Ein konfigurierbarer ASIC für nachrichtentechnische Anwendungen

Diplomarbeit

Universität Hamburg-Fachbereich Informatik
Arbeitsbereich Technische Grundlagen der Informatik

Sebastian Wallner
Pinneberg

Dezember 1999

Inhaltsverzeichnis

1 Einleitung 1
1.1 Motivation 4
1.2 Stellung des ASICs in einem nachrichtentechnischen System 5

2 Grundlagen 6
2.1 Signale 6
2.2 Darstellung digitaler Signale 7
2.3 Nyquist-Abtasttheorem 9
2.4 Analytische Signale 10
2.5 Unterabtastung um einen ganzahligen Faktor 12
2.6 Abtasttheorem für Bandpaßsignale 14
2.6.1 Abtastung reeller Bandpaßsignale 15
2.6.2 Abtastung komplexer Bandpaßsignale 18
2.6.3 Zusammenfassung 19
2.7 Versatz von Signalen auf der Frequenzachse 20
2.8 Digitale Filter 23
2.8.1 Einleitung 23
2.8.2 Grundlagen digitaler Filter 24
2.9 Eigenschaften nichtrekursiver Digitalfilter (FIR-Filter) 30
2.9.1 Struktur und Filterfunktionen 30
2.9.2 Impulsantwort 30
2.9.3 Phasengang und Gruppenlaufzeit 30
2.10 Unerwünschte Effekte bei Digitalfiltern 31
2.10.1 Zahlendarstellung 31
2.10.2 Wertquantisierung 32
2.10.3 Überläufe 32
2.10.4 Quantisierung der Filterkoeffizienten 33
2.11 Digitale Direkt Synthese (DDS) 33
2.12 Hilbertfilter und Quadraturmischung 35

3 Anforderungen und Lösungsansätze 37
3.1 Wesentliche Anforderungen an den ASIC 37
3.1.1 Grundsätzliche Designentscheidungen 38
3.2 Untersuchungen digitaler Filter in Hardware 38
3.2.1 FIR Filter durch direkte Umsetzung der Faltungssumme 38
3.3 Spezielle FIR Filterrealisierungen 42
3.4 Ergebnisse und Bewertung realisierter FIR Filterstrukturen 45
3.5 Zusammenfassung 49
3.6 Realisierung eines DDS Generators in Hardware 50

3.7 Berechnung der Anzahl der Tabelleneinträge 52
3.7.1 Fehlerrechnung ohne Tabelleninterpolation 52
3.7.2 Fehlerrechnung mit Tabelleninterpolation 53
3.8 Implementationsmöglichkeiten und Ergebnisse 54
3.9 Untersuchung und Bewertung der erzeugten Signale 56

4 Übersicht über den ASIC 58
4.1 Zusammenfassung 64

5 Realisierung und Implementation 65
5.1 Aufbau und Einbindung der Multiratenfilterkaskade 65
5.2 Realisierung der komplexwertigen Hardwarestruktur 70
5.3 Zusammenwirken der Multiratenfilterkaskade und der komplexwertigen Hardwarestruktur 72
5.3.1 Beispiel einer Ablaufsteuerung mit Hilfe der FSM 73
5.4 Registerbelegung und Programmierung 75
5.5 Timingkonzept des ASICs 78
5.5.1 Taktung und Ablaufsteuerung 79
5.6 Überleitung 79

6 Einsatz von VHDL zur Entwicklung des ASICs 80
6.1 Allgemeines Entwurfsvorgehen mit VHDL 80
6.1.1 Top-Down Entwurf 80
6.1.2 Simulation 82
6.1.3 Synthese 85
6.2 Überlegungen bei der Entwicklung von Verhaltensmodellen 86
6.2.1 Vorgehen bei der Entwicklung des ASICs 86
6.2.2 Allgemeine Realisierungsabschätzungen 88
6.2.3 Simulation der Verhaltensmodelle 88

7 Bewertung und Ausblick 90
7.1 Ergebnisse 90
7.1.1 Geschwindigkeit 90
7.1.2 Fläche 91
7.1.3 Zusammenfassung und Bewertung 92
7.2 Aufgetretene Probleme 95
7.3 Vergleich mit anderen Chips 97
7.4 Mögliche Erweiterungen und Verbesserungen mit Ausblick 98
7.4.1 Maßnahmen zur Durchsatzsteigerung 99
7.4.2 Zeitmultiplex Teilrealisierungen zur Flächeneinsparung 99
7.4.3 Shannon Hardwareinterpolator 100

Danksagung 101

Literaturhinweis 102

Anhang **104**
Übersicht über die Register des ASICs 104
Quellcodes der VHDL Verhaltensmodelle 106

Kapitel 1

Einleitung

In der Nachrichtentechnik werden immer mehr digitale Verfahren eingesetzt. Sie bieten deutliche Vorteile im Bezug auf Elimination der Einflüsse von Fertigungstoleranzen, Alterung und Zufallsprozessen wie etwa Rauschen. Auch ist durch digitale Verfahren vieles besser und eleganter zu lösen und handzuhaben, was die Einsatzmöglichkeiten der digitalen Technik rapide ansteigen läßt.
Statt kontinuierlicher (analoger) Größen wie elektrische Spannungen und Ströme, werden diskrete Zahlenfolgen verarbeitet. Durch die Digitaltechnik kann eine höhere Qualität der Signalverarbeitung und damit eine höhere Wirtschaftlichkeit erreicht werden, die sich aus folgenden Punkten ergibt:

- vereinfachte Produktion,
- einfache Wartung,
- Zuverlässigkeit,
- Stabilität und
- Programmierbarkeit.

Nachteilig ist der hohe Schaltungsaufwand, der jedoch, angesichts des zunehmenden Integrationsgrades digitaler Schaltungen, immer weniger ins Gewicht fällt.

Entscheidet man sich aufgrund der oben genannten Vorteile für eine digitale Verarbeitung nachrichtentechnischer Signale, bei der die Geschwindigkeit im Vordergrund steht, so ist man auf Spezialhardware angewiesen.
Die heutigen zur Verfügung stehenden *digitalen Signalprozessoren* (DSP), stellen die benötigte Verarbeitungsleistung, wie sie beispielsweise bei der schnellen digitalen Datenübertragung mit großen Bandbreiten benötigt werden, noch nicht zur Verfügung. Dieses ist unter anderem darauf zurückzuführen, dass diese Spezialprozessoren universell einsetzbar sein sollen und die Signalverarbeitung sequentiell mit Hilfe eines Programms durchgeführt wird. Trotzdem bieten sie Leistungen, die in bestimmten Bereichen von konventionellen Mikroprozessoren nicht erreicht werden können.
Aufgrund der Einsatzgebiete digitaler Signalprozessoren wurden spezielle Befehle implementiert, über die ein konventioneller Mikroprozessor in der Regel nicht verfügt. Der insbesondere bei digitalen Signalverarbeitungsaufgaben benötigte „multiply and accumulate (MAC)“ Befehl, der in einem Takt multiplizieren und addieren kann, ist hier ein typisches Beispiel. Er ist in dem Befehlssatz eines jeden digitalen Signalprozessors implementiert. Ferner verfügen die meisten digitalen Signalprozessoren über Mechanismen, Schleifen direkt in Hardware auszuführen.
Zusätzlich zur Implementation spezieller Befehle konnte durch die Trennung von Programm und Daten (Harvardarchitektur) eine Anpassung des Rechenwerkes an typische Signal-

verarbeitungsoperationen durchgeführt werden. Durch Verbesserung des Datenzugriffs mit Hilfe lokaler Speicher und mehrerer Bussysteme konnte die Signalverarbeitungsleistung gegenüber konventionellen Mikroprozessoren erheblich gesteigert werden.
Mehrere Pipelinestufen und komplexe Befehle, die direkt in Hardware ausgeführt werden, reichen allerdings bei weitem noch nicht aus, vielen geschwindigkeitsrelevanten Anforderungen gerecht zu werden.

Die Vorteile einer Spezialhardware ist nicht nur eine höhere Leistungsfähigkeit gegenüber einer Problemlösung, die in Software mit einem genügend leistungsfähigen Universalrechnersystem oder DSP arbeitet, sondern auch eine geringere Leistungsaufnahme. Zudem spielt die Größe eines Universalrechnersystemes im Bezug auf die Gesamtkosten eines Systems eine entscheidende Rolle. In naher Zukunft haben nur kleine leistungsfähige Systeme eine Chance, die speziellen Preisvorstellungen entsprechen.
Als ein Nachteil einer Spezialhardware kann mangelnde Flexibilität angesehen werden. Sie deckt oft nur die Aufgaben ab, für die sie speziell konstruiert wurde. Die vielfältigen Möglichkeiten, die ein programmierbarer Signalprozessor bietet, kann und soll eine Spezialhardware auch nicht zur Verfügung stellen. Aufgrund dieser fehlenden Flexibilität spezieller Hardware, ist sie im allgemeinen auf langsamere Einheiten aufgesetzt, die schließlich Endverarbeitungen oder Datenentscheidungen durchführen.

Ziel dieser Diplomarbeit ist es ein ASIC (Application Specific Integrated Circuit) für die digitale Signalverarbeitung, überwiegend für nachrichtentechnische Anwendungen wie der schnellen digitalen Datenübertragung oder der Überwachungstechnik zur entwickeln, bei der die Geschwindigkeit und Konfigurierbarkeit im Vordergrund steht. Bei der Entwicklung sollte zudem auf eine voll skalierbare Architektur geachtet werden.
Die Anwendungsgebiete des ASICs finden sich dabei vornehmlich in Bereichen, in denen Bandpaßsignale in eine äquivalente Basisbandlage transformiert und weiter verarbeitet werden müssen. Sie können dann von konventioneller und langsamerer Hardware weiterverarbeitet werden. Konkrete Einsatzgebiete wären Modulations- bzw. Demodulationsaufgaben mit hohen Bandbreiten oder in der Multiträgersignalverarbeitung zu sehen.

Dem zu entwickelnden ASIC liegen drei Hauptgedanken zu Grunde:

- *Multiratensignalverarbeitung mit variablen Datenraten.*
  Die internen Datenverarbeitungsraten der zu entwerfenden Hardware sollten zur Entlastung der arithmetischen Einheiten jeweils so niedrig wie möglich gehalten werden. Zu diesem Zweck bietet sich eine Verarbeitung mit variablen Taktraten an (Multiratenverarbeitung).

- *Analytische Verarbeitung von Signalen, bei der nicht auf Spiegelfrequenzen geachtet werden muß.*
  Eine reelle Verarbeitung von Signalen ist aufgrund der Berücksichtigung von Spiegelfrequenzen nicht geeignet. Besser ist es, das reelle Eingangssignal vorab in ein analytisches Signal zu wandeln und dieses dann weiter zu verarbeiten.
  Als direkter Vorteil erweist sich die Entschärfung des Abtasttheorems für komplexe Bandpaßsignale. Wie noch dargestellt wird, bedarf es allerdings eines Mehraufwandes bei der zu entwerfenden Architektur, die analytische Verarbeitung von Signalen durchzuführen.

- *Rekonfiguration in jedem Sampletakt.*
  Die Konfiguration und Steuerung der internen Abläufe soll von einem endlichen Steuerautomaten FSM (Finite State Machine) durchgeführt werden. Da er synchron zu der je-

weiligen Samplerate arbeitet, ist eine Umkonfiguration der Teilkomponenten in jedem Sampletakt möglich. Aus dem Einsatz des endlichen Steuerautomaten soll die hohe Flexibilität des ASICs resultieren.

Eine einfache Schnittstelle erlaubt die Kommunikation mit einem digitalen Signalprozessor, einem Mikrocontroller oder einem anderen Hostrechner.

Der heutige Markt bietet eine begrenzte Anzahl von Chips mit ähnlichen Eigenschaften. Sie sind allerdings in der Regel nur bedingt konfigurierbar und damit nur für spezielle Anwendungen geeignet [HAR94] [QUA98].
Der hier zu entwickelnde ASIC soll allgemeiner einsetzbar sein und eine Reihe von Eigenschaften bieten, die in kommerziellen Produkten nicht verfügbar sind. Dazu zählt auch eine hohe Konfigurierbarkeit und die daraus resultierende Flexibilität.
Zur Hardwarerealisierung dieses Vorhabens fanden Entwicklungstools der Firmen *Synopsys* und *Cadence* Verwendung. Als Hardwarebeschreibungssprache wurde *VHDL* „Very High Speed Integreated Circuit Hardware Description Language“ eingesetzt.

Diese Diplomarbeit beginnt mit einigen signaltheoretischen und nachrichtentechnischen Grundlagen. Die Abtastung und Unterabtastung reeller und die Einführung der komplexer Signale und der daraus resultierenden Vorteile in der digitalen Signalverarbeitung werden behandelt. Auf die Vorteile, speziell der Unterabtastung komplexer Signale, wird in Kapitel 4 eingegangen.
Da digitale Filter und zur Signalgenerierung benötigte DDS Generatoren (Digitale Direkt Synthese) in dieser Diplomarbeit eine zentrale Rolle spielen, sollen die in diesem Zusammenhang benötigten Grundlagen genauer betrachtet werden.
Zum Abschluß dieses Kapitels werden zwei Wandlerstrukturen näher besprochen, mit denen reelle in komplexe Signale gewandelt werden können.

Im Kapitel 3 werden allgemeine Untersuchungen für schnelle digitale Filterstrukturen und zur Signalerzeugung geeignete DDS Generatoren in Hardware vorgestellt. Bei den Filterstrukturen werden ausschließlich Filter im Zeitbereich betrachtet. Ein Überblick über vier untersuchte FIR Filter- und DDS Hardwarestrukturen mit einer jeweiligen Bewertung schließen dieses Kapitel ab. Die geeignetsten Teilrealisierungen sollen später in eine ASIC Gesamtrealisierung implementiert werden.

Kapitel 4 beinhaltet eine Zusammenfassung des Gesamtchips. Es wird hier auf die einzelnen Funktionsblöcke eingegangen. Der DDS Generator, eine Multiratenfilterkaskade, der endliche Steuerautomat (FSM) zur Steuerung der internen Abläufe des ASICs und die komplexwertige Hardwarestruktur werden dabei näher betrachtet. Die ASIC Gesamtarchitektur wird durch ein Blockschaltbild modelliert und dargestellt. Im darauffolgenden Kapitel soll dann genauer auf die Funktion und das Zusammenspiel der Einzelblöcke eingegangen werden soll.

In Kapitel 5 werden die im Zuge dieser Arbeit für die ASIC Gesamtrealisierung als geeignet befundenen Teilkomponenten noch einmal genauer erörtert. Dabei wird gezielt auf die Einbindung und das Zusammenspiel der einzelnen Komponenten eingegangen. Des weiteren wird eine Steuerregisterbelegung definiert, mit der die Teilkomponenten von dem implementierten Steuerautomaten speziell für eine Aufgabe konfiguriert werden können. Ein Beispiel eines Steuerautomaten-Mikrocodes zur Verdeutlichung der sich aus dem Einsatz eines endlichen Automaten und der realisierten Hardwarestruktur ergebenden Flexibilität und das hierfür benötigte Timingkonzept zur Ablaufsteuerung schließen dieses Kapitel ab.

Auf die eigentliche Realisierung in VHDL soll in Kapitel 6 eingegangen werden. Dabei soll zuerst der grundsätzliche Vorgang eines Hardwareentwurfs in VHDL erläutert werden. Es folgen einige Vorabschätzungen im Bezug auf Größe und Taktrate des ASICs.
Einen zentralen Punkt soll hier zudem die Simulation und Synthese der Einzelkomponenten und des gesamten ASICs darstellen.

Kapitel 7 diskutiert einige aufgetretene Probleme, die sich insbesondere im Zusammenhang mit den verwendeten Werkzeugen ergeben haben. Es werden Abschätzungen des verwendeten Synthesetools im Bezug auf Geschwindigkeit und Fläche, sowie endgültige Ergebnisse des ASIC-Layouts gegeben. Den Abschluß soll eine Gegenüberstellung mit einem konventionell erhältlichen ASIC der Firma *Harris Semiconductor* bilden. Eine Bewertung und einige Ausblicke und Verbesserungen, insbesondere im Bezug auf eine geeignete Interpolationshardware, runden dieses Kapitel ab.

1.1 Motivation

Speziell in der digitalen Signalverarbeitung gilt es, große Datenmengen, z.B. von einem schnellen A/D (Analog-Digital) Wandler in Echtzeit, d.h. zwischen jedem Eintreffen neuer Samplewerte, zu verarbeiten. Für den Audiobereich können konventionelle digitale Signalprozessoren (DSP) eingesetzt werden, da die anfallenden Datenraten in Bereichen liegen, die von ihnen noch verarbeitet werden können. Die für den Audiobereich maximale verwendete Samplerate beträgt 48 Khz. Für eine Verarbeitung von Datenraten, die beispielsweise bei der schnellen digitalen Datenübertragung anfallen, reichen Lösungen mit einem digitalen Signalprozessor nicht mehr aus. DSPs können höchstens dort eingesetzt werden, wo die Datenraten schon reduziert wurden, der eigentliche Informationsgehalt jedoch erhalten geblieben ist.
Abhilfe kann hier nur eine dedizierte Hardwarerealisierung bringen.
In vielen Fällen wird diese Spezialhardware einen digitalen Signalprozessor zwar nicht ersetzen, jedoch unterstützen.
Ein DSP wird aufgrund seiner freien Programmierbarkeit immer vielfältiger sein als eine für spezielle Anwendungen entwickelte Hardware.
Eine direkte Umsetzung eines sequentiellen Programmes in Hardware läßt nach bisherigen Erfahrungen einen Faktor von > 10 zu, da einerseits die Interpretation von Prozessorbefehlen entfällt, andererseits in Hardware oft eine hochparallele Realisierung möglich ist.
Mit dem zu entwickelnde ASIC soll versucht werden, Signalverarbeitungsalgorithmen intern hoch parallel und damit schneller ablaufen zu lassen, als es mit einem konventionellen digitalen Signalprozessor möglich wäre. Nur durch eine hohe Parallelität und die zum Einsatz kommende Multiratensignalverarbeitung, ist eine direkte Verarbeitung hochfrequenter Signale möglich.
Die genauen Anforderungen an den zu realisierenden ASIC sind in Kapitel 4 zusammengefaßt.

1.2 Stellung des ASICs in einem nachrichtentechnischen System

Wie bereits dargestellt, soll der ASIC Verarbeitungs- oder Vorverarbeitungsaufgaben im Bereich der digitalen Nachrichtentechnik übernehmen, z.B. digital anfallende Datenraten soweit zu reduzieren, dass die eigentlich zu übertragenden Informationen nicht verloren gehen und andere langsamere Einheiten diese auswerten können. Die Stellung des ASICs ist aufgrund dessen ziemlich weit vorne, innerhalb eines digitalen nachrichtentechnischen Systems anzusiedeln.
Abbildung 1.1 zeigt anhand eines Blockschaltbildes und einigen optionalen Komponenten einen möglichen Datenfluß.
Da der ASIC ausschließlich digitale Signale verarbeiten kann, muß ein Eingangssignal zuerst mit Hilfe eines geeigneten schnellen Analog/Digital Wandlers (A/D Wandler) in eine diskrete Zahlenfolge gewandelt werden.
Nach einer Teilverarbeitung im ASIC und einer in den meisten Fällen durchgeführten Sampleratenreduktion (Downsampling), können die digitalen Daten durch eine aufgesetzte Hardware weiterverarbeitet werden.
Als aufgesetzte Hardware kann beispielsweise ein Personal Computer (PC), ein Digitaler Signalprozessor (DSP) oder auch ein Mikrocontroller (MC) eingesetzt werden.
Für eine analoge Ausgabe müssen sie zur Rückwandlung in ein analoges Signal geeignete Digital/Analog Wandler (D/A Wandler) zur Verfügung stellen.
Zumindest der ASIC arbeitet in Echtzeit, das heißt mit der vollen Samplerate, mit der auch der A/D Wandler arbeitet. Er stellt keine Speicher für digitale Daten zur Verfügung.
Der zu entwerfende ASIC soll ausschließlich digitale Daten verarbeiten und ist somit in eine rein digitale Umgebung einzuordnen.

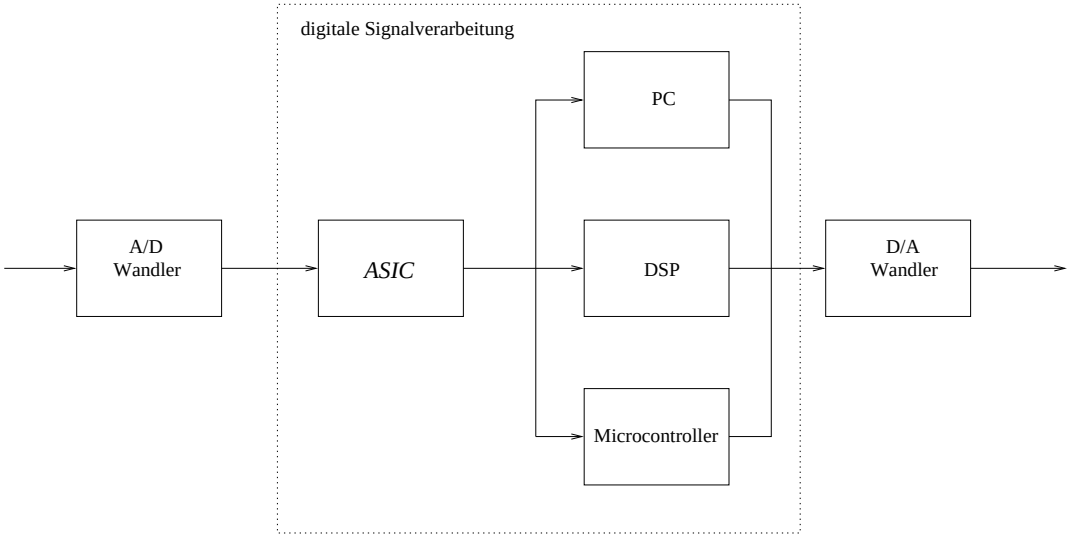

Abbildung 1.1: Stellung des ASICs in einem nachrichtentechnischen System

Kapitel 2

Grundlagen

In diesem Kapitel sollen einige Grundlagen der digitalen Signalverarbeitung besprochen werden, die für das weitere Verstehen der zu entwerfenden Teilkomponenten und damit des ASICs unabdingbar sind.

2.1 Signale

Allgemein kann der Begriff des *Signals* wie folgt definiert werden [LÜK95]:

Definition 2.1: *Ein Signal ist die physikalische Darstellung von Nachrichten oder Daten.*

Ein Signal kann zeitabhängig sein, oder aber darstellbar als Funktion einer oder mehrerer Veränderlicher. Es sollen zuerst nur *eindimensionale* analoge Signale betrachtet werden. Eindimensionale Signale liegen meist als Zeitfunktionen vor und sollen als *x(t)* bezeichnet werden. Der Wert *x(t)* stellt die *Augenblicksamplitude,* also den *Momentanwert* oder anders gesagt den Wert eines Signals zu einem bestimmten Zeitpunkt *t* dar.
Beispiele sind:
Akustische Signale aller Art (Musik, Geräusche), biologische oder physikalische Signale wie elektromagnetische oder seismische Schwingungen [HAM87].
Einen wichtigen Fall mehrdimensionaler *Signale,* die in Abschnitt 2.3 dieser Arbeit noch eingeführt werden, stellt das komplexe Signal dar [KRO97].

Auf die Nachrichtentechnik bezogen, ist ein Signal prinzipiell eine analoge elektromagnetische Schwingung mit einem Informationsgehalt I, die von einem Sender emittiert und von einem entsprechend räumlich getrennten Empfänger empfangen wird. Der Empfänger hat die Aufgabe, die Information I wieder zu extrahieren.
Wird im weiteren Verlauf der Diplomarbeit von Signal gesprochen, so ist im allgemeinen ein digitales Signal gemeint.

2.2 Darstellung digitaler Signale

Die Darstellung eines *digitalen Signals* kann allgemein in Form einer Zahlenfolge erfolgen:

$$x(t) \rightarrow \{x_t\} . \tag{2.1}$$

Die *Umsetzung* (Digitalisierung) eines analogen Signals erfolgt in der Regel mittels eines A/D Wandlers. Das digitale Signal liegt dann *diskret* in der Zeit *(t)* und Amplitude *(x)* vor.
Allgemein spricht man in diesem Zusammenhang von einer *Diskretisierung* der Zeit und einer *Quantisierung* des Eingangswertes (Amplitudenquantisierung) [HES93].
Im folgenden soll zunächst getrennt auf die Zeitdiskretisierung und Amplitudenquantisierung eingegangen werden.

Durch die Diskretisierung der Zeit wird eine *zeitdiskrete Abtastung* vorgenommen, die ein Signal in eine Folge von diskreten *Abtastwerten* zerlegt:

$$x(t) \rightarrow \{... x(t_1), x(t_2), ...\} . \tag{2.2}$$

Als zusätzliche Vereinbarung soll gelten, dass die Abtastung in *äquidistanten* Schritten erfolgt.

$$X(n) = x(nT): \quad n = -\infty \ldots -1, 0, 1, \ldots \infty \qquad n \in Z \tag{2.3}$$

n bezeichnet hierbei den *Index* oder Adresse (Meßwertadresse) des Abtastwertes und *T* das *Abtastintervall*.

Wegen der endlichen Wortlänge bei Verarbeitung und Speicherung abgetasteter Signale muß eine Quantisierung in der Amplitude vorgenommen werden.
Bei einer binären Festkommadarstellung beispielsweise ergibt sich die Anzahl der unterscheidbaren Amplitudenstufen N aus:

$$N = 2^W \tag{2.4}$$

wobei W der Wortlänge ihn Bit entspricht.
Wird eine Gleitkommadarstellung eingesetzt, so kann ein wesentlich größerer Dynamikbereich auf Kosten einer komplizierteren Arithmetikrealisierung abgedeckt werden [HES93] [TIE93]. Aus Aufwandsgründen sollen in dieser Arbeit ausschließlich Festkommadarstellungen Verwendung finden.
Durch die begrenzte Auflösung des A/D Wandlers ergibt sich zwangsläufig auch ein Quantisierungsfehler, der in jede nachfolgende signalverarbeitende Operation mit einbezogen wird [ZÖL97]. Dieser Fehler soll in dieser Arbeit keine weitere Betrachtung finden.

Erst die Diskretisierung in der Zeit und die Quantisierung des Wertes machen aus einer Funktion der Zeit ein digitales Signal. Die Probleme der Abtastung in der Zeit spielen in der Signalverarbeitung eine bedeutendere Rolle, als die der Quantisierung der Amplitude. Eine Erhöhung der Wortbreite ist im allgemeinen mit weniger Aufwand verbunden als die Erhöhung der Abtastfrequenz [ZÖL97].

Zur bildlichen Veranschaulichung der Diskretisierung und Quantisierung ist in Abbildung 2.1 a) ein zeitkontinuirliches (analoges) Signal zu sehen, in Abbildung b) eine diskrete Abtastung in der Zeit und in c) eine diskrete Abtastung in Zeit und Quantisierung der Amplitude.

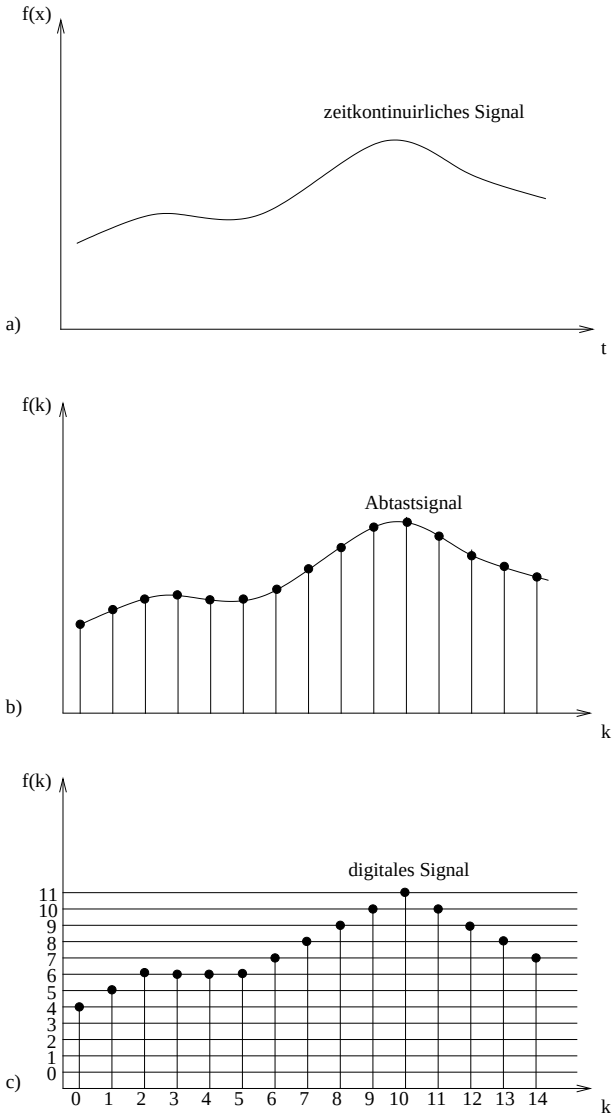

Abbildung 2.1 Vorgang der Digitalisierung: a) zeitkontinuirliches Signal b) diskret abgetastet in der Zeit c) diskret abgetastet in der Zeit und quantisiert der Amplitude (digitalisiertes Signal)

2.3 Nyquist-Abtasttheorem

Wie im vorherigen Abschnitt angesprochen, läßt sich ein kontinuierliches Eingangssignal mit Hilfe eines A/D Wanders in eine Folge diskreter Werte wandeln. Im folgenden soll festgelegt werden, wie hoch die Abtastfrequenz mindestens zu wählen ist, um ein reelles Orginalsignal noch fehlerfrei d.h. ohne Informationsverlust rekonstruieren zu können.
Dabei sei vom folgenden Satz auszugehen:

Satz: *Das Orginalspektrum mit allen seinen Frequenzen steht nur dann unverändert zu Verfügung, wenn die Abtastfrequenz (f_a) mindestens doppelt so hoch gewählt wird, wie die höchste Frequenz f_{max} , die im abzutastenden Signal noch vorkommt.*

d.h. es muß gelten:

$$f_a \geq 2f_{max} \tag{2.5}$$

Diese Bedingung wird als *Nyquist Abtasttheorem* bezeichnet.
Üblicherweise wird die Bedingung durch einen Tiefpaßfilter (Aliasfiler) garantiert, der das Spektrum des Signals zuvor auf f_{max} begrenzt hat.
Der Tiefpaßfilter befindet sich hierbei noch vor dem A/D Wandler [HES93] [TIE93]. Er hat die Aufgabe, Spektralanteile oberhalb von ½ f_a abzuschneiden. Der Aliasfilter muß so dimensioniert werden, dass die Dämpfung f_{max} noch null und bei ½ f_a bereits eine durch die Anwendung vorgegebene Dämpfung aufweist. Es ist immer notwendig, diesen Aliasfilter vorzusehen. Erst diese Frequenzbereichseinschränkung macht es überhaupt möglich, die Nyquistbedingung einzuhalten [GRÜ93].

Im folgenden sollen die Auswirkungen bei Verletzung und Einhaltung des Nyquisttheorems getrennt für den Frequenz- und Zeitbereich behandelt werden. Zuvor sei festgelegt, ½ f_a als Nyquistfrequenz (n_y) zu bezeichnen.
Zur Illustration des Nyquisttheorems im Frequenzbereich ist in Abbildung 2.2 a) ein Signalspektrum mit seiner maximalen Frequenz f_{max} aufgezeigt. Abbildung b) illustriert das Spektrum eines abgetasteten Signals bei Nichtverletzung des Abtasttheorems. Nach der Abtastung ist das Spektrum eine periodische Funktion. Ihre Periode ist gleich der Abtastfrequenz f_a. Abbildung 2.2 c) zeigt abschließend die Verletzung dieses Theorems.
Das Ursprungssignal kann in Abbildung 2.2 b) genau rekonstruiert werden, während in Abbildung 2.2 c), bedingt durch die Frequenzüberlappung, eine Orginalrekonstruktion nicht mehr möglich ist. Negative und positive Frequenzanteile schieben sich ineinander. Eventuell enthaltende Informationen in dem Signalspektrum gehen verloren.

Im Zeitbereich betrachtet, stehen nicht genug Samplewerte pro Zeiteinheit zur Rekonstruktion der Orginalsignals zur Verfügung. Es entstehen Spektralanteile mit der Differenzfrequenz $f_a - f < f_{max}$. Die entstehen Frequenzen waren im Orginalsignal nicht enthalten.
Dieses Phänomen wird als *Aliasing* (Spiegelungsverzerrungen) bezeichnet [TIE93].
Zusammenfassend kann gesagt werden, dass man aus den Abtastwerten einer kontinuierlichen, bandbegrenzten Zeitfunktion die ursprüngliche Funktion wieder vollständig rekonstruieren kann, wenn das Nyquist Abtasttheorem erfüllt ist.

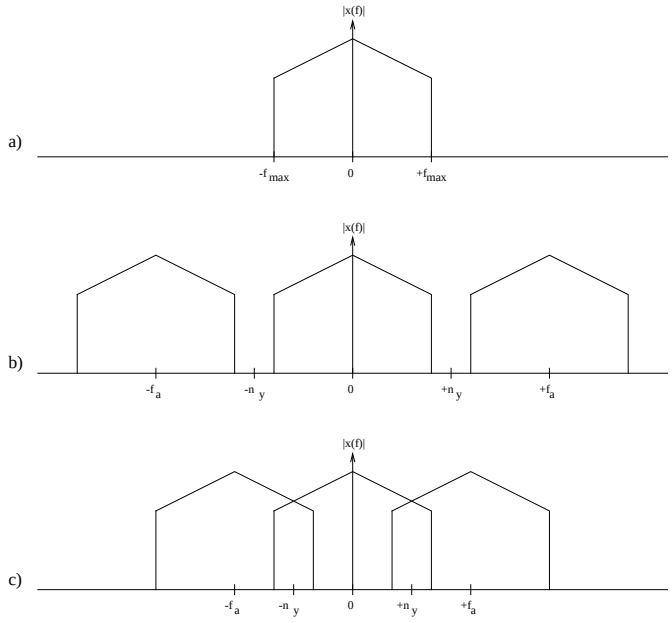

Abbildung 2.2: a) Spektrum eines abgetasteten Signals b) Nyquistkriterium nicht verletzt c) Nyquistkriterium verletzt

2.4 Analytische Signale

In diesem Abschnitt sollen zuerst die komplexen Signale eingeführt werden. Wie im Verlauf dieses Abschnitts noch zu sehen sein wird, stellen die analytischen Signale einen Spezialfall der komplexen Signale dar. Des weiteren soll in diesem Zusammenhang gezielt auf die Unterschiede zwischen reellen und analytischen Signalen hingewiesen werden.

Eine reelle Schwingung

$$f(t) = a\cos(\omega t + \varphi(t)) \tag{2.6}$$

kann zu einer komplexen Schwingung ergänzt werden, indem eine um 90° Grad phasenverschobene Schwingung

$$\hat{f}(t) = a\sin(\omega t + \varphi(t)) \tag{2.7}$$

als *Imaginäranteil j* hinzugefügt wird.

Ein reelles Zeitsignal *f(t)* kann einem komplexen Zeitsignal

$$f'(t) = f(t) + j\hat{f}(t) \tag{2.8}$$

zugeordnet werden. Ein komplexes Signal besteht aus zwei unabhängigen reellen Signalen: dem Realen und dem Imaginären Anteil.

Es gilt:

$$f(t) = Re\{f'(t)\}$$
$$\hat{f}(t) = Im\{f'(t)\} \qquad (2.9)$$

Erzeugt man den Imaginärteil durch Phasendrehung um 90° aus dem Realteil, so spricht man von einem *analytischen Signal*. In diesem Fall sind Real- und Imaginäranteile nicht unabhängig voneinander. Sie enthalten beide die selben Informationen.
Im weiteren Verlauf dieser Diplomarbeit wird der Realteil eine *gerade* und der Imaginärteil eine *ungerade* Funktion darstellen.
Man sagt dann auch $Re\{f'(t)\}$ und $Im\{f'(t)\}$ stellen die *Quadraturkomponenten* von $f'(t)$ dar.

Im nachfolgenden soll näher auf das Spektrum eines analytischen Signals eingegangen werden. Vergleiche hierzu Abbildung 2.3.
Zur Illustration ist dazu in Teilbild a) das Betragsspektrums $U_{BP}(f)$ eines reell abgetasteten *Bandpaßsignals* im Frequenzbereich mit seiner negativen *Spiegelfrequenz* und der *Bandmittenfrequenz* f_T aufgezeigt. Im Spektralbereich gilt:

$$U_{BP}(f) = \frac{1}{2}U_{BP}(f - f_T) + \frac{1}{2}U_{BP}(f + f_T). \qquad (2.10)$$

Nach der Transformation in ein analytisches Signal und einer anschließenden Betragsbildung entsteht ein Spektrum, welches für $f < 0$ verschwindet:

$$U_{BP}(f) = U_{BP}(f + f_T). \qquad (2.11)$$

Teilbild b) in Abbildung 2.3 soll diesen Fall illustrieren:

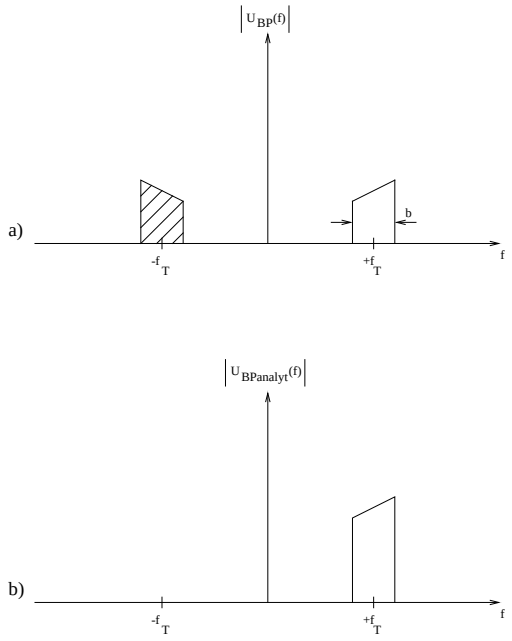

Abbildung 2.3: a) reelles Bandpaßsignal b) analytisches Bandpaßsignal

Zusammengefaßt gilt:

Das Spektrum eines analytisches Signals enthält gegenüber dem Spektrum eines reellen Signals keine negativen Frequenzen.

In diesem Zusammenhang sei bemerkt, dass Teilbild b) keiner Darstellung des analytischen Bandpaßsignals entspricht, sondern nur seinen Betrag aufzeigt und dass eine anschließende Wandlung in ein reelles Signal wieder ein zweiseitiges Spektrum zur Folge hätte [KAM96].

2.5 Unterabtastung um einen ganzzahligen Faktor

Wie im Abschnitt 2.3 dargestellt, muß bei einem abgetasteten Signal *x(t)* die Nyquistbedingung zur eindeutigen Rekonstruktion des Eingangssignals erfüllt sein. Das Abtasttheorem besagt, dass die spektrale Komponente eines Signals mit der höchsten Frequenz mindestens zweimal im Verlauf ihrer Periodendauer abgetastet werden muß.
Angesichts der hohen Anforderungen an die Verarbeitungsgeschwindigkeit bei Signalverarbeitungsaufgaben ist man immer bestrebt, die Abtastfrequenz möglichst niedrig zu halten. Ist zusätzlich eine Abwärtsmischung in die *Nullage* d.h in den Frequenzbereich um *f=0* (Basisband) möglich, so ist nur noch eine Verarbeitung von Niederfrequenzsignalen erforderlich [KRO97]. Welche Auswirkungen dieses auf die arithmetischen Komponenten hat, soll im weiteren Verlauf dieses Kapitels erörtert werden. Wie zudem noch genauer dargestellt wird, zeigt sich speziell für Bandpaßsignale, dass aufgrund ihrer Bereichseinschränkung eine wesentlich geringere Abtastrate nötig ist [KAM96].
Im folgenden soll genauer auf die Unterabtastung und ihre Folgen im Frequenzbereich eingegangen werden.

Wird das Nyquistkriterium bewußt verletzt, kann man von einer *Unterabtastung* sprechen. Die Unterabtastung digitaler Signale wird auch als *Abtastdezimation* (Dezimation) bezeichnet. Allgemein kann gesagt werden, dass eine Abtastdezimation eines diskreten Signals immer dann sinnvoll erscheint, wenn die Bandbreite des Nutzsignals wesentlich unter der halben Abtastfrequenz f_a liegt [FLI93].
Zur Verdeutlichung illustriert Abbildung 2.4 die Auswirkungen einer Unterabtastung im Frequenzbereich.
Das resultierende Spektrum e) entsteht durch Summation der Einzelspektren b) – d), die jeweils einen Teil eines Analogspektrums a) repräsentieren sollen. Zu beachten ist, dass jedes zweite Spektrum zu spiegeln ist. Da es sich beim resultierenden Spektrum e) um ein digitales Signal handelt, ist das Spektrum periodisch mit der Samplefrequenz f_s und symmetrisch um die Nyquistfrequenz ½ f_a. Durch eine Unterabtastung erscheinen also mehrere Frequenzen eines Analogspektrums in einem Bereich des Digitalspektrums.
Um dieser „Mischung“ vorzubeugen, müssen durch entsprechende Filter (Dezimationsfilter) die nichtgewünschten Frequenzanteile weggefiltert werden.
Zusammenfassend ist daher eine *Abtastratendezimation* diskreter Signale immer mit einem *Dezimationsfilter* als Aliasfilter und der eigentlichen *Abwärtsabtastung* zu sehen [FLI93].

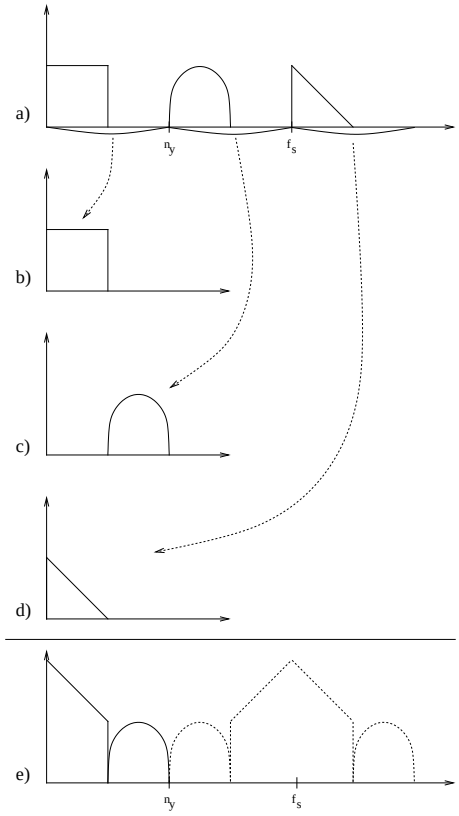

Abbildung 2.4: a) Unterabtastung eines Analogspektrums e) resultierendes Spektrum aus den Teilspektren b) c) d)

Abbildung 2.5 illustriert den Fall einer Abtastdezimation unter Verwendung eines Dezimationsfilters an einem Beispiel.
Es werden mehrere Bandpaßspektren aufgezeigt, eine Bandpaßfilterung zur Selektion eines bestimmten Nutzspektrums und eine abschließende Dezimation um den Faktor vier.
Teilbild a) zeigt die unterschiedlichen Bandpaßspektren, Teilbild b) die Bandpaßfilterung für ein bestimmtes Spektrum und Teilbild c) die Unterabtastung und die damit verbundene Zusammenführung der einzelnen Teilspektren. Wie zu erkennen ist, wurde das gewünschte reelle Bandpaßspektrum durch die Dezimation auf der Frequenzachse in einen niederfrequenteren Bereich versetzt.
Es wird im nächsten Abschnitt zu klären sein, wie hoch die Abtastrate mindestens gewählt werden muß, damit die Informationen das Bandpaßsignals voll erhalten bleiben.
Auf einzelne Samplewerte, also den Zeitbereich bezogen, gilt:
Durch eine Dezimation mit einem Faktor *m* entsteht aus einem diskreten Eingangssignal *x(n)* ein Signal *y(n)* indem nur jeder *n*-te Wert aus *x(n)* zur weiteren Verarbeitung benutzt wird.

$$y(n) = x(m \cdot n) \tag{2.12}$$

Durch Unterabtastung eines Signals (1,4,6,9,13,18,20,26,30,...) mit einem Faktor $m = 4$ entsteht also das Signal (1,13,30,...).

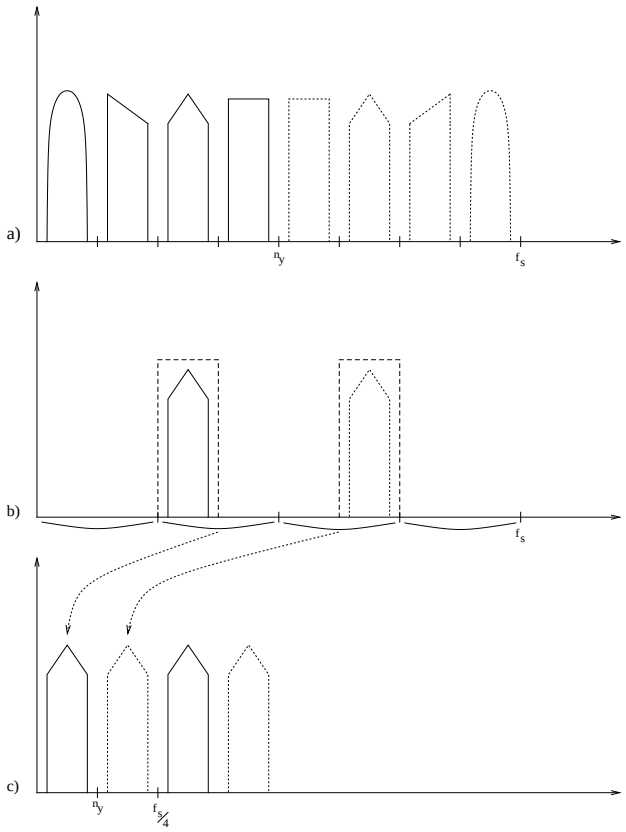

Abbildung 2.5 Unterabtastung eines reellen Bandpaßsignals: a) mehrere Bandpaßspektren b) Bandpaßfilterung c) Unterabtastung um den Faktor 4 mit Dezimationsfilter

Die vorherige Datenrate sinkt. Die weggelassenen Werte werden einfach nicht mehr betrachtet.
Ein entscheidender Vorteil einer Unterabtastung ist, dass bei niedriger Samplefrequenz die Anforderungen an die benötigte Rechenleistung geringer ist, obwohl Signale mit hohen Trägerfrequenzen verarbeitet werden können [KAM96]. Wie später in Abschnitt 2.6.2 zu sehen, ist dieses allerdings nur mit analytischen Bandpaßsignalen zufriedenstellend zu lösen.
Bezogen auf einzelne arithmetische Einheiten gilt:
Nach einer Dezimation mit dem Faktor *m* muß nur mit dem *m*-fachen Sampletakt gearbeitet werden. Wie noch gezeigt wird, ist aufgrund dessen eine Verarbeitung von Daten mit relativ langsamen und damit kleinen arithmetischen Einheiten möglich.

2.6 Abtasttheorem für Bandpaßsignale

Im diesem Abschnitt soll näher auf die Abtastung von reellen und analytischen Bandpaßsignalen eingegangen werden. Dabei soll auch auf Probleme im Zusammenhang mit der Verarbeitung von reellen Signalen hingewiesen werden. Die wichtigsten Unterschiede und Vorteile einer analytischen Signalverarbeitung runden diesen Abschnitt ab.

2.6.1 Abtastung reeller Bandpaßsignale

Es soll zunächst auf die Abtastung von reellen Bandpaßsignalen eingegangen werden. Dabei soll auch auf Probleme, die sich aus der Lage des Bandpaßsignals auf der Frequenzachse ergeben, eingegangen werden.

Für ein reelles Bandpaßsignal $X_{BP}(t)$ soll die Spekralfunktion

$$X_{BP}(f) = F\{x_{BP}(t)\} \quad , \quad |f| \leq f_1 \quad ; \quad |f| \geq f_2 \tag{2.13}$$

mit der Bandbreite

$$b = f_2 - f_1 \tag{2.14}$$

gelten.

Für die Lage eines Bandpaßsignals auf der Frequenzachse sei die Variable λ eingeführt.
Allgemein kann für abgetastete Bandpaßsignale eine *untere* und *obere Grenzfrequenz* f_1, f_2 mit

$$f_1 = \lambda * b \ , \ f_2 = (\lambda+1) * b \qquad \lambda \in N \tag{2.15}$$

angegeben werden.
Für ein überlappungsfreies Spektrum muß gelten:

$$f_T - \frac{b}{2} = \lambda * b \tag{2.16}$$

mit f_T als *Bandpaßmittenfrequenz* des Bandpaßsignals.
Die untere Grenzfrequenz $f_T - b/2$ muß ein *ganzzahliges Vielfaches* der Bandbreite b sein.
Die Abtastfrequenz mit

$$f_a = 2b \tag{2.17}$$

führt zu einer überlappungsfreien Abtastung eines reellen Bandpaßsignals [KRO97].

Zur graphischen Veranschaulichung illustriert Abbildung 2.6 a) ein reelles Bandpaßsignal mit seiner negativen Spiegelfrequenz. Teilbild b) zeigt dazu graphisch die Höhe der Abtastfrequenz $(f_a=2b)$ auf der Frequenzachse, mit dem das Bandpaßsignal abgetastet wird.
Ist die Bedingung aus Gleichung 2.16 mit λ als *gerader* Wert erfüllt, dann ergibt sich in Teilbild c) ein *überlapungsfreies periodisches* Spektrum des reellen Bandpaßsignals in *Regellage*.

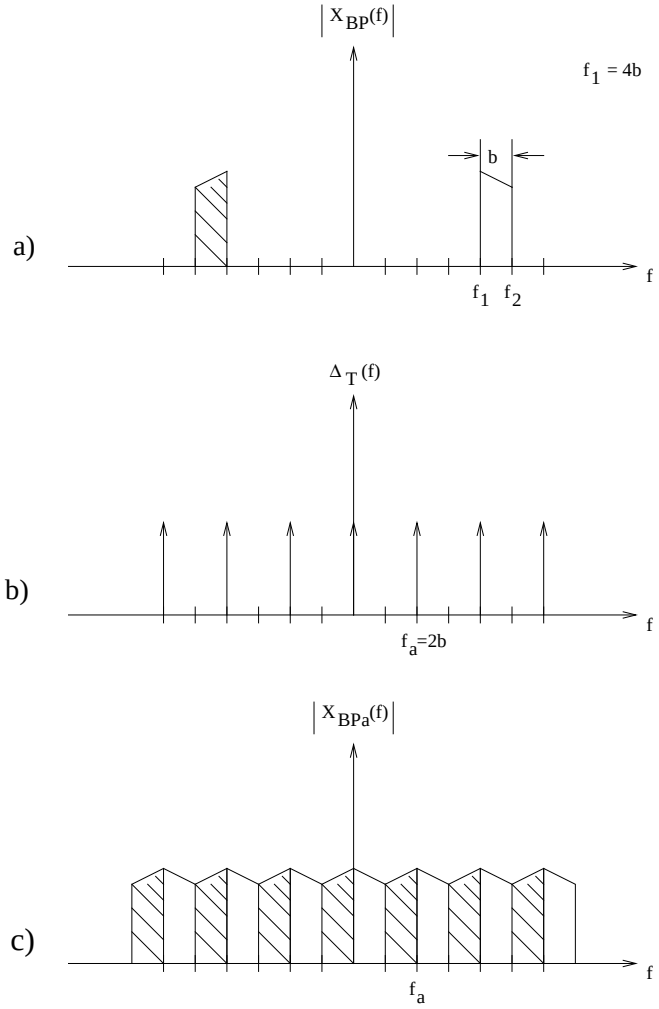

Abbildung 2.6 abgetasteter reeller Bandpaß für einen geraden Abtastwert λ in Regellage: a) reelles Bandpaßsignal b) Abtastung c) abgetasteter Bandpaß in Regellage

Stellt die natürliche Zahl λ einen *ungeraden* Wert dar, dann erscheinen die einzelnen Spektralkomponenten in *Kehrlage*. Die Komponenten des Spektrum erscheinen gespiegelt.
Ist das Nyquistkriterium für reelle Bandpaßsignale $f_a=2b$ erfüllt, die untere Bandgrenze f_1 jedoch kein ganzzahliges Vielfaches der Bandbreite b, dann entstehen spektrale Überlappungen.
Abbildung 2.7 illustriert diesen Fall an einem Beispiel eines reellen Bandpaßsignales mit einer unteren Grenzfrequenz von $f_1 = 3{,}5b$. Obwohl die Nyquistkriterium erfüllt ist, fallen jeweils in Teilabbildung c) ein Spektrum in Regellage und ein Spiegelspektrum in Kehrlage übereinander.
Durch diesen Überlappungseffekt ist eine Trennung der beiden Bandpässe nicht mehr möglich. Die Information des Bandpaßsignals wird unbrauchbar.
Die Bedingungen zur Abtastung eines reellen Bandpaßsignals sollen noch einmal zusammengefaßt werden:

- Die untere Bandgrenze f_1 des Bandpaßsignals muß ein ganzzahliges Vielfaches der Bandbreite b sein.

- Die Abtastfrequenz muß doppelt so groß sein wie die Bandbreite *b* des Bandpaßsignals. Bedingung 2.16 muß gelten.

Da die Bedingung aus Gleichung 2.16 zur Vermeidung des Überlappungsproblems normalerweise in der Nachrichtentechnik nicht erfüllt ist, sollen im folgenden zwei Lösungen dieses Problems vorgestellt werden.

- Um ein reelles Bandpaßsignal abzutasten, kann zur Vermeidung des Überlappungseffektes die Abtastfrequenz in geeigneter Weise erhöht werden. Es ist eine Bandbreitenerhöhung symmetrisch zur Mittenfrequenz f_T des Bandpaßsignals durchzuführen [KRO97].

- Das reelle Bandpaßsignal wird in ein Analytisches gewandelt, das charakterisiert ist durch das Fehlen seiner Spiegelfrequenzen. Eine Überlappung mit seinen Spiegelfrequenzen ist hier nicht mehr möglich.

Bei der ersten Methode geht Bandbreite verloren, was in der Nachrichtentechnik nicht gewünscht ist. Aus diesem Grund stellte die zweite Methode die bessere dar. Im nächsten Abschnitt soll daher das Abtasttheorem auf komplexe Bandpaßsignale erweitert werden.

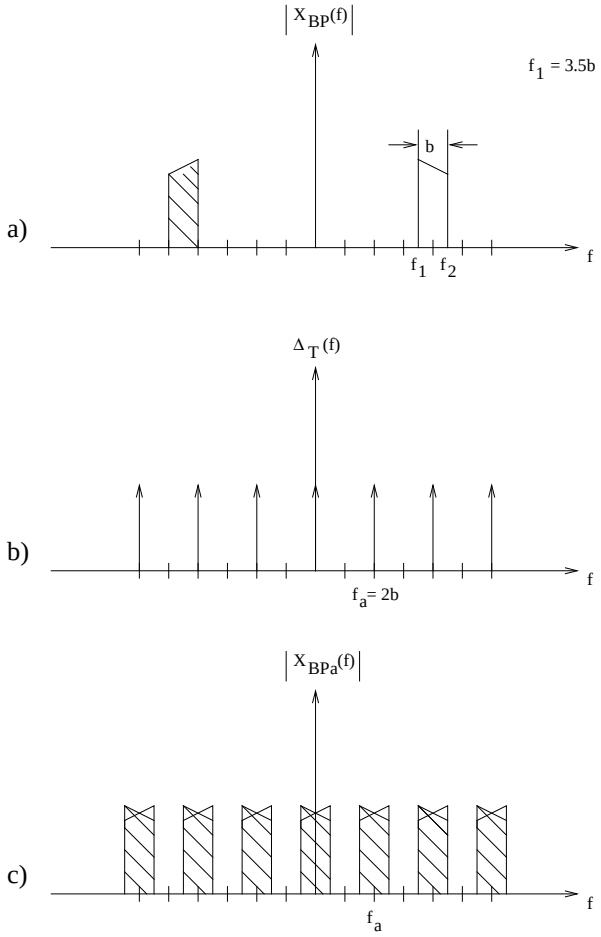

Abbildung 2.7: abgetastetes reelles Bandpaßsignal bei Verletzung der Bedingung $f_1/b \in \mathbb{N}$: a) reelles Bandpaßsignal b) Abtastung c) spektrale Überlappungen

2.6.2 Abtastung komplexer Bandpaßsignale

Im folgenden soll eine weitere Verallgemeinerung des Abtasttheorems für analytische Bandpaßsignale angestrebt werden.
Wie schon aus Abschnitt 2.3 zu entnehmen war, ist das Spektrum eines analytischen Signals einseitig d.h. das Spektrum enthält keine negativen Spiegelfrequenzen.
Tastet man dieses Spektrum ab, so stellt sich heraus, dass die Abtastfrequenz

$$f_a \geq b \tag{2.18}$$

ausreicht, ohne dass es zu spektralen Überlappungen kommen kann [KRO97].
Abbildung 2.8 illustriert die Abtastung eines analytischen Signals für ein nicht ganzzahliges Vielfaches der Bandbreite b. Die untere Grenzfrequenz beträgt $f_1 = 3{,}5b$.

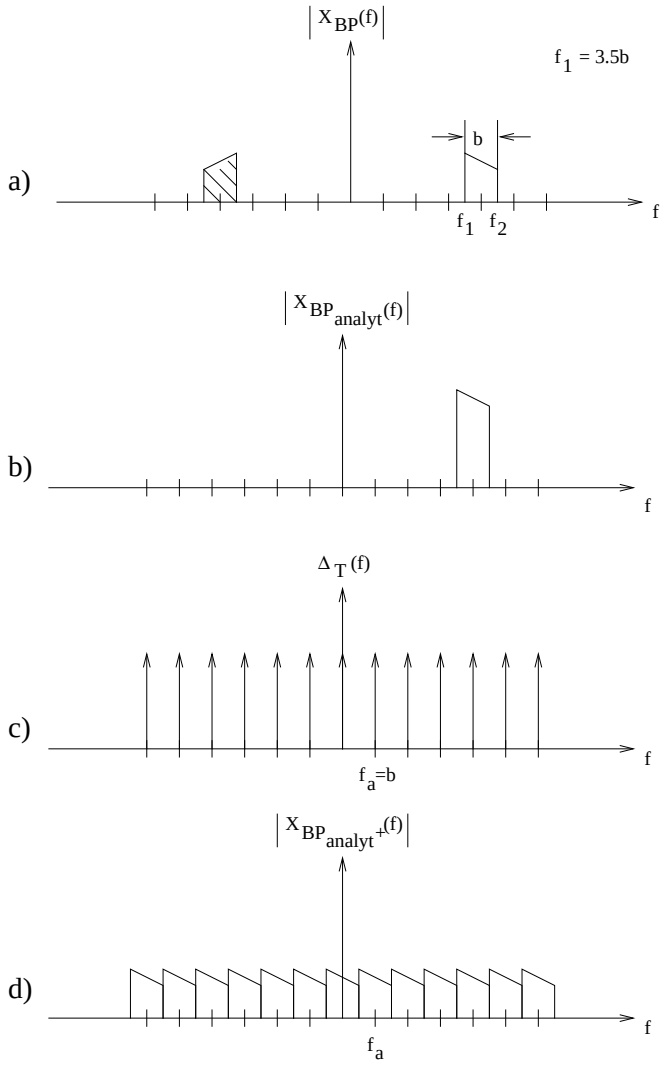

Abbildung 2.8 Abtastung eines analytischen Bandpaßsignals mit einer unteren Grenzfrequenz von f_1=3.5: a) reelles Bandpaßsignal b) analytisches Bandpaßsignal c) Abtastung d) resultierendes überlappungsfreies Spektrum

Teilbild a) zeigt ein reelles Bandpaßsignal. Das Ergebnis der Wandlung in ein komplexes Signal mit positiven Spektrum, welches in Teilbild b) illustriert ist. Es wird hier ausschließlich der Betrag der real- und imaginär Anteile aufgezeigt. Nach der entschärften Abtastung aus Gleichung 2.18 erhält man schließlich in Teilbild d) ein überlappungsfreies Spektrum. Das so abgetastete Signal ist komplex, aber nicht mehr analytisch.
Ist die Mittenfrequenz des Bandpaßsignals ein ganzzahliges Vielfaches der Abtastfrequenz f_a , dann entsteht ein Spektrum, das gleichwertig zu einer Abtastung eines equivalenten niederfrequenten Signals im Basisband ist.
Um eine gewünschte Herabmischung in die Nullage zu erreichen, muß die Abtastfrequenz so gewählt werden, dass die Bandmittenfrequenz f_T ein ganzzahliges Vielfaches der Abtastfrequenz f_a ist.
Zum Abschluß dieses Abschnitts sollen noch einmal die wichtigsten Vorteile einer Verarbeitung von analytischen Signalen zusammengefaßt werden:

- Die Spektralfunktion eines analytischen Signals ist im Bereich negativer Frequenzen vom Betrag her ausgelöscht.

- Die Abtastbedingung für reelle Bandpaßsignale $f_a = 2b$, kann für analytische Bandpaßsignale weiter auf $f_a = b$ entschärft werden.

- Das Spektrum ist unabhängig von der absoluten Frequenzlage.

2.6.3 Zusammenfassung

Bei reellen Bandpaßsignalen ist eine Unterabtastung von $f_a=2b$ möglich. Voraussetzung ist, dass bei einer periodischen Wiederholung des Spektrums infolge der Abtastung keine spektralen Überlappungen auftreten.
Es wurde gezeigt, dass durch eine Verarbeitung von analytischen Bandpaßsignalen die Probleme der spektralen Überlappung vermieden werden können. Wird ein analytisches Bandpaßsignal abgetastet, so muß zur Vermeidung von Überlappungen nur die Bedingung $f_a \geq b$ erfüllt sein. b stellt dabei die Bandbreite des Nutzsignals dar.
Des weiteren ist das Spektrum unabhängig von der absoluten Frequenzlage, d.h. die untere Bandgrenze des Bandpaßsignals muß kein ganzzahliges Vielfaches der Bandbreite sein. Eine Umsetzung des Bandpaßsignals in das equivalente komplexe Basisbandsignal (*komplexe Einhüllende)* und damit eine Verdopplung des Dynamikbereiches ist leicht durchzuführen.
Wie allerdings noch zu zeigen sein wird, besteht für die komplexe Signalverarbeitung ein Mehraufwand aufgrund der benötigen Wandlung des reellen in ein analytisches Signal und der doppelten Ausführung von benötigten Teilkomponenten, wenn eine parallele Verarbeitung von reellen und imaginären Signalen durchgeführt werden soll. Darauf soll in späteren Kapiteln noch genauer eingegangen werden.

2.7 Versatz von Signalen auf der Frequenzachse

Im folgenden sollen zwei verschiedene Methoden vorgestellt werden, ein Signal auf der Frequenzachse zu verschieben.
Diese können grob wie folgt unterschieden werden:

- Versatz durch Mischung (bei analytischer Signalverarbeitung => Quadraturmischung) und
- Versatz durch die Abtastung.

Es soll zunächst die Mischung näher betrachtet werden.
Als Mischung wird eine Multiplikation zweier Signale bezeichnet. Die Quadraturmischung bezieht sich dabei auf analytische Signale [KAM96] [GEI93].

Grundsätzlich können weiter bei einer Mischung unterschieden werden:

- die Aufwärtsmischung und
- die Abwärtsmischung.

Von besonderer Bedeutung sind in der Nachrichtentechnik die Abwärtsmischung ins *Basisband* und die Aufwärtsmischung in eine *trägerfrequente Lage*. Die Abwärtsmischung findet häufig Verwendung, da man wie in den vorherigen Abschnitten angedeutet, aus technischen Gründen immer möglichst niedrige Frequenzlagen anstrebt.
Bei einem Signal im Basisband sind wiederum zwei Fälle zu unterscheiden:

- das Tiefpaßsignal und
- die komplexe Hüllkurve (Einhüllende) eines komplexen Basisbandsignals.

Das *Tiefpaßsignal* entspricht der physikalischen Darstellung der zu übertragenen Nachricht [KAM96]. Bei der *komplexen Hüllkurve* handelt es sich um ein Bandpaßsignal, welches das komplexe Signal eines modulierten Sinusträgers mit der Frequenz *null* darstellt [KAM96].
Es soll genauer auf die Mischung eines reellen Signales im Frequenzspektrum eingegangen werden. Grundsätzlich sind die im folgenden vorgestellten Ergebnisse auch auf komplexe Signale anwendbar.
Multipliziert man zwei reelle harmonische Signale unterschiedlicher Frequenz ω_1 und ω_2, dann entstehen dadurch im resultierenden Signal die Frequenzen $\omega_1 + \omega_2$ und $|\omega_1 - \omega_2|$. Das Spektrum $S(\omega)$ eines Nutzsignals $s(t)$ soll innerhalb der Frequenz ω_1 und ω_2 bandbegrenzt sein (Bandpaßsignal). Durch eine Multiplikation dieses Signals mit einem harmonischen Signal der Frequenz ω_t entsteht ein Spektrum, in dem das Frequenzband aus $S(\omega)$ zweimal vorkommt. Dabei macht es einen Unterschied, ob ω_t ober oder unterhalb des Spektrums $S(\omega)$ vorkommt.
Abbildung 2.9 a), b) illustriert diese Fälle.
Ist des Spektrum $S(\omega)$ symmetrisch um die Frequenz $\omega_t = \omega_2 - \omega_1 / 2$, dann wird das Spektrum um die Frequenz null verschoben. Im umgekehrten Fall d.h. wenn das Spektrum $S(\omega)$ symmetrisch um die Frequenz null ist, wird es zur Frequenz ω_t hin verschoben.

Teilbild 2.9 c) illustriert zunächst den Fall, bei dem eine Symmetrie des Spektrums nicht vorliegt. Es überlagern sich im Basisband die beiden Seiten.
Teilbild 2.9 d) zeigt dann den Fall der Verschiebung eines symmetrischen Spektrums um die Frequenz null. Zu beachten ist bei einer reellen Verarbeitung von Signalen, dass das nicht benötigte Spiegelband durch entsprechende Filter weggefiltert wird [KRO97].
Mit der eben vorgestellten Methode der Mischung, lassen sich Signale *beliebig* auf der Frequenzachse verschieben.

Die zweite auch schon mehrfach angedeutete Methode einen Frequenzversatz durchzuführen, besteht in der Abtastung selbst. Es ist möglich durch *Unter*- oder *Überabtastung* ein Signal auf der Frequenzachse zu verschieben.

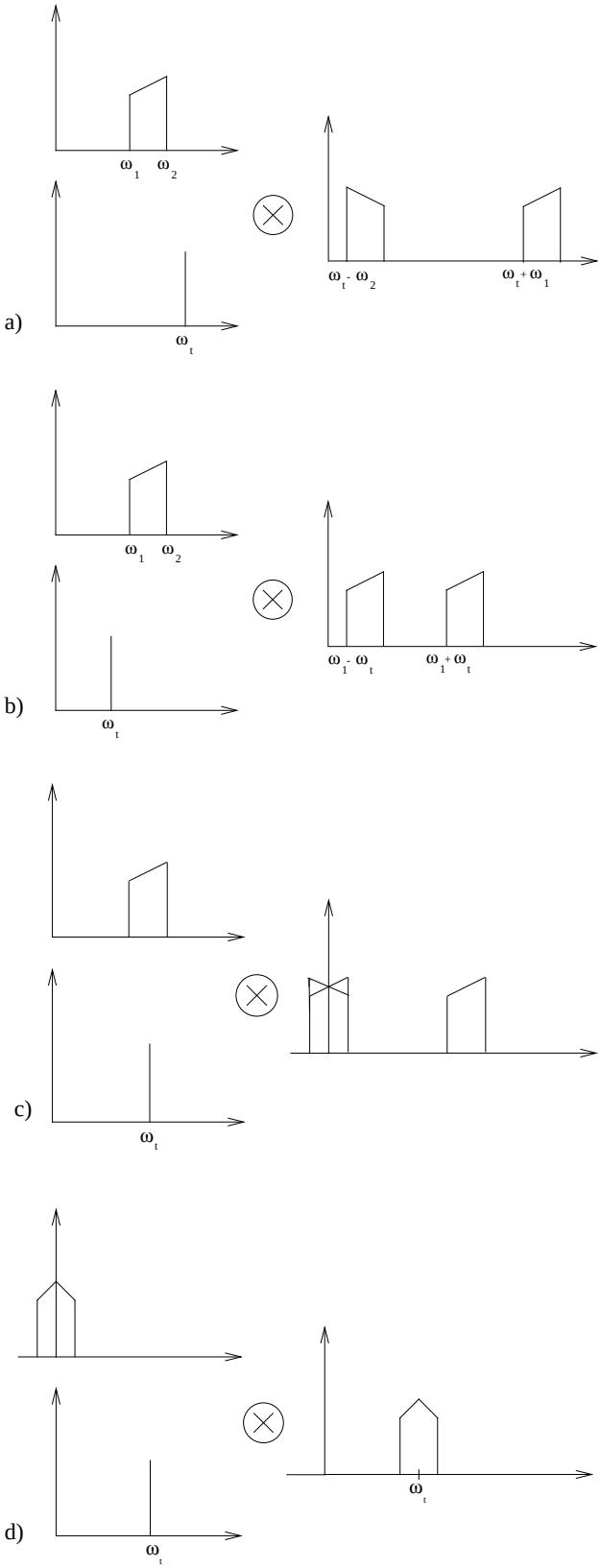

Abbildung 2.9: Mischung zweier reeller Signale

Man erinnere sich, dass für die oft gebrauchte Abwärtsmischung in die Nullage durch Unterabtastung, die Abtastfrequenz so gewählt werden muß, dass die Bandmittenfrequenz f_T ein ganzzahliges Vielfaches der Abtastfrequenz f_a ist. Mit einem entsprechenden Faktor bei der Unterabtastung kann somit das Bandpaßsignal in das Basisband verschoben werden [KRO97].

2.8 Digitale Filter

Da die digitalen Filter ein zentrales Themengebiet dieser Diplomarbeit darstellen, sollen nachfolgend einige wichtige Grundlagen diskussiert werden. Aufgrund der Komplexität der Thematik, werden allerdings nur für die Diplomarbeit relevante Gebiete angesprochen.
Die nachfolgenden Kapitel umfassen eine Einführung der Vorteile digitaler Filter, wichtige theoretische Grundlagen und Eigenschaften und die sich bei einer Realisierung ergebenden unerwünschten Effekte. Erst in den nachfolgenden Kapiteln soll dann auf Realisierungsmöglichkeiten in Hardware und einige Ergebnisse realisierter Filterstrukturen eingegangen werden.

2.8.1 Einleitung

Die Behandlung und Manipulation diskreter Signale erfolgt bei digitaler Signalverarbeitung durch arithmetische und logische Operationen wie Addition, Multiplikation oder Vergleich. Digitale Filter stellen dabei eine wichtige Teilmenge der *Systeme* zur digitalen Signalverarbeitung dar. In der Nachrichtentechnik übernehmen sie dabei Aufgaben, wie Phasenentzerrung, Trennung von Signalen und Bandbreitenbegrenzung. Diese Filter sind bisher vorwiegend mit passiven Bauelementen wie Widerstand, Spule, Kondensator und Transformators sowie auch unter Zuhilfenahme von aktiven Bauelementen wie Transistoren und Operationsverstärker realisiert worden.
Bedingt durch die Integration zunehmend komplexer Funktionsgruppen in digitalen Schaltkreisen, ist mittlerweile die digitale Form der Filterrealisierung möglich geworden.

Sie bietet folgende Vorteile gegenüber herkömmlichen Filterrealisierungen:

- Keine Schwankungen der Filterparameter durch Temperatur oder Alterungseffekte,
- Integration komplexer Systeme,
- Reproduzierbarkeit des Übertragungsverhaltens (keine Bauteiltoleranzen),
- Linearphasigkeit bei bestimmten Filtertypen,
- Vergleichsweise einfache Realisierung von Filtern mit steilen Flanken und
- Filterung extrem niederfrequenter Signale.

Nachteile einer digitalen Filterrealisierung können sein:

- Das vorgeschriebene Systemverhalten wird aufgrund einer endlichen Anzahl von Bits nur näherungsweise erreicht,
- Digitale Systeme weisen ein Eigenrauschen (Quantisierungsrauschen) auf,
- direkte Verarbeitung von Hochfrequenz (Ghz Bereich) noch nicht möglich und
- längere Verarbeitungszeiten aufgrund digitaler arithmetischer Operationen.

Der ursprüngliche hohe Aufwand bei der Realisierung digitaler Filter ist durch beträchtliche Fortschritte in der Integrationsdichte verringert worden.
Des weiteren ermöglicht die hohe Verarbeitungsgeschwindigkeit der digitalen Bauelemente die Mehrfachausnutzung bestimmter Systemteile (Multiplexing, Pipelining), was bei analoger Signalverarbeitung weitaus schlechter zu handhaben wäre.
Im Gegensatz zu analogen Systemen lassen sich Fehler bei digitalen Filtern zu lasten eines erhöhten Aufwandes beliebig verringern bzw. in einigen Fällen gänzlich vermeiden [GRÜ93].
Digitale Filter ersetzen und verdrängen so nach und nach die analogen Systeme und Baugruppen.
Hierdurch wurden im Audiobereich die Compakt Disk (CD), Digital Audio Tape (DAT) oder der digitale Hörrundfunk und die Mobilkommunikation möglich [STR93].

2.8.2 Grundlagen digitale Filter

Geht man von einem digitalen Signal *x(n)* aus, können als digitale Filter alle *signalverarbeitenden* und *zeitinvarianten* Einheiten bezeichnet werden, die aus einem Signal *n(x)* nach bestimmten Gesetzen ein neues Signal *y(n)* bildet. Ein- und Ausgabesignal haben die gleiche physikalische Größe.
Im weiteren Verlauf dieser Arbeit soll nur das Filter im Zeitbereich betrachtet werden. Filter, die im Frequenzbereich arbeiten, sollen hier nicht weiter angesprochen werden, da sie für diese Diplomarbeit nicht relevant sind.

Als digitale Operationen werden bei digitalen Filtern im Zeitbereich ausschließlich Addition, Subtraktion, Multiplikation mit konstanten Koeffizienten sowie die Speicherung um einen oder mehrere Taktzyklen, was einer Verzögerung um eine oder mehrere Abtastintervalle entspricht, verwendet. Nur diese Operationen garantieren die *Linearität* des Filters. Alle anderen Rechenoperationen würden zu nichtlinearen Filtern führen [HES93].
In der technischen Realisierung haben auch lineare Filter nichtlineare Effekte. Die Filter können jedoch so realisiert werden, dass die nichtlinearen Effekte eine untergeordnete Rolle spielen [GRÜ93].
In den nachfolgenden Abschnitten werden zunächst digitale Filter im Zeitbereich kurz charakterisiert und später deren Entwurfsablauf beschrieben. Es werden deren Eigenschaften im Zeit- und Frequenzbereich näher besprochen. Dabei sollen ausschließlich lineare digitale Filter betrachtet werden, die in der Literatur als FIR (Finite Response Filter) bekannt sind.
Digitale Filter gehören zu einer Klasse von Systemen, die auch als *lineare zeitinvariante Systeme* (LTD) bezeichnet werden.

Es sollen nacheinander folgende drei Möglichkeiten näher erörtert werden, um lineare zeitinvariante Systeme (FIR Filter) zu beschreiben:

- Beschreibung durch ihre Impulsantwort,
- Beschreibung über die Übertragungsfunktion und den Frequenzgang,
- Beschreibung durch Differenzengleichungen.

Ein lineares zeitinvariantes und zeitdiskretes System kann vollständig durch seine *Impulsantwort h(·)* beschrieben werden. Die Impulsantwort *h(·)* ist die Reaktion dieses Systems auf einen *Einheitsimpuls* $\delta(\cdot)$, der wie folgt definiert ist:

$$\delta(k) = \begin{cases} 1 & \textit{für } k = 0 \\ 0 & \textit{sonst} \end{cases} \tag{2.19}$$

Die Reaktion *y(·)* eines diskreten, linearen zeitinvarianten Systems auf eine Eingangsfolge *x(·)* wird über eine *Faltungsoperation* im Zeitbereich mit der Impulsantwort des Filters bestimmt [HES93].

Mit * als *Faltungsoperator* gilt:

$$y(i) = h(i) * x(i) \tag{2.20}$$

wobei die Faltung *kommutativ* ist.
Es gilt also:

$$y(i) = h(i) * x(i) = x(i) * h(i). \tag{2.21}$$

Dieses ist die Kurzschreibweise für

$$y(i) = \sum_{k=-\infty}^{+\infty} h(k)\, x(i-k) \tag{2.22}$$

wobei *y(i)* das Ergebnis der Faltung ist. Man sagt *y(i)* ist gleich *h(k)* gefaltet mit *x(i)*.

Für einen FIR Filter mit der Länge der Impulsantwort von *N* Abtastwerten gilt dann die *diskrete Faltungssumme*:

$$y(i) = \sum_{k=0}^{N-1} h(k)\, x(i-k) \tag{2.23}$$

Man kann sagen, dass nach Gleichung 2.23 der Ausgangswert ein gewichteter Mittelwert der Eingangsdaten *x(i-k)* mit den Filterkoeffizienten *h(k)* darstellt [HAM87] [GRÜ93].

Zur besseren Vorstellung illustriert Abbildung 2.10 graphisch die Faltungsoperation zweier Rechtecksignale. Dargestellt werden soll hier, wie ein Rechtecksignal unter einem anderen über die Faltungsoperation quasi „durchgeschoben“ wird, wobei die resultierende Fläche als *Faltungsintegral* nach jedem Schiebevorgang aufgezeigt wird.
Das Resultat der beiden Rechteckimpulse weist eine Dreiecksstruktur als Faltungsintegral auf. Man kann sagen, dass ein Rechtecksignal dem eigentlichen Eingangssignal und das andere den gespiegelten Filterkoeffizienten entsprechen.
Das Ausgangssignal entsteht damit durch die Faltung seiner Impulsantwort mit dem Eingangssignal.

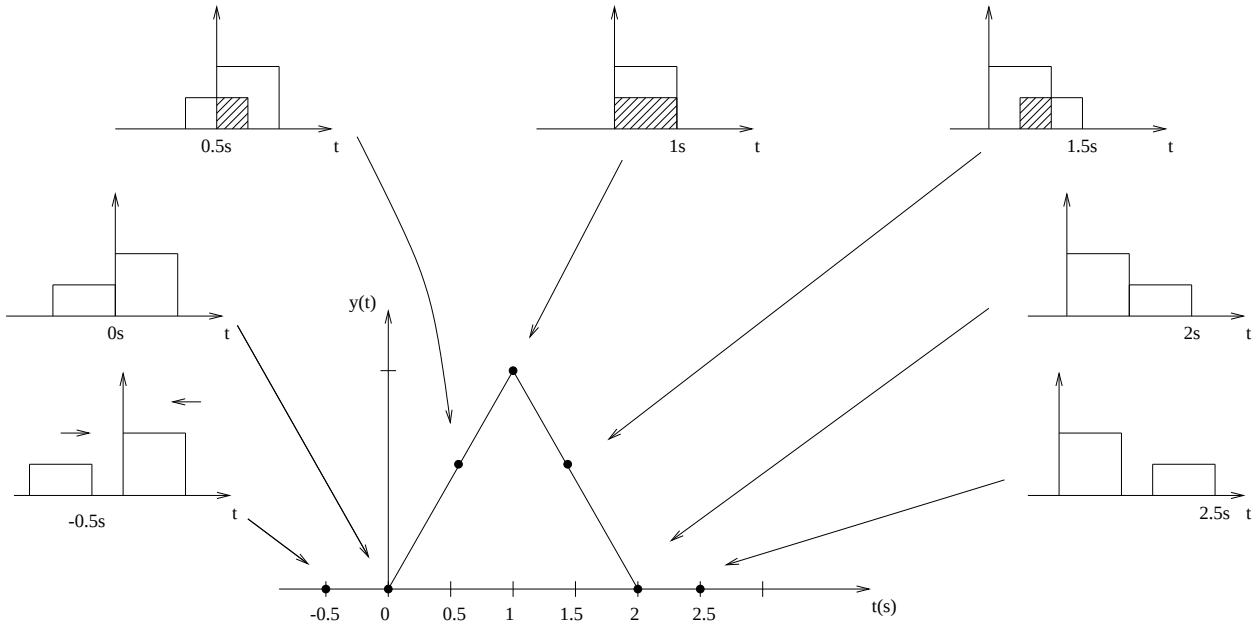

Abbildung 2.10: graphische Verdeutlichung der Faltungprozesses im Zeitbereich

Neben der Darstellung im *Zeitbereich* soll auch eine Betrachtung im *Frequenzbereich* oder *transformierten* Bereich erfolgen.
Analog zur *Laplace-Transformation* bei kontinuierlichen Signalen, wird bei diskreten Signalen die *z-Transformation* verwendet [HES93].
Für die *z*-Transformation der Impulsantwort *h(k)* eines FIR Filters gilt:

$$Z[h(k)] = \sum_{k=-\infty}^{+\infty} h(k) z^{-k} \tag{2.24}$$

Z[h(k)] ist eine *komplexwertige Funktion* und *z* die *komplexwertige Variable*. Die *z*-Transformierte läßt sich aufgrund von Gleichung 2.24 und der Eigenschaften der *z*-Transformation berechnen. Auf gleiche Weise kann auch die *z*-Transformierte der Eingangsfolge *x(·)* formuliert werden. Zur weiteren Vertiefung soll auf [HAM87] [TIE93] hingewiesen werden.

Die zweite Möglichkeit linear zeitinvarinate Systeme zu beschreiben, besteht über die *Übertragungsfunktion H* und den *Frequenzgang*.
Unter der Übertragungsfunktion *H(z)* eines LTD Systems versteht man den Quotienten aus *Y(z)* und *X(z)*.

$$H(z) = \frac{Y(z)}{X(z)} \tag{2.25}$$

X(z) ist dabei die *z*-Transformierte des zeitdiskreten Eingangssignals *x(n)* und *Y(z)* ist die *z*-Transformierte des diskreten Ausgangssignals *Y(z)* [HES93].
Die z-Transformation auf den Einheitskreis angewendet, ergibt die diskrete Fourier-Transformation mit:

$$z = e^{j\omega} \tag{2.26}$$

Durch das Einsetzen von Gleichung 2.29 in die *z*-Transformierte der Impulsantwort erhält man den *Frequenzgang* der Übertragungsfunktion:

$$H(e^{j\omega}) = \sum_{k=-\infty}^{+\infty} h(k) e^{-j\omega k} \tag{2.27}$$

Aufgrund der Eigenschaften der *Exponentialfunktion e* mit seinem imaginärem Argument *j*, ist die Übertragungsfunktion periodisch in ω. Die Periode ist $\omega = 2\pi fT$. Es wird dabei von einer normierten Frequenz *f=1* ausgegangen.
Der Frequenzgang der Übertragungsfunktion kann nach Gleichung 2.30 in einen *Amplitudengang* (Betrag) und *Phasengang* (Argument) aufgeteilt werden. In vielen Anwendungen, besonders bei der digitalen Nachrichtenverarbeitung werden Filter mit *linearer Phase* gewünscht [GER97]. Wie im Verlaufe dieses Kapitels noch zu sehen, haben Filter mit linearer Phase eine konstante *Gruppenlaufzeit*. Sie zeichnen sich ferner durch eine *symmetrische* Impulsantwort aus [HES93] [HAM87].
Die dritte Möglichkeit zur Beschreibung linearer zeitinvarianter Systeme besteht in der Anwendung von *Differenzengleichungen*.
Die Differenzengleichung der Transversalfilter (FIR Filter), die man durch die inverse *z*-Transformation der Übertragungsfunktion erhält, sieht wie folgt aus:

$$y(i) = h_0 x_i + h_1 x_{i-1} + \ldots + h_{i-1} x_1 + h_N x_0 \,,\; y(i) = \sum_{k=0}^{N} h_k x_{i-k} \,. \tag{2.28}$$

Für die Übertragungsfunktion *H* gilt dann:

$$H(z) = \frac{Y(z)}{X(z)} = \sum_{k=0}^{N} h_k z^{-k} \,. \tag{2.29}$$

Damit ergibt sich aus der Differenzengleichung:

$$H(z) = [h_0 + h_1 z^{-1} + h_2 z^{-2} + ... + h_{N-1} z^{-(N-1)} + h_N z^{-N}] X(z) \tag{2.30}$$

Es soll im folgenden auf spezielle Eigenschaften von Filterkoeffizienten eingegangen werden. Dabei sei von Gleichung 2.30 ausgegangen:
Setzt man in diese Gleichung die Eulersche Beziehung

$$z^{-1} = e^{-j2\pi f} = \cos 2\pi f - j \sin 2\pi f \tag{2.31}$$

ein, dann folgt der komplexe Frequenzgang eines FIR Filters:

$$H(j\omega) = \sum_{k=0}^{N} h_k e^{-j2\pi ft}. \tag{2.32}$$

Diese Beziehung läßt sich vereinfachen, wenn die Filterkoeffizienten symmetrisch sind. Diese Eigenschaft der Filterkoeffizienten soll später für alle zu realisierenden Filter gelten.
Es können zwei *Filterkoeffizientensymmetrien* unterschieden können. Deren komplexer Frequenzgang soll im folgenden besprochen werden.

Für die Filterkoeffizienten *h(k)* kann gelten:

$h_{N-k} = h_k$ für eine *gerade* Symmetrie und

$h_{N-k} = -h_k$ für eine *ungerade* Symmetrie. (2.33)

Für den komplexen Frequenzgang lassen sich dann jeweils zwei Terme mit betragsmäßig gleichen Koeffizienten zusammenfassen und der gemeinsame *Phasenfaktor* $e^{-j\pi Nf}$ läßt sich ausklammern.
Bei *gerader Symmetrie* vereinfacht sich der komplexe Frequenzgang und damit Gleichung 2.35 zu:

$$H(j\omega) = e^{-j\pi Nf} \sum_{k=0}^{N} h_k \cos(\pi(N-2k) f). \tag{2.34}$$

Für eine *ungerade Symmetrie* gilt:

$$H(j\omega) = je^{-j\pi Nf} \sum_{k=0}^{N} h_k \sin(\pi(N-2k) f). \tag{2.35}$$

Bei ungerader Symmetrie und einer geraden Anzahl N von Filterkoeffizienten verschwindet der mittlere Koeffizient d.h. *h(½N)* muß *0* sein.
Eine ungerade Übertragungsfunktion äußert sich in einer ungeraden Symmetrie der Filterkoeffizienten. In diesem Zusammenhang sei auf den noch zu besprechenden Hilbertfilter hingewiesen, der eine ungerade Übertragungsfunktion aufweist und zur Wandlung eines reellen

in ein analytisches Signal eingesetzt wird. Auf ihn wird in Abschnitt 2.12 noch genauer eingegangen.

Zwei Eigenschaften im Zusammenhang mit digitalen Filtern sind der *Verschiebungssatz* in den *z*-Bereich mit den dazugehörigen *Korrespondenzen* und das *Faltungstheorem*.
Aufgrund der Wichtigkeit soll auf die Zusammenhänge näher eingegangen werden:

Das $\bullet\!\!-\!\!o$ Symbol soll das *Transformationssymbol* vom Zeit- in den Frequenzbereich darstellen [PIR96]. Ist *X(z)* die *z*-Transformierte von *x(k)* und ist *x(k)* die inverse *z*-Transformierte von *X(z)*, dann bilden *x(k)* und *X(z)* ein *z-Transformationspaar*.
Dieses kann symbolisch dargestellt werden durch:

$$x(k) \quad \bullet\!\!-\!\!o \quad X(z) \tag{2.36}$$

Daraus ergibt sich die für FIR Filter wichtige Zeitverschiebung eines zeitdiskreten Signals *x(n)* um *i* Abtastintervalle.
Man erhält dann folgendes Transformationspaar:

$$x(k-k_0) \quad \bullet\!\!-\!\!o \quad X(z)z^{-k0} \tag{2.37}$$

Die Multiplikation im *z*-Bereich mit z^{-k0} entspricht somit einer Verzögerung um *i* Abtastpunkte im diskreten Zeitbereich.

Auf die Faltung bezogen gilt:
Die Faltung im diskreten Zeitbereich ist über eine Multiplikation im z-Bereich möglich.
Daraus ergibt sich:

$$\begin{aligned} x(i) * h(i) \quad &\bullet\!\!-\!\!o \quad X(z) \cdot H(z) \\ x(i) \cdot h(i) \quad &o\!\!-\!\!\bullet \quad X(z) * H(z) \end{aligned} \tag{2.38}$$

Es gilt damit das *Faltungstheorem*:

Eine Faltung im Zeitbereich entspricht einer Multiplikation im Frequenzbereich;
Eine Multiplikation im Zeitbereich entspricht einer Faltung im Frequenzbereich.

2.9 Eigenschaften nichtrekursiver Digitalfilter (FIR-Filter)

Das *Transversalfilter* (FIR Filter) ist aufgrund einiger Vorteile das am häufigsten verwendete nichtrekursive Digitalfilter. Es sollen daher einige wichtige Charakteristika zusammengefaßt werden.

2.9.1 Struktur und Filterfunktion

Eine wichtige Eigenschaft des FIR Filters ist ihre einfache und reguläre Struktur, die wie noch zu sehen sein wird, besonders bei einer Hardwarelösung Vorteile bietet. Sie erlaubt die Realisierung von *Tiefpaß-*, *Hochpaß-*, *Bandpaß-*, *Bandsperr-* und *Mehrfachbandfiltern* sowie von *Hilbert-transformatoren* und *Differenzierern*. Die *Ordnung* (Güte) des Filters kann leicht geändert werden, indem Multiplizierer, Addierer und Register (MAC-Einheiten) kaskadiert oder weggelassen werden [TIE93].

2.9.2 Impulsantwort

Das Transversalfilter *N-ter* Ordnung hat eine endliche Impulsantwort der Länge *N+1*.

$$\{h_0, h_1, h_2, \ldots, h_n\} = \{b_0, b_1, b_2, \ldots, b_N\} \tag{2.39}$$

Wegen der endlichen Impulsantwort gehört der Transversalfilter zur Klasse der FIR Filter.
Gibt man in einen FIR Filter einen Einheitsimpuls, dann erhält man an Ausgang zuerst den Koeffizienten $y(t) = h_0$ und dann $y(t+1) = h_1 \ldots$, also einen Koeffizienten nach dem anderen.
Es läßt sich in der Systemtheorie zeigen, dass die Impulsantwort eines digitalen Filters die *inverse Fouriertransformierte* seines Frequenzganges $H(j\omega)$ darstellt [TIE93] [HAM87].
Es gilt:

Die Einheitsimpulsantwort eines FIR Filters ist die Folge seiner Koeffizienten. Sie ist N+1 Werte lang.

2.9.3 Phasengang und Gruppenlaufzeit

Falls die Impulsantwort *h(n)* eines FIR Filters symmetrisch ist, verläuft sein Phasengang stückweise linear [GRÜ93]. Seine Laufzeit oder Gruppenlaufzeit τ_g ist dann konstant und hat den Wert:

$$\tau_g = \frac{N}{2} T \tag{2.40}$$

wobei *N* der Filterordnung und *T* dem Abtastintervall ($T=1/f_a$) entspricht.

Unter der Gruppenlaufzeit versteht man die Zeit, in der ein Signal, das in einem engen Frequenzbereiches liegt benötigt, um das Filter zu durchlaufen. Eine konstante Gruppenlaufzeit hat den Vorteil, dass ein Signal das im Durchlaßbereich liegt nur verzögert, aber nicht verzerrt wird.
Diese Eigenschaft ist für die Übertragungstechnik von entscheidender Bedeutung, d.h. für solche Aufgaben lassen sich nur FIR Filter einsetzen.
Die *Gruppenlaufzeit* und der *Phasengang* sind über folgende Gleichung miteinander verknüpft:

$$\tau_{g(f)} = -\frac{1}{2\pi}\frac{d\Phi(f)}{df} \tag{2.41}$$

Ist der Phasengang $\Phi(f)$ eines Filters stückweise *linear*, dann folgt aus der obigen Gleichung, dass die Gruppenlaufzeit konstant ist. Sie ist *frequenzunabhängig*. Laufzeitverzerrungen können nicht auftreten.

2.10 Unerwünschte Effekte bei Digitalfiltern

In der Realität stellt die Genauigkeit der Repräsentation von Koeffizienten und Ergebnissen von Addition und Multiplikation bei digitalen Filterstrukturen ein Problem dar. Bei jedem digitalen Rechner können numerische Zahlen nur durch eine endliche Anzahl von Bits dargestellt werden. Zahlenwerte und Ergebnisse mathematischer Operationen sind daher im allgemeinen mit Fehlern behaftet. Dieses tritt besonders bei der später benutzten Festkommadarstellung für die Filterkoeffizienten und arithmetischen Einheiten auf [HES93].
Diesem Problem könnte man mit einer Fließkommadarstellung, bedingt durch den großen Dynamikbereich entgegen kommen. Aufgrund des hohen Aufwandes und der sich aus dem Einsatz einer Fließkommaarithmetik ergebenden geringen Vorteile für die Einsatzgebiete des ASICs, wurden in dieser Arbeit in diesem Bezug keine weiteren Untersuchungen durchgeführt [TIE93].
In diesem Unterkapitel soll es darum gehen, sich ergebende Fehler im Bezug auf die später verwendete Integer-Festkommadarstellung zu diskutieren.

2.10.1 Zahlendarstellungen

Da unter den binären Zahlendarstellungen besonders die *2'er Komplementdarstellung* für digitale Filter geeignet ist, soll sich im Nachfolgenden auf diese beschränkt werden.
Die Darstellung einer ganzen Zahl Z im 2'er Komplement lautet:

$$Z = -b_{B-1}2^{B-1} + \sum_{n=0}^{B-2} b_n 2^n \tag{2.42}$$

B stellt die Wortlänge dar d.h. die Anzahl der Bits mit der die Zahl Z dargestellt wird, während b_i ein Bit repräsentiert. Das erste Bit in einer 2'er Komplementdarstellung nennt man *Vorzeichenbit*.
Als Vereinbarung soll gelten:
Ist das Vorzeichenbit *null* dann liegt eine positive Zahl vor, ist es dagegen *eins* dann eine negative Zahl. Das hinterste Bit b_0 ist das Bit mit der niedrigsten Wertigkeit und wird als *LSB (Least Signifikant Bit)* bezeichnet [WAL97].

Wichtig im Bezug auf digitale Filterrealisierungen ist der *2'er Komplementüberlauf*, d.h. ein Überlauf ins Vorzeichenbit, was durch entsprechende Maßnahmen zu verhindern ist.
Die 2'er Komplementdarstellung bietet den Vorteil, dass sich das arithmetische Rechenwerk für die Multiplikation und insbesondere für die Addition und Subtraktion sehr einfach realisieren läßt. Genaueres ist in entsprechender Literatur nachzulesen [TIE93] [WAL97].
Im weiteren wird davon ausgegangen, dass nur Rechenwerke mit Integer – 2'er Komplement Festkommadarstellung verwendet werden. Mögliche Fließkommarechenwerke würden einen zu großen Aufwand bedeuten [ZÖL97] [TIE93].

2.10.2 Wertquantisierung

Unter *Wertquantisierung* versteht man das *Runden* oder *Abschneiden* einer beliebigen reellen Zahl *x*. Die Wertquantisierung wird durch eine treppenförmige Kennlinie, der sogenannten *Quantisierungskennlinie* beschrieben [GRÜ93].
Die quantisierte Zahl x_q kann nur Höhen annehmen, die den Höhen der Treppenstufen entsprechen. Die beiden häufigsten Quantisierungskennlinien sind *Rundungs-* und *Abschneidekennlinien*.
Während die Rundungskennlinie vor allen bei A/D Wandlern auftritt, findet man die Abschneidekennlinie vorwiegend bei der Ergebnisquantisierung von *Festkommamultiplikationen*.
Die Treppenhöhe und Breite nennt man *Quantisierungsstufe q*. Ein *Quantisierungsfehler e* läßt sich wie folgt definieren:

$$e = x_q - x \tag{2.43}$$

Es lassen sich Grenzen angeben.
Für den *Rundungsfehler* gilt:

$$-\frac{q}{2} \le e < \frac{q}{2} \tag{2.44}$$

und für den *Abschneidefehler* :

$$-q \le e < 0 \tag{2.45}$$

2.10.3 Überläufe

Ist die Eingangsgröße außerhalb festgelegter Grenzen, so spricht man von einem *Überlauf*. Die Zahl kann durch die bereitgestellten Bits nicht mehr dargestellt werden. Ein 2'er Komplementüberlauf äußert sich mit der Zeit, als sägezahnähnliche Struktur der Quantisierungskennlinie. Dieses kann bei einer 2'er Komplementarithmetik insbesondere dann auftreten, wenn wie bei FIR Filter-realisierungen benötigt, viele Addierstufen mit zu kleinen Busstrukturen kaskadiert werden. Auch der A/D Wandler hat einen Quantisierungsfehler. Auf diesen soll hier jedoch nicht weiter eingegangen werden. Es wird auf weitere Literatur hingewiesen [ZÖL97][TIE93].

2.10.4 Quantisierung der Filterkoeffizienten

Filterentwurfsprogramme liefern Koeffizienten der dazugehörigen Übertragungsfunktion mit einer Genauigkeit, die für die meisten Applikationen oft nicht gebraucht werden [MAT95]. Der sich hieraus ergebende Rechenaufwand wäre zu groß.
Aus diesem Grund werden diese in der Regel zur weiteren Verwendung *wertquantisiert*.
Die Quantisierung der Koeffizienten verändert die *Übertragungsfunktion* und damit den *Amplitudengang*, den *Phasengang* und die *Gruppenlaufzeit*. Die Veränderungen können soweit gehen, dass das Toleranzschema und die Filterspezifikation verletzt wird.
Nach der Wertquantisierung sollte daher immer an Hand einer nachträglichen Simulation geprüft werden, ob die Filterspezifikation noch eingehalten wurde. Sollte dieses nicht der Fall sein, dann kann die Filterordnung erhöht oder die Filterspezifikationen geändert werden.
Die hier verwendeten Transversalfilter haben die vorteilhafte Eigenschaft, dass die unerwünschten Effekte, die aus der Wertquantisierung resultieren, wie ungenauer Frequenzgang und Rundungsrauschen, klein und meistens vernachlässigbar sind.

2.11 Digitale Direkt Synthese (DDS)

Für den Frequenzversatz und für die später noch zu besprechende Quadraturmischstruktur werden sinusförmige digitale Signale mit definierter Frequenz und Phase benötigt. Prinzipiell ist deren Generierung mit Hilfe einer *Taylor Reihenentwicklung* möglich [BAR91] [BRO81]. Aufgrund des damit verbundenen hohen Aufwandes im Hinblick auf eine Hardwarerealisierung, soll auf diese Art der Signalgenerierung nicht weiter eingegangen werden.
Für die spätere Hardwareimplementierung soll im folgenden die *Digitale Direkt Synthese (DDS)* vorgestellt werden.
Die DDS ist eine diskrete Methode, Signale unterschiedlicher Form und Amplitude zu generieren [OSI99]. Der benötigte DDS Generator soll die Aufgabe haben, einen periodischen digitalen Sinus und ein um 90° Grad phasenverschobenes Cosinus Signal zu erzeugen.
Zur Veranschaulichung des dahinterstehenden Prinzips, wird einfachheitshalber davon ausgegangen, dass der DDS Generator eine Tabelle enthält, die Werte einer vollen Sinusperiode 2π repräsentiert.
Zu erzeugen ist eine Sinusfrequenz mit einem *Phasenwinkel* φ und dem *Amplitudenwert A*.

$$A(t) = a \sin(\omega t_i + \varphi) \tag{2.46}$$

für fortlaufende Zeitpunkte:

$$t_i = i * \tau_a \qquad mit \;\; i \in N \tag{2.47}$$

mit $f_a = \frac{1}{\tau_a}$.

Wird der Sinus mit Hilfe einer Sinustabelle generiert, so ergibt sich das Argument Ψ_i des Sinussignals aus $\omega\tau_a i + \varphi$, wobei i als fortlaufend anzusehen ist. Der zeitliche Abstand zwischen den Sinuswerten der Ausgangsfolge $A(t)$ entspricht der Dauer τ_a.

Die fortlaufenden Werte des zu berechnenden Argumentes $\Psi_{i+1} = \omega t_i + \Psi_i$, $\Psi_0 = \varphi$ bis zu einer vollen Periode τ lassen sich mit einem *Phasenakkumulator* realisieren. Für jedes *Argument* Ψ ist ein diskreter Sinuswert zu bestimmen, der als *Nachschlagwert* der Sinustabelle entnommen werden soll.
Abbildung 2.11 soll die Generierung eines Sinus Signals an einem Einheitskreis bildlich veranschaulichen.
Zu diesem Zweck soll der *Phasenakkumulationsprozess* und die dazugehörende *Amplitudenrepräsentation* für einen Zyklus von je *2π/8* Schritten aufgezeigt werden. Die Punkte *A-H* zeigen den *Amplitudenwert A* zu einer bestimmten Zeit t_i des Sinus Signals auf. Das „Nachschlagen" in der Sinustabelle konvertiert die *Phaseninformation* in die dazugehörige *Sinus-Amplituden* Information.

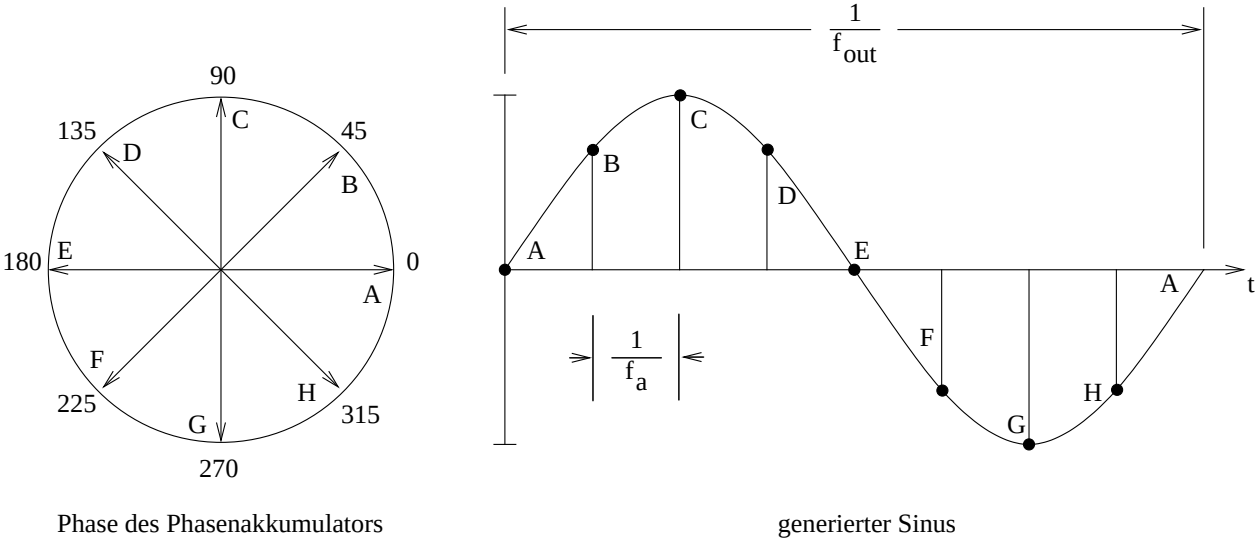

Abbildung 2.11: Illustration der Sinusgenerierung auf dem Einheitskreis mit 8 Punkten

Prinzipiell ist es möglich, auf Kosten der Länge der Tabelle, eine Sinusperiode in *n* diskrete Sinuswerte aufzuteilen. Zur Illustration sollen allerdings acht Werte ausreichend sein.
Der Amplitudenwert *a* des Sinussignals entspricht der Größe der Repräsentation des diskreten Wertes in einer Tabellenrepräsentation und ist durch die Anzahl der repräsentierenden Bits festgelegt.
Diese Art der Sinusgenerierung mit Hilfe einer Sinusrepräsentationstabelle und einem Phasenakkumulator mit einem Phasenmodulo von 2π ist, wie im nächsten Kapitel zu sehen, in Hardware mit wenig Aufwand zu realisieren. Auf die genaue Hardwarerealisierung wird in Abschnitt 3.6 eingegangen.

2.12 Hilbertfilter und Quadraturmischung

Im nachfolgenden sollen, nach der Einführung des komplexen Signals in Abschnitt 2.4, zwei grundsätzliche Konzepte zur Wandlung von reellen in analytische Signale unter dem besonderen Aspekt einer einfachen Hardwareimplementierung behandelt werden.
Dazu sollen zwei Wandlerstrukturen vorgestellt werden:
Die erste vorzustellende Wandlerstruktur soll als *Hilberttransformators H* bezeichnet werden. Sie ist aus einem *Hilbertfilter* und einer *Verzögerungsschaltung* aufgebaut.
Die auf den Mathematiker *Hilbert* zurückgehende Hilberttransformation, deren Übertragungsfunktion ungerade ist, stellt einen Zusammenhang zwischen Real- und Imaginärteil des Spektrums eines kausalen Signals her [KAM96]. Dabei bezieht sich die Hilberttransformation nur auf die Spektralfunktion und nicht auf eine Funktion der Zeit.
Die spektralen Komponenten nach einer Hilberttransformation *H{f(t)}* eines reellen Signals *f(t)* sind um 90° Grad gegenüber des Ursprungssignals *phasenverschoben*.
Ein Signal das einen Hilbertfilter durchläuft dreht somit das Spektrum eines reelles Eingangssignal um 90° Grad in seiner Phase [KAM96][GER97]. Da alle ursprünglichen Frequenzen erhalten bleiben, stellt dieser Filtertyp einen *Allpaßfilter* dar.
Zusätzlich zu dem eigentlichen Hilbertfilter wird eine Verzögerungssschaltung benötigt, die Sampledaten um die Gruppenlaufzeit des eigentlichen Filters, verzögert als Realanteil ausgibt.
Abbildung 2.12 a) illustriert den Gesamtaufbau eines Hilbertfilters.

Die zweite Struktur zur Wandlung ein reelles in ein analytisches Signal illustriert Abbildung 2.12 b). Sie besteht aus einer *Quadraturmischstruktur*. Dazu wird das reelle Eingangssignal *f(t)* mit einer *komplexen Schwingung* $e^{(j2\pi ft)}$, d.h. mit einem Sinus für den Imaginäranteil und einem um *90*° Grad phasenverschobenen Cosinus Signal für den Realanteil multipliziert und anschließend tiefpaßgefiltert. Die beiden Tiefpässe in der Abbildung haben die Aufgabe, spektrale Anteile doppelter Frequenz zu unterdrücken [GER97].
Wie schon angedeutet, sind in der digitalen Signalverarbeitung die Sonderfälle der *Abwärtsmischung* ins Basisband und die Aufwärtsmischung in eine *trägerfrequente* Lage von besonderer Bedeutung. Durch einen Quadraturmischer ist es möglich, ein analytisches Signal zu erzeugen und gleichzeitig eine Abwärtsmischung in das Basisband durchzuführen. Durch den Einsatz einer Quadraturmischstruktur kann eine Einsparung an Fläche für eine Hardwarerealisierung resultieren, da der zusätzlich zur Wandlung in ein analytisches Signal benötigte Hilberttransformator entfallen kann. Die erzeugten real und imaginär Ausgangssignale der beiden Tiefpäße sind von der Frequenzverschiebung abgesehen äquivalent mit denen eines Hilberttransformators.

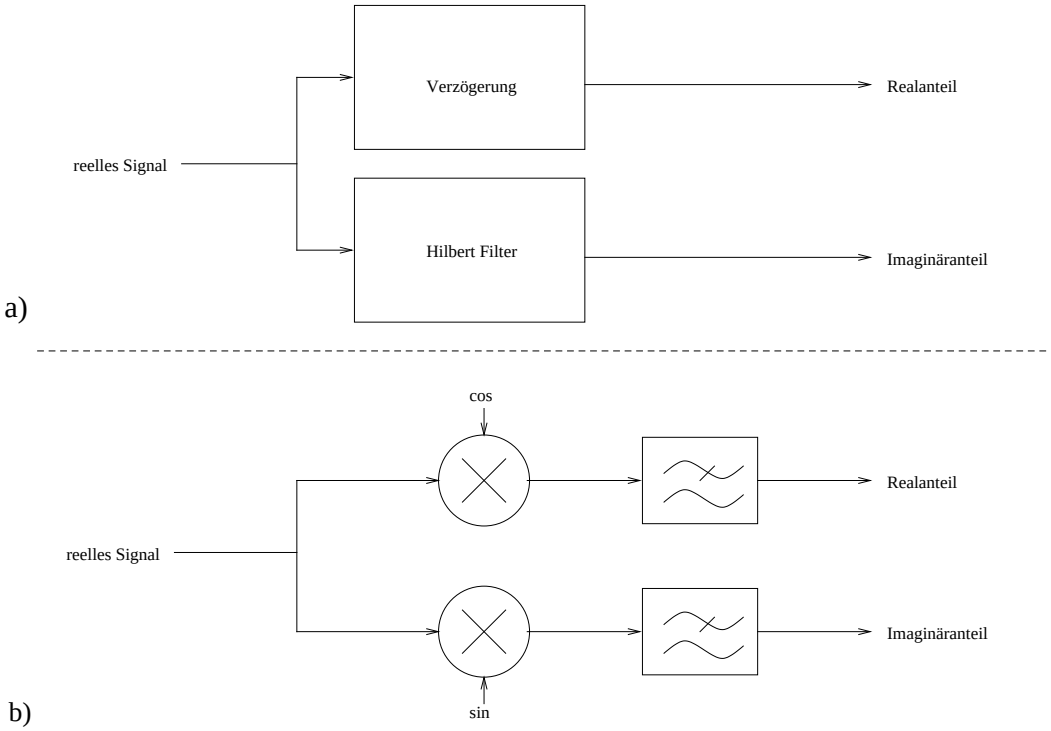

Abbildung 2.12: a) Hilberttransformatorstruktur b) Quadraturmischstruktur mit den dazugehörenden Tiefpaßfiltern

Kapitel 3

Anforderungen und Lösungsansätze

3.1 Wesentliche Anforderungen an den ASIC

Dieses Kapitel befaßt sich mit Fragestellungen, wie benötigte nachrichtentechnische Grundfunktionen schnell und effizient in Hardware umgesetzt werden können. Dabei muß für eine möglichst hohe Flexibilität genau überlegt werden, welche Funktionen in den ASIC implementiert werden sollen und welche von einer aufgesetzten Hardware durchzuführen sind.

Bei diesen Überlegungen war zwischen mehreren Kriterien zu unterscheiden:

- Eine hohe Flexibilität macht zwar das Produkt in verschiedenen Gebieten einsetzbar, erhöht aber auch die Chipfläche und somit die Kosten.

- Ein höherer Funktionsumfang kann eine aufgesetzte Hardware entlasten. Der Mehraufwand im ASIC sollte jedoch immer im Zusammenhang mit der tatsächlichen Rechenzeitersparnis der jeweiligen Endverarbeitungseinheit gesehen werden.

- Die Notwendigkeit späterer Modifikation aufgrund eines unzureichenden Funktionsumfanges sollte vermieden werden, da aufgrund des hohen Entwicklungsaufwandes das Protoyping und die Verifikation teuer und aufwendig sind.

Bei der Entwicklung des ASICs wurde vorab, besonders im Hinblick auf eine möglichst hohe Flexibilität überlegt, welche Funktionen in den ASIC implementiert werden und welche von einer aufgesetzten Hardware ausgeführt werden sollten.
Dabei war offensichtlich, dass versucht wird vorzugsweise alle Funktionen, die schnell abgearbeitet werden müssen, in den ASIC mit möglichst parallelen Teilrealisierungsstrukturen zu implementieren. Für die nicht Geschwindigkeitsrelevanten Aufgaben, kann dann eine aufgesetzte Hardware eingesetzt werden, die in der Regel Vorteile im Bezug auf einer elegantere und schneller zu realisierende Endverarbeitung bietet.
Nach einigen vorab gemachten Überlegungen ist davon auszugehen, dass ein vollsynchrones Design zu entwerfen ist. Wie später noch genauer erörtert wird, resultiert dies aus der Überlegung, in jedem Takt eine andere Teilaufgabe bearbeiten zu müssen, ohne dabei auf zusätzlichen Speicher zurückgreifen zu können.

3.1.1 Grundsätzliche Designentscheidungen

Bevor die Funktionalität eines ASICs festgelegt wird, werden in der Regel Rahmenbedingungen für seinen Einsatz getroffen. Folgende wichtigste Entwurfsentscheidungen für den zu entwerfenden ASICs waren im Hinblick auf die ASIC Hardwarerealisierung von großer Bedeutung:

- die internen Busbreiten und die Anzahl der zu verwendeten Filterkoeffizienten,
- die Art der Signalverarbeitung und
- geeignete Filterstrukturen für einen minimalen Flächenbedarf .

Die VHDL Verhaltensbeschreibung sollte von vornherein auf Skalierbarkeit ausgelegt werden, um eine variable Änderung der Filterkoeffizientenanzahl oder der internen Busbreiten zu ermöglichen. Aufgrund einer Flächenabschätzung einer ASIC Realisierung, sollte für einen ersten Überblick nur eine Version mit 8 Bit Datenbussen und effektiv 25 Filterkoeffizienten realisiert werden. Nach einigen weiteren Überlegungen im Bezug auf die Einsatzgebiete des ASICs, sollten in einer späteren Version die internen Datenbusse auf 14 Bit erweitert werden. Für die meisten Anwendungsgebiete in der digitalen Nachrichtentechnik ist dieses ausreichend.
Zudem wurde grundlegend festgelegt, die zu realisierende Hardwarestruktur überwiegend auf eine analytische Signalverarbeitung auszulegen.
Des weiteren sollten geeignete Filterstrukturen in den ASIC implementiert werden, deren Flächenverbrauch durch Ausnutzung bestimmter Eigenschaften der Filterkoeffizienten als gering gegenüber anderen Realisierungen zu betrachten ist.
Anhand einer Liste möglicher Aufgabengebiete des ASICs, wurden Vorüberlegungen über die zu implementierenden Einheiten gemacht. Eine Zusammenfassung ist in Kapitel 4 zu finden.

3.2 Untersuchungen digitaler Filter in Hardware

In diesem Kapitel sollen Untersuchungen und Realisierungsmöglichkeiten von digitalen FIR Filterstrukturen in Hardware vorgestellt werden, die in dieser Arbeit eine zentrale Rolle spielen. Es folgen Ergebnisse einiger realisierter FIR Filterstrukturen in VHDL. Des weiteren werden für die Verschiebung auf der Frequenzachse und zur Wandlung eines reellen in ein analytisches Signal geeignete DDS Sinus und Cosinus Frequenzgeneratoren in Hardware untersucht und bewertet.

3.2.1 FIR Filter durch direkte Umsetzung der Faltungssumme

Aus der Formulierung der diskreten Faltung im Zeitbereich in Abschnitt 2.8 kann direkt eine Hardwarerealisierung für eine Transversalfilterstruktur (FIR-Filterstruktur) abgeleitet werden. Dabei können grundsätzlich zwei Realisierungsarten unterschieden werden.

- Die serielle Realisierung der Faltungssumme.
- Die parallele Realisierung der Faltungssumme.

Abbildung 3.1 illustriert zunächst die serielle Realisierung der Faltungssumme mit Hilfe zweier *Ringpuffer*, die Speicherbereiche für Samplewerte und Filterkoeffizienten der Länge N darstellen. Zur Adressierung der einzelnen Speicherzellen existieren entsprechende Zeiger auf das gerade zu bearbeitende Element.

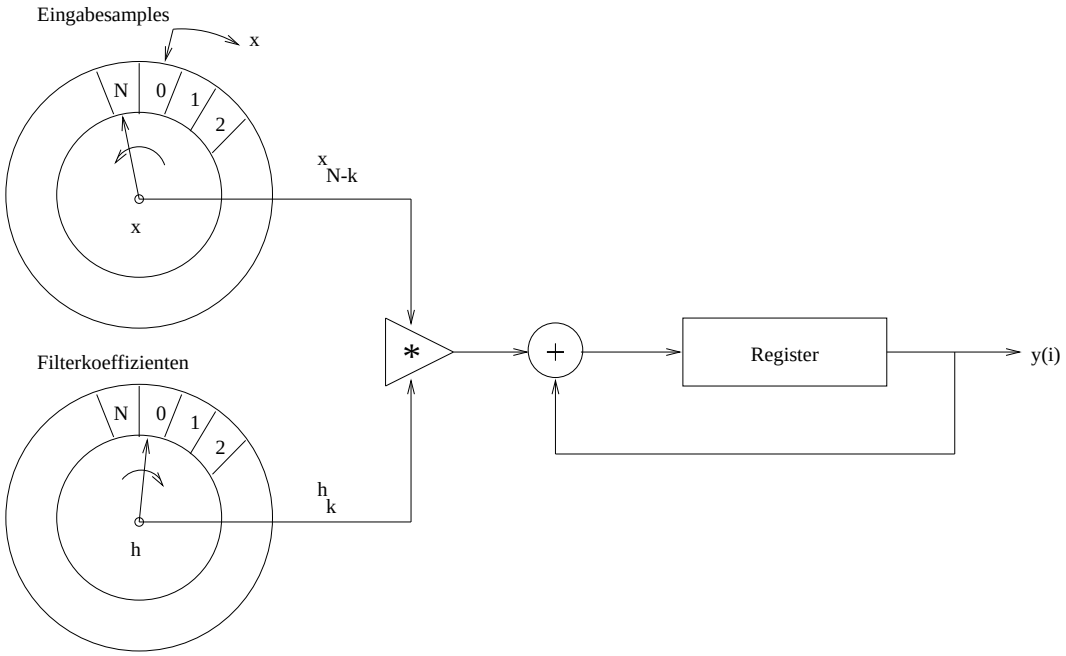

Abbildung 3.1: Serielle Berechnung der Faltungsumme

Es wird davon ausgegangen, dass sich in einem Ringpuffer die Filterkoeffizienten *h* und in dem anderen die Eingangssamplewerte *x* befinden. Filterkoeffizienten und Eingangssamples werden multipliziert. Ein globaler Summierer summiert die Multiplikationsteilergebnisse.
Zur Ermittlung des Ausgangswertes gemäß der Faltungssumme, läßt man beide Ausgabezeiger einmal entgegengesetzt rotieren und summiert alle Teilprodukte. Erst nach einer vollen Rotation der Ringpufferzeiger steht ein voller Faltungssummenwert zur Verfügung.
Legt man einer möglichen parallelen Verarbeitung eine bestimmte Abtastfrequenz zugrunde, so erreicht man bei der seriellen Verarbeitung nur den N- ten Teil davon.
Ein Beispiel soll dieses verdeutlichen:
Wollte man beispielsweise 64 Filterkoeffizienten verarbeiten, so könnte die serielle Realisierung nur noch Abtastfrequenzen f_a von

$$f_a = \frac{1}{64 * \tau_{MAC}} = \frac{1}{6.4ns} \approx 156\,\text{Khz} \tag{3.1}$$

verarbeiten, wenn eine Multiplikation und Additionszykluszeit von 100 ns angenommen wird. Die Multiplikation und Addition wird auch als MAC (Multiply and Accumulate) Operation bezeichnet. Für eine Hardwarerealisierung stellt eine MAC Einheit eine Konstellation aus einem Multiplizierer, einem Addierer und einem Register dar.
Es ist abzuschätzen, dass bei einer seriellen Implementation in Hardware mit nur einer MAC Einheit, sich eine große Flächenersparnis auf Kosten der Geschwindigkeit ergeben würde

[TIE93]. Alle digitalen Signalprozessoren (DSPs) arbeiten bei einer Filterrealisierung nach dem seriellen Prinzip, mit einer oder mehreren MAC Einheiten [LAC95].
Wollte man sehr hohe Verarbeitungsgeschwindigkeiten erreichen, so würde eine serielle Realisierung der Faltungssumme nicht ausreichen. Heutige DSPs werden so kaum an die Geschwindigkeit einer dedizierten Hardware herankommen.

Im nachfolgenden sollen parallele Realisierungen der Faltungssumme vorgestellt und untersucht werden. Dabei soll die direkte parallele Umsetzung in Hardware betrachtet werden [PIR96][KUN85].
Eine allgemeinere Realisierung hierzu zeigt Abbildung 3.2. Die Struktur besteht grob aus einer Filterarithmetik und einer Verzögerungsschaltung. Die Filterarithmetik beinhaltet Multiplizierer und Addierer. Die Verzögerungsschaltung soll parallel mehrere Abtastwerte zur Verfügung stellen, also den aktuellen und den N-1 verzögerten Abtastwert. Feiner aufgelöst soll sie als *Direktform I* bezeichnet werden. Ihr Signalflußgraph ist in Abbildung 3.3 dargestellt. Die Faltungssumme wurde hier in mehrere parallele Einheiten aufgeteilt. Die Verzögerung um ein Taktintervall wird durch Verzögerungseinheiten z^{-1} modelliert, die Multiplikation wird gekennzeichnet durch eine Dreiecksstruktur mit einem Multiplikationssymbol. Die Filterkoeffizienten sind mit *h(x)* bezeichnet.
Nachteilig an der Direktform I und damit an dieser direkten parallelen Umsetzung der Faltungssumme in Hardware, erweist sich der abschließende globale Addierer.
Es ist mit dieser Struktur nicht möglich, die anfallenden Teilergebnisse in einem Takt zu addieren. Das Aufsummieren würde zusätzliche Takte in Anspruch nehmen und damit zu keinem Vorteil gegenüber einer seriellen Realisiererung führen [LAC95].

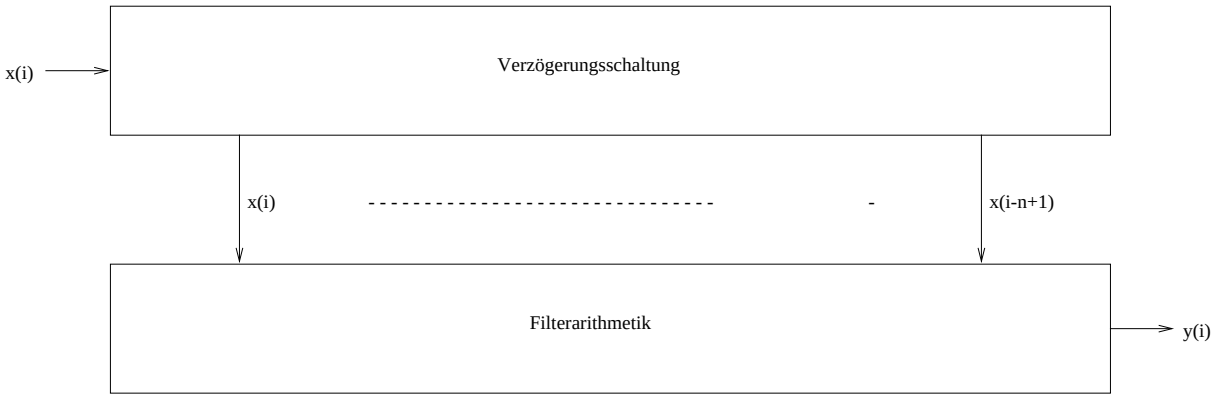

Abbildung 3.2: allgemeiner struktureller Aufbau eines FIR Filters

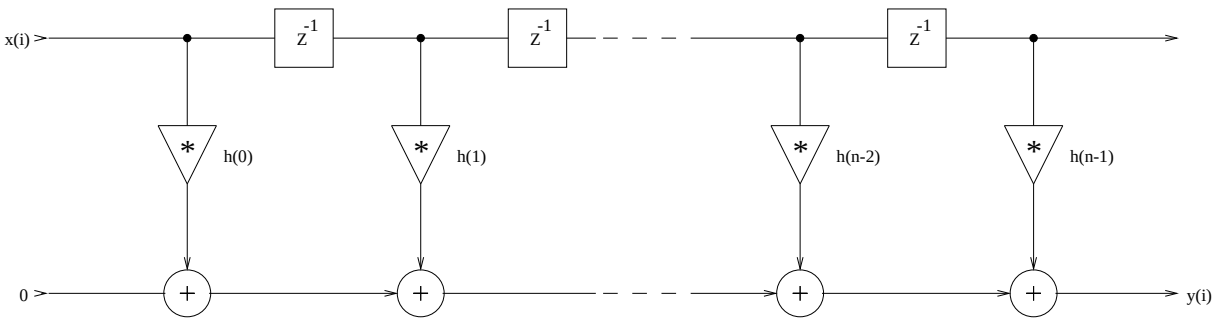

Abbildung 3.3: FIR Direktform I (FirDir1)

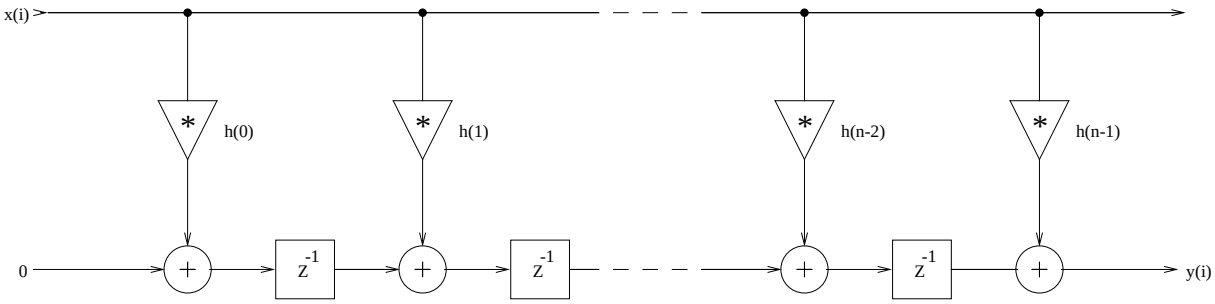

Abbildung 3.4: FIR Direktform II (FirDir2)

Durch die Ausnutzung der *Assoziativität* der Addition kann eine äquivalente Struktur zur Filterrealisierung abgeleitet werden, die im Bezug auf eine Hardwarerealisierung günstiger ist [PIR96]. Dabei kann die Reihenfolge der Addition zur Bestimmung des Ausgangswertes geändert werden. Mit der *Cut-Set Methode* können dazu die Verzögerungselemete vom oberen in den unteren Pfad verschoben werden. Cut Set bedeutet, dass Teilelemente wie der Addierer, der Multiplizierer und das Verzögerungselement beliebig ausgeschnitten und versetzt werden können. Der sich ergebende *Signalflußgraph* wird als *Direktform II* bezeichnet. Charakteristisch ist die Trennung der Addierer durch Verzögerungsglieder z^{-1}. Durch diese Maßnahme wurde der globale Addierer der Direktform I in eine Pipelinestruktur umgewandelt die, wie noch dargestellt wird, im Bezug auf eine Hardwarerealisierung günstiger ist. Die zusätzliche Rechenzeit für die Summation der Multiplikationsteilergebnisse entfällt. Die Anzahl der arithmetischen Einheiten bleibt dabei gleich.
Abbildung 3.4 illustriert graphisch die Direktform II . In Abschnitt 3.3 wird auf diese Filterstruktur und deren Realisierung in Hardware genauer eingegangen.
Zusammenfassend gilt für die eben vorgestellte direkte Umsetzung der Faltungssumme für Hardwarefilterstrukturen:

Das Eingangssignal für die Verzögerungskette ergibt sich bei den vorgestellten Strukturen aus den Eingangssampeln x und allen gewichteten Zwischwerten mit den Filterkoeffizienten h(x). Das Ausgangssignal ist die gewichtete Summe aller Zwischenwerte.

Es soll eine kurze Zusammenfassung des Aufwandes einer seriellen und einer parallelen FIR Filterrealisierung gegeben werden.
Zur Berechnung der diskreten Faltungssumme durch Gleichung 2.23, müssen alle Filterkoeffizienten und die entsprechenden Abtastwerte gespeichert werden. Daraus ergibt sich im seriellen wie auch in einem parallelen Filterrealisierungsfall ein Speicherbedarf von *2N+1* Werten.
Die erforderliche Rechenzeit ist bei der seriellen Realisierung durch die Anzahl der Filterkoeffizienten gegeben. Sie beträgt N+1 Takte. Bei einer parallelen Realisierung beträgt die Rechenzeit nur einen Takt. Die Anzahl der verwendeten arithmetischen Einheiten steigt bei einer parallelen Realisierung auf N Addierer und N+1 Multiplizierer. Im seriellen Fall wird jeweils nur eine Arithmetikeinheit verwendet.
Aufgrund der Notwendigkeit einer schnellen Hardwarerealisierung, wurde die serielle Filterrealisierung für eine weitere Implementation ausgeschlossen.
Tabelle 3.1 fast die Überlegungen für eine serielle und parallele FIR Filter Realisierung noch einmal zusammenfassen.

Verarbeitung	Multiplizierer	Summierer	Rechenzeit	Speicher
parallel	N+1	N	1 Takt	2N+1
seriell	1	1	N+1 Takte	2N+1

Tabelle 3.1: Abschätzung für FIR Filter N-ter Ordnung bei paralleler bzw. serieller Verarbeitung

3.3 Spezielle FIR Filterstrukturen

Wie schon angedeutet, ist für eine Hardwarerealisirung von FIR Filtern nach dem Parallelverfahren die Direktform II in Abbildung 3.4 besonders geeignet. Bei dieser Struktur ist jede *Multiplizier-Akkumulator-Stufe (MAC)* von der nächsten durch ein *Verzögerungsglied* (Register) getrennt. Dadurch steht für jede Operation eine ganze Taktperiode (τ) zur Verfügung. Der globale Addierer einer Direktstruktur I entfällt. Die Verzögerungsglieder der Direktform II ergeben eine *Pipelinestruktur*, da in jedem Takt ein Teil der Faltungssumme berechnet wird.
Es ist leicht einzusehen, dass die Multiplizierer die größte Fläche einer parallelen FIR Filter Realisierung einnehmen. Daher ist man für eine Hardwarerealisierung bestrebt, die Anzahl der Multiplizierer möglichst gering zu halten. Berücksichtigt man die Symmetrieeigenschaften der Filterkoeffizienten der Übertragungsfunktion linearphasiger Systeme, dann lassen sich weitere Filterstrukturen für Direktform I und II ableiten, die mit weniger Multiplizierern auskommen [HES93] [LAC95].
Durch Ausnutzung der Symmetrie der Filterkoeffizienten erhält man für einen parallelen FIR Filter eine deutliche Flächenersparnis, verglichen mit der Gesamtfläche eines Filters mit der selben Anzahl von Filterkoeffizienten, realisiert in der Direktform I oder II. Die Anzahl der Multiplizierer kann fast halbiert werden.
Abbildung 3.5 a) und b) illustrieren die abgeleiteten Direktformen I und II für eine ungerade Anzahl von Filterkoeffizienten. Nach außen verhalten sich diese Filter äquivalent zur entsprechenden Direktform I und II.

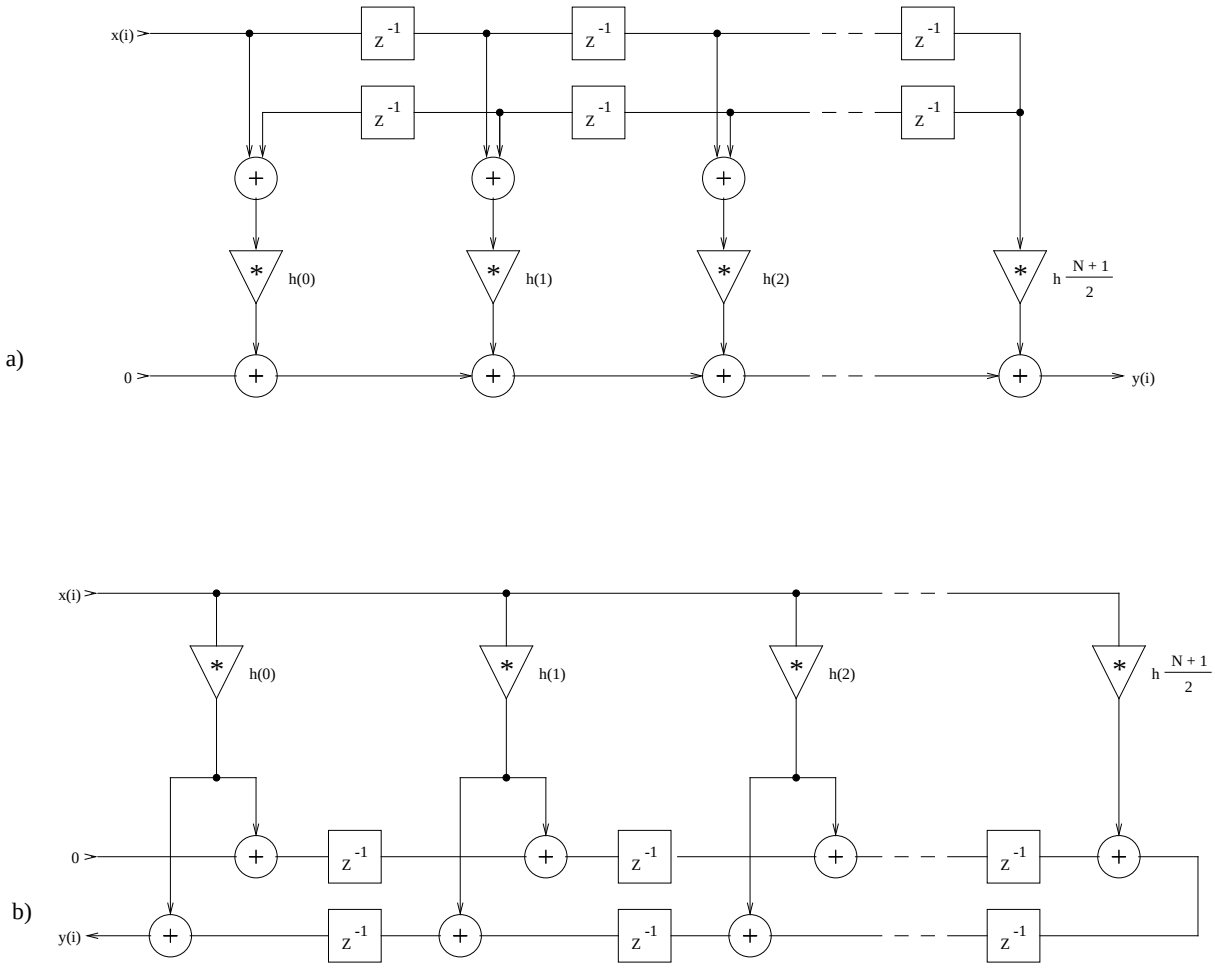

Abbildung 3.5: Modifizierte Signalflußgraphen unter Ausnutzung der Symmetrieeigenschaften für eine ungerade Filterkoeffizientenanzahl (SymFir)
a) Direktform I b) Direktform II

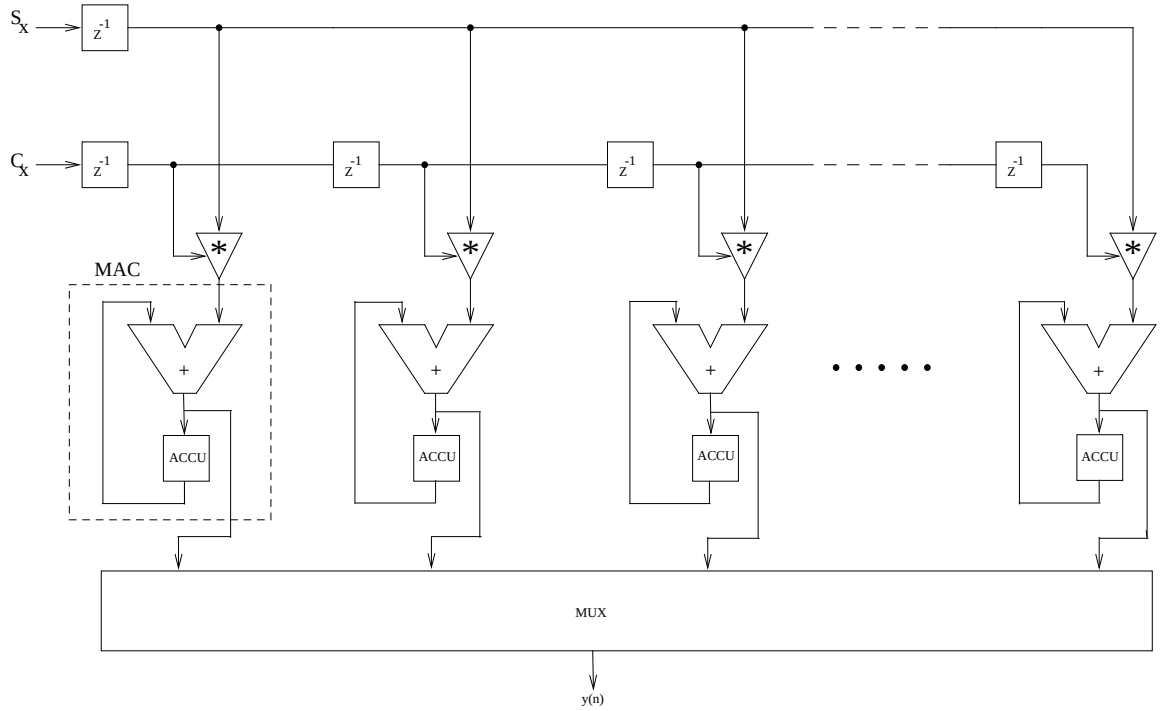

Abbildung 3.6: abgewandelter FIR Filter mit parallelen MAC Einheiten und globalem Ausgangsmultiplexer (MACFir)

Takt	MAC Zelle 0	MAC Zelle 1	MAC Zelle 2	MAC Zelle 3	Faltungsumme
0	$C_3 * S_0$	0	0	0	-
1	$C_2 * S_1$	$C_3 * S_1$	0	0	-
2	$C_1 * S_2$	$C_2 * S_2$	$C_3 * S_2$	0	-
3	$C_0 * S_3$	$C_1 * S_3$	$C_2 * S_3$	$C_3 * S_3$	Zelle 0
4	$C_3 * S_4$	$C_0 * S_4$	$C_1 * S_4$	$C_2 * S_4$	Zelle 1
5	$C_2 * S_5$	$C_3 * S_5$	$C_0 * S_5$	$C_1 * S_5$	Zelle 2
6	$C_1 * S_6$	$C_2 * S_6$	$C_3 * S_6$	$C_0 * S_6$	Zelle 3
7	$C_0 * S_7$	$C_1 * S_7$	$C_2 * S_7$	$C_3 * S_7$	Zelle 0

Tabelle 3.2: Filter Taktschemabeispiel für 4 MAC Zellen

Eine auf den ersten Blick etwas andere Filterrealisierung illustriert Abbildung 3.6.

Der Aufbau und die Funktionsweise kann grob wie folgt charakterisiert werden:

- Der Filter ist aus *N* parallelen MAC Einheiten aufgebaut. Sie bestehen aus einem Multiplizierer und einem Addierer mit einem Register. Die Anzahl der MAC Einheiten bestimmt die Anzahl der verwendeten Filterkoeffizienten C_x.

- Im Gegensatz zur Direktstruktur II werden die Samplewerte S_x gleichzeitig an alle Multiplizierereingänge der MAC Einheiten gelegt.

- Jede MAC Einheit berechnet für sich die volle Faltungssumme.

Durch die sich ergebende MAC Pipelinestruktur können über einen Multiplexer, nacheinander die vollen Faltungssummen ausgegeben werden.
Die genaue Funktionsweise mit Hilfe eines Taktschemas ist für vier Koeffizienten C_x und vier MAC Einheiten in Tabelle 3.1 zusammengefaßt.
Wie der Tabelle zu entnehmen ist, wird im ersten Takt der letzte Filterkoeffizient C_3 gemäß Gleichung 2.24 der ersten MAC Einheit übergeben. Bei jedem weiteren Takt zirkuliert dieser zur nächsten MAC-Einheit. An seine Stelle gelangt jeweils ein neuer Filterkoeffizient.
Hat ein Filterkoeffizient alle MAC Einheiten durchlaufen, gelangt er wieder zur ersten MAC Einheit. Dieses fortlaufende Verschieben der Filterkoeffizienten kann mit Hilfe eines Ringpuffers realisiert werden. Der Filterstruktur übergebene Samplewerte werden in jedem Takt übernommen, während die Filterkoeffizienten zyklisch durch jede MAC Einheiten laufen.
Nach der Berechnung einer vollen Faltungssumme multiplext ein Ausgabemultiplexer (MUX) die einzelnen MAC-Ergebnisse an den Ausgang *y(n)* und löscht den entsprechenden Akkumulator. Ist die Filterpipeline (MAC-Pipeline) voll, so kann bei dieser Filterstruktur in jedem Takt eine volle Faltungssumme ausgegeben werden.
Aus der Struktur ergibt sich, dass die Anzahl der Filterkoeffizienten der Anzahl der benötigten MAC Zellen entspricht.
Ein Vorteil dieser Filterrealisierung ist, dass es zu keiner Ausgabe von noch nicht gültigen Faltungssummenwerten kommt. Hier ist ein Unterschied zu der vorher realisierten Direktform II zu sehen. Aufgrund seiner Gruppenlaufzeit dauert es einige Zeit, bis vollständige Faltungssummenwerte ausgegeben werden. Man spricht in diesem Zusammenhang von einem *Einschwingvorgang* des digitalen Filters [HES93].
Ansonsten verhalten sich diese Filterrealisierung wie die Direktformen. Durch einfache Kaskadierung von MAC Stufen kann die Anzahl der Filterkoeffizienten leicht erhöht werden. Negativ auf die Größe des Filters könnte sich der abschließende Multiplexer auszuwirken. Dieses wird noch zu untersuchen sein.
Alle vorgestellten Hardware-Filterstrukturen weisen folgende Eigenschaften auf:

- einfaches Austauschen der Filterkoeffizienten für *adaptive Filterung*, auch während des Betriebes,

- die Nutzung des Koeffizienteneinganges als zweiten Signaleingang zur Berechnung einer *Korrelationsfunktion* zur Ähnlichkeitsbestimmung zweier unabhängiger Signale im Zeitbereich [KAM96] und

- leichte Kaskadierbarkeit von MAC Einheiten zur Steigerung der Filtergüte.

3.4 Ergebnisse und Bewertung realisierter FIR Filterstrukturen

Im Hinblick auf eine Hardwarerealisierung wurden zur weiteren Untersuchung in der Hardwarebeschreibungssprache VHDL vier Filterstrukturen programmiert, simuliert und für einem 0.6 µm AMS Prozeß auf Logikebene synthetisiert. Um die Synthesezeiten nicht künstlich zu verlängern und um die Filterrealisierungen vergleichen zu können, wurden ausschließlich 8 Bit Sampledaten und 8 Bit Filterkoeffizienten verwendet. Als Daten- und Koeffizientenrepräsentation wurde die 2‘er Komplement-Integer-Darstellung aus Abschnitt 2.10.1 verwendet. Die Anzahl der verwendeten Filterkoeffizienten wurde auf *sechzehn* festgelegt.

Da die Multiplizierstrukturen der parallelen Filterrealisierungen nach vorab gemachten Annahmen die größte Fläche einnehmen, wurde versucht, durch eine spezielle Multiplizierrealisierung den Gesamtflächenbedarf der FIR Filterrealisierungen zu minimieren.
Zu diesem Zweck wurden in einer vorangegangenen Studienarbeit [WAL97] unterschiedliche 8 Bit Addier- und Multiplizierstrukturen für digitale Filterstrukturen vorgestellt und untersucht. Die Realisierungen wurden jeweils einer flächen- und geschwindigkeitsoptimalen Synthese unterzogen. Resultat dieser Arbeit war, dass die arithmetischen Realisierungen gegenüber den optimierten Synopsys-Realisierungen keine nennenswerten Verbesserungen ergaben. Es stellte sich heraus, dass die von Synopsys generierten arithmetischen Realisierungen zum Teil in Fläche und Geschwindigkeit den von Hand entworfenen Realisierungen überlegen waren.
Zu Testzwecken wurde entschieden, eine in Fläche und Geschwindigkeit den Synopsys-Realisierungen nicht sehr unterlegene Multiplizierstruktur, die zudem in der Synopsys Arithmetik Bibliothek nicht enthalten war, in einer Filterrealisierung zu implementieren.
Aus den Ergebnissen der Studienarbeit wurde aus Flächen- und Geschwindigkeitsuntersuchungen eine *Pezarismultiplikationsstruktur* als Schaltnetz Realisierung in Betracht gezogen [PEZ71]. Diese Struktur stellt eine 2‘er Komplementmultipliziererstruktur dar. Auf die genaue Struktur und die Hardwarerealisierung sei auf die Studienarbeit [WAL97] hingewiesen.
Tabelle 3.3 zeigt Zeit- und Flächenoptimal eine Gegenüberstellung mit einem von Synopsys vorgeschlagenen Multiplizierer und zusätzlich einen für die Filterrealisierungen benötigten 19 Bit Addierer auf. Die jeweiligen Namen der von Synopsys gewählten Realisierungen sind unter den Zyklusangaben aufgeführt.
Mit der Erhöhung der Anzahl der zu verarbeitenden Bits des Addierers von eigentlich nur 16 auf 19 Bit und der damit verbundenen Erhöhung der Busweiten zwischen den MAC Zellen, hat es folgende Bewandtnis:
Nach einer Multiplikation entstehen Datenworte mit doppelter Wortbreite (*2w*). Bei 8 Bit Multiplizierern ergibt sich ein 16 Bit Produkt. Nach der Berechnung der Summe bei den Direktstrukturen, kann die Wortbreite nach jeder zweiten MAC Zelle *N* um ein Bit zunehmen. Der benötigte Bus muß daher auf *2w+N/2* erweitert werden. Aufgrund der Festlegung auf 16 Filterkoeffizienten, wurde ein $ld_2 8 = 3$ *Bit* zusätzlicher Datenbus zwischen den MAC Zellen implementiert. Die Gesamtbusweite erhöhte sich somit auf *16+3=19 Bit*.
In der Praxis ist die tatsächliche Zunahme geringer, da die Mehrzahl der Filterkoeffizienten $h_k << 1$ sind.

Arithmetik: 2’erKomplement ohne Setup-/Holdzeiten	PezMul Pezarismultiplizierer 8 Bit		Synopsys Multiplizierer 8 Bit		Synopsys Addierer 19 Bit	
	flächen	zeitoptimal	flächen	zeitoptimal	flächen	zeitoptimal
Fläche:	0.371 mm^2	0.63 mm^2	0.36 mm^2	0.7 mm^2	0.086 mm^2	0.22 mm^2
Zykluszeit:	12.26 ns	6.08 ns	11.44 ns CSA	5.15 ns WALL	11.91 ns RPL	1.16 ns CLA

Tabelle 3.3: Ergebnisse einiger arithmetischer Rechenwerke

Mit den zeit- und flächenoptimalen Logiksyntheseergebnissen hat es folgende Bewandtnis:
Grundsätzlich wird bei einer flächenoptimalen Synthese versucht, eine möglichst kleine Fläche für eine Realisierung zu belegen, während bei einer zeitoptimalen Synthese versucht wird, eine möglichst schnelle Realisierung zu erhalten. Zur Auswahl der Syntheseart kann dem Synthesetool eine Zykluszeit (Timing Konstraint) vorgegeben werden. Je nach der vorgegebenen Zykluszeit entscheidet sich das Synthesetool für eine Realisierung, die in die Kategorie der gemachten Vorgabe paßt.
Wird kein Parameterwert vorgegeben, so führt das Tool aufgrund von voreingestellten Werten automatisch eine flächenoptimale Synthese durch.
Auffallend ist an den gemachten Flächenabschätzungen in Tabelle 3.3, dass die schnellsten arithmetischen Schaltnetze den größten Platz einnehmen.
Als Beispiel sei der Wallace Tree 2'er Komplement Multiplizierer (WALL) zu nennen, der zwar nur eine Verzögerungszeit von nur 5.15 ns aufweist, jedoch auch 0.7 mm^2 Fläche belegt. Eine Multiplizierrealisierung mit einem Carry Save Addierer (CSA) hat dagegen bei einem Flächenbedarf von nur 0.36 mm^2 eine Verzögerung von 11.44 ns.
Ähnliches gilt für den 19 Bit Addierer, bei dem je nach Syntheseart zwischen Ripple Carry Adder (RPL) und einem Carry Look Ahead Addierer (CLA) unterschieden wurde.
Zu klären sind nun die Ergebnisse des Pezaris 2'er Komplementmultiplizierers, der bei der zeitoptimalen Synthese doppelt so schnell ausgefallen ist, obwohl seine Grundstruktur gleich geblieben ist.
Nach der genauen Betrachtung der Syntheseergebnisse auf Logikebene wurde offensichtlich, dass der Geschwindigkeitsvorteil aus einer besseren Optimierung im Bezug auf die Gatterwahl und Stufenzahl der verwendeten Logik resultiert. Scheinbar wurde dieses aufgrund der Anzahl der benötigten Logikgatter erkauft. Die benötigte Logikfläche ist auf fast das doppelte angestiegen.
Zusammenfassend und wie aus den Syntheseergebnissen zu erkennen ist, resultiert in der Regel ein Geschwindigkeitsvorteil immer auf Kosten einer größeren Fläche.
In diesem Zusammenhang sei erwähnt, dass sich die Verzögerungszeitangaben der arithmetischen Realisierungen nur auf die Durchlaufzeiten der für die Berechnung notwendigen Gatter bezieht, da eine zusätzliche Speicherung in Registern im VHDL Code nicht berücksichtigt wurde. Aufgrund der Einhaltung von *Setup*- und *Holdzeiten* der FlipFlops der Register, liegt die tatsächliche Arbeitsgeschwindigkeit weit unter dem angegebenen Wert. Nach Herstellerangaben liegt allein die *Holdzeit* der verwendeten Flipflops des Synthesetools bei ca. *2 ns*.
Die maximale Taktrate wird errechnet, indem das Synthesetool den längsten Pfad zwischen zwei Registern ermittelt und die Einzelverzögerungen der dazwischenliegenden logischen Gatter addiert. Mit Hilfe dieser Setup- und Holdzeiten der verwendeten FlipFlops kann die maximale Taktrate der Schaltung berechnen werden.
Nachfolgend seien die Syntheseergebnisse der realisierten FIR Filter zusammengefaßt. Es sind in allen Fällen zeitoptimale Synthesen durchgeführt worden. Tabelle 3.3 fast die Ergebnisse zusammen.

FIR Filterart: 8 Bit Sample- und Koeffizientendaten 16 Filterkoeffizienten	FirDir2 Synopsys *BK Addierer* und *PezMul* Multiplizierer	SymFir Synopsys *BK Addierer* und *Wall Tree* Multiplizierer	MACFir Synopsys *BK Addierer* und *Wall Tree* Multiplizierer	SynMFir Synopsys *MAC Einheit*
Fläche:	13.18 mm^2	8.7 mm^2	14.46 mm^2	13.7 mm^2
Zykluszeit:	9.03 ns 111 Mhz	7.66 ns 130 Mhz	9.1 ns 110 Mhz	8.81 ns 113.5 Mhz

Tabelle 3.4: FIR Logiksyntheseergebnisse

Tabellarisch aufgeführt werden hier:

- eine Realisierung der Direktform II (FirDir2) mit dem Pezaris 2‘er Komplement-multiplizierer und sein Equivalent unter Ausnutzung der Symmetrieeigenschaften der Filterkoeffizienten (SymFir) mit Synopsys instanzierter Arithmetik und

- eine Filterrealisierung mit parallelen MAC Einheiten (MACFir) und zum Vergleich noch eine Realisierung mit Synopsys generierten und optimierten MAC Zellen (SynMFir).
  Im Prinzip stellt sie die gleiche Struktur wie die MACFir Filterrealisierung dar.

Die Filterergebnisse sind wie folgt zu bewerten:
Wie der Tabelle zu entnehmen und wie vorher schon angenommen, belegt die symetrische Direktform II (SymFir) aufgrund der Ausnutzung der Symmetrieeigenschaften der Filterkoeffizienten die kleinste Fläche. Die Realisierung kommt mit fast der Hälfte der eigentlich benötigten Multiplizierer aus. Der Flächenbedarf wurde hier auf 8.7 mm^2 geschätzt. Zudem stellt sie aufgrund der von Synopsys generierten Arithmetik, mit einer Zykluszeit von 7.66 ns (≈130 Mhz), die schnellste Filterrealisierung dar.
Für die Addierstruktur wählte das Synthesetool einen Brent Kunk Addierer (BK), der scheinbar für 19 Bit und der vorgegebenen Zykluszeit die geeignetste Struktur in Kombination mit dem Wallace Tree 2‘er Komplementmultiplizierer darstellte.
Der Geschwindigkeitsunterschied zu der FirDir2 Filterstruktur resultiert aus der größeren Verzögerungszeit der verwendeten Pezaris Multiplizierstruktur. Prinzipiell sollten die Realisierungen die gleiche Verzögerungszeit aufweisen, da der längste Pfad aufgrund der ähnlichen Struktur bei beiden gleich ist.
Am langsamsten und am größten fiel die MACFir-Realisierung wegen der abschließenden Multiplexstruktur (MUX) aus. Die hierfür benötigte Verdrahtung für 19 Bit Ausgangsdatenleitungen je MAC Zelle belegen eine verhältnismäßig große Fläche.
Die strukturell gleiche Filterrealisierung (SynMFir), mit den von Synopsys generierten MAC Zellen, zeigte aufgrund ihrer optimalen Realisirung bessere Zeit- und Flächenabschätzungen.
Die vom Synthesetool für die zeitoptimale Synthese verwendeten Addier- und Multiplizierstrukturen der Filterrealisierungen wurden, wenn vom Synthesetool angegeben, unter den Strukturnamen aufgelistet.

3.5 Zusammenfassung

Abschließend soll die Vorgehensweise bei der Synthese und eine Zusammenfassung der Ergebnisse der realisierten FIR Filterstrukturen vorgenommen werden. Dazu sei zuerst noch einmal auf die zeit- und flächenoptimale Logiksynthese eingegangen:

Prinzipiell ist eine Logiksynthese mit dem verwendeten Synopsys Synthesetool im Bezug auf eine zeitoptimale oder flächenoptimale Resultate durch *Timing Constrains* möglich. Soll eine flächenoptimale Synthese durchgeführt werden, so versucht das Synthesetool eine möglichst kleine Fläche für die Realisierung zu belegen. In der Regel fallen die Ergebnisse dann mit niedrigen Taktraten auf.
Wird eine zeitoptimale Synthese durchgeführt, so haben die Strukturen zwar eine kürzere Zykluszeit, belegen jedoch auch in der Regel eine größere Fläche.
Bei der Synthese fand bei allen Filterrealisierungen die Option „zeitoptimale Synthese“ Verwendung. Zu diesem Zweck wurde dem Synthesetool ein Timing Constrain von *5 ns* fest vorgegeben. Die Information, in wieweit dieses von Synthesetool eingehalten wurde, konnte mit dem Kommando „*report timing*“ und der dazugehörige Flächenbedarf mit „*report area*“ aufgezeigt werden. Abstriche werden und wurden beim *Timingreport* als „*Slack (Violated)*“ angegeben.
Eine wichtige Erkenntnis im Bezug auf Timing Constraint Angaben soll im folgenden erörtert werden.
Es hatte sich herausgestellt, dass es nicht sinnvoll ist, die Timing Constrain Angabe intuitiv zu wählen. Es zeigte sich, dass man bessere Schaltungsergebnisse erhält, wenn man die Randbedingungen nicht auf unmögliche Werte (z.B. 0 ns) setzt, sondern Werte nahe des Machbaren verwendet. Dies liegt unter anderem daran, dass bei unmöglichen Angaben die Gewichtungen bei den heuristischen Methoden der Synthesewerkzeuge ungünstig gesetzt werden [LEH94].
Das Fehlen der Angabe eines Timing Constrains äußerte sich in einer flächenoptimalen Synthese und der Verwendung von Ripple Carry Addierern (RPL) anstelle der schnelleren Carry Look Ahead (CSA) oder anderer langsamer Addierrealisierungen der Synopsys Standardbibliothek.
Die Entscheidung welche Syntheseart durchgeführt werden soll, hängt entscheidend vom Einsatzgebiet und weiteren ökonomischen Faktoren ab.
Besondere Beachtung sollten Timing Constraint Angaben finden, wenn aus der *Synopsys-Design-Ware-Komponentenbibliothek* Funktionen oder fertige Realisierungen genutzt werden. In dieser Bibliothek befinden sich Komponenten, wie verschiedenartige Addierer, Multiplizierer, Dividierer, MAC Zellen oder sogar fertige Speicher [SYN97]. Auch Hardwarerealisierungen für Spezialfunktionen stehen zur Verfügung. Je nach den gewünschten Randbedingungen wählt das Synthesetool, die nach den voreingestellten Parametern passende Realisierung, von den zur Verfügung stehenden aus. Bei fehlenden Angaben können sich ungewollte Strukturen ergeben, die vielleicht zu einer vorher gemachten Spezifikation nicht passen. Zur Wahl geeigneter Realisierungen ist die Information über die Implementierungsalternativen, die Synopsys zur Verfügung stellt, hilfreich [SYN97].
Wie erwähnt, wurde versucht, Grundkomponenten eines digitalen Filters, wie Addierer, Multiplizier oder MAC Einheiten, aus einzelnen Logikgattern aufzubauen, in der Hoffnung, besseres Ergebnis für die Zeit- und Flächenabschätzung zu erhalten.
Eine nach den Ergebnissen einer vorher durchgeführten Studienarbeit entworfenen Multiplizierstruktur, die sich dort im Bezug auf Zeit und Fläche als geeignet für eine Filterrealisierung in Hardware einsetzen ließe, wurde versuchsweise in einer Filterrealisierung

implementiert. Das Resultat zeigte allerdings nicht akzeptable Ergebnisse und wurde daher verworfen. Auf die Implementation eigener Addierstrukturen wurde verzichtet.
Probleme mit dem Synthesetool und der Addierrealisierung gab es dahingehend, dass es auch bei kurzen Timing Constraint Angaben langsame Addierstrukturen wie Ripple Carry Adder verwendet wurden. Um dieses Problem zu umgehen, mußte dem Synthesetool mit einem „*set_implementation*“ Befehl explizit angegeben werden, einen bestimmten Addierertyp zu verwenden.
Zur weiteren Realisierung des ASICs fiel die Entscheidung zu Gunsten der SymFir Filterstruktur mit von Synopsys generierten Multiplizier- und Addierstrukturen. Sie wurde aufgrund der höheren Takt- und der relativ kleinen Flächenabschätzung gewählt.
Die maximale Taktfrequenz, die das Synthesetool für diese Struktur angab, lag bei 130 Mhz und einer Fläche von 8.7 mm^2.

Der längste Pfad bei allen Filterrealisierungen wurde vom Synthesetool zwischen dem Sampleregister und dem Summenregister einer MAC Zelle ermittelt. Aus der Verzögerungszeit des dazwischenliegenden Multiplizierers und Addierers mit den jeweiligen Setup und Holdzeiten der Register konnte die maximale Taktrate einer MAC Zelle und damit des gesamten Filters errechnet werden.

Eine gemachte Erkenntnis war:
Ein Entwickler sollte sich nach diesen Ergebnissen genau überlegen, ob es sich lohnt, die Zeit für den Aufbau eigener arithmetischer Logik zu investieren. Der Meinung des Autors nach sollte dieses nur geschehen, wenn diese für Spezialaufgaben bestimmt sein sollen, oder wenn eine genaue Kenntnis des Aufbaus einer arithmetischen Einheit gewünscht wird.

3.6 Realisierung eines DDS Generators in Hardware

Wie schon im Abschnitt 2.11 angesprochen, soll das Grundprinzip des zu realisierenden DDS-Generators die digitale Berechnung diskreter Signalwerte S_k an gewünschten äquidistanten Abtastzeitpunkten t_k sein. Die wichtigsten Überlegungen seien hier noch einmal rekapituliert.
Im Falle der benötigten diskreten Sinusgenerierung muß allgemein gelten:

$$S_k = \sin(\omega t_k + \varphi). \tag{3.2}$$

Es sind somit zunächst die Abtastpunkte t_k mit $t_{k+1} = t_k + \tau_a$ und danach die resultierenden S_k zu bestimmen.
Durch Umformung ergibt sich aus Gleichung 3.2:

$$S_k = \sin(\Psi_k) \quad mit \quad \Psi_k = \omega t_k + \varphi. \tag{3.3}$$

Die Berechnung der Ψ_k kann dann direkt nach

$$\Psi_{k+1} = \Psi_k + \Delta\Psi \tag{3.4}$$

mit dem Phasenschritt $\Delta\Psi = \omega\tau_a$ durchgeführt werden.
Die übliche Realisierung der DDS, basierend auf Gleichung 3.4, kann mit einem *Phasenakkumulator* und einem *Phasenschritt* $\Delta\Psi$ durchgeführt werden.
Die Berechnung von $\sin(\Psi_k)$ ist auf verschiedene Weise möglich.

Aufgrund der hohen Abtastraten ist man allerdings auf eine Sinus Tabellenrealisierung angewiesen. Da die Sinustabelle nur eine endliche Anzahl von diskreten Sinuswerten repräsentiert und in der Regel kleiner ist als der volle Addressierungsbereich des gesamten Phasenakkumulators, werden nur die obersten *m* Bits für die Tabellenadressierung herangezogen. Diese entsprechen nur einer ungenauen Repräsentation von Ψ. Die eigentliche Berechnung des nächsten Phasen Wertes geschieht mit der vollen Genauigkeit des N-Bit Phasenakkumulators. Wie in den nächsten Abschnitten noch angesprochen, kann zur Erhöhung der Genauigkeit des zu generierenden Sinussignals eine Tabelleninterpolation oder eine größere Tabelle eingesetzt werden.
Die zur Adressierung der Tabelle verwendeten *m* Phasenbits repräsentieren eine volle Sinusperiode. Die verwendeten 2^m diskreten Sinuswerte entsprechen genau der 2π Periode. Eine mögliche Hardwarerealisierung bis hin zu einem analogen Sinus Ausgangssignal illustriert Abbildung 3.7.
Eine Hardwarerealisierung besteht zur Hauptsache aus dem *N-Bit Phasenakkumulator* und der *Sinus Lookup Table*. Für die analoge Ausgabe des Sinus Signals sind zusätzlich ein Digital-Analog Wandler und ein Tiefpaßfilter erforderlich. Diese beiden Komponenten sind vollständigkeitshalber aufgezeigt und werden bei einer späteren ASIC Realisierung nicht implementiert, da sie für den zu realisierenden ASIC nicht relevant sind.

Die Funktionsweise der DDS Hardwarerealisierung kann wie folgt veranschaulicht werden:
Der Phasenschritt $\Delta\Psi$ wird in einem Frequenzregister gehalten und bei jedem Abtasttakt f_a zu dem alten Phasenakkumulatorwert hinzuaddiert.
Wie der Graphik zu entnehmen ist, addressieren die obersten *m* Bits (MSB) des Phasenakkumulators die Sinus Lookup Table, in der sich die digitale Sinusrepräsentationen befinden.
Die unteren Teilbilder von Abbildung 3.7 sollen von links nach rechts das aufakkumulieren der Phasenakkumulators, das Auslesen diskreter Sinuswerte und eine anschließende D/A Wandlung mit Tiefpaßfilterung für ein analoges Sinussignal darstellen.
Sie entsprechen den Funktionen der darüberliegenden Teilblöcke.
Es sei darauf hingewiesen, dass sich durch diese Art der Signalgenerierug, durch einfaches Austauschen der Tabellenwerte, auch andere Wellenformen erzeugen lassen.
Aufgrund des benötigten digitalen Sinussignals für die ASIC Realisierung, ist nur der *Digital Direkt Synthesizer* Block zu realisieren.

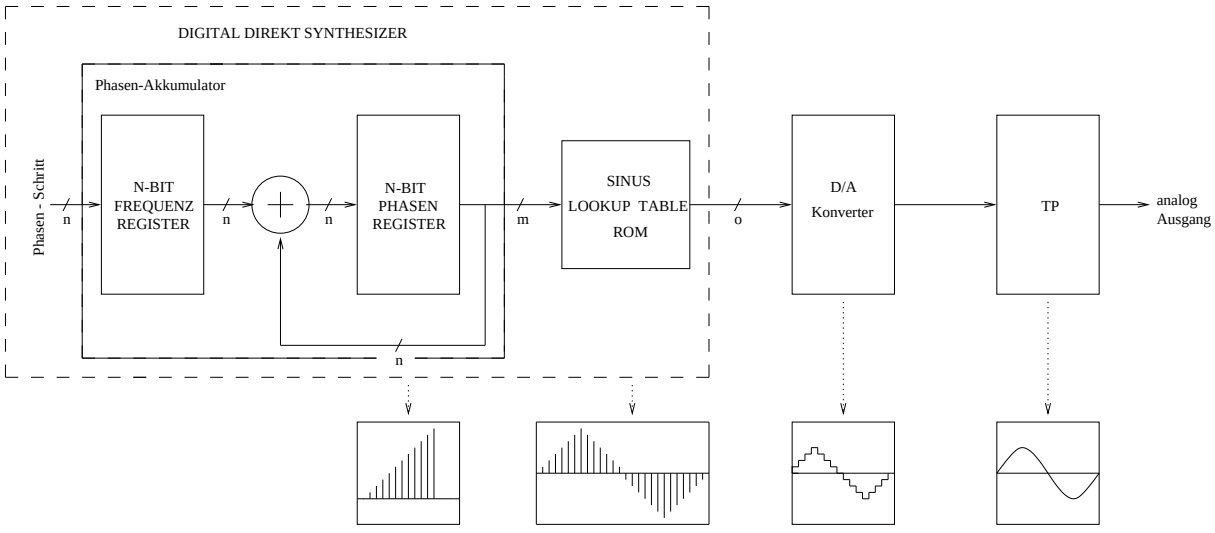

Abbildung 3.7: DDS Hardware Komponenten zur Erzeugung eines analogen Sinussignals

Um ein Signal einer bestimmte Frequenz zu erzeugen muß das *Phasenschrittregister* vorab mit einem *Phasenschrittwert* geladen werden.
Die *Ausgabefrequenz* f_{out} kann wie folgt errechnet werden:

$$f_{out} = \frac{f_a * \Delta\varphi}{2^N} \tag{3.5}$$

wobei f_a der *Abtastfrequenz*, $\Delta\varphi$ dem *Phasenschritt* und N der Anzahl der Bits des Phasenakkumulator entsprechen. Die Größe der Phasenschritts und die dadurch erzeugte Frequenz sind proportional.
Um beispielsweise eine Frequenz von 7.5 Mhz zu erzeugen, bei einer willkürlich gewählten Abtastfrequenz von 30Mhz und einem 32 Bit Phasenakkumulator, muß der Phasenschritt (7.5Mhz * 2^{32}/30Mhz) = 1073741824 betragen.
Die *Frequenzauflösung* Δf, also die minimale Schrittweite ergibt sich aus:

$$\Delta f = \frac{f_a}{2^N} \tag{3.6}$$

Als Beispiel soll ein 31.25 Mhz Abtasttakt und der *N=32* Bit Phasenakkumulator dienen. Die resultierende Frequenzauflösung beträgt hier *0.0072 Hz (31.25Mhz/2^{32})*. Es ist also möglich die Frequenz in *0.00727 Hz* Schritten zu ändern.

3.7 Berechnung der Anzahl der Tabelleneinträge

Im nachfolgenden soll die Anzahl der diskreten Sinuswerte *n* für einen digitalen 8 Bit DDS Sinus/Cosinus-Generator berechnet werden Es sollen weiter zwischen DDS Realisierungen mit und ohne Interpolation unterschieden werden.
Die 8 Bit Sinuswerte soll dann mit einigen Abwandlungen, im Bezug auf die Ausnutzung der Symmetrieeigenchaften einer Sinusschwingung, in verschiedene DDS Realisierungen implementiert werden.

3.7.1 Fehlerrechnung ohne Tabelleninterpolation

Die Winkelschritte Δy (Abstände) der einzelnen Werte in einer digitalen Sinus Tabelle für eine volle Periode mit *n* Elementen errechnen sich aus:

$$\Delta y = \frac{2\pi}{n} \tag{3.7}$$

Werden n = 2048 Sinuswerte (11 Bit) angenommen, entspricht dies einem Winkelschritt von $2\pi/2^{11} = 0.0015$.
Der maximale Fehler Δ, umgerechnet auf die benötigte Sinuswert Bitrepräsentation ergibt:

$$\Delta = ld(1/\sin\frac{2\pi}{n}) \tag{3.8}$$

mit $2\pi/n$ als Winkelschritt einer Sinustabellenrepräsentation.

Um den sich ergebenen Fehler für die benötigte 8 Bit Repräsentation des Sinus möglichst gering zu halten, muß die Anzahl der Sinuswerte in der Tabelle *n=2048* betragen.
Genauer gerechnet entsprechen 2048 Sinus Werte 8.3 Bit. Der Fehler der sich aus der Anzahl der diskreten Sinuswerte ergibt ist damit niedriger als gefordert.
Die diskreten Sinuswerte für eine volle Periode von 2π und der Auflösung von 8 Bit wurden mit Matlab und einer speziellen Funktion berechnet. Da Matlab intern mit mehr als 8 Bit Genauigkeit rechnet, mußten die Ergebnisse für die Hardwarerealisierung nachträglich skaliert werden.

3.7.2 Fehlerrechnung mit Tabelleninterpolation

Durch Tabelleninterpolation ist es möglich, einen Wert zwischen zwei gegebenen Stützstellen der Sinustabelle zu berechnen, um die „wahre“ Signalform besser anzunähern. Durch diese Maßnahme kann die Qualität des zu generierenden Signals und damit des daraus resultierenden Spektrums verbessert werden [ZÖL97].
Es ist grundsätzlich möglich, verschiedene Interpolationsarten, die aus der Mathematik bekannt sind, einzusetzen [BRO81] [BAR91]. Die meisten benötigen in Hardware aufgrund der benötigten Arithmetik einen großen Flächenbedarf und sind für diese Arbeit nur bedingt geeignet. Bei einer Tabellenrepräsentation des Sinussignals könnte die lineare und die Taylorinterpolation in Betracht gezogen werden. Die Taylorinterpolation bietet sich insofern an, da die benötigte erste Ableitung *f‘* der Sinusfunktion in den Tabellenwerten schon vorhanden ist und nur ausgelesen werden muß. Aufgrund der zusätzlich benötigten arithmetischen Hardware soll hier jedoch nicht weiter auf diese eingegangen werden.

Es soll die lineare Tabelleninterpolation weiter untersucht und auf eine mögliche Realisierung eingegangen werden.
Abbildung 3.8 illustriert allgemein mit Hilfe einer linearen Interpolation, zwischen zwei gegebenen Stützstellen (x_i,x_{i+1},y_i,y_{i+1}) einen Zwischenwert (x,y) zu errechnen.

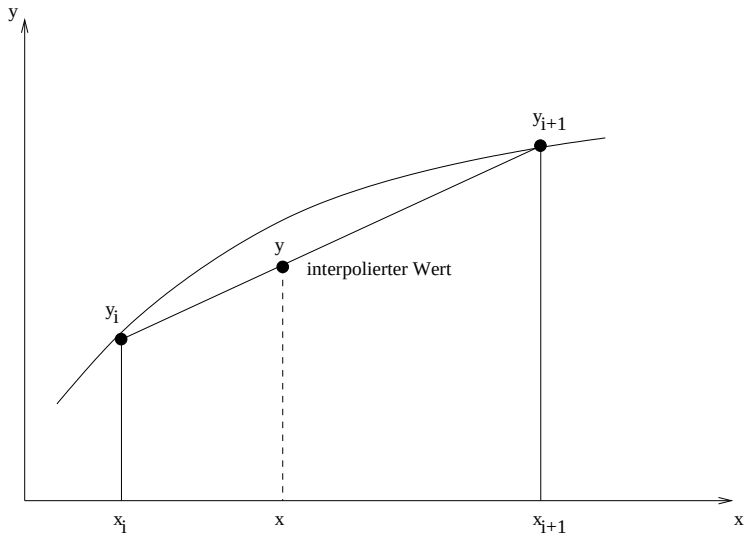

Abbildung 3.8: allgemeine graphische Darstellung der linearen Interpolation

Die benötigte Arithmetik kann in Hardware bei der Annahme, dass ein Zwischenwert genau zwischen zwei Stützstellen bestimmt werden soll, leicht und schnell durch eine Verschiebung der entsprechenden Bits ersetzt werden [LAG87].
Unter der zuvor gemachten Annahme, kann der linear interpolierte Zwischenwert y mathematisch durch folgende Gleichung errechnet werden:

$$y = \frac{y_{i+1} - y_i}{2} + y_i \tag{3.9}$$

Durch einfache Umformung vereinfacht sich die Gleichung zu:

$$y = \frac{y_i + y_{i+1}}{2} \tag{3.10}$$

Die benötigte Division durch *zwei* kann für alle 2‘er Potenzen leicht durch ein Verschieben der Bits nach rechts durchgeführt werden. Diskrete Werte, die außerhalb der Mitte der beiden Stützstellen liegen, können mit dieser Methode nicht berechnet werden.
Hier kann nur eine andere Interpolationsart mit einem größeren Aufwand Abhilfe schaffen [BAR91] [BRO81].

3.8 Implementationsmöglichkeiten und Ergebnisse

Zur weiteren Untersuchung und für einen abschließenden Vergleich, wurden äquivalent zu den digitalen Filterstrukturen, mehrere DDS Generatoren mit verschiedenen Tabellenlängen in VHDL programmiert und auf Logikebene synthetisiert. Die Grundprinzipien der DDS Generatoren sind bei allen Realisierungen als gleich anzusehen.
Die Realisierungen der DDS Generatoren wurden in VHDL nach Abbildung 3.8 ohne den D/A Wandler und den abschließenden Tiefpaßfilter durchgeführt. Zusätzlich wurde bei einer Realisierung ein um 90° phasenverschobener Cosinus abgeleitet. Für die Cosinus Generierung wurde lediglich ein Offset zur Tabellenstartadresse der Sinustabelle addiert, d.h. ein Phasenstartwert von ½ π vorgegeben. Die Einbindung erwies sich als unproblematisch.
Im nachfolgenden sollen die realisierten DDS Generatoren genauer vorgestellt und bewertet werden.
Sie unterscheiden sich durch folgende Merkmale:

- volle Sinustabellenrepräsentation (2π),
- ¼ der Sinustabellenwerte und Spiegelung der einzelnen Hälften (Hybrid) und
- Ausnutzung eines Spezialfalls der linearen Intertpoaltion.

Die Hauptunterschiede der realisierten DDS Generatoren liegen überwiegend in der Anzahl der Sinuswerte und dem Ansprechen der Sinus Repräsentationstabellen. Bei der DDS Realisierung mit linearer Interpolation mußte zusätzlich arithmetische Logik in den DDS-Funktionsgenerator eingebunden werden.
Der Phasenakkumulator war bei allen Realisierungen 32 Bit groß.

Um die Synthesezeit zu verkürzen und aufgrund der 2‘er Komplementdarstellung der Sinuswerte wurden bei dem DDS Generator mit voller 2π Sinustabelle *n = 1024* Sinuswerte verwendet. Dieses entspricht ungefähr den 8 Bit 2‘er Komplementwerten der Sinustabelle.
Die diskreten Sinuswerte wurden nicht wie vielleicht üblich in ein *ROM (Read Only Memory)* gespeichert, sondern direkt in die Logik übernommen. Flächen und Geschwindigkeit für einen ROM-DDS-Generator wurden aus Zeitgründen nicht weiter untersucht.

Typ: 8 Bit Sinuswerte	DDS 1024 Sin Werte 2π Periode	DDS256 256 Sin Werte ¼ 2π Periode	DDSI128 128 Sin Werte ½ π Periode und lin. Interpolation	DDS256SinCos 256 SinCos Werte ¼ 2π Periode
Fläche:	1.85 mm^2	1.64 mm^2	1.9 mm^2	1.71 mm^2
Zykluszeit:	5.9 ns = 145Mhz	6.09 ns = 144 Mhz	7.3 ns = 136 Mhz	6.1 ns = 144 Mhz

Tabelle 3.5 : DDS Syntheseergebnisse

Tabelle 3.5 zeigt vier realisierte 8 Bit DDS Generatoren mit entsprechenden Flächen und Geschwindigkeitsabschätzungen. Die *DDS256SinCos* Generator Variante entspricht der DDS256 Realisierung. Sie ist allerdings um einen digitalen 8 Bit Cosinus Signalausgang erweitert.
Interessant ist der Vergleich zwischen dem *DDS256* und dem *DDS256SinCos* Generator. Die zusätzliche Cosinus Generierung verändert nicht wesentlich die mögliche Taktrate des gesamten DDS Generators. Das Cosinus Signal erhält man quasi gratis, ohne das zusätzlich Fläche benötigt wird.
Der Flächenbedarf der *DDSI128* ist aufgrund der zusätzlichen arithmetische Logik für die lineare Interpolation am größten, obwohl die hierfür benötigte Sinustabelle lediglich 128 Werte enthält. Dieses gilt allerdings nur für die hier verwendeten kleinen Tabellenlängen.
Zur Synthesezeit kann gesagt werden, daß sie beim DDS Generator mit vollen 2π Sinuswerten am längsten ausfiel, da die Sinuswerte direkt in die Logik übernommen wurden und das Synthesetool somit eine längere Optimierungszeit benötigt. Wie schon im Vorfeld vermutet, erhielt der DDS Generator mit ¼ der Periode die beste Flächen- Taktabschätzung.
Die Flächen- und Geschwindigkeitsabschätzung wurde von dem verwendeten Synthesetool als erste Näherung angegeben. Als Syntheseprozeß fand wie bei den Filterrealisierungen der AMS 0.6 µm Prozeß Verwendung.

3.9 Untersuchung und Bewertung der erzeugten Signale

Zur weiteren Untersuchung wurde der generierte Sinus einer *Spektralanalyse* unterzogen. Abbildung 3.9 zeigt das resultierende Spektrum in einer Dezibel (dB) Skala. Es wird hier ein 8 Bit Sinussignal von 3.75 Mhz aufgezeigt, bei einer Abtastfrequenz f_a von 31.25 Mhz.

Mit der Untersuchung des Spektrums in Dezibel hat es folgende Bewandtnis:
In der Regel werden Signale durch den Signal-Rauschabstand SNR (Signal to Noise Ratio), oder auch Störabstand genannt, in dB beschrieben. Der SNR ist der Quotient aus der Leistung des Signals ohne Rauschen und der Leistung des Rauschens.

Nach [TIE93] muß der Signal-Rauschabstand *S* rechnerisch für 8 Bit bei:

$$S = 20dB \lg \frac{U_{s\,eff}}{U_{r\,eff}} = N*6dB + 1.8dB \approx N*6dB \qquad (3.11)$$

⇨ 8 Bit * 6dB = *48dB* liegen.

Ein Vergleich mit Abbildung 3.9 bestätigt diese Rechnung. Die Störfrequenzen liegen alle unter – 48dB.

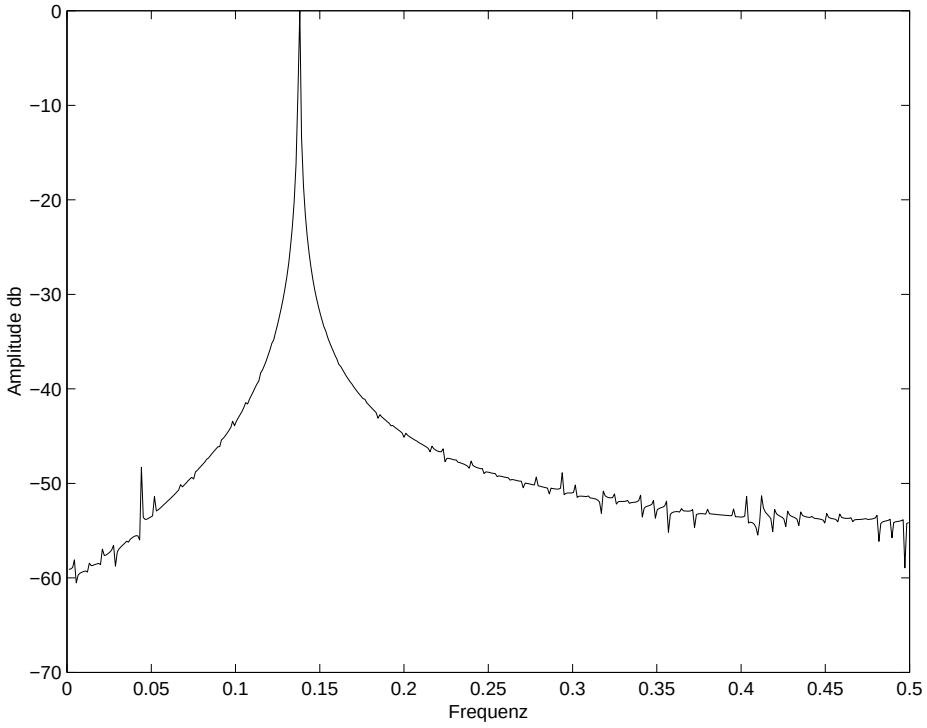

Abbildung 3.9: 8 Bit DDS Sinus Spektrum in dB

Die Qualität des Ausgangsspektrums des digitalen Sinussignals hängt zusammenfassend von folgenden Faktoren ab:

- Anzahl der Phasenbits *m* bei einer Realisierung mit einer Sinustabelle
- Wortbreite mit dem der Sinus berechnet wird
- Interpolationsmaßnahmen

Aufgrund der Vorteile, insbesondere bei der Fläche, soll der DDS Sinus/Cosinus Generator *DDS256SinCos* mit ¼ der Sinusperiode ohne lineare Interpolation in den ASIC als eigenständige Einheit implementiert werden.
Die Realisierung stellt einen eigenständigen Funktionsblock dar, der nur der diskreten Signalerzeugung dienen soll.

Kapitel 4

Übersicht über den ASIC

In diesem Kapitel soll eine grobe Übersicht über den zu entwickelnden ASIC gegeben werden, dessen Struktur sich überwiegend aus Vorüberlegungen und den sich daraus resultierenden Konsequenzen ergab.
Der zu realisierende ASIC läßt sich grob in verschiedene Funktionseinheiten (Blöcke) aufteilen, die miteinander verbunden sind. Zum Teil bilden mehr als zwei Funktionsblöcke eine eigenständige Funktionseinheit.
Im weiteren Verlauf der Arbeit wird nach einer „Top-Down“ Struktur vorgegangen. Diese Vorgehensweise kam auch bei der Hardwarerealisierung des ASICs zum Einsatz.

Es sollen grob im Sinne einer *Spezifikation*, die *Aufgaben* der einzelnen Funktionsblöcke und ihre *Beziehung* zueinander dargestellt werden. Dabei soll allerdings nicht auf die genaue algorithmische Realisierung der Einzelkomponenten in der Hardwarebeschreibungssprache VHDL eingegangen werden.
Eine detaillierte Darstellung über die funktionellen Zusammenhänge der einzelnen Funktionsblöcke findet man in Kapitel 5.

Beim Entwurf des ASICs wurden einzelne Teilkomponenten realisiert, ihre Funktionsweise überprüft und zu einem größeren Teilblock zusammengefaßt. Zur Verifikation der einzelnen Teilkomponenten und des Gesamtchips durch Simulation, wurden spezielle Testumgebungen erstellt. Mit ihnen konnte deren Funktionsweise leicht überprüft werden. Das hier angewandte Vorgehen wird von der Hardwarebeschreibungssprache VHDL explizit unterstützt [MAE93].

Die wichtigsten Überlegungen im Bezug auf eine ASIC Hardwarerealisierung seien noch einmal im Sinne einer Spezifikation zusammengefaßt.

- Es ist ein ASIC zu entwickeln, das in der Lage ist, hochfrequente nachrichtentechnische Signale direkt zu verarbeiten. Dabei kann der ASIC anfallende Datenraten soweit reduzieren, dass eine aufgesetzte Hardware eine Endverarbeitung oder Auswertung durchführen kann. Der ASIC könnte in diesem Fall als Vorverarbeitungseinheit in der digitalen Nachrichtentechnik eingesetzt werden.

- Es soll eine Verarbeitung hoher Bandbreiten möglich sein. Als Basis sollte daher die direkt zu verarbeitende Samplerate des ASICs entsprechend hoch ausfallen.

- Es soll eine voll skalierbare Architektur entworfen werden.

- Oberste Priorität im Hinblick auf die maximale Taktrate und den daraus resultierenden Flächenbedarf soll die maximale Größe des ASICs haben, die 100 mm^2 nicht überschreiten sollte.

- Die Filterkoeffizientengröße sollte mindestens 8 Bit betragen.
- Die Leistungsaufnahme ist als unkritisch zu betrachten.

Tabelle 4.1 fast die wichtigsten Vorgaben noch einmal zusammen.

Taktfrequenz	maximal mögliche
Gesamtgröße	< 100 mm^2
Koeffizientengröße	≥ 8 Bit
Leistungsaufnahme	unkritisch

Tabelle 4.1: Vorgaben des Chips

Eine effiziente und schnelle Implementierung von Teilkomponenten zu einer Gesamtrealisierung stand zudem als Herausforderung an vorderster Stellt.

Zur Realisierung dieses Vorhabens sollen die drei wichtigsten Grundideen aus der Einleitung sollen noch einmal rekapituliert werden.

1) Multiratensignalverarbeitung mit variablen Datenraten.

2) Analytische Verarbeitung von Signalen, bei der nicht auf Spiegelfrequenzen geachtet werden muß.

3) Rekonfigurierbarkeit in jedem Sampletakt.

Der erste Punkt führt zu folgender Konsequenz.

- Zur Wandlung eines analogen in ein digitales Signal existieren schnelle A/D Wandler (Analog/Digital Wandler), die in der Lage sind, in einer Sekunde mehrere hundert Megasamples zu erzeugen [TIE93]. In der digitalen Nachrichtentechnik will man in der Regel aus diesen digitalen Sampledaten eine oder mehrere Informationen I extrahieren.
  Probleme sind bei vielen digitalen Signalverarbeitungseinheiten gegeben, die nicht in der Lage sind, diese hohen Datenraten direkt zu verarbeiten. Abhilfe könnte hier eine Unterabtastung des Eingangssignals bringen.
  Die dahinterstehende Idee ist folgende:
  Zur Auswertung einer bestimmten Information I wird nicht jeder Samplewert benötigt. Daraus folgt, dass trotz der Unterabtastung aus einem Signal eine bestimmte Information I extrahiert werden kann, indem nur jeder *n'te* Samplewert verwendet wird.
  Zur hardwaretechnischen Realisierung bietet sich eine *Multiratenfilterkaskade* [FLI93] an. Das Wort „Multiratenfilter" resultiert aus den unterschiedlichen Datenraten am Eingang beziehungsweise Ausgang des Filters.

Wie im nächsten Kapitel noch ausführlich dargestellt wird, ist es durch den Einsatz einer Multiratenfilterkaskade möglich, die Datenrate und Bandbreite eines Signals um den Faktor *m* zu reduzieren. Zudem ist das hierdurch realisierte *resampling* im Bezug auf eine aufgesetzte Hardware von großer Bedeutung, da diese Einheiten in der Regel nur mit geringen Datenraten

arbeiten. Es kann eine *Datenratenanpassung* für andere nachrichtentechnische Einheiten durchgeführt werden.
Ein weiterer Vorteil ist, dass bei einer Multiratenfilterung mit vergleichsweise wenigen Filterkoeffizienten gearbeitet werden kann. Für äquivalente schmalbandige Filterungen, ohne den Einsatz der Multitatenfiltertechnik, würden viel mehr Filterkoeffizienten benötigt.
Auf den Aufbau der Multiratenfilterkaskade und weiterer Hintergründe, wird in Kapitel 5 noch genauer eingegangen.

Der zweite Punkt führt zu folgender Konsequenz.

- Da das Nyquist Abtastthoerem durch die zum Einsatz kommende Dezimation (Unterabtastung) verletzt wird, würden sich zwangsläufig Probleme mit den Spiegelfrequenzen ergeben. Auf Grund dessen ist eine reelle Verarbeitung von Signalen mit der Multiratenfiltertechnik nur eingeschränkt durchführbar. Ausgehend von den in Abschnitt 2.6 angesprochenen Nachteilen einer reellen Verarbeitung von Signalen, soll die Hardwarestruktur des ASICs auf eine analytische Signalverarbeitung ausgelegt werden.
  Zu diesem Zweck muß das reelle Signal in ein komplexes, d.h. in ein äquivalentes Signal mit Real- und Imaginäranteilen gewandelt werden. Der realisierten Hardwarearchitektur ist es durch spezielle Komponenten möglich, die Wandlung vor der eigentlichen Verarbeitung des Signals durchzuführen.

Der dritte Punkt führt schließlich zu folgender Konsequenz.

- Um eine hohe und in jedem Takt duchführbare Konfigurierbarkeit zu ermöglichen, soll die Gesamtarchitektur des ASICs so ausgelegt werden, dass eine zentrale Steuereinheit die internen Abläufe koordiniert. Mit Hilfe dieser Steuereinheit, die als endlicher Automat FSM (Finite State Machine) realisiert wurde, ist es möglich, in jedem Sampletakt eine Umkonfiguration der realisierten Teilkomponenten (Einheiten) durchzuführen. Es besteht so die Möglichkeit, in jedem Takt einen anderen Dezimationsfilter rechnen zu lassen. Damit ist man in der Lage, in jedem Takt einen anderen Frequenzbereich (Kanal) zu bearbeiten.

Zusammengefaßt soll der Steuerautomat folgende Aufgaben erfüllen:

- Steuerung der einzelnen Dezimationsfilter,

- Steuerung der Datenmultiplexer und damit des internen Datenflusses und

- Konfiguration der übrigen Teileinheiten zu bestimmten Zeitpunkten.

Die Implementation des endlichen Steuerautomaten soll den eigentlichen Schlüssel für die Vielseitigkeit und Flexibilität der realisierten Hardwarestruktur darstellen.

Im nachfolgenden soll in Abbildung 4.1 eine detaillierte Struktur der ASICs angegeben werden. Die optional benötigten analogen Komponenten außerhalb der ASICs werden hier nicht betrachtet.

Der ASIC kann grob in folgende Funktionsblöcke aufgeteilt werden:

- zwei Dateneingangsmultiplexer (DataMUX) zur Realisierung der Multiratenfilterkaskade aus einzelnen Dezimationsfiltern,
- zwei Multiratenfilterblöcken zur getrennten Verarbeitung der real und imaginär Anteile eines komplexen Signals,
- zwei Quadraturmischer,
- der Arithmetik der komplexwertigen Systems. Sie besteht aus vier Multiplizierern, einem Subtrahierer und Addierer,
- dem DDS Sinus/Cosinus Generator (Digital Direkt Synthese),
- dem Steuerautomaten (Steuer-FSM) zur internen Konfiguration und Steuerung der realisierten internen Komponenten des ASICs und
- den Ausgangsmultiplexern (OutMUX).

Die beiden Multiratenfilterkaskaden sind über eine *komplexwertige Hardwarestruktur* miteinander gekoppelt [KRO97]. Sie stellt die Basis für die eigentliche Verarbeitung komplexer Signale dar. Auf deren Implementierung wird in Abschnitt 5.2 noch genauer eingegangen.
Die Verarbeitung eines analogen Eingangssignals läßt sich grob in Teilaufgaben, die außerhalb des ASICs und Teilaufgaben die im ASIC selbst durchgeführt werden, aufteilen.

Zuerst soll die optionale Komponente vorgestellt werden:
Sie kann im Grunde genommen nur aus einem A/D Wandler (analog/digital Wandler) bestehen, der mit einem Aliasfilter ausgestattet ist, so das die Nyquistbedingung auf jeden Fall eingehalten wird.
Der Wandler entsprechender Wortbreite wird mit einer festen Samplefrequenz betrieben. Da in dem ASIC eine synchrone Signalverarbeitung zur jeweiligen Samplerate des Wandlers durchgeführt wird, entspricht die Samplerate der Taktrate des ASICs. Auf den A/D Wandler soll in dem weiteren Verlauf dieser Arbeit nicht näher eingegangen werden. Es sei auf weiterführende Literatur verwiesen [TIE93] [ZÖL97].
Die Aufgaben der digitalen Komponenten im ASIC sollen nachfolgend erläutert werden.

Die Dateneingangs- und Ausgangsmultiplexer:
Die Dateneingangsmultiplexer (DataMUX) multiplexen die Sampledaten des A/D Wandlers oder die Daten der einzelnen Dezimationsfilter der Multiratenfiltertkaskade.
Aufgrund der analytischen Signalverarbeitung existiert jeweils ein Dateneingangsmultiplexer für den Real- und den Imaginär- Multiratenfilterblock. Zur beachten ist, dass es nur dem Dateneingangsmultiplexer des Real- Multiratenfilterblocks möglich ist, neue Sampledaten zu übernehmen. Vergleiche Abbildung 4.1. Der Grund soll in Kapitel 5 angesprochen werden.
Gleichzeitig können über die Ausgangsmultiplexer (OutMUX) Dezimationsfilterdaten getrennt als Real- und Imaginäranteile ausgegeben werden.

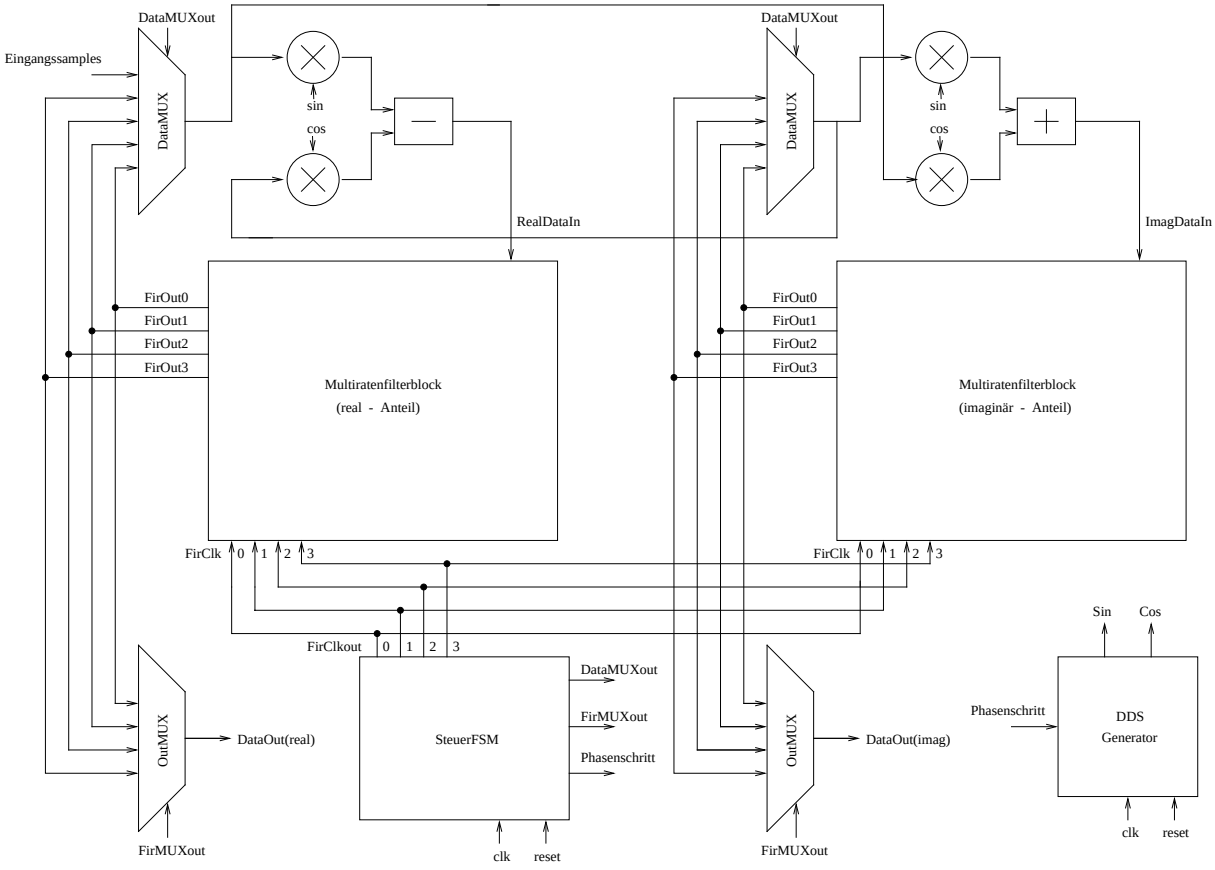

Abbildung 4.1: Grundstruktur des ASICs

Nach der Selektion der Quelle für die Eingangsdaten und der wahlweisen Ausgabe schon verarbeiteter Daten durch entsprechende Multiplexstrukturen, ist es nötig, die Komponenten zur Wandlung des reellen Eingangssignals in ein komplexes Signal zu erörtern. Die Idee war die Wandlung mit Hilfe von Quadraturmischern durchzuführen, die zusätzlich andere oft gebrauchte Aufgaben in der Nachrichtentechnik übernehmen können.

Die Quadraturmischer:
Die doppelt ausgelegten Quadraturmischer sind in Kombination mit einem Frequenzgenerator (DDS Generator) und jeweils zwei Multiplizierern zu sehen. Man erinnere sich, dass zur Wandlung das reelle Eingangssignal mit einem erzeugten digitalen Sinus- und einem um 90° phasenverschobenen Cosinussignal multipliziert werden muß. Die Quadraturmischer können dabei zwei Aufgaben erfüllen:

- Wandlung eines reellen in ein analytisches Signal durch Quadraturabwärtsmischung und
- Versatz des Eingangssignals in eine höher- oder niederfrequentere Lage.

Welche der beiden möglichen Arbeitsweisen dynamisch im jeweiligen Takt durchgeführt werden soll, wird maßgeblich durch den Steuerautomaten bestimmt. Kapitel 5 wird darauf genauer eingehen.
Die nach der Wandlung benötigten Tiefpaßfilter zum Wegfiltern eines Spiegelfrequenzanteils, sind in den beiden Multiratenfilterblöcken realisiert. Vergleiche Abschnitt 2.12.

Zum DDS Generator:
Der DDS Generator (Digitale Direkt Synthese) ist zur Generierung eines digitalen Sinus und eines um 90° Grad in der Phase verschobenes Cosinus Signals zuständig.
Die in diesem Zusammenhang zu nennenden Quadraturmultiplizierer multiplizieren diese digitalen Signale mit den Ausgangsdaten der Datenmultiplexer (DataMUX). Der DDS Generator kann durch den Steuerautomaten in jedem Takt in Frequenz und Ausgabezeitpunkt umkonfiguriert werden. Der DDS Generator und die Quadraturmultiplizierer können als eine Einheit gesehen werden.

Das komplexwertige System:
Für die komplexe Verarbeitung von Signalen ist in dem ASIC eine komplexwertige Hardwarestruktur implementiert. Sie basiert auf der Umsetzung der komplexen Multiplikation in Hardware. Wie in Kapitel 5 noch zu sehen, setzt sie für eine parallele Umsetzung allerdings voraus, dass alle Verarbeitungseinheiten doppelt ausgelegt werden. Vier Multiplizierer, ein Addierer und ein Subtrahierer stellen diese komplexwertige Hardwarestruktur dar. Für eine analytische Verarbeitung von Signalen wird der ASIC über den Steuerautomaten so konfiguriert, dass eine komplexe Multiplikation in Hardware durchgeführt wird. In Kapitel 5 wird hierauf näher eingegangen.

Zur Multiratenfilterkaskade:
Nach der Wandlung in einen Real- und Imaginäranteil, gelangen diese Signale getrennt in je einen Multiratenfilterblock. Jeder für sich besteht aus einer Kaskade von je vier Dezimationsfiltern. Der Steuerautomat generiert zu diesem Zweck getrennt die Taktsignale für die Register der jeweiligen Teilfilterstruktur. Durch diese Maßnahme können die Dezimationsfilter in jedem Takt getrennt arbeiten.
Die jeweiligen Ausgabedaten (FirOut) der Dezimationsfilter können wahlweise über den Eingangsmultiplexer rückgeführt oder über den zusätzlichen Datenausgangsmultiplexer (Out-MUX) getrennt als Real- und Imaginär- Anteile ausgegeben werden.

Der endliche Steuerautomat (Steuer-FSM):
Ein implementierter endlicher Zustandsautomat stellt die eigentliche Steuereinheit der ASICs dar. Durch ihn ist eine Umkonfiguration der realisierten Komponenten des ASICs in jedem Takt möglich. Die Steuer-FSM arbeitet bei der Steuersignalgenerierung synchron zu dem jeweiligen Sampletakt des A/D Wandlers. Eine ihrer Hauptaufgaben ist die Erzeugung der Taktsignale für die Multiratenfilterkaskaden und die Steuerung der Eingangs- bzw. Ausgangsmultiplexer. Die Steuer FSM stellt die Basis für die hohe Flexibitität des ASICs dar.

Der ASIC soll intern ausschließlich digitale Daten verarbeiten. Es existieren keine internen Speicher, die eventuell Sampledaten für eine verzögerte Verarbeitung zwischenspeichern können. Aus dieser Konsequenz resultiert für den ASIC, dass in jedem Sampletakt eine Teilverarbeitung durchgeführt werden muß.

Zusätzlich zur analytischen Verarbeitung von Signalen besteht die Möglichkeit, über entsprechende Multiplexstrukturen, die in Abbildung 4.1 aus Übersichtlichkeitsgründen nicht dargestellt sind, die beiden Multiratenfilterblöcke zu kaskadieren. Durch diese Maßnahme ist es möglich, eine reelle Verarbeitung von Eingangssignalen durchzuführen. Auch hierauf wird in Kapitel 5 noch genauer eingegangen.

4.1 Zusammenfassung

Die Grundidee einer analytischen Signalverarbeitung wurde mit Hilfe der Quadraturmischer, und der Umsetzung der komplexen Multiplikation in Hardware verwirklicht. Die Verarbeitung hochfrequenter Signale und die Anpassung an langsamere Einheiten durch die realisierte Multiratenfilterkaskade. Zudem konnte eine hohe Flexibilität mit Hilfe eines endlichen Steuerautomaten verwirklicht werden, der eine Rekonfiguration der realisierten Teilkomponenten in jedem Takt erlaubt. Das ASIC erhält erst durch den Steuerautomaten das hohe Maß an Flexibilität, das in der Spezifikation gefordert wurde.
Die folgenden Kapitel gehen detailreicher auf die Einbindung der Multiratenfilterkaskade, die Realisierung der komplexwertigen Hardwarestruktur und auf die Steuerung durch den Steuerautomaten ein. Zum besseren Verständnis soll ein Beispiel die Flexibilität der Hardwarestruktur in Kombination des realisierten Steuerautomaten illustrieren.

Kapitel 5

Realisierung und Implementation

In diesem Kapitel werden einige, in den vorherigen Abschnitten realisierte und untersuchte Hardwarestrukturen, zu größeren Funktionseinheiten zusammengefaßt. Es wurden dabei nur Strukturen ausgewählt, die sich als geeignet im Bezug auf die gemachte Spezifikation erwiesen haben. Des weiteren wird auf ihr Zusammenspiel näher eingegangen.
Als einer der wichtigsten Teilblöcke des ASICs, wird der Aufbau und die Implementation der realisierten *Multiratenfilterkaskade* erläutert. Auf deren Einbindung in eine *komplexwertige Hardwarestruktur* soll dann im weiteren Abschnitt eingegangen werden. Die Registerbelegung, Programmierung des endlichen Steuerautomaten und das Timingkonzept des gesamten ASICs mit einem Beispiel zur Ablaufsteuerung runden dieses Kapitel ab.

5.1 Aufbau und Einbindung der Multiratenfilterkaskade

Nachdem in Abschnitt 2.8 die Grundlagen von FIR Filtern behandelt wurden, ist bekannt, dass die Anzahl der benötigen Rechenoperationen bei nichtrekursiven digitalen Filtern (FIR–Filtern) direkt proportional zur Abtastfrequenz f_a [HES93] steigt. Es soll daher im Interesse einer Hardwarerealisierung versucht werden, die Abtastfrequenz möglichst gering zu halten.
Wie schon angedeutet, besteht eine Möglichkeit diesem Ziel näher zu kommen, in der Verwendung der *Multiratenfiltertechnik* [FLI93].
Für die ASIC Realisierung wurde die Einbindung zweier *Multiratenfilterkaskaden* in Kombination einer *komplexwertigen Hardwarestruktur* zur Verarbeitung komplexer Signale berücksichtigt. Die Multiratenfilterkaskaden haben die Aufgabe, eine Datenreduktion und damit eine Entlastung der Filterarithmetik durchzuführen. Die eigentliche auszuwertende Information I sollte durch die zum Einsatz kommende und in Abschnitt 2.5 eingeführte *Unterabtastung* ihren vollen Informationsgehalt nicht verlieren.
Als positiver Nebeneffekt ist zudem die Verschiebung der *Spektralanteile* der zu verarbeitenden Information, in das Basisband möglich. Vergleiche dazu Abschnitt 2.7.
Die Unterabtastung digitaler Signale soll des weiteren als *Dezimation* bezeichnet werden.

Für die ASIC-Hardwarearchitektur wurden je vier *Dezimationsfilterstufen* verteilt auf zwei Multiratenfilterblöcke realisiert. Real- und Imaginärsignalanteile werden dabei getrennt behandelt. Ein Multiratenfilterblock beinhaltet jeweils einen Dezimationsfilter in jeder Stufe. Untersuchungen ergaben, dass die Anzahl der Dezimationsfilterstufen aufgrund der Einsatzgebiete als ausreichend zu bewerten ist.

Es sollen zuerst folgende wichtige Aspekte einer Multiratenfilterung genauer besprochen werden:

- Auswirkungen eines Multiratenfilters auf die Filtersteilheit digitaler Filter und
- der Bandbreite eines Signals.

In jeder Stufe eines Multiratenfilterblocks kann die Anzahl der Eingangssamples um den Faktor *m* geschmälert werden. Daraus ergibt sich für eine Kaskade von *n* Stufen, dass eine bestimmte Anzahl von Eingangssamples auf ein m^n–tel der ursprünglichen Sampleanzahl reduziert wird. Für eine nachträgliche arithmetische Einheit resultiert daraus eine Erniedrigung der zu verarbeitenden Datenrate und damit eine Entschärfung der Geschwindigkeit auf ein m^n–tel der ursprünglich benötigten Taktrate.
Interessant wird dieser Aspekt im Zusammenhang mit der Filtersteilheit von digitalen Filtern. Die realisierte Multiratenfilterkaskade schmälert nicht nur die Datenrate, sondern erhöht auch die Steilheit der Filterflanken der in der Multiratenfilterkaskade verwirklichten Dezimationsfilter. Damit ist es möglich, mit einer kleinen Anzahl von Filterkoeffizienten sehr steile Filter zu realisieren.

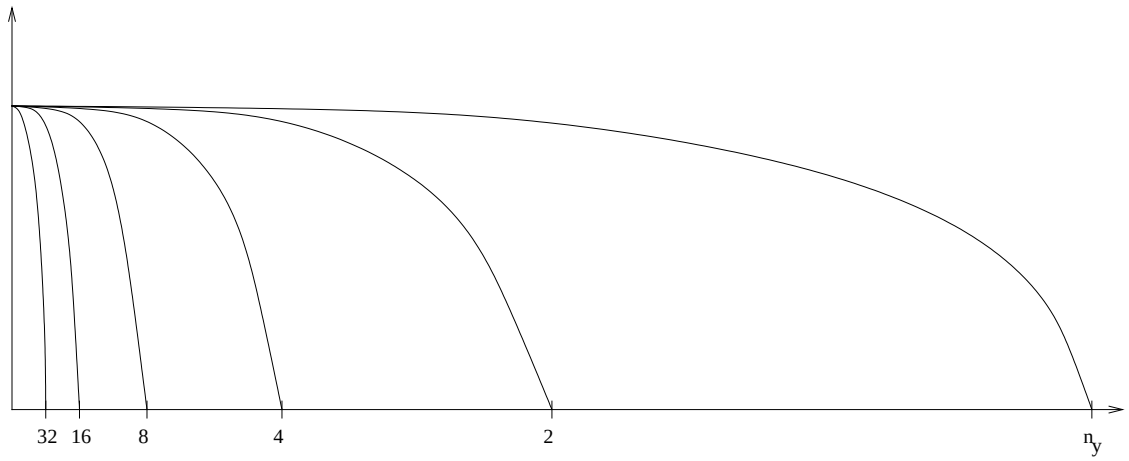

Abbildung 5.1: Filtersteilheit eines Tiefpaßfilters mit verschiedenen Dezimationsgraden

Zur Illustration zeigt Abbildung 5.1 den Frequenzgang eines Tiefpaßfilters mit nur wenigen Filterkoeffizienten und verschiedenen *Dezimationsgraden* von 2, 4, 8, 16, 32. Es wird zuerst mit einem *Dezimationstiefpaßfilter* vorgefiltert und anschließend *dezimiert*. Zu erkennen ist die immer größer werdende Steilheit der Filterflanken nach jedem Dezimationsschritt, ohne dabei die Anzahl oder die Genauigkeit der Filterkoeffizienten zu erhöhen.
Dieser Effekt resultiert aus der fortlaufenden Dezimation, da der Bereich bis zur Nyquistfrequenz (n_y) entsprechend des Dezimationsgrades verkleinert wird.
Wie in Abbildung 5.1 zu sehen, resultieren aus dieser Technik sehr schmalbandige Filter durch einfache *Kaskadierung* von Dezimationsfilterstufen. Um mit einer gewöhnlichen Filterrealisierung eine Filtersteilheit, die einer Dezimation von 32 entspricht zu erhalten, würden wesentlich mehr Filterkoeffizienten benötigt.
Es ist allerdings zu beachten, dass nach jedem Dezimationsschritt das Signal immer weiter verzöget wird.

Beachten wir nun die Auswirkungen kaskadierter Dezimationsfilterstufen auf die Bandbreite eines Nutzsignals.
Abbildung 5.2 a) illustriert dazu mehrere Bandpaßspektren, aus denen in Teilbild b) durch einen Bandpaßfilter ein bestimmtes Spektrum herausgefiltert wurde. Ausgehend von Teilbild b) zeigt Teilbild c) eine Dezimation um den Faktor 4 und Teilbild d) eine weitere Dezimation um den Faktor 16. Zu erkennen ist hier die *Schmälerung* der Bandbreite nach den jeweiligen Dezimationen. Das Bandpaßsignal hat dabei bis auf einen *Skalierungsfaktor*, der durch den *Dezimationsgrad* bestimmt wird, die gleiche Gestalt.

Durch Dezimation ist demnach nicht nur die Datenratenreduktion zu erzielen, sondern auch eine Schmälerung des Eingangsfrequenzbandes um den Faktor *m*. Äquivalent zur vorher besprochenen Erhöhung der Filtersteilheit durch die Sampleratenreduktion, bringt eine Kaskade von *n* Stufen das Eingangsspektrum auf ein m^n–tel der Bandbreite des ursprünglichen Spektrums.
Praktisch ist es mit dieser Technik möglich, eine bestimmte Frequenz sehr genau aus einem Frequenzspektrum herauszufiltern [WIE98].
Die Möglichkeit einer fortlaufenden Dezimation wurde im ASIC durch die implementierten Multiratenfilterkaskaden in Kombinaton mit den jeweiligen Eingangsdatenmultiplexern (DataMUX) realisiert.

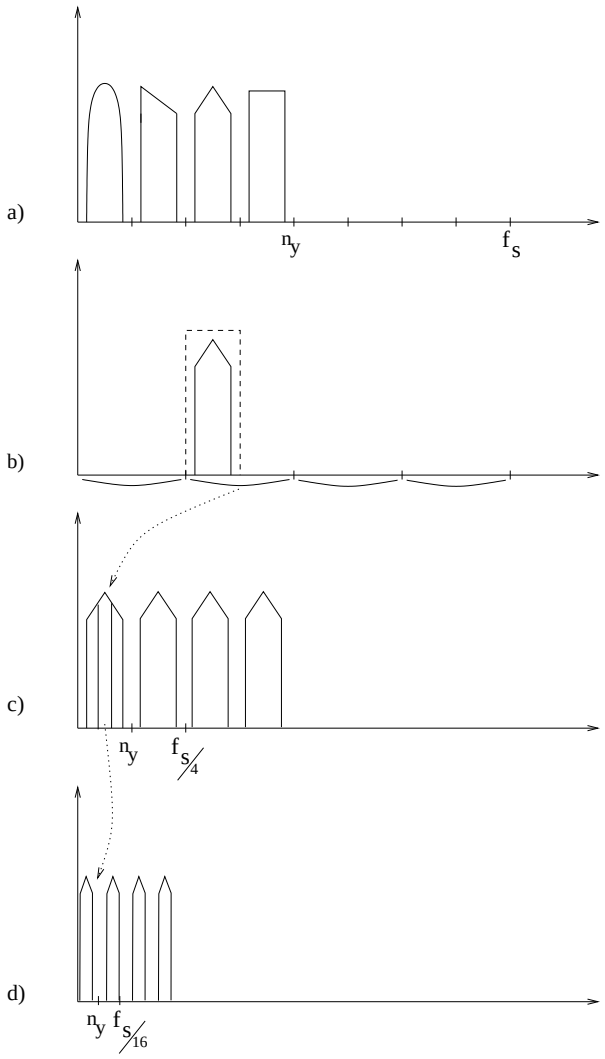

Abbildung 5.2: Auswirkungen der Kaskadierung von mehreren Dezimationsfiltern auf die Bandbreite eines analytischen Bandpaßsignals

Im nachhinein soll der genaue Aufbau eines realisierten Multiratenfilterblocks in Hardware besprochen werden. Abbildung 5.3 soll dabei zur Veranschaulichung den internen Aufbau illustrieren.

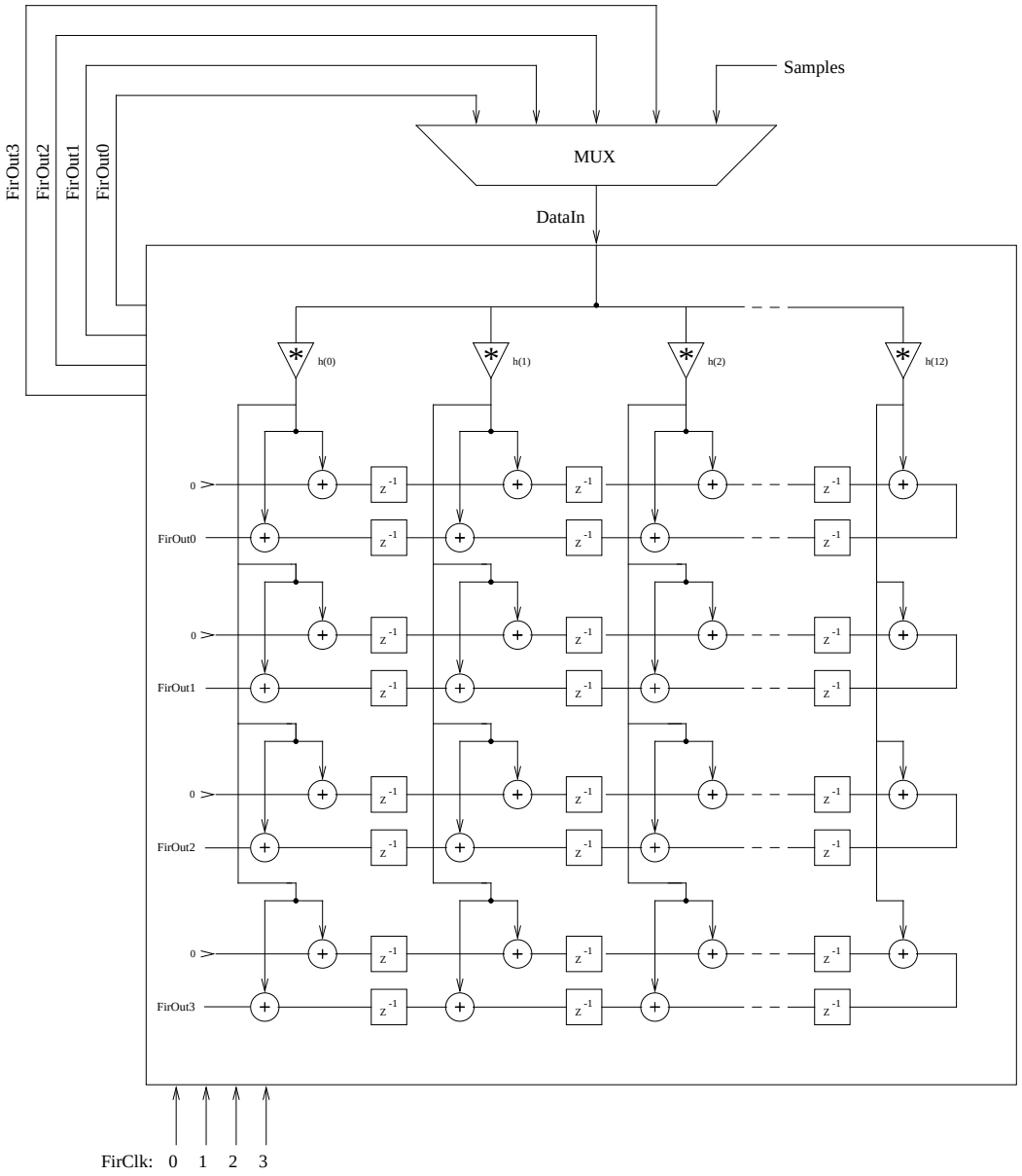

Abbildung 5.3: Struktureller Aufbau eines Multiratenfilterblocks

Die beiden im ASIC für den Real und Imaginäranteil realisierten Multiratenfilterblöcke bestehen aus jeweils:

- vier getrennten *Akkumulator-Verzögerungs* Einheiten mit *2*13-1 Filterkoeffizientenregistern h(x)* je Dezimationsfilter und

- gemeinsamen (globalen) Multiplizierern.

Die Hardwarestruktur mit den globalen Multiplizierern für jeden Teilfilter wurde aufgrund der sich ergebenden Flächeneinsparung gewählt. Eine Abschätzung ergab, dass eine Realisierung mit entsprechenden Multiplizierern für jeden Dezimationsfilter die gemachte Spezifikation im Bezug auf die benötigte Fläche bei dem zur Verfügung stehenden 0.6 µm AMS Prozeß verletzen würde. Es wären zusätzlich *3*13=39* Multiplizierer je Dezimationsfilterblock zu realisieren.

Ausgehend von der realisierten Struktur ergibt sich zwangsläufig, dass die globalen Multiplizierer in jedem Sampletakt für jeweils eine der selektierten Akkumulator - Verzögerungseinheiten arbeiten müssen.
Faßt man die Multiplizierer und einen Akkumulator-Verzögerungsblock zusammen, so erhält man die in Kapitel 3 vorgestellte SymFir II Filterstruktur. Wie in Abschnitt 3.4 schon erwähnt, sollte die SymFir II Direktstruktur aufgrund der hohen Geschwindigkeit und des kleinen Platzbedarfes, durch die Ausnutzung der Symmetrieeigenschaften der Filterkoeffizienten, eingesetzt werden. Die realisierten Filterstrukturen sind für eine symmetrische Impulsantwort und eine ungerade Anzahl von Filterkoeffizienten ausgelegt.

Es ist nun angebracht, genauer auf die Arbeitsabläufe des Multiratenfilterblocks einzugehen:

- Die einzelnen Register der Dezimatonsfilterstufen und die Steuerung des Dateneingangsmultiplexers (DataMUXin) werden durch den implementierten *Steuerautomaten* (Steuer-FSM) mit vier *FirClkin 0-3* Datenübernahmeleitungen und 3 Selektionsleitungen des Eingangsmultiplexers (DataMUX) gesteuert.

- Die FirClk Leitungen sind dabei direkt mit den Verzögerungsregistern (z^{-1}) der jeweiligen Teilfilterstruktur verbunden.

- Durch den Steuerautomaten wird bestimmt, zu welcher Zeit welche Eingangsdaten über den Dateneingangsmultiplexer den globalen Multiplizierern übergeben und welcher Dezimationsteilfilter zu welcher Zeit rechnen soll. Der Steuerautomat generiert entsprechende Steuersignale für die Register eines jeweiligen Dezimationsfilters.

- Der Ausgang einer Filterstufe kann neue Sampledaten oder schon verarbeitete Daten als *Rückkoppelwert* über den Dateneingangsmultiplexer gleich der nächsten Filterstufe übergeben.

- Die Eingangswerte werden mit den Filterkoeffizienten multipliziert und können dann von einer anderen Teilfilterstruktur übernommen werden. Der Eingangsmultiplexer erlaubt die Kaskadierung der einzelnen Dezimationsstufen.

- Gleichzeitig besteht die Option, über einen Ausgangsmultiplexer (OutMUX) selektiv zu bestimmten Zeiten schon verarbeitete Daten auszugeben.

Wie schon angedeutet ist es durch die realisierte Hardwarestruktur möglich, in jedem Takt einen anderen Filter rechnen zu lassen und damit in jedem Takt einen anderen Frequenzbereich (Kanal) zu bearbeiten. Der Steuerautomat läßt zu diesem Zweck in jedem Sampletakt eine Umkonfigurierung zu. Es läßt sich jeder Dezimationsfaktor für jede Dezimationsstufe getrennt einstellen.

Zudem besteht die Möglichkeit, die beiden realisierten Multiratenfilterblöcke getrennt für eine analytische Verarbeitung zu konfigurieren, oder sie für eine reelle Signalverarbeitung kaskadiert arbeiten zu lassen. Zu diesem Zweck wurden zusätzlich zwischen den Eingängen der Multiratenfilterkaskaden Multiplexer für den Datenfluß der jeweiligen Filterausgänge (FirOut) implementiert. Aus Gründen der Übersichtlichkeit fehlen diese in Abbildung 5.3. Bei der Kaskadierung der Multiratenfilterblöcke steigt die Filterkoeffizientenanzahl von effektiv *2*13-1=25* auf *2*25-1=49*.

5.2 Realisierung der komplexwertigen Hardwarestruktur

Im folgenden Abschnitt soll die *komplexwertige Hardwarestruktur* vorgestellt werden, wie sie im ASIC im Zusammenhang mit der analytischen Verarbeitung von Signalen realisiert wurde. Es wird davon ausgegangen, dass die Filterkoeffizienten der Filterstrukturen im Real- und Imaginärmultiratenfilterblock identisch sind. Entsprechende Voruntersuchungen mit Matlab ergaben, dass eine entsprechende Architektur für den zu entwickelnden ASIC und seine Anwendungsgebiete geeignet erscheint. Eine andere Struktur mit getrennten Filterkoeffizienten für Real und Imaginärfilterblöcke findet sich in [KRO97].
Wie schon angedeutet müssen für die analytische Verarbeitung von Signalen zwei Multiratenfilterblöcke parallel realisiert werden. Die beiden Blöcke verarbeiten dabei getrennt die Real- und Imaginäranteile eines reellen Eingangssignals.
Für die weitere Hardwarerealisierung des ASICs bedeutet dieses die Notwendigkeit der parallelen Hardwarerealisierung der *komplexen Multiplikation* [BAR91] [BRO81].

Im folgenden soll zuerst die komplexe Multiplikation mathematisch dargestellt werden. Im weiteren Verlauf dieses Abschnitts soll daraus eine Hardwarestruktur abgeleitet werden. Der Realanteil sei mit *u* und der Imaginäranteil mit *v* gekennzeichnet.
Es seien zur Veranschaulichung zwei Punkte *a* und *b* in der komplexen Ebene (z-Ebene) [HES93] [KRO97] gegeben:

$$\begin{aligned} a &= u + jv \qquad a,b \in C \\ b &= x + jy \qquad u,v \in R \end{aligned} \tag{5.1}$$

Die komplexe Multiplikation dieser Punkte ergibt dann:

$$a\,b \;=\; (u + jv)\,(x + jy) \;=\; Z \;=\; (ux - vy) + j(vx + uy) \tag{5.2}$$

wobei *Z* das Ergebnis der komplexen Multiplikation in der *z*-Ebene darstellt.

Voraussetzung für eine analytische Verarbeitung von Signalen ist die Wandlung des reellen Eingangssignals mit Hilfe eines Hilbertfilters oder Quadraturmischers in Real und Imaginär Anteile.
Aus Gründen einer einfacheren Implementation und einiger weiterer Vorteile im Bezug auf die zu realisierende Hardwarestruktur, wurde die Quadraturmischerstruktur aus Abschnitt 2.12 implementiert. Als unmittelbarer Vorteil erweist sich hier zusätzlich die Möglichkeit, das Informationsbandpaßsignal auf der Frequenzachse beliebig zu verschieben. Es ist somit nicht nur möglich, ein Nutzsignal in eine niederfrequentere, sondern auch zurück in eine höherfrequente Lage zu versetzen.
Abbildung 5.4 zeigt die implementierte Hardwarestruktur nach Gleichung 5.2 zur Berechnung der komplexen Multiplikation auf.

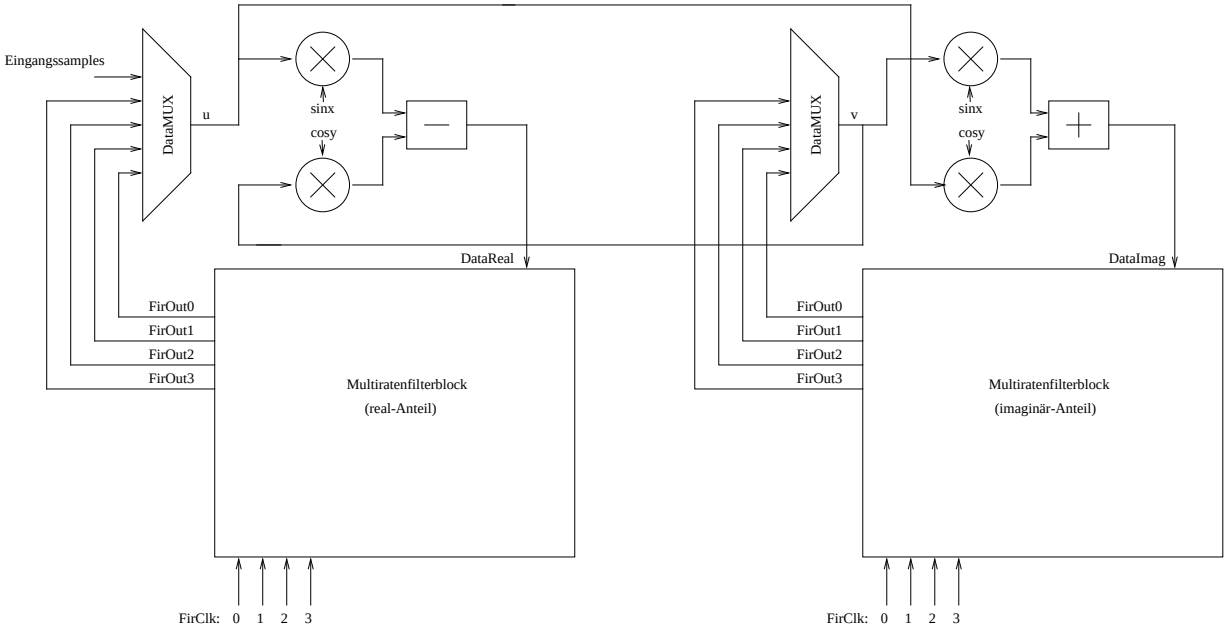

Abbildung 5.4: Hardwarestruktur zur analytischen Verarbeitung von Signalen mit Hilfe von Quadraturmischern

Die komplexwertige Hardwarestruktur läßt sind grob in folgende Teilkomponenten aufteilen:

- Doppelt ausgelegte Multiratenfilterblöcke mit den dazugehörigen Eingangsmultiplexern (DataMUX), zur getrennten Verarbeitung von Real- und Imaginärsignalen,

- je zwei Multiplizierstrukturen für einen Multiratenfilterblock zur Multiplikation von Eingangsdaten mit dem Sinus- bzw. Cosinussignal des DDS Generators (Quadraturmischung),

- einen Addierer und Subtrachierer.

Aufgrund der analytischen Verarbeitung der Signale wurde die Quadraturmischstruktur gleich mit in die Hardwarestruktur zur Berechnung der komplexen Multiplikation implementiert. Zum besseren Überblick im Bezug auf Gleichung 5.2 sind in der Abbildung 5.4 zusätzlich die verwendeten Symbole für Realteil *u* und den Imaginärteil *v* eingezeichnet.
Die Verarbeitung des Real- und Imaginärsignalanteils wird mit Hilfe der Komplexmultiplikation durchgeführt. Im Gegensatz zu einer reellen Signalverarbeitung, bedeutet dies für eine möglichst parallele Realisierung der komplexen Multiplikation, die Notwendigkeit der Implementation zweier Signalpfade für den Real- und Imaginäranteil.
Zusammenfassend kann gesagt werden, dass eine parallele Realisierung einer analytischen Verarbeitung von Signalen im Gegensatz zu einer reellen Signalverarbeitung eine doppelte Auslegung der Hardwarekomponenten benötigt. Für eine parallele Realisierung ist daher mit einer größeren Fläche zu rechnen.

5.3 Zusammenwirken der Multiratenfilterkaskade und der komplexwertigen Hardwarestruktur

Im nachfolgenden sollen die im Zuge dieser Arbeit entwickelten Teilkomponenten in einen Kontext zueinander gebracht werden. Es sollen dabei *drei* mögliche Arbeitsweisen (Betriebsmodi) des ASICs besprochen werden. Die Steuerung einzelner Betriebsmodi geschieht über den Steuerautomaten (SteuerFSM). Der Steuerautomat soll in den weiteren Abschnitten einfachheithalber als FSM bezeichnet werden.
Es sollen folgende Betriebsmodi unterschieden und näher besprochen werden:

- Wandlung eines reellen in ein analytisches Signal durch Quadraturabwärtsmischung,
- komplexe Verarbeitung der Sampledaten und
- Verschiebung eines Bandpaßsignals in eine höherfrequente Lage.

1) *Wandlung eines reellen in ein analytisches Signal durch die Quadraturabwärtsmischung:*

Da ein A/D Wandler (Analog/Digital-Wandler) nur ein reelles digitales Signal zur Verfügung stellt, der ASIC allerdings überwiegend für eine analytische Verarbeitung von Signalen ausgelegt ist, muß vor der eigentlichen Verarbeitung eine Wandlung in ein äquivalentes komplexes Signal durchgeführt werden.
Mit der im vorherigen Abschnitt realisierten Hardwarestruktur kann diese Wandlung vor der eigentlichen Verarbeitung des Eingangssignals durch die Multiratenfilter durchgeführt werden. Da die komplexe Multiplikation mit jeweils zwei Multiplizierstrukturen für den Real- und Imaginäranteil in Hardware realisiert werden muß, darf für eine Wandlung in ein komplexes Signal nur einer der Multiplizierer im Real- und Imaginärblock aktiv sein. Aus diesem Grund ist es dem Steuerautomaten über bestimmte Steuerleitungen möglich, die Sinus/Cosinus Signalausgänge getrennt auf einen Signalausgabewert von *null* zu setzen. Durch diese Maßnahme gehen die nicht benötigten Terme der in Hardware realisierten komplexen Multiplikation nicht in die Berechnung mit ein, d.h. die entsprechenden Terme der Gleichung 5.2 fallen weg. Durch diese Möglichkeit kann eine durch die FSM gesteuerte und konfigurierte Quadraturabwärtsmischstruktur realisiert werden.
Die Multiplikation mit dem DDS Sinus bzw. Cosinus Signal übernimmt dann jeweils nur ein Multiplizierer des jeweiligen Real- und Imaginärblock.
Das Wegfiltern des durch die Mischung entstandenen höherfrequenten Anteils kann in der ersten Dezimationsfilterstufe des Multiratenfilterblocks durchgeführt werden, entsprechend der Quadraturabwärtsmischstruktur in Abschnitt 2.12. Die FSM kann zu diesem Zweck explizit konfiguriert werden.

2) *Komplexe Verarbeitung der Sampledaten:*

Nachdem das reelle Eingangssignal in ein analytisches Signal gewandelt wurde, muß für eine weitere komplexe Signalverarbeitung Gleichung 5.2 in Hardware ausgeführt werden. Zu diesem Zweck werden über die FSM die DDS Sinus/Cosinus Signalausgänge auf einen festen Wert von *eins* gesetzt. Es wird eine Multiplikation mit der Konstante eins durchgeführt. Durch diese Maßnahme wird erreicht, dass die Multiplizierstrukturen die Eingangsdaten, die zur Berechnung der komplexen Multiplikation benötigt werden, nicht beeinflußt.

Nach der arithmetischen Verarbeitung, gelangen die Daten getrennt zu weiteren Verarbeitung, in die jeweiligen Multiratenfilterblöcke.

3) *Verschiebung eines Bandpaßsignals in eine höherfrequente Lage:*

Mit der realisierten Quadraturmischern ist es zusätzlich möglich, das in der Regel in einer niedrigen Frequenzlage verarbeitete Signal, wieder in eine höherfrequente Frequenzlage zu verschieben.
Zu diesem Zweck werden mit Hilfe der vier Multiplizierer die Eingangssignale mit einer vorgegebenen Frequenz des DDS Sinus/Cosinus Generators gemischt.
Nach der Quadraturmischung ist es möglich, mit Hilfe des Ausgangsmultiplexers (FirOut) das auf der Frequenzachse verschobene Signal getrennt als Real- bzw. Imaginärsignal auszugeben. Einer weiteren aufgesetzten Einheit, die sich außerhalb der ASICs befinden kann, stehen diese Signale getrennt zur weiteren Verarbeitung zur Verfügung. Es sei zu bemerken, dass bei der Frequenzverschiebung komplexer Signale keine Spiegelfrequenzen entstehen.

5.3.1 Beispiel einer Ablaufsteuerung mit Hilfe der FSM

Im folgenden soll ein Beispiel die Steuerung der komplexen Hardwarestruktur durch die FSM verdeutlichen. Ausgegangen werden soll von einer realisierten FSM mit 16 Zuständen. Dementsprechend stehen 16 Zeitscheiben zur Verfügung, in denen Teilverarbeitungsaufgaben gerechnet werden können. Daraus resultiert, dass maximal 16 Frequenzbänder oder Frequenzkanäle bearbeitet werden können.
Der Einsatz der FSM in Verbindung mit der realisierten Hardwarearchitektur führt zu folgenden Korrespondenzen:

- in jedem Takt kann ein anderer Filter gerechnet werden und
- in jedem Takt können alle möglichen Parameter im ASIC geändert werden.

Diese Sachverhalte sollen an einem Beispiel näher verdeutlicht werden:
Dabei soll die Ablaufsteuerung für eine Multiratenfilterkaskade mit den vier Dezimationsfiltern betrachtet werden. Es soll eine Dezimationsfolge von 2=>2=>2=>2 und einer anschließenden Ausgabe des Samples durch den Ausgangsmultiplexer (OutMux) durchgeführt werden. Der erste Dezimationsfilter verarbeitet dabei nur jeden 2. Samplewert, von dieser Ausgabe wird wiederum nur jeder 2. Samplewert weiterverarbeitet u.s.w. , bis die Dezimationsfolge abgearbeitet wurde. Abbildung 5.5 soll diesen Sachverhalt für fortlaufende Zeitscheiben und die vier realisierten Dezimationsfilter (Dezfilter 1-4) graphisch darstellen.

Zeitscheibe:	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
	|	|	|	|	|	|	|	|	|	|	|	|	|	|	|	|	|	|	|
Dezfilter1	*		*		*		*		*		*		*		*		*		*
Dezfilter2		*				*				*				*				*	
Dezfilter3				*								*							
Dezfilter4								*											

Abbildung 5.5: Beispieltaktschema einer fortlaufenden Dezimation von 2=>2=>2=>2

Ein Stern (*) in den jeweiligen Zeitscheiben bedeutet, dass zu diesem Zeitpunkt der entsprechende Dezimationsfilter gerechnet wird. Es kann eine Gleichung zur Bestimmung der resultierenden Samplerate f_d unter der Annahme, dass der Dezimationsfaktor in jeder Dezimationsstufe *s* gleich ist, angegeben werden:

$$f_d = \frac{f_a}{d^s} \tag{5.3}$$

f_a ist dabei die Abtastfrequenz, *d* die Höhe der Dezimation und *s* die Anzahl der realisierten Dezimationsstufen.
Ein Beispiel soll dieses verdeutlichen:
Wird ein Eingangssignal mit f_a = 10 Mhz abgetastet, so erhält man für die vier realisierten Dezimationsfilter nach dem obigen Taktschema eine Resamplingrate von $f_d = 10Mhz/2^4 = 625$ Khz. Der frei zur Verfügung stehende 16. Zeitscheibe ist frei für einen weiteren Dezimationsfilter oder für die Ausgabe der Sampledaten über den Ausgabemultiplexer (OutMUX).
Der dazugehörende Mikrocode für die FSM zur Realisierung des obigen Ablaufschemas ist in Tabelle 5.1 zusammengefaßt. Jeder Taktzustand ist hier mit den Parameterübergaben seiner entsprechenden Teilkomponente aufgezeigt. Die zuvor benötigte Wandlung des reellen in ein analytisches Signal soll hier nicht weiter betrachtet werden.
Die einzelnen binären Steuerworte sind dem Abschnitt 5.4 oder der Zusammenfassung im Anhang zu entnehmen.

Zustand (Takt):	DataMUXin:	FilCLK:	OutMUXout:	DDS Modus:	Phasenschritt:
1	000	0001	**	01	f = **
2	001	0010	**	01	f = **
3	000	0001	**	01	f = **
4	010	0100	**	01	f = **
5	000	0001	**	01	f = **
6	001	0010	**	01	f = **
7	000	0001	**	01	f = **
8	011	1000	**	01	f = **
9	000	0001	**	01	f = **
10	001	0010	**	01	f = **
11	000	0001	**	01	f = **
12	010	0100	**	01	f = **
13	000	0001	**	01	f = **
14	001	0010	**	01	f = **
15	000	0001	**	01	f = **
16	011	1000	11	01	f = **

Tabelle 5.1: Mikrocodebeispiel für eine Dezimation von 2=>2=>2=>2=>out mit der FSM

Die mit ** gekennzeichneten Parameterworte sind für die jeweilige Verarbeitung nicht relevant.
Im folgenden soll die maximal mögliche Dezimation für die realisierte FSM mit 16 Zuständen errechnet werden:
Sie beträgt bei vier realisierten Dezimationsfiltern genau 16^4=65536. Es ist somit möglich, eine *Sampleratenreduktion* (downsampling) von maximal 65536 durchzuführen, d.h. nach jedem 65536‘ten Sampletakt wird ein neuer Samplewert ausgegeben.

Aufgesetzte Einheiten können bei dieser Einstellung des Dezimationsfaktors 65536 mal so langsam arbeiten, was für die meisten Anwendungen ausreichend ist.

5.4 Registerbelegung und Programmierung

Die Konfigurierung und Steuerung der einzelnen Komponenten des ASICs über die FSM geschieht über bestimmte Konfigurations- und Steuerleitungen. Damit ist es einem Benutzer möglich, durch eine entsprechende Konfigurierung der Steuerleitungen und Register, den Arbeitsablauf über die FSM im ASIC zu bestimmen.
Die FSM ist nur in der Lage bestimmte Steuerleitungen zu beschreiben.
Ein Auslesen von Registerinhalten wurde nicht vorgesehen. Dieses ist auch nicht notwendig, da das ASIC nur nach einem vom Benutzer streng vorgegebenen Ablauf arbeiten soll.

Es soll näher auf Steuerleitungen und Registerbelegungen eingegangen werden:
Der Betriebsmodus des ASICs kann für eine reelle oder analytische Verarbeitung von Signalen, über ein nach außen gelegtes Pin „/reell-analyt“ durch einen Hostrechner oder eine andere Einheit eingestellt werden. Diese Konfigurationsleitung ist die einzige, die als Pin nach außen geführt wurde. Alle anderen Steuerleitungen sind nur intern, direkt von der FSM ansprechbar.
Tabelle 5.2 faßt die Funktion dieses nach außen gelegten Steuerpins zusammen.

Pinbezeichung	Bedeutung	Funktionsbeschreibung
/reell-analyt	Reelle/analytische Signalverarbeitung	0 – relle Verarbeitung der Sampledaten 1 – analytische Verarbeitung der Sampledaten

Tabelle 5.2: Beschreibung des Betriebsmodipins des ASICs

Das „ / “ Symbol in der Tabelle kennzeichnet die low Aktivität des Pins, zur Selektion einer reellen Signalverarbeitung.
Es soll im folgenden auf die Steuerung über die FSM näher eingegangen werden.

Die von der FSM direkt ansprechbaren Steuerleitungen der Einzelkomponenten sind:

- der Dateneingangsmultiplexer (DataMUX),
- der Datenausgangsmultiplexer (OutMUX),
- die Taktleitungen der einzelnen Dezimationsfilter (FirCLK),
- die Phasenschrittwerte der 16 Phasenakkumulatoren des DDS Sinus/Cosinus Generators (PS) und
- die Sinus/Cosinus enable Leitungen (DDS Modus).

Es soll nun nacheinander die genaue Belegung der einzelnen Steuerleitungen betrachtet werden. Dabei sollen tabellarisch in obiger Reihenfolge die möglichen Steuerworte aufgelistet werden.

Die Steuerung des Dateneingangsmultiplexers (DataMUX) geschieht über ein drei Bit Steuerwort und hat folgende Bedeutung:

Fktnummer	Bezeichnung DataMUX	Funktionsbeschreibung
0	000	Übernahme neuer Sampeldaten
1	001	Übernahme der Daten des 1. Dezimationsfilters (FirOut0)
2	010	Übernahme der Daten des 2. Dezimationsfilters (FirOut1)
3	011	Übernahme der Daten des 3. Dezimationsfilters (FirOut2)
4	100	Übernahme der Daten des 4. Dezimationsfilters (FirOut3)
5	101	--- keine Funktion
6	110	--- keine Funktion
7	111	--- keine Funktion

Tabelle 5.3: Belegung der Steuerbits zur Dateneingangsselektion

Mit Hilfe des ersten Funktionswortes können neue Sampledaten des A/D Wandlers in die komplexe Hardwarestruktur übernommen werden. Die übrigen Funktionsworte selektieren den Pfad der rückgekoppelten Daten. Die letzten drei Funktionsworte (--- nicht benutzt) haben keine Bedeutung. Die Zuweisung auf die Steuerworte durch die FSM hat gleiche Auswirkungen auf den Real- und den Imaginäreingangsmultiplexer.

Eine mögliche Ausgabe von Sampledaten während der Bearbeitung von Teilaufgaben im ASIC oder nach einer abgeschlossenen Verarbeitung ist durch den Ausgangsmultiplexer (OutMUX) in jedem Sampletakt möglich. Äquivalent zu den Dateneingangsmultiplexern werden auch hier mit nur einem Steuerwort die Real- und Imaginärausgangsmultiplexer gleichzeitig angesprochen. Die zwei Bit Steuerleitungen dieses Ausgabemultiplexers haben dabei folgende Bedeutung:

Fktnummer	Bezeichnung OutMUX	Funktionsbeschreibung
0	00	Dezimationsfilter 1 Ausgabe
1	01	Dezimationsfilter 2 Ausgabe
2	10	Dezimationsfilter 3 Ausgabe
3	11	Dezimationsfilter 4 Ausgabe

Tabelle 5.4: Belegung der Steuerbits zur Datenausgangsselektion

Die Taktung der einzelnen Dezimationsfilter in den Multiratenfilterkaskaden geschieht über vier *FirClk* Leitungen. Jedes Bit der FirClk Signalleitungen ist mit dem dazugehörigen Register z^{-1} einer entsprechenden Dezimationsfilterstufe verbunden. Ein gesetztes Bit bedeutet, dass die Register dieser Stufe die Ausgangsdaten des globalen Multiplizierers übernehmen. Ein gelöschtes Bit auf diesen Steuerleitungen spricht die Register und damit die jeweilige Stufe nicht an. Man kann auch sagen, dass bei einem gesetzten Steuerbit der jeweilige Filter arbeitet. Der Übersichtlichkeit halber ist das Steuerwort 4 Bit groß, obwohl 2 Bit ausreichen würden.

Fktnummer	Bezeichnung FirClk	Funktionsbeschreibung
0	0001	Dezimationsfilter 1 aktiv
1	0010	Dezimationsfilter 2 aktiv
2	0100	Dezimationsfilter 3 aktiv
3	1000	Dezimationsfilter 4 aktiv

Tabelle 5.5: Steuerbits zur Selektion der Dezimationsfilterstrukturen

Zur Frequenzgenerierung wurde der in Kapitel 3 vorgestellte DDS Sinus/Cosinus Generator auf 16 parallele Phasenakkumulatoren erweitert. Im Prinzip wurde eine Anpassung des DDS Generators an die FSM mit 16 Zuständen durchgeführt.
Aufgrund der möglichen Konfiguration über die FSM, könnte jedem Zustand ein Phasenakkumulator zugeordnet werden. Es können so 16 Frequenzbereiche bearbeitet werden. Man kann sagen, dass durch diese Maßnahme maximal 16 unabhängige Kanäle realisiert werden können. Um einen Frequenzbereich zu wechseln und damit einen anderen Kanal bearbeiten zu können, muß das Phasenschrittregister über die FSM mit einem neuen 32 Bit Wert konfiguriert werden. Die Hardwarestruktur ist dabei so konzipiert, dass in jedem Sampletakt ein anderer Phasenakku mit dem gerade gültigen Phasenschrittwert arbeiten kann, d.h. in jedem Takt ein anderer Kanal bearbeitbar ist.
Der neue Phasenschrittwert beeinflußt gleichermaßen die Sinus und Cosinus DDS Frequenzgenerierung. Durch die FSM kann die Frequenz in jedem Sampletakt geändert werden.
Da nur ein Phasenschrittregister für beide Frequenzgeneratoren existiert, ist es in einem Zeitintervall nicht möglich, unterschiedliche Sinus- bzw. Cosinus Frequenzen zu generieren.

Bitposition	Bezeichnung	Funktionsbeschreibung
0...31	Phasenschrittregister (PS)	Phasenschrittregister zur Frequenzgenerierung (32 Bit)

Tabelle 5.6: Phasenschrittregister zur Frequenzgenerierung

Zur Realisierung der drei möglichen Betriebsmodi des ASICs ist es wichtig, getrennt die Sinus und Cosinus DDS Generatoren zu kontrollieren. Zu diesem Zweck können über zwei Modisteuerleitungen, die Frequenzgeneratoren für einen der drei Betriebsmodi des ASICs getrennt eingestellt werden. Tabelle 5.7 faßt die dafür vorgesehenen Steuerworte zusammen. Das eingetragene *PS* (Phasen Schritt) in der Tabelle bedeutet, dass ein Eingangssignal mit einem Sinus bzw. Cosinus-Signal einer bestimmten Frequenz multipliziert wird.

Betriebsmodus	Bezeichnung DDSModus	Funktionsbeschreibung
reell-analyt Wandlung	00	Sinx(real) = 0 ; Cosy(real) = PS Sinx(imag) = PS ; Cosy(imag) = 0
Analytische Verarbeitung	01	Sinx(real) = 1 ; Cosy(real) = 1 Sinx(imag) = 1 ; Cosy(imag) = 1
Frequenz-Verschiebung	10	Sinx(real) = PS ; Cosy(real) = PS Sinx(imag) = PS ; Cosy(imag) = PS
---	11	--- keine Funktion

Tabelle 5.7: Steuerworte der drei Betriebsmodi

Mit den in diesem Abschnitt eingeführten Steuerworten ist es möglich, die einzelnen Komponenten des ASICs durch die FSM beliebig zu konfigurieren und damit zu steuern. Die FSM generiert dabei die *Timing Signale* für die entsprechenden Teileinheiten. In der derzeitigen Version des ASICs müssen die Steuerworte direkt in den VHDL-Code der FSM eingetragen werden. Der ASIC ist damit für eine bestimmte Aufgabe konfiguriert. Die FSM Konfiguration des Beispiels in Tabelle 5.1 ist dem Anhang „Quellcodes“ zu entnehmen. In einer erweiterten Version des ASICs soll eine automatische Konfiguration über eine entsprechende Schnittstelle auch während des Betriebes durch eine aufgesetzte Einheit (Hosteinheit) möglich sein.

5.5 Timingkonzept des ASICs

Nach der genaueren Untersuchung einzelner Funktionseinheiten des ASICs und einiger Aspekte der *Timings* der Multiratenfilterkaskade mit Hilfe der FSM in den vorherigen Kapiteln, ist es notwendig, die Timingabläufe aller Komponenten zu betrachten.
Im ASIC können grundlegend *zwei* Timingsysteme unterschieden werden:

- das Timing aller taktsensitiven Einheiten, die der Samplerate entsprechen und

- die konfigurierbare Timinggenerierung über die FSM für alle anderen Teileinheiten.

Zum ersten Timing:
Wie schon angedeutet, entspricht die Taktrate des ASICs der maximal möglichen Samplerate. Der DDS Generator und der endliche Steuerautomat erhalten als einzige Komponenten den vollen Sampletakt. Alle anderen Signale sind von ihm abgeleitet.

Zur zweiten Timinggenerierung über die FSM:
Die FSM generiert entsprechende Takt- und Steuersignale für die Multiratenfilterkaskade und die dazugehörigen Multiplexer.
Diese richten sich nach der gerade aktuellen Konfiguration der FSM. Die FSM agiert dabei als globales Schaltwerk zur Steuerung aller internen Abläufe, die nicht dem Sampletakt unterliegen. Alle durch die FSM generierten Steuersignale sind mit der Sampletaktrate *synchronisiert.*

Auf *Timingdiagramme*, d.h. der graphischen Darstellung der Logikpegel von Adressen, Daten und Steuerleitungen zu bestimmten Zeitpunkten, wurde aus Übersichtlichkeitsgründen verzichtet.

5.5.1 Taktung und Ablaufsteuerung

Bei der Entwicklung des ASICs wurde ein vollständig synchron getaktetes Design zur jeweiligen Samplerate realisiert. Die Samplerate entspricht der Taktrate des ASICs. Der Einsatz einer bedingten Taktung würde besondere Ansprüche an das Entwurfssystem und den Entwickler stellen, was die Realisierung nur verkomplizieren würde. Ein synchroner Entwurf war möglich, da die Leistungsaufnahme in der gemachten Spezifikation als unkritisch zu betrachten ist. Die gesamten im ASIC befindlichen Register bestehen aus vorderflankengesteuerten D-Flip Flops. Mit jeder positiven Flanke eines Steuersignals werden anliegende Daten in die Register übernommen.
Wie schon in dem vorhergehenden Kapitel deutlich wurde, gibt es innerhalb der ASICs zwei voneinander unabhängige Taktsysteme.
Die FSM generiert zur Samplefrequenz taktsynchron die abgeleiteten Takte der vier realisierten Dezimationsfilter.
Bei einer positiven Taktflanke werden die anliegenden Daten in die jeweiligen Pipelineregister der realisierten Filter übernommen und bei jedem fortlaufenden Takt an weitere Einheiten weitergegeben.
Wie weiter angedeutet, kann durch die FSM in jedem Takt ein anderer Filter gerechnet werden, indem die FSM entsprechende Taktsignale für den jeweiligen Dezimationsfilter generiert.
Es ist festzustellen, dass der Datendurchsatz der Multiplizierer der Multiratenfilterkaskade und der Arithmetik zur Realisierung der komplexen Multiplikation in der Regel höher ist, als die der Arithmetik der Dezimationsfilter. Aus diesem Grund wird die Höhe der Verarbeitungsleistung (Durchsatz) pro Zeiteinheit maßgeblich durch die Geschwindigkeit dieser Arithmetik bestimmt.

5.6 Überleitung

Nach einer kurzen Einleitung, Stellung und den Aufgabenbereich des ASICs in einem nachrichtentechnischen Systems in Kapitel 1, wurden in Kapitel 2 spezielle signaltheoretische Grundlagen besprochen. Kapitel 3 gibt einen Überblick über geeignete und weiter verwendete Hardwarefilter- und Funktionsgeneratorstrukturen und deren Realisierung in der Hardwarebeschreibungssprache VHDL.
Kapitel 4 gibt schließlich einen groben Überblick über realisierte Einheiten und deren grobes Zusammenwirken. In Kapitel 5 wurde dann auf die Einbindung der realisierten Teilkomponenten eingegangen. Zusätzlich wurde die Programmierung und das Zusammenspiel der realisierten Teilkomponenten genauer erläutert.
Es folgt das Vorgehen bei der Realisierung des ASICs in VHDL. Dazu wird zunächst entsprechend der zu verwirklichenden Hardwarearchitektur ein Verhaltensmodell in der Hardwarebeschreibungssprache VHDL erstellt. In diesem Zusammenhang wird anfangs auf ein grundsätzliches Designvorgehen in VHDL und später auf allgemeine Vorabschätzungen und Simulationen der Verhaltensmodelle der verwendeten Teilrealisierungen eingegangen.

Kapitel 6

Einsatz von VHDL zur Entwicklung des ASICs

In diesem Kapitel werden zuerst allgemeine Grundlagen zur Erstellung eines ASICs in der Hardwarebeschreibungssprache VHDL behandelt. Nachfolgend soll das Vorgehen bei der Entwicklung des ASICs vorgestellt werden. Einige vorab gemachte Abschätzungen und die Vorgehensweise, die bei der Simulation der Verhaltensmodelle der realisierten Teilkomponenten angewendet wurde, schließen diese Kapitel ab.

6.1 Allgemeines Entwurfsvorgehen mit VHDL

Zur Beschreibung von Hardwarestrukturen werden speziell hierfür entwickelte Hardwarebeschreibungssprachen verwendet.
Als Beispiel soll die Hardwarebeschreibungssprache VHDL angeführt werden, die 1983 vom amerikanischen „Department of Defence“ initiiert wurde. Sie ist seit Ende 1987 als IEEE Standart genormt [VHD94]. Inzwischen ist VHDL insbesondere in Europa, quasi der Standard der Hardwarebeschreibungssprachen. Sie wird für die Beschreibung und Simulation digitaler Systeme und deren Umgebung benutzt.
VHDL nutzt eine Syntax, die mit einer prozedualen Programmiersprache vergleichbar ist, jedoch hochparallele Ansätze aufweist. VHDL unterstützende Simulations- und Synthesewerkzeuge ermöglichen es, einen Großteil der Schritte von der Spezifikation bis zum Layout eines Chips nahezu nahtlos aneinanderzufügen. Es sind keine Konvertierungen zwischen einzelnen Designstufen nötig. Da VHDL eine Programmiersprache ist, kann eine Hardwarestruktur detailliert oder aber völlig frei von konkreten Strukturbeschreibungen beschrieben werden. Ein Entwurf kann in mehrere Schritte aufgeteilt werden, die systematisch von einer sehr abstrakten Beschreibung bis zur einer konkreten Struktur in Hardware führen [BLE96] [LEH94].

6.1.1 Top-Down Entwurf

Es ist nicht möglich, eine integrierte Schaltung ausgehend von einer spezifizierten Problemstellung direkt in einem Schritt zu entwickeln. Der Entwurf geschieht vielmehr in mehreren Stufen mit Hilfe eines *Top-Down* Verfahrens. Diese führt systematisch von einer abstrakten Beschreibung zu einer konkreten Struktur. Da die Realisierung allerdings auf der von einen Halbleiterhersteller zur Verfügung gestellten Technologie basiert, spielen bei der Entwicklung einer integrierten Schaltung auch sogenannte *Bottom-Up* Einflüsse eine gravierende Rolle.
Der zu entwickelnde ASIC soll mit Hilfe eines *Standardzellentwurfssystems* als ASIC entwickelt werden. Die Anteile der Bottom-Up Entwicklung sind bei einem Standardzellentwurf schon von dem Halbleiterhersteller erbracht. Die Standardzellen bestehen aus einfacheren Grundkomponenten, wie einfacher Gatter einer tieferen Realisierungsebene

[LEH94]. Der Entwickler hat die Aufgabe, mit Hilfe geeigneter Werkzeugen seine Schaltung hierarchisch von einer hohen Abstraktionsebene her nach einer Top-Down Strategie zu entwickeln. Er benötigt für den Entwurf keine umfangreichen Kenntnisse der Halbleiterphysik und der Schaltungstechnik.
Allerdings ist die Beherrschung spezieller Entwurfsmethoden und Softwarewerkzeuge eines jeweiligen Hersteller eine unabdingbare Voraussetzung zur Entwicklung eines ASICs. Deren Anwendung soll auch in diesem Abschnitt im Vordergrund stehen.

Bei einem Entwurf mit Hilfe einer Hardwarebeschreibungssprache ist der Ausgangspunkt die *Verhaltensbeschreibung* der Schaltung, deren Funktionalität durch Simulation geprüft werden soll. Die Top-Down Strukturierung erhält man, indem man im weiteren Vorgehen die Schaltung in Funktionsblöcke gliedert, bis man letztlich nur noch eine Strukturbeschreibung erhält. Dieser Prozeß des algorithmischen Entwurfes von Funktionseinheiten, hierarchischer Verfeinerung und Umsetzung in Strukturbeschreibungen wird weiter rekursiv ausgeführt, bis man schließlich bei Elementen einer Zellbibliothek eines Herstellers angekommen ist. Die Schaltung kann nun praktisch realisiert werden. Dadurch dass die Schnittstelle der Schaltung und deren eigentliche Realisierung voneinander getrennt sind, können Alternativen im Entwurf unterstützt werden (exploring the design space). Durch den Einsatz von Synthesewerkzeugen kann dann die Entwurfsaufgabe auf den unteren Abstraktionsebenen zunehmend automatisiert werden.
Abbildung 6.1 illustriert graphisch den allgemeinen Ablauf zu Entwicklung einer integrierten Schaltung mit Hilfe einer Hardwarebeschreibungssprache. Eine klare Trennung zwischen Systementwurf und dem Entwurf der einzelnen Blöcke und die zeitliche Abfolge der aufgeführten Schritte der Systementwicklung werden allerdings in der Praxis aufgrund von zeitlichen Zwängen oft nicht eingehalten.
Da die Simulation und die Synthese wichtige Unterpunkte bei der Entwicklung einer integrierten Schaltung darstellen, soll auf sie in nachfolgenden Abschnitten genauer eingegangen werden.

Ein Entwurf kann grob in folgende fünf Teilschritte gegliedert werden:

- Spezifikation,
- Algorithmenentwurf,
- Architekturentwurf,
- Synthese und
- Layout.

Der erste Schritt bei der Entwicklung eines ASICs beinhaltet eine *Spezifikation* der zu entwerfenden integrierten Schaltung. Hier werden alle Informationen aufgeführt, die für die zu entwerfende Schaltung und sein Umfeld wichtig sind. Dazu zählt auch sein Ein- und Ausgabeverhalten.
Die integrierte Schaltung wird hierbei als *black box* betrachtet. Anschließend wird der Entwurf in funktionale Blöcke aufgespalten (Partitionierung). Nach der Aufspaltung wird versucht, eine Verhaltensbeschreibung der zu realisierenden Algorithmen in der Hardwarebeschreibungssprache zu erstellen. In dieser Stufe des *Algorithmenentwurfs* wird auch geprüft, ob diese Blöcke überhaupt technisch realisierbar sind und mit welchen Aufwand zu rechnen ist.

Für eine hardwarenahe Implementierung sollte der Entwurf vorausschauend in überschaubare und einzeln überprüfbare Komponenten aufgeteilt werden. Dabei ist es wichtig, sich Gedanken über nebenläufig arbeitende Prozesse zu machen und diese Teilprozesse im Kontext zum Gesamtablauf im ASIC zu sehen.
In diesem Zusammenhang werden Schnittstellen und Kommunikationsprotokolle für die einzelnen Einheiten spezifiziert. Sie sollten dabei möglichst lokal in den betroffenen Komponenten realisiert werden, ohne bei einer nachträglichen Änderung das Gesamtsystem ändern zu müssen.
Soll die Realisierung besonders zeitkritische Komponenten oder aufwändige Teilrealisierungen, wie große Speicher enthalten, so empfiehlt sich immer von vornherein eine *hardwarenahe Implementierung* der Verhaltensmodelle. In diesem Fall hängen viele Aspekte der Realisierung von Zeitverhalten oder Größe einzelner Komponenten des ASIC- Herstellers ab. Um Bottom-Up Einflüsse zu berücksichtigen und um einer Rückkehr in vorhergehende Realisierungsebenen vorzubeugen, ist eine genau Kenntnis der Unterlagen der ASIC Hersteller wichtig. Unvorhersehbare Bottom-Up Einflüsse führen oftmals zu einer ungewollten Rückkehr zu einer früheren Entwurfsphase.

In einer nächsten Stufe wird über den *Architekturentwurf* versucht, eine *synthesefähige hierarchische Struktur* auf Basis der von einem Hersteller zur Verfügung gestellten Logikzellen zu erstellen. Eine nachträgliche *Logiksynthese* ergibt dann eine *hierarchische Netzliste* des Designs auf Basis der Logikgatter des Herstellers.
Aus der hierarchischen Netzliste kann interaktiv bzw. automatisch ein *physikalisches Layout* für den Chip generiert werden. Mit Hilfe von speziellen Plazierungs- und Verdrahtungswerkzeug wird in einem weiteren Schritt das endgültige Layout generiert. Für den zu realisierenden ASIC wurde das *Placement & Routing* mit *Cadence Design Framework Tools (DFII)* durchgeführt. In dieser Phase ist es auch für ein erstes Prototyping möglich, mit Hilfe von FPGA (Field Programmable Gate Array) Backend Tools ein direktes mapping auf ein FPGA durchzuführen.
Anschließend kann das physikalische Layout zusammen mit speziell generierten *Testvektoren*, zur Verifikation der Funktionsweise des ASICs, an den Halbleiterhersteller übergeben werden. Eine *Fertigung* ergibt schließlich den ASIC.
Es soll im folgenden näher auf die Simulation in verschiedenen Designstufen eingegangen werden.

6.1.2 Simulation

Bei der Entwicklung einer integrierten Schaltung werden in der Regel schon nach der Algorithmenentwurfsstufe *Simulationsmodelle* erstellt, um die Funtionalität eines Entwurfs im Detail zu überprüfen. Zu diesem Zweck kann eine Simulation der entworfenen Hardwarestrukturen direkt vom VHDL Code aus durchgeführt werden.
Allgemein findet zur Simulation ein *Stimulus-File* in VHDL Verwendung, dass vom Entwickler explizit zu Testzwecken erstellt wurde.
Es soll in diesem Zusammenhang der Begriff der *Entity* als Schnittstelle eingeführt werden:
Die Entity definiert für eine Komponente des Entwurfs die *externe Sichtweise*. Sie stellt eine Verbindung (Port) zu anderen Komponenten her. Die Entity ist dem zu testenden Entwurf hierarchisch übergeordnet. Sie kann für mehrere Architekturen gelten, d.h. verschiedene Designarchitekturen (Realisierungen) können einer bestimmten Entity zugeordnet werden.
Die Entity beinhaltet Namen, Ein- bzw. Ausgänge und zusätzliche Deklarationen des Entwurfs.

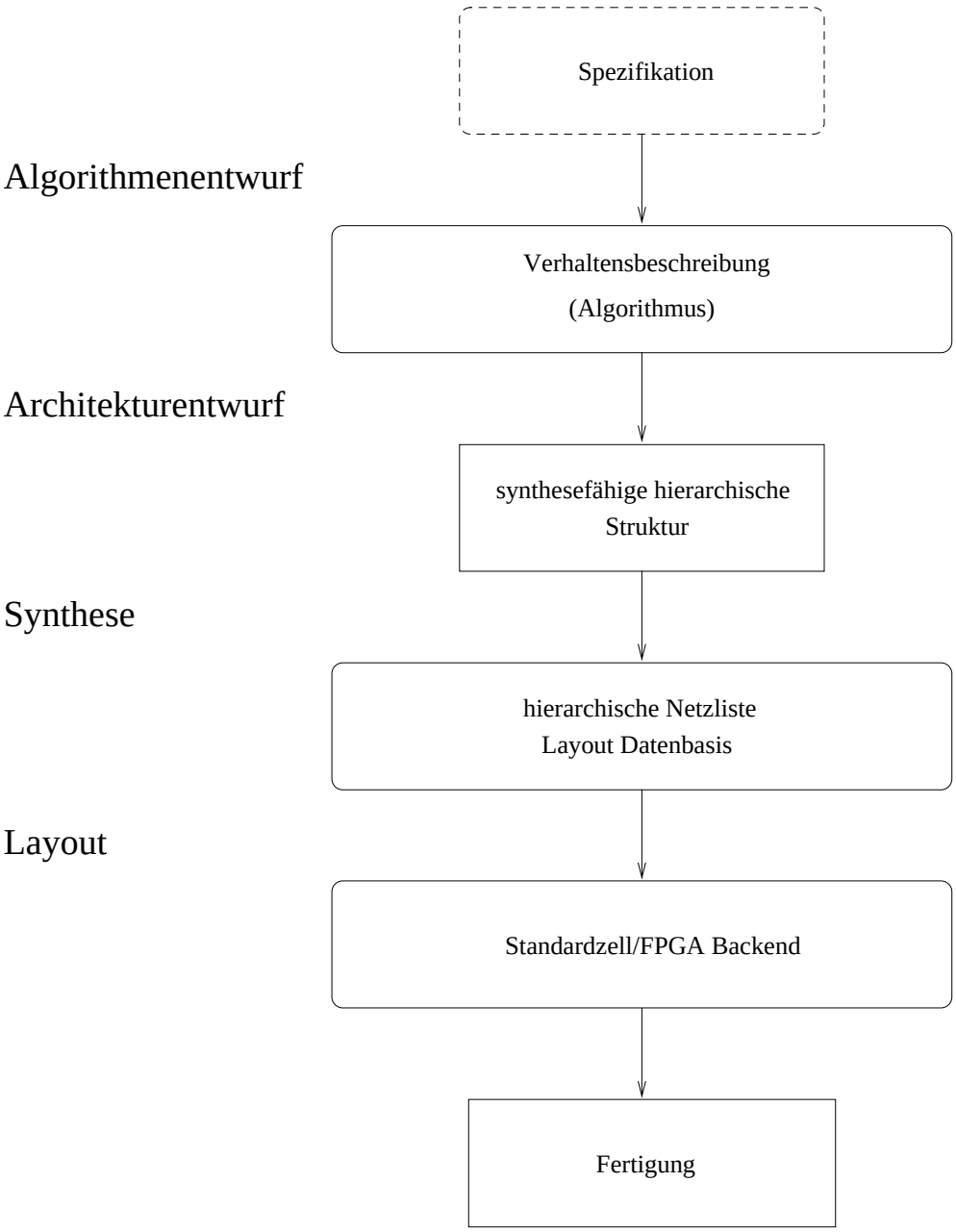

Abbildung 6.1: allgemeine Vorgehensweise bei der Entwicklung einer integrierten Schaltung

Über die Entity können VHDL-Prozessen spezielle Werte zugewiesen werden und resultierende Antworten eines Entwurfes an definierten Output Port Signalen abgegriffen werden und graphisch dargestellt werden. Mit Hilfe dieser Ports kann über Simulationen geprüft werden, inwieweit Teilhardwarestrukturen funktionsfähig sind.

Es können drei Aspekte von Simulationen unterschieden werden:

- Die Funktionalität der Schaltung, beschrieben durch die logische Funktion der dazu gehörenden Schaltnetze *(funktionale Simulation).*

- Das detaillierte zeitliche Verhalten der Schaltungen unter Berücksichtigung der auftretenden Verzögerungszeiten der verwendeten Logikgatter *(timing Simulation).*

- Das detaillierte zeitliche Verhalten der verwendeten Standardzellen und der sich durch das Placement & Routing ergebenden Leitungsverzögerungszeiten *(backannotation Simulation).*

Alle drei Aspekte können und müssen durch eine jeweilige Simulation mit geeigneten Stimuli dahingehend untersucht werden, ob sie der geforderten Spezifikation entsprechen.

Für das Aufspüren von Fehlern in VHDL-Beschreibungen ist es unerläßlich, alle Signale und Variablen zu beobachten, oder gar der ursprünglichen Beschreibung zuordnen zu können [MAE93]. Sollte die Simulation nicht das gewünschte Ergebnis erbringen, so muß die VHDL-Verhaltensbeschreibung neu überarbeitet werden, um dann in einer weiteren Simulation neu geprüft zu werden.
Hierbei ist es sehr hilfreich, wenn man Gatterverzögerungszeiten zunächst aus Geschwindigkeitsgründen nicht berücksichtigt. Diese Simulationsart wird als *funktionale Simulation* bezeichnet [SYN93] [MAE93]. Von dem verwendeten *Synopsys-Simulator* aus, wird die funktionale Simulation eines Schaltungsmodells unterstützt.

Nach der Architekturentwurfstufe wird in der Regel eine *timing Simulation*, mit allen Verzögerungszeiten der Logikgatter durchgeführt. Da zu diesem Zweck die physikalische Realisierung der Schaltung bekannt sein muß, müssen diese Verzögerungszeiten von einem entsprechenden Entwurfswerkzeug aus dem Entwurf extrahiert werden, nachdem sie auf die Logikzellen eines Herstellers gemappt worden sind. Das Entwurfswerkzeug kann zu diesem Zweck optional eine Strukturbeschreibung in VHDL als hierarchische Netzliste erzeugen, in der diese Verzögerungszeiten enthalten sind.
Sowohl die hierarchische Strukturierung der ursprünglichen VHDL-Beschreibung des Entwurfs, wie die auch eventuell verwendeten Namen für Signale, Instanzen, Automaten, Automatenzustände und die I/O Signale des Entwurfs stimmen in der hierarchischen Netzliste mit denen in der VHDL-Beschreibung überein, sind jedoch zusätzlich erweitert. Höhere Datentypen werden durch entsprechende Bitvektoren ersetzt. Dieses führt im allgemeinen zu einer Unübersichtlichkeit, die das *Debuggen* des Entwurfs erschweren kann.

Nach dem abschließenden Placement & Routing d.h. nach der Layoutgenerierung findet eine *backannotation-Simulation* unter Beachtung der *Leitungsverzögerungszeiten* des physikalischen Layouts statt. Erst wenn diese abschließende Simulation ein korrektes Verhalten zeigt, kann der Entwurf gefertigt werden.

6.1.3 Synthese

Simulation ist nur ein wichtiger Schritt im Entwurfsablauf einer integrierten Schaltung. Einen andern wichtigen Teilschritt im Ablauf einer ASIC Entwicklung stellt die *Logiksynthese* mit Hilfe der aus dem VHDL-Code generierten hierarchischen Netzliste dar.
Allgemein wird als Synthese beim Standardzellentwurf die Transformation von *Verhaltensbeschreibungen* in *Strukturbeschreibungen* aus Elementen niedrigerer Abstraktionsebenen bezeichnet, die einer Logikzellbibliothek eines Herstellers entsprechen [SMI96]. Es ist möglich eine Synthese in den Bereich der Registertransferebene (RT-Ebene), teilweise sogar bis hin zur Hauptblockebene problemlos durchzuführen. Die einzelnen Komponenten sind dabei durch Verhaltensbeschreibungen spezifiziert [MAE93].

Im nachhinein soll genauer auf den Syntheseprozeß eingegangen werden:
Die Hardwarebeschreibungssprache VHDL deckt mit seinen Ausdrucksmöglichkeiten alle im digitalen IC Entwurf verwendeten Abstraktionsebenen ab, während andere Beschreibungen zur Erstellung einer integrierten Schaltung mit dem Datentypen „0“ und „1“ nur einen Aufbau aus Logikgattern nachbilden. Heutzutage werden bei kommerziellen Werkzeugen die Logiksynthese von Schaltnetzen, die Synthese von Datenpfaden für Register und Funktionseinheiten und die Synthese endlicher Automaten relativ problemlos beherrscht. Das Verhalten kompletter Systeme mit abstrakten Datentypen auf RT-Ebene kann modelliert und simuliert werden.
Ein Problem bei der Entwicklung integrierter Schaltungen ist, dass nicht jede VHDL Beschreibung möglicher Ausgangspunkt für Synthesetools ist, d.h. auf Logikzellen eines Herstellers abbildbar ist. Wie in Programmiersprachen lassen sich auch in VHDL Verhaltensweisen auf viele verschiedene Arten ausdrücken, wobei abhängig auch vom Werkzeug einige besonders gut für die Synthese geeignet sind, andere wiederum nicht synthetisierbar sind oder zu uneffizienten Hardwarelösungen führen.
Selbst einfache Ausdrücke, wie die Art der Zuweisungen an Prozesse, entscheiden über die Synthetisierbarkeit des VHDL-Codes [MAE93]. Ist ein VHDL-Code mit Hilfe eines Synthesewerkzeuges synthetisierbar, so spricht man von *synthesis ready code*.
Neben der Art der Sprachbeschreibung wirken sich vor allem noch die Art der Zielbibliothek, d.h. welche Elemente stellt der Hersteller zur Verfügung und die Art und Weise, wie der Benutzer den Syntheseprozeß steuert, auf des Syntheseergebnis aus.
Als noch wichtiger im Bezug auf gute Syntheseergebnisse erscheinen die vielfältigen Möglichkeiten, die das Synthesewerkzeug selbst zur Verfügung stellt.
Hier kann über die Angabe folgender Randbedingungen, das Syntheseergebnis entscheidend beeinflußt werden:

- Zeitabhängigkeiten zwischen Signalen,
- Schranken für Flächen,
- Belastung vorangeschalteter Stufen und
- Treiberleistung einzelner Gatter.

Durch diese Angaben wird einerseits der Suchraum des Synthesesystems eingeschränkt, andererseits können Anforderungen der späteren Schaltungsumgebung im Bezug auf schnellere und effizientere Ergebnisse berücksichtigt werden.

Nach der Logiksynthese können vom Synthesewerkzeug die resultierenden *Gatterverzögerungszeiten* und eine geschätzte maximale Chipfläche, also die Summe der benötigten Fläche für die verwendeten Logikzellen plus einen Schätzwert für den Flächenbedarf der Verdrahtung angegeben werden. Zusätzlich kann eine Reihe anderer Syntheseergebnisse extrahiert werden, wie beispielsweise verwendete Ressourcen, Verlustleistung und Temperaturverhalten.
Ist das Syntheseresultat nicht zufriedenstellend, so kann auch ein Eingriff in die VHDL-Verhaltensbeschreibung andere Resultate bringen. Es liegt überwiegend im Ermessen des Entwicklers, wie effizient eine VHDL-Verhaltensbeschreibnug in Hardware umgesetzt werden kann.

6.2 Überlegungen bei der Entwicklung von Verhaltensmodellen

Ausgehend von der Spezifikation der benötigten Teilkomponenten, können zur Realisierung eines ASICs ein oder mehrere VHDL-Modelle entwickelt werden und durch Simulationen in ihrem Verhalten überprüft werden. Die Spezifikation kann aus einer natürlichsprachlichen Beschreibung, Blockschaltbildern oder Flußdiagrammen bestehen.
Es liegt im Ermessen des Entwicklers und ist von der Art der Problemstellung abhängig, in welcher Art die erste Version des Verhaltensmodells erstellt wird.
Ist die Spezifikation oder der Algorithmus für den Entwickler in wesentlichen Teilen sehr ungenau, so können das Verhaltensmodell oder Teile davon zuerst funktional in programmähnlicher Form entwickelt werden. Durch den sequentiellen Ablauf können so Unklarheiten auf der Seite des Entwicklers beseitigt werden und die Spezifikation nachträglich ergänzt werden. Diese Vorgehensweise ist einfach zu beherrschen und schnell zu realisieren. Vorteilhaft scheint dieses Vorgehen besonders dann, wenn unterschiedliche Realisierungen denkbar sind und Klarheit darüber geschaffen werden soll, welche Realisierung die meisten Vorteile bietet.
Sind die zu implementierenden Algorithmen durch bestimmte Spezifikationen fest vorgegeben, dann kann ein Verhaltensmodell direkt auf eine spätere Synthetisierbarkeit hin entwickelt werden. Aspekte der Synthese, wie ein bestimmtes Timingverhalten oder Flächenbedarf, müssen hierbei berücksichtigt werden.

6.2.1 Vorgehen bei der Entwicklung des ASICs

Bei der Entwicklung des ASICs waren die Überlegungen und Vorgehensweisen die folgenden:
Zu Beginn der Arbeit wurden grundlegende Beschreibungen zum Einsatzgebiet des ASICs gemacht. Aus weiteren Überlegungen ergab sich, welche Funktionen vorzugsweise in den ASIC verlegt und welche von einer zusätzlich aufgesetzten Hardware ausgeführt werden sollen. Hier wurde auch die Schnittstelle festgelegt, die eine getrennte Ausgabe von Real und Imaginär Teil erlaubt. Es wurde folgendes festgelegt:
Aufgrund des Haupteinsatzgebietes als Vorverarbeitungseinheit in der Nachrichtentechnik, sollen alle nicht zeitkritischen und komplizierten Berechnungen einer aufgesetzten Hardware überlassen werden. Eine Teilverlagerung in den ASIC würde nur die Komplexität stark erhöhen und keine Vorteile bringen. Das durch die Multiratenfilterkaskade und den Steuerautomaten realisierte *Resampling*, macht die Kommunikation und damit Verarbeitung durch leistungsschwächere Einheiten möglich.

In weiteren Überlegungen wurde der Gesamtfunktionsumfang auf sinnvolle, in Hardware realisierbare Funktionsblöcke beschränkt. Alle Teilfunktionsblöcke wurden auf RT-Ebene beschrieben.
Um geeignete Teilstrukturen für eine ASIC-Gesamtimplementation zu finden, wurden zunächst verschiedene digitale Hardwarefilter- und DDS Hardwarestrukturen als Basisstrukturen realisiert und auf Geschwindigkeit und Flächenverbrauch hin untersucht. Die für den ASIC geeignetste Realisierung sollte in die ASIC-Gesamtstruktur integriert werden. Die Hardwarerealisierung ergab dabei neue Erkenntnisse, wie sehr reguläre Strukturen in VHDL effizient umzusetzen sind.
Des weiteren wurde überlegt, wie eine der beiden vorgestellten Wandlerstrukturen eines reellen in ein analytisches Signal aus Abschnitt 2.13, am einfachsten in die ASIC-Architektur zu integrieren sind.
Nach einigen Überlegungen wurde eine Quadraturmischer-Hardwarestruktur ausgewählt, die zur Gesamtarchitektur des ASICs paßte und sich leicht implementieren ließ. Sie bot zudem die in der Nachrichtentechnik oft gebrauchte Funktion des Frequenzversatzes.

Bei der ASIC-Realisierung wurde auf zwei Entwurfspunkte besonderen Wert gelegt:

- Die einzelnen Teilrealisierungen sollten so unabhängig wie möglich voneinander entworfen werden, so das eine leichte Portierung von Einzelkomponenten auf einen FPGA möglich ist. Als eigenständige Einheit realisiert, könnte eine entsprechende Realisierung der weiteren Lehre an der Universität Hamburg dienen. Als FPGA Realisierung wurde speziell der DDS Generator in Betracht gezogen, der in der digitalen Nachrichtentechnik vielfältige Einsatzgebiete hat. Auch eine FIR-Teilfilterstruktur könnte weitere Verwendung finden. Inwieweit dieses aus Platzgründen möglich ist, wurde nicht weiter untersucht.

- Zudem sollte die VHDL-Gesamtbeschreibung durch die Verwendung von *generischen Paramentern* auf eine mögliche leichte Skalierbarkeit hin ausgelegt werden und damit eine Möglichkeit der leichten Anpassung der Hardwarerealisierung an andere Anforderungen geschaffen. Eine solche VHDL-Beschreibung zeichnet sich durch eine leichte Modifizierbarkeit und damit hohe Flexibilität aus.

Durch diese Maßnahmen, insbesondere im Hinblick auf die zu verarbeitenden Bitbreiten, ist ein Entwurf für spezielle Einsatzgebiete leicht und schnell durchzuführen.

Bei der Entwicklung des ASICs wurde von vornherein versucht, ein sehr hardwarenahes Verhaltensmodell zu beschreiben. Dieses war möglich, da sich die zu realisierende Struktur in überschaubare Einzelblöcke mit einem einfachen Datenfluß zerlegen lies.
So war es schon im Verhaltensmodell möglich, das Zusammenspiel und die Schnittstellen der zu realisierenden Teilkomponenten aufeinander abzustimmen.
Ausgehend von einer hardwarenahen Beschreibung mit entsprechend definierten Schnittstellen, wurden nach der Simulation auf Korrektheit, die Teilimplementierungen auf Logik synthetisiert.
Abschließend konnte durch ein Gesamtsimulationsmodell das ASIC auf seine Funktionsfähigkeit hin überprüft werden. Die Verhaltensmodelle wurden auf SPARC Workstations entwickelt und mit den an der Universität Hamburg vorhandenen Synopsys-Simulationswerkzeugen simuliert. Die Synthese wurde mit dem *Design Compiler* von Synopsys durchgeführt. Für das abschließende physikalische Layout wurden, wie schon erwähnt, das *Design Framework (DFII)* von Cadance verwendet.

6.2.2 Allgemeine Realisierungsabschätzungen

Es wurden schon im Vorfeld der eigentlichen ASIC Entwicklung versucht, einige Abschätzungen im Bezug auf den Realisierungsaufwand zu machen und diesen durch die Wahl von speziellen Strukturen möglichst gering zu halten. Aufgrund der sehr regulären Struktur der realisierten Filter, konnte eine Flächenabschätzung für die Multiratenfilterblöcke durchgeführt werden. Durch die Aufteilung der Filterstrukturen in einzelne Blöcke (MAC Zellen) war es nach der Synthese möglich, den Flächenbedarf für *n* MAC Zellen also *n* Filterkoeffizienten zu errechnen. Man erinnere sich, das sich ein digitaler FIR-Filter durch Kaskadierung von MAC Einheiten ergab.
Auf ähnlichem Wege konnte auch die Zykluszeit der Filterrealisierungen ermittelt werden, da die einzelnen arithmetischen Einheiten jeweils durch Register getrennt wurden. Nach durchgeführten Vorabuntersuchnungen über die zu verwendeten Arithmetikeinheiten, mußte davon ausgegangen werden, dass die verwendeten parallelen Filterstrukturen und damit die Multiratenfilterblöcke den größten Flächenverbrauch aufweisen werden.
Von dieser Annahme aus wurde versucht, die Filter so auszulegen, dass sie mit einer möglichst geringen Anzahl von parallelen Multiplizierern auskommen. Eine Filterstruktur unter Ausnutzung der Symmetrieeigenschaften der Filterkoeffizienten erwies sich als geeignet und wurde implementiert. Eine serielle Filterrealisierung und damit eine große Flächenersparnis wurde dabei aufgrund des Geschwindigkeit nicht in Betracht gezogen. Eine Implementierung würde die gemachte Spezifikation in dem Geschwindigkeitspunkt verletzen.
Des weiteren war davon auszugehen, dass durch die große Anzahl von benötigten Datenbussen zwischen den einzelnen MAC Zellen, der benötigte Verdrahtungsflächenbedarf nicht zu unterschätzen sein wird. Besonders wenn man sich vor Augen hält, dass bei dem nur zur Verfügung stehenden 2-Lagen-Prozess die Verdrahtung bei großen Busstrukturen immer eine große Fläche einnimmt. Noch gravierender könnte dieses bei der Gesamtrealisierung des ASICs und den hier realisierten Bussen ausfallen. Diese Annahme wurde auf Grund des zur Verwendung stehenden 0.6 µm AMS 2 Lagen Verdrahtungsprozesses gemacht.
Aus dieser Überlegung wurden die Busse und damit auch die Arithmetik in einer ersten Version des ASICs auf 8 Bit begrenzt. Für die genauen Logiksyntheseresultate der realisierten Teilkomponenten in Bezug auf Fläche und Geschwindigkeit sei noch einmal auf Kapitel 3 hingewiesen.

6.2.3 Simulation der Verhaltensmodelle

Um die Korrektheit der erstellten Verhaltensrealisierungen zu überprüfen, wurde nach jeder Umsetzung von Teilkomponenten in VHDL eine Simulation durch eine dafür entwickelte Simulationsumgebung durchgeführt.
Zur Verifikation der Funktion der FIR Filterstrukturen wurden folgende Ideen verfolgt:
Mit Hilfe von *Matlab* [MAT95] und einigen in der *Signalprozessorbibliothek* zur Verfügung gestellten Routinen zur Berechnung von FIR Filterkoeffizienten, konnten 16 Filterkoeffizienten für ein Tiefpaßverhalten errechnet werden. Nach der Skalierung auf 8 Bit, wurden mit Hilfe einer dafür erstellten Simulationsumgebung die einzelnen Filterkoeffizienten dem VHDL-Simulationsmodell übergeben.
Wie in Abschnitt 2.9 nachzulesen, entspricht die Impulsantwort eines FIR-Filters auf einen Diracimpuls, seinen Filterkoeffizienten. Nach der softwaremäßigen Erzeugung eines Diracimpulses durch die Simulationsumgebung, konnte graphisch in einem Ausgabefenster verfolgt werden, wie nach jedem Takt ein Filterkoeffizient nach dem anderen ausgegeben wurde. Das Ausgabeergebnis war als ein Beweis dafür anzusehen, dass die realisierte Hardware FIR-Filterstruktur ein korrektes Arbeitsverhalten aufweisen mußte.

Des weiteren wurden mit Hilfe eines Frequenzgeneratorprogrammes zwei unterschiedliche 8 Bit-Sinus-Singale im unteren Frequenzbereich mit einem Frequenzabstand von 1 Khz generiert.
Nach einer additiven Mischung dieser Signale, konnte mit entsprechenden Filterkoeffizienten zur nachträglichen Signaltrennung, das Signalgemisch wieder separiert werden. Die Filterkoeffizienten wurden vorab in Matlab durch Simulation bestimmt.
Eine nachträglich durchgeführte *Fourier Analyse* (FT) mit den Ausgabedaten der Filterstruktur ergab, dass diese Hardwarestruktur mit den entsprechenden Filterkoeffizienten ein Tiefpaßverhalten aufweist. Der Spektralanteil der höchsten generierten Frequenz fehlte. Ein zusätzlich erfolgter abschließender Audiohörtest bestätigte zudem das Fehlen dieser Frequenz.

Die Funktionsweise des DDS Frequenzgenerators wurde überprüft, indem seine Ausgangsdaten mit Hilfe eines dafür geschriebenen Programms zur Darstellung von Wellenformen auf einem Ausgabefenster der SPARC-Workstation ausgegeben wurde. Durch Änderung des Phasenschritts, konnte in einem gewissen Rahmen die Frequenzänderung verfolgt werden. Um Gleichung 3.5 in Abschnitt 3.6 zu überprüfen, wurde ein Phasenschritt für eine bestimmte Frequenz errechnet und dem Simulationsmodell eines DDS Frequenzgenerators übergeben. Die ausgegeben Samplewerte des DDS Generators wurden dabei direkt zur weiteren Untersuchung in einer Datei gespeichert. Die VHDL Syntax enthält entsprechende Befehle zur direkten Datenspeicherung.
Mit Hilfe von Matlab und entsprechenden Routinen zur Fourier-Analyse konnte auch die Richtigkeit der Gleichung 3.5 bestätigt werden. Die realisierte Hardware war damit in der Lage, gleichzeitig beliebige Sinus- und Cosinussignale einer vorher festgelegten Frequenz zu erzeugen.
Nachdem die Einzelkomponenten getestet waren, konnte eine funktionale Simulation des Gesamtchips durchgeführt werden. Aufgrund der Komplexität der ASIC Gesamtrealisierung, sollte allerdings auf einen vollständigen Test verzichtet werden.
Um trotzdem einen ersten groben Überblick über die Funtionsweise zu erhalten, wurden über eine Simulationsumgebung fortlaufende Eingabewerte für Eingangssamplewerte und Filterkoeffizienten vom Wert *eins* generiert und dem VHDL Modell der Gesamt ASIC Realisierung übergeben. Der endliche Steuerautomat wurde dabei so konfiguriert, dass das Beispiel von Abbildung 5.5 in Kapitel 5 realisiert wurde. Nach einer graphischen Simulationsausgabe als Wellenform, konnte die korrekte Arbeitsweise der Multiratenfilterkaskaden und das gewünschte Zusammenspiel der einzelnen realisierten Teilkomponenten bestätigt werden.

Kapitel 7

Bewertung und Ausblick

Im letzten Abschnitt wurde grundlegend aufgezeigt, in welchen Schritten ein ASICs in der Hardwarebeschreibungssprache VHDL entwickelt werden kann.
In diesem Kapitel sollen nun Ergebnisse im Bezug auf Fläche und Geschwindigkeit der ASIC-Gesamtrealisierung vorgestellt werden. Es wird dabei auf Logiksynthese und die tatsächlichen Ergebnissen nach der Durchführung eines vollständigen physikalischen Layouts eingegangen. Des weiteren werden Probleme und Lösungswege aufgezeigt, die während der Entwicklung aufgetreten sind. Ein Vergleich mit einem anderen kommerziellen Chip, der ähnliche Aufgaben erfüllt und einige Verbesserungen, die in einer weiteren Realisierung berücksichtigt werden können, schließen dieses Kapitel ab.

7.1 Ergebnisse

Der gesamte Entwurf und die einzelnen Teilentwürfe liegen als Verhaltensbeschreibung in der Hardwarebeschreibungssprache VHDL vor.
Ausgehend von den Teilrealisierungen des ASICs, wurden zur Auswertung der Realisierungsergebnisse in einem ersten Schritt die einzelnen VHDL-Verhaltensmodelle für eine 8 Bit und eine 14 Bit-ASIC-Version mit dem *Synopsys Design Analyzer* auf Logikebene synthetisiert. In einem darauffolgenden Schritt, sollte ein physikalisches Layout mit Hilfe von Standardzellen durchgeführt werden.
Zu diesem Zweck wurde aus den Daten der Logiksynthese eine *Verilog Datei* der ASIC-Gesamtrealisierung generiert und mit Hilfe von *Cadence Design Framework (DFW II)* Tools ein abschließendes *Placement & Routing* von Standardzellen durchgeführt. Die Cadence DFW II Tools erlauben eine vollständige physikalische Synthese mit Hilfe einer Standardzellbibliothek eines Herstellers. Für den ASIC kam ein AMS 0.6 µm 2 Lagenprozeß zum Einsatz [AMS98].

7.1.1 Geschwindigkeit

Um einen Schätzwert für die maximale Taktrate eines ASICs zu erhalten, lies sich der *längste Pfad* zwischen zwei Registern über das Synthesetool angeben. Dieser wurde für die zu entwickelnde ASIC-Realisierung zwischen dem Register eines Ausgangsmultiplexers (OutMUX) und dem Register der Eingangsmultiplizierstrukturen (DataMUX) der Multiratenfilterkaskade bestimmt.
Die maximale Taktrate resultiert damit aus den Gatterdurchlaufzeiten des Addierers und Multiplizierers der komplexwertigen Hardwarestruktur.
Als erste Näherung für die 8 Bit-Realisierung gab das Synthesetool eine Verzögerungszeit von 9.54 ns an, was einer maximalen Taktrate von 104.8 Mhz entspricht.

Zum Vergleich wurde zusätzlich eine 14 Bit-Realisierung mit einer maximalen Taktrate von 98 Mhz auf Logikebene synthetisiert. Der Unterschied zur 8 Bit-Realisierung beläuft sich dabei ausschließlich auf die Größe der internen Datenbusse und Datenregister. Die sich ergebende Differenz für die maximale Taktrate ergibt sich überwiegend aufgrund der größeren Arithmetikeinheiten der 14 Bit-Version des ASICs [SYN93].

7.1.2 Fläche

Nach der Synthese auf Logikebene mit dem Synopsys Synthesetool, kann zusätzlich zur Taktabschätzung eine Flächenabschätzung der einzelnen Teilrealisierungen und damit des gesamten ASICs vorgenommen werden. Das Synthesetool analysiert die VHDL Beschreibung und gibt die Fläche des Logikbedarfs und einen Schätzwert für die benötigte Verdrahtung an. Man erinnere sich in diesem Zusammenhang, dass aufgrund der gemachten Spezifikation für die Realisierung der Arithmetik, schon ausschließlich schnelle und damit auch relativ große *Brend Kung* (BK*)* Addierer und *Wallace Tree* (Wall) Multiplizierer verwendet wurden. Vergleiche hierzu Abschnitt 3.4.
Für einen Überblick faßt Tabelle 7.1 einige Flächenabschätzungen der Logiksynthese für einzelne 8 Bit-ASIC Teilrealisierungen zusammen.

Komponenten der 8 Bit ASIC Realisierung:	Logikflächenabschätzung für einen 0.6 µm AMS Prozeß
Multiratenfilterkaskade SymFir Filterstruktur mit 25 Filterkoeffizienten (8 Bit)	12.72 mm^2
8 Bit DDS Sinus/Cosinus Frequenzgenerator	3.75 mm^2
Komplexwertige Struktur - 4 Multiplizierstrukturen (8 Bit) - 2 Addierstrukturen (16 Bit) - 2 Eingangsmultiplexer	2.75 mm^2
SteuerFSM (Finite State Machine)	0.15 mm^2
Logik- und Verdrahtungsfläche (gesamt)	35.83 mm^2

Tabelle 7.1: Flächenabschätzung auf Logikebene von 8 Bit-Teilrealisierungen es ASICs

Auffallend ist die Größe der Multiratenfilterkaskade von 12.72 mm^2, aufgrund der internen parallelen Filterrealisierungen. Durch die doppelte Auslegung dieser Einheit, bedingt durch die komplexe Signalverarbeitungsstruktur, nehmen diese Teilrealisierungen die größte Fläche des ASICs ein. Aus der Differenz der angegebenen Fläche und der Summe aller Teilkomponenten ergibt sich die geschätzte Fläche für die Verdrahtung.
Eine zusätzlich durchgeführte Logiksynthese der 14 Bit-Version des ASICs brachte eine Flächenabschätzung von 74 mm^2. Der Flächenzuwachs von fast 50% resultiert überwiegend aus der Vergrößerung der internen Busse und der dafür benötigten Verdrahtung.
Wie sich aus Erfahrungen im Arbeitsbereich TECH zeigt, ist insbesondere beim Einsatz von großen Busstrukturen die gemachte Abschätzung der Logiksynthesetools mit Vorsicht zu behandeln. Aufgrund der Realisierung einer großen Anzahl von Busstrukturen im ASIC, müssen Unterschiede einkalkuliert werden. Für einen ersten Überblick sind die Ergebnisse allerdings

brauchbar. Den tatsächlichen Flächenbedarf wird erst später das physikalische Standardzelllayout geben.
Es soll auf zwei wichtige Optionen, die im Zusammenhang mit dem Synthesetool stehen und direkt das Syntheseergebnis beeinflußt haben, eingegangen werden.
Zur Durchführung einer Synthese auf Logikebene wurden dem Synthesetool optional folgende zwei Parameterworte angegeben:

- ein Wire Load Modell und

- ein Map Efford.

Durch die Wire Load Modell Angabe wird die Treiberleistung und damit die Größe interner Treiberstufen im ASIC spezifiziert [SYN93]. Der voreingestellte Wert hat insbesondere Auswirkungen auf die Fläche der Hardwarerealisierung. Als Wire Load Modell wurde ein Wert von *10K* vorgegeben.
Als weitere Option wurde bei allen Logiksynthesen ein Map Efford von „high“ angegeben.
Dem Synthesetool wird hierdurch angewiesen, eine möglichst effiziente Logikrealisierung mit möglichst wenig Logikstufen zu finden [SYN93]. Aufgrund der Suche nach einer geeigneten Realisierung kann die Logiksynthese entsprechend lange dauern.
Zum Ende der Diplomarbeit ergab sich die Verfügbarkeit eines 3-Lagen 0.6 µm Prozesses [AMS98]. Die Ergebnisse eines nachträglich durchgeführten physikalischen Layouts ergaben bei der 14 Bit ASIC-Version eine Reduktion der Fläche des ASICs von 72 mm^2 auf 41 mm^2. Die benötigte Chipfläche reduzierte sich gegenüber dem 2-Lagen Prozeß um fast die Hälfte.

7.1.2 Zusammenfassung und Bewertung

Der ASIC wurde nach der Validierung seiner Funktionen erfolgreich auf Logikebene synthetisiert. Die probeweise Synthese ergab, dass bei einer 8 Bit-Version des ASICs mit einem Gesamtflächenbedarf von mindestens 35.83 mm^2 zu rechnen ist. Die Taktrate wurde hier mit 104.8 Mhz festgelegt.
Die Erweiterung der Busbreiten auf 14 Bit ergab einen Flächenbedarf von mindestens 74 mm^2 und einer maximalen Taktfrequenz von 98 Mhz.
Erstaunlich war der nur ungefähr lineare Anstieg der Fläche, gegenüber einer vorher gemachten Annahme, die interne Buserweiterung auf 14 Bit führe zu einem mehr als quadratischen Zuwachs an benötigter Fläche.
Um eine Vorstellung über den tatsächlichen Flächenbedarf zu erhalten, wurde zusätzlich zur Logiksynthese mit Cadence Plazierunges und Verdrahtungs Tools ein physikalisches Layout auf Standardzellebene durchgeführt. Erstaunlicherweise vergrößerte sich die insgesamt benötigte Fläche im Gegensatz zur Flächenabschätzung des Synopsys Synthesetools nicht. Im Gegenteil, das Layout der 8 Bit-Version wurde auf 32 mm^2 spezifiziert. Dieser Wert deckt sich mit den Erfahrungen, die bisher an Arbeitsbereich TECH gemacht wurden. Die Anzahl der Standardzellen lag bei *28.277*.
Das Ergebnis der 8 Bit-Version konnte auch auf die 14 Bit-Realisierung bezogen werden. Die Fläche des Layouts lag bei 72 mm^2, entgegen einer vorher gemachten Annahme, die Busweiten würden bei dem verwendeten 2 Lagen-Verdrahtungsprozeß zu einer extremen Vergrößerung der benötigten Fläche führen.
Tabelle 7.2 faßt die Layout- und Logiksyntheseergebnisse der 8 und 14 Bit-Version des ASICs zusammen.

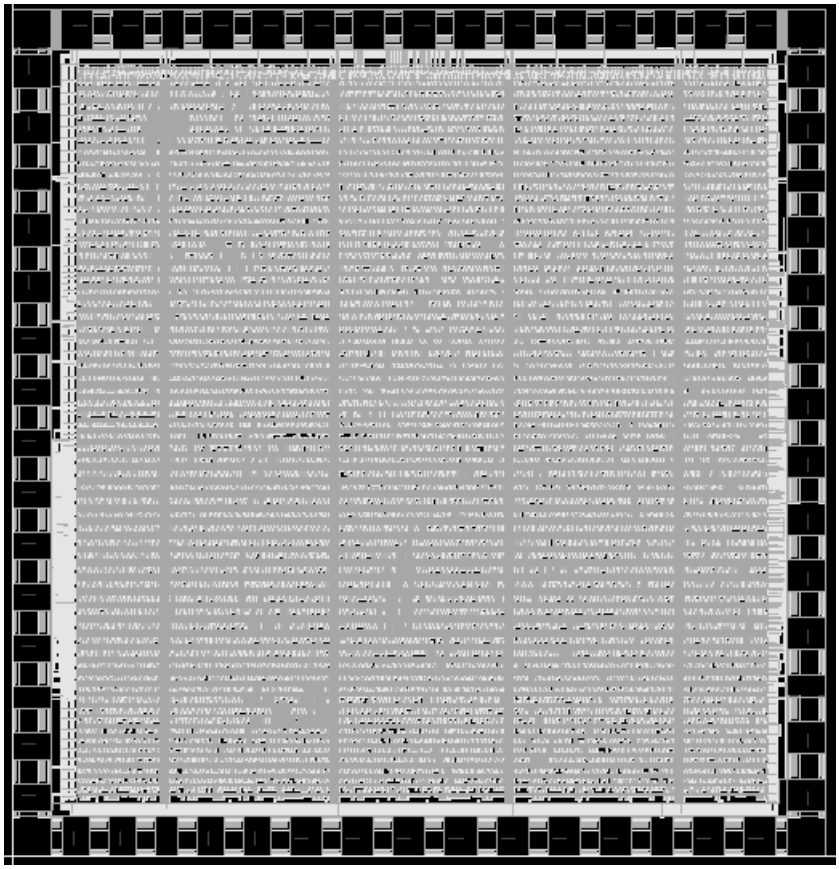

Abbildung 7.1: 8 Bit ASIC Layout

ASIC-Realisierungen:	Max. Taktrate		Fläche	
	log. Synthese	phys. Layout	log. Synthese	phys. Layout
8 Bit Realisierung	104.8 Mhz	104.8 Mhz	35.8 mm^2	32 mm^2
14 Bit Realisierung	98 Mhz	98 Mhz	74 mm^2	72 mm^2

Tabelle 7.2: Logik & Layoutergebnisse der 8 und 14 Bit ASIC Realisierung

Um eine Vorstellung über das Aussehen des generierten Layouts zu erhalten, illustriert Abbildung 7.1 die 8 Bit-ASIC Realisierung.

Das Layout stellt einen *core limitierten* Entwurf mit getrennt realisierten Versorgungsspannungen für den Core- und Padzellenbereich dar [MAE93].

Betrachtet man das Layout genauer, so fallen vertikal durchgeführte Artefakte auf. Sie sollen die zusätzlich implementierten VCC und GND Spannungsversorgungsleitungen darstellen, die der Stromversorgung der Standardzellen dienen. Zudem fällt die große Fläche für die Padzellen an den Rändern des Layouts auf.

Zu erkennen sind weiterhin die annähernd gleich verteilten Kanaldichten.

Lokale Konzentrationen von Leitungsverbindungen machen sind nur am linken unteren Layoutrand bemerkbar.

Zusammenfassend sollen einige durchgeführte Maßnahmen angesprochen werden, die nennenswert zur einer direkten Flächeneinsparung beigetragen haben. Im weiteren Verlauf dieses Abschnitts sollen in einem kurzen Einschub zusätzlich Möglichkeiten vorstellt werden, die zu einer direkten Flächeneinsparung führen würden.

- Die größte Einsparung an Fläche im ASIC gelang, indem die Symmetrieeigenschaften der Filterkoeffizienten ausgenutzt wurden. Die realisierten Filterstrukturen kamen mit fast der Hälfte der ursprünglich benötigten Multiplizierer aus. Da die Multiplziererstrukturen von den benötigten Arithmetikrealisierungen im ASIC die größte Fläche einnehmen und parallel realisiert vorliegen, resultiert hieraus eine Flächeneinsparung von fast 50% für eine Multiratenfilterkaskade. Die Ausnutzung der Symmetrie der verwendeten Filterkoeffizienten und die daraus resultierende Flächeneinsparung bei der Hardwarerealisierung wurde zufriedenstellend verwirklicht. Die realisierte 14 Bit-Version der ASICs wäre ohne diese Maßnahme unter der vorgegebenen Spezifikation nicht möglich gewesen.

- Weitere Logikfläche konnte durch die Ausnutzung der Symmetrieeigenschaften der Sinusfunktion eingespart werden. Vergleiche hierzu Abschnitt 3.8.

Es folgen einige Überlegungen und Bemerkungen, um die benötigte Fläche noch weiter zu reduzieren:
Eine Idee wäre, die diskreten Sinuswerte in einem *ROM* (Read Only Memory) zu speichern, anstatt sie direkt in die Logik zu übernehmen. Die Synopsys Komponentenbibliothek stellt zu diesem Zweck verschiedene ROM Instanzierungen mit variabler Speicher- Weite und - Tiefe zur Verfügung. Allerdings ergab eine Schätzung, dass aufgrund der kleinen Anzahl der verwendeten Sinus Werte, sich keine nennenswerte Flächenensparung ergeben würde.
Im Gegenteil, allein die Fläche der benötigten Adressierungslogik und weiterer zusätzlicher Overhead der Synopsys ROM-Instanzierung, könnte die Fläche schon größer ausfallen lassen, als bei der direkten Speicherung der Filterkoeffizienten in die Logik. Es ist allerdings, davon auszugehen dass sich bei einer großen Anzahl von Sinus Werten der Einsatz eines ROMs lohnt. Ferner ist davon auszugehen, dass aufgrund der Zugriffszeit des ROMs mit größeren Geschwindigkeitseinbußen gegenüber der direkten Speicherung der Sinus-Werte in der Logik zu rechnen ist. Weitere Untersuchungen wurden in dieser Arbeit nicht durchgeführt.

Eine weitere Verkleinerung der benötigten Fläche und zusätzlich eine Erhöhung der Geschwindigkeit würde durch den Einsatz eines *Full Custom Designs* erreicht werden. Besonders im Bezug auf den sehr regulären Aufbaus der Filterstrukturen, könnte eine solche Implementierung in Betracht gezogen werden.
Der Einsatz eines Full Custom Designs und der damit verbundenen Flächeneinsparung gegenüber dem verwendeten Standardzellentwurf resultiert dabei aus folgenden Begebenheiten:

- Bei einem Full Custom Entwurf entfallen zum Teil benötigte Standardzell Treiber.

- Um die Plazierung mit Standardzellen zu ermöglichen, muß jede Standardzelle die gleiche Höhe haben, egal was sie beinhaltet. Bei einem Full Custom Design würde dieses entfallen.

Es wäre auch daran zu denken, nur bestimmte Teilrealisierungen des ASICs als Full Custom Design auszulegen. Diese könnte dann als *Megazelle* in einen restlichen Standardzellentwurf implementiert werden [CAD97].

Es folgen einige weitere Bemerkung zum realisierten ASIC:
Wie schon mehrfach angesprochen, entspricht die Höhe der Taktrate des ASICs der maximal zu verarbeitenden Samplerate. Aufgrund der analytischen Verarbeitung der Signale, ist die Samplerate ein theoretisches Maß für die maximal zu verarbeitende Bandbreite. Da Übertragungskanäle in der Regel ein Tiefpaßverhalten aufweisen, wird die maximal mögliche Bandbreite eingeschränkt [KAM96].
Des weiteren ist eine Begrenzung der Bandbreite schon von der realisierten Architektur her gegeben. Die Hintergründe sollen nun angesprochen werden.
Es wurden eine Quadraturmischstruktur zur Wandlung des reellen in ein analytisches Signal in die ASIC Hardwarestruktur implementiert. Da die Wandlerstufe mit dem jeweiligen Sampletakt und dem reellen Eingangssignal des A/D Wandlers arbeitet, gilt das Nyquisttheorem, das die tatsächlich mögliche Bandbreite auf die Hälfte reduziert. Aufgrund der direkten Implementation und der damit verbundenen gleichen Taktrate des Quadraturmischers und der übrigen Einheiten, kommen die Vorteile der Verdopplung der Bandbreite einer analytischen Signalverarbeitung nicht zur Geltung. Abhilfe würde hier die Auslagerung der Quadraturmischeinheit mit einer doppelt so hohen Taktrate gegenüber den anderen Einheiten bringen, was allerdings einen Mehraufwand an Hardware bedeuten würde.

Zusammenfassend läßt sich sagen, dass die Vorgaben aus Kapitel 4 nicht zuletzt aufgrund der hohen Parallelität der verwendeten speziellen Strukturen und der schnellen Arithmetik mit den gegebenen Randbedingungen erreicht wurden. Als Endergebnisse existiert ein funktionsfähiger Standardzellentwurf.
Einsatzgebiete sind, wie schon angedeutet, hauptsächlich in der schnellen digitalen Datenübertragung zu sehen, z.B. in der Realisierung von Sende- und Empfangsfiltern [KAM96].
Des weiteren wäre ein Einsatzgebiet als Vorverarbeitungseinheit in Multiträgersystemen oder der Mobilkommunikation möglich.
Die im ASIC implementierte Idee der Steuerung einer Multiratenfilterkaskade mit Hilfe eines endlichen Steuerautomaten und die damit verbundene Umkonfigurierbarkeit in jedem Takt, könnte auch in anderen Anwendungen Verwendung finden. Einsatzgebiete könnten sich überall dort finden, wo große Frequenzbereiche überwacht werden müssen. Wie weiter angedeutet, könnte die eigentliche Überwachungsaufgabe (Auswertung) ein Standard PC oder Mikrocontroller durchführen.

Nun ist man bemüht, für ein erstes Prototyping einen Entwurf auf einen FPGA abzubilden. Inwieweit dieses mit den heute zur Verfügung stehenden FPGAs möglich ist, wurde nicht untersucht. Es ist anzunehmen, dass die große Anzahl von Bussen die Verdrahtungsressourcen der FPGAs überschreiten könnte. Aus diesem Grund stellt dieser Entwurf noch eine typische Anwendung für einen Standardzellentwurf dar.

7.2 Aufgetretene Probleme

Im Rahmen der Diplomarbeit, besonders hinsichtlich der Werkzeuge, die für den Entwurf des ASICs zur Verfügung standen, gab es einige Schwierigkeiten und Komplikationen. Diese sollen zuerst im nachhinein näher diskussiert werden.
Probleme mit den zur Verfügung stehenden Tools zum ASIC Entwurf:

Die Ausführungsgeschwindigkeit der am Arbeitsbereich TECH zur Verfügung gestellten SPARC Workstations reicht nicht aus, um die Synthese des realisierten ASICs in akzeptabler Zeit durchzuführen.

Die sehr umfangreichen Entwürfe, besonders die der Hardwarefilterrealisierungen, stellen ausgesprochen hohe Anforderungen an Rechenleistung und Speicherplatz. Aufgrund des Speichermangels war die 14 Bit-ASIC-Version nur in Teilen durch zusätzliche Interaktionen des Entwerfers synthetisierbar, die dann zu dem Gesamt-ASIC zusammengefügt werden mußten.
Als Beispiel seien die Multiratenfilterkaskaden genannt, die getrennt synthetisiert und nachträglich in die ASIC-Gesamtrealisierung implementiert werden mußten.

Zudem wurden Zeitattribute, die dem Synthesetool für eine mehr benutzerdefinierte Synthese übergeben wurden, zum Teil ignoriert. Es schien so, daß diese Zeitattribute nicht in jede Hierarchiestufe des ASICs übergeben wurden. Anstelle schneller Carry Look Ahead Addierer (CLA) wurden langsame Ripple Carry Adder (RPL) verwendet. Erst eine direkte Einbindung eines speziellen Addierers in den VHDL-Code brachte die gewünschten Ergebnisse. Gleiches galt für die Multipliziererstrukturen.
Des weiteren dauerte die Simulation der Multiratenfilterkaskade aufgrund der vielen zu bearbeitenden Signale lange und war zudem noch sehr unübersichtlich. Besonders das Debuggen der Multiratenfilterkaskade machte Schwierigkeiten.
Probleme bei den Placement & Routing Tools gab es insbesondere bei den Datenkonvertierungen zwischen den eingesetzten Werkzeugen.
Durch falsche Einstellungen der sehr umfangreichen Werkzeuge, traten immer wieder Fehler auf, die ein weiteres Vorgehen bei der Synthese zum Teil unmöglich machten. Des weiteren traten Fehler der Werkzeuge und des Betriebssystems auf, die nicht näher erklärt werden konnten. Besonders das Layouten der 14 Bit-Version des ASICs machte Schwierigkeiten aufgrund der instabilen Werkzeuge.

Die Probleme bei der eigentlichen Realisierung in Hardware waren:
Da die Filterkoeffizienten mit Matlab und einer hohen Genauigkeit berechnet wurden, war eine nachträgliche Skalierung der Filterkoeffizienten nötig.
Es zeigte sich bei der nachträglichen Verifikation der Filterstrukturen, dass sich durch die Skalierung die Übertragungsfunktion ändert und so die gemachte Spezifikation verletzen kann. Aufgefallen ist dieses Problem unter anderem bei der Signaltrennung aus Abschnitt 6.2.3.
Gravierende Probleme gab es, wenn die erzeugten Frequenzen auf der Frequenzachse dicht beieinander lagen. Das Filterresultat war dann eine unvollständige Trennung der gemischten Sinusfrequenzen. Die Filterspezifikationen wurden aufgrund der Skalierung nicht mehr eingehalten, obwohl eine vorherige Simulation in Matlab mit größeren Bitweiten der Koeffizienten korrekte Ergebnisse geliefert hat.
Um diesem Problem von vornherein entgegenzuwirken, wurden in Matlab vorab Simulationen mit den später zum Einsatz kommenden Bitgrößen durchgeführt. Es sei bemerkt, dass dieses bei einer Hardwarerealisierung ein allgemeines Problem darstellt und daher immer durch entsprechende Simulation schon im Vorfeld ausgeschlossen werden muß.

7.3 Vergleich mit anderen Chips

Eigenschaften:	**Realisierter ASIC**	**Harris HSP 50214B Programmable Downconverter**
Max. Samplerate	104.8 Mhz 8 Bit Realisierung 98 Mhz 14 Bit Realisierung	65 Mhz 14 Bit Realisierung
Verarbeitende Bitgrößen	8 Bit / (14 Bit)	14 Bit
Anzahl der Filterkoeffizienten	je 25 bei analytischer Verarbeitung 49 bei reeller Verarbeitung	Bis 255 Filterkoeffizienten
Max. Dezimation	0 – 65535	4 – 16384
Analytische Signalverarbeitung	Ja	Ja
Reelle Signalverarbeitung	Ja	Möglich
Frequenz-Genauigkeit	32 Bit Phasenakkumulator	32 Bit Phasenakkumulator
Signalausgabe	real und imaginär getrennt	real und imaginär getrennt
Möglichkeit der Demodulation	Außerhalb der ASICs	Direkt im Chip möglich
Besonderheiten	- hohe Flexibilität durch FSM - in jedem Takt umkonfigurierbar - in jedem Takt ein anderer Frequenzbereich bearbeitbar	- Demodulationsaufgaben direkt im Chip möglich

Tabelle 7.3: Gegenüberstellung des ASICs mit einem anderen ähnlichen Chip

Der entwickelte ASIC unterscheidet sich von dem im Jahre 1998 vorgestellten und kommerziell erhältlichen Typ *HSP 50214B Programmable Downconverter* von Harris Semiconduktor [HAR98] insbesondere durch:

- die hohe Konfigurierbarkeit und
- die Möglichkeit des internen Umkonfigurierens in jedem Sampletakt durch die implementierte FSM und der sich daraus ergebenden Flexibilität, besonders im Bezug auf die Bearbeitung mehrerer Frequenzbereiche (Kanäle).

Diese Eigenschaften machen den ASIC gegenüber dem Harris Chip flexibler im Bezug auf bestimmte Einsatzgebiete wie Mehrkanalsignalverarbeitung. Die analytische Verarbeitung von Signalen stellt dabei die Basis dar, um eine Mehrkanalsignalverarbeitung akzeptabel durchzuführen.
Bei dem Harris Chip resultiert aus dem sehr speziellen Aufbau der internen Hardwarearchitektur die nur eine eingeschränkte Konfigurierbarkeit und damit Flexibilität erlaubt. Einen Vorteil im Bezug auf die direkte Einsetzbarkeit bietet der Harris Chip allerdings durch die internen Demodulationseinheiten für Amplituden- Frequenz- und Phasendemodulation und seine Standard µP Schnittstelle, die es dem Chip erlaubt, direkt mit einem konventionellen Mikroprozessor zu kommunizieren. Zudem wurde ein universeller serieller Port implementiert.

Für spezielle Aufgaben in der Mehrkanalüberwachung, in dem die Haupteinsatzgebiete des realisierten ASIC's zu sehen sind, ist der Harris Chip nur bedingt einsetzbar.
Über die benötigte Fläche, dem verwendeten Prozeß mit Strukturgrößen und die Anzahl der Verdrahtungslagen des Harris Bausteines konnte nichts in Erfahrung gebracht werden.

7.4 Mögliche Erweiterungen und Verbesserungen mit Ausblick

Nach der Realisierung oder dem praktischen Einsatz einer integrierten Schaltung, ergeben sich oft nachträglich einige Wünsche nach zusätzlichen oder erweiternden Funktionen und Teilrealisierungen. Da diese Erfahrungen aber erst beim Einsatz des ASICs gemacht werden, der direkte Einsatz jedoch nie erprobt wurde, mußten in diesem Fall vorab Annahmen über Erweiterungsmöglichkeiten gemacht werden.
Neben der Diskussion offensichtlicher Erweiterungen und Verbesserungen, wie einer Erhöhung der internen Busbreiten und der Anzahl der Filterkoeffizienten, sollen in den nachfolgenden Abschnitten wichtigere Erweiterungen und Realisierungsmöglichkeiten angesprochen werden.

Die Verwendung von 8 Bit-Datenbussen und fünfundzwanzig 8 Bit-Filterkoeffizienten der ASIC Realisierung für maximal – 48 dB reicht in der Nachrichtentechnik in den meisten Fällen nicht aus. Hier bedarf es einer Erhöhung der Busbreiten und damit des Dynamikbereiches in nachfolgenden Realisierungen.
Zu Beachten ist dabei, dass nach den Syntheseergebnissen die Verdrahtungsfläche bei dem zur Verfügungstehenden und relativ alten 0.6 µm AMS 2-Lagenprozeß eine große Fläche einnimmt. Entsprechende Untersuchengen mit einer Erhöhung der Busbreiten auf 14 Bit wurden durchgeführt und in dem vorherigen Abschnitt diskutiert.
Zur Lösung des Verdrahtungsproblems könnte ein Mehrlagenprozeß eingesetzt werden. Es ist anzunehmen, dass bei einem 4 oder 5 Lagenprozeß die Verdrahtung gänzlich unter den Standardzellen verschwindet.
Weiterhin kann durch die Verwendung von kleineren Strukturgrößen dem Flächenbedarf bei einer Realisierung mit großen Busbreiten entgegengewirkt werden.
Die Anzahl der Filterkoeffizienten ist in der ASIC-Grundstruktur aufgrund der Ausnutzung der Symmetrieeigenschaften der Koeffizienten auf effektiv je 25 (13 Koeffizienten + 12 Spiegelkoeffizienten) getrennt für Real- und Imaginär-Anteile beschränkt. Zur Einhaltung spezieller Spezifikationen könnte die Anzahl und die Genauigkeit der Filterkoeffizienten erhöht werden.
Wie schon angedeutet, ist die VHDL-Verhaltensbeschreibung der Filterstrukturen zu diesem Zweck durch die Verwendung von „generics" auf Erweiterbarkeit ausgelegt [MAE93].

Auf der Implementierungsseite wäre prinzipiell ein flexibles Interface nach außen wünschenswert. Hierdurch würde eine Kommunikation mit einem PC, DSP oder einem Mikrocontroller für bestimmte Applikationen mit einem geringen Aufwand möglich. Auch die Konfiguration der internen FSM könnte auf diesem Wege während der Betriebes durchgeführt werden. Hier wäre ein Einsatz für adaptive Anwendungen möglich.
Es soll in den nächsten Abschnitten auf spezielle Erweiterungen und Verbesserungen der ASIC Hardwarestruktur eingegangen werden.

7.4.1 Maßnahmen zur Durchsatzsteigerung

Um noch höherfrequentere Signale und damit größere Bandbreiten direkt verarbeiten zu können, ist ein Übergang auf eine kleinere und damit schnellere Technologien, als den derzeit verwendete 0.6 µm AMS Prozeß möglich. Dieser Schritt ergibt zwangsläufig eine Geschwindigkeitssteigerung der verwendeten Arithmetik und damit eine Erhöhung der zu verarbeitenden Bandbreite. Es ist aber auch möglich durch schaltungstechnische Maßnahmen die Geschwindigkeit zu steigern. Dieses soll Thema dieses Abschnitts sein.

Einige durchführbare Maßnahmen sollen im folgenden angesprochen werden:
Um die Geschwindigkeit der Arithmetik zu erhöhen, könnte die eingesetzte Multiplizierstruktur aufgrund seiner hohen Durchlaufzeit geändert werden. Dabei könnte die Möglichkeit genutzt werden, die Multiplikation für 2n Bit Zahlen auf mehrere kleinere und damit schnellere Multiplikationen für n Bit Zahlen aufzuteilen [KNU98]. In wieweit nach der Zusammenfassung der Teilergebnisse eine Geschwindigkeitssteigerung gegenüber einer direkten Realisierung möglich ist, müßte untersucht werden.

Eine andere Möglichkeit besteht in der Verkürzung des längsten Pfades. Wie schon angedeutet, kann über das Synopsys Synthesetool die Durchlaufverzögerung für den längsten Pfad zwischen zwei Registern bestimmt werden. Die maximale Taktfrequenz f kann aus der Signallaufzeit t dieses „längsten Pfades“ einer Realisierung berechnet werden. Wird die Durchlaufverzögerung mit t = 5 ns spezifiziert, so ergibt sich eine maximale Taktfrequenz von $f = 1/t = (1/5*10^{-9}) /1*10^{6} = 200$ *Mhz*.
Wie in Kapitel 3 schon angedeutet, ist dann die Verzögerungszeit der dazwischenliegenden Logikgatter ein Maß für die Höhe der maximalen Taktrate des ASICs. Nach den Angaben des Synthesewerkzeugen wurde der längste Pfad für die ASIC Gesamtrealisierungen durch die Logik des Addierers und Multiplizierers des komplexwertigen Hardwarestruktur festgelegt.
Zur Verkürzung dieses längsten Pfads könnte ein Register zwischen diesen arithmetischen Einheiten eingebaut werden, d.h. die arithmetische Verarbeitung als Pipeline implementiert werden. Ein entsprechendes Timing müßte dafür Sorge tragen, dass die Samplewerte zu den benötigten Zeiten der Multiratenfilterkaskade übergeben werden. In wieweit ein geeignetes Timing ohne die Zwischenspeicherung von Sampledaten und damit einer Änderung der Architektur des ASICs gefunden werden kann, wurde in dieser Diplomarbeit nicht weiter untersucht.

7.4.2 Zeitmultiplex Teilrealisierung zur Flächeneinsparung

Eine des öfteren durchgeführte Maßnahme den Flächenbedarf bei einer ASIC Realisierung einzusparen, liegt in der Mehrfachausnutzung von Teilrealisierungen. Aufgrund des großen Flächenbedarfes der realisierten Filterstrukturen, würde sich die Mehrfachausnutzung eines Multiratenfilterblocks anbieten.
Zu diesem Zweck könnte im ASIC ein entsprechendes Multiplexsystem implementiert werden, dass über eine zusätzliche Ablaufsteuerung mit Hilfe eines zusätzlichen Steuerautomaten die benötigten Filterrealisierungen durchführt.
Die Mehrfachausnutzung von Teilrealisierungen würde allerdings eine drastische Flächenersparnis auf Kosten der Verarbeitungsgeschwindigkeit bringen.
Für bestimmte Applikationen, bei der die Geschwindigkeit nicht im Vordergrund steht, könnte eine entsprechende Struktur durchaus ihren Einsatz finden. Interessant wäre auch eine derartige Realisierung insbesondere für große Busbreiten.

7.4.3 Shannon Hardwareinterpolator

Aufgrund der in der Regel langsameren aufgesetzten Hardware wie ein PC, DSP oder Mikrokontroller, muß der ASIC eine *Sampleratenreduktion* (Downsampling) durchführen. Durch diese Maßnahme wird ein reibungsloses Zusammenarbeiten der Einheiten gewährt und die gewünschte Datenrate zur Verfügung gestellt. Durch die in dem ASIC verwirklichte Multiratenfilterkaskade und der FSM ist eine Sampleratenreduktion und damit ein Downsampling möglich. Als nachteilig erweist sich allerdings, dass das Ausgangssampling synchron zu der jeweiligen Abtastfrequenz des ASICs und nicht frei wählbar ist. Abhilfe könnte ein zusätzlicher *Abtaster* bringen, der beliebige Sampleraten erlaubt [KAM96].
Um eine gewünschte Genauigkeit der ausgegebenen Sampledaten für eine bestimmte Resamplingfrequenz zu gewährleisten, bieten sich nachfolgende, aus der Mathematik bekannte *Interpolationsverfahren* an:

- kubische Interpolation,
- Tailorinterpolation und
- Splineinterpolation.

Die benötigte Fläche für eine parallele oder die benötigte Zeit für eine überwiegend serielle Hardwarerealisierung der genannten Interpolationsarten scheinen allerdings für diese ASIC Realisierung als nicht geeignet. Der Aufwand an Fläche oder benötigter Zeit, unter anderem für eine benötigte Division, machen den Einsatz für diesen ASIC nicht akzeptabel [BRO81].
Um dennoch für ein Resampling den möglichst genauen interpolierten Samplewert schnell und in akzeptabler Genauigkeit zu errechnen, bietet sich eine asynchrone *Shannon Interpolation* an.
Bei dieser Interpolationsart wird durch jeden gegebenen Samplewert eine $sin(\pi x)/(\pi x)$ Funktion (sinc-Funktion) gelegt.
Für eine praktische Realisierung muß ein *sinc Fenster* tabelliert werden. Die aus der Tabelle gelesenen Werte für eine umgebende Abtastung x sind dann mit den x-Werten zu multiplizieren und zu addieren.
Die Operationen können mit Hilfe einer FIR Filterstruktur und dem zusätzlichen *Umtaster* für die gewünschte Samplingrate durchgeführt werden.
Der mögliche Einsatz einer Shannon Interpolation für den realisierten ASIC liegt darin begründet, dass sie ohne eine Division und leicht mit einer in Kapitel 3.2 vorgestellten FIR Hardwarefilterstruktur durchgeführt werden kann. Ein interpolierter Wert kann für jede gewünschte Samplerate ausgegeben werden.
Es ist anzunehmen, dass für höhere Genauigkeiten die tabellierte sinc Funktion zusätzlich entsprechend interpoliert werden muß. Für eine hardwarenahe Implementierung bietet sich die lineare Interpolation an, die schon bei einer DDS Generatorrealisierung aus Abschnitt 3.6 zum Einsatz gekommen ist.
In wieweit die resultierende Genauigkeit durch eine lineare Koeffizienteninterpolation ausreicht, oder in wieweit andere Interpolationsarten eingesetzt werden müssen, ist jeweils für jedes Einsatzgebiet getrennt zu ermitteln. Weitere Untersuchungen im Bezug auf einen Shannon Interpolator wurden nicht durchgeführt.
Die Shannon-Interpolation bedeutet zwar einen zusätzlichen Aufwand, ist allerdings angesichts der Geschwindigkeit und Genauigkeit die geeignetste Interpolationsart für diese ASIC Realisierung. Sie kann allgemein für schnelle Signalverarbeitungsaufgaben eingesetzt werden.

Danksagung

Meinen Dank möchte ich meinem Betreuer Herrn Prof. Klaus von der Heide vom Fachbereich Informatik der Universität Hamburg, Arbeitsbereich Technische Grundlagen der Informatik aussprechen. Er gab mir zahlreiche konstruktive Hinweise durch anregende Diskussionen und Erläuterungen und begleitete das Entstehen der Arbeit über viele Monate als stets ansprechbarer Ratgeber. Ohne seine Anregungen und Unterstützung wäre das Ergebnis in dieser Form nicht möglich gewesen.
Für die Einführung in VHDL und der Hilfestellung bei der Verwendung des Synopsys Entwicklungssystems möchte ich Herrn Andreas Mäder und den übrigen Assistenten des Arbeitsbereichs TECH danken. Besonderen Dank gebührt Siegmund Gorr und Lars Larsson für ihre persönliche Unterstützung in vielerlei Hinsicht.

Literaturverzeichnis

[AMS98] *AMS Hit-Kit V3.12 Documentation*, AMS, 1998

[BAR91] H. Bartsch: *Taschenbuch mathematischer Formeln*, Fachbuchverlag Leibzig, 1991

[BLE96] Bleck, Goedecke, Huss,Waldschmidt: *Praktikum des modernen VLSI-Entwurfes*, Teubner Stuttgart, 1996

[BRO81] I.N. Bronstein, K.A. Semendjajew: *Taschenbuch der Mathematik*, Harry Deutsch Verlag, 1981

[FLI93] N. Fliege: *Multiraten Signalverarbeitung*, Teubner Stuttgart, 1993

[GER96] P. Gerdsen: *Digitale Nachrichtenübertragung*, Teubner Stuttgart, 1996

[GER97] P. Gerdsen P. Kröger: *Digitale Signalverarbeitung in der Nachrichten-übertragung*, Springer Verlag, 1997

[GEI93] R. Geißer, W. Kammerloher, H. Schneider: *Berechnungs und Entwurfs-verfahren der Hochfrequenztechnik*, Vieweg Verlag,1993

[GRÜ93] D. C. von Grüningen: *Digitale Signalverarbeitung*, AT Verlag, 1993

[HAM83] R.W. Hamming: *Digitale Filter*, Prentice Hall Signal Processing Series,1983

[HAR94] Harris Semicondictor: *Digital Signal Processing Handbook*, 1994

[HAR98] Harris Semiconduktor: HSP 50214 Programmable Downkonverter, Produkt Description, 1998

[HES93] W. Hess: *Digitale Filter*, Springer Verlag ,1993

[KAM96] K.D. Kammeyer: *Nachrichtenübertragung*, Teubner Stuttgart, 1996

[KRO97] K. Kroschel/K.D. Kammeyer: *Digitale Signalverarbeitung*, Teubner Stuttgart, 1997

[KNU98] E. Knuth: *The Art of Computer Programming*, Vol. 2, Seminumerical Algorithms, Addison Wesley, 1998

[KUN85] S.Y. Kung, H.J. Whitehouse, T. Kailath: *VLSI and Modern Signal Processing*, Prentice Hall Signal Processing Series, 1985

[LAC95] A. Lacroix: *Filter Digitale*, Teubner Stuttgart, 1995

[LAG87] K. Lagemann: *Rechnerstrukturen*, Springer Verlag, 1987

[LEH94] G. Lehmann, B. Wunder, M. Selz: *Schaltungsdesign mit VHDL*, Franzis Verlag, 1994

[LÜK95] P. Lüke: *Signalübertragung*, Springer Verlag, 1995

[MAE93] A. Mäder: *VHDL Kurzbeschreibung zum Projekt „VLSI-Entwurf*“, Universität Hamburg, Fachbereich Informatik, 1993

[MAT95] Matlab for Windows, *User's Guide*, The Math Works Inc., 1995

[MIL96] O. Mildenberger, *System- und Signaltheorie*, Vieweg Verlag, 1996

[OMO94] A. Omondi Amos R.: *Computer Arithmetik System; Algorithmus, Architekture and Implementations*, Prentice Hall Inernational, 1994

[OSI99] Osicom Technologie: *Direct Digital Frequency Synthesis*, ... a basic tutorial, Osicom Technologie, Santa Monica, California,1999 *http://www.osicom.com/notes/ddstutor.HTM*

[PEZ71] S. D. Pezaris: *A 40 ns 17-bit-by-17-bit Array Multiplier*, IEEE Trans. on Computers, Vol. C-20,No. 4, S. 442-447, 1971

[PIR96] P. Pirsch: *Architekturen der digitalen Signalverarbeitung*, Teubner Stuttgart, 1996

[QUA98] Qualcomm Technologies: *Qualcomm Product Handbook*, 1998

[SPA76] O. Spaniol: *Logik und Entwurf Arithmetik in Rechenanlagen*, Teubner Stuttgart, 1976

[SMI96] Douglas J. Smith: *HDL Chip Design*, Doone Publikations, 1996

[STR93] H. Straub, M. Häßler: Praxis der Digitaltechnik, Franzis Verlag, 1993

[SYN93] *Synopsys VHDL Online Documentation & Reference Manual*, Synopsys Inc., Mauntain View, California, 1993

[TIE93] U. Tietze, C. Schenk: *Halbleiter-Schaltungstechnik*, Springer Verlag, 1993

[VHD94] IEEE Standard VHDL Language Reference Manual – *VHDL Language Reference Manual*, IEEE Standard 1076-1993, IEEE, New York, 1994

[WAL97] S. Wallner: *Untersuchung und Bewertung von Hardwarestrukturen für die VHDL Implementation schneller digitaler Rechenwerke*, 1997

[WIE98] M. Wierich: *Ein digitaler DCF77 Empfänger mit hoher Empfindlichkeit*, Universität Hamburg, Fachbereich Informatik, Diplomarbeit 1998

[ZÖL97] U. Zölzer: *Digitale Audiosignalverarbeitung*, Teubner Stuttgart, 1997

Anhang

Übersicht über die Register des ASICs

Fktnummer	Bezeichnung DataMUX	Funktionsbeschreibung
0	000	Übernahme neuer Sampeldaten
1	001	Übernahme der Daten des 1. Dezimationsfilters (FirOut0)
2	010	Übernahme der Daten des 2. Dezimationsfilters (FirOut1)
3	011	Übernahme der Daten des 3. Dezimationsfilters (FirOut2)
4	100	Übernahme der Daten des 4. Dezimationsfilters (FirOut3)
5	101	--- nicht benutzt
6	110	--- nicht benutzt
7	111	--- nicht benutzt

Belegung der Steuerbits zur Dateneingangsselektion

Fktnummer	Bezeichnung OutMUX	Funktionsbeschreibung
0	00	Dezimationsfilter 1 Ausgabe
1	01	Dezimationsfilter 2 Ausgabe
2	10	Dezimationsfilter 3 Ausgabe
3	11	Dezimationsfilter 4 Ausgabe

Belegung der Steuerbits zur Datenausgangsselektion

Fktnummer	Bezeichnung FirClk	Funktionsbeschreibung
0	0001	Dezimationsteilfilter 1 aktiv
1	0010	Dezimationsteilfilter 2 aktiv
2	0100	Dezimationsteilfilter 3 aktiv
3	1000	Dezimationsteilfilter 4 aktiv

Steuerbits zur Selektion der Dezimationsfilterstrukturen

Bitposition	Bezeichnung	Funktionsbeschreibung
0...31	Phasenschrittregister (PS)	Phasenschrittregister zur Frequenzgenerierung (32 Bit)

Phasenschrittregister zur Frequenzgenerierung

Betriebsmodus	Bezeichnung DDSModus	Funktionsbeschreibung
Reell-analyt Wandlung	00	Sinx(real)=0;Cosy(real)=PS;Sinx(imag)=PS;Cosy(imag)=0
Analytische Verarbeitung	01	Sinx(real)=1;Cosy(real)=1;Sinx(imag)=1;Cosy(imag)=1
Frequenz-Verschiebung	10	Sinx(real)=PS;Cosy(real)=PS;Sinx(imag)=PS;Cosy(imag)=PS
---	11	--- keine Funktion

Steuerworte der drei Betriebsmodi

Quellcodes der VHDL Verhaltensmodelle

Der VHDL Quellcode der realisierten Teilkomponenten und des gesamten ASICs mit den MATLAB .m Files zu Simulation der Multiratenfilterkaskade befinden sich auf der Diskette, die dieser Diplomarbeit beiliegt.

Erklärung

Hiermit erkläre ich an Eides statt, die vorliegende Arbeit selbständig durchgeführt zu haben und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt zu haben.

Pinneberg, den 31.1.2000

_____

Sebastian A. Wallner