Zeitschrift: Bulletin des Schweizerischen Elektrotechnischen Vereins, des Verbandes Schweizerischer Elektrizitätsunternehmen = Bulletin de l'Association suisse des électriciens, de l'Association des entreprises électriques suisses

Herausgeber: Schweizerischer Elektrotechnischer Verein ; Verband Schweizerischer Elektrizitätsunternehmen

Band: 80 (1989)

Heft: 15

Artikel: Asic-Entwicklung, wann und wie?

Autor: Kaeslin, H.

DOI: https://doi.org/10.5169/seals-903697

Asic-Entwicklung, wann und wie?

H. Kaeslin

Bei der Realisierung von Funktionseinheiten aus der digitalen Nachrichtentechnik und Signalverarbeitung stehen mehrere Alternativen zur Auswahl: Asic, Katalog-IC, Mikro- oder Signalprozessor. Am Beispiel eines Faltungsdecoders werden diese verglichen. Asic bieten einzigartige Vorteile, bedingen allerdings gegenüber Prozessoranwendungen ein Mehrfaches an Entwicklungsaufwand. Für den Erfolg komplexer Asic ist eine frühzeitige Zusammenarbeit zwischen IC-Designern und Algorithmenentwicklern unerlässlich.

Plusieurs solutions se prêtent à la réalisation de blocs fonctionnels pour la télécommunication numérique et le traitement de signaux: circuit intégré à la demande (Asic), pièce catalogue, microprocesseur ou processeur de signaux (DSP). En partant d'un décodeur convolutionnel ces alternatives sont discutées et comparées. Les Asic offrent des avantages uniques, l'effort nécessaire à leur mise au point dépasse cependant de beaucoup celui nécessaire à l'application de processeurs. Une collaboration dès le début entre les ingénieurs experts en la conception de circuits intégrés et ceux chargés de développer les algorithmes est indispensable au succès des Asic complexes.

Adresse des Autors

Hubert Kaeslin, Dr. sc. techn., Institut für Integrierte Systeme, ETH Zürich, 8092 Zürich.

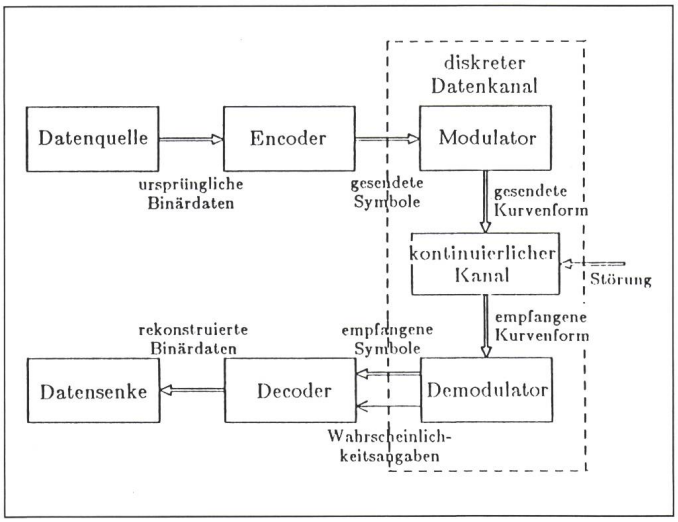
Grundlage dieser Arbeit sind Erkenntnisse, welche bei der Entwicklung eines integrierten Viterbi-Decoders gesammelt worden sind. Die rein technischen Aspekte sind bereits andernorts publiziert worden [1;2]. Der vorliegende Bericht ist ein Versuch, die praktischen und ökonomischen Konsequenzen, welche sich aus der Lösung einer grösseren nachrichtentechnischen Teilaufgabe in Form einer anwendungsspezifischen integrierten Schaltung (Asic) ergeben, mit alternativen Lösungen zu vergleichen und zu diskutieren. Es geht hier also um anwendungsspezifische LSI- oder VLSI-Schaltungen; die Bedeutung kleinerer Asic als platzsparender und wirtschaftlicher Ersatz für eine Anzahl von SSI- und MSI-Bausteinen (Glue Logic) steht *nicht* zur Diskussion. Für jeden, der über etwas Designerfahrung verfügt, sind die hier gezogenen Schlussfolgerungen selbstverständlich. Ausserhalb dieser Kreise scheint jedoch einige Unklarheit in bezug auf die Implikationen von Entwurf und Einsatz anwendungsspezifischer Schaltungen zu bestehen.

Gleich zu Beginn muss klargestellt werden, dass viele der Parameter, auf denen diese Studie beruht, aufgrund des technischen Fortschritts und der Marktentwicklung ständig im Fluss sind.

Was die finanziellen Angaben betrifft, haben wir nicht versucht, die billigsten Angebote einzuholen; wir rechnen mit exemplarischen Beträgen, wie sie uns im Winter 1988/89 von Herstellern und Lieferanten mitgeteilt worden sind. Die zahlenmässigen Werte und die sich daraus ergebenden Schlussfolgerungen werden zwangsläufig bereits nach kurzer Zeit überholt sein, wir hoffen allerdings, dass die Kriterien und die Methodik des Vergleichs trotzdem über einige Jahre hinweg anwendbar bleiben werden. Interessenten wird ausdrücklich nahegelegt, sich die für ihre Projekte massgebenden Daten und Zahlen selber zu beschaffen und nicht einfach die nachstehend abgedruckten Werte umzurechnen.

Problemstellung

Der Viterbi-Algorithmus ist das verbreitetste Decodierverfahren für Faltungscodes. Faltungscodes werden in der digitalen Nachrichtentechnik zur



Figur 1
Allgemeines Modell der digitalen Datenübertragung

Parameter	Symbol	Wert
Codierungsgewinn	G	$\geq$ 5,2dB bei Fehlerrate $P_b=10^{-5}$
Coderate	$R = 1/n$	1/2
Constraint length	k	7
Generatorpolynome	$g_1(x)$	$1+x+x^2+x^3+x^6$
	$g_2(x)$	$1+x^2+x^3+x^5+x^6$
Decodierungsverfahren		Soft Decision Decoding
Kantenmetrik	$D(r_i,h_i)$	Euklidische Distanz
Decodiertiefe	d	$\geq$ 35

Tabelle I
Code- und Decoderspezifikationen

Korrektur von Übertragungsfehlern dort eingesetzt, wo besonders hohe Codierungsgewinne verlangt werden. Anwendung finden sie beispielsweise im Satelliten- und Mobilfunk. Ihr Nachteil liegt im grossen Rechenaufwand bei der Decodierung. Die Figur 1 zeigt die Rolle des Decoders innerhalb des Datenübertragungssystems. Eine ausführliche Behandlung der Grundlagen von Faltungscodes sowie deren Decoder findet sich in [3], ein Tutorial über den Viterbi-Algorithmus in [4] und eine knappe Zusammenfassung im Anhang.

Im folgenden gehen wir davon aus, dass die in Tabelle I zusammengefassten Spezifikationen von allen Lösungsalternativen eingehalten werden müssen. Die Entwicklung, so wie sie im nachfolgenden Text verstanden wird, bezieht sich ausschliesslich auf den Viterbi-Decoder, weitere innerhalb eines Gesamtsystems notwendige Teilfunktionen sind nicht berücksichtigt. Der Entwurf geht von vorhandenen Spezifikationen aus und umfasst sämtliche Schritte bis zum Testen von Prototypen und Debugging von Programmen. Spielraum für grössere Konzeptänderungen und Schaltungskorrekturen (Redesign) ist allerdings nicht eingeschlossen.

Lösungsalternativen

Die Software-Lösung

Faltungscodes werden bei der Firma ABB im Rahmen eines Forschungsprojekts zur Übermittlung digitaler Daten über das elektrische Verteilnetz eingesetzt. Da man bei dieser Anwendung nur bescheidene Datenraten von etwa 300 bit/s anstrebt, kann ein Prozessor neben dem eigentlichen Faltungsdecoder auch noch andere Aufgaben übernehmen. Ein zweites Projekt sieht die Anwendung von Faltungscodes im digitalen Sprechfunk vor, wobei Datenraten von 16 kbit/s geplant sind.

Im Rahmen des ersten Projekts ist der Viterbi-Algorithmus sowohl auf Mikro- (µP) als auch auf Signalprozessoren (DSP) programmiert worden. Die Programmierung des Signalprozessors Motorola 56 001 erfolgte dabei in Assemblersprache, diejenige des Mikroprozessors Motorola 68 020 in Pascal. In beiden Fällen ist der Sourcecode im Hinblick auf eine möglichst hohe Arbeitsgeschwindigkeit gründlich optimiert worden. Während beim Signalprozessor der interne Schreib-Lese-Speicher für die Aufgabe genügt, benötigt der Mikroprozessor ein zusätzliches externes RAM. Der Programmcode wird in beiden Fällen in einem externen ROM gespeichert.

Die Asic-Lösung

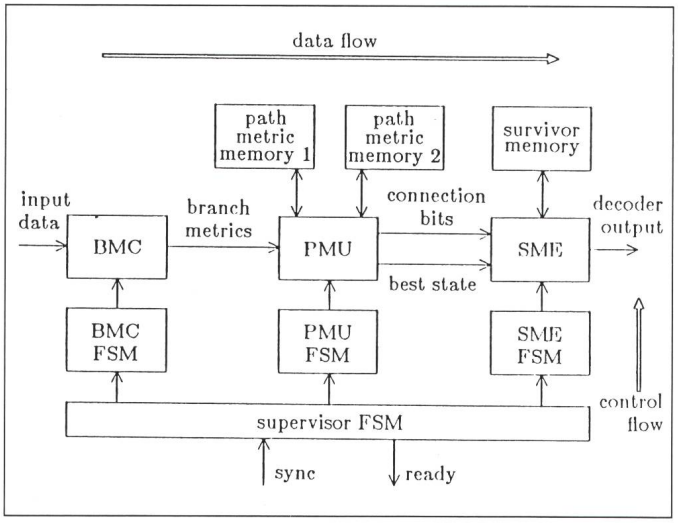
Am Institut für Integrierte Systeme der ETH Zürich ist ein vollständiger Viterbi-Decoder samt dem notwendigen Speicher auf einem Chip entwikkelt worden [1]. Die Schaltung ist in Standard- und Makrozellentechnik für den Prozess CMN20A von VLSI Technology Inc. ausgearbeitet worden, einem 2-µm-Doppelmetall-Poly-Gate-Wannen-CMOS-Prozess. Die Herstellung erfolgte auf einem Multi Project Wafer bei CSEM in Neuchâtel. Als Datenrate waren mindestens 64 kbit/s gefordert worden, an Prototypen wurden bei 20 MHz Taktfrequenz 300 kbit/s gemessen. Für die Leistungsaufnahme ergaben sich die Werte der Tabelle II.

Wie aus Figur 2 hervorgeht, besteht die Grobarchitektur des Viterbi-Decoders aus den aufeinanderfolgenden Blöcken Branch Metric Computation (BMC), Path Metric Update (PMU) und Survivor Memory Evaluation (SME). Um eine maximale Datenrate zu erzielen, kommunizieren diese drei Stufen über Pipeline-Register. Aufgrund der auf die jeweilige Aufgabe spezialisierten Strukturen würde ein Verzicht auf Pipelining die Schaltungskomplexität nur minim reduzieren. Auch innerhalb der einzelnen Blöcke ist kaum Parallelität vorhanden, durch deren Abbau man noch Komplexitätsersparnisse erzielen könnte. Die Chipfläche eines hardwaremässig realisierten Decoders könnte bei Beschränkung auf tiefere Datenraten nur wenig reduziert werden.

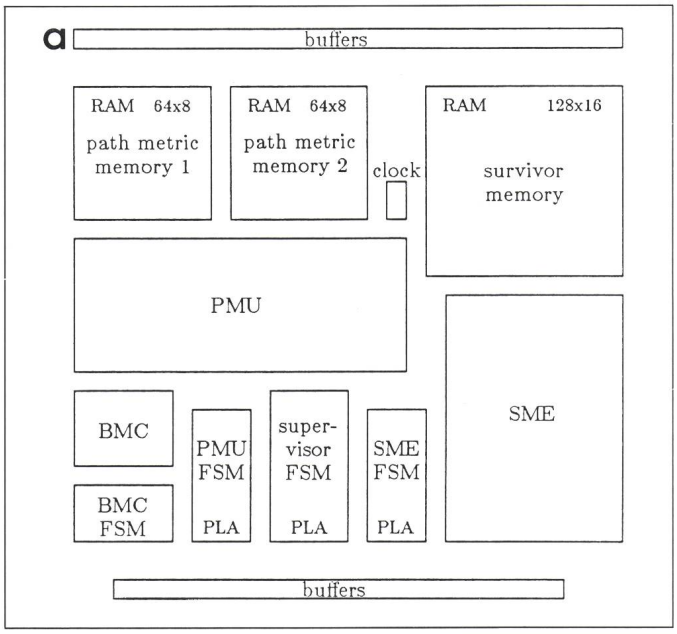
Taktfrequenz MHz	Datenrate kbit/s	Verlustleistung mW
1	15	55
5	75	110
20	300	280

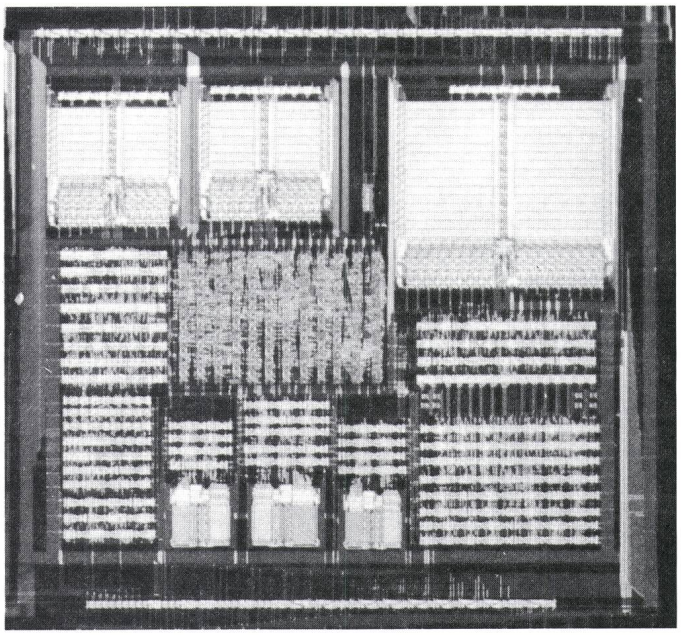
Tabelle II Messresultate am integrierten Viterbi-Decoder

Die Figur 3a zeigt den Floorplan der Schaltung und die Figur 3b den fertigen Chip, dessen Aussenabmessungen 6,4 mm auf 6,4 mm betragen. Die Leerflächen zwischen Schaltungskern und Padring sind durch die Integration auf einem Multi Project Wafer bedingt.



Figur 2
Grobarchitektur des integrierten Viterbi-Decoders





Figur 3 Integrierter Viterbi-Decoder
a Floorplan
b Photo des fertigen Chips

Die Katalog-IC-Lösung

Derzeit bieten mindestens zwei Hersteller auf dem freien Markt komplette integrierte Viterbi-Decoder an. Sorep vertreibt einen Baustein [5], der Coder und Decoder enthält. Der Baustein kann umschaltbar sechzehn verschiedene Faltungscodes mit Coderaten zwischen ¼ und ⅞ verarbeiten; zudem ist seine Decodiertiefe in acht Schritten zwischen 40 und 96 wählbar. Gegenüber dem Decoder beansprucht der Coder nur sehr wenig Ressourcen, da ein Faltungscoder im wesentlichen nur aus einem Schieberegister und einigen XOR-Gattern besteht. Im Unterschied zur Asic-Lösung sind im Datenpfad sieben Pipelinestufen vorhanden. Ursprünglich für 2-µm-Technologie geplant, wird der Baustein heute bei VLSI Technology Inc. in der 1,5-µm-Technologie CMN16 gefertigt, der gleichartigen aber dichteren Nachfolgerin von CMN20A. Dies bringt ihn gegenüber unserem Asic in eine deutlich günstigere Ausgangsposition bezüglich Arbeitsgeschwindigkeit, Leistungsverbrauch und Platzbedarf.

Qualcomm hat einen Chip entwikkelt, welcher allein den in Tabelle I spezifizierten Code verarbeitet. Mit einer erzielbaren Datenrate von 17 Mbit/s ist dieser Baustein allerdings für Anwendungen mit erheblich höheren Datenraten bestimmt und gehört auch preislich einer anderen Klasse an.

Allgemeine Unterschiede in der Entwicklung von Software und Asics

Elementarfunktionen und Ressourcen

Moderne Mikro- und Signalprozessoren weisen recht leistungsfähige Instruktionssätze auf, welche neben logischen auch grundlegende arithmetische Operationen einschliessen (z.B. Addition, Multiplikation). Compiler übersetzen komplexere Operationen in Maschinensprache (z.B. Quadratwurzel, trigonometrische Funktionen). Der *Prozessoranwender* kann also seinen Algorithmus direkt in einer höheren Programmiersprache formulieren.

Der *Schaltungsentwickler* dagegen baut auf einer Zellbibliothek auf, deren Elemente etwa den Funktionen von SSI-Bausteinen der sechziger Jahre entsprechen, d.h. im wesentlichen Gatter und Flip-Flops. In fortschrittlichen CAD-Umgebungen stehen zusätzlich Generatoren für Makrozellen wie RAM, ROM und PLA zur Verfügung. An arithmetischen Funktionsblöcken werden allenfalls Addier- und Multiplizierwerke einigermassen automatisch erzeugt. Zwischen Platz- und Rechenzeitbedarf können unterschiedliche Kompromisse geschlossen werden, Multiplikationen bleiben aber kostspielige Operationen.

Die Platzökonomie zwingt einerseits den IC-Entwickler dazu, grössere Funktionsblöcke sowie On-Chip-Speicher möglichst voll auszulasten, anderseits aber auch den Verbindungsaufwand mininmal zu halten. Wenn der zu implementierende Algorithmus nicht ausgesprochen regulär ist, stellt sich als flächeneffizienteste Lösung oft eine Prozessorarchitektur heraus (Fetch-Load-Execute-Store-Modell).
Da gerade Prozessoren als handoptimierte Universalbausteine in enormen Stückzahlen produziert werden, muss die Eigenentwicklung von Prozessoren sorgfältig daraufhin geprüft werden, ob sie tatsächlich messbare technische und wirtschaftliche Vorteile erbringt, die das damit verbundene Risiko rechtfertigen. Als Bezugspunkt mögen die Angaben dienen, welche Motorola 1985 für die Entwicklung seiner 32-Bit-CPU veröffentlicht hat [6]. Demnach hat die Entwicklung des Mikroprozessors 68 020 (180 000 Transistoren), des Koprozessors für Gleitkommaarithmetik 68 881 (140 000 Transistoren) und der Memory Management Unit 68 851 (190 000 Transistoren) zusammen mehr als 150 Mannjahre verschlungen. Eine ökonomische Alternative zur Eigenentwicklung stellen die von mehreren Firmen zusätzlich zu den Standardzellbibliotheken angebotenen Megazellen dar. Dabei handelt es sich um fertig ausgelegte Zentraleinheiten, die der Entwickler nur noch um die jeweils benötigten Peripherieschaltungen zu ergänzen braucht.

- Prozessorstrukturen gemäss von-

Neumann- oder Harvard-Architektur sind bezüglich ihrer Universalität bei gleichzeitiger konzeptioneller Einfachheit nach wie vor unübertroffen.

- Grössere Speicher vergeuden auf dem Chip zuviel Fläche und sollen daher aus dem Asic ausgelagert werden. Niemand vermag Speicher dermassen dicht und wirtschaftlich herzustellen wie die spezialisierten Produzenten.

Numerische Aspekte

Aufgrund der beschränkten Datenwortbreiten ergeben sich bei der digitalen Verarbeitung von Signalen zwangsläufig Quantisierungsfehler, welche die Genauigkeit des Endresultats beeinträchtigen. Je nach Anwendung wirkt sich dies in einer Verschlechterung des Signal-Rausch-Verhältnisses oder - wie im Falle des Viterbi-Decoders - des Codierungsgewinns oder einer anderen Leistungsgrösse (Figure of Merit) aus. Um diese Einbussen minimal zu halten, muss der Entwickler versuchen, die verfügbaren Wortbreiten mit Hilfe von Skalierungsmassnahmen optimal auszunützen. Bei Mikro- oder Signalprozessoren mit einer Wortbreite von 8, 16, 24 oder mehr Bit gelingt dies bereits nach einer verhältnismässig groben Betrachtung; wo nötig kann in Double-Precision-Arithmetik gerechnet werden. Vollends entschärft wird das Problem beim Einsatz moderner Gleitkommaprozessoren [7;8].

Bei einer speziellen Hardware dagegen kann und muss die minimal vertretbare Wortbreite für jede einzelne Grösse und damit für jedes Register, jeden Bus, jeden Speicher und jedes Rechenwerk separat festgelegt werden, und zwar bei der Entwicklung von ICs in Schritten von einzelnen Bits. Unnötig grosse Wortbreiten verschwenden nicht nur Chipfläche, sondern reduzieren auch die Arbeitsgeschwindigkeit und erhöhen die Verlustleistung. Zudem reduzieren sie auf dem Umweg über eine höhere Schaltungskomplexität die Produktionsausbeute und erhöhen den Testaufwand. Der Hardwareentwickler ist auch nicht an eine vorgegebene Zahlendarstellung und arithmetisch-logische Einheit (ALU) gebunden. Er kann und muss zwischen mehreren Alternativen auswählen, z.B. 1er-, 2er-Komplement, Sign&Magnitude, BCD, Moduloarithmetik [9], verteilte Arithmetik [10], Cordic [11]. Beim Asic-Entwurf stellen Fragen der Zahlendarstellung, der Arithmetik und der Skalierung zentrale Optimierungsprobleme dar, deren Lösung über die technische und wirtschaftliche Tauglichkeit des Chips entscheidet.

Auf eine möglichst geringe Empfindlichkeit gegenüber Quantisierungseffekten muss bereits bei der Entwicklung oder Auswahl des Algorithmus geachtet werden. So weisen zum Beispiel die verschiedenen bekannten Strukturen digitaler Filter ein sehr unterschiedliches Quantisierungsverhalten auf. Bevor mit dem eigentlichen Schaltungsentwurf begonnen werden kann, müssen sämtliche numerischen Einzelheiten mit Hilfe detaillierter Simulationen, welche alle Quantisierungseffekte und Skalierungsmassnahmen berücksichtigen, festgelegt werden. Damit ist ein erheblicher Arbeitsaufwand verbunden. Zudem scheint in dieser Zeit leistungsfähiger Signalprozessoren das Bewusstsein für die Quantisierungsproblematik abhanden gekommen zu sein. Bezeichnend ist die Antwort eines Softwareentwicklers auf die Frage nach den Präzisionsanforderungen in seinem Algorithmus: «Wir haben alles mit 24-Bit-Genauigkeit gerechnet, und das hat sich stets als ausreichend erwiesen.»

- Bei Beginn des eigentlichen Schaltungsentwurfs darf hinsichtlich Algorithmen und numerischer Fragen nicht mehr die geringste Unklarheit bestehen. Integrierte Halbleiterstrukturen bieten im Gegensatz zu PCB und Software keinerlei Möglichkeiten für empirische Produktentwicklung.
- Um zu einer integrationsgerechten Struktur zu gelangen, müssen Algorithmen und Architektur optimal

Anforderungen an einen integrationsgerechten Algorithmus

1. Der Algorithmus ist unempfindlich gegenüber Quantisierungseffekten und kommt mit kleinen Wortbreiten aus.
2. Sein Speicherbedarf ist bescheiden und durch eine feste obere Schranke begrenzt.
3. Der Algorithmus kann in Form eines festen wirbelfreien (d.h. zeitlich vorwärtsgerichteten) Signalflussgraphen formuliert werden.
4. Sein Ablauf ist einfach und hängt nicht von Datenwerten ab. Die Anzahl Durchläufe ist für sämtliche Schleifen a priori bekannt. Die Ablaufsteuerung kann mit einfachen endlichen Automaten realisiert werden.
5. Der Algorithmus ist ausgesprochen regulär und bietet Hand zur Parallelisierung. Die die einzelnen Verarbeitungsschritte implementierenden Blöcke können in einer Ebene angeordnet werden, dazwischen genügen feste lokale Kommunikationsnetzwerke.
6. Multiplikationen und andere höhere arithmetische Operationen (Division, trigonometrische Funktionen, Vektordrehungen, mit Hilfe von Tabellen approximierte Funktionen usw.) werden nicht benötigt.
7. Falls von solchen höheren Operationen doch Gebrauch gemacht werden muss, erfolgt dies in einer ausgesprochen systematischen Art und Weise. Aufwendige, aber nur schwach ausgenützte Funktionsblöcke treten keine auf.
8. An höheren Operationen mit zwei Argumenten sind jeweils eine Variable (Daten) und eine Konstante (Koeffizient) beteiligt, so dass die letztere gegebenenfalls einer Vorverarbeitung unterworfen werden kann.
9. In der vorgesehenen Anwendung ist der Durchsatz wichtiger als die Minimierung der Verzögerung, so dass es zulässig ist, den Algorithmus nach dem Pipelineprinzip zu formulieren.
10. Die kombinatorische Tiefe der einzelnen Abschnitte in der Pipeline ist ausgeglichen und gering, so dass über eine hohe Taktfrequenz eine maximale Schaltungsausnützung ermöglicht wird.
11. Eine Initialisierung abgespeicherter Variablen vor Beginn der Verarbeitung ist unnötig oder dann sehr einfach (Flip-Flops weisen Setz- und Rücksetzeingänge auf, RAM hingegen müssen mit Hilfe zusätzlicher Schaltungen initialisiert werden).
12. Der Algorithmus erlaubt eine Verarbeitung in bit-serieller Form. Beim bit-seriellen Ansatz werden die einzelnen Binärstellen in einem festen Takt und mit den LSB beginnend zeitlich nacheinander berechnet [12]. Anstelle weniger grosser, ganze Wörter aufs Mal verarbeitender Prozesse sind viele kleinere, bitweise und daher langsamer arbeitende Prozesse vorhanden, woraus sich unter dem Strich mindestens eine Einsparung an Verdrahtungsaufwand ergibt.

Tabelle III

aufeinander abgestimmt werden. Eine Zusammenarbeit zwischen IC-Designern und Algorithmenentwicklern ist daher unerlässlich und kann nicht früh genug beginnen.

Physische und technologische Aspekte

Prozessorbausteine werden von ihren Herstellern regelmässig an neue Fabrikationsprozesse angepasst und werden dadurch kleiner, schneller und billiger. Ohne eigenes Dazutun profitieren deren Anwender von der technologischen Entwicklung. Will der Asic-Anwender nach einiger Zeit auf eine fortschrittlichere Technologie umstellen, wird auf jeden Fall eine Überarbeitung (Redesign) des Schaltungsentwurfs notwendig. Der Aufwand dazu hängt allerdings stark von der gewählten Entwurfstechnik (Gate Array, Standard- und Makrozellen oder Handlayout) ab.

Mit den folgenden Stichworten seien die diversen Realisierungsfragen zusammengefasst, welche der Entwickler digitaler ICs neben allen bisher behandelten Problemen auch noch bearbeiten muss: Auswahl von Zieltechnologie und Entwurfstechnik, Interfacing, Logikminimierung, Testbarkeit, Initialisierung, Simulationsmodelle, Tor- und Signallaufzeiten, Timing, Clockerzeugung und -verteilung, Floorplan, Layout, Anzahl und Belegung der Pins, Gehäuse, elektrische Dimensionierung, Speisung, Verlustleistung, Wärmeabfuhr, parasitäre Effekte (Kapazitäten, Widerstände, Induktivitäten, Noise, Latch-up), Schutzmassnahmen gegen elektrostatische Entladungen, Designverifikation.

- Entwicklungstiefe und -aufwand sind beim IC-Entwurf um ein Vielfaches grösser als bei der Entwicklung von Software für Prozessorbausteine.
  Da die Gründe dafür bei weitem nicht nur beim physischen Entwurf liegen, wird sich dies in absehbarer Zukunft auch nicht entscheidend ändern.

Integrationsgerechter Algorithmus, was heisst das?

In Tabelle III ist eine Anzahl allgemeiner Eigenschaften aufgeführt, welche ein integrationsgerechter Algorithmus idealerweise aufweisen sollte. In der Praxis wird es kaum Algorithmen geben, die sämtliche Anforderungen vollständig erfüllen. Dennoch darf man sagen, dass es um so schwieriger wird, einen Algorithmus effizient in Silizium umzusetzen, je mehr dieser Kriterien verletzt sind. Die Gewichtung der einzelnen Punkte hängt stark davon ab, ob primär minimale Fläche oder maximale Verarbeitungsleistung angestrebt wird. Es liegt auf der Hand, dass Parallelisierung und intensive Anwendung von Pipelinestrukturen neben dem Durchsatz auch den Flächenbedarf vervielfachen.

Der Viterbi-Algorithmus erfüllt, so wie er in unserem Asic implementiert worden ist, die Forderungen 1, 2, 4, 6, 9 und 11; Punkt 7 und 8 sind gegenstandslos.

Lösungsvariante	Hardware			Software	
	ETHZ	Sorep	Qualcomm	auf μP	auf DSP
zentrales Bauteil	BiKaTo	SOR5033	Q1401	68020	56001
Kategorie	ASIC	Katalog-IC		Prozessor	
Encoder inbegriffen	nein	ja	ja	ohne weiteres	ohne weiteres
Anzahl Codes	1	16	1	beliebig	beliebig
max. Datenrate	300kbit/s	500kbit/s	17Mbit/s	1.5kbit/s	7kbit/s
bei Taktfrequenz	20MHz	39MHz	17MHz	20MHz	20MHz
Technologie	CMOS	CMOS	CMOS	(CMOS)	(CMOS)
min. Strukturbreite	2μm	1.5μm	1.5μm	(2->1.5μm)	(1.5μm)
Entwurfstechnik	SC&MC	SC&MC	?	(HL)	-
Komplexität					
Logik	3628GE	8200GE	?	-	-
RAM Speicher	2688Bit	9216Bit	?	-	-
Anzahl Transistoren	≈35 000	≈100 000	?	(180 000)	-
Chipgrösse netto	35mm^2	67mm^2	168mm^2	-	-
Programmiersprache	-	-	-	Pascal	Assembler
Umfang Sourcecode	-	-	-	1600 Zeilen	1500 Zeilen
Anzahl Chips	1	1	1	3	2
Gehäuse	wählbar	LCC	PGA	PGA/DIL	PGA/DIL
Anzahl Pins	16	68	155	170	140
Verlustleistung	100mW	500mW	1.25W	1...2W	≤1W
bei Datenrate	64kbit/s	500kbit/s	12Mbit/s	unabhängig	unabhängig
Skalierungsfragen	kritisch	-	-	beachtenswert	beachtenswert
Entwicklung					
Algorithmen/Hardware	sequentiell	-	-	überlappend	überlappend
Gesamtaufwand	18Mm	0	0	1Mm	3Mm
Durchlaufzeit	1...3m	-	-	<1d	<1d
Entwicklungsrisiko	gross	keines	keines	gering	gering
Flexibilität	teuer	beschränkt	keine	sehr gross	gross
Zusatzfunktionen	möglich	nein	nein	ohne weiteres	ohne weiteres
Kopierschutz	gut	keiner	keiner	schlecht	schlecht
feste Kosten	420 000Fr	0	0	20 000Fr	60 000Fr
bewegliche Kosten	8Fr	450Fr	1500Fr	200Fr	250Fr
Kosten pro Stück	10Fr	450Fr	1500Fr	200Fr	250Fr

Tabelle IVa Gegenüberstellung der Lösungsvarianten (s. Bemerkungen Tab. IVb)

Gegenüberstellung und Diskussion der Lösungsvarianten

Die Tabelle IV gibt einen Vergleich der verschiedenen Lösungsvarianten. Die zugehörigen Erläuterungen und Anmerkungen sind in Tabelle IVb zu finden.

Leistungsfähigkeit

Auf den ersten Blick enttäuscht der Signalprozessor hinsichtlich der erzielten Datenrate, arbeitet er doch lediglich fünfmal schneller als der Mikroprozessor und über 40mal langsamer als der Asic. Dem ist entgegenzuhalten, dass ein DSP in der untersuchten Anwendung seine Trümpfe gar nicht ausspielen kann. Hauptmerkmale von Signalprozessoren sind separate ALU für Adressen und Daten, ein breiter Datenpfad sowie ein schnelles Multiplizierwerk. Der erwähnte Motorola 56 000 benötigt für eine 24×24-Bit-Multiplikation samt anschliessender 56-Bit-Akkumulation knapp 100 ns [13]. Damit unterbietet er bereits ge-

Erläuterungen zu einzelnen Vergleichskriterien der Tabelle IVa

Maximale Datenrate: Bei den zwei Prozessorvarianten wurden die Zahlen hochgerechnet für den Fall, dass die volle Rechenleistung für den Viterbi-Algorithmus zur Verfügung steht.

Entwurfstechnik: SC = Standardzellen, MC = Makrozellen, HL = Handlayout.

Komplexität und Anzahl Transistoren: Beide verstehen sich ohne Padframe.
GE = Gatteräquivalent; ein GE eintspricht einem NAND-Gatter mit zwei Eingängen.

Umfang Sourcecode: Der Algorithmus kann wesentlich kompakter formuliert werden, jedoch sind bestimmte Teilabläufe zwecks Einsparung von Ausführungszeit ausprogrammiert worden, was dem Code einen ausgesprochen repetitiven Charakter verleiht [14].

Anzahl Pins: 16 Pins sind beim Asic von der Funktion her ausreichend und schliessen eine globale Scan Path-Testeinrichtung ein; die fabrizierten Muster nützen die ohnehin verfügbaren 64 Pins für Prüfzwecke voll aus.

Durchlaufzeit: Der Zeitbedarf von der Beendigung einer Modifikation oder Korrektur am Entwurf bis zur Verfügbarkeit eines Testmusters. Bei Asic ist dies die Herstellungszeit plus eine allfällige Wartezeit, bis die Produktionslinie verfügbar ist, bei der Softwarelösung die Zeit für Compilation und Neuprogrammierung der EPROM. Aufgrund der langen Durchlaufzeiten besteht beim IC-Entwurf eine starke Verpflichtung zum First-Time-Right-Design.

Entwicklungsrisiko: Die Gefahr, die verlangten technischen Spezifikationen nicht im Rahmen der zeitlichen und finanziellen Vorgaben erreichen zu können.

Flexibilität: Die Möglichkeit, Änderungen vorzunehmen oder unterschiedliche Versionen - etwa aufgrund abweichender Anforderungen und Normen - nebeneinanderher zu produzieren. Im Falle des Viterbi-Decoders könnten dies beispielsweise andere Codes sein.

Zusatzfunktionen: Die Möglichkeiten, weitere innerhalb des Gesamtsystems erforderliche Aufgaben miteinzubeziehen. Im Falle einer Fehlerkorrekturschaltung wären dies zum Beispiel die Vorverarbeitung der Eingangssignale oder die Anzeige von Decodier- oder Synchronisationsfehlern.

Kopierschutz: Die Sicherheit gegenüber direkten Nachahmungen durch Konkurrenten.

Feste Kosten: Annahmen: Arbeitsplatzkosten inkl. Entwicklungs- resp. CAD-System 20 000 Fr./Mm, bei Asic zusätzlich Maskenkosten und Prototypfabrikation (NRE Costs) 60 000 Fr. (Mm = Mannmonat).

Bewegliche Kosten: Material- und soweit notwendig Boardkosten für Teilsystem Viterbi-Decoder. Beruhend auf Tausenderpreisen für Katalog- und Prozessorbausteine, auf Abnahme von mindestens 50 000 Stück (geprüft) jährlich für Asic. Überschlagsmässige und unverbindliche Angaben!

Kosten pro Stück: Annahme: Abwälzung der Festkosten auf 250 000 Stück in 5 Jahren.

Tabelle IVb

ringfügig die Verarbeitungszeit, welche in der 2-µm-Technologie CMN20A ein mit dem Zellcompiler erzeugtes paralleles Multiplizierwerk gleicher Wortlänge allein benötigt (in 1,5-µm-CMN16 etwa 81 ns). Nun verlangt der Viterbi-Algorithmus aber weder Multiplikationen noch grössere Wortbreiten bei den übrigen arithmetisch-logischen Operationen.

Allgemein ist die Architektur von Signalprozessoren auf eine schnelle Berechnung von Skalarprodukten ausgerichtet. Ein analoger Vergleich anhand einer entsprechenden Signalverarbeitungsaufgabe (z.B. Filter in direkter Form, Korrelator usw.) fiele für einen DSP zweifellos wesentlich vorteilhafter aus, zumal grosse parallele Multiplizierwerke in Asics sehr viel Fläche beanspruchen, z.B. etwa 20 mm² für eine 24×24-Bit-Multiplikation in CMN20A (etwa 12,5 mm² in CMN16). Umgekehrt büssen Signalprozessoren einiges von ihrer Effizienz ein, wenn die Verarbeitungsalgorithmen nicht auf Skalarprodukte zurückgeführt werden können. Eine Asic-Lösung gewinnt an Attraktivität, falls sich solche Algorithmen integrationsgerecht formulieren lassen.

- Welche Lösung technisch sinnvoll ist, hängt stark von der Art des Problems ab. Tendenziell eignet sich die Integration eher für reguläre, datenunabhängige und bitorientierte Algorithmen und der Einsatz von Signalprozessoren eher für vom Ablauf her irreguläre, arithmetisch anspruchsvolle, speicherintensive, auf grosse Wortbreiten angewiesene oder auf Skalarprodukte zurückführbare Berechnungsaufgaben.

Kosten

Wie nicht anders zu erwarten, ist die IC-Entwicklung erst dann wirtschaftlich, wenn die hohen Entwicklungskosten auf grosse Stückzahlen verteilt werden können. Falls integrationsgerechte Lösungen gefunden werden können, ermöglichen aber gerade Asic die kostengünstigste Herstellung grosser Stückzahlen. Der Vergleich in Tabelle IV wird verzerrt durch den Umstand, dass bei Katalog- und Prozessorbausteinen Preise für Stückzahlen über Tausend nicht erhältlich waren. Entsprechende Korrekturen vermöchten das Gesamtbild aber nicht grundsätzlich umzukrempeln; zudem sind die angenommenen Stückzahlen typisch für die jeweilige Lösungsalternative. Hingegen sollte man nicht vergesse, dass, sofern es die geforderte Datenrate erlaubt, mit denselben Prozessorbausteinen auch andere Teilaufgaben bearbeitet werden könnten und zwar praktisch ohne Mehrkosten.

- Wirtschaftlich gesehen bedingen sich LSI/VLSI und Massenproduktion gegenseitig.

Gate Array als Alternative

Eine Alternative bei der Entwicklung von Asic bilden die Gate Arrays, bei denen nur gerade die Metallisierungsmaske(n) kundenspezifisch sind. Ihr Hauptvorteil liegt in den wesentlich kürzeren Entwicklungs- und Durchlaufzeiten im Vergleich zur Standardzellentechnik oder zum Full-Custom-Design. Ein weiterer Vorteil liegt in der weniger engen Bindung an eine bestimmte Zieltechnologie. In der konventionellen Gate-Array-Technik, bei der zwischen vorfabrizierten Transistorgruppen Verdrahtungskanäle fester Breite ausgespart sind, ist die Realisierung grösserer Speicher auf demselben Siliziumplättchen allerdings nicht flächeneffizient. Der beschriebene Viterbi-Decoder mit seinen 3-kbit-RAM stellt heute wohl gerade einen Grenzfall dar. Die zurzeit in Einführung befindliche Technik der Sea of Gates, bei der Transistoren flächendeckend vorhanden sind, ermöglicht den Bau kompakterer und damit wirtschaftlicherer und schnellerer On-Chip-Speicher. Sie dürfte daher Gate Arrays in Zukunft erheblich attraktiver machen. Für Anwendungen mit kleineren Stückzahlen und geringeren Komplexitäts- und Geschwindigkeits-

anforderungen stellen programmierbare Gate Arrays [14] wegen ihrer hohen Flexibilität eine weitere interessante Alternative dar.

Schlussfolgerungen

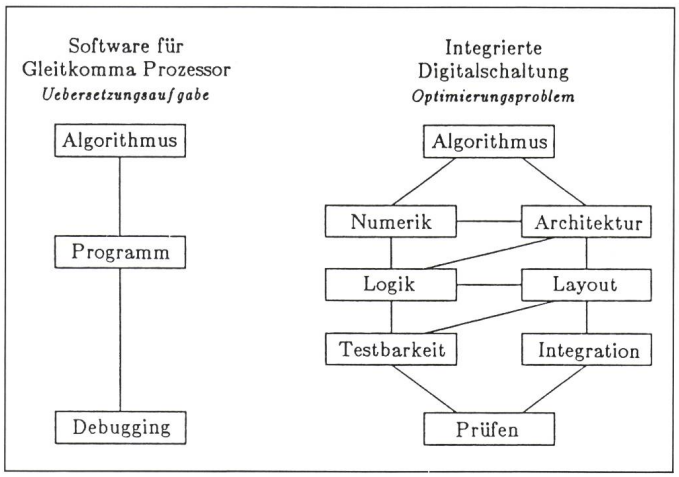
Die Entwicklung von integrierten Schaltungen für Zwecke der digitalen Nachrichtentechnik und Signalverarbeitung benötigt im Vergleich zu den diskutierten Alternativen erheblich mehr Know-how, Entwicklungsaufwand und Zeit. Sicher werden zukünftige CAD-Werkzeuge die Produktivität des Asic-Designers noch erheblich steigern. Die prinzipiell grössere Entwicklungstiefe sowie der unumgängliche Aufwand und Zeitbedarf für die Herstellung der Schaltungen werden dieses Handicap aber nie verschwinden lassen, selbst wenn die Utopie des Silicon Compilers Wirklichkeit werden sollte.

- Vorteilhaft ist die Entwicklung komplexer Asic derzeit dort, wo Massenproduktion gegeben ist und/oder geringer Leistungsverbrauch, kleinster Raumbedarf und hohe Nachahmungssicherheit entscheidend sind. Hohe Geschwindigkeitsanforderungen, Einbezug von Sensoren oder von analogen Komponenten sind weitere Gründe, welche Asic befürworten.

Zudem darf eine höhere Systemzuverlässigkeit erwartet werden, wenn mit Hilfe von Asic die Anzahl der Chips und externen Verbindungsleitungen reduziert werden können.

In der industriellen Praxis kommt es oft vor, dass die Verarbeitungsalgorithmen noch nicht bis in alle Einzelheiten festgelegt oder mengenmässig genügende Absatzmöglichkeiten noch ungewiss sind und dass das Produkt dennoch in möglichst kurzer Zeit auf den Markt gebracht werden muss. In solchen Situationen wird es sinnvoll sein, eine erste softwaremässige Version mit einem Mikro- oder Signalprozessor zu realisieren, mit der bei kleinerem Risiko die Marktchancen getestet werden können und mit der auf Veränderungswünsche kurzfristig reagiert werden kann. Aufgrund der definitiven Spezifikationen und der tatsächlichen Absatzmöglichkeiten kann etwas später fundiert über die Entwicklung kundenspezifischer Schaltungen entschieden werden. Als zweite Version auf den Markt gebracht, können sie bei gleicher Funktionalität den Leistungsverbrauch und die Herstellungskosten senken. Dieses Vorgehen, welches dem Rapid Prototyping der Softwareentwicklung entspricht, ist unter anderem bei frühen Compact-Disc-Plattenspielern und bei 32-kbit/s-ADPCM-Modems von japanischen Firmen angewendet worden [15].

- Die Entwicklung von VLSI-Schaltungen ist keine blosse Übersetzungsaufgabe, sondern ein vielschichtiges Optimierungsproblem (Fig. 4). Unbestimmte oder schwankende Spezifikationen bei festen Terminen kann es daher nicht geben.



Figur 4
Entwicklungsschritte und ihre Wechselwirkungen bei Soft- und Hardwareentwurf

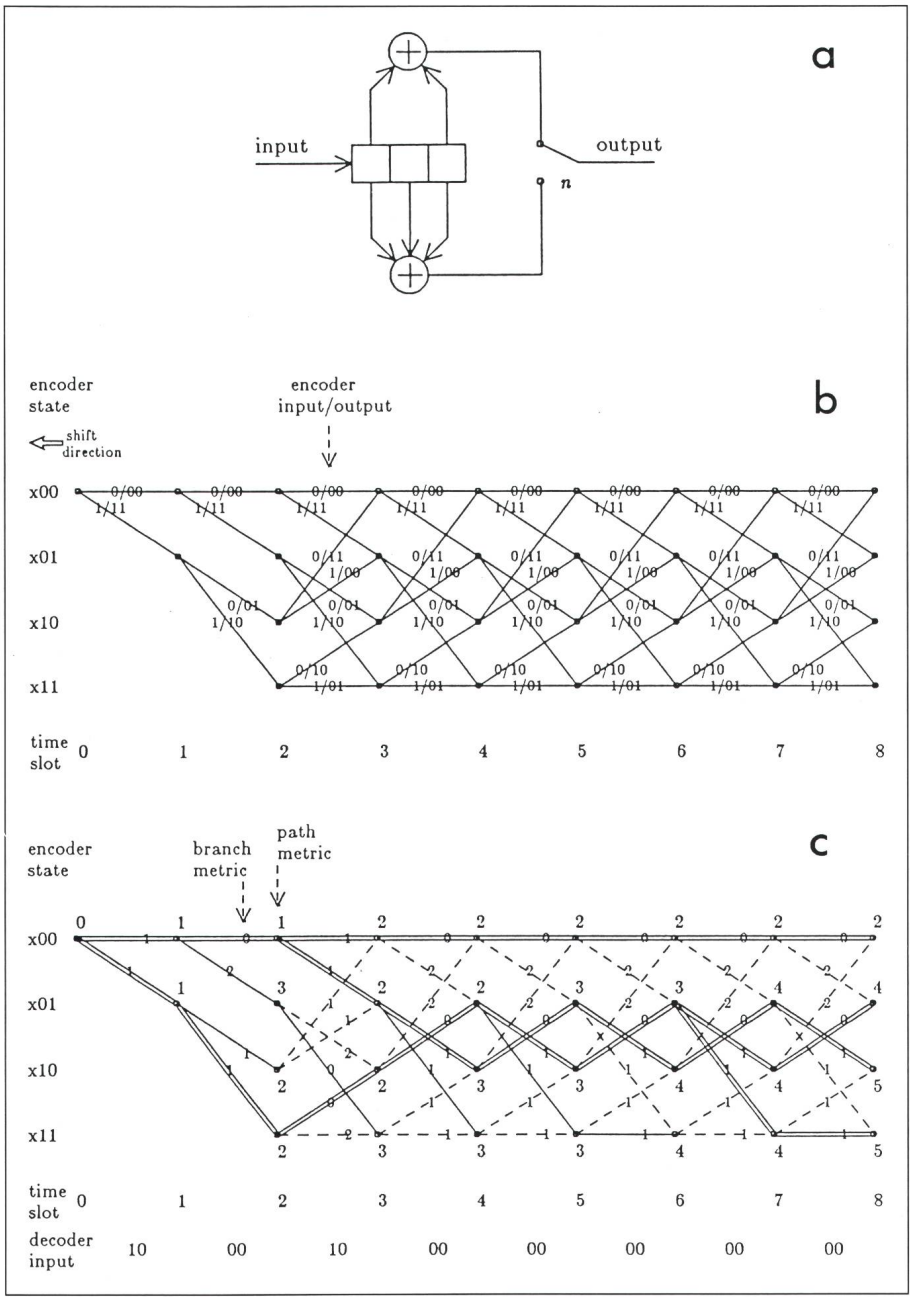
Anhang: Faltungscodes und ihre Decodierung mit dem Viterbi-Algorithmus

Zur Korrektur von Übertragungsfehlern wird in der digitalen Nachrichtentechnik die sogenannte Kanalcodierung eingesetzt. Bei Benützung geeigneter Codes kann das zur Erzielung einer geforderten Bitfehlerwahrscheinlichkeit notwendige Signal-zu-Rauschverhältnis (SNR) erheblich reduziert werden. Die mit einem bestimmten Verfahren erzielte Verbesserung wird dabei als *Codierungsgewinn* bezeichnet. Fehlerkorrigierende Codes erfordern die Einführung zusätzlicher Information vor der Übertragung. Bei Faltungscodes geschieht dies - im Gegensatz zu Blockcodes - kontinuierlich.

Der Encoder enthält ein Schieberegister, welches den Strom binärer Daten aufnimmt. Die Länge k dieses Registers wird als *Constraint Length* bezeichnet. Bestimmte Anzapfungen werden einer Anzahl von Modulo-2-Addierern zugeführt, welche die zu übertragenden Symbole berechnen (Fig. 5a). Die gewählten Anzapfungen bestimmen den Code und werden in Form von *Generatorpolynomen* spezifiziert. Da ein derartiger Encoder ein Eingangssymbol zu n Ausgangssymbolen verarbeitet, spricht man von einer *Coderate* von $R = 1/n$. Als *Zustand* oder *State* des Coders wird ein Vektor, gebildet aus den Binärwerten im Schieberegister ausser der letzten Stelle, bezeichnet. Das Verhalten eines Faltungscoders kann in Form eines gitterartigen Graphen, des sog. *Trellis*, dargestellt werden. In diesem gerichteten und wirbelfreien Graphen steht jeder Knoten für einen Zustand, so dass der Graph pro Zeitschlitz um 2^{k-1} Knoten wächst. Jeder Kante sind ein Eingangssymbol und n Ausgangssymbole zugeordnet (Fig.5b).

Der Decoder hat die Aufgabe, aus den empfangenen Symbolen denjenigen Datenstrom zu rekonstruieren, der mit der grössten Wahrscheinlichkeit in den Encoder eingespeist worden ist. Aufgrund von Übertragungsfehlern wird die empfangene Symbolfolge im allgemeinen nicht mit der ursprünglich gesendeten übereinstimmen. Die Decodierung kann als Problem des kürzesten Pfades durch einen zweiten Trellisgraphen betrachtet werden (Fig. 5c). Die Länge eines Pfades wird *Pfadmetrik* genannt und entspricht der Summe der Gewichte aller beschrittenen Kanten.

Statt die empfangenen Symbole unmittelbar binär als 0 oder 1 zu klassifizieren, kann der Empfänger auch so konstruiert werden, dass er dem Decoder Q-wertige Symbole liefert. Die Kantengewichte werden dabei entsprechend einer Kantenmetrik $D(r_i, h_i)$ bestimmt, die der Abweichung zwischen den tatsächlich empfangenen Symbolen r_i und den den einzelnen Kanten zugeordneten nominellen Symbolen h_i Rechnung trägt. Dieses Soft Decision Decoding genannte Ver-



Figur 5 Beispiel eines einfachen Faltungscodes mit $R = ½$, $k = 3$, und den Generatorpolynomen $1+x^2$ und $1+x+x^2$

a Prinzipschaltbild des Encoders
b Trellisgraph für den Encoder
c Trellisgraph des Decoders beim Empfang der Folge '10 00 10 00 00 00 00 00'

----- dead branches
——— past survivors
▭ actual survivors

fahren trägt zu einer weiteren Verbesserung des Codierungsgewinns bei.

Der Viterbi-Algorithmus arbeitet zyklisch. Pro Zeitschlitz nimmt er n Empfangssymbole auf und liefert ein decodiertes Datenbit. Aus naheliegenden Gründen kann der Decoder den Trellisgraphen nicht bis zu einer beliebigen Tiefe speichern, sondern muss sich mit einer endlichen *Decodiertiefe* d begnügen. In jedem Zeitschlitz werden drei grössere Teilaufgaben gelöst.

1. *Branch Metric Computation (BMC):* Weise allen Kanten entsprechend der gewählten Kantenmetrik $D(r_i, h_i)$ und aufgrund der empfangenen Symbole ein Gewicht zu.

2. *Path Metric Update (PMU):* Bestimme unter allen Kanten, welche in einen Knoten münden, diejenige mit den geringsten akkumulierten Kosten. Speichere diese Kante als jüngstes Glied eines überlebenden Pfades, eines sog. *Survivor Path,* ab und schliesse die übrigen Kanten von der Weiterverarbeitung aus. Aktualisiere die den einzelnen Zuständen zugeordneten Pfadmetriken entsprechend.

3. *Survivor memory Evaluation (SME):* Vergleiche die Metriken aller überlebenden Pfade und finde denjenigen mit dem geringsten Wert. Verfolge diesen Pfad im Trellisgraphen über die volle Decodiertiefe zurück, und gib als Resultat dasjenige Datenbit aus, welches als Encoder-Eingangswert zu dessen ältester Kante gehört.

Dankeswort

Den Herren Dr. *W. Braun,* Dr. *D. Dzung,* Dr. *W. Hagmann* und *S. Ramseier,* ABB Dättwil, möchten wir für die gewährte Unterstützung sowie Dr. *F. Bonzanigo,* ETH Zürich, für anregende Diskussionen herzlich danken.

Literatur

[1] *M. Biver, H. Kaeslin* and C. Tommasini: Architectural design and realization of a single-chip Viterbi decoder. Integration 8(1989)1.

[2] *M. Biver, H. Kaeslin* and *C. Tommasini:* In-place updating of path metrics in Viterbi decoders. IEEE Journal of Solid-State Circuits 24(1989)4.

[3] *G. C. Clark* and *J.B. Cain:* Error-correction coding for digital communications. New York/London, Plenum Press, 1981.

[4] *G. D. Forney:* The Viterbi algorithm. IEEE Proc. 61(1973)3, p. 268...278.

[5] *B. Badefort. M. Cartier* et *A. Louineau:* Des composants pour la correction des erreurs en transmission de données: un décodeur de Viterbi et un codec de Reed-Solomon. Proc. Euro ASIC Grenoble, 25...27 janvier 1989; p. 269...270.

[6] *B. Beims:* The MC68020 32-bit MPU: Opening new application doors. Wescon Conference Record 1985, paper 1/4, p. 1....17.

[7] Special issue on digital signal processing. IEEE Micro 8(1988)6.

[8] *E. A. Lee:* Programmable DSP architectures: Part I. IEEE ASSP Magazine 5(1988)4, p. 4...19.

[9] *M. A. Soderstrand* a.o.: Residue number system arithmetic: Modern applications in digital signal processing. New York, IEEE Press, 1986.

[10] *S. Zohar:* A VLSI implementation of a correlator/digital-filter based on distributed arithmetic. IEEE Trans. ASSP 37(1989)1, p. 156...160.

[11] *H. M. Ahmed:* Alternative arithmetic unit architectures for VLSI digital signal processing. In: S.Y. Kung, H. J. Whithehouse and T. Kailath: VLSI and modern signal processing. Englewood Cliffs/N. J., Prentice-Hall, 1985; p. 277...303.

[12] *P. Denyer* and *D. Renshaw:* VLSI signal processing: A bit-serial approach. Reading/Mass. a. o., Addison-Wesley, 1985.

[13] *K. L. Kloker:* The Motorola DSP56000 digital signal processor. IEEE Micro 6(1986)6, p. 29...48.

[14] *R. Freeman:* User-programmable gate arroys. IEEE Spectrum 25(1988)13, S. 32...35.

[15] *R. Maruta* and *T. Nishitani:* Signal processing VLSI developments: Prospects through experience. S.Y. Kung, R.E. Owen and J. G. Nash: VLSI signal processing II. New York, IEEE Press, 1986; p. 223...231.