Verarbeitung der nicht normalisierten Operanden zu suchen, da trotz einer entsprechenden Dimensionierung der Festkomma-Divison für die Mantissen (intern 12 Bit) immer noch Bereichsüberschreitungen im Quotienten auftreten können. Besonders bei der Subtraktion zeigen sich die erwarteten erhöhten Fehler ($\varepsilon > \varepsilon_{m\_14}$), die durch die o. g. Auslöschungseffekte entstehen.

Da keine allgemeingültige (IEEE-konforme) Arithmetik entworfen werden soll und kann, wird die Funktionstüchtigkeit der Operatoren hauptsächlich an den Ergebnissen im Sprachqualitätstest mit dem Perzeptionsmodell selbst und dem ETSI-Datensatz gemessen. In Abb. 4.25 wird die 32 Bit-IEEE-Fließkomma-Version der optimierten *xfloat*-Implementierung gegenübergestellt. Bei der Ermittelung dieser unteren Schranke für die Wortlängen wurde zunächst nur die Division, unter Beibehaltung des vollen 32 Bit-IEEE-Fließkomma-Formats für alle anderen Operationen reduziert. Es zeigte sich sehr schnell, daß die Division auf Operanden mit 6 Mantissenbits und 6 Bit Exponent optimiert werden konnte (interne Wortbreite der Mantisendivision: 12 Bit). Die ursprüngliche Tiefpaßgleichung dagegen erforderte bis zu 20 Bit Mantisse in der Multiplikation, was nicht signifikant unter den 23 Bit des IEEE-Formats liegt.

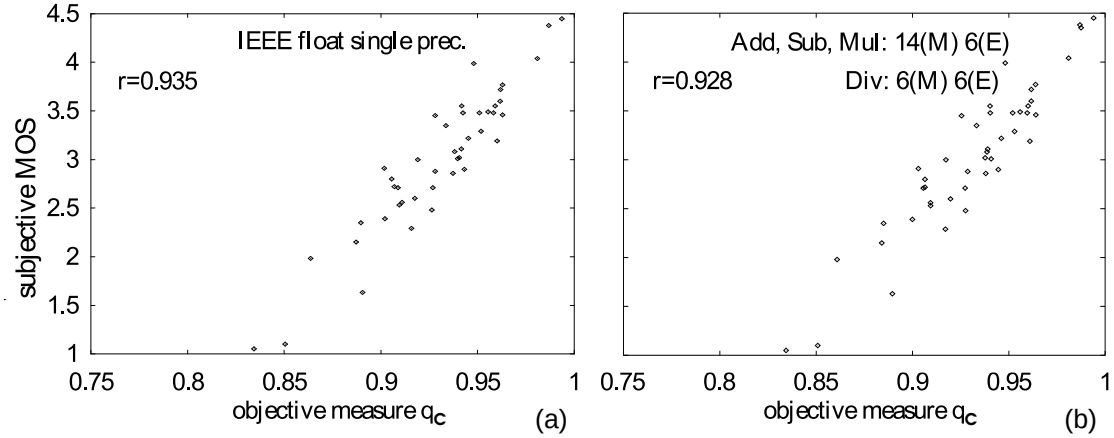


Abb. 4.25: Sprachqualitätstest mit 32 Bit-IEEE-Format (*single precision)* (a) und den optimierten Operanden im *xfloat*-Format (b). Für die weiteren Operationen im nunmehr kombinierten Skalierer/Tiefpaß bei (b) wurde außerdem die bereits in Tab. 4.12 spezifizierte Festkomma-Arithmetik eingesetzt.

Die Umstellung durch Konstantenersetzung (Optimierung Variante A, Gl. (4.28)) erbrachte durch die günstigere Konstantenquantisierung einen entscheidenden Fortschritt: 14 Mantissenbits erwiesen sich als ausreichend für alle anderen Operationen in den Nachregelschleifen. Weil außerdem der interne Aufwand für die Division auf 12 Bit statt der 34 Bit für die Festkomma-Variante reduziert werden konnte, ist eine Schaltungssynthese in greifbare Nähe gerückt. Auch weitere Bedingungen bzw. Einschränkungen, wie z. B. die der begrenzten Multipliziererbreiten der *Xilinx*-FPGA-Technologie, sind damit als akzeptabel nachgewiesen.

4.4 Zusammenfassung

Ausgehend von den Entwurfsanforderungen (Kap. 4.1), den Designsystemen, den algorithmischen und Designalternativen, den Möglichkeiten und der Verfügbarkeit von Halbleitertechnologien, der Designmethodik und den aus der Literatur bekannten Systementwürfen (Kap. 3) kann ein Raum von Designentscheidungen aufgespannt werden (siehe Abb. 4.26).

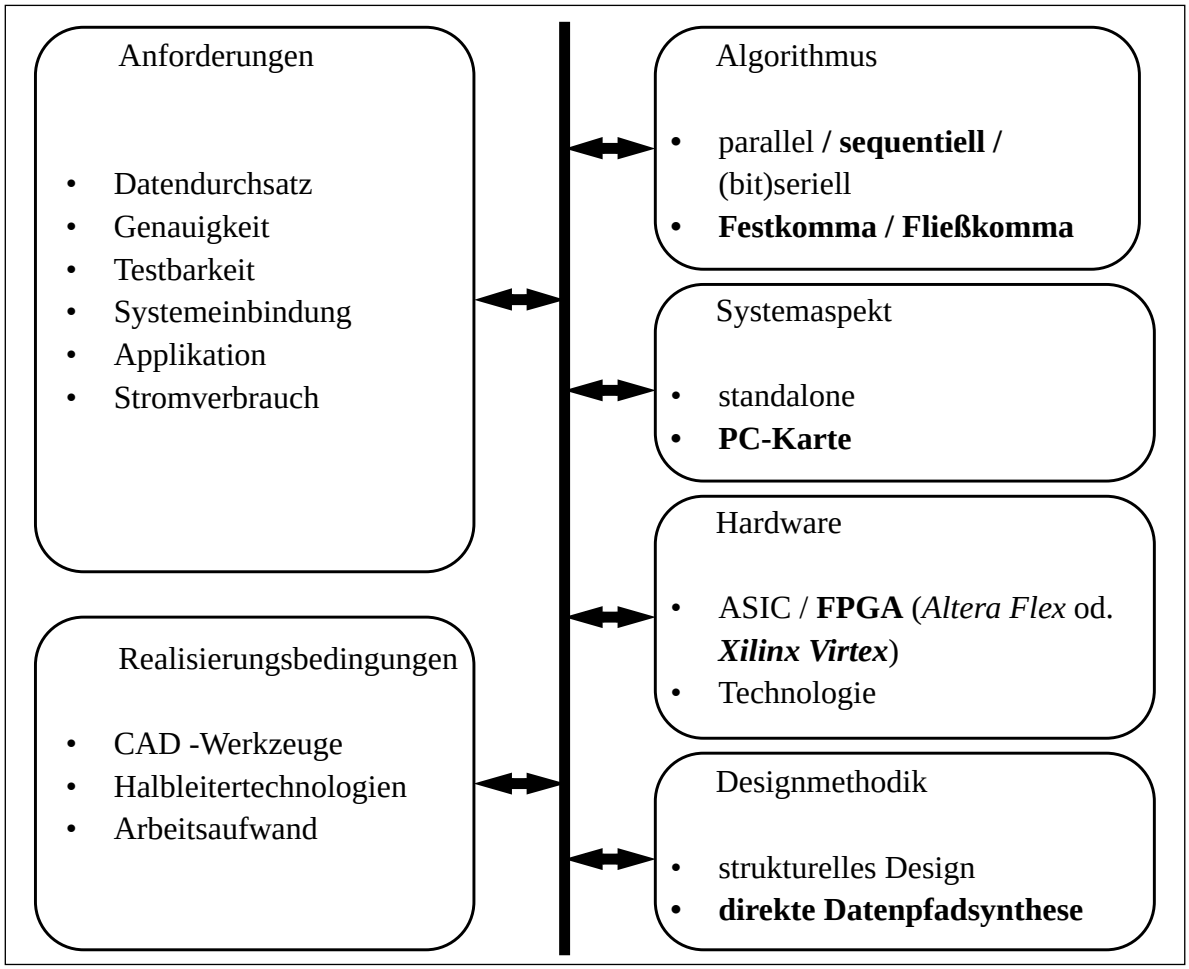


Abb. 4.26: Mögliche Designentscheidungen für die Hardware-Umsetzung der Signalverarbeitung in den Nachregelschleifen des Perzeptionsmodells. Die **fett** hervorgehobenen Einträge bezeichnen die letztendlich ausgewählten bzw. realisierten Optionen.

Aus den Anforderungen der Applikationen aber auch durch das Vorhandensein der automatischen Designsysteme zur Digitalschaltungssynthese, wird die Umsetzung des Algorithmus in ausschließlich digitale Schaltungen vorgenommen. Der geforderte hohe Berechnungsumfang verhindert aufgrund der begrenzten Schaltungsressourcen (der vom CAD-System handhabbaren Halbleitertechnologien) den Einsatz beschleunigender paralleler Strukturen und erzwingt ein sequentielles Design. Damit besteht die Zielarchitektur aus einem minimalen, die benötigten Operatoren möglichst nur einmal enthaltenden Rechenwerk. Durch geeignetes Scheduling ist dieses Rechenwerk entsprechend den Datenabhängigkeiten im Zugriff zu optimieren, wobei

die Implementierung der arithmetischen Einheiten und des Steuerautomaten infolge der Echtzeitbedingungen allerdings höchste Anforderungen an den erzielbaren Datendurchsatz und an die Qualität der Synthese stellt. Ebenfalls aufgrund der knappen Logikressourcen muß ein Übergang von der naheliegenden Festkomma-Architektur auf Fließkomma-Architekturen erfolgen. Der Chip 2 muß primär überhaupt synthetisierbar sein und mit einer verfügbaren Technologie gefertigt werden können. Obwohl eine Aufteilung des gesamten Perzeptionsmodells auf mehrere Chips vorgenommen wurde, könnte die Zusammenfassung auf einem einzigen Chip eine durchgängige Synthese bei gleichen Randbedingungen und die Integration in Zielssysteme erleichtern. Das sekundäre globale Ziel für die Flächenbedarfsminimierung des Chip 2 ist demnach die mögliche Zusammenführung mit Chip 1 in einem Design. Der notwendige Datendurchsatz kann durch die Abtastrate des Chip 2 genau angegeben werden: in 4,096 µs ist ein Stereoabtastwert zu verarbeiten.

Die Voraussetzung dafür ist wiederum eine ausreichend schnelle Technologie. Um gleichzeitig dem Experimentierstatus eines Prototypensystems gerecht zu werden, kommen nur reprogrammierbare Schaltungsarrays - FPGA - der *Xilinx Virtex*-Technologie in Betracht. Verfügbare Bausteine enthalten bis zu 1.000.000 Gatteräquivalente, so daß die o. g. Zusammenführung der Teildesigns möglich erscheint.

Da solche FPGA-Systeme mittlerweile auf PC-Einsteckkarten verfügbar sind, ergibt sich ein multifunktionales Hardware-System, daß über den Host-PC an die Software-Applikation angebunden werden kann.

Der Hardware-Synthese steht an dieser Stelle ein verifizierter Prototyp des Algorithmus mit verringerter Wortlänge und Ressourcenbedarf zur Verfügung. Partiell konnte und mußte dessen Synthetisierbarkeit bereits im Vorfeld bestimmt werden. Außerdem existiert durch den C/C++ Prototypen eine Referenzimplementierung, mit dem einige Applikationen bereits ausgetestet werden können.

Mit der Sprachgütequalität, die das reduzierte System noch übertragen kann, ist eine ideale Methodik gefunden, um das perzeptiv relevante Maß an Verzerrungen und Quantisierungsartefakten und damit einen minimal dimensionierten Algorithmus zu bestimmen.

Hervorzuheben ist insbesondere, daß das bisherige Vorgehen nicht streng sequentiell erfolgen konnte. Die Entscheidung darüber, ob eine synthetisierte Schaltung zu brauchbaren bzw. überhaupt zu Resultaten führt, wurde hier im Rahmen der Spezifikationen (siehe Kap. 4) schon getroffen, obwohl genaue Entwurfsuntersuchungen erst danach erfolgen konnten. Die Anforderungen, Applikationssysteme und Realisierungsbedingungen initiieren nur Ansatzpunkte für die Spezifikation; darüber hinaus herrscht ein starker dynamischer Einfluß sogar rückwirkend auf die Ausprägung und Adaption des Algorithmus. Im nächsten Kapitel wird beschrieben, zu welchen Ergebnissen diese Wechselwirkungen unter dem Aspekt der automatischen Schaltungssynthese und -optimierung führen.

Kapitel 5

High-Level-Schaltungssynthese für das Perzeptionsmodell

Ausgehend von der Analyse und Spezifikation des Algorithmus und des aufzubauenden Gesamtsystems werden in diesem Kapitel die Designalternativen, einzelne Schritte und Ergebnisse für die Realisierung des Chip 2 des Perzeptionsmodells in eine integrierte Schaltung dargestellt. Ein struktureller (Teil-)Entwurf wird der direkten Datenpfadsynthese gegenübergestellt. Direkt einsetzbar waren beide Wege vorerst nicht. Der strukturelle Entwurf erfordert einen sehr hohen Aufwand beim stufenweisen bottom-up Aufbau des Designs. Alle Teilkomponenten müssen nach Einzeluntersuchungen erst im lokalen Zusammenspiel getestet werden, bevor ein timing-gerechter Zusammenbau in der nächst höheren Ebene möglich ist. Dieser „handgemachte“ Entwurf, der einer Implementierung des Blockschaltbildes gleichkommt, erfordert das Design eines oder mehrerer Steuerautomaten als arbeitsintensivste Komponente. Die direkte Datenpfadsynthese dagegen führt sehr schnell auf ein komplettes Design samt automatischer Generierung eines Steuerautomaten, wenn der Entwurfsfluß entsprechend vorbereitet ist. Vom Entwurf geeigneter Operatorschaltnetze über die Spezifikation notwendiger Bibliothekselemente bis hin zur Erarbeitung eines funktionierenden Entwurfsflusses wird im folgenden der gesamte Ablauf von der Synthese der VHDL-Beschreibung bis hin zur korrekt simulierenden zeitbewerteten (*back-annotation*) Netzlistenbeschreibung des Hardware-Layouts dargestellt.

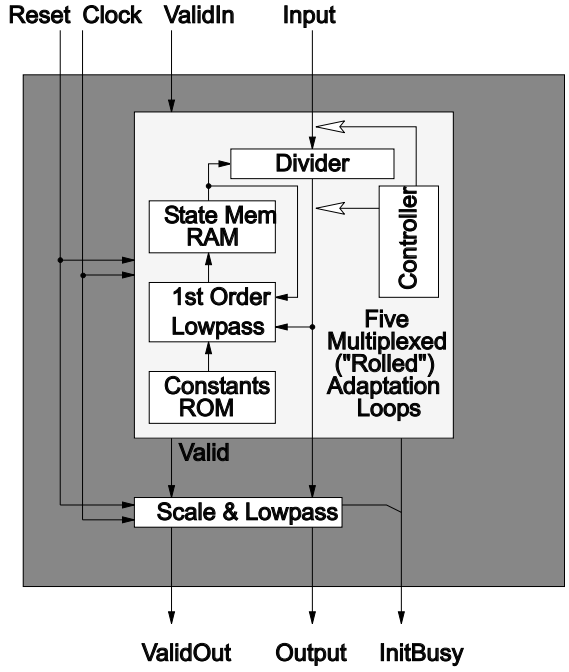


Abb. 5.1: Blockschaltbild für Chip 2. Ein Rechenwerk mit Variablen- und Konstantenspeicher kommuniziert flag-gesteuert mit der Modellumgebung.

Die Abbildung 5.1 zeigt das allgemeine Blockschaltbild des Chip 2 mit den Nachregelschleifen aus Systemsicht. Ziel ist es, die Arithmetik einer Nachregelschleife sequentiell für alle fünf Schleifen und alle Frequenzkanäle zu nutzen. Auf die Konstanten und temporären Speicher muß abhängig vom Schleifen- und Frequenzkanalindex zugegriffen werden. Der Skalierer und der Tiefpaß wurden zusammengefaßt. Die Synchronisation des Datentransfers mit der Systemumgebung und innerhalb des Chips erfolgt mit einfachen Flags (*valid*). Neben einem Rücksetzeingang (*reset*) und der Taktversorgung (*clock*) muß der Systemumgebung noch die Initialisierungsphase der RAM-Speicher (*initbusy*) nach einem Reset mitgeteilt werden.

5.1 Strukturelle gegenüber verhaltensbasierter Algorithmus-Synthese

5.1.1 Entwurfsfluß für ein strukturelles Design

Die klassische Methode - der Aufbau eines strukturellen Designs - erfordert die komplette manuelle Ausarbeitung der notwendigen Ressourcen, des Scheduling und den erst danach möglichen Entwurf eines Steuerautomaten. Dieser hat sowohl die Ein- und Ausgangsports und Speicheroperationen als auch die Schleifendurchläufe und Operationsfolgen des Algorithmus zu steuern. Dafür müssen alle kritischen Teilpfade durch z. B. Operatorschaltnetze vorher manuell bestimmt werden. Aus Gründen der Übersichtlichkeit eines solchen Entwurfs wurde zunächst auch mit einem solchen Entwurfsfluß unter Verwendung von Festkomma-Operatoren der *Synopsys DesignWare* begonnen.

Der Entwurfsfluß für einen solchen Zugang unter Verwendung von *Synopsys* wird in der Abb. 5.2 dargestellt. Das strukturelle Modell muß keine reine Register-Transfer-Beschreibung sein, denn die Verfügbarkeit komplexer Funktionsblöcke, (Arithmetik, Registerstrukturen, spezielle Funktionseinheiten) in der *Synopsys DesignWare* erlaubt neben einer wirklichen Instanz auch das „Inferieren“ von z. B. Operatoren (+, -, *, /), siehe auch Kap. 4.2.4.

Nach einer syntaktischen Analyse des Verhaltensmodells wurde das Modell mit dem *Synopsys FPGA Compiler* synthetisiert und die Beschreibung der entstandenen Schaltung als *edif*-File[65] in das herstellerspezifische Backend-Werkzeug gegeben. Hier erfolgt das Plazieren und Verdrahten, in FPGA die Belegung der Designspeicher und deren Verknüpfung. Reale Angaben zu Signalverzögerungen und Flächenbedarf lassen sich erst danach geben.

Die begleitende Simulation auf allen Designstufen war jedoch bei der Verwendung der *Synopsys FPGA-Compiler*-Bibliotheken nicht möglich. Die entscheidende Simulation ist die

[65] EDIF - *Electronic Design Interchange Format.*

Netzlistenbeschreibung des Layouts (*back-annotation*), die aus der plazierten und verdrahteten Schaltung unter Berücksichtigung der Leitungslängen und Gatterlasten generiert wird. Zwischen der Simulation der Verhaltens- und Register-Transfer-Ebene, die nur das logische korrekte Verhalten nachweisen kann und der langwierigen Simulation der Layout-Netzliste unterstützt *Synopsys* bei FPGA mit Look-Up-Tables (*Altera Flex10K* und *Xilinx Virtex*) keine Zwischensimulation auf Gatterebene. Die eigentlich notwendige Simulation realer Datensätze ist so nur noch auf Verhaltensebene möglich, da die Simulationszeiten der i. a. megabyte-großen VHDL-Netzlisten der Register-Transfer- und Layout-Ebene jeden zeitlichen Rahmen sprengen. Hier konnten nur noch Stichproben genommen werden.

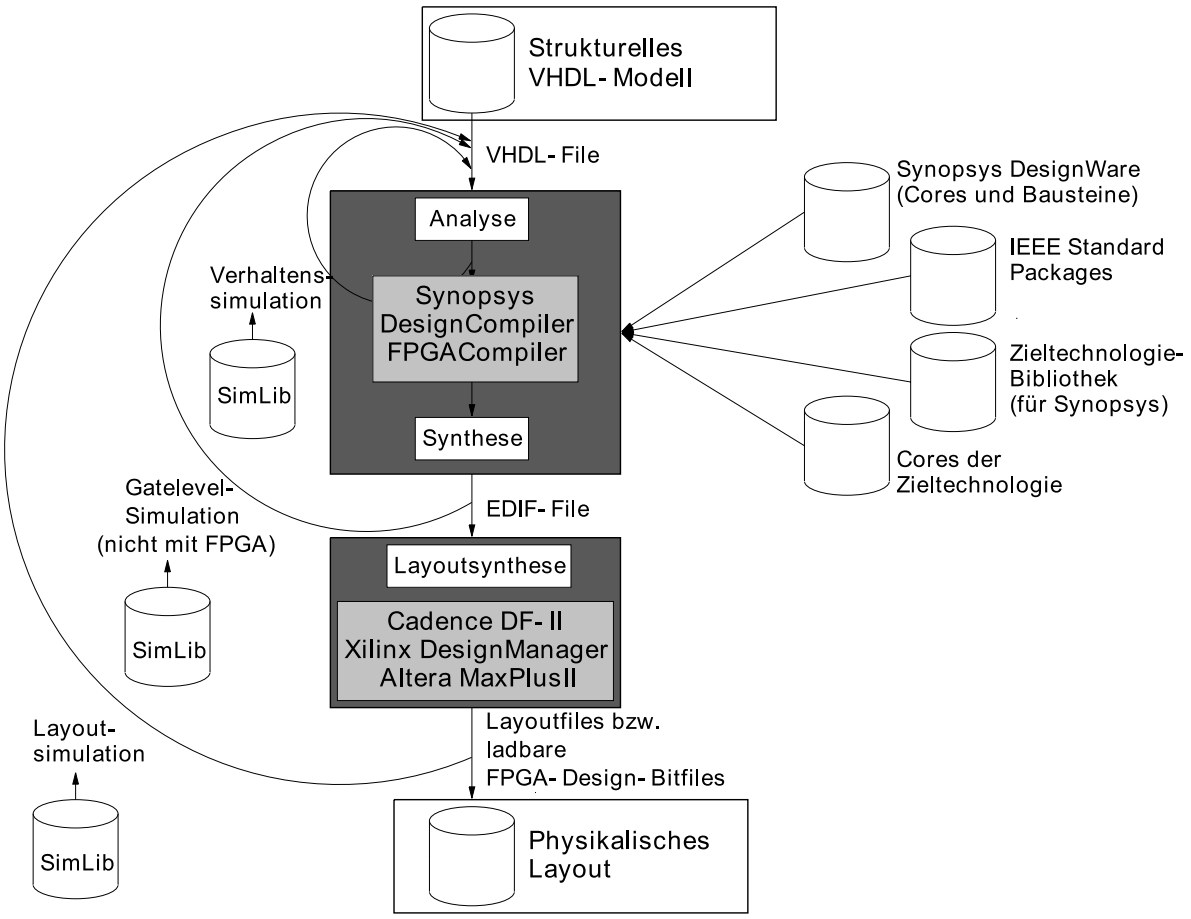


Abb. 5.2: Entwurfsfluß zur Schaltungssynthese und Simulation des strukturellen Entwurfs mit *Synopsys*. Diese Abfolge aus Designzyklen und Simulation steht für alle drei untersuchten Zieltechnologien. Die FPGA-Bibliotheken bieten als Besonderheit architekturoptimierte Operator-Makros an.

Die Abbildung 5.3 zeigt das Blockschaltbild einer manuell entworfenen Nachregelschleife, für die neben der eigentlichen Signalverarbeitung ein interner Steuerautomat, entsprechende Register (Flip-Flops) zum Halten und Synchronisieren der Ausgangswerte und die *enable*-Leitungen für diese Register vorzusehen sind. Die eigentliche Aufgabe besteht jedoch darin, manuell eine Flußsteuerung zu finden, die nicht nur die Datenabhängigkeiten berücksichtigt, sondern

auch ein zeitliches Auslastungsmodell (Scheduling) im Zusammenhang mit einer Mehrfach-Nutzung der Hardware-Ressourcen (Ressourcen-Sharing) ermöglicht. Das Multiplizierer-Schaltnetz im IIR-Tiefpaß kann z. B. in der skizzierten Weise nicht mehrfach genutzt werden, da der Tiefpaß dafür noch entsprechende Register-Files an Ein-und Ausgängen benötigen würde. Die Aufgabe des Steuerautomaten besteht hauptsächlich darin, mit Hilfe interner Zähler stabile Ausgangsdaten an den Schaltnetzen abzuwarten, die Datenabhängigkeiten zu berücksichtigen und, soweit möglich, ein Ressourcen-Sharing zwischen Hardware-Komponenten zu steuern. Daraus ist ersichtlich, wie komplex die Entwurfsaufgabe wird, wenn genau das für die drei Verarbeitungsebenen realisiert werden soll: lokal in einer Schleife, zwischen den fünf Schleifen eines Kanals und zwischen den 30 verschiedenen Frequenzkanälen.

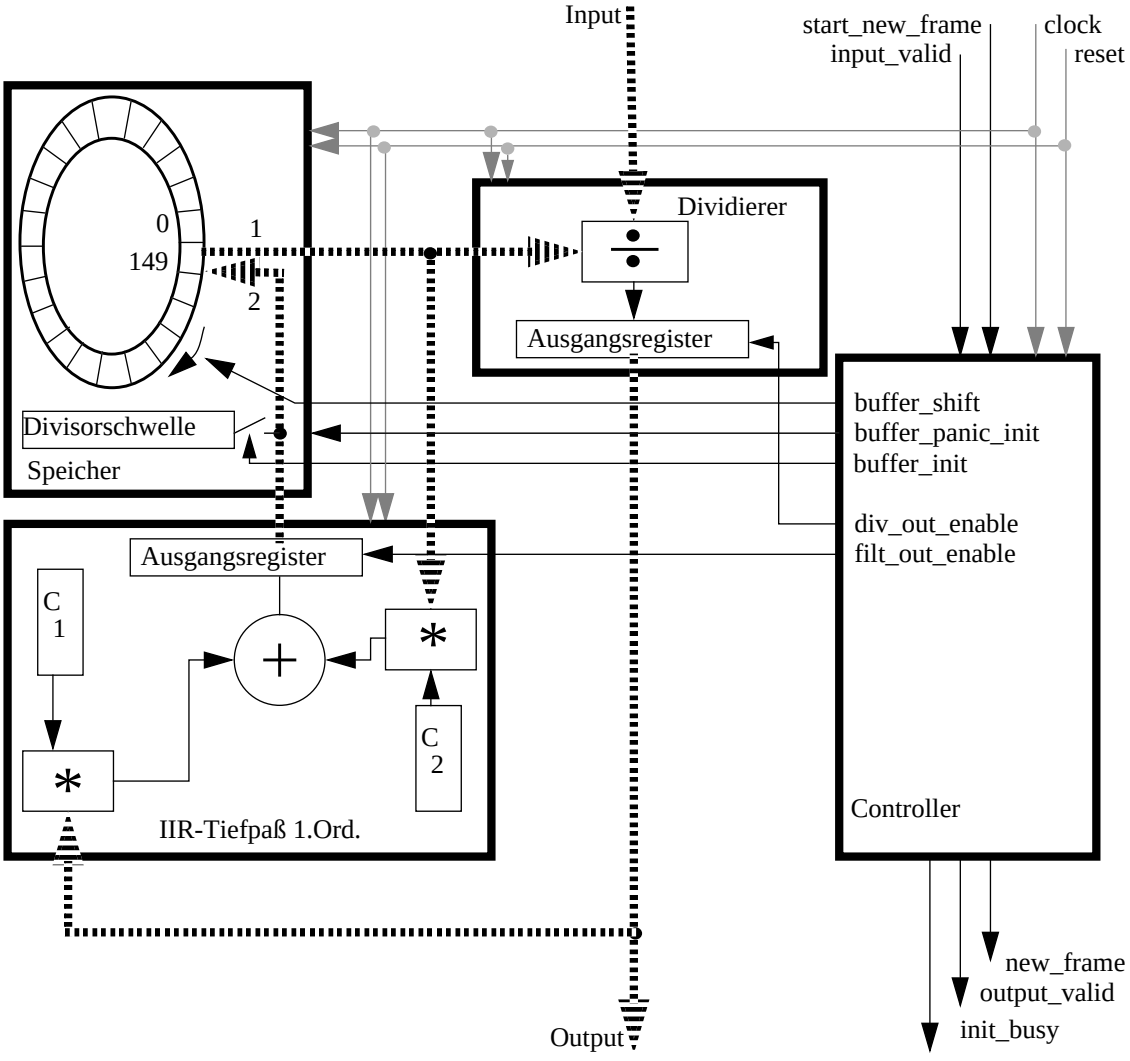


Abb. 5.3: Nachregelschleife, wie sie für einen hierarchischen und strukturbasierten Entwurf vorbereitet und gegliedert wurde. Die vier *entities* (Dividierer, Speicher, Controller, Tiefpaß) sind mit Datenpfaden, lokalen Steuer- und Flag-Signalen und globalen Signalen (*clock, reset*) verbunden.

5.1.2 Direkte Datenpfadsynthese mit dem *Synopsys Behavioral Compiler*

Die direkte Synthese einer Schaltung aus einer speziellen Verhaltensbeschreibung des Modells ist mit dem *Synopsys Behavioral Compiler* möglich. Da hier alle Schritte zur Analyse des Designs und letztendlich zur Synthese des Steuerautomaten automatisch vorgenommen werden, ändert sich auch der Entwurfsfluß (siehe Abb. 5.4). Das Ergebnis einer erfolgreichen *„Behavioural Synthesis“* ist nicht nur der Steuerautomat, sondern auch eine optimal dimensionierte Hardware, d. h. ein Ressourcen-Sharing wird nicht nur für die großen (expliziten) Arithmetiknetze gefunden, sondern auch für die impliziten Steuersignale und Register.

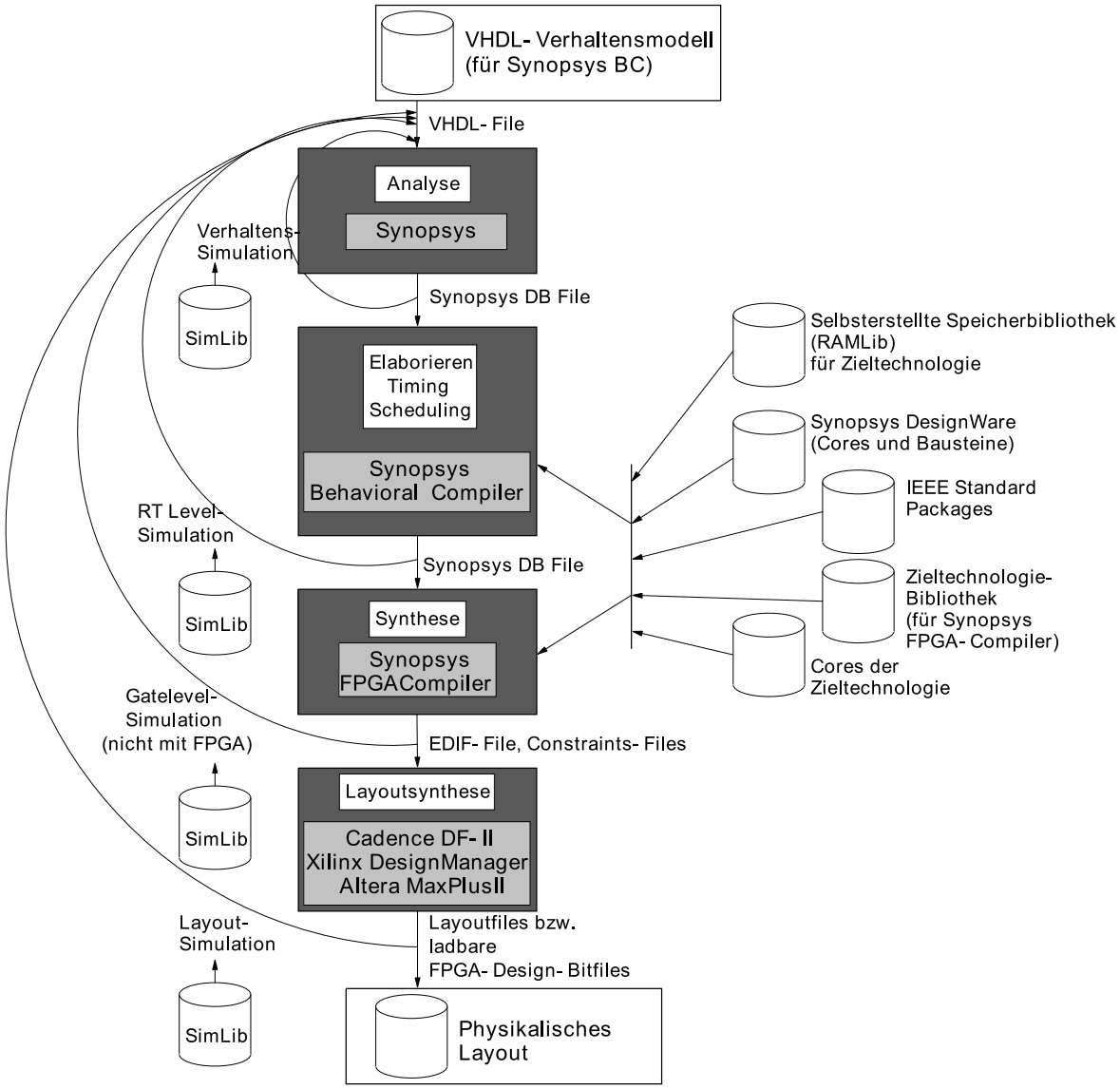


Abb. 5.4: Entwurfsfluß zur direkten Schaltungssynthese und Simulation des Verhaltensmodells mit dem *Synopsys Behavioral Compiler* und der Layout-Generierung für verschiedene Technologien.

Nach der syntaktischen Prüfung wird mit dem *elaborate* das synthetische Design zusammengestellt. Die insgesamt notwendige Hardware, die der VHDL-Kode explizit und implizit verlangt, wird hier in einer Schaltung aus symbolischen (aber synthetisierbaren) Blöcken (Arithmetik, Multiplexer, Selektoren, Einzelgatter) zusammengesetzt. Mit dem *bc_time_design* werden alle Pfade untersucht und zeitlich ausgemessen; schließlich findet das *schedule* eine optimierte Operationsabfolge zusammen mit einem Steuerautomaten. Für die Implementation der Nachregelschleifen bedeutete dies äußerst umfangreiche Iterationen zwischen VHDL-Kode und der Netzlistenbeschreibung der Register-Transfer-Ebene, da das Scheduling einer Vielzahl sich gegenseitig beeinflussender Parameter unterliegt, die manuell kombiniert und ausprobiert werden müssen. Nun wird das Design mit *compile* vollständig synthetisiert und optimiert. Mit dem Export der *edif*-Netzliste erfolgt der Übergang in das Backend-Werkzeug von *Xilinx* und die Plazierung und Verdrahtung der FPGA-Zellen und Pads.

Ein erfolgreiches Scheduling ist an viele Voraussetzungen und Vorarbeiten gebunden:

- Die *„Behavioral Synthesis“* erfordert eine angepaßte VHDL-Kodierung. Das Scheduling arbeitet nur auf einzelnen Prozessen, die mit dem gewünschten Takt arbeiten. Die im Algorithmus der Nachregelschleifen vorhandenen geschachtelten Schleifen über 30 Kanäle und fünf Regelschleifen stellen gewissermaßen den Standard-Anwendungsfall für diese Art der Datenpfadsynthese dar. Das VHDL-Modell ist in der Struktur dem C-Kode ähnlich.

- Damit diese Beschreibung für den *Behavioral Compiler* verwendbar ist, werden neben dem synthetischen RAM-Block zur Variablenspeicherung die synthetischen Operatoren in Form von Bibliotheksfunktionen oder -blöcken benötigt (siehe Abb. 5.5).

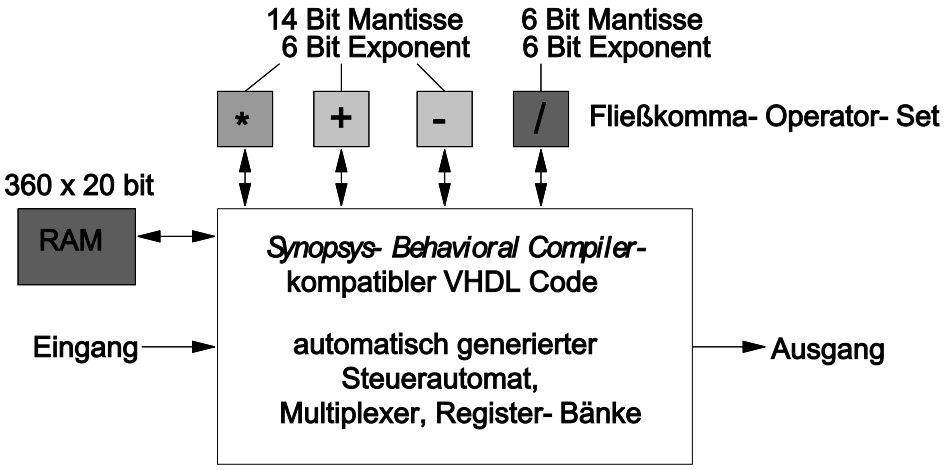


Abb. 5.5: Die direkte Datenpfadsynthese mit dem *Behavioral Compiler* setzt eine kompatible synthetische RAM-Bibliothek, direkt umsetzbare Operator-Blöcke und den speziellen VHDL-Kode voraus.

- Für die synthetische RAM-Bibliothek muß zuerst der *Xilinx Core-Generator* bemüht werden: hiermit wurde ein 20 x 360 Bit großer RAM aus drei 8 x 512 Bit technologiespezifischen *Block-RAM* Zellen mit den für RAM üblichen Steuer-Pins (*Read/Write, Chip-Enable, Reset, Clock, VCC, Ground*) generiert und als *edif*-Netzliste exportiert. Diese muß jedoch vor der Verwendung mit *Synopsys* manuell an den Stromversorgungs-Pins mit einem Textersetzungsskript repariert werden, sonst wird der RAM-Block nicht angeschlossen. Danach kann jedoch der RAM als *Synopsys Database-File* (*db*) im Designpfad abgelegt werden. Nebenher ist noch eine Textdatei, ein *Synthetic-Library-File* (*sl*), zu editieren, in der die Ansteuerung (Pins, Ports, Binding der Ports, etc.) in einer Meta-Sprache für den *Synopsys Behavioral Compiler* definiert ist und zu einem *Synthetic-Library-Database-File* (*sldb*) kompiliert werden muß. Hier ist größte Sorgfalt zwingend erforderlich, da evtl. Fehler oder ungünstige Einstellungen für das Binding der RAM-Pins implizit und erst zu einem viel späteren Zeitpunkt (nach dem Scheduling) an Daten-Inkonsistenzen beim Speicherzugriff sichtbar werden[66].

- Für die verwendeten FPGA bietet *Xilinx* zwei unterschiedliche Designbibliotheken an: eine für den *Synopsys FPGA Compiler* und eine gleichen Inhalts für den *Synopsys Design Compiler.* Eigentlich müßte der *Behavioral Compiler* mit den *Design Compiler*-Bibliotheken betrieben und die Synthese nachher auch mit dem *Design Compiler* durchgeführt werden. Das führt trotz gleicher Bedingungen wie der vorgestellte Entwurfsfluß mit *Synopsys FPGA Compiler* und -Bibliotheken auf völlig inakzeptable Syntheseergebnisse (siehe Tab. 5.3). Nur die hier dargestellte Werkzeugkombination erwies sich als brauchbar.

Prinzipiell kann für den Entwurf der Operatoren zwischen einem Operator-Design mit Pipeline oder einem reinen Schaltnetz gewählt werden. Von einem Pipeline-Design wurde jedoch aus Gründen des zusätzlichen Gatterverbrauchs und der wachsenden Komplexität abgesehen. Damit sind alle Operatoren im Chip 2 „einfache“ Schaltnetze, deren längste Pfade im allgemeinen eine Taktperiode überschreiten und damit zu Multicycle-Netzen werden[67]. Alle eigenen arithmetischen Operatoren und Sonderfunktionen sind in einem speziellen VHDL-Package definiert (*xfloat*) und zusammengefaßt (ähnlich der C++ Prototypen).

[66] Siehe hierzu auch [Kurup und Abbasi 1995, Knapp 1996].

[67] Singlecycle-Operationen benötigen genau einen Takt, Multicycle-Operationen folglich mehrere Taktzyklen.

5.2 High-Level-Synthese der Nachregelschleifen

5.2.1 Ein synthesefähiges Verhaltensmodell der Nachregelschleifen

Sind im Verlauf der Vorarbeiten die notwendigen VHDL-Funktionen und Bibliotheken für das Synthesewerkzeug bereitgestellt, gestaltet sich der eigentliche VHDL-Kode für die Nachregelschleifen sehr übersichtlich und sieht einer C-Implementation ähnlich. Die Abbildung 5.6 enthält einen auf die wesentlichen Schritte reduzierten VHDL-Meta-Kode und läßt dessen Struktur erkennen. Nicht dargestellt ist die Deklaration der „Hülle“ (*entity*) samt Port- und Signaldeklarationen und Attribut-Vergaben.

```
architecture behavioral of chip2

        main: process
        reset_loop
            init_loop for 0 to 59                   -- Initialisierung der RAM Zellen aller 60
                RAM(address):= XF_Init_1i;          -- 60 Filterbankkanäle mit
                address:= address+1 ;               -- XF_Init_1i (5 x Nachregelschleifen)
                ....                                -- und
                RAM(address):= XF_Init_2;           -- XF_Init_2 (1 x Tiefpaß)
                address:= address+1 ;               -- initialisieren
            end loop ;
            work_loop
                ...                                 -- Handshake mit der Eingangsschnittstelle
                channel_loop for 0 to 59            -- 2 x 30 Frequenzkanäle (links/rechts)
                FK_Input<=In_Port;                  -- Eingangsabtastwert übernehmen
                XF_Input:=conv_xfloat(FK_Input);    -- Umwandlung Festkomma -> Fließkomma

                    stage_loop for 0 to 4           -- 5 Adaptationsschleifen
                        XF_Temp:= RAM(address) ;    -- Tiefpaßspeicher lesen
                        if stage_loop 0 then        -- Division (xfloat_div)
                            Q := XF_Input / XF_Temp ;
                            else
                            Q := Q / XF_Temp ;
                        RAM(address) := XF_Temp     -- Tiefpaßgleichung (optimiert)
                                - C1 * XF_Temp      -- (xfloat_add, xfloat_sub, xfloat_mul)
                                + C1 * Q;           -- und Tiefpaßspeicher schreiben
                        address:= address+1 ;       -- Speicher-Adresse inkrementieren
                    end loop ;

                FK_Temp:=conv_fixed(Q);             -- Umwandlung Fließkomma -> Festkomma
                TP_Sum:=C7 * (Q - C3) + C6 * TP_Sum; -- Skalierer/Tiefpaß (optimiert)
                Out_Port<=TP_Sum;                   -- Ergebnis ausgeben
                address:= address+1 ;               -- Speicher-Adresse inkrementieren
                end loop ;
                ...                                 -- Handshake mit der Ausgangsschnittstelle
            end loop ;
        end loop ;
        end process ;main
end behavioral ;
```

Abb. 5.6: Prinzipieller Aufbau des VHDL-Kodes für die Nachregelschleifen.

Der gesamte Algorithmus ist in einem einzigen Prozeß *main* untergebracht. Damit kann eine - wiederum von Hand - zu erstellende Steuerung der Interprozeßkommunikation zwischen mehreren Prozessen vermieden werden. Das Scheduling arbeitet immer nur auf einem Prozeß. Die äußerste Schleife ist die *reset_loop*, mit welcher der Prozeß startet und in die im Falle extern ausgelöster Resets zurückgesprungen wird. Der wichtigste Teil ist hierbei die Initialisierung der Variablen und RAM-Zellen mit den minimalen Schwellwerten für alle Divisoren des Systems in der *init_loop*. Hierbei werden die RAM-Zellen lediglich mit dem Schleifenzähler indexiert angesprochen und beschrieben. Der Schleifenzähler umfaßt die geforderten 2 x 30 = 60 Frequenzkanäle (stereo). Ist die *init_loop* abgelaufen, geht der Prozeß in die Arbeitsschleife *work_loop*. Die *work_loop* besteht aus zwei weiteren geschachtelten Schleifen. Die *channel_loop* umschließt die Abarbeitung der 30 Frequenzkanäle, in denen jeweils die *stage_loop* die Berechnung der fünf Nachregelschleifen steuert. Da die Eingangssignale für den Algorithmus sequentiell über den Eingangsport eintreffen, erfolgt die ebenso sequentielle Adressierung des Speichers, d. h. die o. g. Speicherinitialisierung entspricht genau der Belegung der Daten-Frames (siehe Kap. 4.1.4).

Wird der VHDL-Kode gemäß dem Entwurfsfluß aus (Kap. 5.1.2) angewandt, erhält man nach erfolgreichem Scheduling eine vorsynthetisierte Netzliste, die neben den ursprünglichen Operationen auch den aus Datenabhängigkeiten und Signallaufzeiten optimierten Steuerautomaten enthält. Da hierbei alle Wartezyklen (um das Ergebnis eines großen Schaltnetzes abzuwarten), in einzelnen Zuständen des Automaten gespeichert werden, ist das zugehörige Zustandsübergangsdiagramm umfangreich und viel weniger anschaulich, als die Darstellung der vom Automaten gesteuerten Schaltnetz(Operator)-Aktivität (Abb. 5.7). Diese Diagramme werden nach dem Scheduling vom *Synopsys Behavioral Compiler* generiert und mit dem *BC_View* Werkzeug dargestellt. Neben den in Abb. 5.7 gezeigten und für das Verständnis der Scheduling-Varianten wichtigsten Zusammenhänge können die Diagramme noch weitaus detaillierter abgefragt werden. Die Zeilen des Diagramms entsprechen immer der Anzahl von Taktzyklen, die zum Anordnen aller explizit im VHDL-Kode verlangten und implizit beim Scheduling erzeugten Ressourcen benötigt werden. Die Spalte zur grafischen Darstellung der Schleifenverschachtelung enthält (neben weiteren z. T. implizit entstehenden Einträgen durch *wait*-Anweisungen) die zeitliche Lage der im VHDL-Kode sichtbaren Schleifen. Dabei wird nur die Position und Länge einer iterativ durchlaufenen Schleife, nicht aber ihre Wiederholung dargestellt. Ein einzelner Durchlauf durch z. B. die *stage_loop* benötigt ebensoviele Takte, wie Zeilen zugeordnet sind. In den nachfolgenden Spalten wird die Aktivierung verschiedener Hardware-Ressourcen dargestellt. Dabei kann deutlich zwischen Singlecycle- und Multicycle-Operatoren unterschieden werden. Im Bereich der *stage_loop* gut wiederzufinden sind z. B. die Datenabhängigkeiten und die Reihenfolge der über mehrere Takte verlaufenden Multicycle-Netze der arithmetischen Fließkomma-Operatoren. Im wesentlichen ist die erkennbare Abfolge von Operatoraktivität, Speicherzugriff und Schleifenzugehörigkeit interessant, wobei anhand der Querverbindungen auch die Datenabhängigkeiten veranschaulicht werden. Weitere hier dargestellte Ressourcen, wie Register, Multiplexer und Zähler werden beim Scheduling implizit erzeugt und sollen nicht weiter betrachtet werden. In der Anordnung der Operatorschaltnetze spiegelt sich auch evtl. mögliches Ressourcen-Sharing wider. Im Falle der Festkomma-Multiplizierer

(*fix_MUL1* und *fix_MUL2*) werden beispielsweise zwei unabhängige Multiplizierer für den abschließenden Tiefpaß alloziert, obwohl jeweils gleichdimensionierte Operanden verwendet werden. Zugunsten eines geringeren Flächenbedarfs könnte auf Kosten eines weiteren Taktes ein Ressourcen-Sharing erzwungen werden. Durch die Anpassung des VHDL-Kodes oder durch das Setzen von *constraints* im Scheduling-Skript wäre diese manuelle Steuerung der Alllokation bzw. der Mehrfach- oder Exklusivnutzung von Operatoren möglich. Die Festkomma-Multiplikationen im Skalierer/Tiefpaß können allerdings zur Durchsatzsteigerung parallel verbleiben, da sie als spezielle *Xilinx*-Multiplizierer optimal sind.

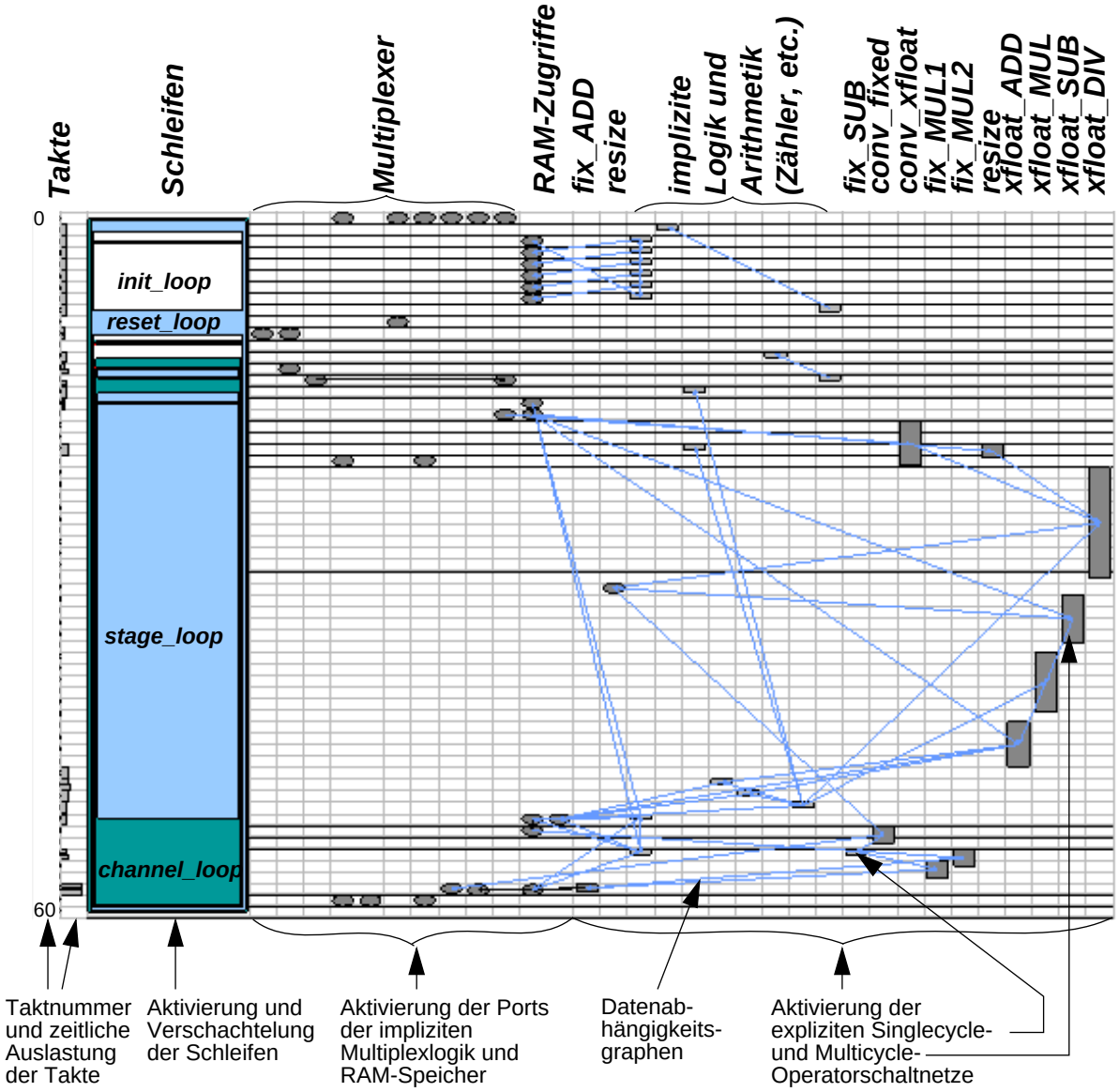


Abb. 5.7: Das Scheduling-Diagramm für die Nachregelschleifen mit Algorithmus-Variante B, Gl. (4.29), erzwingt eine nicht verschiebbare Reihenfolge der Operationen, so daß sich in der Summe der Signallaufzeiten durch alle vier großen Operatornetze der Hauptanteil des kritischen Pfades ergibt und damit die Gesamtperformanz des Designs an dieser Stelle entscheidend beeinflußt wird.

5.2.2 Die Operatorbibliothek

Die o. g. Operatoren werden als *functions* in einem VHDL-Package kodiert und umfassen die in der *xfloat*-C++ Klasse existierenden arithmetischen Funktionen. Nachfolgend soll die interne Verarbeitungsstruktur nur der vier arithmetischen Schaltnetze (*xfloat_ADD, xfloat_SUB, xfloat_MUL, xfloat_DIV*) näher beschrieben werden, da hier der Schwerpunkt des VHDL-Entwurfs für die Nachregelschleifen liegt. Dargestellt wird der für die C++ und die VHDL-Implementierung gleichermaßen geltende Datenfluß. Eine möglichst genaue Übertragung der C++ Variante wird unterstützt durch:

- das Typkonzept von VHDL; die Operanden werden als aus Mantisse und Exponent zusammengesetzter Daten-Record beschrieben.

- die Definition von (arithmetischen) Funktionen mit Operatorüberladung, die im Zielalgorithmus auf eine vergleichbare Struktur führt.

Andererseits existieren für die Umwandlung in synthetisierbaren VHDL-Kode auch hinderliche Faktoren, wie die notwendige feste Vorab-Dimensionierung der Operandenbreiten und Schaltnetze. Werden wie hier gleiche Funktionen (Operatoren) mit verschiedenen Wortbreiten benötigt, ist die generische Dimensionierung der Operanden (aus ein und demselben *package*) nicht möglich und erzwingt für jede Wortbreite fest vordimensionierte, separate VHDL-Packages.

Das verarbeitete Fließkomma-Format ist als *record* aus Mantisse und Exponent definiert:

```
type xflM_E is record
      m: xflM_E_mantissa;
          e: xflM_E_exponent;
end record;
```

Unter Verwendung von vorzeichenlosen Subtypen mit jeweils skalierbarer Breite der Mantisse *CiMan* bzw. des Exponenten *CiExp*:

```
subtype xflM_E_mantissa
      is unsigned(CiMan-1 downto 0);

subtype xflM_E_exponent
      is unsigned(CiExp-1 downto 0);
```

Der Dezimalpunkt befindet sich vor der ersten Stelle der Mantisse, dem für die normalisierte Zahl das erste gesetzte Stellenbit (Wert „1/2“) folgt. Einzig die Null besitzt als „Nullvektor“ diese „1“ nicht. Eine Definition mit impliziter (nichtgespeicherter) „Eins“ ist eher eine Behinderung und ein Nachteil bei der Entwicklung. Der Aufwand für die fortlaufende Sonderbehandlung dieser Stelle ist durch die mögliche Ersparnis bei Registerbreiten und Speichern nicht gerechtfertigt, da er deutlich unter 10 % liegt [68].

[68] 8,3 % für den (6,6)-Dividierers und 5 % bei der (14,6)-Addition, -Subtraktion und -Multiplikation.

Die nachfolgenden Schemata erläutern die interne Arbeitsweise der Operatoren, wobei die jeweils wichtigsten Schritte einbezogen sind. Der eingesetzte Rundungsalgorithmus beispielsweise ist dabei nicht im Detail berücksichtigt.

Addition

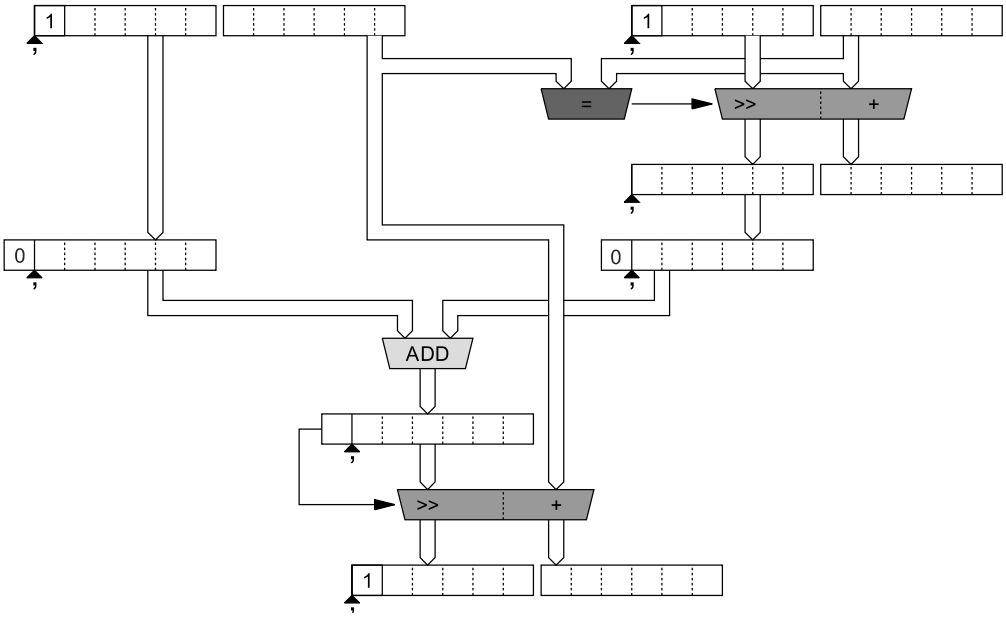


Abb. 5.8: Struktur des VHDL-Operators für Fließkomma-Addition mit generischer Präzision.

1. Die Operanden werden verglichen und entsprechend der Größe angeordnet. (der größere links in der Abb. 5.8). Dieser Schritt ist aus Gründen der Übersichtlichkeit im Verarbeitungsschema nicht enthalten.
2. Im Regelfall haben die Operanden nicht den gleichen Exponenten. Um aber die Addition durchführen zu können, müssen die Exponenten angeglichen werden. Das geschieht durch eine einmalige Anpassung der Mantisse mit vorberechneter Rechtsverschiebungen der Mantisse („>>“) des kleineren (rechten) Operanden und mit einer Angleichung des Exponenten („+“). Ist die Exponentendifferenz größer als die Mantissenbreite, verschwindet der behandelte Operand vollständig. Wie der Vergleich verschiedener Varianten zeigte, ist diese Implementierung (für ein Schaltnetz !) hinsichtlich der synthetisierten Strukturen wesentlich ökonomischer, als die bitweise iterative Anpassung von Mantisse und Exponent, weil dabei zusätzliche Hardware für den Vergleich und die Inkrementierung (Zähler) vermieden wird.
3. Nun kann die Mantisse addiert werden („*ADD*“), wobei ein möglicher Überlauf durch eine temporäre Erweiterung der Operanden um eine Vorkommastelle abgefangen wird. Diese Erweiterung wäre allerdings nur für die Summe erforderlich. Das Synthesewerkzeug wird die Addition mit einem über den „+“ Operator infe-

rierten Addierer synthetisieren. Dieses *DesignWare*-Bibliothekselement läßt aber nur gleiche Wortlängen in Summanden und Summe zu.

4. Im Falle eines Überlaufs wird nun die Mantisse mit einer Rechtsverschiebung normalisiert („>>“) und der Exponent entsprechend angepaßt („+“). Tritt bei dieser Addition ein Überlauf auf, wird die größte darstellbare Zahl ausgegeben.

Subtraktion

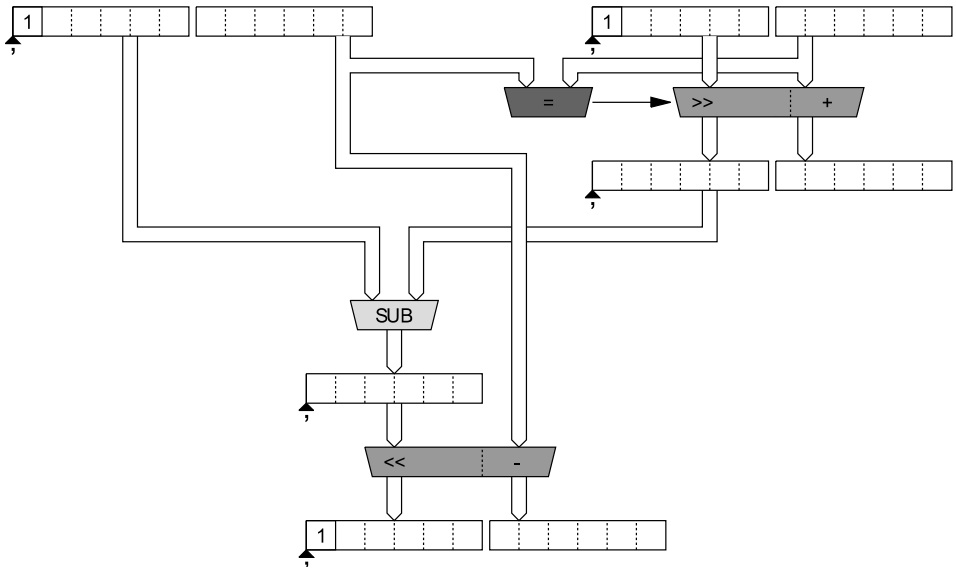


Abb. 5.9: Struktur des VHDL-Operators für Fließkomma-Subtraktion mit generischer Präzision.

1. Die Subtraktion wird als Pendant der Addition ganz ähnlich implementiert und beginnt mit der Anpassung von Mantisse und Exponenten des Subtrahenden. Wird die Bedingung $Minuend \geq Subtrahend$ nicht vorab erfüllt, (der initiale Vergleich ist in Abb. 5.9 nicht dargestellt), ist das Ergebnis „Null“ [69].
2. Ist die Exponentendifferenz wiederum größer als die Mantissenlänge, verschwindet der behandelte Subtrahend ganz und das Ergebnis ist gleich dem Minuenden.
3. Subtraktion der Operanden. Ein Übertrag in eine Vorkommastelle ist nicht möglich, womit die Operatorbreite unverändert bleiben kann.
4. Im Falle einer erfolgreichen Subtraktion ist das Ergebnis nachträglich zu normalisieren, d. h. durch eine über den notwendigen Stellenbereich erfolgende Linksverschiebung und entsprechende Exponentenverringerung zu korrigieren. Ergibt die Exponentendekrementierung einen Unterlauf, wird „Null“ als kleinstmögliches Resultat zurückgegeben.

[69] Im ursprünglichen Algorithmus der Nachregelschleifen treten weder Subtraktion noch negative Werte auf. Die Umformulierung der Tiefpaßfunktion nach Variante A und B in Kap. 4.3.2 mit der hier beschriebenen Fließkomma-Subtraktion kann durch die Beziehung der Tiefpaßkonstanten zueinander aber kein negatives Ergebnis hervorrufen.

Multiplikation

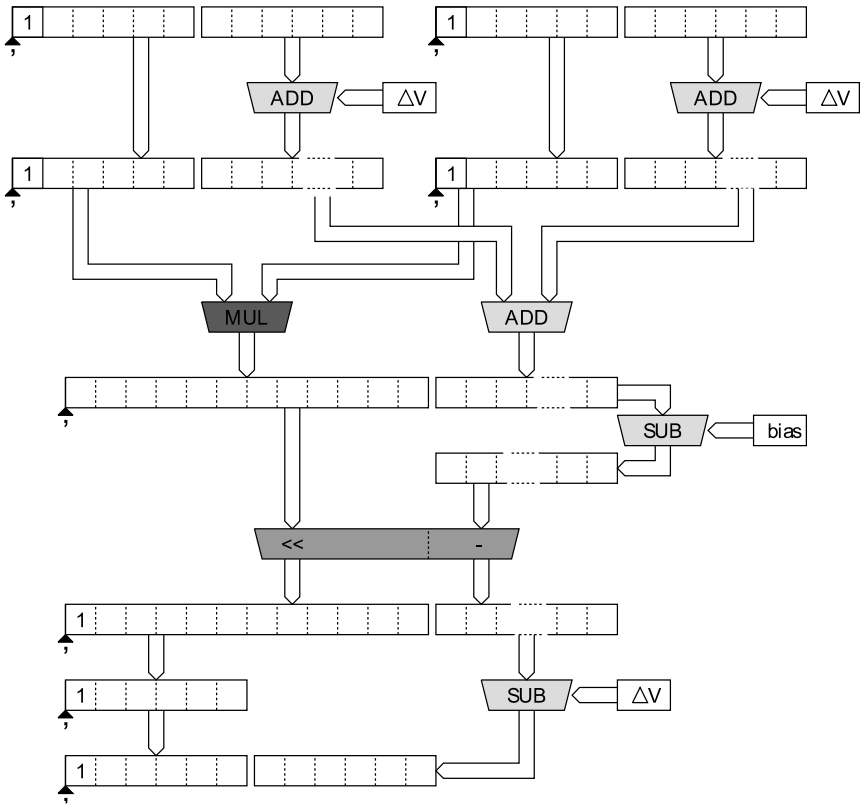


Abb. 5.10: Struktur des VHDL-Operators für Fließkomma-Multiplikation mit generischer Präzision.

1. Gegenüber der Addition und Subtraktion sind die Multiplikation und die Division im Fließkomma-Format auf den ersten Blick einfacher zu implementieren, denn es entfällt die vorangehende Anpassung der Exponenten. Die Multiplikation normalisierter Mantissen („*MUL*“) mit dem Dezimalpunkt vor der ersten signifikanten Stelle produziert eine doppelt genaue Ergebnismantisse. Die Exponenten werden addiert („*ADD*“). Da hierbei ein Überlauf auftreten kann, müssen die Exponenten erweitert und deswegen auch eine entsprechende Korrektur der Verschiebung vorgenommen werden („*ADD*“ Δ*V*) (siehe Abb. 5.10).
2. Bei der Exponentenaddition wird auch die Verschiebung (*bias*) addiert, was im nachfolgenden Schritt mit einer Subtraktion zu korrigieren ist („*SUB*“ *bias*).
3. Beim Normaliseren des Ergebnisses wird die notwendige Linksverschiebung („<<“) und eine Exponentenreduktion vorgenommen („-“).
4. Die Mantisse des Produktes wird durch Abschneiden und Runden auf die ursprüngliche Größe gebracht. Der Exponent des Produktes muß ebenfalls auf die gewünschte Länge gekürzt werden, wobei die Verschiebung zu korrigieren und mögliche Über- bzw. Unterläufe abgefangen werden müssen („*SUB*“ Δ*V*).

Division

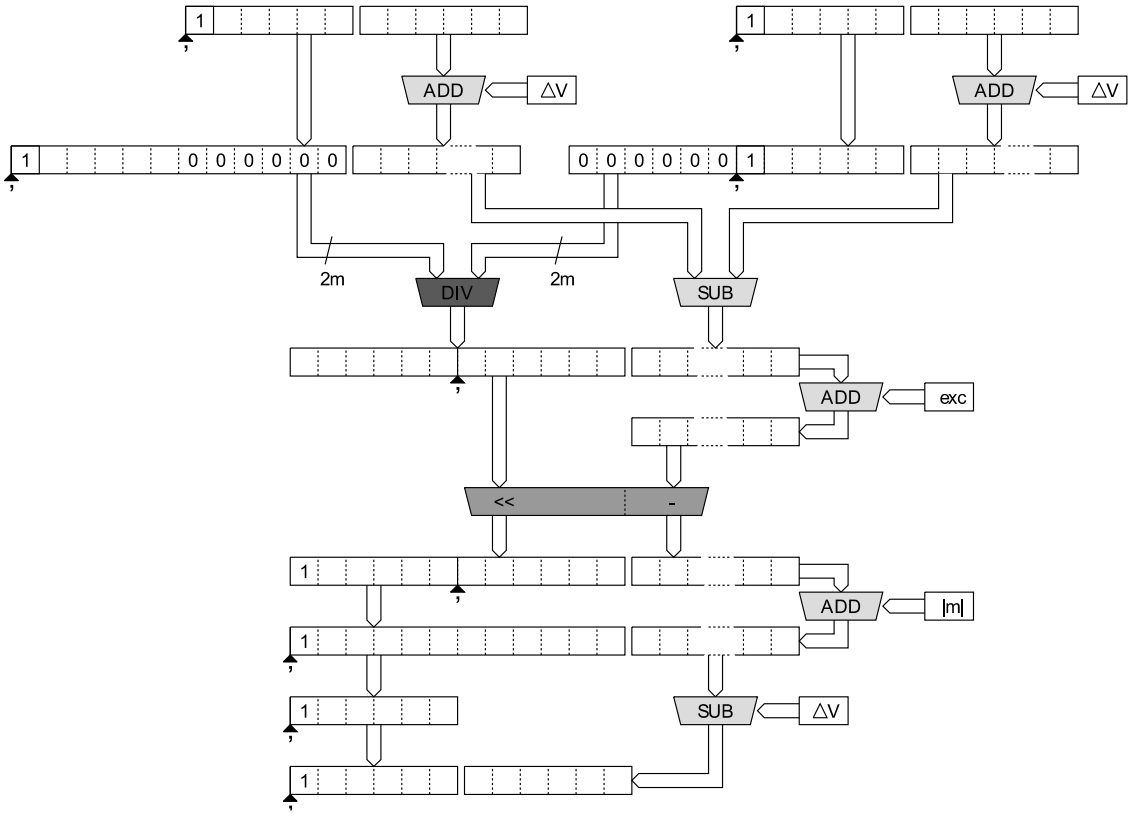


Abb. 5.11: Struktur der VHDL-Operators für Fließkomma-Division mit generischer Präzision.

In Abb. 5.11 wird die Arbeitsweise des Fließkomma-Dividierers veranschaulicht. Damit das Ergebnis der Festkomma-Division der Mantissen komplett abgebildet werden kann, werden die Operanden auf die doppelte Größe (*2m*) verbreitert und gegeneinander verschoben. Aus Gründen der Flexibilität für weiterführende Optimierungen können hier auf diese Weise auch nicht normalisierte Mantissen dividiert werden. Außer in der Testphase wurde für die implementierten Version des Algorithmus kein Gebrauch von dieser Möglichkeit gemacht, da die Division normalisierter Zahlen für die Gesamtperformanz des Designs zunächst ausreichend war.

1. Um die Exponenten dieser Änderung anpassen zu können, müssen auch sie erweitert und wiederum die Verschiebung („*ADD*“ ΔV) korrigiert werden.
2. Die Mantissen werden nun dividiert und die Exponenten voneinander subtrahiert. Da dabei auch die Verschiebung verschwindet, muß diese neu addiert werden („*ADD*“ *bias*). Die Korrektur des Kommas aus der Mitte der Mantisse samt anschließender Normalisierung erfordert die Addition der ursprünglichen Mantissenbreite $|m|$ auf den Exponenten.
3. Für die Wiederherstellung des eingangsseitigen Datenformates, wird die Mantisse abgeschnitten, der Exponent verkleinert und ein entsprechendes ΔV wieder subtrahiert („*SUB*“ ΔV). Falls es bei der Exponenten-Verkleinerung zu Über- oder Unterläufen kommt, werden die Maximalwerte für dieses Datenformat ausgegeben.

Die oben beschriebenen Operatoren wurden mit den anderen benötigten Funktionen in einer VHDL-Bibliothek (*package*) zusammengefaßt. Die Tabelle 5.1 enthält die Hauptbestandteile dieses *xfloat-packages*.

Name	return Typ	Parameter	Verwendung
"+"	xflM_E	-	Addition
"-"	xflM_E	-	Subtraktion
"*"	xflM_E	-	Multiplikation
"/"	xflM_E	-	Division
"="	xflM_E	-	Zuweisung
"<"	*boolean*	-	Relation
xfloat_norm	xflM_E	-	Normierung
xfloat_round	xflM_E	boolean (Runden ein/aus)	Runden (aktivierbare Unterfunktion)
conv_xfloat_fracM_E	xflM_E	-	Umwandeln Festkomma -> *xfloat*
conv_xfloatM_E	xflM_E	-	Umwandeln
conv_unsigned	unsigned	-	Umwandeln xfloat -> *unsigned*
conv_fixed	unsigned	integer, integer (Vorkomma, Nachkommaanteile	Umwandllung *xfloat* -> Festkomma
resize	std_logic_vector	integer, integer (Mantisse, Exponent)	Umwandeln xfloat -> xfloat (Umquantisieren)

Tab. 5.1: Hauptfunktionen des VHDL-Packages *xfloat* für skalierbare Fließkomma-Operationen.

Bei der Implementierung der o. g. arithmetischen Funktionen kommt es in der Regel zu einer Abfolge von sequentiellen Zuweisungen und Berechnungen sowie bedingten Entscheidungen (Bit-Abfragen etc.). Für eine korrekte Funktion und Synthese mit dem *Synopsys Behavioral Compiler* sind dabei weitere Regeln zu beachten.

- Werden Multicycle-Netze, wie z. B. die Fließkomma-Operatoren, nicht vor der Zerlegung durch das Scheduling des *Behavioral Compilers* geschützt, wird dieser versuchen, ein optimales Scheduling auch innerhalb der Operatoren zu finden. Das ist weder beabsichtigt noch zweckmäßig, denn damit weitet sich der Suchraum für die Optimierung stark aus. Die Multicycle-Netze müssen deswegen mit dem -- *synopsys preserved_function* Pragma im VHDL-Kode vor der Zerlegung geschützt werden. Damit werden sie als Block behandelt und sind auch als geschlossene Funktion in den Scheduling-Diagrammen sichtbar (siehe Abb. 5.7).

- Alle in einer Funktion verwendeten temporären Variablen sind zuerst mit Null bzw. Nullvektoren zu initialisieren. Keinesfalls besteht die Sicherheit, daß sich im Verlaufe der einzelnen Zuweisungen sinnvolle Werte „von selbst“ ergeben. Das Ergebnis der Synthese ist ein Schaltnetz; nicht initialisierte Variablen können als „offene Drähte“ bestehen bleiben und zu sehr schwer auffindbaren Rechenfehlern führen.

- Alle *if-then* Konstrukte müssen alternative *else* Zuweisungen besitzen, was in geschachtelten Abfragen schwierig zu überblicken, aber unumgänglich ist. Andernfalls können aus den o. g. Gründen auch hier bei der Variablenbelegung Pfade offenbleiben und unbestimmte Werte hervorrufen. Insbesondere gilt das für die *return* Anweisung einer Funktion.

- Prinzipiell ist es wünschenswert, daß Operatoren z. B. in der Wortbreite generisch sind. Ein *Behavioral Compiler*-Design mit einem eigenen VHDL-Package (keine synthetische *Synopsys DesignWare*-Struktur) erfordert jedoch festdimensionierte Operatoren, weil es beim *bc_time_design* zu einer „schnellen“ Synthese kommt, die generische Schaltnetze als *unbound* quittiert und abbricht. Für jede verwendete Wortbreite muß also ein VHDL-Package vorliegen. Mit einem separaten Shell-Skript mit Textersetzung werden deshalb aus dem generischen *xfloat*-Package jeweils fest dimensionierte VHDL-Packages herstellt (*xfloat6_6* für die Division und *xfloat14_6* für alle anderen Operationen). Von einer Generizität ist insbesondere in Konvertierungsroutinen (z. B. zum Umwandeln von/nach Festkomma *conv_xfloat*, *conv_fixed*) abzuraten, denn dabei kommt es leicht zu „offen“ liegenden Pins, da der fremde Typ nicht bekannt ist und alle Fälle abgedeckt werden müssen. „Offene“ Pins, die obendrein datenabhängig zu unterschiedlichen Problemen erst bei der Simulation führen, findet man nur mit dem testweisen Vergeben des -- *synopsys preserved_function* Pragmas, das dann eine *unbound* Meldung hervorruft.

- Die *Xilinx DesignWare*-Bibliotheksfunktionen (XDW) sind u. U. in einem separaten Entwurfsfluß vorab in ein Netzlistendesign zu überführen (*compile*). Diese Bibliothek enthält die hinsichtlich der FPGA-Architektur optimierten arithmetischen Operatoren und sollte aufgrund von Performanz-Vorteilen auch Verwendung finden.[70]

- Die Simulation eines Designs mit *Xilinx Virtex*-Technologie (wie bei allen anderen auf Look-Up-Tables beruhenden Technologien) wird von *Synopsys* auf Gatter-Ebene nicht unterstützt. Eine simulierbare Netzliste muß durch Synthese-Skripte mit Sonderbehandlung aller verwendeten --*synopsys preserved_functions* manuell hergestellt werden.[71]

[70] Leider gelingt dem *Synopsys Behavioral Compiler* die Synthese der Binary-Tree-Multiplizierer aus der *Xilinx DesignWare*-Bibliothek im normalen Entwurfsfluß nicht, so daß ein Synthese-Skript die benötigten Operatoren in der gewünschten Dimension separat vorab synthetisieren muß. Später muß man die fehlerhaften Blöcke entfernen und die vorkompilierten Operatoren in das Top-Level-Design einbauen (siehe auch Anhang A).

5.2.3 Optimiertes automatisches Scheduling

Die oben dargelegten Erfahrungen führten nach vielen Designiterationen aber erst dann zu einem stabilen Ergebnis mit automatischem Scheduling, nachdem die passenden Scheduling-Optionen gefunden waren. Flächenbedarf, maximale Taktrate und Gesamtverzögerung zur Bearbeitung eines Abtastwertes sind dabei die drei wichtigsten Optimierungsziele:

- ***Fläche***

  Hohe Anforderungen werden an den Flächenverbrauch des Chip 2-Entwurfs gestellt. Die Logik- und RAM-Ressourcen eines *Xilinx Virtex*-FPGA müssen für die Aufnahme des gesamten Perzeptionsmodells neben dem Chip 2 auch für die Gammatone-Filterbank (Chip 1) und die Schnittstellen zum Host-System ausreichen. Die Schritte Plazieren und Verdrahten für das gesamte Design haben nur eine Chance, wenn außerdem eine Flächenreserve als Spielraum verbleibt.

- ***Taktrate und Gesamtverzögerung***

  Die maximale Taktrate wird primär von der Technologie bestimmt und ist erst nach dem *place&route*, also der Layout-Generierung, bekannt. Das Scheduling erzeugt theoretisch zu jeder beliebigen Taktrate immer ein gültiges Design. Entsprechend den Gatterlaufzeiten werden in den Schaltungspfaden lediglich mehr oder weniger viele Takte eingeplant. Ob jedoch dieses Design mit dem festgelegten Takt überhaupt operabel ist, entscheidet der letzte Schritt, das *place&route* (was obendrein nach sehr langer Laufzeit des Routers u. U. erst nach Stunden abschließt). Dadurch ist es nicht möglich vorherzusagen, ob ein erfolgreiches Scheduling zu einer ausreichend schnellen Schaltung führt. Wird der geforderte Takt nach dem *place&route* nicht erreicht, oder ergeben sich höhere Verzögerungen als vor dem Scheduling geschätzt, muß die Schaltung entsprechend langsamer getaktet werden. Das geht aber nur, wenn die dadurch entstehende Gesamtverzögerung für die Bearbeitung eines Abtastwertes noch zulässig ist. Auf diese Weise sind die maximale Taktrate und die Gesamtverzögerung eng miteinander verknüpft und empirisch über mehrere Versuche zu bestimmen. Vorteilhaft für das Scheduling wirkt sich zunächst die Auswahl einer etwas kleineren als der minimal notwendigen Taktperiode aus, wenn dadurch keine signifikant höhere Anzahl von Wartetakten entsteht. Dadurch kann man das Design nach dem *place&route* mit der dann möglichen (i. a. geringeren) Taktrate betreiben, ohne die zulässige Gesamtverzögerung zu überschreiten.

[71] Die beim *compile* belegten Look-Up-Tables sind nicht als VHDL-Beschreibung exportierbar und können damit nicht in einer Testbench simuliert werden. Alle *--synopsys preserved_functions* werden jedoch bereits beim *bc_time_desgin* für die Gatter-Ebene kompiliert und sind damit für die Simulation auf Register-Transfer-Ebene unbrauchbar. Hier mußte mit der gleichen Technik des blockweisen Rückersetzens der *preserved_functions* aus der *Elaborate*-Stufe gearbeitet werden, um überhaupt eine simulierbare Netzlistenbeschreibung auf Register-Transfer-Ebene nach dem Scheduling zu erhalten (siehe Anhang A).

Die manuelle Steuerung der Optimierung des Designs ist an mehreren Stellen möglich und nötig. Für das Scheduling ist der *schedule mode* der ausschlaggebende Parameter. Der *Synopsys Behavioral Compiler* bietet drei Scheduling-Modi an:

- ***cycle-fixed***

  Hier muß manuell dafür gesorgt werden, daß die Operatorenabfolge der Datenabhängigkeit entspricht und daß einzelne Operationen zu den gewünschten Zeitpunkten gültige Ergebnisse liefern. Tatsächlich wird auch hier ein manuelles Scheduling im voraus verlangt, nur die Umsetzung der feststehenden Operationsfolge in einen Steuerautomaten übernimmt der *Behavioral Compiler*. Dieser Modus ist aufgrund der eingeschränkten Freiheitsgrade aber auch der sicherste. Multicycle-Netze sind in diesem Modus nicht verwendbar.

- ***superstate-I/O***

  Hier werden Operationen zwischen im- und expliziten Takt-Flanken frei verschiebbar; das gilt aber nur in den zeitlich dehnbaren Grenzen eines *superstate*. Aufgrund der Beweglichkeit der Operationen in den *superstates* muß verstärkt auf korrektes I/O-Verhalten bei der Kommunikation mit den Ein- und Ausgangsports und bei Speicherzugriffen geachtet werden. Dieser Modus ist voll kompatibel mit Multicycle-Netzen und wird für den Entwurfsfluß von Chip 2 ausgewählt.

- ***free floating***

  Dieser Modus bietet dem Scheduling die größten Freiheiten, ist aber aufgrund der vollen Beweglichkeit aller Operationen nur für wenige Anwendungen geeignet.

Als einflußreiche *Behavioral Compiler*-Optionen für die Gesamtperformanz des Designs haben sich die Parameter erwiesen, welche die Erstellung und Dimensionierung der FSM[72] des Steuerautomaten bestimmen. Im folgenden sollen einige Parameter erläutert werden.

- ***Taktperiode***

  Die Wahl der Taktperiode ist ein mehrdimensionales und nur durch viele Versuche lösbares Optimierungsproblem. Der *Behavioral Compiler* muß für die Multicycle-Netze eine Anzahl Takte einplanen, nach welcher stabile Ergebnisse an den Ausgängen anliegen. Da die Abfolge der Operationen z. B. in der *stage_loop* bekannt ist und unter vergleichbaren Bedingungen auch auf prinzipiell gleiche Scheduling-Ergebnisse führt, kann im Vorfeld der dabei entstehende Verschnitt manuell optimiert und eine günstige Taktrate gewählt werden (siehe Tab. 5.2 anhand des Scheduling nach Algorithmus-Variante A, Abb. 5.12). Nach dem Scheduling ist die Takt-Vergabe je Operation bekannt und bestimmt damit zusammen mit der Anzahl der Schleifendurchläufe entscheidend die Gesamtverzögerung (zur Bearbeitung eines Abtastwertes). Zu dieser Verzögerung kom-

[72] FSM - *Finite State Machine*.

men noch die im VHDL-Kode explizit verlangten Wartetakte und weitere den Datenabhängigkeiten entsprechenden Singlecycle-Operationen (Input/Output, Inkrementierung von Zählern, Speicherzugriffe, etc.) hinzu, aber es entsteht kein zusätzlicher Verschnitt.

Taktperiode [ns]	Kritischer Pfad durch die Multicycle-Netze eingeplante Anzahl Takte je Op bei Scheduling nach Abb. 5.12						Σ-Takte in der kritischen Operator-Kette	Σ-Verschnitt in der kritischen Operator-Kette [ns]
	conv_xfloat 92 ns	xfloat_div 223 ns	xfloat_mul 124 ns	xfloat_add 94 ns	resize 27 ns	conv_fixed 31 ns		
24	4 Takte	10 Takte	6 Takte	4 Takte	2 Takte	2 Takte	28	81
25	4 Takte	9 Takte	5 Takte	4 Takte	2 Takte	2 Takte	26	59
31	3 Takte	8 Takte	4 Takte	4 Takte	1 Takt	1 Takt	21	60

Tab. 5.2: Der Verschnitt durch die maßgeblich an der Gesamtverzögerung beteiligten Multicycle-Netze der *stage_loop* für einen Durchlauf. Zusätzlich werden die für einen Frequenzkanal nur einmal benötigten Konvertierungsroutinen (*conv_xfloat, conv_fixed*) berücksichtigt, da sie mit einem deutlichen Einzelanteil zur Gesamtverzögerung beitragen. Das Minimum liegt bei 25 ns Taktperiode.

- ***dont_unroll Attribut***

  Dieses Attribut hat die größten Auswirkungen auf den Flächenbedarf der Schaltung, da hiermit die Parallelisierung von Hardware gesteuert werden kann. Schleifen im VHDL-Kode können mit dem *Behavioral Compiler* gerollt gehalten werden, d. h. Hardware-Ressourcen werden im Zeit-Multiplex-Betrieb im Takt der Schleifendurchläufe genutzt. Rollt man eine solche Schleife aus, wird die benötigte Hardware entsprechend der Anzahl der Schleifendurchläufe häufig synthetisiert und liegt damit parallel vor.

  Beispielsweise könnte man die Schleife über die 30 Frequenzkanäle ausrollen. Das Ausrollen wird bei kleinen Schleifenzählern (z. B. fünf in der *stage_loop*) empfohlen, führt aber in diesem Fall zu ungenügenden Ergebnissen (erhöhter Hardware-Aufwand, großer und langsamer Steuerautomat, siehe Tab. 5.3). Wird die Hardware wie im Chip 2 jedoch sowieso sequentiell genutzt, gegeben durch die aufeinander aufbauende Berechnung in den Nachregelschleifen, wird diese Datenabhängigkeit durchaus vom *Behavioral Compiler* erkannt und die Hardware auch trotz ausgerollter Schleife nur einfach alloziert (siehe Abb. 5.13).

- ***bc_set_margin***

  Dieser Parameter verlangt eine zusätzliche Zeit-Reserve (*timing-margin*) in [ns] bei der Zuteilung von Taktzyklen und erzeugt damit einen zusätzlichen beabsichtigten Verschnitt. Nötig kann das in allen Fällen sein, in denen zum Zeitpunkt des Scheduling die wirkliche Verzögerung auf Pfaden noch nicht bekannt ist oder unterschätzt wird (sichtbar nach *compile*, insbesondere aber nach *place&route*). Eine höhere *margin* kann gegebenenfalls die Umwandlung einzelner Operationen in Multicycle-Operationen und damit das Einfügen von zusätzlichen Zuständen im Steuerautomaten bewirken.

- ***fsm_style***

  Hiermit wird der Kodierungs-Stil des Steuerautomaten festgelegt. Zur Auswahl stehen *one_hot*, *two_hot* und *binary*. *One_hot* verlangt keine Dekodierung, aber einen sehr großen Zustandsvektor (für jeden Zustand ein Bit). *Binary* verwendet dagegen den minimalen Zustandsvektor der Länge n $(2^n \geq Zustandszahl)$ benötigt aber die Dekodierungslogik. Die *two_hot*-Kodierung führt für die relativ hohe Anzahl von Zuständen der FSM im Chip 2 (zwischen 65 und 174) als Zwischenlösung auf die besten Resultate im Zeitverhalten.

- ***register_control***

  Die FSM-Eingänge und/oder Ausgänge können Buffer-Register erhalten. Diese verlangsamen durch eine Taktverzögerung zwar die FSM und damit das Design, begrenzen aber auch lange (hochbelastete) Steuerpfade, was damit insgesamt zu einer Verkürzung des kritischen Pfades und zu einer Beschleunigung des Designs führt. Die Pufferung der FSM-Ausgänge lieferte die besten Ergebnisse hinsichtlich der maximal möglichen Taktrate.

Im Anhang A kann die Verwendung der erläuterten *constraints* in dem vollständigen Skript für das Scheduling nachvollzogen werden.

Wie schon Abb. 5.7 erkennen läßt, bestimmt die Anordnung der Operationen und mithin die Summenverzögerung durch die fünffache Iteration der inneren Schleife (*stage_loop*) maßgeblich den Durchsatz des gesamten Designs. Vier Architekturen, entstanden durch Variation der Schleifenstrukturen, wurden untersucht, wobei die Durchlaufzeiten der *stage_loop* relativ zueinander verglichen werden können. Alle nachfolgend dargestellten Architektur- bzw. Scheduling-Varianten wurden im *superstate-I/O* Scheduling-Modus behandelt, die konkreten Taktzuteilungen und Zeitangaben enthält Tab. 5.3.

Optimierte Tiefpaßgleichung Variante A (*stage_loop* nicht ausgerollt)

Das Scheduling für die Nachregelschleifen mit der Variante A der Tiefpaßgleichung, Gl. (4.28), führt ohne das Setzen zusätzlicher *constraints* auf eine Operationsfolge nach Abb. 5.12. Die Schleifen bleiben eingerollt, d. h. das Scheduling hat im Automaten sowohl den

fünffachen Durchlauf der *stage_loop* als auch die bestehenden Datenabhängigkeiten berücksichtigt und dabei die Operationen zeitlich zueinander verschieben und gegebenenfalls „zusammenrücken“ können.

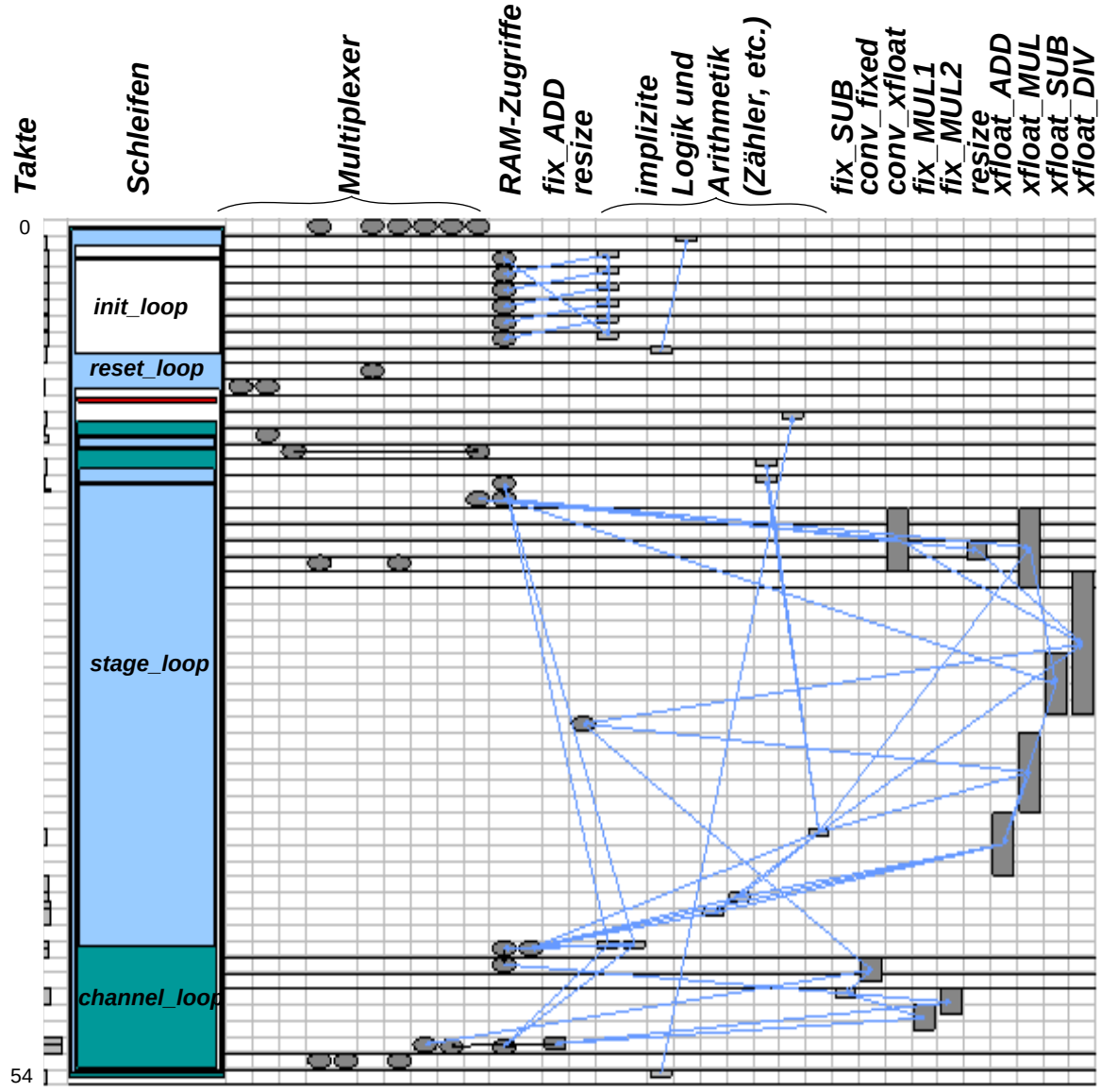


Abb. 5.12: Das Scheduling der Nachregelschleifen mit der Tiefpaßgleichung A bei 25 ns Taktperiode. Die *stage_loop* wird iteriert, eine Multiplikation und die Subtraktion können bereits zeitüberlappend mit der Division berechnet werden, während die zweite Multiplikation und die Addition vom Ergebnis der Division abhängig sind.

So kann z. B. eine Fließkomma-Multiplikation und die -Subtraktion bereits vor und während der Division stattfinden, obwohl die Sequenz im VHDL-Kode allein mit der Division beginnt. Da mit dem gleichen Operator multipliziert wird, können beide Multiplikationen zeitversetzt mit der gleichen Hardware arbeiten (Ressourcen-Sharing). Aus der Tab. 5.3 wird ersichtlich, daß trotz der parallelen Operatornutzung diese Variante eine zu große Gesamtverzögerungszeit besitzt und damit nicht brauchbar ist.

Optimierte Tiefpaßgleichung Variante A (*stage_loop* ausgerollt)

Wird die *stage_loop* ausgerollt, verändert sich das Scheduling nach Abb. 5.13. Die Sequenz im Bereich einer Schleife (wie in Abb. 5.12) bleibt erhalten, der Automat ordnet aber jetzt fünf dieser Sequenzen in Reihe an. Nach [Knapp 1996] sollten Schleifen mit einer ähnlich geringen Anzahl von Iterationen generell ausgerollt werden. Für den vorliegenden Fall kann das jedoch nicht empfohlen werden.

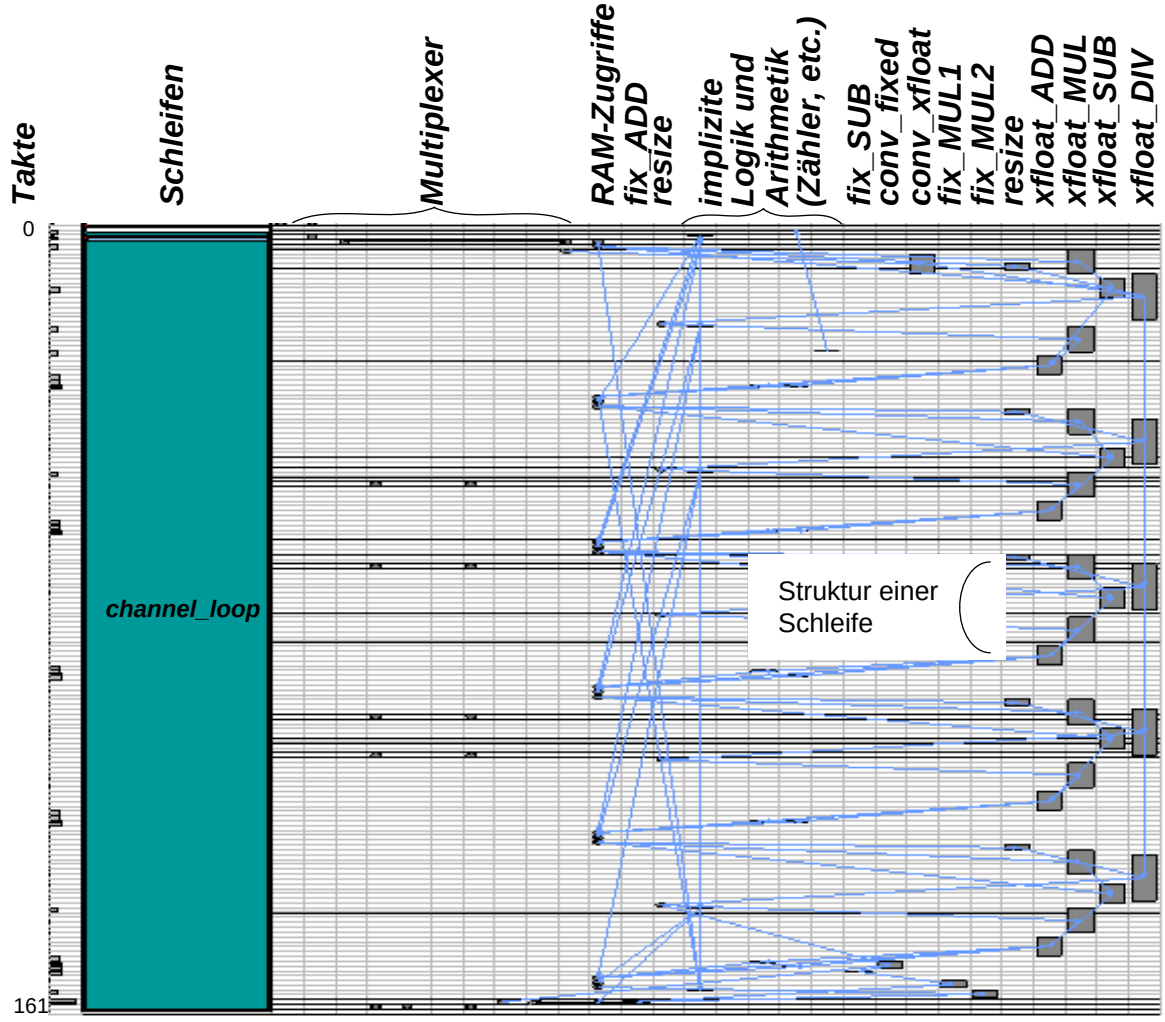


Abb. 5.13: Das Scheduling der Nachregelschleifen mit der Tiefpaßgleichung Variante A bei 25 ns Taktperiode. Die *stage_loop* wird ausgerollt und damit aus dem Design entfernt. Der Ablauf innerhalb der ehemaligen *stage_loop* entspricht dem in Abb. 5.12 dargestellten Scheduling. Auch hier kann eine Multiplikation und die Subtraktion zeitüberlappend mit der Division berechnet werden.

Einerseits benötigt das Scheduling für diese Variante überproportional viel Rechenzeit, weil sich durch das Ausrollen der Suchraum für ein optimales Scheduling entsprechend vergrößert. Andererseits sind die Ergebnisse keineswegs besser als die eingerollte Variante, d. h. es ergibt sich zuviel Verschnitt beim Anschluß der Einzelsequenzen, so daß die Gesamtdurchlaufzeit höher ausfällt.

Optimierte Tiefpaßgleichung Variante B

Nach der Umformung des Algorithmus (Ausklammern) zeichnet sich die Tiefpaßgleichung Variante B durch eine eingesparte Multiplikation aus. Aus dieser Einsparung zieht das Scheduling ebenfalls keinerlei Nutzen bezüglich der Gesamtdurchlaufzeit. Die Sequenz der Operationen ist durch die Datenabhängigkeiten nicht mehr veränderbar. Die Operationen einer Schleife können nicht zueinander verschoben werden und die Durchlaufzeit erhöht sich weiter.

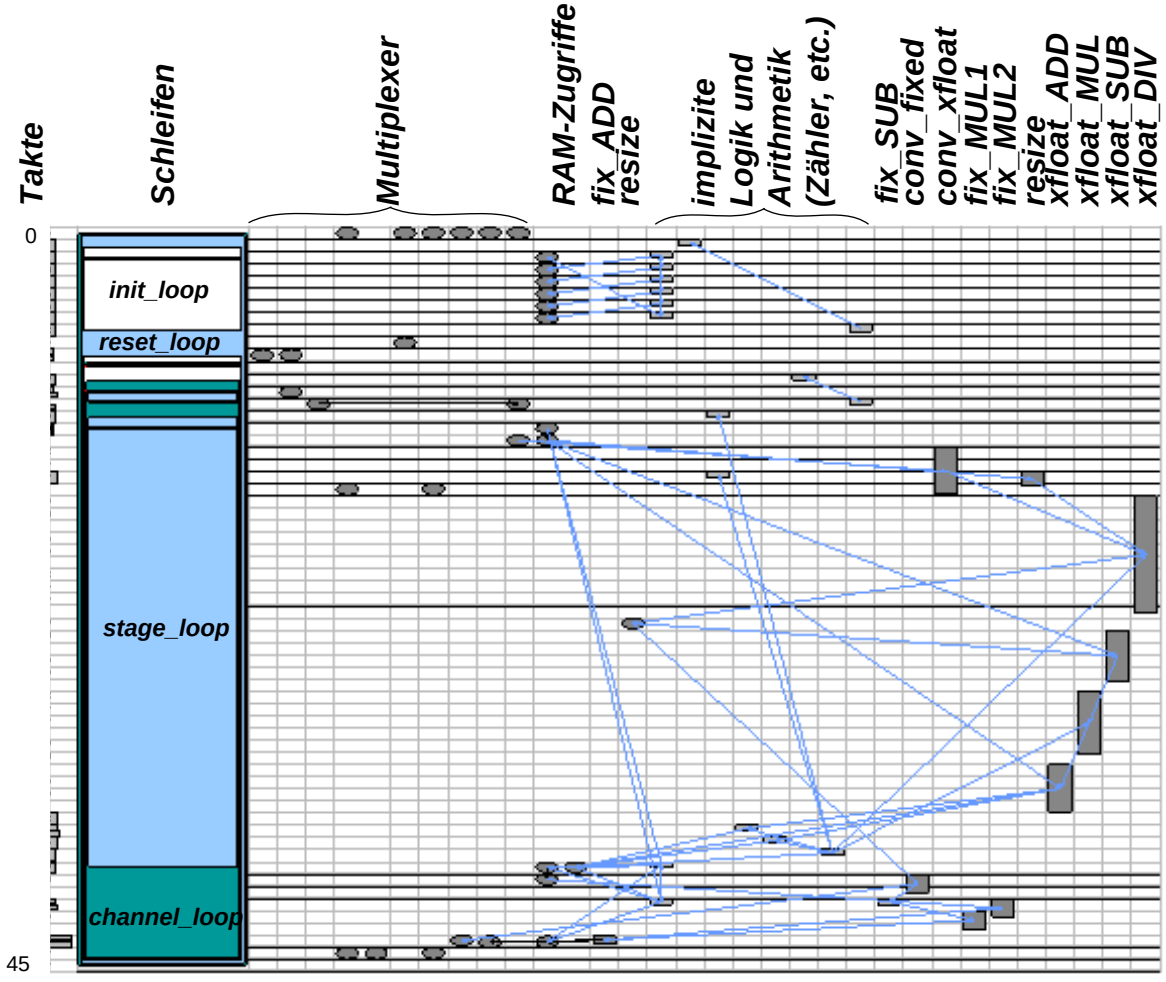


Abb. 5.14: Das Scheduling für fünf *stage_loops* bei 24 ns Taktperiode. Die im Algorithmus mit der Tiefpaßgleichung Variante B vorgegebene Operatorsequenz führt zur unerwünschten Erhöhung der Schleifendurchlaufzeit.

Hierbei werden also im Vergleich zur Variante A weder Ressourcen gespart, noch ist das Design schneller. Dieses Ergebnis ist insofern bemerkenswert, da die Umformung des Algorithmus' zunächst als optimale algebraische Lösung empfunden werden kann und ein genau gegenteiliger Effekt nicht erwartet wird. Diese Algorithmusvariante ist somit zu verwerfen.

Beschleunigung des Designs durch eine erweiterte Schleifeniteration

Alle drei vorigen Varianten haben leider eine Eigenschaft gemeinsam: die Berechnungsdauer für einen Abtastwert aus der Gammatone-Filterbank ist zu hoch; in 4,096 µs können die Berechnungen mit der *Xilinx Virtex*-Technologie nicht bewältigt werden. Das Scheduling konnte dabei die Datenabhängigkeiten, welche die ungünstige Aufsummierung der großen Verzögerungen der arithmetischen Operationen hervorrufen, nicht umgehen. Sowohl mit einer angepaßten Notation des Algorithmus' durch Einfügung von *wait*-Anweisungen, besonders aber mit einer Modifikation der Schleifenorganisation konnte ein Ausweg durch folgende Erweiterung gefunden werden:

Im VHDL-Kode wurde ein zusätzlicher sechster Schleifendurchlauf vorgesehen, der die Division um eine Iteration vorwegnimmt (Abb. 5.15).

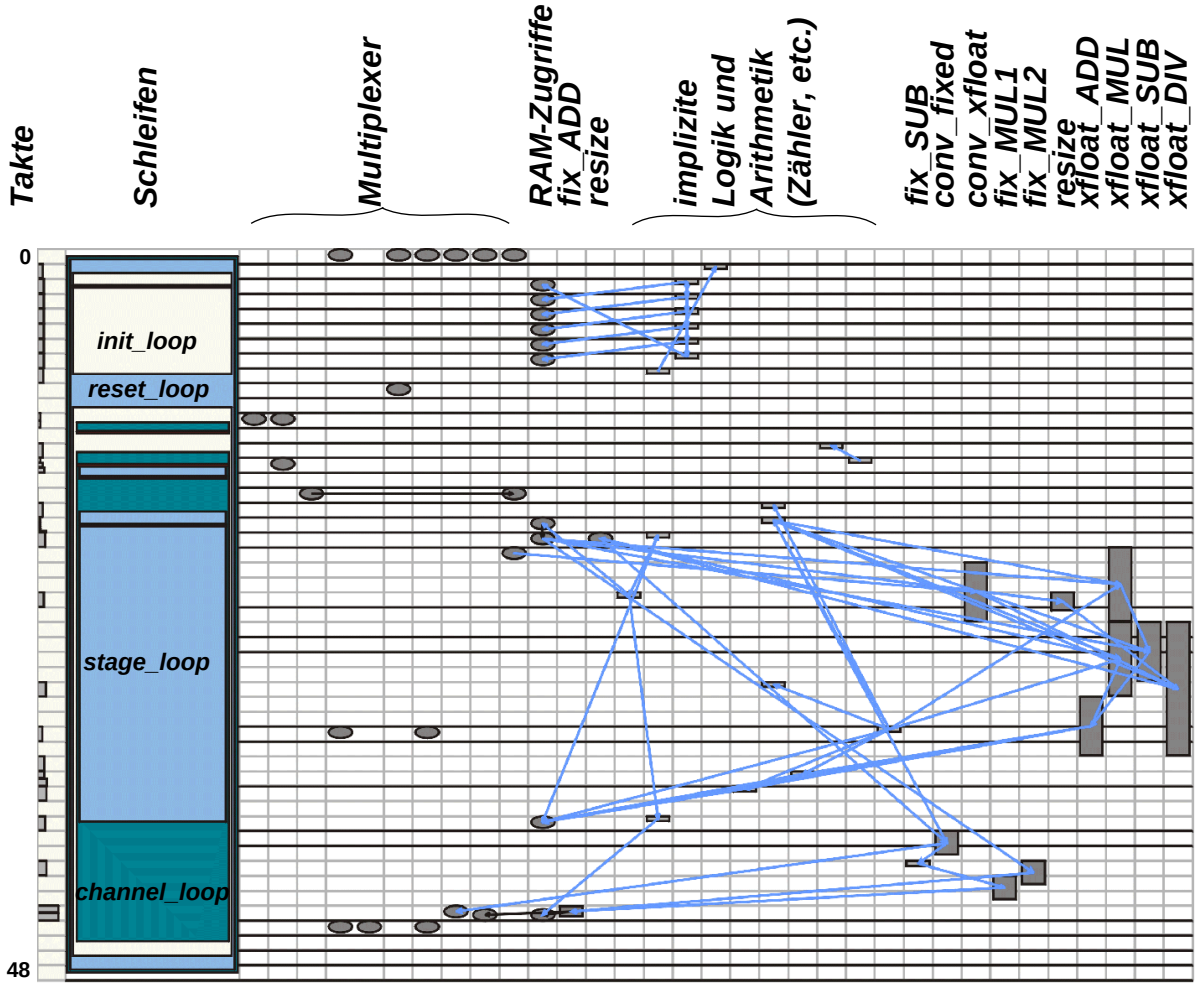


Abb. 5.15: Das Scheduling für sechs *stage_loops* bei 25 ns Taktperiode. Trotz des zusätzlichen Durchlaufs ist durch die Parallelisierung von Division und Schleifen-Tiefpaß die Gesamtverzögerung in der *stage_loop* durch Entkopplung der Operationen kleiner.

Im ersten Durchlauf wird nur dividiert und parallel findet eine Dummy-Tiefpaßberechnung statt. In den nächsten vier Durchläufen wird - wie beschrieben - dividiert und gefiltert, während im sechsten Durchlauf die Division eine Dummy-Operation ist, aber die letzte gültige Tiefpaßberechnung durchgeführt wird. Zunächst scheint das die Durchlaufzeit eher weiter zu erhöhen. Allerdings ist nun die Division vom Tiefpaß entkoppelt: die Datenabhängigkeiten innerhalb einer Schleife sind nicht mehr gegeben. Dem Scheduling ist es nun möglich, die zugehörigen Operatorschaltnetze parallel zu betreiben, da die Ergebnisse ja erst im nächsten Durchlauf benötigt werden. Die Wertübergabe zwischen den Iterationen erfolgt im VHDL-Kode über explizite Speichervariablen, die nach dem Scheduling (wie auch die vom Scheduling generierten Variablen) als Registerbänke im Design enthalten sind.

> Das entscheidende Ergebnis dieser Optimierung ist also die zum Tiefpaß insgesamt parallel ausführbare Division, die als längstes Multicycle-Netz die größte Einzelverzögerung verursacht.

Die Tabelle 5.3 stellt eine Reihe wesentlicher Ergebnisse mit den genannten Algorithmusvarianten bei den verschiedenen Scheduling-Parametern zusammen. Allen Versionen gemeinsam (außer der letzte Zeile der Tabelle) ist die Verwendung der *Synopsys FPGA Compiler* Bibliotheken für *Xilinx Virtex*-Bausteine. Entscheidend für die Bewältigung der Abtastrate ist die Summe der Takte für eine *channel_loop* multipliziert mit der Taktperiode. Die Angaben zur Verzögerungszeit nach dem *compile*-Schritt des Synthesewerkzeuges (logische Optimierung und Technologie-Mapping) sind eine meist sehr konservative Schätzung auf der Basis von Lastmodellen der verwendeten Gatter und Register. Die Schätzung der Verzögerungszeit des Layouts berechnet der *Xilinx Timing-Analyzer* und wertet dazu die tatsächlich benötigten Leitungslängen und Gatterlasten aus. Aus dieser Verzögerungszeit und der Taktanzahl für eine *channel_loop* berechnet sich die maßgebliche Verzögerung bei der Verarbeitung eines Abtastwertes. Ein klares Merkmal für die Leistungsfähigkeit des Steuerautomaten ist die Anzahl der vom Scheduling vorgesehenen Zustände. Die in allen Fällen recht hoch erscheinende Zustandszahl macht sich vor allem in einem langsamen Zeitverhalten des (räumlich ausgedehnten) Steuerautomaten bemerkbar. Bereits die Dimension des dafür notwendigen Zustandsvektors hat einen starken Einfluß. So ist die ausgerollte Version (vgl. Abb. 5.13) auch deshalb unakzeptabel langsam, weil sie einen doppelt so großen Zustandsvektor benötigt.

Taktperiode [ns]	FSM Zustandsvektor	FSM Register	Anzahl der Takte je Abtastwert (channel_loop)	FSM Zustandszahl	Max-Pfad nach compile [ns]	Max-Pfad nach place&route [ns]	Verzögerung des Layouts je Abtastwert [µs]
5 Schleifen, Tiefpaß Variante A							
25	two_hot	output	5*30+1+11=162	86	51,5	26,5 (37,7 MHz)	4,293
26	two_hot	output	5*30+1+10=161	74	47	26,2 (38,2 MHz)	4,218
31	two_hot	output	5*26+1+8=139	74	43	31 (32,2 MHz)	4,309

Takt periode [ns]	FSM Zustands-vektor	FSM Register	Anzahl der Takte je Abtastwert (channel_loop)	FSM Zustands-zahl	Max-Pfad nach compile [ns]	Max-Pfad nach place&route [ns]	Verzögerung des Layouts je Abtastwert [µs]
5 Schleifen, ausgerollt, Tiefpaß Variante A:							
24	two_hot	output	160 +1 =161	174	65	34,1 (29,3 MHz)	5,524
6 Schleifen (Operator-Pipelining), gerollt, Tiefpaß Variante A							
20	two_hot	output	6*26+1+11=168	78	49	26,7 (37,4 MHz)	4,49
23	two_hot	output	6*24+1+11=156	74	51	26,44 (37,8 MHz)	4,195
24	two_hot	output	6*23+1+11=150	72	46,8	27,1 (36,9 MHz)	4,293 mit 28 ns Takt
25	two_hot	output	6*21+1+11=138	68	49	24,91 (40,1 MHz)	3,534
		output	6*24+1+11=156 bei *margin*=9 ns	74	42	25,8 38,8 MHz	4,028
		input	6*21+1+11=138	69	48,2	25,5 ns 39,2 MHz	3,67
		input/ output	6*27+2+12=176	83	51,4	25,5 ns 39,2 MHz	4,523
		keine	6*19+1+11=126	65	61,8	31 32,2 MHz	4,008
	one_hot	output	6*21+1+11=138	68	39,7	25,9 (38,6 MHz)	3,674
	binary	output	6*21+1+11=138	68	39,3	26.1 (38,3 MHz)	3,689
		input	6*21+1+11=138	69	45,2	28,3 ns 35,3 MHz	4,014
		input/ output	6*27+2+12=176	83	34,3	25,6 39 MHz	4,505
		/	6*19+1+11=126	65	65,3	31,8 31,4 MHz	4.496
26	two_hot	output	6*21+1+11=138	68	46	25,7 (38,9 MHz)	3,634
			6*22+1+11=144 bei *margin*=9 ns	70	41	25,99 38,4 MHz	3,833
5 Schleifen, gerollt, Tiefpaß Variante A, mit *Synopsys Design Compiler*-Bibliotheken							
24	two_hot	output	5*47+1+15=251	121	38,3	24,2 (41,3 MHz)	6,074

Tab. 5.3: Die jeweils besten Ergebnisse für das Scheduling bei unterschiedlichen *constraints* (Spalten 1 bis 3) und einer *timing-margin* von 4,4 ns (wenn nicht anders angegeben). Die maximal zulässige Gesamtverzögerung für einen Abtastwert liegt bei 4,096 µs. Dunkelgrau schraffiert ist das beste Ergebnis; hellgrau sind weitere hinreichende Parameter-Kombinationen [73] hervorgehoben.

Die Verzögerung durch den zusätzlichen Schleifendurchlauf ist weitaus geringer als die Summe der o. g. Fünf-Schleifen-Struktur. Mit dieser Optimierung konnte die Gesamtverzögerung bei einer Taktperiode von *25 ns* auf *3,5* µs reduziert werden; sie erreicht damit erstmals Echtzeit-Fähigkeit.

> Ausschlaggebend für die Echtzeitfähigkeit der Sechs-Schleifen-Variante ist damit die Kombination aus der Taktperiode von 25 ns, der gewählten *two_hot*-Kodierung für den Steuerautomaten sowie der Einbau von Buffer-Registern an dessen Ausgangssteuersignalen.

5.2.4 Synthese und Simulation

Die Möglichkeiten des entwickelten Entwurfsflusses für den *Synopsys Behavioral Compiler* nutzend, wurde der Entwurf auf eine Standardzell- und zwei FPGA- Technologien abgebildet. Die Tabelle 5.4 stellt die Syntheseergebnisse für diese und die ASIC-Zieltechnologie für den Chip 2 des Perzeptionsmodells gegenüber.

	Altera Flex 10k100A-1	***Xilinx Virtex 800-4***	***ES2-07 µm***
Gatternutzung	2.983 logic cells = 59 %	1.807 Slices = 19 %	15,5 mm^2
Onchip-Speicher	7.200 Bits = 28 %	7.200 Bits in 3 BlockRAMs = 10 %	
Max. Taktfrequenz Gatter-Ebene (*Synopsys*-Schätzung)	31 MHz	21 MHz	57 MHz
Max. Taktfrequenz Layout-Ebene (Messung des jeweiligen-Backend-Werkzeugs)	24 MHz	40 MHz	

Tab. 5.4: Ressourcenverbrauch und maximale Taktfrequenz der für den Chip 2 untersuchten Technologien.

Frühzeitig mußte festgestellt werden, daß die Kapazität der eingesetzten *Altera*-Bausteine nicht ausreichte. Logik- und Speicherressourcen genügten nicht dem Bedarf des gesamten Perzeptionsmodells. Obwohl unterdessen die *Flex*- von der neuen *Altera Apex*- und *Acex*-Bausteingeneration abgelöst ist, wurde aus Verfügbarkeitsgründen auf die *Xilinx Virtex*-Technologie

[73] Die letzte Zeile dokumentiert die Ergebnisse bei Verwendung der eigentlich den *FPGA Compiler*-Bibliotheken äquivalenten *Design Compiler*-Bibliotheken - ein deutlicher Hinweis auf bibliotheksinterne Inkompatibilitäten. Offensichtlich enthalten die *Design Compiler*-Bibliotheken andere (im Verhältnis von Flächenbedarf und Verzögerungszeit ungünstigere) Funktionseinheiten, die für die Technologieabbildung beim *compile* allerdings auf die gleiche Zielfunktion führen.

übergegangen. Die modernere *Xilinx Virtex*-Technologie ist der *Altera Flex*-Technologie in Kapazität und Geschwindigkeit weit überlegen. Die ES2-07 Standardzell-Bibliothek kommt bezüglich der auf Gatter-Ebene in *Synopsys* abgeschätzen Taktrate unmittelbar auf die höchsten Werte. Ein Layout wurde dafür nicht erzeugt, da das den aufwendigen Übergang in eine weitere CAD-Werkzeug-Umgebung bedeutet hätte (*Cadence Design Framework II*).

Diese Ergebnisse illustrieren aber die Performanz-Einschränkungen eines FPGA-Designs im Vergleich zu einem äquivalenten ASIC-Design. In Anbetracht der heute im *Europractice*[74]-Programm verfügbaren Technologien mit 0,35 und 0,25 µm ist hier eine signifikante Beschleunigung des Designs zu erwarten.

Die Relationen bezüglich der Kapazitätsauslastung des *Altera Flex10K100* für den 34 Bit-Festkomma-Dividierer und den alternativen Fließkomma-Dividierers (intern 12 Bit) stehen stellvertretend für alle drei Technologien. Abgesehen von dem hohen Ressourcenbedarf des 34 Bit-Festkomma-Dividierers stößt dessen Synthese an die Grenzen der Handhabbarkeit der *Synopsys*-Synthesewerkzeuge (bis 1.600 MB Hauptspeicherbedarf bzw. generelle Instabilität des Werkzeugs) (Tab. 5.5).

	Altera Flex 10k100A-1
Fließkomma-Dividierer (6 Bit Mantisse, 6 Bit Exponent)	94 Logikzellen, (= 1,9 %) 205 ns Verzögerung
Festkomma-Dividierer (34 Bit)	1.186 Logikzellen, (= 24%) 1.527 ns Verzögerung

Tab. 5.5: Vergleich des Ressourcenbedarfs für den Fest- und Fließkomma-Dividierer anhand der Syntheseergebnisse für die *Altera Flex*-Technologie.

Die Simulationen auf Verhaltens- und Register-Transfer-Ebene stellten zunächst die generelle Funktionstüchtigkeit des Designs sicher. Dabei bestätigt die Simulation des Verhaltensmodells die Werte des *xfloat*-C/C++-Prototypen für den ETSI-Sprachdatensatz (44 Mio Abtastwerte) (siehe Kap. 4.3). Der zeitliche Aufwand ist beträchtlich: das Verfahren benötigte (exklusiv) auf einer *Sun-Enterprise*-Workstation (250 MHz) eine Laufzeit von 8 Tagen. Dabei wurden alle Bestandteile des Perzeptionsmodells, außer den Nachregelschleifen (d. h. Gammatone-Filterbank und Sprachgüteberechnung), in der Referenznotation (C/C++, IEEE-Fließkomma-Arithmetik) belassen. Die Größe der simulierten VHDL-Netzliste liegt bei 120 kB (Chip 2 einschl. Fließkomma-Package und Unterstützungsfunktionen). Die Simulationen der Register-Transfer- und Layoutebene können nur noch für einzelne Werte (Stichproben) durchgeführt wer-

[74] *Europractice* - eine Initiative der Europäischen Union zur Verbesserung der Wettbewerbsfähigkeit der Europäischen Industrie durch die Einführung elektronischer Spitzentechnologien. Mit *Europractice* wird insbesondere Forschungsinstituten und Universitäten der Zugang zur Schaltkreisfertigung ermöglicht.

den[75]. Die Berechnung eines einzelnen Abtastwertes liegt im Minutenbereich. Dabei wächst auch die Größe der VHDL-Netzliste: für die Beschreibung auf Register-Transfer-Ebene werden 1 MB benötigt, für die Layout-Ebene 3,2 MB.

Um auf allen Ebenen neben Beispielvektoren auch die Sprachdaten für eine Sprachgütebeurteilung verwenden zu können, mußte die Testbench für den Chip 2 befähigt werden, diese Daten in großen Blöcken aus einer Datei einzulesen (Abb. 5.16).

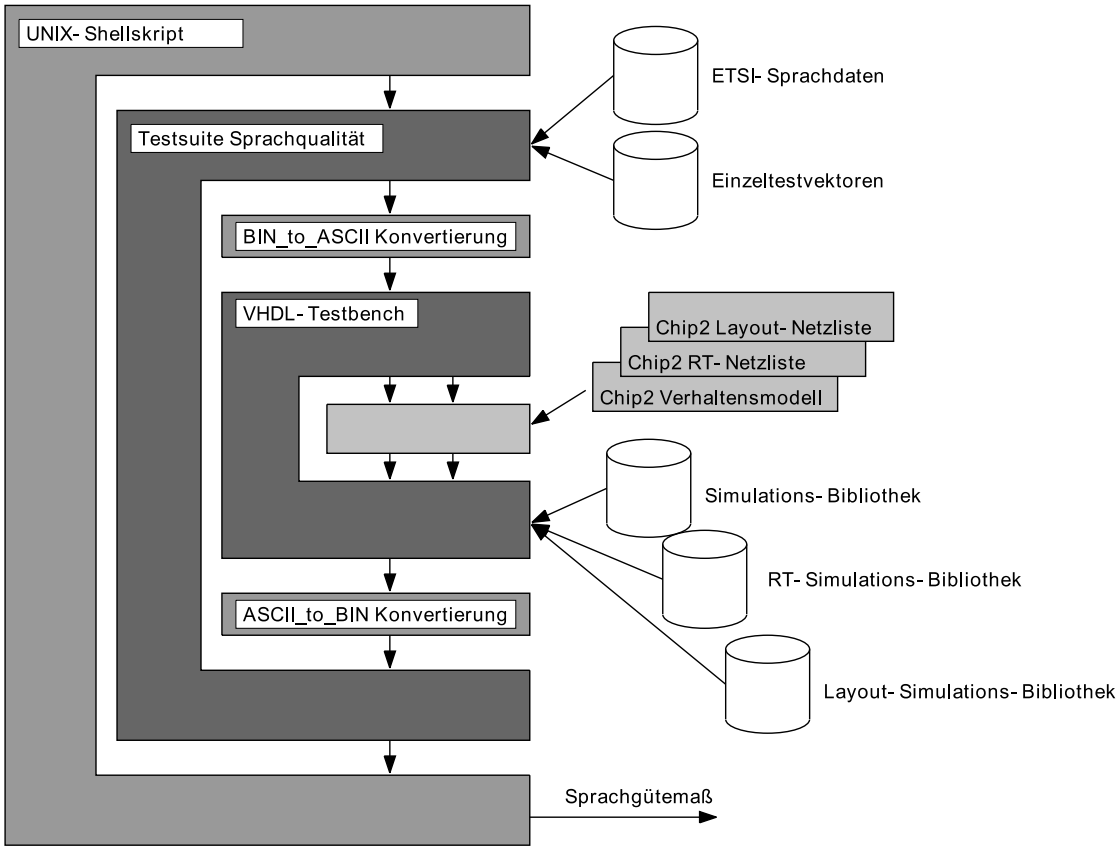


Abb. 5.16: Aufbau der Testbench für den Chip 2 für die wahlweise Simulation der verschiedenen VHDL-Netzlisten. Über das VHDL-Konstrukt *configuration* kann die Instanz des Chip 2-Designs für alle drei Beschreibungsebenen (Verhaltens-, Register-Transfer- und Layout-Ebene) gewählt werden.

Der Stimuli-Konverter der *Synopsys*-Simulationsumgebung akzeptiert nur ASCII-Daten. Deswegen wurden die dafür vorgesehenen ASCII-Export/Import-Schnittstellen für den Prototypen des Perzeptionsmodells einschließlich der Sprachgütebestimmung verwendet. Damit können die (binären) 16 Bit-Ein- und Ausgangsdaten des Chip 2-Designs in ASCII-Daten umgewan-

[75] Erinnert sei auch daran, daß für die Simulation der Register-Transfer-Ebene die ursprünglich vom Synthesewerkzeug generierte Netzliste nicht simulierbar war. Mit Synthese-Skripten mußte diese erst so aufbereitet werden, daß keine vom *compile* belegten Look-Up-Tables (in die alle *--synopsys preserved_functions* bereits abgebildet waren, siehe Kap. 5.2.3 und Anhang A) mehr enthalten sind.

delt und durch die Testbench an die UUT[76], also den Chip 2, gegeben werden. Die Abbildung 5.17 zeigt die Simulation der Layout-VHDL-Netzliste.

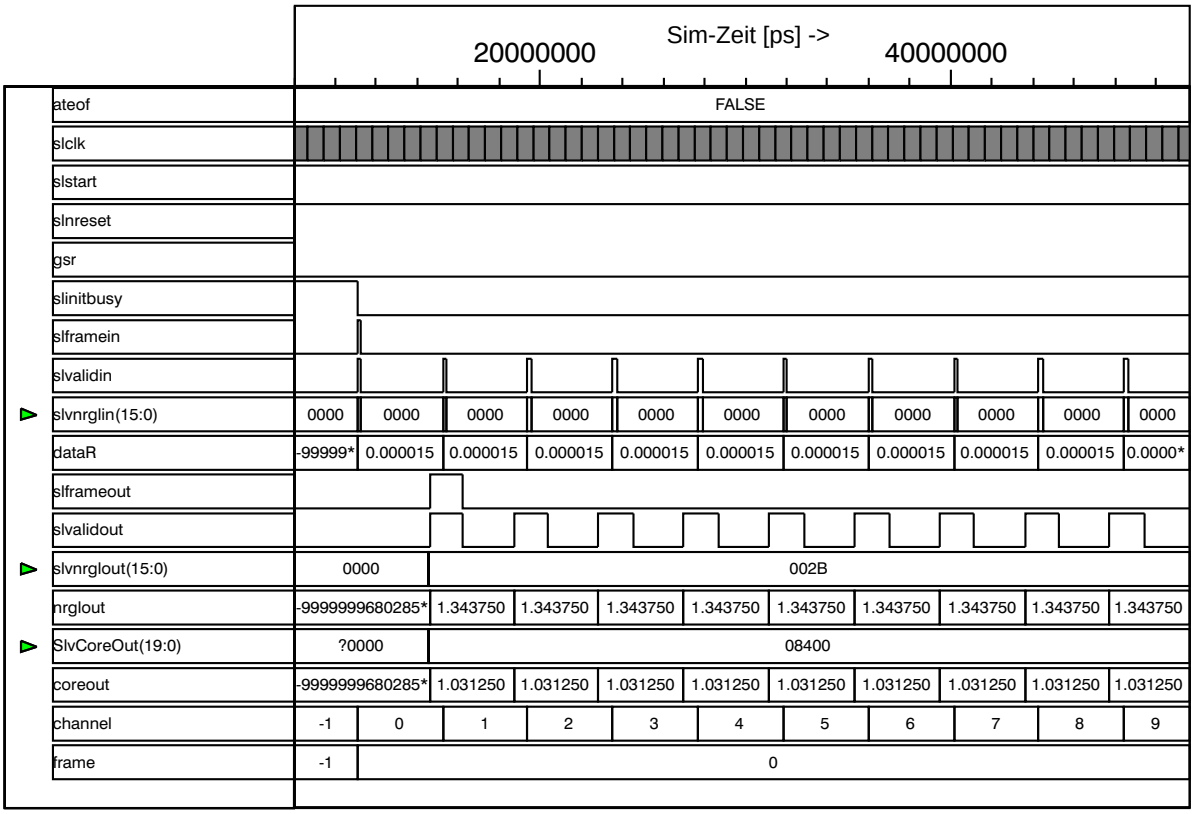


Abb. 5.17: Timing-Diagramm der Layout-VHDL-Netzliste. Die Signale (bis auf *slnreset* high-aktiv) bezeichnen:

sleof:	Flag, End Of File
slclk:	Systemtakt
slstart:	Startflag der Simulation
slnreset:	Reset, active low
slinitbusy:	Flag, Initialisierungsphase läuft
slframein:	Flag, mit aktuellen Eingangsdaten startet ein neuer Frame
slvalidin:	Flag, aktuelle Eingangsdaten sind gültig
slvnrglin:	Vektor, Eingangsdaten (16 Bit)
dataR:	Eingangsdaten in *Real* Darstellung
slframeout:	Flag, mit aktuellen Ausgangsdaten startet ein neuer Frame
slvalidout:	Flag, aktuelle Ausgangsdaten sind gültig
slvnrglout:	Vektor, Ausgangsdaten (16 Bit)
nrglout:	Ausgangsdaten in *Real* Darstellung
SlvCoreOut:	Vektor, Ausgangsdaten der fünften Nachregelschleife (20 Bit)
coreout:	Ausgang der fünften Nachregelschleife in *Real*-Darstellung
channel:	aktuelle Frequenzkanalnummer
frame:	aktuelle Frame-Nummer (Index der Stereo-Abtastwerte)

Nachdem die Initialisierung abgeschlossen ist (*slinitbusy* -> ‘0‘), erfolgt das Anlegen eines Eingangswertes (*slvnrglin*, bzw. *dataR*) und der entsprechenden Flags. Das Signal

[76]UUT - *unit under test*.

slframein = ‘1‘ zeigt den ersten Frame an, der die 2 x 30 Frequenzkanäle für einen Abtastwert umspannt. Der 16 Bit-Eingangs-Bitvektor *slnrglin* (15:0) wird aus der ASCII-Stimulidatei gelesen, angelegt und mit *slvalidin* = ‘1‘ als gültig signalisiert. Hat der Chip 2 diesen Abtastwert verarbeitet, wird sowohl der Ausgangs-Bitvektor *slvnrglout* (15:0) aktualisiert, als auch der 20 Bit-Testvektor *slvcoreout* (19:0) ausgegeben, der das Resultat der letzten Nachregelschleife direkt zur Verfügung stellt. Bis zum Zeitpunkt der ersten vollständigen Berechnung nach Reset und Initialisierung können die Ausgänge noch ungültige Signale führen („?“ bzw. falsche Interpretationen der Ausgänge in Abb. 5.17). Mit *slvalidout* =‘1 ‘ wird dann die Ausgabe eines gültigen Ergebnisses für einen Frequenzkanal und mit *slframeout* = ‘1‘ die vollständige Berechnung aller Frequenzkanäle eines Abtastwertes (d. h. eines Frames) angezeigt. Diese Flags steuern somit die Übergabe des Ausgangs von Chip 2 an die nachfolgenden Teildesigns des Perzeptionsmodells. Die Signale *coreout* und *nrglout* stellen die den Bitvektoren äquivalenten *Real*-Zahlenwerte dar. Diese Stichproben stimmen exakt mit den Daten aus der C++ Simulation überein.

Ergebnisse

Damit ist es insgesamt gelungen, den Algorithmus der binauralen Nachregelschleifen effizient und in Echtzeit lauffähig als FPGA-Design zu implementieren. Als Vorstufe wurde ein Set von generisch parametrisierbaren Fließkomma-Operatorschaltnetzen sowohl in C++ als auch in VHDL-Kode entwickelt, die als Arithmetik-Kern den Flächen- und Durchsatz-Anforderungen des Perzeptionsmodells gerecht werden können. Auf einem *Xilinx Virtex*-FPGA eines Prototypensystems verbraucht der binaurale Chip 2 allein 19 % der Logik- und Gatterressourcen (äquivalenter *gate count*: 80.000) und 10 % des Onchip-RAM-Speichers. Damit verbleiben 81 % der Logikressourcen des FPGA für Chip 1 und die internen und externen Schnittstellen. Die maximale Taktfrequenz erreicht 40 MHz. Die für eine Echtzeit-Datenverarbeitung zulässige maximale Verzögerung von 4,096 µs je Abtastwert wird um etwa 0,5 µs unterschritten. Das Prinzip der direkten Datenpfadsynthese ermöglicht das Abbilden ein und desselben VHDL-Kodes auf ganz unterschiedliche Zieltechnologien, was im Falle des Chip 2-Designs mit einer *Altera*-FPGA, *Xilinx*-FPGA- und einer Standardzell-Technologie (ES2-07) erfolgreich nachgewiesen wurde. Für die Synthese der Schaltung wurde die direkte Datenpfadsynthese des *Synopsys Behavioral Compilers* genutzt, wobei die Entwicklung eines für die FPGA-Technologie adaptierten durchgängigen Entwurfsflusses wesentlich höhere Ansprüche stellte als vorab angenommen und sich als größter Einzelanteil am Arbeitsumfang des Projektes erwies. Die korrekte Arbeitsweise des Algorithmus’ konnte durch Simulation auf Verhaltens-, Register-Transfer- und Layout-Ebene bestätigt werden. Den gesamten dafür entwickelten Entwurfsfluß, beginnend mit der ersten Syntaxprüfung des VHDL-Kodes bis zum Export der Netzlistenbeschreibungen nach dem Scheduling, enthält Anhang A. In Form des Scheduling-Skripts ist hier die gesamte skriptbasierte Steuerung des *Synopsys Behavioral Compilers* einschließlich aller Anweisungen zur Umgehung bzw. Korrektur von Werkzeug- bzw. Bibliotheksfehlern dokumentiert.

Kapitel 6

Das Prototypensystem

Da das „*Silicon Ear*" von Beginn an als praktisch einsetzbares System konzipiert war, gibt dieses Kapitel einen Überblick zum Einsatz und der Integration aller Designanteile auf dem Demonstrator, einem FPGA-Entwicklungs-Board. Darüber hinaus wird dessen Einbindung als Echtzeit-Hardware in ein übergeordnetes PC-Signalverarbeitungssystem skizziert.

6.1 „Silicon Ear" Signalverarbeitungssytem

Bereits bei der Spezifikation des Chip-Entwurfs waren genaue Vorüberlegungen zum späteren Einsatz des Chip-Satzes erforderlich. Da das „*Silicon Ear*" noch keine „Stand Alone"-Anordnung ist, muß die Systemumgebung sowohl verschiedene Inbetriebnahme-Tests als auch den Datentransfer zu und von den Chips im Normalbetrieb gewährleisten. Nur mit einer durchsatzstarken Anbindung an ein übergeordnetes Host-System kann die Leistungsfähigkeit der Chips genutzt werden. Zum Zeitpunkt der Spezifikation des Demonstrators boten allerdings nur sehr wenige Prototypen-Systeme mit einem *Xilinx Virtex*-FPGA die notwendige Logikdichte und Takt-Performanz auf einem Standard-PCI-Board[77]. Das Prototypen-System „*Ballynuey 2*"[78] jedoch vereint ausreichende Logikausstattung, Performanz und Modularität, da vier Sockel für weitere Subsysteme oder Testadapter vorgesehen sind. Es enthält einen separaten PCI-Core, ein vollständig nutzbares *Virtex* FPGA mit 800.000 Gatteräquivalenten und 4 MB externen ZBT-RAM[79]. Die PCI-Bus-Anbindung beherrscht das selbständige *busmastering* und stellt somit einen DMA-gesteuerten Blocktransfer zur Verfügung. Das Anwenderdesign wird während des Entwurfsprozesses auf einer Workstation oder einem PC zu einem ladbaren Bit-File kompiliert. Da das FPGA einen flüchtigen Designspeicher besitzt, muß das Design nach jedem Einschalten neu geladen werden, was aber auch im Betrieb per Laderoutine erfolgen kann. Im jederzeit möglichen Austauschen des Designs besteht der eigentliche Hauptvorteil eines solchen FPGA-Prototypensystems. Das Board besitzt drei unabhängige Taktnetzwerke für alle Schaltungen und Designs auch auf den Subplatinen, so daß sowohl ein synchroner Datentransfer, als auch eine dedizierte Taktversorgung gewährleistet sind. Die Modularität der Karte erlaubt desweiteren eine komfortable Prüfung und den Betrieb von ASIC-Versionen des Perzeptionsmodells, indem die Chips über die Testadapter angeschlossen werden und in derselben Umgebung

[77] PCI - *Peripheral Component Interconnect.*

[78] „*Ballynuey 2*" - PCI-FPGA-Prototyping System der *Nallatech, Ltd.*, UK.

[79] ZBT - *Zero Bus Turnaround*, Schreiben und Lesen (Datenrichtung des Busses) kann ohne Verzögerung mit jedem Bustakt wechseln.

arbeiten können. Abbildung 6.1 zeigt die FPGA-PCI-Karte und die über Steckverbinder anschließbare AD/DA-Wandler-Baugruppe (Eigenentwicklung).

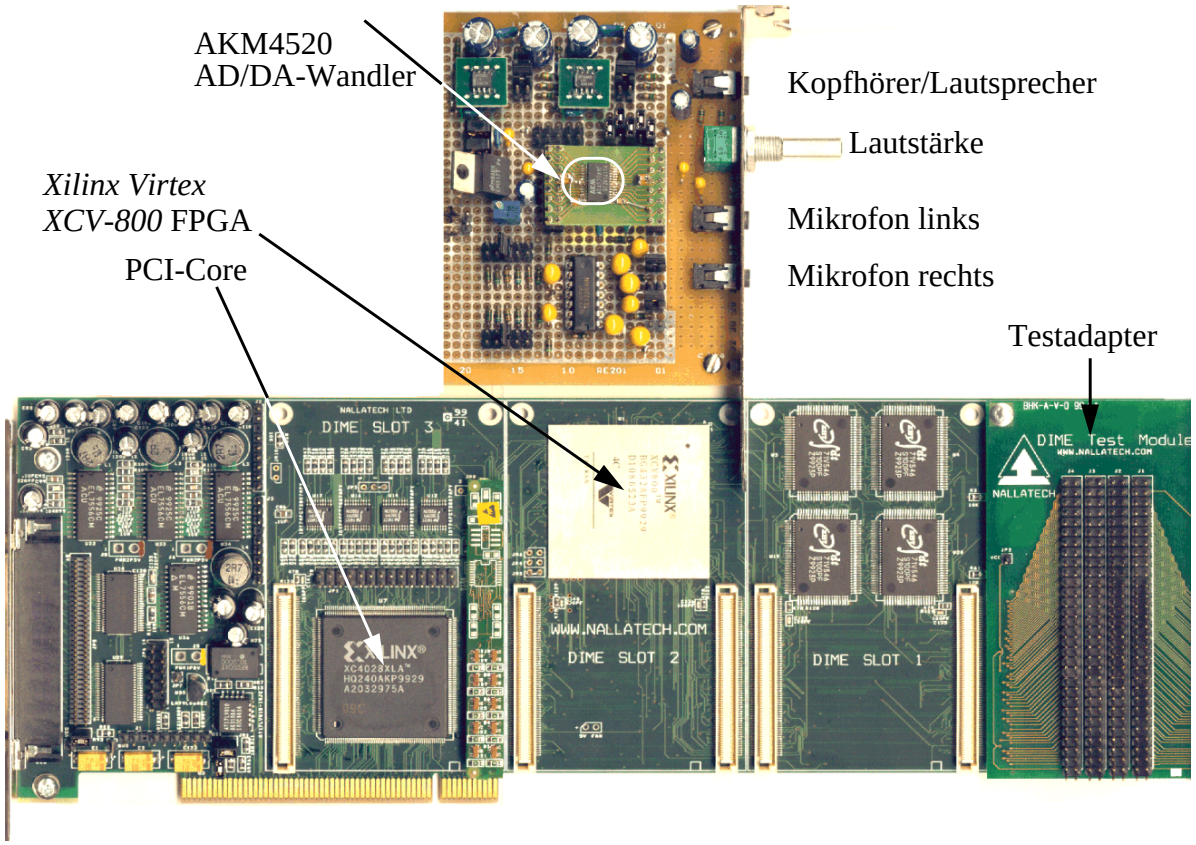


Abb. 6.1: Aufbau des Prototypensystems aus *Nallatech*-FPGA-Entwicklungsboard und Audio-Erweiterung mit dem AD/DA-Wandler.

6.1.1 Kommunikation zum Host-System

Betrachtet man den Gesamtaufwand zur Implementierung des Perzeptionsmodells auf dem Demonstrator, so kommen notwendige Steuer-Strukturen für den AD/DA-Wandler und den Datentransport von/zur internen Nutzer-Schnittstelle des Demonstrators und zum PCI-Bus hinzu. Dieser I/O-Master genannte Kommunikationsblock enthält neben einem Steuerautomaten auch Pufferspeicher (FIFO), da der Datenfluß über die Bussysteme des Host-PCs in das übergeordnete Anwendungsprogramm keinesfalls als kontinuierlich vorausgesetzt werden kann. Die Hauptaufgabe des I/O-Masters besteht in der Synchronisation der Daten-Ports des Perzeptionsmodells, des AD/DA-Wandlers und der internen PCI-Schnittstelle des FPGA-Boards. Zusätzlich können diese Kommunikationspfade in zwei Arbeitssmodi betrieben werden:

- **Host-Modus**: das Perzeptionsmodell erhält abgetastete Audiodaten über das Anwendungsprogramm oder aus der Soundkarte des PC.
- **Codec-Modus**: das Perzeptionsmodell erhält abgetastete Audiodaten aus dem eigens vorgesehenen AD/DA-Wandler.

In beiden Modi werden für jeweils vier sequentielle Eingangs-Abtastwerte die Ausgangssignale aller 30 Frequenzkanäle berechnet (vierfaches Unterabtasten etc., siehe Kap. 4). Um den PCI-Bus nicht für jeden Ein- und Ausgangswert unterbrechen zu müssen (was mit der Abtastrate von 16 kHz nicht zu bewerkstelligen wäre), sollen bei jedem PCI-Blocktransfer 64 Stereo-Eingangsdaten übertragen werden. Damit wird der PCI-Bus alle *64/16.276 Hz = 3,93 ms* unterbrochen. Um die Busbreite von 32 Bit voll nutzen zu können, werden die jeweils 16 Bit breiten Daten vom linken und rechten Kanal zu einem 32 Bit-Wort zusammengefügt. Für den o. g. Block aus 64 Eingangsdaten (zu je 2 x16 Bit) produziert das Perzeptionsmodell 480 Ausgangsdaten (zu je 2 x 16 Bit). In Abb. 6.2 ist das Block-Schaltbild des Demonstrators für beide Arbeitsmodi dargestellt.

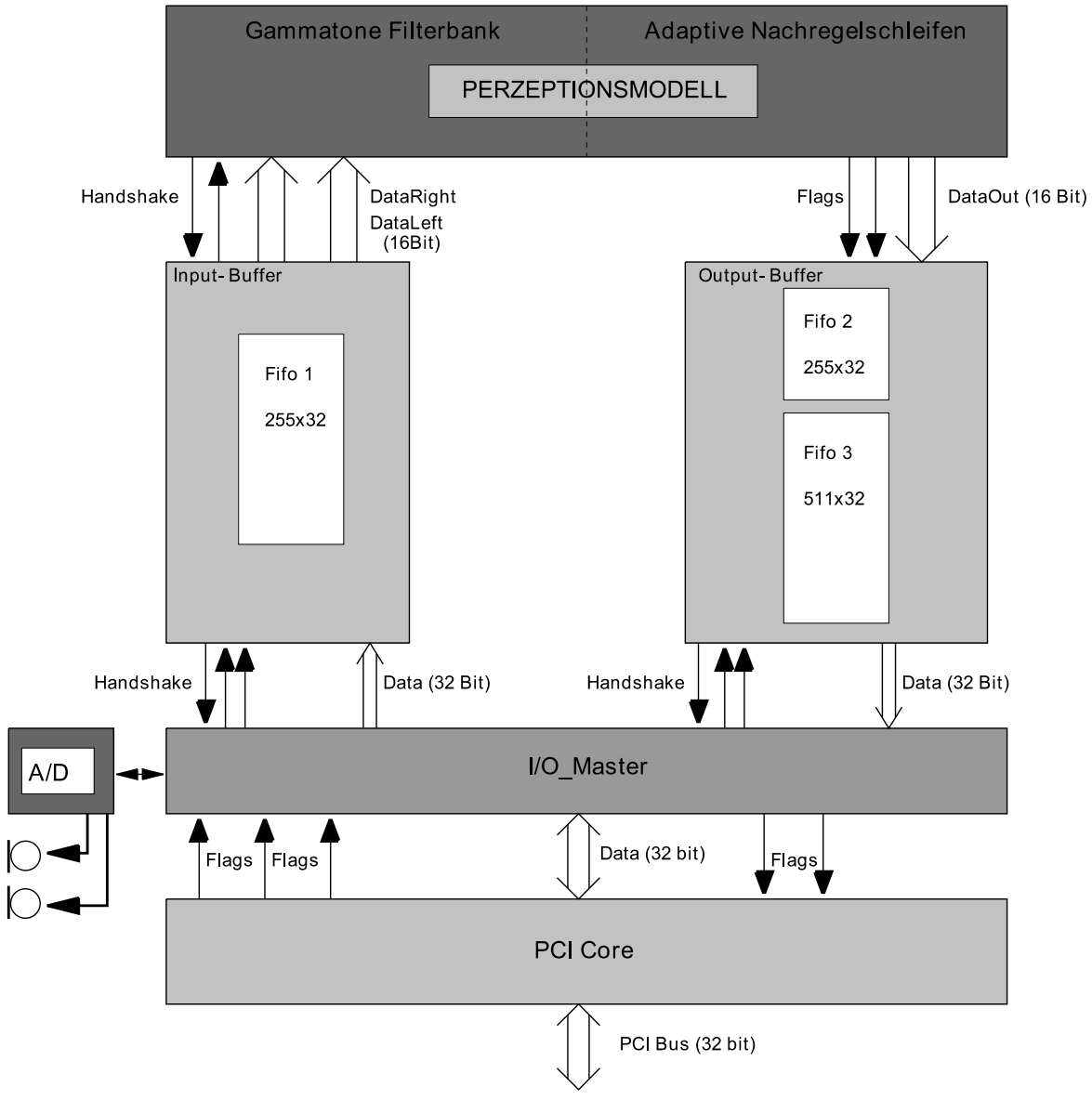


Abb. 6.2: Innere Struktur des Demonstrators, der beide Arbeitsmodi beherrscht.

Trotz des Blocktransfers für Ein- und Ausgangsblöcke, kann eine Unterbrechung der Verarbeitung im Perzeptionsmodell nicht ausgeschlossen werden. Um Datenverlusten vorzubeugen,

werden deshalb die Puffer-FIFO mit doppelter Blocklänge vorgesehen. Da andererseits die Puffer-FIFO als *Xilinx*-Makros aus Onchip-RAM-Blöcken bereitgestellt werden und dabei Restriktionen in der Dimensionierung bestehen[80], wird das Eingangs-FIFO 255 Positionen tief gewählt. Am Eingang der Gammatone-Filterbank (Chip 1) ist ein echtes Handshake möglich, d. h. der I/O-Master bestimmt, wann das Perzeptionsmodell mit der Berechnung eines Abtastwertes beginnen kann. Am Ausgang der Nachregelschleifen ist es jedoch erforderlich, ein Polling der Port-Flags durchzuführen, um dem Perzeptionsmodell Daten abnehmen zu können. Das Ausgangs-FIFO läßt sich nun aus Gründen der begrenzten Anzahl von internen RAM-Blöcken des FPGA nicht auf die gewünschten 960 Positionen ausdehnen, sondern nur als sequentielles Doppel-FIFO mit 511 + 255 Positionen einsetzen. Diese Teilung ermöglicht außerdem die Auswertung des *fifo_full* Flags des kleineren FIFO. Ist es gesetzt, befindet sich mindestens ein kompletter Block aus 480 Positionen im Ausgangs-Puffer.

Der integrierte AD/DA-Umsetzer[81] enthält keinen eigenen Generator zur Erzeugung der notwendigen internen Taktsignale, sondern erwartet diese über die serielle Datenschnittstelle vom I/O-Master. Dieser muß dafür insgesamt drei zueinander synchrone Takte erzeugen: den System-Takt für die internen Filter des Wandlers, den Bit-Takt der seriellen Schnittstelle und den Links/Rechts-Takt, welcher der Abtastfrequenz entspricht. Außerdem sind die abgetasteten Mikrophonsignale nach Empfang zu parallelisieren und in das Eingangs-FIFO zu schreiben. Für die Takterzeugung wird einer der vier programmierbaren Taktgeneratoren des *Virtex*-FPGA genutzt. Aus 33 MHz können mit ganzzahligen Teilern alle benötigten Takte abgeleitet werden. Ein wesentliches Problem ist die Ansteuerung der PCI-Bridge. Diese ist fest auf dem FPGA-Board installiert und besteht aus einem kommerziellen PCI-Core in einem separaten *Xilinx Spartan*-FPGA. Die Übertragung zum I/O-Master geschieht asynchron über einen 32 Bit breiten Datenbus und wird mit fünf Handshake-Signalen gesteuert. Da die PCI-Bus-Daten mit 33 MHz getaktet werden, sind auch hier bei der Ansteuerung vom I/O-Master auf der „Rückseite“ 33 bzw. 40 MHz vom Hersteller empfohlen. Dazu wird ein weiterer Taktgenerator des *Virtex*-FPGA eingesetzt, der aber gleichzeitig den Takt für die Nachregelschleifen bereitstellen kann.

6.1.2 Synthese des Kommunikationsblocks

Der I/O-Master benötigt auf dem FPGA nur eine geringe Anzahl von Ressourcen:

- 372 Slices (entspricht 3 % der Gesamtkapazität)
- zwei (von max. vier) Takt-Generatoren
- 51 User-I/O-Pins (von max. 316 des gesamten FPGA)
- 8 Block-RAM-Zellen zu je 4.096 Bit. Der I/O-Master benötigt diese für die Datenpuffer, die restlichen werden vom Perzeptionsmodell selbst belegt.

[80] Kombinationen aus Breite x Tiefe sind nicht beliebig wählbar [Xilinx Inc. 1999].

[81] AKM-4520a der *Asahi Kasei Microsystems Co., Ltd.*, Japan.

Obwohl der *Xilinx Timing Analyzer* auftretende Zeit-Verletzungen von bis zu 5 ns auf Layout-Ebene an fast allen Datenleitungen meldet, konnte der I/O-Master selbst mit der vollen Geschwindigkeit getestet werden. Offensichtlich erfolgte hier eine sehr pessimistische Timing-Schätzung insbesondere der Block-RAM-Ansteuerung im FIFO, aus denen die Verzögerungen stammen. Das Design wurde auf eine Taktfrequenz von 40 MHz hin optimiert. Die theoretisch mögliche Datentransfergeschwindigkeit reduziert sich jedoch durch die asynchrone Schnittstelle zum PCI-Core, denn hier werden zum Übertragen eines 32 Bit-Wortes vom PCI-Core zum I/O-Master durch das Handshake-Verfahren mindestens vier Takte benötigt. Geht man von 10 MHz mittlerer genutzter Taktfrequenz aus, so werden dennoch hinreichend hohe Datentransferraten von 40 MByte/s erzielt.

6.2 Systemdemonstration

Die Systemdemonstration erfolgt mit dem flußgraphenorientierten Signalverarbeitungsprogramm *Hypersignal*[82]. Diese Teilaufgabe war nicht Gegenstand der vorliegenden Arbeit, soll aber zum Verständnis des Gesamtprojektes hier kurz erläutert werden[83].

Die Software *Hypersignal* ist vornehmlich für die Audiodatenverarbeitung entwickelt worden und gestattet die Einbindung selbst definierter Blöcke, die wiederum lokal eine direkte Hardware-Unterstützung einschließen können. Anwendungen werden als graphische Schaltbilder aus vorgefertigten oder selbst erstellten Blöcken entwickelt. Eigene Verarbeitungseinheiten können als C/C++ Implementationen einbezogen werden. Entsprechend den verschiedenen Modi des Demonstrators ist diese Umgebung schnell zu adaptieren und stellt graphische Ausgabemöglichkeiten (z. B. Zeit-Frequenzdarstellungen) zur Verfügung. Damit ermöglicht die Software außerdem die Einbindung und den direkten Vergleich zur C/C++ Referenz des Perzeptionsmodells. Die Einbeziehung der FPGA-Prototypenkarte erfolgt mit den Funktionen der mitgelieferten API[84], die als C/C++ Bibliothek direkt in den Rahmen der Blockdiagramm-Umgebung übernommen wird. Zu Beginn wird das Bit-File des FPGA-Designs geladen, danach startet die Datenkommunikation, wobei die o. g. Datenblöcke im PCI-Burst-Modus übertragen werden.

Der *Hypersignal-workspace* für den Demonstrator besteht im wesentlichen aus dem programmierten Perzeptionsmodell (PEMO), den austauschbaren Datenquellen (hier zwei Audiodateien, die in definierten Blocklängen ausgegeben werden können), graphischen Direktanzeigen, einer Kreuzkorrelation (als Voraussetzung zur Bestimmung der Sprachqualität zwischen ver-

[82] *Hyperception, Inc.*, USA.

[83] Die Applikationssoftware zur Systemdemonstration wurde von einem Projektpartner, der Arbeitsgruppe Medizinische Physik (Fachbereich Physik) an der "Carl-von-Ossietzky" Universität Oldenburg erstellt.

[84] API - *Application Program Interface*.

zerrtem und unverzerrtem Signal, siehe Kap. 3.4.2) und zwei Zeit-Frequenzdarstellungen der internen Repräsentation, d. h. der Perzeptionsmodell-Ausgänge (Abb. 6.3). Als Quellen für die Audiodaten können entweder einzelne Sprachsignaldateien, die PC-Soundkarte oder der AD/DA-Wandler des Demonstrators gewählt werden.

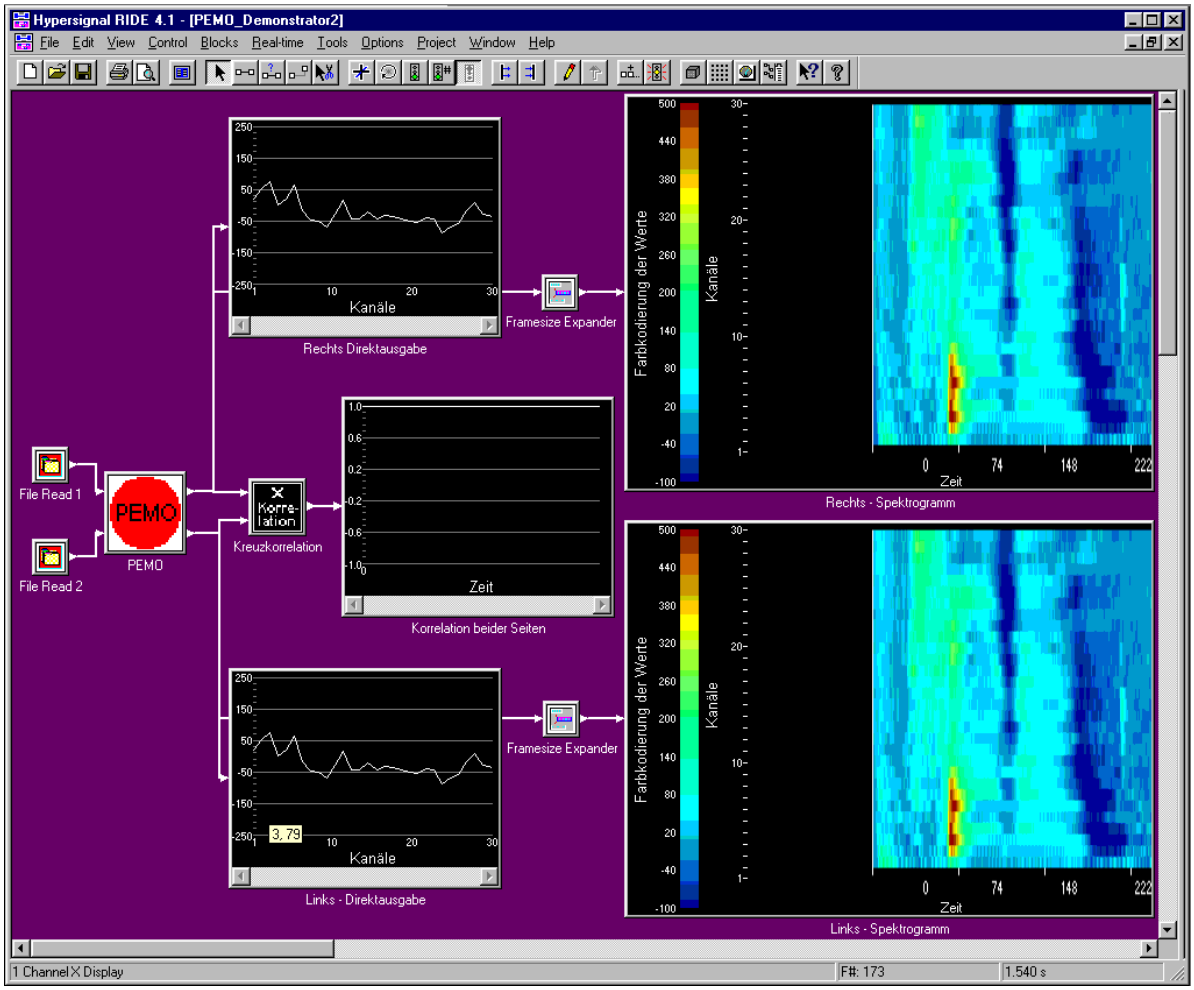


Abb. 6.3: Der *workspace* des „*Hypersignal*“-Blockdiagramms für eine offline-Demonstration des Perzeptionsmodells. Linker und rechter Kanal enthalten hierfür identische Daten, das gesprochene Wort „Sieben“. Das Zeit-Spektrogramm der internen Repräsentation veranschaulicht u. a. die Silbenstruktur und die hohe Signalenergie im Bereich der Formanten des Vokals „i“.

Durch die Entlastung der CPU von der Vorverarbeitung werden nun Rechenkapazitäten für die beabsichtigten Applikationen frei. Die objektive Sprachgütebeurteilung von Kommunikationskanälen, die Einzelwort- oder kontinuierliche Spracherkennung und der Analysepfad als Voraussetzung zur Resynthese nach Frequenz- und Dynamikkorrektur (für digitale Hörgeräte) werden durch dieses modular erweiterbare Echtzeitsystem ideal unterstützt. Die optische Darstellung alltäglicher akustischer Signale in einer Form, wie sie den höheren neuronalen Stufen des menschlichen Nervensystems z. B. zur Erkennung der Sprache dargeboten wird, bietet darüber hinaus einen hohen verständnisfördernden Anschauungswert.

Kapitel 7

Zusammenfassung und Ausblick

Die vorliegende Dissertation beschreibt die Konzeption, Entwicklung und Realisierung eines digitalen Signalverarbeitungssystems zur gehörgerechten Sprachvorverarbeitung. Die Arbeiten waren Bestandteil eines interdisziplinären Projektes[85]. Nach einer Integration der dabei entstandenen Ergebnisse und Teildesigns kann das vollständige binaurale Modell der effektiven auditorischen Signalverarbeitung (Oldenburger Perzeptionsmodell) als flexibles digitales Koprozessorsystem den Zielapplikationen zur Verfügung gestellt werden.

Die technische Anwendung gehörgerechter Sprachvorverarbeitung erfordert die modellbasierte und echtzeitfähige Umwandlung der akustischen Signale in die benötigte interne neuronale Repräsentation. Trotz eines zugeschnittenen Funktionsumfanges verbleibt aufgrund der algorithmischen Komplexität ein hoher Berechnungsaufwand, der mit einer anwendungsspezifischen integrierten Schaltung bewältigt werden kann. Der sich aus den hier vorgestellten Arbeiten ergebende Anteil am Gesamtsystem umfaßt die Untersuchungen und Lösungsvorschläge zur Optimierung und Implementierung der adaptiven nichtlinearen Nachregelschleifen des Perzeptionsmodells.

Entstanden sind sowohl eine Konzeption zur gehörgerechten Optimierung und Validierung der im Hinblick auf einen minimalen Schaltungsaufwand reduzierten Algorithmen als auch der Entwurfsfluß zur Umsetzung in eine integrierte Schaltung. Dabei galt es insbesondere, die Einflüsse und Rückwirkungen aus Architekturvarianten, Randbedingungen und Beschränkungen des Designs-Systems und der untersuchten FPGA- und ASIC-Technologien auf die Optimierung des Entwurfsflusses und des Algorithmus' selbst so zu berücksichtigen, daß ein den Spezifikationen entsprechendes Signalverarbeitungssystem überhaupt erarbeitet und aufgebaut werden konnte.

Nachfolgend werden die dabei berücksichtigten Teilaspekte und die erreichten Ergebnisse zusammengefaßt.

Algorithmusanalyse und -optimierung für die adaptiven Nachregelschleifen.

Die adaptiven Nachregelschleifen des Oldenburger Perzeptionsmodells berechnen für jeden Frequenzkanal der eingangsseitigen Filterbank die nachgeregelten und zeitlich kontrastierten „Energie"-Werte, die zur Darstellung einer „internen" psychoakustischen Repräsentation auditorischer Signale benötigt werden. Dazu wurden am Algorithmus mit einem dafür entwickelten

[85] DFG-gefördertes Projekt *„System- und Schaltungstechnik einer integrierten Cochlea für Sprachanalyse, Spracherkennung und Sprachcodierung"* (Me 1289/4 / Ko 942/12).

C/C++ Prototypen umfangreiche Untersuchungen zu notwendigen Wortbreiten und möglichen Optimierungen für den Algorithmus und das Datenformat durchgeführt. Dieser Prototyp wurde sowohl in einer Festkomma-Variante als auch für eine Fließkomma-Implementation mit skalierbarer Präzision erstellt. Die variable Quantisierung war die Voraussetzung für die nachfolgende systematische Minimierung der Wortlängen.

Konzeption und Umsetzung einer gehörgerechten Optimierung bzw. Validierung der notwendigen Anpassungen für den Algorithmus und das Datenformat.

Aufgrund der Ressourcenbeschränkungen und der mangelnden Synthetisierbarkeit erfolgte zunächst der Übergang vom ursprünglich geplanten, parallelen Festkomma-Design zu einem sequentiell arbeitenden Festkomma-Prozessor und später zu einen speziellen, selbstentwickelten Fließkomma-Kern.

Dieser teilweise Vorgriff auf den Syntheseschritt war erforderlich, weil die Algorithmusoptimierung in erster Linie die zur Verfügung stehenden Hardware-Ressourcen und die CAD-Werkzeugeigenschaften zu berücksichtigen hatte. Die zentralen arithmetischen (vorzeichenlosen) Operationen wie Addition, Subtraktion und Multiplikation arbeiten mit der Präzision einer 14 Bit Mantisse und einer exzeß-kodierten Charakteristik von 6 Bit. Die Division, die in der Festkomma-Variante (34 Bit) einen außerordentlichen Ressourcenverbrauch verursachte und sowohl das Design insgesamt als auch das CAD-System instabil werden ließ, konnte in der Fließkomma-Variante mit einer Operandenpräzision von lediglich 6 Bit Mantisse und 6 Bit Charakteristik erfolgreich implementiert werden. Diese wesentlichen Optimierungen wurden dadurch ermöglicht, daß nach geeigneten Umformungen des Algorithmus' bestimmte Zwischenergebnisse in gut quantisierbaren Zahlenbereichen verbleiben konnten. Die optimale Architektur besteht aus einer gemischten Festkomma-Fließkomma-Struktur. Der nichtlineare Teil (Nachregelschleifen) wurde mit einem Fließkomma-Rechenwerk realisiert, der lineare Teil des ausgangsseitigen Tiefpaßfilters wurde dagegen in der Festkomma-Notation belassen, weil damit die hier benötigten Operationen effektiv synthetisiert werden konnten. Außerdem ergibt sich dadurch ein Minimum an Formatkonvertierungen zwischen dem Ein- und dem Ausgang des Designs. Gezielte algebraische Umformungen zur Reduktion der benötigten Operationen erwiesen sich dagegen nach den Syntheseschritten insgesamt als nachteilig.

Im Zusammenwirken mit einem dafür angepaßten Verfahren zur Sprachgütebeurteilung durch das Perzeptionsmodell selbst konnte die optimale Architektur entwickelt werden. Bei diesem Verfahren wird eine objektive, maschinelle Sprachgütevorhersage für unverzerrte und verzerrte Eingangsdaten vorgenommen und anschließend mit einer subjektiven Bewertungen derselben Datenbasis korreliert. Damit kann zunächst die Zuverlässigkeit des objektiven Kriteriums nachgewiesen werden. Da die objektive Bewertung aus den perzeptiv relevanten Signalen („interne" Repräsentation am Ausgang des Perzeptionsmodells) gewonnen wird, konnten die Schwankungen der nachfolgenden Korrelation mit dem subjektiven Maß erfolgreich zur Bewertung der zusätzlichen arithmetischen Quantisierungsverzerrungen im reduzierten Algorithmus herangezogen werden.

Erstellung der synthetisierbaren VHDL-Beschreibung für den Algorithmus.

Zunächst wurde der VHDL-Kode auf generelle Synthetisierbarkeit hin optimiert. Das betraf nicht nur den Algorithmus selbst, sondern vielmehr die wesentlich umfangreichere Umsetzung der arithmetischen Fließkomma-Operatoren. Die mit den C/C++ Prototypen spezifizierten VHDL-Fließkomma-Funktionen wurden so erstellt, daß sie durch Operatorüberladung aus einer synthetisierbaren VHDL-Bibliothek inferiert werden können. Damit wird deren Verwendung wesentlich vereinfacht und vom CAD-Werkzeug (*Synopsys Behavioral Compiler)* ebenso unterstützt, wie dessen eigene Festkomma-Operatoren. Durch ein Reihe weiterer Hilfs- und Konvertierungsroutinen wurden die ebenfalls benötigten Schnittstellen zu den Festkomma-Datentypen bereitgestellt.

Entwicklung eines durchgängigen Entwurfsflusses zur direkten Schaltungssynthese der Algorithmen mit dem *Synopsys Behavioral Compiler*, sowohl für FPGA- als auch ASIC-Technologien.

Für die Synthese des Schaltkreises wurde kein konventioneller struktureller Ansatz gewählt. Vielmehr wurde ein durchgängiger Entwurfsfluß für die direkte Datenpfadsynthese mit einem speziellen CAD-Werkzeug (*Synopsys Behavioral Compiler*) entwickelt. Dabei wird der Signalverarbeitungsalgorithmus nach der Bereitstellung von Operatorschaltnetzen in Designbibliotheken und der Erzeugung individueller technologieabhängiger RAM-Bausteine direkt synthetisiert. Als Ergebnis der dabei durchlaufenen Schritte Ressourcen-Allokation, Scheduling und Binding ergibt sich das ressourcen- und zeitoptimierte Gesamtdesign, bestehend aus der geforderten Berechnungsabfolge und einem automatisch generierten Steuerwerk.

Der herausragende Vorteil dieses Vorgehens besteht vor allem darin, daß ein und derselbe technologieunabhängige VHDL-Kode auf unterschiedliche Zieltechnologien abbildbar ist. Mit dem entwickelten Entwurfsfluß wurde der verhaltensbeschreibende VHDL-Kode für den Nachregelschleifenalgorithmus auf zwei FPGA- (*Altera, Xilinx*) und eine Standardzell-Technologie (ES2-07) erfolgreich abgebildet.

Da die verwendete FPGA-Technologie systembedingt nicht dieselben hohen Taktraten wie die Standardzell-Designs erreichen, konnte mit der direkten Umsetzung des Algorithmus in VHDL zunächst keine Echtzeitfähigkeit erlangt werden. Somit bestand ein weiterer, wesentlich aufwendigerer Schritt darin, eine optimierte Operationsreihenfolge zu bestimmen. Erst die weitere Verschiebung von Operationen im VHDL-Kode durch Einführung einer zusätzlichen Berechnungsiteration in der zentralen Schleife gestattete dem automatischen Scheduling eine effiziente Operatorentkopplung und damit eine wesentlich geringere, nun echtzeitfähige, Gesamtverzögerung des Nachregelschleifenteils. Eine weitere Bedingung dafür wurde mit einer geeigneten Beeinflussung der Synthese im VHDL-Kode geschaffen. Die Hardware-Vervielfältigung wurde hier durch explizites Nicht-Ausrollen der Iterationsschleifen so gesteuert, daß ein im Flächenbedarf minimiertes und durch Verkürzung der Signalwege schnelleres Design erstellt werden konnte.

Die direkte skript-basierte Steuerung für das Scheduling im *Synopsys Behavioral Compiler* ermöglichte nicht nur die gezielte Korrektur und Behebung von Bibliotheks- und Werkzeuginkompatibilitäten, sondern erbrachte auch eine weitere Verbesserung sowohl der Taktrate als auch des Flächenverbrauchs.

Die Erstellung einer flexiblen Testumgebung für alle Entwurfsebenen.

Das o. g. Verifikationsverfahren für die Optimierungen an dem Algorithmus und dem Datenformat, wurde so ausgebaut, daß die Simulation mit dem ursprünglichen C-Kode, dem optimal quantisierenden C++ Prototypen und den VHDL-Modellen in jeweils derselben Umgebung möglich ist. In der Testumgebung kann auf allen Entwurfseben der komplette Standard-ETSI-Sprachdatensatz verwendet werden, was für die Verhaltenssimulation des VHDL-Modells auch vollständig durchgeführt wurde. Die Simulation der Register-Transfer- und Layout-Netzlisten mußte sich aufgrund deren Umfanges und der daraus resultierenden langen Simulationslaufzeiten auf Stichproben beschränken.

Die Spezifikation und Entwicklung des Prototypensystems.

Ein abschließender Bestandteil der Arbeit befaßte sich mit der Integration der Hardware-Designs auf einem Prototypensystem. Das Kernstück bildet dabei eine kommerzielle, mit einem *Xilinx Virtex*-FPGA *XCV800* ausgestattete, PCI-Karte für einen Standard-PC. Der FPGA-Entwurf des Perzeptionsmodells wurde hierfür durch Schnittstellen-Designs zum Datenaustausch mit einer Host-Applikation und eine eigens entwickelte externe AD/DA-Wandler-Baugruppe ergänzt. Das ursprüngliche in C/C++ implementierte Perzeptionsmodell, dessen globale Überarbeitung für eine softwaretechnische Einbindung in Applikationen und die Hardware-Realisierung der Filterbank des Perzeptionsmodells einschließlich der Zusammenführung der Teilentwürfe auf dem FPGA wurde von den Projektpartnern eingebracht.

Bewertung und Ausblick

Für die Hardware-Implementierung auditorischer Verarbeitungsalgorithmen existieren heute Ansätze unter Verwendung von mechano-optischen, analogen elektronischen und digitalen Technologien. Die hochintegrierten analogen Schaltungssysteme können durch die inhärente Parallelität solche Stufen wie Basilarmembranfilterung, Adaptation und Signalwandlung bzw. -kodierung in den Haarzellen und die ersten Stufen des Hörnervs weitgehend abbilden. Diese Analog-VLSI Schaltungen, wie sie von [Lyon und Mead 1988], [Mead 1989], [Sarpeshkar et al. 1998] und [van Schaik et al. 1996a] entworfen und realisiert wurden, stellen jedoch höchste Ansprüche an geringe Fertigungstoleranzen und die Umgebungsbedingungen insbesondere an die Temperatur. Dadurch sind sie nur in wenigen, diesbezüglich toleranten Anwendungen einsetzbar. Programmierbare Lösungen für spezielle Prozessoren (DSP) aber auch Hardware-Realisierungen wurden bislang - bedingt durch die hohe Komplexität der Algorithmen - meist nur für Teilsysteme realisiert. Ausnahmen bilden hier stark reduzierte Ansätze z. B. von [Ahn

und Westerkamp 1990] und [Meyer-Bäse et al. 1997a]. In umfangreichen Modellansätzen, wie z. B. in [Baumgarte 2000], mit denen die Modellierung bestimmter Aspekte des Gehörs untersucht werden soll, ist dagegen die Echtzeitverarbeitung oft nicht das primäre Ziel. Bei der Durchführung von Simulationen wird dabei eine vielfach langsamere Verarbeitung in Kauf genommen. Die in dieser Dissertation vorgestellten Ergebnisse gehen dagegen insbesondere über die digitalen Hardware-Entwürfe bzw. Vorstufen von [Schwarz 1993], [Meyer-Bäse et al. 1997b], [Temple et al. 1996], [Lim et al. 1997b] und [Jones et al. 1998] insofern hinaus, daß nun ein vollständiges Modell in praktisch einsetzbare Schaltungen umgesetzt werden konnte. Vor allem ist hier der Aspekt der Stereoverarbeitung im binauralen Modell zu betonen, der für integrierte Lösungen eine Ausnahme darstellt.

Die inzwischen verfügbaren PC-Prozessoren mit Taktraten im Gigahertz-Bereich sind ebenfalls in der Lage, die Berechnungen des Perzeptionsmodells in Echtzeit auszuführen. Trotzdem ist die hohe Systemleistung einer PC-CPU mit den hier beschriebenen Arbeitsergebnissen nicht zu vergleichen. Diese Prozessoren sind hinsichtlich ihrer Funktionalität überdimensioniert und können aufgrund der Bauform und des Leistungsverbrauchs in eingebetteten Systemen, wie z. B. digitalen Hörgeräten, nicht eingesetzt werden. Die im Rahmen dieser Dissertation entwickelte Entwurfs- und Optimierungstechnik für die Hardware-Implementation, die beliebige Reprogrammierbarkeit der verwendeten FPGA-Bausteine und das aufgebaute Prototypensystem insgesamt erlauben dagegen die Weiterentwicklung der Hardware-Entwürfe, die Anpassung an neue Applikationen und die Spezifikation bzw. Tests nachfolgender Standardzell-Designs (ASIC) direkt in den gewünschten Applikationen.

Für die Anwendung in miniaturisierten Signalverarbeitungssystemen, z. B. in digitalen Hörgeräten, kommt als wesentlicher Aspekt der minimale Stromverbrauch hinzu. Dafür könnten in fortführenden Arbeiten alternative (z. B. bitserielle oder hochparallele) Architekturen für fortgeschrittene Halbleitertechnologien implementiert und getestet werden. Die hier entwickelte und für den *Synopsys Behavioral Compiler* optimierte VHDL-Beschreibung der Nachregelschleifen ist dafür bereits vorbereitet. Insbesondere die Verlustleistungsoptimierung, die durch den Übergang von Variablen-Rechenwerken zu Konstanten-Operationen herbeigeführt werden kann, wird durch das steuerbare „Ausrollen“ und das teilweise Parallelisieren der Schleifeniterationen unterstützt. Diese Ansätze sind u. a. Forschungsgegenstand eines ebenfalls DFG-geförderten Nachfolgeprojekts zur Verlustleistungsreduzierung von audiosignalverarbeitenden Algorithmen[86].

Der hier erarbeitete Entwurf für den Nachregelschleifenteil des Perzeptionsmodells konnte auf Verhaltens-, Register-Transfer- und Layoutebene korrekt simuliert werden; er verbraucht im verwendeten *Xilinx Virtex*-FPGA *XCV800* nur 19 % der Gatterressourcen (äquivalenter Gatterzahl: 80.000) bzw. 10 % des Onchip-RAM-Speichers und erzielt eine Taktfrequenz von 40 MHz. Mit einer Gesamtverzögerung von 3,5 µs je Abtastwert werden die für eine Echtzeit-

[86] "Power Reduction for Digital Audio Processing" (PRO-DASP, Me 1289/6), ein Projekt im Rahmen des DFG-Schwerpunktprogramms "Grundlagen und Verfahren verlustarmer Informationsverarbeitung" (VIVA).

Datenverarbeitung maximal zulässigen 4,096 µs deutlich unterschritten. Diese Optimierung der benötigten Logikressourcen ermöglicht auch die Integration der anderen Teildesigns des Perzeptionsmodells auf einem einzigen FPGA-Baustein. Somit konnte im Zusammenhang mit der effizienten Struktur und den Eigenschaften des Perzeptionsmodells, der hier erarbeiteten Verfahren zur Algorithmusminimierung bzw. Hardware-Umsetzung und unter Einbeziehung der anderen Projektkomponenten ein flexibles Koprozessorsystem aufgebaut werden. Mit der Integration des Koprozessorsystems in eine PC-basierte Arbeitsumgebung steht damit ein vollständiges und echtzeitfähiges Audiosignalverarbeitungssystem zur psychoakustischen Modellierung, Spracherkennung, Sprachgütebewertung und Entwicklung von Hörgerätealgorithmen zur Verfügung.

Literaturverzeichnis

AHN, S.-G. UND WESTERKAMP, J. J. 1990. Cochlear modeling using a general purpose digital signal processor. In *Proceedings of the IEEE 1990 National Aerospace and Electronics Conference. NAECON*. Vol. 1. IEEE, Dayton, OH, USA, 57–63.

ANDRAKA, R. J. 1993. FIR filter fits in a FPGA using a bit-serial approach. In *In Third Annual PLD Design Conference and Exhibition*.

ANDRAKA, R. J. 1996. Building a high performance bit-serial processor in an FPGA. In *Proceedings of Design Supercon 1996*. 5.1 – 5.21.

ANDRAKA, R. J. 2000. Multiplikation in FPGAs. Technical Report, Andraka Consulting Group, Inc., http://www.andraka.com/multipli.htm#Wallace Trees.

BAUMGARTE, F. 2000. Ein psychophysisches Gehörmodell zur Nachbildung von Wahrnehmungsschwellen für die Audiocodierung. Ph.D. thesis, Universität Hannover, Hannover, Germany.

BERENDS, J. G. UND STEMERDINK, J. A. 1994. A perceptual speech quality measure based on a psychoacoustic sound perception. *J. Audio Eng. Soc. 42*, 3, 115–123.

BIRBAUMER, N. UND SCHMIDT, R. F. 1990. *Biologische Psychologie*. Springer, Berlin-Heidelberg-New York.

BITTERLING, J. 1999. FPGA-Redesign eines Signalverarbeitungsalgorithmus zur gehörgerechten Sprachvorverarbeitung. Studienarbeit, Universität Hamburg.

BODDEN, M. 1993. Modeling human sound-source localization and the cocktail-party-effect. *acta acustica*, 43–55.

BOLLEROTT, M., DESPANG, H. G., KLUGE, W., UND SCHWARZ, A. 1996. Software model of the natural cochlea. *Acustica / Acta Acustica 82*, 1 (January), 102–113.

BRUCKE, M., NEBEL, W., SCHWARZ, A., MERTSCHING, B., HANSEN, M., UND KOLLMEIER, B. 1999. Silicon cochlea: A digital VLSI implementation of a quantitative model of the auditory system. In *Journal of the Acoust. Soc. of Am.: 137th Regular Meeting of the Acoust. Soc. of Am. and Forum Acousticum 99 integrating the 25th German Acoustics DAGA Conf.* Acoust. Soc. of Am., European Acoustics Assoc., TU Berlin, Germany, 1192.

BRUCKE, M., NEBEL, W., SCHWARZ, A., MERTSCHING, B., TCHORZ, J., UND KOLLMEIER, B. 1998. Digital VLSI-implementation of a psychoacoustical and physiologically motivated speech preprocessor. In *Proc. of the NATO Advanced Study Institute on Computational Hearing*, S. Greenberg und M. Slaney, Eds. NATO Advanced Study Institute, Il Ciocco, Italy, 157–162.

COHEN, A., SHAMMA, S., INDIVERI, G., HORIUCHI, T., DOUGLAS, R., KOCH, C., UND SEJNOWSKI, T. 1999. Workshop on neuromorphic engineering. Technical Report, National Science Foundation, Telluride, CO, http://zig.ini.unizh.ch/telluride99/report99/. June-July.

DAU, T. 1999. Modell der effektiven Signalverarbeitung im Gehör. *Einblicke, Carl von Ossietzky Universität Oldenburg* 29, 16–18.

DAU, T., PÜSCHEL, D., UND KOHLRAUSCH, A. 1996a. A quantitative model of the "effective" signal processing in the auditory system. I. model structure. *Journal of the Acoustical Society of America (JASA) 99,* 6, 3615–3622.

DAU, T., PÜSCHEL, D., UND KOHLRAUSCH, A. 1996b. A quantitative model of the "effective" signal processing in the auditory system. II. simulations and measurements. *Journal of the Acoustical Society of America (JASA) 99,* 6, 3623–3631.

DRAKE, L. A., RUTLEDGE, J. C., UND COHEN, J. 1993. Wavelet analysis in recruitment of loudness compensation. *IEEE Transactions on Signal Processing 41,* 12 (December), 3306–3312.

ERCEGOVAC, M. D. UND LANG, T. 1994. *Division and Square Root, Digit-Recurrence Algorithms and Implementations*. Kluver Academic Publishers, Boston, Dordrecht, London.

ETSI. 1991. TM/TM5/TCH-HS global analysis of selection tests: Basic data. Technical Report 91/74, ETSI.

ETSI. 1992. TM/TM5/TCH-HS selection test phase II: Listening test results with german speech samples, Experiment 1, IM4. Technical Report 92/35, FI/DBP-Telekom.

EVANS, E. F. 1982. Functional anatomy of the auditory system. In *The Senses*, 1st ed., H. B. Barlow und J. D. Mollon, Eds. Cambridge University Press, Cambridge, London, New York, Chapter 14, 251–306.

FLETCHER, H. UND MUNSON, W. A. 1933. Loudness, its definition, measurement, and calculation. *Journal of the Acoustical Society of America (JASA) 5*, 82– 91.

FRANCIS, I. F. UND ANDERSON, T. R. 1997. Binaural phonem recognition using the auditory image model and cross-correlation. In *Proccedings of the International Conference On Acoustics, speech, and signal processing*. 1231–1234.

GOLDBERG, D. 1991. What every computer scientist should know about floating-point arithmetic. *Computing Surveys*, 153–230. Association for Computing Machinery, Inc., reprinted by permission.

GOSSE, K., DE SAINT-MARTIN, F. M., DUROT, X., DUHAMEL, P., UND RAULT, J. B. 1997. Subband audio coding with synthesis filters minimizing a perceptual criterion. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*. Vol. 1. 347–350.

HANSEN, M. 1998. Assessment and prediction of speechtransmission quality with an auditory processing model. Ph.D. thesis, University of Oldenburg, Oldenburg, Germany.

HANSEN, M. UND KOLLMEIER, B. 1995. Anwendbarkeit eines psychoakustisch motivierten Sprachvorverarbeitungsmodells für die Sprachqualitätsmodellierung. Elektronische Sprachsignalverarbeitung, ITA, TU Dresden.

HANSEN, M. UND KOLLMEIER, B. 1998a. Kontinuierliche Beurteilung von Sprach(übertragungs)qualität: Messung und Modellierung. In *Fortschritte der Akustik - DAGA 98*. DAGA 98, Zürich; DEGA, Oldenburg, 394–395.

HANSEN, M. UND KOLLMEIER, B. 1998b. Continous assesment and modelling of speech transmission quality. In *Proccedings ICA/ASA*. ICA/ASA, Seattle, USA, 229–230.

HARTMANN, C. 2000. Gehörgerechte Signalverarbeitung für die Spracherkennung auf Basis von Wortuntereinheiten. M.S. thesis, Carl von Ossietzky Universität Oldenburg, Oldenburg, Germany.

HAUENSTEIN, M. 1997. Psychoakustisch motivierte Maße zur instrumentellen Sprachgütebeurteilung. Arbeiten über digitale Signalverarbeitung 10, LNS, Christian-Albrechts-Universität Kiel, Shaker Verlag GmbH, Aachen.

HELLBRÜCK, J. 1993. *Hören, Physiologie, Psychologie und Pathologie*. Hogrefe, Verlag für Psychologie, Göttingen, Germany.

HENNESSY, J. L. UND PATTERSON, D. A. 1996. *Computer Architecture - A Quantitative Approach*. Morgan Kaufmann Publishers, Inc., San Francisco, California, United States.

HEWITT, M. J. UND MEDDIS, R. 1994. A computer model of amplitude-modultaion sensitivity of single units in the inferior colliculus. *Journal of the Acoustical Society of America (JASA) 95,* 4 (April), 2145–2159.

HOHMANN, V. UND HANSEN, M. 1997. Implementation der Gammatone Filterbank. Interner Bericht. AG MEDI, Universität Oldenburg.

HOWARD, D. M., TYRRELL, A. M., UND BROOKES, T. 1998. Transputer-based human hearing modelling. *Simulation practice and theory 6,* 5, 479–491.

IEEE. 1985. *IEEE Standard 754-1985 for binary Floating-point Arithmetic*. IEEE.

IRINO, T. UND PATTERSON, R. D. 1997. A time-domain, level-dependent auditory filter: The gammachirp. *Journal of the Acoustical Society of America (JASA) 101*, 412–419.

JONES, S., LIM, S.-C., UND TEMPLE, A. R. 1998. Digital hardware implementation of a neuromorphic pitch extraction system. WWW, http://lboro.ac.uk/departments/el/research/sys/presentations/auditory/stril6%.pdf.

JONES, S., MEDDIS, R., LIM, S.-C., UND TEMPLE, A. R. 2000. Toward a digital neuromorphic pitch extraction system. *IEEE Transactions on Neural Networks 11,* 4 (July), 978–987.

KATES, J. M. 1991. A time-domain digital cochlea model. *IEEE Transaction on Signal Processing 39,* 12, 2573–2592.

KATES, J. M. 1993. Accurate tuning curves in a cochlea model. *IEEE Transaction on Speech and Audio Processing 1,* 4, 453–462.

KELLY, J. P. 1991. Hearing. In *Principles of neural science*, 3rd ed., E. R. Kandel, J. H. Schwartz, und T. M. Jessel, Eds. Prentice-Hall International Inc., Chapter "V/ 32", 481–499.

KIM, D.-S., LEE, S.-Y., UND KIL, R. M. 1999. Auditory processing of speech signals for robust speech recognition in real-world noisy environment. *IEEE Transactions on Speech and Audio Processing 7,* 1 (January).

KLEINSCHMIDT, M. 1999. Störgeräuschunterdrückung und gehörgerechte Vorverarbeitung für die automatische Spracherkennung. M.S. thesis, Carl von Ossietzky Universität Oldenburg.

KLEINSCHMIDT, M., TCHORZ, J., WITTKOPP, T., UND HOHMANN, V. 1998. Robuste Spracherkennung durch binaurale Richtungsfilterung und gehörgerechte Vorverarbeitung. In *Fortschritte der Akustik - DAGA 98*. DAGA 98, Zürich; DEGA, Oldenburg, 396–397.

KNAPP, D. W. 1996. *Behavioral Synthesis*. Prentice Hall PTR, Upper Saddle River, NJ, United States.

KOLLMEIER, B. 1998. Auditorische Signalverarbeitung und Wahrnehmung bei Innenohrschwerhörigkeit, Vorkolloquium. In *Fortschritte der Akustik - DAGA*. DAGA 98, Zürich; DEGA, Oldenburg, 42–45.

KOLLMEIER, B. 1999. Detailed teaching notes for graduate courses in acoustics: Audiologie. WWW; http://www.physik.uni-oldenburg.de/Docs/medi/download/docs/lehre/ index.html.

KOLLMEIER, B., DERLETH, R.-P., UND HOHMANN, V. 1998. Modeling the effective auditory signal processing for hearing-impaired listeners. In *Psychophysical and Physiological Advances in Hearing*, A. R. Palmer, A. Rees, A. Q. Summerfield, und R. Meddis, Eds. Whurr Publisher, London, 482–490.

KUMAR, N. UND ANDREOU, A. G. 1997. Auditory feature extraction using self-timed continous-time discrete signal procesign circuits. *IEEE Transactions on circuits and systems-II: Analog and digital signal processing 44,* 9 (september), 723–728.

KUMAR, N., HIMMELBAUER, W., CAUWENBERGHS, G., UND ANDREOU, A. G. 1997. An analog VLSI architecture for auditory based feature extraction. In *Proccedings of the International Conference On Acoustics, speech, and signal processing*. Vol. 5. ICASSP, Munich, Germany, 4081–4084.

KURUP, P. UND ABBASI, T. 1995. *Logic Synthesis Using Synopsys*. 2. Kluwer Academic Publishers, Kluwer Academic Publishers, Norwell, MA, USA.

LAZARRO, J. P. UND MEAD, C. 1989. Circuit models of sensory transduction in the cochlea. In *Analog VLSI Implementations of Neural Networks*. Kluwer, Norwell, MA, United States.

LEHMANN, G., WUNDER, B., UND SELZ, M. 1994. *Schaltungsdesign mit VHDL*. Franzis', Franzis Verlag, Poing, Austria.

LIM, S. C., TEMPLE, A. R., UND JONES, S. R. 1997a. VHDL based design of biologically inspired pitch detection system. In *Proceedings of 1997 IEEE International Conference on Neural Network*. Housten, Texas, 922–927.

LIM, S. C., TEMPLE, A. R., UND JONES, S. R. 1997b. Digital hardware implementation of a neuromorphic pitch extraction system. In *1st European Workshop on Neuromorphic Systems*. Stirling, Scotland.

LYON, R. F. UND MEAD, C. 1988. An analog electronic cochlea. *IEEE Transactions on Acoustics, Speech, and Signal Processing 36*, 7, 1119–1134.

MALLAT, S. G. 1989. A theory for multiresolutional signal decomposition: The wavelet representation. *IEEE Trans. Patt. Anal. Mach. Intell. 11*, 674–693.

MEAD, C. 1989. *Analog VLSI and neural systems*. Addison-Wesley.

MEAD, C. A., ARREGUIT, X., UND LAZZARO, J. 1991. Analog VLSI model of binaural hearing. *IEEE Transactions on neural Networks 2*, 2 (march), 230–236.

MEDDIS, R. 1986. Simulation of mechanical to neural transduction in the auditory receptor. *Journal of the Acoustical Society of America (JASA) 79*, 3, 702–711.

MEDDIS, R. 1988. Simulation of auditory-neural transduction: Further studies. *Journal of the Acoustical Society of America (JASA) 83*, 1056–1063.

MEYER-BÄSE, U., MELLOT, J., UND TAYLOR, F. 1997a. Design of RNS frequency sampling filter banks. In *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing*. Vol. 3. Munich, Germany, 2061–2064.

MEYER-BÄSE, U., MEYER-BÄSE, A., UND SCHEICH, H. 1997b. An auditory neuron model for cochlea implants. In *Proceedings of the Aerosense 97 *SPIE**. Vol. 3077. Orlando, 582–593.

MOORE, B. C. J. UND GLASBERG, B. R. 1987. Formulae describing frequency selectivity as a function of frequency and level and their use in calculating excitation patterns. *Hearing Research 28*, 209–225.

MURDOCCA, M. 1997. *Principles of Computer Architecture*. Miles Murdocca and Miletus Research, M. Murdocca (Rutgers University, New Brunswick, NJ, USA), Miletus Research (Jackson, NJ, USA), http://www.miletus.com.

OEKEN, F. W. 1984. *Hals- Nasen- Ohren Heilkunde*, 4. ¸berarbeitete Auflage ed. VEB Verlag Volk und Gesundheit, Berlin, German Demokratic Republic.

O'LEARY, J., ZHAO, X., GERTH, R., UND SEGER, C.-J. H. 1999. Formally verifying IEEE compliance of floating-point hardware. *Intel Technology Journal*.

PATTERSON, R. D., ALLERHAND, M., UND GIGUERE, C. 1995. Time-domain modelling of peripheral auditory processing: A modular architecture and a software platform. *Journal of the Acoustical Society of America (JASA) 98*, 1890–1894.

PATTERSON, R. D., NIMMO-SMITH, J., HOLDSWORTH, J., UND RICE, P. 1987. An efficient auditory filterbank based on the gammatone function. Technical Report, Paper presented at a meeting of the IOC Speech Group on Auditory Modelling at RSRE.

PATTERSON, R. D., ROBINSON, K., HOLDSWORTH, J., MCKEOWN, D., ZHANG, C., UND ALLERHAND, M. 1992. Advances in biosiences. In *Auditory Physiology and Perception*, Y. Cazals, L. Demany, und K. Horner, Eds. Pergamon Press, 429–443.

PHILIPS HEARING INSTRUMENTS. 2001. How do we hear. Werbe- und Produktbroschüre, http://www.hearing.philips.com/how.htm.

PÜSCHEL, D. 1988. Prinzipien der zeitlichen Analyse beim Hören. Ph.D. thesis, Universität Göttingen, Göttingen, Germany.

SARPESHKAR, R., LYON, R. F., UND MEAD, C. 1998. A low-power wide-dynamic-range analog VLSI cochlea. *Analog Integrated Circuits and Signal Processing 16,* 3 (August), 245–274.

SCHWARZ, A. 1993. Softwaremodell der natürlichen Cochlea für ein Cochlea-Implant-System. M.S. thesis, Technische Universität Dresden, Dresden, Germany.

SCHWARZ, A., MERTSCHING, B., BRUCKE, M., NEBEL, W., HANSEN, M., UND KOLLMEIER, B. 1999a. Silicon cochlea: a digital VLSI implementation of psychoacoustically and physiologically motivated speech preprocessor. In *Psychophysics, Physiology and Models of Hearing*, T. Dau, V. Hohmann, und B. Kollmeier, Eds. University Oldenburg, Germany, World Scientific, Singapore, 245–248.

SCHWARZ, A., MERTSCHING, B., BRUCKE, M., NEBEL, W., HANSEN, M., UND KOLLMEIER, B. 1999b. Implementing a quantitative model for the 'effective' signal processing in the auditory system on a dedicated VLSI hardware. In *Proc. of the 25th EUROMICRO Conf.* EUROMICRO, Milan, Italy, 133–139.

SENEFF, S. 1986. A computational model for the peripheral auditory system: application to speech recognition research. In *Proccedings of the International Conference On Acoustics, speech, and signal processing*. ICASSP, Tokyo, Japan, 37.8.1–37.8.4.

SENEFF, S. 1988. A joint synchrony/mean-rate model of auditory speech processing. *Journal of Phonetics* 16, 55– 76.

SHAMMA, S., COHEN, A., HORIUCHI, T., INDIVERI, G., DOUGLAS, R., KOCH, C., UND SEJNOWSKI, T. 1998. Workshop on neuromorphic engineering. Technical Report, National Science Foundation, Telluride, CO, http://zig.ini.unizh.ch/telluride98/report98/. June-July.

SHIRAZI, N., WALTERS, A., UND ATHANAS, P. 1995. Quantitative analysis of floating point arithmetic on FPGA based custom computing machines. Technical Report, Virginia Polytechnic Institute and State University, Blacksburg, Virginia, United States.

SLANEY, M. 1993. An efficient implementation of the patterson-holdsworth auditory filterbank. Technical Report 35, Apple Computer Inc., Perception Group - Advanced Technology Group, http://www.slaney.org/malcolm/pubs.html #AuditoryModeling.

SOLBACH, L., WÖHRMANN, R., UND KLIEWER, J. 1998. *Readings in Computational Auditory Scene Analysis*. Erlbaum Publishers.

SPANIOL, O. 1976. *Arithmetik in Rechenanlagen*, Teubner Studienbücher Informatik ed. B. G. Teubner, Stuttgart, Germany.

STEVENS, S. S. 1975. *Psychophysics. Introduction to its perceptual, neural, and social prospects*. Wiley, New York.

Synopsys Inc. 1996. *Synopsys DesignWare Foundation Library Online Databook*. Synopsys Inc., http://www.synopsys.com.

Synopsys Inc. 1999. *Synopsys DesignWare Foundation Library Online Databook*. Synopsys Inc., http://www.synopsys.com.

TANAKA, K., ABE, M., UND ANDO, S. 1998. A novel mechanical cochlea "fishbone" with dual sensor/actuator charakteristics. *IEEE/ASME Transactions on Mechatronics 3,* 2 (June).

TCHORZ, J., KASPER, K., REINIGER, H., UND KOLLMEIER, B. 1997. On the interplay between auditory-based features and locally recurrent neural networks. In *Proc. Eurospeech 1997, ESCA*. Eurospeech, ESCA, Patras, Greece, 2075–2078.

TCHORZ, J., WESSELKAMP, M., UND KOLLMEIER, B. 1996. Gehörgerechte Merkmalsextraktion zur robusten Spracherkennung in Störgeräuschen. In *Fortschritte der Akustik-DAGA 96*. Oldenburg, Germany, 532–533.

TEMPLE, A. R., LIM, S. C., UND JONES, S. R. 1996. Digital implementation of a stellate cell using the macgregor model. In *Proceedings of Workshop on Design Methodologies for Signal Processing*. Zakopane Poland, 111–117.

TEMPLE, A. R., LIM, S. C., UND JONES, S. R. 1997. Digital realisation of the mammalian hair cell. In *Proc. of MicroNeuro'97*. MicroNeuro, Dresden, Germany, 119–122.

TONG, B. 1999. New area saving architecture: Ripple Carry Select. Technical Report 4(2), Synopsys Inc. Q2/Q3.

TYRRELL, A. M., HOWARD, D. M., UND BEASLEY, N. A. 1992. Transputer model of the human peripheral hearing system. *Microprocessing and Microprogramming 35,* 619–624.

TYRRELL, A. M., HOWARD, D. M., UND BROOKES, T. S. 2000. Real-time transputer-based model of human peripheral hearing. http://www.york.ac.uk/depts/elec/resrev/sub7/sub717.htm.

VAN SCHAIK, A., FRAGNIËRE, E., UND VITTOZ, E. 1996a. Improved silicon cochlea using compatible lateral bipolar transistors. In *Advances in Neural Information Processing Systems 8*, D. Touretzky, et al., Ed. Number 8. MIT press, Cambridge MA, 671–677.

VAN SCHAIK, A., FRAGNIËRE, E., UND VITTOZ, E. 1996b. An analogue electronic model of ventral cochlear nucleus neurons. In *Proceedings of the Fifth International Conference on Microelectronics for Neural Networks and Fuzzy Systems*. MicroNeuro, IEEE Computer Society Press, Los Alamitos CA, 52–59.

VAN SCHAIK, A., FRAGNIËRE, E., UND VITTOZ, E. 1997a. A silicon model of amplitude modulation detection in the auditory brainstem. In *Advances in Neural Information Processing Systems 9*, M.C. Mozer, et al., Ed. Number 9. MIT Press, Cambridge MA, 741–747.

VAN SCHAIK, A., FRAGNIËRE, E., VITTOZ, E., UND MEDDIS, R. 1997b. Analogue vlsi building blocks for an electronic auditory pathway. In *Proceedings of the 11th International Symposium on Hearing*. Grantham, UK, 157–163.

VARY, P., HEUTE, U., UND HESS, W. 1998. *Digitale Sprachsignalverarbeitung*. Teubner, Stuttgart, Germany.

VON BEKESY, G. 1928. *Physik Z.* No. 29, 793–810.

VON BEKESY, G. 1953. Description of some mechanical properties of the organ of corti. *Journal of the Acoustical Society of America (JASA) 25*, 770–785.

VON BEKESY, G. 1960. *Experiments in Hearing*. McGraw-Hill, New York.

WALKER, R. UND THOMAS, D. 1985. A model of design representation and synthesis. In *22nd Design Automation Conference*. Las Vegas.

WANG, J., SARPESHKAR, R., JABRI, M., UND MEAD, C. 1997. A low power analog front-end module for cochlear implants. In *Proceedings of the XVI World Congress on Otorhinolaryncholog, Head and Neck Surgery - cochlear implants*. Sydney, Australia,http://www.pcmp.caltech.edu/anaprose/rahul/cochlea/implant.ps.

WATTS, L., KERNS, D., LYON, R., UND MEAD, C. 1992. Improved implementation of the silicon cochlea. *IEEE Journal of Solid-state Circuits 27*, 5 (May), 692–700.

WESTERMAN, L. UND SMITH, R. 1984. Rapid and short-term adaption in auditory nerve responses. *Hearing Research*, 249–260.

WÜST, H., KASPER, K., UND REININGER, H. 1998. Hybrid number representation for the FPGA-realization of a versatile neuro-processor. In *Proc. EUROMICRO98*. EUROMICRO, Västeras, Sweden, 694–701.

Xilinx Inc. 1999. *Virtex Series FPGAs*. Xilinx Inc., http://www.xilinx.com/products/virtex.htm.

ZHOU, G., BINTZ, L., ANDERSON, D. Z., UND BRIGHT, K. E. 1993. A life-sized physical model of the human cochlea with optical holographic readout. *Journal of the Acoustical Society of America (JASA)*, 1516–1523.

ZÖLZER, U. 1996. *Digitale Audiosignalverarbeitung*. B. G. Teubner, Stuttgart, Germany.

ZWICKER, E. 1982. *Psychoakustik*. Springer-Verlag, Berlin, Heidelberg, New York.

ZWICKER, E. 1986. A hardware cochlear nonlinear preprocessing model with active feedback. *Journal of the Acoustical Society of America (JASA) 80,* 1, 146–152.

ZWISLOCKI, J. J. 1948. Theorie der Schneckenmechanik: Qualitative und quantitative Analyse. *Acta Otolaryngol. Suppl. 72*, 1–76.

Abkürzungsverzeichnis

Die in den Kapiteln verwendeten Abkürzungen, zuerst erwähnt und erläutert auf Seite

Anhang

Syntheseskript für Synopsys Behavioral Compiler

Nachfolgend werden die für die Generierung des synthetisier- und simulierbaren Behavioral-Compiler-Designs der Nachregelschleifen einschließlich des Steuerautomaten erforderlichen Compiler-Anweisungen in Skriptform aufgeführt. Damit soll sowohl der prinzipielle als auch versionsspezifische praktische Entwurfsfluß dokumentiert werden. Nicht enthalten ist die Erzeugung des RAM-Designs und die Generierung der entsprechenden Bibliotheken. Für detaillierte Erläuterungen der Anweisungen und Optionen sei auf die Synopsys-Dokumentation bzw. entsprechende Literatur verwiesen [Kurup und Abbasi 1995, Knapp 1996].

```
/******** Voreinstellungen*/
report_lib xfpga_virtex-4
remove_design -all
sh rm -rf $HOME/synopsys_cache_1999.10/xdw_virtex/xdw_mult*
suppress_errors = {"CHR-003", "RTDC-36", "RTDC-7"}


/********** Vergabe von Namen für Designs und Pfade, Taktfestlegung*/
TOP = bc_floatcore
XILPATH = ./xilinx20/
DBPATH = ./db20/
RPTPATH = ./rpt20/
CLOCKPERIOD = 20
CLOCK_NAME  = SlClk
RESET_NAME  = SlnReset
edifout_design_name = TOP
designer = "AG IMA"
company  = "Universitaet Hamburg"
part     = "XCV800-4-BG432"

/* Vergabe von Design-Namen für diverse Multiplizierer*/
XDWMULSS = mfrac_mul_20x18_xdw_mult_ss_20_18_0
XDWMULSS1 = mfrac_mul1_20x18_xdw_mult_ss_20_18_0
XDWMULUU = xfloat14_6_mul_xdw_mult_uu_16_16_0


/******* Einstellung der Pfade für das Synthesewerkzeug*/
synthetic_library = {xdw_virtex.sldb, standard.sldb, dw01.sldb, dw02.sldb, dw03.sldb, dw04.sldb, dw05.sldb,
dw06.sldb, dw07.sldb}
link_library = {"*", "xfpga_virtex-4.db"} + synthetic_library
define_design_lib ramlib -path BCXIL + "/ram/ramlib"
search_path=search_path + { BCXIL + /ram }
synthetic_library=synthetic_library + ramlib.sldb


/********** Xilinx-Multiplizierer (xdw_mult) vorbereiten, vorab kompilieren */
/* Xilinx Virtex-Multiplizierer haben nur beschränkte Wortbreiten und werden aufgrund von Systemfehlern nicht korrekt eingebunden, deswegen hier in der benötigten Wortlänge vorab kompiliert und als Synopsys-Designfile gespeichert, siehe auch Kommando „report_synlib xdw_virtex“ */

elaborate xdw_mult_ss_bt -arch "binary_tree" -lib XDW_VIRTEX -update -param "A_width = 20, B_width = 18"
set_wire_load_model -name "xcv800-4_avg" -library "xfpga_virtex-4" xdw_mult_ss_bt
compile  -map_effort medium
rename_design xdw_mult_ss_bt XDWMULSS
write -format db -hierarchy -output DBPATH + XDWMULSS + ".db"
rename_design XDWMULSS XDWMULSS1
write -format db -hierarchy -output DBPATH + XDWMULSS1 + ".db"
```

```
remove_design -all
elaborate xdw_mult_uu_bt -arch "binary_tree" -lib XDW_VIRTEX -update -param "A_width = 14, B_width = 14"
set_wire_load_model -name "xcv800-4_avg" -library "xfpga_virtex-4" xdw_mult_uu_bt
compile  -map_effort medium
rename_design xdw_mult_uu_bt XDWMULUU
write -format db -hierarchy -output DBPATH + XDWMULUU + ".db"
remove_design -all
```

/******** Allgemeine Einstellungen für den *Behavioral Compiler*/**

```
bc_enable_analysis_info = true
```

/******** Analyse aller VHDL-Quelldateien (parsing) */**

```
analyze -format vhdl -lib WORK {"../xfloat/xfloat6_6.vhd" "../xfloat/xfloat14_6.vhd"}
analyze -format vhdl -lib WORK {"../util/our_package.vhd"}
analyze -f vhdl TOP + ".vhd"
```

/******** Elaborieren des synthetischen Designs für das Scheduling*/**

```
elaborate -schedule TOP
write -hier -output DBPATH + TOP + "_elab.db"
bc_report_memories -used_memories  > RPTPATH + TOP + "_elabmem.rpt"

/*Abbruch, falls Elaborieren nicht erfolgreich*/
if (dc_shell_status != 1) {
  quit
  }
```

/******** Speichern der synthetischen Operatoren (*synopsys preserved_functions*)*/**

```
/* Fließkomma-Operatoren*/
foreach(xfloat_op, find(design, "*xfloat*")) {
        current_design xfloat_op
        write -format db -hierarchy -output DBPATH + xfloat_op + ".db"
}
/* Festkomma-Operatoren*/
foreach(frac_op, find(design, "*frac_*")) {
        current_design frac_op
        write -format db -hierarchy -output DBPATH + frac_op + ".db"
}
/* Umwandlung der Operandenbreite*/
foreach(resize_op, find(design, "*resize*")) {
        current_design resize_op
        write -format db -hierarchy -output DBPATH + resize_op + ".db"
}
/*Umwandlung von Fließ- nach Festkomma*/
current_design find(design, "*conv_fixed*")
write -format db -hierarchy -output DBPATH + conv_fixed + ".db"
```

/******** Spezielle Behavioral Compiler - Einstellungen */**

```
current_design TOP
bc_fsm_coding_style = binary
bc_use_fsm_compiler = false
/*bc_enable_chaining = false*/
/*bc_chain_read_into_oper = false*/
/*bc_chain_read_into_mem = false*/
bc_estimate_timing_effort = high
bc_estimate_mux_input = "8"
bc_estimate_mux_delay = true /* true wenn bc_estimate_timing_effort = high (default)*/
bc_estimate_control_chaining_delay = true /* true wenn bc_estimate_timing_effort = high (default)*/
```

```
bc_dont_ungroup -fsm -register -random_logic -prio_logic
register_control -output
/*set_behavioral_reset -fsm -async -fsm -port RESET_NAME -active low*/
auto_wire_load_selection = false
set_wire_load_model -name "xcv800-4_avg" -library "xfpga_virtex-4"


/********** Vorab-Kompilieren der arithmetischen Operatoren (synopsys preserved_functions)*/
foreach(xfloat_op, find(design, "*xfloat*")) {
        current_design xfloat_op
        auto_wire_load_selection = false
        set_wire_load_model -name "xcv800-4_avg" -library "xfpga_virtex-4" xfloat_op
        set_max_area 0.0
        compile -map med
        set_wire_load_model -name "xcv800-4_avg" -library "xfpga_virtex-4" xfloat_op
        set_dont_touch xfloat_op
}
foreach(frac_op, find(design, "*frac_*")) {
        current_design frac_op
        auto_wire_load_selection = false
        set_wire_load_model -name "xcv800-4_avg" -library "xfpga_virtex-4" frac_op
        compile -map med
        set_wire_load_model -name "xcv800-4_avg" -library "xfpga_virtex-4" frac_op
        set_dont_touch frac_op
}


/********** Rückersetzen der Xilinx Virtex-Multiplizierer*/
/* Da das bc_time_design die Xilinx Virtex-Multiplizierer fehlerhafterweise nicht analysieren kann, werden die im
vorhergehenden Schritt entstandenen Multiplizierer entfernt und die vorsorglich (oben) erzeugten rückersetzt */

current_design TOP
write -hier -o DBPATH + TOP + "_elab_precomp.db"

remove_design XDWMULSS
remove_design XDWMULSS1
remove_design XDWMULUU

read -format db DBPATH + XDWMULSS + ".db"
read -format db DBPATH + XDWMULSS1 + ".db"
read -format db DBPATH + XDWMULUU + ".db"

current_design TOP
link


/********** Design-Timing-Analyse für Behavioral Compiler*/
create_clock CLOCK_NAME -p CLOCKPERIOD
/*bc_set_margin 11.2*/
bc_time_design /*-force -fastest*/
write -hier -o DBPATH + TOP + "_timed.db"


/********** Scheduling vorbereiten und einstellen*/
FUNCTIONNAME = "xfloat14_6_mul"
/*Erzwingen, daß der Fließkomma-Multiplizierer nur einfach instanziiert und mehrfach verwendet wird */
set_common_resource { find(cell, "*"+ FUNCTIONNAME +"*", -h) } -max_count 1 -force_sharing

/* Explizite Zeit-Constraints zwischen Punkten im Design setzen */
current_design TOP
current_instance
current_instance main/reset_loop/work_loop/channel_loop
```

```
set_min_cycles 1 -from find(cell, "*lpsum*") -to find(cell, "*MEM_WRITE_cell_ram*")
set_min_cycles 1 -from find(cell, "*MEM_WRITE_cell_ram*") -to find(cell, "*SlvNrglOut*")
set_min_cycles 1 -from_end find(cell, "*steploop*") -to find(cell, "*VsScaleIn*")

current_instance steploop
set_min_cycles 2 -from find(cell, "*MEM_READ_cell_ram*") -to find(cell, "VxDivisor*")
set_min_cycles 1 -from find(cell, "*lpout*") -to find(cell, "*MEM_WRITE_cell_ram*")
set_min_cycles 2 -from find(cell, "MEM_WRITE_cell_ram*") -to find(cell, "*ramincr*")

current_instance
current_design TOP


/********* Kontrollausgabe der Scheduling-Einstellungen*/
list bc_fsm_coding_style
report_scheduling_constraints


/********* Setzen der Zeitreserve in ns bei der Delay-Bewertung von Design-Pfaden*/
bc_set_margin 9


/********* Scheduling */
/* Scheduling im superstate-I/O modus, Abbruch wenn nicht erfolgreich*/
schedule -io_mode superstate -effort medium
  if (dc_shell_status != 1) {
  quit
  }


/********* Export des Ergebnisses und diverser Reports */
write -hier -o DBPATH + TOP + "_sched.db"
report_schedule -var -op -summ -a > RPTPATH + TOP + "_sched_rep.rpt"
report_constraints -all -verb > RPTPATH + TOP + "_sched_constr.rpt"
report_multicycles > RPTPATH + TOP + "_sched_multicyc.rpt"


/********* Export des RT-VHDL-Codes zur Simulation mit synthetischen Arrays als RAM-Speicher*/
/* A) Da Synopsys den eigenen Divisions-Operator fehlerhafterweise nicht auflösen kann, bleibt die Division im von
Synopsys erzeugten Register-Transfer-VHDL-Code einfach leer und muß manuell in einem separaten VHDL-File
bereitgestellt werden. Zur Simulation müssen noch die Simulationsbibliotheken der Xilinx Virtex-Technologie
(UniSim) mit einem Unix-Shell-Script scr/(insert_unisim.sh) im RT-VHDL-Code eingefügt werden*/

/* B) Die im VHDL-Code der Verhaltensbeschreibung als preserved_function deklarierten arithmetischen Operato-
ren werden für das Scheduling kompiliert. Damit existieren im Design initialisierte FPGA-Look-Up-Table-Zellen mit
der Programmierinformation. Designs auf Gatter-Ebene (nach compile) kann Synopsys nur im Interchange-Format
EDIF exportieren. Der VHDL-Export solcher FPGA-Designs wird weder vom Behavioral Compiler noch vom FPGA-
Compiler unterstützt, das exportierte Register-Transfer-VHDL-File enthält nur leere Look-Up-Tables und ist damit
nicht simulierbar. Deswegen müssen die synthetischen Operatoren aus der „Elaborate“-Stufe rückersetzt werden*/


/* aktuelle Designs löschen*/
current_design TOP
foreach (des, find(design, *xfloat*)){
        current_design des
        remove_design des
}
foreach (des, find(design, *frac*)){
        current_design des
        remove_design des
}
```

```
foreach (des, find(design, *resize*)){
        current_design des
        remove_design des
}
remove_design find(design, *conv_fixed*)

/*vorab elaborierte Designs laden*/
read -f db DBPATH + "frac_add_20.db"
read -f db DBPATH + "mfrac_mul_20x18.db"
read -f db DBPATH + "mfrac_mul1_20x18.db"
read -f db DBPATH + "conv_xfloat_frac6_6.db"
read -f db DBPATH + "xfloat14_6_add.db"
read -f db DBPATH + "xfloat14_6_mul.db"
read -f db DBPATH + "xfloat14_6_sub.db"
read -f db DBPATH + "xfloat6_6_div.db"
read -f db DBPATH + "resize14_6_to_6_6.db"
read -f db DBPATH + "resize6_6_to_14_6.db"
read -f db DBPATH + "conv_fixed.db"

current_design TOP
link

/* weitere Export-Einstellungen und Export der RT-VHDL-Netzliste */
vhdlout_levelize=true
vhdlout_debug_mode=true
vhdlout_use_packages = {IEEE.std_logic_1164, IEEE.std_logic_arith, IEEE.std_logic_signed}
vhdlout_architecture_name="RT_sched"
write -hier -f vhdl -out DBPATH + TOP + "_rt_sched_temp.vhd"

/* und nun muß noch die Division extern von einem UNIX-Textersetzungsskript nachersetzt werden, s. o.*/
```

/******** Export des Register-Transfer-VHDL-Codes für die Simulation mit echten Xilinx-RAM-Speichern*/**

```
/* Hierfür werden die RAM-Zellen vom Rumpfdesign getrennt, und in dem neuen Sub-Design durch echte Zellen
ersetzt. Damit muß für die Simulation ein neues Top-Level-Design erstellt werden, das Rumpf- und RAM-Design
wieder zusammenfügt*/

/*Aufräumen und Laden des Schedule-Designs */
remove_design -designs
read -f db DBPATH + TOP + "_sched.db"

/******** Gruppieren, Separieren und Ersetzen der RAM-Zellen */
current_design TOP
find (cell, "seq_cell*")
seq_cel_list = dc_shell_status
group seq_cel_list -design_name ramcells
current_design ramcells
replace_synthetic
write -hier -o DBPATH + TOP + "_sched_ram.db"
remove_design "ramcells*"

/******** Gruppieren und Separieren des Rumpf-Designs*/
current_design TOP
filter (find(cell, "*"), "@ref_name == ramcells")
ram_list = dc_shell_status
find cell
all_list = dc_shell_status
body_cel_list = all_list - ram_list
group body_cel_list -design_name TOP + "_body"
current_design TOP + "_body"
write -hier -o DBPATH + TOP + "_sched_body.db"
```

/******** Rückersetzen der rein synthetischen Blöcke*/**
/* Da der *FPGA-Compiler* keine kompilierten FPGA-Look-Up-Table-Zellen exportieren kann (s.o.), müssen die synthetischen Operatoren aus der „Elaborate“-Stufe wiederum rückersetzt werden*/

```
remove_design -designs
read -f db DBPATH + TOP + "_sched_body.db"
current_design TOP + "_body"
foreach (des, find(design, *xfloat*)){
        current_design des
        remove_design des
}
foreach (des, find(design, *frac*)){
        current_design des
        remove_design des
}
foreach (des, find(design, *resize*)){
        current_design des
        remove_design des
}
remove_design find(design, *conv_fixed*)

read -f db DBPATH + "frac_add_20.db"
read -f db DBPATH + "mfrac_mul_20x18.db"
read -f db DBPATH + "mfrac_mul1_20x18.db"
read -f db DBPATH + "conv_xfloat_frac6_6.db"
read -f db DBPATH + "xfloat14_6_add.db"
read -f db DBPATH + "xfloat14_6_mul.db"
read -f db DBPATH + "xfloat14_6_sub.db"
read -f db DBPATH + "xfloat6_6_div.db"
read -f db DBPATH + "resize14_6_to_6_6.db"
read -f db DBPATH + "resize6_6_to_14_6.db"
read -f db DBPATH + "conv_fixed.db"

current_design TOP + "_body"
link
write -hier -o DBPATH + TOP + "_rt_sched_body.db"
read -f db DBPATH + TOP + "_sched_ram.db"
```

/******** Export des RT-VHDL-Codes mit den echten Xilinx-RAM-Speichern*/**

```
/*Export des Rumpfes*/
current_design TOP + "_body"
vhdlout_levelize=true
vhdlout_use_packages = {IEEE.std_logic_1164, IEEE.std_logic_arith, IEEE.std_logic_signed}
vhdlout_architecture_name="RT_sched_body"
write -hier -f vhdl -out DBPATH + TOP + "_rt_sched_body_temp.vhd"

/*Export des RAMs*/
current_design ramcells
vhdlout_levelize=false
vhdlout_use_packages = {IEEE.std_logic_1164, IEEE.std_logic_arith, IEEE.std_logic_signed}
vhdlout_architecture_name="RT_sched_ram"
write -hier -f vhdl -out DBPATH + TOP + "_rt_sched_ram_temp.vhd"
```

/******** Ende*/**

```
quit
```

Publikationen

SCHWARZ, A., MERTSCHING, B., BRUCKE, M., NEBEL, W., HANSEN, M., UND KOLLMEIER, B. 1999a. Silicon cochlea: a digital VLSI implementation of psychoacoustically and physiologically motivated speech preprocessor. In *Psychophysics, Physiology and Models of Hearing*, T. Dau, V. Hohmann, und B. Kollmeier, Eds. University Oldenburg, Germany, World Scientific, Singapore, 245–248.

SCHWARZ, A., MERTSCHING, B., BRUCKE, M., NEBEL, W., HANSEN, M., UND KOLLMEIER, B. 1999b. Implementing a quantitative model for the 'effective' signal processing in the auditory system on a dedicated VLSI hardware. In *Proc. of the 25th EUROMICRO Conf.* EUROMICRO, Milan, Italy, 133–139.

BOLLEROTT, M., DESPANG, H. G., KLUGE, W., UND SCHWARZ, A. 1996. Software model of the natural cochlea. *Acustica / Acta Acustica 82*, 1 (January), 102–113.

BRUCKE, M., NEBEL, W., SCHWARZ, A., MERTSCHING, B., HANSEN, M., UND KOLLMEIER, B. 1999. Silicon cochlea: A digital VLSI implementation of a quantitative model of the auditory system. In *Journal of the Acoust. Soc. of Am.: 137th Regular Meeting of the Acoust. Soc. of Am. and Forum Acousticum 99 integrating the 25th German Acoustics DAGA Conf.* Acoust. Soc. of Am., European Acoustics Assoc., TU Berlin, Germany, 1192.

BRUCKE, M., NEBEL, W., SCHWARZ, A., MERTSCHING, B., TCHORZ, J., UND KOLLMEIER, B. 1998. Digital VLSI-implementation of a psychoacoustical and physiologically motivated speech preprocessor. In *Proc. of the NATO Advanced Study Institute on Computational Hearing*, S. Greenberg und M. Slaney, Eds. NATO Advanced Study Institute, Il Ciocco, Italy, 157–162.

Mitbetreute Studienarbeiten:

Ole Blaurock	"Entwurf und Aufbau einer Microcontroller-Karte für ein ASIC-Inbetriebnahmesystem" (1997)
Johannes Bitterling	"Design eines FPGAs für einen Algorithmus zur gehörgerechten Sprachvorverarbeitung" (1999)
Michael Scholz	"Entwurf und Implementierung von Datenschnittstellen und Integration eines kombinierten AD/DA-Wandlers für das "Silicon Ear"-System" (2000)

Mitbetreute Diplomarbeiten:

Nikolaus Voss	"Entwurf und Implementation einer durch parallele Hardware beschleunigten Gaborfilterbank" (2000)