FORTSCHRITT-BERICHTE VDI

Dipl.-Inform. Ulrich Rückert, Dortmund

Integrationsgerechte Umsetzung von assoziativen Netzwerken mit verteilter Speicherung

Reihe **10**: Informatik/
Kommunikationstechnik Nr. **130**

VDI VERLAG
Verlag des Vereins Deutscher Ingenieure · Düsseldorf

Rückert, Ulrich

Integrationsgerechte Umsetzung von assoziativen Netzwerken mit verteilter Speicherung

Fortschr.-Ber. VDI Reihe 10 Nr. 130. Düsseldorf: VDI-Verlag 1990.
154 Seiten, 65 Bilder, 15 Tabellen.

Für die Dokumentation: Assoziative Netzwerke — Neuronale Netze — Assoziativspeicher — Integrationsgerechter Entwurf — Fehlertoleranz — Verteiltes Speicherprinzip — CMOS-Technologie

Assoziative Netzwerke mit verteilter Speicherung, deren grundlegende Aufgabe die Musterabbildung bzw. -vervollständigung ist, bilden eine Untermenge der derzeit viel diskutierten Neuronalen Netze. Für diese Untermenge wird eine einheitliche Beschreibungsform angegeben sowie die für die Anwendung wichtigen Eigenschaften der Speichereffektivität und Fehlertoleranz quantitativ erfaßt. Verschiedene integrationsgerechte Realisierungsalternativen ausgewählter Modelle werden diskutiert und mit Hilfe von integrierten Testschaltungen überprüft. Die erarbeiteten theoretischen und praktischen Ergebnisse bilden die Grundlage für zwei neue VLSI-Systemarchitekturen, die anhand von konkreten Bausteinrealisierungen in CMOS-Technologie analysiert werden. Beide Konzepte zeichnen sich durch einen modularen und regulären Systemaufbau, eine effiziente Informationsspeicherung und einen sehr schnellen assoziativen Datenzugriff aus.

Die Reihen der FORTSCHRITT-BERICHTE VDI:

1 Konstruktionstechnik/Maschinenelemente
2 Fertigungstechnik
3 Verfahrenstechnik
4 Bauingenieurwesen
5 Grund- und Werkstoffe
6 Energieerzeugung
7 Strömungstechnik
8 Meß-, Steuerungs- und Regelungstechnik
9 Elektronik
10 Informatik/Kommunikationstechnik
11 Schwingungstechnik
12 Verkehrstechnik/Fahrzeugtechnik
13 Fördertechnik
14 Landtechnik/Lebensmitteltechnik
15 Umwelttechnik
16 Technik und Wirtschaft
17 Biotechnik
18 Mechanik/Bruchmechanik
19 Wärmetechnik/Kältetechnik
20 Rechnerunterstützte Verfahren (CAD, CAM, CAE, CAP, CAQ, CIM,...)
21 Elektrotechnik

ISSN 0178-9627

ISBN 3-18-143010-2

Danksagungen

Die vorliegende Arbeit entstand während meiner Tätigkeit als wissenschaftlicher Mitarbeiter am Lehrstuhl für Bauelemente der Elektrotechnik der Universität Dortmund.

Herr Prof. Dr.-Ing. Karl Goser, dem ich die Anregung zu diesem Thema verdanke, hat die Arbeit durch sein stetes Interesse und mit vielen wertvollen Diskussionen begleitet. Er hat stets für sehr gute Arbeitsbedingungen gesorgt und damit die Voraussetzungen geschaffen, unter denen die Arbeit gedeihen konnte. Dafür sei ihm an dieser Stelle herzlich gedankt.

Herrn Prof. Dr.-Ing. Heiner Klar danke ich für die anregenden und fördernden Hinweise sowie für die bereitwillige Übernahme des Koreferates.

Großen Anteil an der Durchführung dieser Arbeit haben Dr.-Ing. Hilleringmann und die Mitarbeiter der Technologielinie, die bei technologischen Problemen immer mit Rat und Tat zur Seite standen und die benötigten Testschaltungen stets mit Sorgfalt gefertigt haben.

Von meinen Kollegen erhielt ich in zahlreichen Gesprächen wertvolle Anregungen und Hinweise. Allen, die durch Diskussion und Kritik zum Gelingen dieser Arbeit beigetragen haben, möchte ich recht herzlich danken. Besonders möchte ich mich diesbezüglich bei Herrn Dipl.-Ing. Heite, Herrn Dipl. Phys. Kreuzer und Herrn Dipl.-Ing. Soennecken bedanken.

Für die ständige Hilfsbereitschaft und die Unterstützung während meiner Tätigkeit am Lehrstuhl Bauelemente bedanke ich mich bei Frau Menke und Herrn Prof. Dr.-Ing. Schumacher. Frau Schmidtbauer und Frau Lunte möchte ich für die schnelle Anfertigung von verschiedenen Fotos meinen Dank aussprechen.

Weiterhin möchte ich mich bei den Studien- und Diplomarbeitern Herrn Surmann, Herrn Stiebler, Herrn Groß und Herrn Kleerbaum für zahlreiche Simulationen, Messungen und Diskussionen bedanken.

Ein ganz besonderer Dank gilt meiner Frau Hannelore für ihre hilfreiche Unterstützung bei der Erstellung des Manuskriptes und ihre fürsorgliche Rücksichtnahme während der Entstehung dieser Arbeit.

Inhaltsverzeichnis

1. Einleitung

Mit der Entwicklung integrierter Schaltungen zu immer höheren Integrationsgraden - derzeit stehen wir vor der Einführung der Mega-Logik und des 64-Megabit-Speicherbausteins - erhebt sich die Frage nach neuen, innovativen Schaltungs- und Systemkonzepten in der Großintegrationstechnik. Die Grenzwerte der technologischen Möglichkeiten werden derzeit nur von Speicherbausteinen erreicht, bei denen aufgrund ihrer ausgesprochen regulären und modularen Architektur im wesentlichen "nur" schaltungstechnische und technologische Probleme zu lösen sind. Bei komplexen logischen Bausteinen, wie z. B. Mikrorechnern, ist das Problem der hohen Entwurfs- und Testkomplexität in den Vordergrund getreten. Gefordert sind daher Systemkonzepte, die einerseits die Möglichkeiten der Technologie weitestgehend ausschöpfen und andererseits die Entwurfs- und Testkomplexität auf ein beherrschbares Maß reduzieren.

Eine neue Herausforderung an die Großintegrationstechnik [1] stellt sich diesbezüglich durch die in jüngster Zeit viel diskutierten *neuronalen Netzwerkmodelle.* Diese Modelle basieren auf grundlegenden Kenntnissen bzw. Annahmen aus der Neurophysiologie. Mit ihnen versucht man die Eigenschaften und Leistungen biologischer Nervensysteme von Tieren und letztlich auch von Menschen zu ergründen. Die Natur dient hier einmal mehr als Wegweiser für die Erforschung neuer Prinzipien in der Informationsverarbeitung und für die Entwicklung neuer technischer Verarbeitungsstrukturen.

Neuronale Netzwerke sind sowohl von der Architektur als auch von den Eigenschaften her für die Großintegrationstechnik interessant. Die Netzwerke sind aus vielen (≥ 1000) einfachen und gleichartigen Verarbeitungseinheiten aufgebaut, die über unidirektionale gewichtete Verbindungsleitungen hochgradig miteinander vernetzt sind. Die meisten Modelle gehen von einer regelmäßigen Verbindungsstruktur aus. Neuronale Netzwerkmodelle weisen daher die für einen integrationsgerechten Entwurf charakteristischen Eigenschaften der Regularität und Modularität auf.

Während konventionelle Rechnerarchitekturen eine Trennung in einen aktiven Prozessor und einen passiven Speicher vornehmen, verbinden neuronale Netzwerke die Speicherung und Verarbeitung von Informationen in ihrer Struktur. Die Informationsverarbeitung erfolgt kollektiv durch ein massives paralleles Zusammenwirken aller Verarbeitungseinheiten. Das wesentliche Verarbeitungsprinzip ist die Übertragung von einfachen Signalen zwischen diesen Einheiten. Ein neuronales Netzwerk wird daher nicht herkömmlich programmiert, sondern es muß eine Festlegung der Verbindungsstruktur erfolgen.

Das verteilte und kollektive Verarbeitungsprinzip läßt neuronale Netzwerke unempfindlich gegen den Ausfall einzelner Verarbeitungseinheiten oder Verbindungen werden. Derartige Ausfälle führen nicht zu einem Systemausfall, sondern werden bis zu einem gewissen Grad vollständig kompensiert. Weitere Ausfälle führen dann zu einem allmählichen Absinken der Leistungseigenschaften des Systems (graceful degradation). Die systeminhärente Fehlertoleranz macht neuronale Netzwerke attraktiv für ein integriertes Schaltungskonzept, denn bei den derzeitigen Integrationsgraden ist man auf fehlertolerante Systemkonzepte angewiesen.

Die Zielsetzung, ausgewählte Fähigkeiten biologischer Nervensysteme auf Maschinen mit neuronaler Architektur zu übertragen, ist keinesfalls neu. Sie ist seit den fünfziger Jahren durch die rasante und erfolgreiche Entwicklung der *von-Neumann-Architektur* moderner Digitalrechner in den Hintergund getreten. Heute stoßen wir jedoch allmählich an die Grenzen dieses sequentiellen Rechnerprinzips. Trotz faszinierender Leistungen in vielen Bereichen der Informationsverarbeitung mangelt es den Systemen an Fähigkeiten, die selbst einfache Lebewesen anscheinend mühelos vollbringen. Beispiele sind die Wahrnehmung und Erkennung sowohl visueller als auch akustischer Eindrücke. Fähigkeiten, die auch mit den symbolisch arbeitenden Methoden der *Künstlichen Intelligenz* sowie den analytischen Methoden der Mustererkennung bisher nicht befriedigend gelöst werden konnten. Das hat in den letzten Jahren wieder zu einem verstärkten Interesse an neuronalen Netzwerken geführt.

In der Literatur der letzten 5 Jahrzehnte finden sich viele verschiedene Modelle zu neuronalen Netzwerken, deren Funktionsweise und Anwendbarkeit theoretisch bzw. durch Simulationen auf seriellen Rechnern verifiziert wurden. Die Simulation von Netzwerken mit mehreren tausend Verarbeitungseinheiten, die für eine Anwendung interessant sind, führt aber zu hohen Simulationszeiten im Bereich von Stunden oder gar Tagen. Die vollständige Effizienz und Leistungsfähigkeit dieser hochgradig parallel arbeitenden Netzwerke kann somit nur durch eine geeignete Hardwarearchitektur erreicht werden. Es bieten sich hier zwei unterschiedliche Lösungswege an (Abb. 1.1): eine universell programmierbare Architektur oder eine modellspezifische Realisierung.

Die erste Gruppe, die *Neurocomputer*, sind weitestgehend Anpassungen bzw. Erweiterungen von bekannten parallelen Rechnerstrukturen an die Simulation von neuronalen Netzen. Das Spektrum erstreckt sich von Spezialprozessorkarten, die sich durch schnelle Gleitkommaverarbeitung und große Speicher auszeichnen [2], bis zu systolischen [3] und zellularen Parallelrechnerarchitekturen [4], die durch eine möglichst große Anzahl parallel arbeitender Prozessoren eine Beschleunigung erzielen. Neuronale Netzwerke werden von Neurocomputern nur *virtuell* implementiert, d.h. die Anzahl der parallel arbeitenden Prozessoren ist kleiner als die Anzahl der Verarbeitungseinheiten des neuronalen Netzes.

Die maximale Größe des Netzwerkes hängt nur von dem vorhandenen Speicherplatz ab. Das Hauptproblem beim Entwurf eines Neurocomputers ist der hohe Vernetzungsgrad neuronaler Netzwerke. Eine Aufteilung dieser feinkörnigen Netzwerke auf ein grobkörniges paralleles Mikrorechnernetzwerk führt oft zu einem Kommunikationsengpaß, der den Datendurchsatz eines parallelen Systems erheblich reduzieren kann.

Neurocomputer sind wichtige Hilfsmittel zur Modellierung und Analyse neuer Netzwerkmodelle. Sie sollten in der Lage sein, viele verschiedene theoretische Netzwerkmodelle zu unterstützen. Für diesen Freiheitsgrad wird eine Programmumgebung und eine angepaßte parallele Rechnerarchitektur benötigt. Ein leistungsfähiger Neurocomputer zeichnet sich somit durch eine sorgfältig aufeinander abgestimmte Soft- und Hardwarestruktur aus.

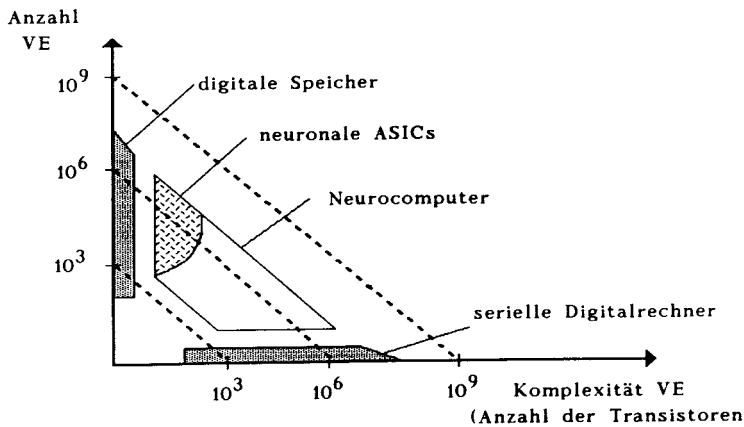


Abb. 1.1: Spektrum neuronaler Rechnerarchitekturen nach Recce / Treleaven [5] (VE = Verarbeitungseinheit).

Die zweite Gruppe, die *neuronalen ASICs* (application specific integrated circuits), bilden integrierte Bausteine, die auf ein bestimmtes Netzwerkmodell und eine spezielle Anwendung zugeschnitten sind. Diese können in rein digitaler, gemischt digital-analoger bzw. rein analoger Schaltungstechnik ausgeführt sein. Im Gegensatz zu Neurocomputern werden die Netzwerke nicht virtuell implementiert, sondern es werden alle Verarbeitungseinheiten des Netzwerkes physikalisch realisiert. Der zusätzliche Programmieraufwand beschränkt sich auf ein Minimum.

Neuronale ASICs sind für praktische Anwendungen von neuronalen Netzwerken wichtig. Hat man für eine bestimmte Anwendung die geeignete neuronale Struktur gefunden, dann ist der Freiheitsgrad der Modellierung nicht mehr notwendig. Es treten die Faktoren Zeit- und Platzbedarf in den Vordergrund, die im allgemeinen durch eine spezielle integrationsgerechte Implementierung optimiert werden können. Während Neurocomputer die systeminhärente Fehlertoleranz und die massive Parallelität neuronaler Netzwerke nur eingeschränkt ausnutzen können, lassen sich beide Eigenschaften direkt auf spezielle Schaltungsimplementierungen übertragen.

1.1 Zielsetzung

Die vorliegende Arbeit leistet einen Beitrag zum Entwurf hochintegrierter neuronaler ASICs. Die Forschungs- und Entwicklungsarbeiten zu diesem Thema stehen derzeit erst am Anfang. Dementsprechend diffus stellt sich der Stand der Technik auf diesem Gebiet in der Literatur dar. Für den Vergleich verschiedenartiger Netzwerkmodelle und deren systematische Umsetzung in eine integrationsgerechte Architektur ist eine einheitliche Nomenklatur und Beschreibungsform notwendig. Aus diesem Grund besteht die erste Aufgabe darin, einen allgemeinen Beschreibungsformalismus anzugeben, mit dem sich viele der heute bekannten Modelle spezifizieren lassen. Im Mittelpunkt stehen assoziative Netzwerkmodelle, die eine Unterklasse innerhalb der neuronalen Netzwerke bilden (Kapitel 2).

Zur Zeit gibt es noch keine allgemeingültigen Aussagen darüber, welches neuronale Netzwerkmodell für welche Anwendung eine effiziente Lösung bietet. Der Entwurfsaufwand für ein neuronales ASIC ist aber nur dann gerechtfertigt, wenn das ausgewählte Modell einen technischen Nutzen hat, d.h. einen Vorteil gegenüber alternativen Lösungen aufweist. Ein Vergleich verschiedener Lösungsansätze läßt sich anhand quantitativer Vergleichsmaße erreichen. Ein weiteres Ziel ist daher auf der Grundlage von Angaben aus der Literatur anwendbare Aussagen hinsichtlich der Speichereffektivität und Fehlertoleranz ausgewählter Modelle zu formulieren (Kapitel 3).

Ist die Entscheidung für ein bestimmtes neuronales Netzwerkmodell gefallen, stellt sich die Frage nach einer integrationsgerechten Realisierung. Die Mikroelektronik bietet verschiedene Realisierungsmöglichkeiten für neuronale ASICs, von denen die wichtigsten in dieser Arbeit aufgezeigt und die Eigenschaften ausgewählter Grundschaltungen mit Hilfe von integrierten Testschaltungen überprüft werden sollen.

Grundlage für die Schaltungsrealisierungen ist der am Lehrstuhl Bauelemente der Elektrotechnik der Universität Dortmund entwickelte 3µm-N-Wannen-CMOS-Prozeß mit Polysilizium-Gates und einer Aluminium Verdrahtungsebene [6]. Anhand dieser Untersuchungen soll eine Bewertung der verschiedenen Schaltungskonzepte hinsichtlich ihrer Auswirkungen auf die Speichereigenschaften assoziativer Netzwerke erfolgen, die für eine integrationsgerechte Umsetzung unerläßlich ist (Kapitel 4).

Auf der Grundlage der theoretischen und praktischen Ergebnisse gilt es abschliessend assoziative Netzwerkmodelle, deren technischer Nutzen nachgewiesen worden ist, in Form von neuen hochintegrierten Bausteinen zu realisieren. Im einzelnen sind verschiedene Realisierungsmöglichkeiten (digital/analog) und Lösungen für die wesentlichen Problemstellungen einer VLSI-Implementierung zu erarbeiten. Im Rahmen dieser Arbeit soll schließlich die integrationsgerechte Umsetzung eines assoziativen Netzwerkes in ein neues mikroelektronisches Bauelement durchgeführt und diskutiert werden (Kapitel 5).

2. Grundlagen assoziativer Netzwerke

2.1 Neurophysiologische Motivation

Die Erforschung biologischer Nervenysteme, besonders der Gehirne höherer Lebewesen, ist seit jeher im Interesse der Wissenschaften. Während die abstrakten, makroskopischen Fähigkeiten von Gehirnen, wie etwa das Denken, Planen oder Entscheiden, noch weitgehend unverstanden sind, hat die Erforschung der anatomischen Struktur und der Funktionsweise einzelner Grundelemente zu einer Fülle von mikroskopischen Fakten geführt. In diesem Abschnitt werden einige der grundlegenden Erkenntnisse und Annahmen aus der Neurobiologie zusammengefaßt, auf denen im wesentlichen die Modellannahmen für die neuronalen Netzwerkmodelle beruhen. Die Angaben stammen aus den Literaturquellen [7-9], in denen ausführliche Informationen zu diesem Thema zu finden sind.

Biologische Nervensysteme bestehen aus einer sehr großen Anzahl von Nervenzellen (*Neuronen*), die intensiv untereinander vernetzt sind. Das menschliche Gehirn zum Beispiel enthält grob geschätzt 10 Milliarden Nervenzellen. Die Neuronen haben eine baumartige, weitverzweigte Erscheinungsform (Abb. 2.1.1) und sind mit bis zu 10^4 anderen Neuronen verbunden. Sie bestehen aus einem Zellkörper, vielen Eingabezweigen (*Dendriten*) und nur einem Ausgabezweig (*Axon*). Über die dendritischen Verzweigungen und den Zellkörper werden Eingabesignale von anderen Neuronen aufgenommen. Der Zustand eines Neurons wird durch das sogenannte *Membranpotential* bestimmt, das im Ruhezustand bei ca. - 70 mV liegt und durch die Eingangssignale verändert wird. Überschreitet das räumlich und zeitlich aufintegrierte Membranpotential einen Schwellenwert (ca. -40 mV), so wird ein Ausgabesignal in Form eines Spannungsimpulses (*Spike*) generiert. Der Impuls hat eine Dauer von ca. 1 ms und eine Amplitude von ca. 120 mV bezüglich des Ruhepotentials. Nach der Aussendung eines Spikes folgt ein Zeitraum, in dem keine Erregung der Zelle möglich ist (*Refraktärzeit*). Die Refraktärzeit beträgt einige Millisekunden, so daß die maximale Impulsfrequenz bei ca. 100 Impulsen pro Sekunde liegt. Die Nervenzellen kommunizieren nach dem Prinzip der Impulsfrequenz-Kodierung. Die Information steckt in der Frequenz und Phase des Auftretens der Ausgangsimpulse.

Das Axon verzweigt sich sehr vielfältig und leitet die Ausgabeimpulse an viele tausend (ebenfalls ca. 10000) andere Nervenzellen weiter. An den Verzweigungsstellen wird der Impuls vervielfacht, d. h. es laufen identische Impulse weiter. Die Verbindungsstellen zwischen den Enden der Axonzweige und den nachfolgenden Nervenzellen heißen *Synapsen*. Synapsen werden nach ihrer Lage und Wirkung typisiert. Je nach Lage

spricht man von axo-dentritischen, axo-somatischen, axo-axonischen und dendro-dentritischen Synapsen. Synapsen der beiden erst genannten Gattungen treten am häufigsten auf. Die Wirkung der Synapsen kann entweder erregend (exzitatorisch), d.h. sie bewirken die Erhöhung des Membranpotentials oder hemmend (inhibitorisch) sein, also eine Erniedrigung des Membranpotentials bewirken. Es wird davon ausgegangen, daß Synapsen ihre Wirkung nicht umkehren können. Eine exzitatorische Synapse kann also nicht zu einer inhibitorischen werden und umgekehrt.

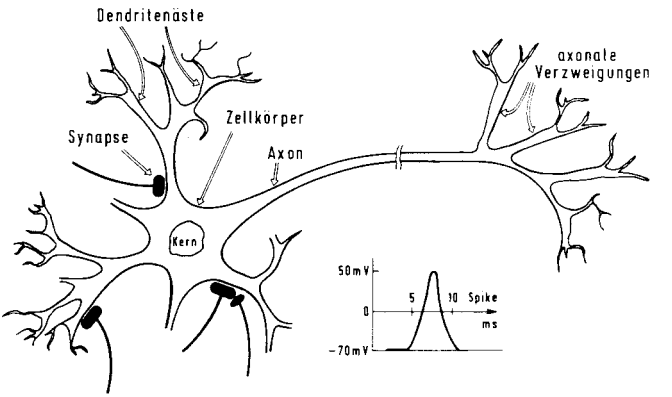


Abb. 2.1.1: Vereinfachtes Schema eines Neurons.

Es konnte ferner nachgewiesen werden, daß für bestimmte Synapsen die Stärke, mit der eine Nervenzelle auf eine andere wirkt, veränderbar ist. Demnach werden diejenigen Neuronen stärker miteinander verknüpft, die häufig gemeinsam erregt sind, d. h. etwa zur gleichen Zeit Spikes generieren. Diese Veränderung der Übertragungseigenschaft von Synapsen bereits 1949 von D. Hebb [10] als Hypothese formuliert worden. Sie wird heute als eine Grundlage für das Lernen betrachtet. Die Synapsen werden somit als die elementaren Einheiten des biologischen Gedächtnisses interpretiert, in denen unser Wissen "gespeichert" wird.

Die Signalübertragungseigenschaften von Neuronen und Synapsen sind im Detail recht kompliziert und noch nicht vollständig erforscht. Noch relativ unklar ist das Verrechnungsprinzip der ankommenden Signale. Untersuchungen deuten darauf hin [11], daß der Einfluß der Synapsen auf die Aktivität eines Neurons von der Lokalität der Synapsen abhängen kann. Somit sind eine Reihe von lokalen Operationen möglich, die die Geometrie

des Dendritenbaumes und die relative Lage der Synapsen zueinander ausnutzen. Auch der Einfluß der vielfältigen chemischen Wechselwirkungen auf die Informationsverarbeitung im Nervensystem ist noch nicht ausreichend verstanden. Es gilt aber als sicher, daß Neuronen zumindest die oben erwähnte Schwellenoperation ausführen. Ferner geht man davon aus, daß die Neuronen asynchron arbeiten, denn es gibt bisher keinen Hinweis auf einen zentralen Systemtakt.

Die Neuronen werden als die strukturellen und funktionellen Grundbausteine biologischer Nervensysteme angesehen. Aufgrund ihrer Erscheinungsformen unterscheidet man verschiedene Neuronentypen, wie z.B. Purkinjezellen, mit einem dichten weitverzweigten Dendritenbaum oder Pyramidenzellen, deren Zellkörper die Form einer Pyramide haben. Anatomische Untersuchungen zeigen, daß verschiedene Gehirnbereiche mit unterschiedlichen Neuronentypen aufgebaut sind. In der Großhirnrinde, die der größte ausgebildete Teil des menschlichen Gehirns ist und für die "höheren" geistigen Fähigkeiten verantwortlich gemacht wird, findet man z.B. vorwiegend zwei Neuronentypen: die Pyramidenzellen (ca. 70%) und die Sternzellen (ca. 30%). Die Sternzellen verzweigen sich überwiegend in ihrer unmittelbaren Umgebung und wirken in der Regel inhibitorisch. Die Pyramidenzellen haben weitreichende Axone mit meist exzitatorisch wirkenden Synapsen. Die verschiedenen Neuronentypen scheinen für bestimmte Aufgaben spezialisiert zu sein. Funktional unterschiedliche Gehirnbereiche sind daher auch architektonisch unterschiedlich.

Die von der Neurobiologie erforschten mikroskopischen Fakten, insbesondere die anatomischen Fakten, sind selbstverständlich viel detaillierter als es hier zum Ausdruck kommen kann. Das Verständnis der Informationsverarbeitung in Nervensystemen ist aber dennoch rudimentär. Es gilt hier die große Kluft zwischen den mikroskopischen Detailkenntnissen und den makroskopischen Fähigkeiten, wie zum Beispiel der Wahrnehmung, zu überbrücken. Dies motiviert Modellbildungen und -analysen, um Hinweise zu bekommen, inwieweit Konfigurationen einfacher Modellneuronen ausgewählte Eigenschaften hervorbringen können und einen naturgetreuen Nachbau notwendig machen.

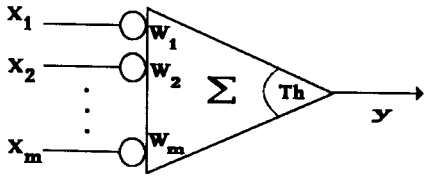
Eine wichtige Frage zielt auf die interne Repräsentationsart der im Gehirn gespeicherten Informationen. Eine heute weitverbreitete Annahme ist, daß Daten im Gehirn nicht lokal, sondern *verteilt* gespeichert werden. In der Gehirntheorie spricht man von *Zellverbänden* (cell assemblies [10,12]). Konzepte werden im Gehirn nicht durch einzelne Neuronen, sondern durch einen Verband von miteinander verknüpften Neuronen repräsentiert. Die interne Informationsdarstellung ist somit ein Aktivitätsmuster gleichzeitig erregter Neuronen, deren synaptischen Verbindungen die Korrelationen zwischen gleichzeitig auftretenden Ereignissen speichern. Die Entstehung und Stabilität dieser Verbände sind so zu erklären, daß gleichzeitig akti-

vierte Neuronen aufgrund der erwähnten synaptischen Plastizität exzitatorische Verbindungen bilden. Ein Verband besteht daher aus sich gegenseitig erregenden Neuronen. Einzelne Neurone eines Verbandes haben immer das Bestreben, bei Aktivierung auch die mit ihnen verbundenen Neuronen zu erregen. Mit Zellverbänden läßt sich anschaulich die Fähigkeit von Gehirnen erklären, unvollständige oder verrauschte Sinneseindrücke, wie z.B. Bilder oder Tonfolgen, richtig zu erkennen. Der Sinneseindruck aktiviert einen genügend großen Teil des Zellverbandes, der diesen Sinneseindruck im Gehirn repräsentiert. Über die exzitatorischen Verbindungen erregen die aktivierten Neuronen die noch fehlenden Neuronen des Verbandes und vervollständigen somit den Sinneseindruck. Diese Annahmen haben insbesondere die Entwicklung assoziativer Speichermodelle motiviert, die sich einer verteilten Speicherung bedienen und im Mittelpunkt dieser Arbeit stehen.

Seit den vierziger Jahren gibt es viele Ansätze anhand von theoretischen Modellen die Informationsverarbeitung und -repräsentation in biologischen Nervensystemen zu ergründen. Erklärtes Ziel dieser Modelle ist es, ausgewählte makroskopische Fähigkeiten des zentralen Nervensystems vereinfacht nachzubilden. In den fünfziger und sechziger Jahren hatte dieses Bestreben im Rahmen der Kybernetik seinen ersten Höhepunkt. Beispielhaft genannte Vertreter dieser Zeit sind Rosenblatt (Perzeptron, [13]), Uttely (Conditioned Probability Computer, [14]), Steinbuch (Lernmatrix, [15]) und Widrow (Adaline, [16]), deren Neuronenmodelle auf den grundlegenden Arbeiten von McCulloch und Pitts [17] sowie Hebb [10] basieren.

Das McCulloch/Pitts-Neuron (Schwellenneuron) ist ein sehr einfaches Modell eines biologischen Neurons (Abb. 2.1.2). Es empfängt m binäre Eingabewerte x_i und hat einen binären Ausgang y. Jeder Eingabe ist ein Gewichtswert w_i eindeutig zugeordnet. Wählt man die Refraktärzeit eines Neurons als einen diskreten Zeitschritt $t \in \mathbb{N}$, so erzeugt das Schwellenneuron einen Ausgabeimpuls zur Zeit t+1, wenn die Aktivierung des Schwellenneurons zur Zeit t den Schwellenwert Th überschreitet. Die Aktivierung des Modellneurons ergibt sich aus der gewichteten Summe der Eingabewerte zur Zeit t. Die Gewichtswerte w_i modellieren die Übertragungsstärke der Synapsen. Sie können positiv, negativ oder null sein entsprechend erregenden, hemmenden oder nicht vorhandenen Synapsen.

Das Schwellenneuron nimmt nur die beiden binären Ausgangswerte 0 und 1 an, so daß man es als ein logisches Schaltelement (Schwellenwertgatter) betrachten kann. Im Sinne der Schaltwerktheorie ist ein Schwellenwertgatter ein vollständiges System logischer Operatoren, d.h. mit einem genügend großen und richtig zusammengeschalteten Netzwerk dieser Gatter kann man alle Booleschen Funktionen realisieren. Mit anderen Worten, ein solches Netz hat die Berechnungsmächtigkeit von Turingmaschinen [18].



$$a(t) = \sum_{i=1}^{m} x_i(t) \cdot w_i(t)$$

$$y(t+1) = \begin{cases} 1, & \text{wenn } a(t) \geq Th \\ 0, & \text{sonst.} \end{cases}$$

$x_i,\ y \in \{0,1\};\ w_i,\ Th \in \mathbb{R}.$

$i = 1, \ldots m.$

Abb 2.1.2: Schema für ein einfaches Schwellenneuron.

Für die Programmierung einzelner Schwellenneurone hat man Lernregeln entwickelt, mit denen sich die Gewichte und der Schwellenwert iterativ einstellen lassen. Beispielhaft sei hier die Perzeptron-Lernregel aufgeführt [13]:

$$w_i(t+1) = w_i(t) + c \cdot (y_L - y(t)) \cdot x_i(t) \qquad (2.1.1)$$

Entsprechend dieser Regel wird ein Gewicht verändert, wenn die erzeugte Ausgabe des Modellneurons $y(t)$ von der geforderten Ausgabe $y_L \in \{0,1\}$ (Lehrersignal) abweicht. Die Konstante $c \in [0,1]$ bestimmt den Betrag der Gewichtskorrektur. Es ist auf verschiedene Weise gezeigt worden [13], daß das gewünschte Ein-/Ausgabeverhalten in einer endlichen Anzahl von Lernschritten erreicht wird, sofern dieses Verhalten von diesem Modellneuron überhaupt erzeugt werden kann.

Mit einem Schwellenneuron können nur linear trennbare Mustermengen erkannt werden. Für komplexere Probleme benötigt man eine Verschaltung von Schwellenneuronen zu einem komplexeren Netzwerk [19]. Eine Verallgemeinerung der Perzeptron-Lernregel auf komplexere, mehrschichtige Netzwerke, bei denen man nicht mehr für jedes Element das entsprechende Lehrersignal bereitstellen kann, ist in den sechziger Jahren nicht gelungen. Damit konnten sich diese Modelle gegenüber dem symbolverarbeitenden Ansatz der *Künstlichen Intelligenz* nicht durchsetzen. Die rasche Entwicklung der elektronischen Datenverarbeitung ermöglichte zunehmend komplexere Programmentwicklungen. Diese Entwicklung nährte die Hoffnungen der Wissenschaftler, die höheren geistigen Fähigkeiten in Form von "intelligenten" Programmen auf Rechenautomaten zu implementieren. Gegen Ende der sechziger Jahre sind die neuronalen Netzwerkmodelle aus dem Rampenlicht der Forschung verschwunden.

Die Forschungsarbeiten auf diesem Gebiet sind in den siebziger Jahren weitergeführt worden, wenn auch das öffentliche Interesse nicht sehr groß war. Das rudimentäre Wissen über Nervensysteme ist erweitert und

es sind neue modelltheoretische Erkenntnisse erlangt worden. Beispielhaft genannte europäische Vertreter dieser Zeit sind Caianiello, Willshaw, von der Malsburg, Kohonen und Braitenberg. Eine Zusammenstellung wichtiger Beiträge dieser Zeit findet sich z.B. in [20,21].

Anfang der achtziger Jahre bekam das ohnehin schon ausgesprochen interdisziplinäre Gebiet durch eine physikalische Betrachtungsweise einen neuen Akzent. Die Physik befaßte sich schon seit langem mit dem Verhalten von Vielteilchensystemen mit lokal wechselwirkenden Einzelelementen. Die Eigenschaften von Materialien entstehen z.B. durch das Zusammenwirken von 10^{23} Atomen oder Molekülen. Eine Reihe von Physikern versucht daher, die physikalischen Modelle von Vielteilchensystemen, wie z.B. Spinglas-Modelle [22], auf neuronale Netze zu übertragen. Zwei der bekanntesten Modelle sind das Hopfield-Netz [23] und die Boltzmann-Maschine [24].

Heute erleben wir eine Renaissance neuronaler Netze. Viele Faktoren haben dazu beigetragen, z.B. die sich abzeichnende Ernüchterung in der Künstlichen Intelligenz bezüglich bestimmter praktischer Anwendungen (z.B. sensorische Informationsverarbeitung), ein wachsendes Interesse an parallelen Rechnerstrukturen und nicht zuletzt die zunehmende Integrationsdichte in der Mikroelektronik. Insbesondere sind auch theoretische Grundlagen für mehrschichtige und selbstorganisierende Netzwerke [25-27] entwickelt worden.

Das enorme Interesse an neuronalen Netzwerken hat sich in jüngster Zeit auch auf die Großintegrationstechnik übertragen. Im Mittelpunkt steht die Frage, in welcher Form und inwieweit sich neuronale Netzwerkmodelle als mikroelektronische Bauelemente verwirklichen lassen [28]. Das Ziel ist dabei weniger ein naturgetreuer Nachbau biologischer Nervensysteme, sondern eher das Finden von alternativen Architekturen für die Informationsverarbeitung. Anhaltspunkte ergeben sich aus den wesentlichen strukturellen und funktionellen Unterschieden von biologischen Nervensystemen und heutigen Rechnersystemen, die qualitativ in Tabelle 2.1.1 zusammengefaßt sind.

Biologische Nervensysteme erreichen ihre Leistungsfähigkeit durch ein kollektives Zusammenwirken einer sehr großen Anzahl von gleichartigen und vermutlich einfachen Verarbeitungskomponenten. Durch die massiv parallele Arbeitsweise wird auch mit langsamen Verarbeitungseinheiten eine hohe Verarbeitungsgeschwindigkeit erreicht. Die Verarbeitungsgeschwindigkeit der Nervenzellen ist dabei um mehrere Größenordnungen kleiner als bei modernen Mikrorechnern. Die typische Zeitskala für Nervenzellen liegt im Millisekundenbereich, typische Speicherzugriffszeiten liegen im Nanosekundenbereich (50 - 100ns) und Prozessortaktraten bei einigen 10 MHz.

Digitalrechner	Eigenschaften	Nervensystem
	des Gesamtsystems:	
gering	Parallelität	hoch
gering	Vernetzungsgrad	hoch
lokal	Speicherprinzip	verteilt
mäßig	Fehlertoleranz	hoch
synchron	Kommunikation	asynchron
	der Komponenten:	
hoch	Rechengeschwindigkeit	niedrig
hoch	Rechengenauigkeit	niedrig
hoch	Zuverlässigkeit	mäßig

Tabelle 2.1.1: Qualitative Gegenüberstellung der Eigenschaften eines biologischen Nervensystems und eines heutigen Digitalrechners.

Der hohe Grad an Parallelität und Kollektivität wird durch eine einfache Kommunikation über unidirektionale Verbindungsleitungen erreicht. Gleichzeitig ergibt sich ein hohes Maß an Fehlertoleranz. Die Unzuverlässigkeit und Ungenauigkeit einzelner Verbindungen oder Verarbeitungseinheiten kann durch das verteilte und kollektive Arbeitsprinzip ausgeglichen werden.

Einen bremsenden Einfluß im Fortschritt der Informationstechnologie hat zunehmend die Entwicklung komplexer Programme übernommen. Da Nervensysteme z. B. akustische und visuelle Wahrnehmungen in Sekundenbruchteilen erledigen, können im Gehirn nur wenige hundert sequentielle Schritte ausgeführt werden. Die kognitiven Fähigkeiten können daher nur durch massive Parallelität ohne komplexe algorithmische Unterstützung erreicht werden.

Diese Eigenschaften biologischer Nervensysteme erklären das erneute wissenschaftliche und kommerzielle Interesse an dieser Thematik.

2.2 Eine allgemeine Modellbeschreibung

Das Forschungsinteresse an neuronalen Netzwerken hat sich in den letzten Jahren rasch auf viele Wissenschaftsbereiche ausgeweitet. Die Begriffsbildung ist daher noch diffus und uneinheitlich. Für den Vergleich der Eigenschaften und Verwendungsmöglichkeiten unterschiedlicher Modelle ist aber eine einheitliche Beschreibungsform gefordert. Die Form der Beschreibung hängt im wesentlichen von der Zielsetzung ab. Für eine mathematische Analyse muß das Neuronenverhalten und die Netzwerkstruktur mit exakten mathematischen Strukturen, im allgemeinen Differentialgleichungen, beschrieben werden. Weitere geeignete mathematische Strukturen finden sich in der Automatentheorie [29], Graphentheorie [30] oder in der Physik [31]. Anhand von Axiomen und Theoremen werden dann die ausgewählten Modelleigenschaften bewiesen und neue Eigenschaften vorhergesagt.

Eine geschlossene mathematische Analyse ist nur für einfache Modelle möglich, die sich anhand weniger Gleichungen und Parameter beschreiben lassen. Im Grunde benötigt man für jedes Neuron im Netzwerk eine Gleichung zur Beschreibung der Übertragungseigenschaften, für jede Synapse eine Gleichung für die Änderung der Kopplungsstärke sowie Parameter, die z.B. die Anfangsverknüpfung im Netzwerk festlegen. Das Verhalten der meisten Modelle wird daher mit Rechnersimulationen untersucht. Eine Modellbeschreibung sollte in dieser Hinsicht viele Modellvarianten überdecken, so daß nicht für jede Variante Änderungen am Simulationsprogramm selbst vorgenommen werden müssen. Andererseits darf die erreichte Flexibilität nicht zu Lasten der Genauigkeit, der Simulationszeit oder des Speicherplatzbedarfs gehen. Hier gilt es einen geeigneten Kompromiß zu finden.

Für den Systementwickler integrierter Schaltungen kommen nur Modelle in Frage, deren Funktionalität und Anwendbarkeit bereits theoretisch bzw. mit Hilfe von Simulationen nachgewiesen worden sind. Im Rahmen dieser Arbeit wird daher eine Beschreibungsform gewählt, die den strukturellen Aufbau des Netzwerkes mit seiner Einbettung in eine gegebene Umwelt deutlicher zum Ausdruck bringt. Sie erhebt nicht den Anspruch auf Universalität, sondern beschränkt sich auf eine Untergruppe von neuronalen Netzwerken, die im folgenden mit *"assoziative Netzwerke"* bezeichnet wird.

Die Beschreibungsform orientiert sich bei der Verarbeitungseinheit an der Definition eines endlichen Automatens und beim Netzwerk an der Definition eines abstrakten Graphens. Sie eignet sich dadurch ebenfalls für die Implementierung eines modellunabhängigen Simulators für assoziative Netzwerke [32]. Die gewählte Beschreibungsform bildet somit eine geeignete Schnittstelle zwischen struktureller Beschreibung, Simulation und Theorie assoziativer Netzwerke.

2.2.1 Definition einer Verarbeitungseinheit

Grundlegendes Systemelement eines künstlichen neuronalen Netzwerkes ist ein Modellneuron, das im weiteren mit *Verarbeitungseinheit* bezeichnet wird:

Definition 1: Eine ***Verarbeitungseinheit*** ist ein Tupel $V=(m,X,A,Y,W,\delta,\alpha,\lambda)$ mit:

- m : Anzahl der Eingabeleitungen;
- X : Menge der Eingabewerte;
- A : Menge der Aktivierungswerte;
- Y : Menge der Ausgabewerte;
- W : Menge der Gewichtswerte;
- δ : Aktivierungsfunktion;
  $\delta : X^m x W^m x A \longrightarrow A$
- α : Ausgabefunktion;
  $\alpha : A \longrightarrow Y$
- λ : Adaptationsfunktion (Lernregel);
  $\lambda : W^m x X^m x A x Y \longrightarrow W^m$

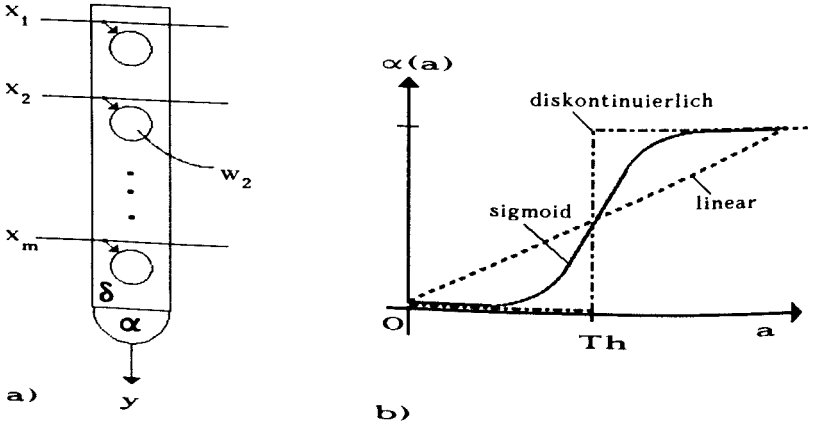


Abb. 2.2.1: Schematischer Aufbau einer Verarbeitungseinheit (a) und Beispiele gebräuchlicher Ausgabefunktionen (b).

Die Struktur einer Verarbeitungseinheit (Abb. 2.2.1) wird durch die m Eingangsleitungen, auf denen Eingabewerte $x_i \in X$ übertragen werden können, einer Ausgabeleitung, auf der Ausgabewerte $y \in Y$ übertragen werden können sowie den m Verbindungsgewichten w_i, die eineindeutig den Eingangsleitungen zugeordnet sind, bestimmt. Die Eingabe- und die Ausgabemenge werden häufig gleich gewählt (X=Y) und die Eingabewerte sowie die Gewichtswerte zu einem m-dimensionalen Eingabevektor $\underline{x}$ bzw. Gewichtsvektor $\underline{w}$ zusammengefaßt.

Zur Vereinfachung wird im folgenden für eine Eingangsleitung die gleiche Bezeichnung x_i gewählt wie für die Eingabewerte auf dieser Leitung. Entsprechendes gilt für die Ausgangsleitung y und die Gewichte w_i. Formal gesehen ist das nicht ganz korrekt, aber für die hier angestellten Betrachtungen führt diese Vereinfachung nicht zu Konflikten.

Der Zustand einer Verarbeitungseinheit wird durch den Aktivierungswert $a \in A$, der neurophysiologisch mit dem Membranpotential vergleichbar ist, und durch den Gewichtsvektor $\underline{w}$ ausgedrückt. Entsprechend der Wahl der Mengen X,Y,W,A (diskret-kontinuierlich, beschränkt-unbeschränkt) ergeben sich unterschiedliche Modelle für formale Neuronen, von denen viele bereits in der Literatur untersucht worden sind [25-27].

Die Dynamik einer Verarbeitungseinheit wird durch die drei Funktionen δ,α,λ bestimmt. Die Verarbeitungseinheiten können bezüglich einer kontinuierlichen oder diskreten Zeitskala arbeiten, entsprechend erhält man eine Beschreibung durch Differentialgleichungen oder Iterationsgleichungen. Der zeitdiskrete Fall ist immer dann zwingend, wenn man die Verarbeitungseinheiten auf einem Rechner simulieren will. Die meisten Modelle beziehen sich daher auf eine diskrete Zeitskala, so daß $t \in \mathbb{N}$ gilt. Diese Betrachtungsweise wird für die weiteren Ausführungen übernommen.

Eine Verarbeitungseinheit arbeitet zu den diskreten Zeittakten $t = 0,1,2, \ldots$, an denen sie in Abhängigkeit von der aktuellen Eingabe $\underline{x}(t)$ und dem aktuellen Zustand ($\underline{w}(t)$, $a(t)$) eine neue Ausgabe $y(t)$ erzeugt und einen Zustandswechsel durchführt. Sowohl die Ausgabe- als auch die Zustandsüberführungsfunktionen (α,λ) sind zeitinvariant.

Für genaue Modellierungen der biologischen Vorbilder muß man eine entsprechend fein auflösende Zeitskala verwenden und relativ aufwendig die Impulsübertragungen modellieren (vgl. 2.1). Da die Analyse und Simulation dieser Systeme sehr viel Zeit in Anspruch nimmt, werden oft zeitlich gemittelte Größen, beispielsweise die der Impulsrate, oder nur binäre Zustände zur Approximation der Neuronenausgabe verwendet.

Die Aktivierungsfunktion bestimmt einen neuen Aktivierungswert in Abhängigkeit vom aktuellen Aktivierungszustand, den aktuellen Gewichten

$\underline{w}(t)$ und der aktuellen Eingabe $\underline{x}(t)$. Die meisten derzeit bekannten Aktivierungsfunktionen basieren auf der gewichteten Summe der Eingabewerte:

$$S(t) = \sum_{i=1}^{m} x_i(t) \cdot w_i(t) \qquad (2.2.1)$$

$$a(t) = \delta(S(t)) \qquad (2.2.2)$$

Die Aktivierung ergibt sich entweder direkt aus dieser Summe oder es wird noch eine nichtlineare Funktion auf diese Summe angewendet. Komplexere Aktivierungsfunktionen enthalten eine zeitliche Integration und komplexere mathematische Operationen als die Summenbildung [26].

Die Ausgabefunktion bestimmt den Ausgabewert in Abhängigkeit von der momentanen Aktivierung. Neben einfachen linearen Ausgabefunktionen findet man häufig sigmoide Ausgabefunktionen oder Schwellenwertfunktionen (Abb. 2.1.2, Abb. 2.2.1b).

Die Adaptationsfunktion verändert die Gewichtswerte und damit den Zustand der Verarbeitungseinheit. Entsprechend dem funktionalen Zusammenhang zwischen $\underline{w}$, $\underline{x}$, a, y und der Änderung $\Delta\underline{w}$, lassen sich Adaptationsfunktionen in verschiedene Klassen einteilen, wie z. B. prinzipiell in überwachte oder nicht überwachte Lernregeln. Überwachte Lernregeln erfordern entweder eine "Lehrereingabe", die die geforderte Ausgabe einer Verarbeitungseinheit direkt vorgibt oder eine Bewertung, die die Richtigkeit der Ausgabe einer Verarbeitungseinheit beurteilt. Eine überwachte Lernregel mit Lehrereingabe ist z.B. die in Abschnitt 2.1 erwähnte Perzeptron-Lernregel.

Die interessanteren Lernregeln sind die nicht überwachten oder selbstorganisierenden Regeln. Sie befähigen ein Netzwerk von Modellneuronen zur selbständigen Anpassung an eine gegebene Aufgabenstellung, indem das System eine zeitlang seine Umwelt "beobachten" muß. "Beobachten" meint hier die Auswertung von Eingaben aus der Umwelt. Ein Beispiel für ein Netzwerkmodell mit nicht überwachter Lernregel sind die selbstorganisierenden Karten von Kohonen [25].

Eine besondere Rolle spielen im Hinblick auf eine schaltungstechnische Realisierung die *lokalen Regeln*, bei denen es nur auf Größen ankommt, die lokal an jeder Verarbeitungseinheit zur Verfügung stehen. Eine einfache, aber sehr wirkungsvolle lokale Lernregel ist die *Hebb-Regel*, die von Hebb (vgl. 2.1) als grundlegend für das Lernen im Gehirn vorgeschlagen wurde:

$$\Delta w_i \sim x_i \cdot y \qquad (2.2.3)$$

Entsprechend dieser Regel werden nur dann Gewichtswerte erhöht, wenn die Verarbeitungseinheit eine Ausgabe erzeugt (d.h. aktiv ist) und gleichzeitig die Eingabe x_i entsprechend hoch ist. Viele der bekannten Lernregeln basieren auf diesem Prinzip.

2.2.2 Definition eines assoziativen Netzwerkes

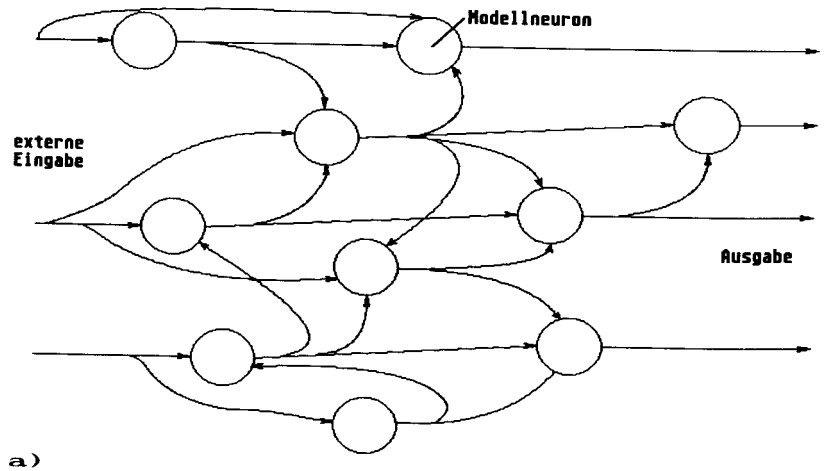
Ein assoziatives Netzwerk (Abb. 2.2.2) besteht aus einer möglichst großen Anzahl (≥ 1000) von Verarbeitungseinheiten. Alle Verarbeitungseinheiten im Netz arbeiten parallel und kommunizieren untereinander sowie mit der Außenwelt über direkte Verbindungsleitungen. Die Verbindungsleitungen werden in externe, von der Außenwelt und interne, von den anderen Einheiten im Netz kommend unterschieden.

Definition 2: Ein ***assoziatives Netzwerk*** ist ein Tupel $AN=(\mathfrak{V}, \mathfrak{E}, \mathfrak{v}, \epsilon, \mathfrak{z})$, mit $\mathfrak{V}$ einer endlichen Menge von Verarbeitungseinheiten, $\mathfrak{E}$ einer endlichen Menge von externen Eingabequellen, und drei Abbildungen $\mathfrak{v} : \mathfrak{V}\times\mathfrak{V} \dashrightarrow \{0,1\}$, $\epsilon : \mathfrak{E}\times\mathfrak{V} \dashrightarrow \{0,1\}$ und $\mathfrak{z} : \mathfrak{V} \dashrightarrow \{0,1\}$. Für alle $V_i, V_j \in \mathfrak{V}$ bestimmt $\mathfrak{v}(V_i,V_j)$, ob eine Verbindung von V_i nach V_j besteht. Für alle $e_j \in \mathfrak{E}$ bestimmt die Abbildung $\epsilon(e_j,V_i)$, ob die Verarbeitungseinheit V_i mit der externen Eingabequelle e_j verbunden ist. Die Abbildung $\mathfrak{z}(V_i)$ bestimmt, ob die Ausgabe von V_j eine externe Ausgabe an die Umwelt ist.

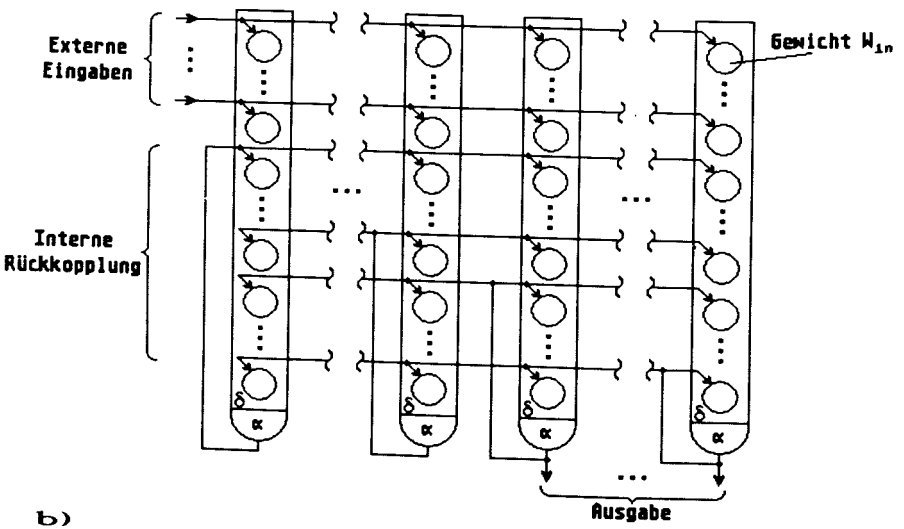
Ein assoziatives Netzwerk besteht aus elementaren Verarbeitungseinheiten und einer Netzwerktopologie. Das zeitliche Verhalten des Netzwerkes wird von der Arbeitsweise der Verarbeitungseinheiten bestimmt. Im zeitdiskreten Fall unterscheidet man synchrone, d.h. ein globaler Systemtakt bestimmt die Zeitpunkte, an denen die Einheiten ihre Ausgabe neu berechnen und asynchrone Netzwerke, bei denen die Einheiten an zufälligen Zeitpunkten eine Ausgabe neu bestimmen. Die Ausgabe $\mathfrak{Z}$ des gesamten Netzwerkes ergibt sich aus einer von der Abbildung $\mathfrak{z}$ festgelegten Teilmenge der Ausgabewerte y_j. In der Literatur sind häufig folgende Bezeichnungen zu finden [27]:

Definition 3: Eine Verarbeitungseinheit $V\in\mathfrak{V}$ eines assoziativen Netzwerkes $AN=(\mathfrak{V},\mathfrak{E},\mathfrak{v},\epsilon,\mathfrak{z})$ mit $|\mathfrak{E}|=m$ externen Eingabequellen heißt:

1) **Eingabeeinheit** : $\exists\ j \in \{1, \ldots, m\}$ mit $\epsilon(e_j,V)=1$;
2) **Versteckte Einheit** : $\forall\ j \in \{1, \ldots, m\}$ gilt $\epsilon(e_j,V)=0$ und $\mathfrak{z}(V) = 0$;
3) **Ausgabeeinheit** : $\mathfrak{z}(V) = 1$;



a)



b)

Abb. 2.2.2: a) Assoziatives Netzwerk;
b) Reguläre Anordnung des Netzwerkes aus a).

Die Netzwerktopologie wird durch die von den Abbildungen ϵ, υ festgelegten Matrizen charakterisiert und kann, wie in Abb. 2.2.2a, beliebige Verbindungsstrukturen (z. B. Mehrebenen- oder Rückkopplungsstrukturen) annehmen. Neben der Wahl der geeigneten Verbindungsstruktur ist die Einstellung der Verbindungsgewichte w_{ij} festzulegen, die entweder explizit vorgegeben werden oder von einer Adaptationsregel bestimmt werden können. Diese Änderung ist im allgemeinen ein viel langsamerer Prozeß als die Aktivitätsänderung, das führt häufig zu einer Trennung in eine *Lernphase* und eine *Ausführungsphase*. Dem Netzwerk werden zunächst in der Lernphase die zu lernenden Muster eingegeben und die Verbindungsstruktur mittels der Adaptationsregel angepaßt. Eine Änderung der Kopplungsstärken bedeutet eine Änderung des Netzwerkverhaltens. Die Lernphase ist beendet, wenn das Netzwerk das geforderte Verhalten erlernt hat, bzw. sein Verhalten nicht mehr ändert. In der Ausführungsphase werden keine Änderungen der Gewichte mehr durchgeführt.

In diesen Spezifikationsrahmen für assoziative Netzwerke lassen sich viele der bisher bekannten Modelle durch eine entsprechende Wahl der Verarbeitungseinheit und der initialen Verbindungstopologie einordnen. Einschränkungen ergeben sich erstens dadurch, daß assoziative Netzwerke gemäß Definition 2 nur mit Verarbeitungseinheiten vom selben Typ aufgebaut sind. Zweitens wird die Wahl der Dynamikfunktionen durch die Vorgabe der Wertemengen eingeschränkt. Beide Einschränkungen lassen sich einfach beheben: die Erste, indem für $\mathfrak{V}$ eine endliche Menge von endlichen Mengen von Verarbeitungseinheiten gewählt wird; die Zweite, indem beliebige Wertemengen zugelassen werden. Die Erweiterungen wirken sich aber negativ auf die Handhabbarkeit der Modellbeschreibung im Hinblick auf eine schaltungstechnische und programmtechnische Implementierung aus.

Für den Systementwickler integierter Schaltungen ist der ausgesprochen modulare und reguläre Aufbau der Netzwerkbescheibung wichtig. Es gibt nur einen Grundbaustein, der selbst wieder regulär und nur aus wenigen unterschiedlichen Baugruppen aufgebaut ist. Auch wenn die drei Funktionsgleichungen kompliziert ausfallen sollten, kann der Entwurf dieser Einheit vorerst ungeachtet der Netzwerkstruktur ausgeführt werden. Zudem lassen sich alle Netzwerktopologien in eine reguläre Matrixstruktur umwandeln (Abb. 2.2.2b). Dies ist sicherlich für große Netzwerke uneffektiv, denn es entspricht einer vollständigen Vernetzung aller Elemente eines Netzes. Für kleinere Teilnetze, die zu einem assoziativen Netzwerkmodul zusammengefaßt sind und sich zu komplexeren Systemen verschalten lassen, ist dieses Vorgehen angebracht.

2.3 Assoziative Informationsverarbeitung

Es stellt sich die Frage, für welche Aufgabenstellungen assoziative Netzwerke geeignet sind. In der Datenverarbeitung ergeben sich immer wieder Forderungen nach Systemlösungen, die große Datenmengen effizient speichern, einzelne Daten schnell wiederfinden sowie unerwünschte Nebeneffekte wie Rauschen und Unsicherheit unterdrücken können. Die Organisation und die effektive Nutzung von Daten ist eine wesentliche Voraussetzung für die Leistungsfähigkeit zukünftiger Informationssysteme.

Heutige Speichertechniken können große Datenmengen aufnehmen, aber nur schwer zugreifbar machen, wenn kein offensichtliches Ordnungsprinzip bekannt ist. Insbesondere ist es schwierig, das Datum herauszufinden, das unter allen gespeicherten Daten das geeignetste in bezug auf eine gestörte oder unvollständige Eingabe ist. Gerade in diesem Bereich sind selbst "niedere" Nervensysteme, beispielsweise von Insekten, allen heutigen Rechnerarchitekturen deutlich überlegen.

2.3.1 Aufgabenstellung

Formal läßt sich die Aufgabenstellung als eine Zuordnung von abstrakten Mustern beschreiben, wobei die Muster als Vektoren aufgefaßt werden. Die grundlegende Funktion eines assoziativen Netzwerkes ist eine Abbildung zwischen zwei endlichen Mustersätzen I und O (Abb. 2.3.1):

$$I = \{ \underline{x}^i : \underline{x}^i \in X^m,\ i = 1, \ldots, z \}$$
$$O = \{ \underline{y}^i : \underline{y}^i \in Y^n,\ i = 1, \ldots, z \}$$

Anschaulich kann man die Musterpaare $(\underline{x}^i, \underline{y}^i)$ als Frage und Antwort oder Reiz und Reaktion auffassen. Damit wird auch die Begriffsbildung *assoziatives* Netzwerk verständlich, denn umgangssprachlich steht *Assoziation* für Verknüpfung oder Verbindung. Insbesondere steht sie für eine Verknüpfung von Gedanken, so daß der eine Gedanke den anderen ins Gedächtnis zurückruft. Bei der Anwendung assoziativer Netzwerke unterscheidet man zwischen der Hetero- und der Autoassoziation:

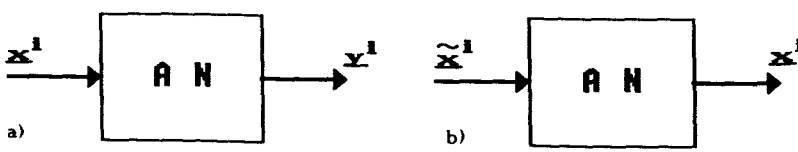


Abb. 2.3.1: Grundlegende Aufgabenstellung assoziativer Netze (AN):
a) Musterabbildung (Heteroassoziation);
b) Mustervervollständigung (Autoassoziation).

Definition 4: a) **Heteroassoziation** (Musterabbildung):
Das assoziative Netzwerk AN speichert z Musterpaare $(\underline{x}^i,\underline{y}^i)$, i=1,...,z. Bei Eingabe eines Musters $\underline{x}^i$, bzw. eines genügend großen Teils von $\underline{x}^i$ bezüglich einer gegebenen Metrik, gibt AN das dazugehörige Muster $\underline{y}^i$ aus.

b) **Autoassoziation** (Mustervervollständigung):
Das assoziative Netzwerk AN speichert z Muster $\underline{x}^i$, i=1,...,z. Bei Eingabe eines Musters $\tilde{\underline{x}}^i$ gibt AN das Muster $\underline{x}^i$ aus, das zu $\tilde{\underline{x}}^i$ am ähnlichsten bezüglich einer gegebenen Metrik ist.

Die Autoassoziation kann als Sonderfall der Heteroassoziation angesehen werden ($\underline{x}^i=\underline{y}^i$), bei der Bruchstücke eines gespeicherten Musters dazu dienen, das vollständige Muster abzurufen. Ein anschauliches Beispiel für die Autoassoziation von gespeicherten Gesichtern (Kohonen [25]) zeigt Abbildung 2.3.2. In einem assoziativen Netzwerk sind 100 Prototypen derartiger Gesichter eingespeichert worden. Jedes Bild hat 54x56=3024 Bildpunkte mit jeweils acht Graustufen. Das assoziative Netzwerk kann sowohl ein unvollständiges (Abb. 2.3.2b oben) als auch ein gestörtes (Abb. 2.3.2b unten) Eingabemuster korrekt ergänzen.

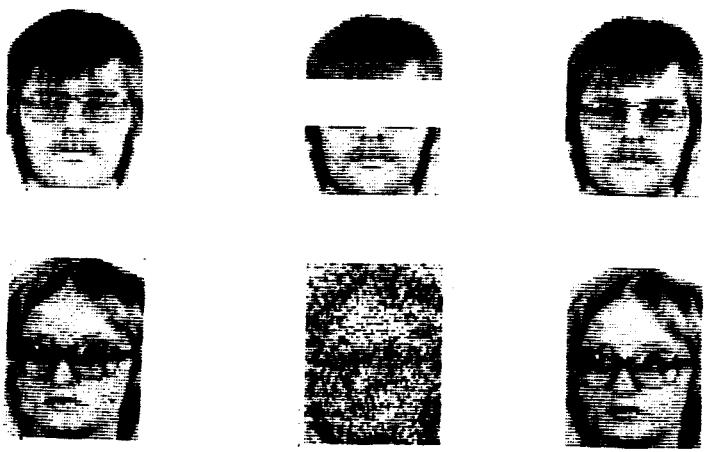


Abb. 2.3.2: Demonstration der Mustervervollständigung (Kohonen [25]): a) gespeicherte Prototypen; b) unvollständige bzw. gestörte Eingabe; c) assoziierte Muster.

Die Musterabbildung bzw. -vervollständigung kann auch mit inhaltsorientierten Zugriffsverfahren der Digital- oder Softwaretechnik realisiert werden. Um die Eigenschaften der inhaltsadressierten Speicher oder der Hash-Kodierung mit assoziativen Netzwerken vergleichen zu können, werden im weiteren nur Anwendungen mit binären Ein- und Ausgabemustern ($X=Y=\{0,1\}$) betrachtet.

2.3.2 Die Pseudoinversen-Technik

Das assoziative Modell von Kohonen erreicht seine Fähigkeiten (Abb. 2.3.2) durch die Verwendung der Pseudoinversen-Technik [25]. Kohonen faßt die Musterzuordnung als lineare Abbildung auf, für die das Abbildungsproblem als Matrix-Vektor-Multiplikation formuliert werden kann:

$$\underline{\mathbf{y}}^i = \mathbf{W} \cdot \underline{\mathbf{x}}^i \text{ , für alle } i = 1, \ldots, z. \qquad (2.3.1)$$

Faßt man die Vektoren $\underline{\mathbf{x}}^i$ und $\underline{\mathbf{y}}^i$ als Matrixspalten der entsprechenden Matrizen $\mathbf{X}$ und $\mathbf{Y}$ auf, so erhält man:

$$\mathbf{Y} = \mathbf{W} \cdot \mathbf{X} \qquad (2.3.2)$$

Sind die Vektoren $\underline{\mathbf{x}}^i$ linear unabhängig, dann kann die Matrix $\mathbf{X}$ invertiert und die Gleichung gelöst werden:

$$\mathbf{W} = \mathbf{Y} \cdot \mathbf{X}^{-1} \qquad (2.3.3)$$

Es können mit $\mathbf{W}$ alle $z \leq m$ Musterpaare exakt bestimmt werden. Sind die Eingabevektoren linear abhängig, dann erhält man eine im Sinne der mittleren quadratischen Abweichung optimale Lösung mit der Pseudoinversen $\mathbf{X}^+$ [25]:

$$\mathbf{W} = \mathbf{Y} \cdot \mathbf{X}^+ \qquad (2.3.4)$$

Eine Pseudoinverse $\mathbf{X}^+$ existiert zu jeder Matrix $\mathbf{X}$. Für linear unabhängige Eingabevektoren entspricht $\mathbf{X}^+$ der inversen Matrix $\mathbf{X}^{-1}$. Es gibt mehrere Verfahren zur Berechnung der Pseudoinversen [33], die aber keine "lokalen" Adaptationsregeln im Sinne von Abschnitt 2.2.1 darstellen. Eine approximative Lösung der Pseudoinversen bietet die folgende lokale, überwachte Adaptationsregel [25]:

$$\mathbf{W}^h = \mathbf{W}^{h-1} + c \cdot \left(\underline{\mathbf{y}}^i - \mathbf{W}^{h-1} \cdot \underline{\mathbf{x}}^i \right) \underline{\mathbf{x}}^{i\,T} \qquad (2.3.5)$$
$$\mathbf{W}^0 = \underline{\mathbf{0}} \text{ , Nullmatrix, } i \in \{1, \ldots, z\}, \; h \in \mathbb{N}.$$

Die zu lernenden Vektorpaare müssen nacheinander und in der Regel mehrfach zur Berechnung der Matrix $\mathbf{W}$ herangezogen werden. Die Lern-

regel ist vergleichbar mit der *Perzeptron-Regel* (2.1.1) und wird allgemein mit *Delta-Regel* oder *Fehler-Korrektur-Regel* (error-correction rule [27]) bezeichnet.

Die Matrix **W** speichert die Muster nicht lokal in einer bestimmten Spalte oder Zeile, sondern nur deren Korrelationen untereinander, *verteilt* über alle Matrixkomponenten. Jede Matrixkomponente w_{ij} kann zur Speicherung von mehreren Mustern beitragen, während bei einer *lokalen* Speicherung jede Komponente nur zur Speicherung eines Musters benutzt wird.

Die Pseudoinversen-Technik wird im allgemeinen für kontinuierliche Ein- und Ausgaben ($X=Y=\mathbb{R}$) angewendet. Für linear unabhängige Eingabemuster ist die Musterabbildung bzw. -vervollständigung fehlerfrei, so daß maximal m Musterpaare ($\underline{x}^i \in X^m$) fehlerfrei gespeichert werden können. Für linear abhängige Muster ist die Zuordnung bezüglich der mittleren quadratischen Abweichung $\| \underline{y}^i - \mathbf{W} \cdot \underline{x}^i \|^2$ minimal. Der Berechnungsaufwand für die Matrix **W** ist hoch, und die Matrixkomponenten w_{ij} können beliebige Werte annehmen. Im Hinblick auf eine schaltungstechnische Realisierung ist es sicherlich sinnvoll, vorerst von einfacheren Modellen auszugehen.

2.3.3 Die assoziative Matrix

Die einfachste und grundlegendste Struktur, die sich einer verteilten Speicherung bedient, ist die *assoziative Matrix* [34,35]:

Definition 5: Eine **assoziative Matrix** ist ein Netzwerk $AM=(\mathfrak{B},\mathfrak{E},\mathfrak{v},\mathfrak{e},\mathfrak{z})$ mit n Verarbeitungseinheiten $AS=(m,X,Y,A,W,\delta,\alpha,\lambda)$:

$$X = Y = W = \{0,1\};\quad A = \mathbb{N};$$

$$\delta(\underline{x}(t), \underline{w}) = \underline{x}(t)^T \cdot \underline{w} =: a(t);$$

$$\alpha(a(t)) = \begin{cases} 1, & \text{wenn } a(t) \geq Th,\ Th \in \mathbb{N}; \\ 0, & \text{sonst.} \end{cases}$$

$$\lambda(\underline{x}^h, \underline{w}^{h-1}, y^h) = \underline{w}^h, \text{ mit } w_i^h = w_i^{h-1} \vee (x_i^h \wedge y^h),$$
$$\forall i \in \{1, \ldots, m\},\quad h=1, \ldots, z;$$
$$\vee \hat{=} \text{ logisches "ODER"}$$
$$\wedge \hat{=} \text{ logisches "UND"}$$
$$\underline{w}^0 \hat{=} \text{ Nullvektor; } \underline{w} := \underline{w}^z$$

$$\mathfrak{B} = \{ AS_j : j = 1, \ldots, n \};\ \mathfrak{E} = \{ e_i : i = 1, \ldots, m \};$$
$$\mathfrak{v}(AS_k, AS_l) = 0,\ \forall\, k,l \in \{1, \ldots, n\};$$
$$\mathfrak{e}(e_i, AS_j) = 1,\ \forall\, j \in \{1, \ldots, n\} \text{ und } i \in \{1, \ldots, m\};$$
$$\mathfrak{z}(AS_j) = 1,\ \forall\, j \in \{1, \ldots, n\}.$$

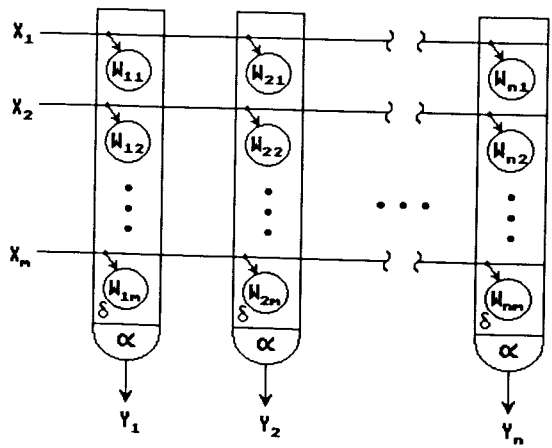


Abb. 2.3.3: Die assoziative Matrix.

Die assoziative Matrix besteht aus n nicht miteinander gekoppelten Verarbeitungseinheiten, die alle dieselbe Eingabe erhalten. Es gibt keine versteckten Verarbeitungseinheiten, alle sind sowohl Eingabe- als auch Ausgabeeinheit. Bei Eingabe eines bestimmten Musters berechnen alle Verbindungseinheiten parallel und unabhängig voneinander den Aktivierungswert a_j. Der nachfolgende Schwellenwertvergleich liefert für jede Verarbeitungseinheit eine binäre Ausgabe, wobei in diesem Modell für alle Verarbeitungseinheiten ein einheitlicher Schwellenwert Th angenommen wird. Der Schwellenwert wird von außen vorgegeben und ist kleiner oder gleich der Anzahl der aktivierten Eingangsleitungen (x_i=1).

Die Zuordnungen werden in der *Lernphase* in die Matrix, deren Spalten die Gewichte der Verarbeitungseinheiten bilden, eingespeichert. Ausgehend von einer gelöschten Matrix $\mathbf{W}_0$ -alle Gewichte w_{ij} haben den binären Wert Null- werden die Gewichte iterativ durch Anlegen der z Musterpaare an die Matrix bestimmt:

$$\mathbf{W}^h = \mathbf{W}^{h-1} \vee \left(\underline{x}^h * (\underline{y}^h)^T \right) , \quad h = 1, \ldots, z \qquad (2.3.6)$$

Die Muster $\underline{x}^h$ und $\underline{y}^h$ werden als Spaltenvektoren aufgefaßt, wobei das Ausgabemuster $\underline{y}^h$ in einen Zeilenvektor transponiert wird ($\underline{y}^T$). Die '*'-Operation auf diesen beiden Vektoren definiert die Matrix **M** mit $m_{kl} = x_l^h \cdot y_k^h$, die die gleiche Dimension hat wie die Verbindungsmatrix **W** (m Zeilen, n Spalten). Die 'v'-Operation auf diesen beiden Matrizen ist als logische 'ODER'-Verknüpfung der jeweiligen Matrixelemente definiert:

$$w_{kl}^h = w_{kl}^{h-1} \vee m_{kl} \qquad (2.3.7)$$

Entsprechend dieser Lernregel überlagern sich die einzelnen Musterpaare in der Gewichtsmatrix, d.h. die Matrix füllt sich mehr und mehr mit Einsen. Ein einfaches Beispiel für die Heteroassoziation zeigt Abb. 2.3.4. Die beiden eingespeicherten Ausgabemuster ($\underline{y}^1$, $\underline{y}^2$) lassen sich selbst beim Fortlassen einer Eins im Eingabemuster vollständig auslesen. Um eine fehlerfreie Abbildung zu gewährleisten, dürfen jedoch nicht beliebig viele Muster eingegeben werden. Eine quantitative Untersuchung erfolgt im Kapitel 3.

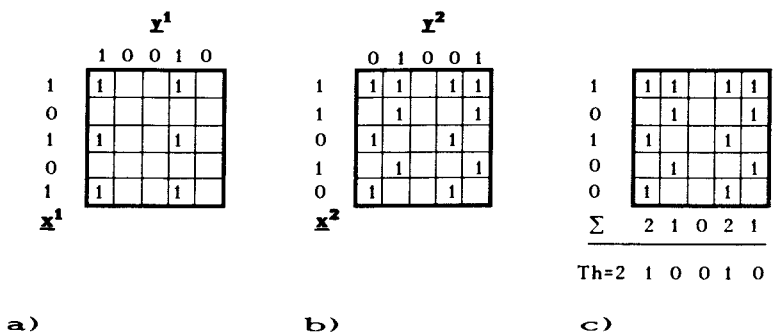


Abb. 2.3.4: Beispiel für das Einspeichern von binären Mustern in eine assoziative Matrix (a,b) und für eine Assoziation mit einem unvollständigen Eingabemuster (c).

2.3.4 Das Hopfield-Netz

Ein sehr bekanntes und intensiv untersuchtes rückgekoppeltes Netzwerk ist das Hopfield-Netz [23]:

Definition 6: Ein **Hopfield-Netz** ist ein Netzwerk $HN = (\mathfrak{V}, \mathfrak{E}, \mathfrak{v}, \mathfrak{e}, \mathfrak{z})$ mit n Verarbeitungseinheiten $HOP=(n,X,Y,A,W,\delta,\alpha,\lambda)$:

$$X = Y = \{-1,+1\};\quad W = A = \mathbb{Z}$$

$$\underline{y}(t=0) = \underline{x} \quad \text{mit set=1;}$$

$$\underline{y}(t>0) = \alpha(a(t-1)) \quad \text{mit set=0;}$$

$$\delta(\underline{y}(t),\underline{w}) = \underline{y}(t)^T \cdot \underline{w} =: a(t);$$

$$\alpha(a(t)) = \begin{cases} +1, \text{ wenn } a > 0 \text{ und select=1} \\ -1, \text{ wenn } a < 0 \text{ und select=1} \\ y(t-1) \text{ sonst.} \end{cases}$$

$$\text{select} \in \{0,1\}, \text{ eine diskrete Zufallsvariable;}$$

$$\lambda(\underline{x}^h, \underline{w}^{h-1}, y^h) = \underline{w}^{h-1} + y^h \cdot \underline{x}^h ; \quad \forall\, h \in \{1, \ldots, z\};$$
$$\underline{w}^0 \equiv \text{Nullvektor};\ \underline{w} = \underline{w}^z;$$

$$\mathfrak{V} = \{ HOP_j : j = 1, \ldots, n \};\ \mathfrak{E} = \{ e_i : i = 1, \ldots, n \};$$

$$\mathfrak{v}(HOP_k, HOP_j) = \begin{cases} 1, \forall\, k,j \in \{1, \ldots, n\} \text{ und } k \neq j \\ 0, \text{ für } k=j \end{cases}$$

$$\mathfrak{z}(HOP_j) = 1, \ \forall\, j \in \{1, \ldots, n\}$$

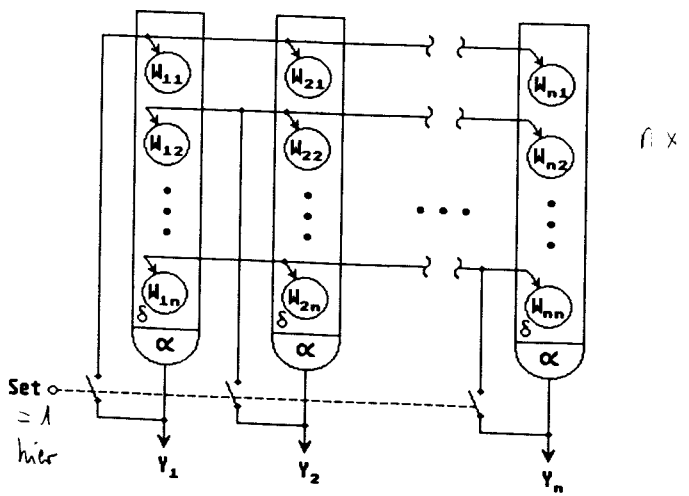


Abb. 2.3.5: Das Hopfield-Netz.

Das Hopfield–Modell basiert ebenfalls auf dem Schwellenneuron (vgl. 2.1), wobei hier der Schwellenwert Th=0 festgelegt ist und die Ein- und Ausgabewerte +1/-1 anstatt 1/0 sind. Beide Fälle lassen sich aber ineinander überführen, so daß diese Auswahl der Ein-/Ausgabesignale und der Schwelle für die Berechnungsmächtigkeit einer Verarbeitungseinheit unbedeutend ist [36].

Alle Verarbeitungseinheiten sind vollständig untereinander vernetzt, nur die Rückkopplung auf sich selbst ist nicht erlaubt. Die Verarbeitungseinheiten arbeiten asynchron, modelliert durch die Zufallsvariable *select*. Zu jedem Zeitpunkt t wird zufällig eine Teilmenge von Elementen ausgewählt, diese dürfen ihren Ausgabewert gemäß der Ausgabefunktion ändern. Die Ausgaben aller anderen Elemente bleiben konstant. In der Regel werden die Verarbeitungseinheiten mit der gleichen Wahrscheinlichkeit und im Mittel nur eine zu jedem diskreten Zeitpunkt ausgewählt.

Für einen Assoziationsvorgang werden die Ausgänge der n Verarbeitungseinheiten mit einem Teilmuster eines gespeicherten Musters belegt. Die Rückkopplung wird während dieser Initialisierung aufgetrennt (Set=1, Abb. 2.3.5) und zur Assoziation wieder geschlossen (Set=0). Jeder Anfangszustand *relaxiert* in einen stabilen Endzustand, der dem Ausgabemuster entspricht. Dieses Modell arbeitet somit autoassoziativ.

Ein wesentlicher Gesichtspunkt der Arbeiten von Hopfield ist die Einführung einer Energiebetrachtung zur Beschreibung der Netzwerkdynamik. Demnach wird jedem Netzwerkzustand eine bestimmte (potentielle) Energie zugeordnet [23]:

$$E = -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} y_i \cdot y_j \cdot w_{ij} \qquad (2.3.8)$$

Für ein symmetrisches Netzwerk, d.h. $w_{ij} = w_{ji}$, mit $w_{ii} = 0$ ergibt sich bei einer Änderung eines Ausganges y_i einer Verarbeitungseinheit eine Energieänderung:

$$\Delta E = -\Delta y_i \sum_{j=1}^{n} y_j \cdot w_{ji} \qquad (2.3.9)$$

Jede Änderung von E ist somit negativ, die Energie nimmt also stetig ab. Sie ist jedoch daran gebunden, einen endlichen Wert anzunehmen. Daraus folgt, daß das System mit der Zeit in einen stabilen Zustand laufen muß, in dem sich die Energie nicht mehr ändert. Das System befindet sich dann in einem Energieminimum. Allerdings ist nicht gewährleistet, daß der Endzustand des Systems in einem Energieminimum auch die beste Lösung bezüglich des Eingabemusters ist. Die beste Lösung ist zum Beispiel die, die den kleinsten euklidischen Abstand zu der geforderten

Lösung hat. Die dem Gradienten folgende Energieabnahme des Systemzustandes kann in einem lokalen Minimum enden und so nur zu einer suboptimalen Lösung gelangen (Abb. 2.3.6).

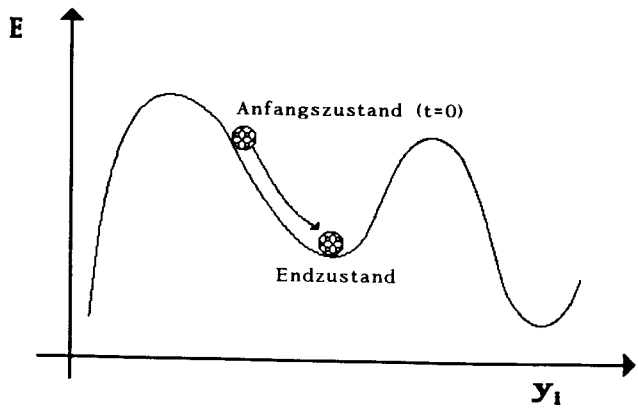


Abb. 2.3.6: Schematisches Bild der Systemenergie E als Funktion einer Koordinate im mehrdimensionalen Zustandsraum.

Voraussetzung für die Konvergenz dieses nichtlinearen Rückkopplungsmechanismusses ist eine symmetrische Verbindungsmatrix mit einer Nulldiagonalen ($w_{ii}=0$). Die Verbindungsmatrix wird in der Lernphase durch die angegebene Lernregel berechnet:

$$\mathbf{W}^h = \mathbf{W}^{h-1} + \overset{\times}{\mathbf{x}}^h \cdot (\mathbf{x}^h)^T, \quad h = 1, \ldots, z \qquad (2.3.10)$$

Im Gegensatz zur assoziativen Matrix können hier die Matrixgewichte ganzzahlige Werte ($-z \leq w_{ij} \leq +z$) annehmen. In Abb. 2.3.7 ist ein einfaches Beispiel für das Einspeichern von 2 Mustern und der Assoziation eines gestörten Eingabemusters aufgeführt. Das Einschwingen auf den stabilen Endzustand beschränkt sich hier auf einen Schritt.

In der Literatur gibt es eine Vielzahl von Variationen zu der hier gewählten Definition eines Hopfield-Netzes. Die Zustandsänderung der Verarbeitungseinheiten kann z.B. auch synchron erfolgen, indem entweder alle Verarbeitungseinheiten im Netz gleichzeitig (parallel) oder einzeln nach einer festgelegten Reihenfolge (seriell) einen neuen Ausgabewert erzeugen. Häufig werden die Verbindungsgewichte nach der Lernphase auf

die Werte {+1,0,-1} beschränkt, das z.B. hinsichtlich einer schaltungstechnischen Realisierung vorteilhaft ist. Die Auswirkungen dieser Variationen auf die Systemeigenschaften sind Gegenstand theoretischer Untersuchungen [37].

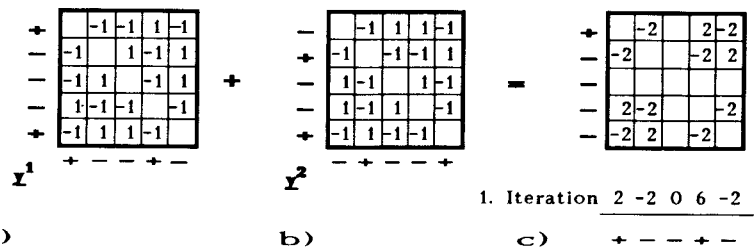


Abb. 2.3.7: Beispiel für das Einspeichern von zwei binären Mustern in ein Hopfield-Netz (a,b) und für eine Assoziation mit einem unvollständigen Eingabemuster (c).

Die drei vorgestellten assoziativen Modelle verfügen über eine einfache und reguläre Matrixstruktur. Die Matrixkomponenten entsprechen den Verbindungsgewichten von linear angeordneten Verarbeitungseinheiten eines assoziativen Netzwerkes. Die Aktivierungsfunktion der Verarbeitungseinheiten ist die gewichtete Summe der Eingabewerte.(2.2.1). Unterschiede bei den Verarbeitungseinheiten ergeben sich hinsichtlich der berechneten Ausgabe- und Adaptationsfunktion sowie bezüglich des Wertebereiches der Verbindungsgewichte. Alle drei Modelle bedienen sich einer verteilten Speicherung und lassen sich für eine fehlertolerante Zuordnung bzw. Vervollständigung von binären Mustern verwenden.

2.4 Anwendungsbeispiele

Ein anschauliches Beispiel für die Mustervervollständigung ist die bereits erwähnte autoassoziative Speicherung von Grauwertbildern. Verwendet man binäre Bildpunkte, dann können derartige Bilder auch in eine assoziative Matrix und ein Hopfield-Netz eingespeichert werden [38]. Die Simulation derart großer Matrizen ist auf einem Rechner mit Mehrprogrammbetrieb sehr zeitaufwendig und benötigt viel Speicherplatz. Zur Speicherung der 3024x3024≈9Millionen Matrixkomponenten benötigt man immerhin 36 MegaByte Speicherplatz, bei einer Wortbreite von 32 Bit pro Komponente. Für eine Mustervervollständigung müssen ca. 9 Millionen

Multiplikationen und Additionen ausgeführt werden. Zur Untersuchung der Eigenschaften von verschiedenen Modellen sind derart komplexe Muster sehr unhandlich.

Häufig verwendet man daher einfachere Muster, wenngleich diese für eine Anwendung unattraktiv sind. Ein ganz einfaches Beispiel sind 7x5-Rasterbilder von ASCII-Zeichen (Abb. 2.4.1). In eine assoziative Matrix mit m=n=35 lassen sich z.B. bis zu 6 Musterpaare fehlerfrei heteroassoziativ einspeichern [32]. Ein Hopfield-Netz kann max. 5 Muster autoassoziativ [39] und mit der Pseudoinversen-Technik lassen sich bis zu 35 Musterpaare heteroassoziativ speichern [32]. Auf den ersten Blick erscheint die Pseudoinversen-Technik als die günstigere Lösung. Es bleibt aber zu beachten, daß die Matrixkomponenten reelle Zahlen sind, während die assoziative Matrix nur binäre Verbindungsgewichte hat (vgl. 3.3).

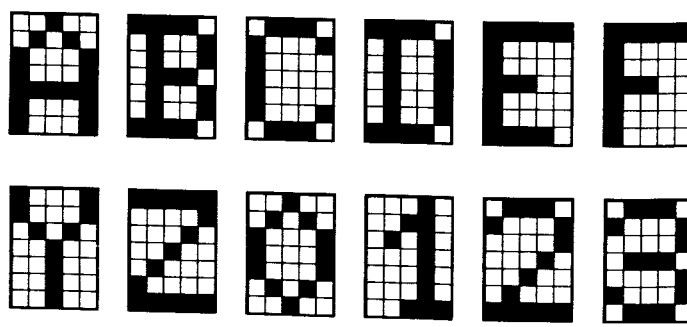


Abb. 2.4.1: Beispiele der verwendeten 7x5-Rasterbilder von ASCII-Zeichen zur assoziativen Speicherung.

Ein attraktiveres Anwendungsbeispiel für heteroassoziative Zuordnungen ist ein Telefonregister. In einer assoziativen Matrix mit m=1800 und n=400 sind z.B. 500 Musterpaare [Name,Telefonnummer] eingespeichert worden, von denen 85% fehlerfrei und der Rest mit einem Bitfehler wieder ausgelesen werden konnte [40]. Aufgrund der verwendeten Kodierung konnten diese Bitfehler eindeutig und mit geringem Aufwand korrigiert werden. Beispiele für die Fehlertoleranz gegenüber Eingabefehlern zeigt Tabelle 2.4.1. Die Ergebnisse können selbstverständlich nicht auf alle Musterpaare verallgemeinert werden. Die Fehlertoleranz hängt von den Musterpaaren selbst, dem Füllungsgrad der Matrix und von der verwendeten Kodierung ab (siehe Kapitel 3).

gespeicherte Musterpaare:	Stelzner, Peter		3592
	Schwarz, Christoph		2323
	Schwarz, Dittrich		2365
Eingabe:	Stelzner	Ausgabe:	3592
	Stelzn, P		3592
	Selzner, P		3592
	Schwarz		232365
	Schwarz, Di		2365

Tabelle 2.4.1: Beispiele für Musterzuordnungen einer assoziativen Matrix bei fehlerhafter Eingabe.

Eine interessante Anwendung für das Hopfield-Netz ist das Finden eines günstigen, nach Möglichkeit optimalen Reiseweges für einen Handlungsreisenden (travelling salesman problem) [41]. Für dieses Optimierungsproblem wächst der Rechenaufwand auf seriellen Rechnern exponentiell mit der Problemgröße (np-vollständiges Problem [42]).

Es gibt eine Vielzahl von weiteren Anwendungsbeispielen für assoziative Netzwerke [25,43-45]. Alle Beispiele verdeutlichen die grundlegenden qualitativen Eigenschaften von assoziativen Modellen. Für eine Bewertung der Leistungfähigkeit der verschiedenen Modelle und insbesondere auch für einen Vergleich mit konventionellen Lösungsmethoden sind quantitative Aussagen notwendig. Die wesentlichen Fragen beziehen sich auf die Anzahl der maximal speicherbaren Muster im Verhältnis zum Aufwand, auf die Fehlertoleranz und auf die Zugriffsgeschwindigkeit bzw. die Anzahl der notwendigen Rechenoperationen für eine Musterzuordnung. Im folgenden Kapitel werden Abschätzungen zur Speichereffektivität und Fehlertoleranz der drei hier vorgestellten Modelle angegeben.

3. Speichereigenschaften assoziativer Netzwerke

3.1 Speichereffektivität

Es stellt sich die Frage, wieviele und wie effektiv binäre Musterpaare in die oben beschriebenen assoziativen Netzwerkmodelle eingespeichert werden können. Für binäre Muster läßt sich nach Shannon [46] der Informationsgehalt für ein binäres Muster der Länge n mit $ld(2^n) = n$-Bit angeben, wenn alle 2^n möglichen Muster zugelassen werden und alle Muster mit der gleichen Wahrscheinlichkeit auftreten. Kann ein Netzwerk nun z Muster fehlerfrei zuordnen, so entspricht das einem Informationsgehalt von $z \cdot n$-Bits. Aufgrund der verteilten Speichertechnik der vorgestellten assoziativen Modelle sind die Musterzuordnungen nicht grundsätzlich fehlerfrei. Für diese Fälle muß man die Information, die zur Korrektur dieser Fehler notwendig ist, wieder abziehen. Bezeichnen wir mit I^l den Informationsgehalt des fehlerfreien Ausgabemusters $\underline{y}^l$ und mit I^l_{Fehler} die Information, die benötigt wird, um den Fehler in der Ausgabe zu korrigieren. Der *Informationsgehalt* eines assoziativen Netzwerkes mit z gespeicherten Musterpaaren $(\underline{x}^l, \underline{y}^l)$ ergibt sich zu:

$$I_H = \sum_{l=1}^{z} (I^l - I^l_{Fehler}) \qquad (3.1.1)$$

Die *Speichereffektivität* S_{eff} eines assoziativen Netzwerkes ergibt sich aus dem Quotienten des Informationsgehaltes und der aufgewendeten Speicherressourcen. Sei b die Anzahl der Bits, die zur Speicherung eines Gewichtswertes in einer mxn Matrix aufgewendet werden muß. Für die Speichereffektivität folgt:

$$S_{eff} = \frac{I_H}{n \cdot m \cdot b} \qquad (3.1.2)$$

3.1.1 Speichereffektivität einer assoziativen Matrix

In eine assoziative Matrix sollen z Musterpaare $(\underline{x}^l \epsilon X^m, \underline{y}^l \epsilon Y^n)$ heteroassoziativ eingespeichert werden. Die Eingabemuster $\underline{x}^l$ haben genau l und die Ausgabemuster $\underline{y}^l$ genau k Einsen. Ausgehend von einer leeren Gewichtsmatrix $\underline{W}_0$ werden für jedes Musterpaar gemäß der Adaptationsregel Glg. (2.3.7) neue Verbindungsgewichte in der Matrix gesetzt (Abb. 3.1.1). Die Anzahl der neuen Verbindungen ist für das erste Musterpaar $l \cdot k$. Sie nimmt aber für jedes weitere Musterpaar in dem Maße ab, wie die Wahrscheinlichkeit p_{on} ansteigt, mit der eine Verbindung bereits vorhanden ist.

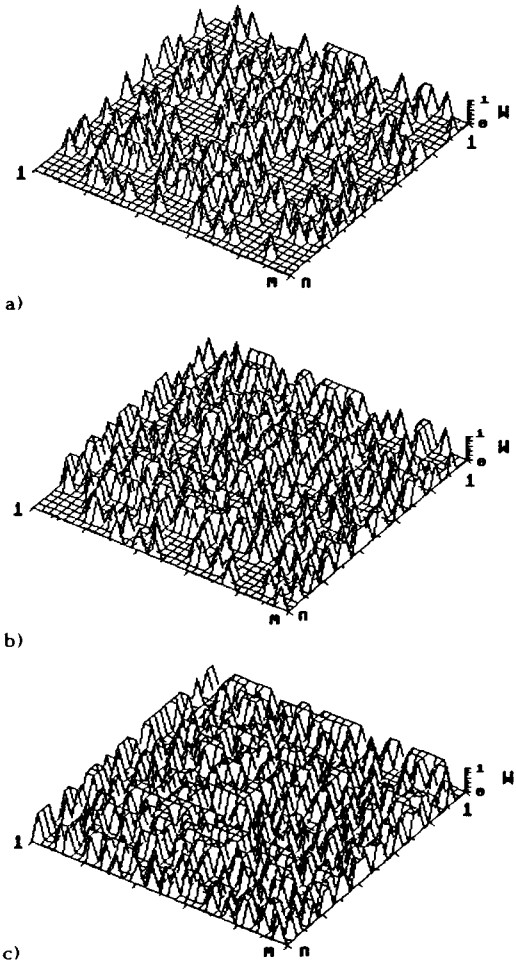


Abb. 3.1.1: Belegungsgrad einer heteroassoziativen Matrix (n=m=35) nach dem Einspeichern von z Musterpaaren mit (k=l=5):
a) z=10 , p_{on}=0.19; b) z=20, p_{on}=0.34; c) z=30, p_{on}=0.45;

Sind die Muster zufällig und unabhängig voneinander ausgewählt worden, so gilt nach dem Anlernen von z Musterpaaren:

$$p_{on} = \left(1 - \left(1 - \frac{k \cdot l}{n \cdot m}\right)^{z}\right) \qquad (3.1.3)$$

Abbildung 3.1.2 zeigt den Verlauf von p_{on} in Abhängigkeit von der Musteranzahl und den Parametern k,l. Die verteilte Speicherung führt zu Überlagerungen der einzelnen Muster innerhalb der Verbindungsmatrix, die ab einem bestimmten Belegungsgrad der Matrix zu fehlerhaften Musterzuordnungen führen. Für die assoziative Matrix äußern sich diese Fehler durch zusätzliche Einsen im Ausgabemuster. Ein Fehler in der Position f des Ausgabemusters $\underline{y}^t$ tritt dann auf, wenn *alle* l Einsen des Eingabemusters $\underline{x}^t$ auf ein aktiviertes Verbindungsgewicht der Verarbeitungseinheit f treffen. Die Wahrscheinlichkeit für das Eintreffen dieser Bedingung kann man grob mit p_{on}^{l} abschätzen. Dies ist insofern inkorrekt, als man von zufällig gesetzten Verbindungen in der Matrix ausgeht. Korrekterweise muß man die Wahrscheinlichkeit $\wp$ bestimmen, mit der für jede Eins im Eingabemuster $\underline{x}^t$ ein Musterpaar $(\underline{x}^s, \underline{y}^s)$ existiert, das an der entsprechenden Stelle im Eingabemuster und an der Stelle f im Ausgabemuster eine Eins hat (Palm [47]):

$$\wp = p\Big[\forall h \in \{1, \ldots, m\} \text{ mit } x_h^t = 1 \;\; \exists s \in \{1, \ldots, z\} : x_h^s = 1 \;\; \underline{\text{und}} \;\; y_f^s = 1 \;, \; f \in \{1, \ldots, n\}\Big]$$

$$= 1 + \sum_{h=1}^{l} (-1)^h \binom{l}{h} \left(1 - \frac{k}{n}(1 - T(m,l,h))\right)^{z-1}, \qquad (3.1.4)$$

$$\text{mit } T(a,b,c) = \prod_{i=0}^{c-1} \frac{a - b - i}{a - i} = T(a,c,b)$$

Der Erwartungswert für die Bitfehleranzahl N_t für ein Ausgabemuster ergibt sich zu:

$$E(N_t) = (n-k) \cdot \wp \qquad (3.1.5)$$

Mit Hilfe der Anzahl der fehlerhaften Einsen N_t der Ausgabemuster $\underline{y}^t$, die man exakt nur für gegebene Musterpaare bestimmen kann, läßt sich gemäß Glg. (3.1.1) der Informationsgehalt einer assoziativen Matrix berechnen:

$$I_H = \sum_{t=1}^{z} I_t = \sum_{t=1}^{z} \mathrm{ld}\binom{n}{k} - \mathrm{ld}\binom{N_t+k}{k}. \qquad (3.1.6)$$

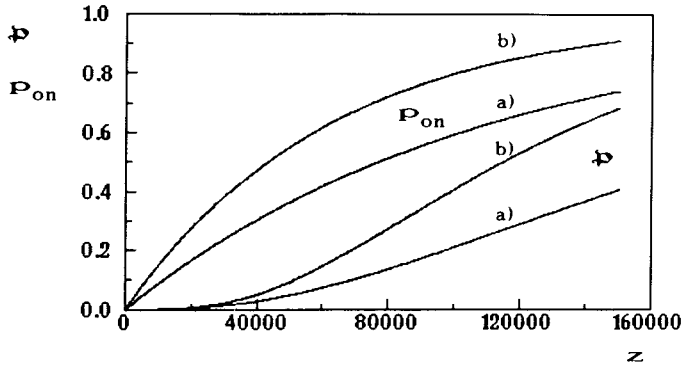


Abb. 3.1.2: Bit-Fehlerwahrscheinlichkeit p und Belegungswahrscheinlichkeit p_{on} der Verbindungsgewichtsmatrix (n,m=1000) in Abhängigkeit von der Musteranzahl z: a) l= k = 3, b) l = k = 4.

Mit diesem Ansatz hat Palm [47] den Erwartungswert für den Informationsgehalt einer heteroassoziativen Matrix für zufällige Muster abgeschätzt:

$$E(I_H) \geq -z \sum_{i=0}^{k-1} ld\left(\frac{(n-k)\cdot p + k - i}{n - i}\right) \qquad (3.1.7)$$

Mit den Gleichungen (3.1.4) und (3.1.7) läßt sich für eine gegebene Matrix der Erwartungswert $E(I_H)$ hinsichtlich der Parameter k, l und z maximieren. Für sehr große Matrizen (n,m ——> ∞) und k=O(log n) geht der Erwartungswert asymptotisch gegen den Grenzwert:

$$E(I_H) \longrightarrow \ln 2 \cdot n \cdot m \approx 0.69 \cdot n \cdot m \quad \text{(Bit)} \qquad (3.1.8)$$

Den Verlauf des Erwartungswertes $E(I_H)$ in Abhängigkeit von der Musteranzahl und der Anzahl von Einsen im Ein- und Ausgabemuster verdeutlicht Abb. 3.1.3. Es zeigt sich, daß man eine große Anzahl (z»n) von spärlich besetzten Mustern, d.h. k=O(logn) und l=O(logm) speichern kann. Für die Beispiele in Abb. 3.1.3 ergibt sich der maximale Erwartungswert $E(I_H)$ bei ca. 43000 Mustern mit k=l=4, bzw. ca. 76 000 für l=k=3. Die Speichereffektivität E(I)/(n·m) liegt bei ca. 0.68 bzw. ca. 0.69.

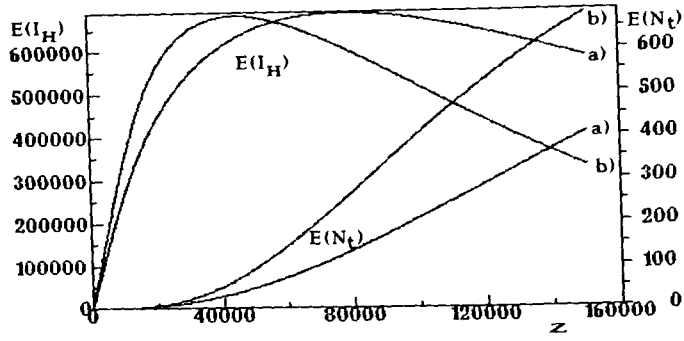


Abb. 3.1.3: Erwartungswert für den Informationsgehalt $E(I_H)$ in Bit und für die Anzahl der Bitfehler $E(N_t)$ in Abhängigkeit von der Musteranzahl z (m = n = 1000): a) l = k = 3, b) l = k = 4.

Es wird aber auch deutlich, daß im Maximum nicht mehr alle Muster fehlerfrei ausgelesen werden können. Verlangt man eine möglichst fehlerfreie Funktionsweise, so muß die Gesamtfehlerwahrscheinlichkeit für alle z Muster kleiner Eins sein:

$$1 > z \cdot (n-k) \cdot p \qquad (3.1.9)$$

Diese Ungleichung kann nur numerisch gelöst werden. Für eine Matrix mit n=m=1024 und Musterpaaren mit (l=20, k=3) können gemäß (3.1.9) maximal 8554 Musterpaare fehlerfrei zugeordnet werden. Das entspricht einem Informationsgehalt von ca. 223000 Bit und damit einer Speichereffektivität von 0.223. Für andere Werte von k und l ergeben sich kleinere Speichereffektivitätswerte (Tabelle 3.1.1).

k	3		6		9	
l	z	S_{eff}	z	S_{eff}	z	S_{eff}
10	6994	0.183	3789	0.182	2650	0.181
15	8318	0.217	4433	0.213	3070	0.209
20	8554	0.223	4521	0.217	3113	0.212
25	8343	0.218	4390	0.211	3019	0.206

Tabelle 3.1.1: Speichereffektivität und maximale Anzahl der einspeicherbaren Musterzuordnungen entsprechend Glg. (3.1.9) für verschiedene k,l-Werte (n=m=1024).

Beschränkt man sich auf die Forderung, daß $E(N_t)$ nur pro Muster kleiner Eins sein soll, so folgt:

$$1 > (n-k) \cdot \not{p} \tag{3.1.10}$$

Mit $(\not{p} - p_{on}^{1})/p_{on}^{1} \longrightarrow 0$ für $n,m \longrightarrow \infty$ folgt [47]:

$$1 > (n-k) \cdot p_{on}^{1} \tag{3.1.11}$$

$$\Leftrightarrow \qquad \left(\frac{1}{n-k}\right)^{1/l} > p_{on} = 1 - \left(1 - \frac{1 \cdot k}{n \cdot m}\right)^{z}$$

$$\Leftrightarrow \qquad \boxed{z < \frac{\log\left(1 - \left(\frac{1}{n-k}\right)^{1/l}\right)}{\log\left(1 - \left(\frac{1 \cdot k}{m \cdot n}\right)\right)}} \tag{3.1.12}$$

Unter Einhaltung der Bedingung (3.1.10) kann man etwa 23300 Musterzuordnungen mit l=10, k=3 und n=m=1024 speichern, bei denen die Wahrscheinlichkeit für einen Bitfehler kleiner Eins ist. Die Speichereffektivität beträgt in diesem Fall bereits 0.565. Weitere Beispiele finden sich in Tabelle 3.1.2.

Eine grobe Faustformel für die Anzahl der speicherbaren Musterzuordnungen unter der Bedingung (3.1.12) erhält man, wenn man den Nenner mit $\log(1-x) \approx -x$ und den Zähler mit $\ln 2$ abschätzt [34]:

$$z < \ln 2 \cdot \frac{m \cdot n}{1 \cdot k} \tag{3.1.13}$$

k	3		6		9	
l	z	S_{eff}	z	S_{eff}	z	S_{eff}
5	19664	0.477	9841	0.447	6566	0.427
10	23313	0.565	11664	0.53	7780	0.506
15	21936	0.532	10975	0.499	7321	0.476
20	17475	0.446	8737	0.413	5824	0.391

Tabelle 3.1.2: Speichereffektivität und maximale Anzahl der einspeicherbaren Musterzuordnungen entsprechend Glg. (3.1.10) für verschiedene k,l-Werte (n=m=1024).

Der maximale Informationsgehalt und die maximale Speichereffektivität für eine autoassoziative Matrix ergeben sich aus entsprechenden Abschätzungen [47]:

$$E(I_A) < \frac{\ln 2}{2} \cdot n^2 \approx 0.35 \cdot n^2 \qquad (3.1.14a)$$

$$S^A_{eff} < 0.35 \qquad (3.1.14b)$$

Der Faktor 1/2 in der maximalen Speicherkapazität bezogen auf die Heteroassoziation (3.1.8) begründet sich in der symmetrischen Verbindungsmatrix, die sich im Fall der Autoassoziation ergibt (Abb. 3.1.4). Die Gewichte der Dreiecksmatrix rechts bzw. links der Diagonalen w_{ii} sind redundant.

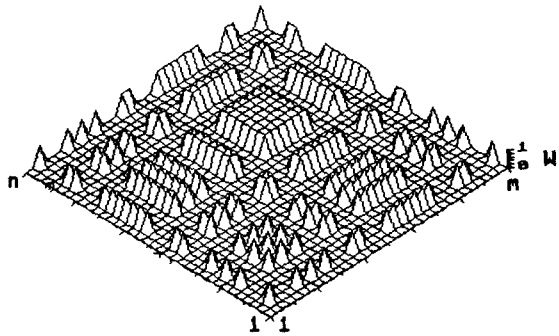


Abb. 3.1.4: Autoassoziative Gewichtsmatrix (n=35).

Die Abschätzungen für die Autoassoziation sind aufwendiger. Bei der Bestimmung des Informationsgehaltes muß man bedenken, daß das unvollständige bzw. fehlerhafte Eingabemuster bereits einen Teil der geforderten Ausgabeinformation enthält. Diesen Teil muß man von dem Informationsgehalt des zu vervollständigenden Musters abziehen. Ferner bleibt zu beachten, daß jedes Teilmuster der eingespeicherten Muster als Eingabe dienen kann und damit bei der Abschätzung der Speichereffektivität berücksichtigt werden muß.

Zusammenfassend läßt sich für assoziative Matrizen eine maximale Speichereffektivität von ln2 bei der Heteroassoziation und (ln2)/2 bei der Autoassoziation erzielen. Voraussetzung ist eine spärliche Kodierung der

zu speichernden Muster, d.h. die Anzahl der Einsen ist viel kleiner als die Anzahl der Nullen (k,l = O(log n)). Die Musterzuordnungen sind aber nicht garantiert fehlerfrei, sondern enthalten $E(N_t)$ Bitfehler. Durch eine entsprechende Wahl von (z,k,l) entsprechend den Bedingungen (3.1.9) bzw. (3.1.10) läßt sich die Bitfehleranzahl kontrollieren.

3.1.2 Speichereffektivität eines Hopfield-Netzes

Die Bestimmung der Speichereffektivität eines Hopfield-Netzes erweist sich aufgrund der Rückkopplung und der mehrwertigen Verbindungsgewichten schwieriger als im Fall der assoziativen Matrix. Auch die Bestimmung des Informationsgehaltes eines fehlerhaft erzeugten Ausgabemusters ist schwieriger, denn es können in diesem Fall nicht nur zusätzliche Einsen hinzukommen, sondern auch einige fehlen. Die meisten Abschätzungen beschränken sich daher auf die Bestimmung der maximalen Anzahl von Mustern, die mit großer Wahrscheinlichkeit richtig vervollständigt werden können. Es gibt eine Reihe von verschiedenen Verfahren, die die maximale Anzahl von möglichst fehlerfrei speicherbaren Mustern abschätzen [22,23,48,49] und zu vergleichbaren Ergebnissen kommen. Beispielhaft seien hier die Abschätzungen von Forshaw aufgeführt, die auf einem ähnlichen Ansatz beruhen wie die zur assoziativen Matrix [50]:

$$z = 0.152 \cdot n \text{ , mit } p=0.99,\ f=1 \text{ ;} \qquad (3.1.15a)$$

$$z = 0.056 \cdot n \text{ , mit } p=0.95,\ f=0.5 \text{ ;} \qquad (3.1.15b)$$

$$z = 0.035 \cdot n \text{ , mit } p=0.99,\ f=0.5 \text{ ;} \qquad (3.1.15c)$$

- p = Anteil der korrekten Komponenten im Ausgabemuster;
- f = Anteil der korrekten Komponenten im Eingabemuster;
- n = Anzahl der Verarbeitungseinheiten.

Die Anzahl der positiven ('+1') und negativen ('-1') Komponenten ist für die zu speichernden Muster in etwa gleich ($\approx n/2$). Die Angaben von Forshaw beziehen sich allerdings auf den Netzwerkzustand nach nur einem parallelen Iterationsschritt, d.h. die Rückkopplung wird nicht betrachtet. Die Ergebnisse verbessern sich jedoch nur unwesentlich, wenn die Rückkopplung zugelassen wird [50]. Die obere Grenze nach (3.1.15a) ist zudem identisch.

Es wird deutlich, daß im Vergleich zur assoziativen Matrix nur sehr wenige Muster gespeichert werden können. Eine obere Grenze ist $0.15 \cdot n$, d.h. die Anzahl der Muster ist kleiner als die Anzahl der Verarbeitungs-

Eingabemuster abgeleitet worden ist. Verlangt man eine Vervollständigung von gestörten Eingabemustern, dann reduziert sich die Anzahl der speicherbaren Muster auf die in (3.1.15b) und (3.1.15c) angegebenen Werte. Auch hier zeigt sich wie bei der assoziativen Matrix ein großer Einfluß der geforderten Zuverlässigkeit der Musterabbildung auf die Anzahl der speicherbaren Muster. Die Zuverlässigkeit entspricht hier dem Anteil p der korrekt assoziierten Komponenten des geforderten Ausgabemusters.

Die Speichereffektivität eines Hopfield-Netzes ist folglich in jedem Fall kleiner als 0.15 und somit auch kleiner als die einer assoziativen Matrix. Eine Erklärung für die geringere Speichereffektivität eines Hopfield-Netzes ist die Lernregel. Während mit der einfachen Hebb-Regel entsprechend (2.3.7) nur O(kl/mn) Verbindungen verändert werden, ändert die Lernregel entsprechend (2.3.10) jeweils alle Verbindungen. (Abb. 3.1.5). In einem Hopfield-Netz kommt es somit schneller zu Überlagerungen und damit zu fehlerhaften Mustervervollständigungen.

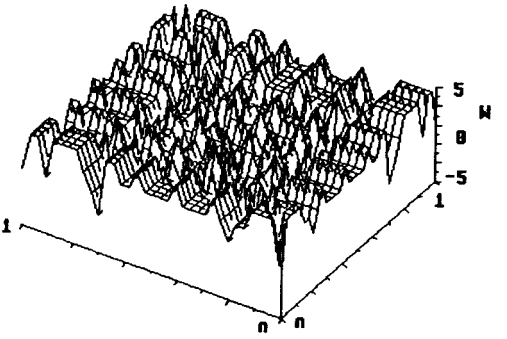


Abb. 3.1.5: Belegungsgrad eines Hopfield-Netzes (n=35) nach dem Einspeichern von 5 Musterpaaren mit etwa gleicher Anzahl von positiven und negativen Komponenten.

3.1.3 Speichereffektivität der Pseudoinversen-Technik

Die Pseudoinversen-Technik wird in der Regel für lineare Abbildungen zwischen kontinuierlichen Ein- und Ausgabemustern angewendet. Selbst bei der Verwendung von binären Mustern werden die Ausgaben im allgemeinen kontinuierlich sein. Ein Vergleich mit einer assoziativen Matrix ist daher nur bedingt möglich und soll hier nur an einem Beispiel erfolgen.

Die Anzahl der fehlerfrei einspeicherbaren Musterzuordnungen ist bei der Pseudoinversen-Technik durch die Dimension m der Eingabevektoren $\mathbf{x}^l \epsilon X^m$ begrenzt. Wie bereits in Abschnitt 2.4 erwähnt, lassen sich zum Beispiel mit der Pseudoinversen-Technik maximal 35 Musterpaare der 7x5-Rasterbilder von ASCII-Zeichen (Abb. 2.4.1) fehlerfrei heteroassoziativ zuordnen. Die kontinuierlichen Werte der berechneten Ausgabewerte $\mathbf{y}^l = \mathbf{W} \cdot \mathbf{x}^l$ sind dabei von einer einfachen Schwellenwertfunktion f_{Th} in binäre umgewandelt worden:

$$f_{Th} : \mathbb{R} \longrightarrow \{0,1\}$$

$$f_{Th}(y_j^l) = \begin{cases} 1, & \text{für } y_j^l \geq Th=0.5 \\ 0, & \text{sonst} \end{cases} \qquad (3.1.16)$$

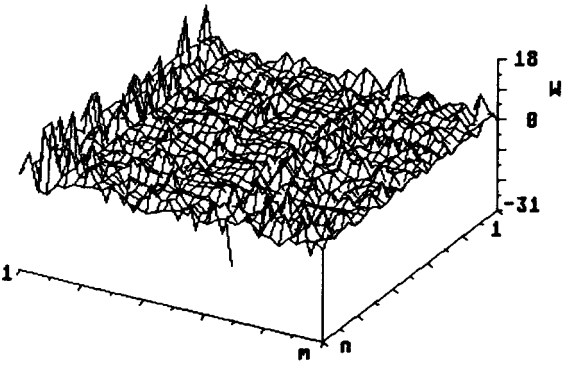


*Abb. 3.1.6: Heteroassoziative Gewichtsmatrix **W** nach dem Einspeichern von 35 ASCII-Mustern mit der Pseudoinversen-Technik.*

Die Komponenten der Matrix **W** nehmen ebenfalls kontinuierliche Werte an (Abb. 3.1.6) und müssen im Hinblick auf eine Bestimmung der Speichereffektivität digitalisiert werden. Für das obige Beispiel müssen die Matrixkomponenten mit mindestens 8 Bit dargestellt werden [32], damit die Musterzuordnungen fehlerfrei bleiben (Abb. 3.1.7). Für diesen speziellen Fall ergibt sich eine Speichereffektivität von 1/8=0.125, wenn man für den Informationsgehalt I^l der Ausgabemuster vereinfachend n=35 Bit annimmt. Durch die Verwendung von mehrwertigen Gewichten wird die

Speichereffektivität bei der Pseudoinversen-Technik gemäß (3.1.2) auf jeden Fall kleiner als 1/2 sein ($z_{max}=m$, $I^i_{max}=n$):

$$S^{PT}_{eff} = \frac{z \cdot I^i_{max}}{n \cdot m \cdot b} < \frac{1}{b} \qquad (3.1.17)$$

Vergleichsweise können in eine assoziative Matrix (n=m=35) im günstigsten Fall 6 Musterpaare fehlerfrei heteroassoziativ eingespeichert werden [32]. Das entspricht einer Speichereffektivität von 0.17 (I^i=35 Bit). Für dieses spezielle Beispiel ist die Speichereffektivität geringfügig günstiger als die der Pseudoinversen-Technik. Der relativ niedrige Wert für die Speichereffektivität der assoziativen Matrix gegenüber dem asymptotischen Grenzwert von 0.69 (3.1.8) begründet sich im wesentlichen in der Forderung nach einer fehlerfreien Zuordnung für alle Musterpaare (3.1.9) und in der Wahl der Muster. Für eine optimale Nutzung der assoziativen Matrix sind diese Muster denkbar ungeeignet, weil die Anzahl der Einsen und der Nullen in etwa gleich groß ist und somit die Muster keinesfalls spärlich kodiert sind.

Anhand dieses Beispiels wird deutlich, daß die Speichereffektivität der assoziativen Matrix gemäß (3.1.2) im allgemeinen nicht schlechter als die der Pseudoinversen-Technik ist. Für den speziellen Fall spärlich kodierter Musterpaare hat die assoziative Matrix den großen Vorteil, daß die Anzahl der speicherbaren Musterzuordnungen gemäß (3.1.13) proportional zur Anzahl der Verbindungen ist. Mit der Pseudoinversen-Technik können dagegen nur maximal m und mit dem Hopfield-Netz nur 0.15·n Muster gespeichert werden.

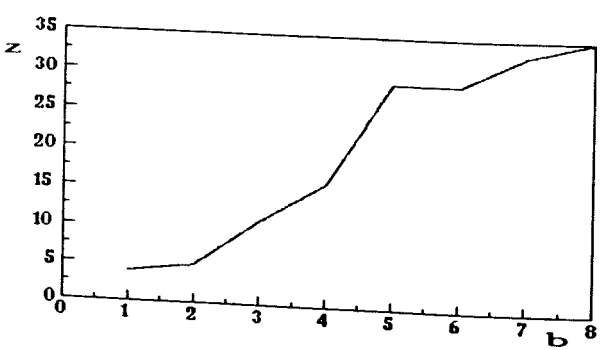


Abb. 3.1.7: Anzahl der fehlerfrei speicherbaren Musterzuordnungen z in Abhängigkeit von der Anzahl der Bits b bei digital realisierten Matrixkomponenten (Pseudoinversen-Technik) [32].

3.2 Fehlertoleranz

Im Rahmen dieser Arbeit werden drei Fehlerarten unterschieden:

1) externe Fehler in der Eingabe, z.B. unvollständige oder verrauschte Eingaben;
2) defekte Komponenten im Netzwerk, z.B. nicht funktionstüchtige Verbindungsgewichte;
3) Berechnungsungenauigkeiten; z.B. bei der Berechnung der Aktivierung.

3.2.1 Externe Fehler in der Eingabe

Die Fehlertoleranz assoziativer Netzwerke bezüglich externer Fehler hängt im wesentlichen von dem Füllungsgrad des Netzwerkes und von der Eingabekodierung ab. Grundsätzlich ist diese Art der Fehlertoleranz beschränkt durch den minimalen Hammingabstand H_{min} der zu trennenden Muster. Die maximale Anzahl der fehlerhaften Bits ist somit kleiner oder gleich $H_{min}/2$. In den folgenden Ausführungen wird davon ausgegangen, daß diese Bedingung erfüllt ist.

Für eine assoziative Matrix kann man die Fehlertoleranz bezüglich externer Fehler in der Eingabe quantitativ durch die Fehlerwahrscheinlichkeit p (3.1.4) bzw. p^1_{on} ausdrücken, mit der im Ausgabemuster eine zusätzliche Eins erzeugt wird. In dieser Formel sind alle wesentlichen Systemparameter (n,m,z,k,l) enthalten. Es zeigt sich, daß hier unterschieden werden muß, ob aktivierte Eingabewerte fehlen (Unvollständigkeit) oder ob Eingabewerte invertiert werden (Störungen).

Im ersten Fall sind im Eingabemuster nur noch l' von den geforderten l Einsen (l'<l) gesetzt (vgl. Abb. 2.3.4). Es ergibt sich eine Erhöhung der Fehlerwahrscheinlichkeit $p(l')$ dadurch, daß ein niedrigerer Schwellenwert Th=l' anstatt l beim Auslesen gewählt werden muß. Der Erwartungswert $E(I_H)$ für den Informationsgehalt einer assoziativen Matrix kann durch Einsetzen von $p(l')$ in (3.1.7) bestimmt werden. Ein Beispiel für die Abhängigkeit des Erwartungswertes von dem Verhältnis der eingegebenen l' zu den geforderten l Einsen im Eingabemuster zeigt Abb. 3.2.1. Eine Erhöhung der Fehlertoleranz geht somit immer zu Lasten der Speichereffektivität, was nicht anders zu erwarten war.

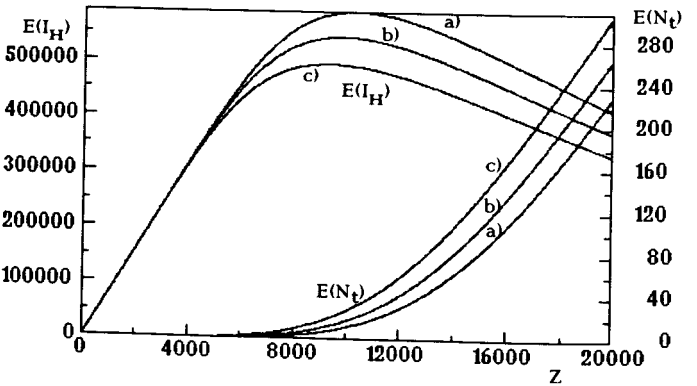


Abb. 3.2.1: Erwartungswert für den Informationsgehalt $E(I_H)$ in Bit und für die Anzahl der Bitfehler $E(N_t)$ für verschiedene Verhältnisse (l'/l) in Abhängigkeit von der Musteranzahl z (n = m = 1000): a) l'/l=1; b) l'/l=0.9; c) l'/l=0.8; (l=k=10).

Die Anzahl der Muster, die mit nur l' von insgesamt l möglichen Einsen mit $E(N_t)<1$ (3.1.5) assoziiert werden können, reduziert sich auf:

$$1 > (n-k)\cdot \binom{l}{l'} \qquad (3.2.1a)$$

bzw.
$$1 > (n-k)\cdot p_{on}^{l'} \qquad (3.2.1b)$$

Beide Ungleichungen können wieder nur numerisch hinsichtlich einer maximalen Musteranzahl bzw. des Erwartungswertes $E(I_H)$ für k und l optimiert werden. In eine assoziative Matrix (n=m=1024) lassen sich für k=3,l=10 maximal 20884 Muster mit l'=9 bzw. 18296 Muster mit l'=8 assoziativ zuordnen (3.2.1a). Das entspricht einer Speichereffektivität von 0.506 bzw. 0.443 (Tabelle 3.2.1).

k	3		6		9	
l'	z	S_{eff}	z	S_{eff}	z	S_{eff}
10	23313	0.565	11664	0.530	7780	0.506
9	20884	0.506	10449	0.475	6971	0.453
8	18296	0.443	9154	0.416	6108	0.397

Tabelle 3.2.1 : Speichereffektivität und maximale Anzahl der Musterzuordnungen nach (3.2.1a) für verschiedene l',k-Werte (n=m=1024, l=10).

Für die Autoassoziation ist die Unvollständigkeit der Eingabe kein Fehler, sondern die eigentliche Anwendung (Mustervervollständigung). Die Abhängigkeit des Erwartungswertes $E(I_A)$ von der Unvollständigkeit (I'/I) der Eingabe ist beispielhaft in Abb. 3.2.2 aufgetragen. Die meisten Muster lassen sich bei Eingabe von mindestens 50% ihrer Einsen noch vervollständigen. Muster, die nach einem Assoziationsschritt noch unvollständig sind, können in einem zweiten Assoziationsschritt bei geänderter Schwelle vervollständigt werden, sofern die Wahrscheinlichkeit für eine fehlerhafte Eins im Ausgabemuster sehr gering ist. Ist das nicht der Fall, dann müssen fehlerhafte Einsen im Eingabemuster bei der Assoziation berücksichtigt werden (siehe unten).

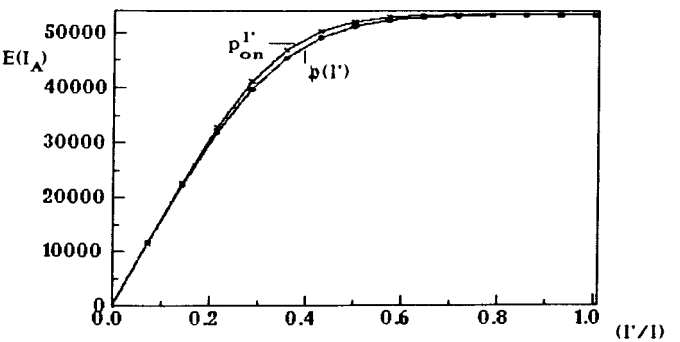


Abb. 3.2.2: Erwartungswert $E(I_A)$ des Informationsgehaltes einer autoassoziativen Matrix (n=500) in Abhängigkeit von der Unvollständigkeit (I'/I) des Eingabemusters (k=I=14, z=600).

Im Eingabemuster können nicht nur Einsen ausfallen, sondern es können auch zusätzliche Einsen erzeugt werden. Bei derartigen Störungen im Eingabemuster kann man ebenfalls von der Fehlerwahrscheinlichkeit p ausgehen. Allerdings muß berücksichtigt werden, daß sich die I' Einsen im Eingabemuster aus I^r richtigen und I^f falschen Einsen zusammensetzen ($I'=I^r+I^f$). Es sind zwei Fehler zu unterscheiden: Erstens kann eine fehlerhafte Eins im Ausgabemuster erzeugt werden; zweitens kann eine geforderte Eins nicht erzeugt werden. Setzt man eine Binomial-Verteilung der Aktivierung a_j der Verarbeitungseinheiten voraus, dann läßt sich die Wahrscheinlichkeit dieser beiden Fehler wie folgt abschätzen:

Wahrscheinlichkeit für die Erzeugung einer fehlerhaften Eins:

$$P(a_j \geq Th,\ y_j=0) = p_1(Th) \approx \sum_{i=Th}^{l'} \binom{l'}{i} \cdot p_{on}^{i} \cdot (1 - p_{on})^{l' - i} \qquad (3.2.2)$$

Wahrscheinlichkeit für das Fehlen einer geforderten Eins:

$$P(a_j < Th,\ y_j = 1) = \begin{cases} 0, & \text{falls } Th \leq l^r \\ p_0 & \text{sonst} \end{cases}$$

$$\text{mit} \quad p_0 \approx \sum_{i=0}^{Th-l^r-1} \binom{l^f}{i} \cdot p_{on}^{i} \cdot (1 - p_{on})^{l^f - i} \qquad (3.2.3)$$

Wird der Schwellenwert Th kleiner als die Anzahl der korrekten Einsen l^r gewählt, dann entfällt der zweite Fehler (3.2.3). Durch das Einsetzen von (3.2.2) in (3.1.7) läßt sich wieder der Erwartungswert $E(I_H)$ für den Informationsgehalt einer assoziativen Matrix abschätzen. In Abb. 3.2.3 ist der Verlauf von $E(I_H)$ in Abhängigkeit von der Musteranzahl an drei Beispielen aufgezeigt.

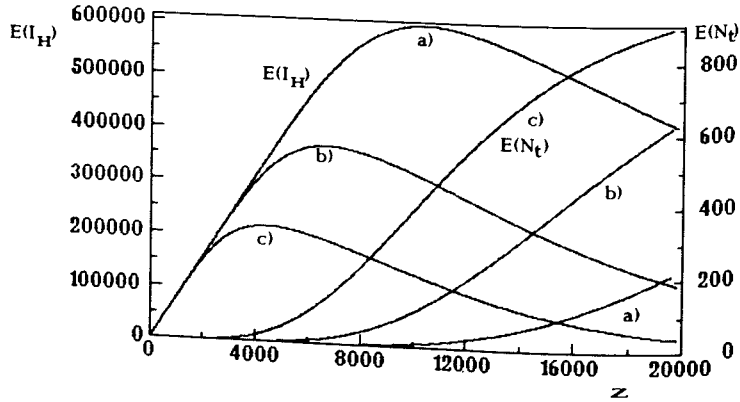


Abb. 3.2.3: Erwartungswert für den Informationsgehalt $E(I_H)$ in Bit und für die Anzahl der Bitfehler $E(N_t)$ für gestörte Eingabemuster in Abhängigkeit von der Musteranzahl z (n=m=1000, k=l=10): a) l^r=l'=Th=10; b) l^r=Th=8, l'=9; c) l^r=Th=6, l'=8.

Für die Anzahl der Muster, die mit $Th=l^r$ assoziiert werden können, ergibt sich entsprechend (3.1.5):

$$1 > (n-k) \cdot \wp_1(l^r) \qquad (3.2.4)$$

In Tabelle 3.2.2 sind beispielhaft numerisch ermittelte Zahlenwerte aufgeführt. Für den ersten Fall mit $l=10$, $l^r=Th=8$ erkennt man im Vergleich zu Tabelle 3.2.1 ($l=10, Th=l^r=8$) eine Reduzierung der Musteranzahl von 18296 auf 14206 ($k=3$). Es wird deutlich, daß Störungen in der Eingabe einen stärkeren Einfluß auf die Speichereffektivität haben als unvollständige Eingabemuster.

k			3		6		9	
l	l^r	l^f	z	S_{eff}	z	S_{eff}	z	S_{eff}
10	9	8	14206	0.343	7106	0.323	4740	0.308
14	13	12	16121	0.391	8064	0.367	5378	0.350
20	18	16	12622	0.306	6313	0.287	4210	0.274

Tabelle 3.2.2 : Speichereffektivität und maximale Anzahl der Musterzuordnungen nach (3.2.4) für verschiedene k,l,l^r,l^f-Werte ($n=m=1024$, $Th=l^r$).

Für das Hopfield-Modell gelten ähnliche Aussagen wie für die assoziative Matrix [31,50], allerdings ist die Speichereffektivität geringer (3.1.15b-c). Der stärkere Einfluß von gestörten Eingaben gegenüber unvollständigen Eingaben auf die Anzahl der speicherbaren Muster zeigt sich auch hier. Ein weiterer Nachteil des Hopfield-Modells gegenüber der assoziativen Matrix ist, daß die fehlerhaften Zuordnungen schwerer erkennbar sind. Bei der assoziativen Matrix sind fehlerhafte Ausgaben anhand von zusätzlichen Einsen zu erkennen. Der stabile Endzustand eines Hopfield-Netzes kann demgegenüber nicht nur zusätzlich Einsen ('+1') enthalten, sondern es können auch geforderte Einsen ('+1') fehlen. Im Endzustand kann sich dadurch ein Muster ergeben, das auch in das Netzwerk eingespeichert worden ist, aber nicht im Sinne des Hamming-Abstandes der nächste Nachbar zur Eingabe ist [51].

Abschätzungen für die Fehlertoleranz der Pseudoinversen-Technik basieren überwiegend auf Rechnersimulationen [25,52]. Interessant im Vergleich zur assoziativen Matrix sind die Ergebnisse von Stiles/Denq [53], die

ebenfalls mit binären Mustern (+1,-1) gearbeitet haben. Tabelle 3.2.3 faßt die Ergebnisse ihrer Arbeit zusammen. Demnach ergibt sich eine hohe Empfindlichkeit der Pseudoinversen-Technik gegenüber Störungen bzw. Unvollständigkeit der Eingabemuster. Die Fehlertoleranz ist somit nicht besser als die der assoziativen Matrix.

p_1	Hetero	Auto
0	n	n
0.05	0.12 · n	0.12 · n
0.15	0.08 · n	0.08 · n
0.25	0.02 · n	0.02 · n

Tabelle 3.2.3: Maximale Anzahl von fehlerfrei speicherbaren Musterzuordnungen bzw. -vervollständigungen bei gestörten Eingabemustern. Der Anteil der gestörten Komponenten ist p_1 [53].

3.2.2 Defekte Komponenten im Netzwerk

In einem assoziativen Netzwerk können Defekte in den einzelnen Verarbeitungseinheiten oder in der Verbindungsstruktur auftreten. Für die Defekte innerhalb einer Verarbeitungseinheit sei vorausgesetzt, daß alle Defekte, die nicht die Gewichte betreffen, zu einem Ausfall der gesamten Verarbeitungseinheit führen. Für Defekte in der Verbindungsstruktur sei vereinfachend vorausgesetzt, daß sie entweder als Fehler in der Eingabe oder als defekte Verbindungsgewichte aufgefaßt werden können.

Für die hier vorgestellten einfachen assoziativen Netzwerke bedeutet eine defekte Verarbeitungseinheit, daß die entsprechende Komponente des Ausgabemusters konstant einen Wert annimmt. Dieser Fehler, der einfach zu detektieren ist, läßt sich nur durch den Austausch dieser Verarbeitungseinheit mit einer funktionstüchtigen beheben.

Den Einfluß von defekten Verbindungsgewichten kann man im Fall der assoziativen Matrix wieder mit Hilfe der Wahrscheinlichkeit für eine fehlerhafte Eins im Ausgabemuster abschätzen (vgl. 3.2.1). Von den m×n Verbindungsgewichten seien aufgrund von Fertigungsfehlern nur der Anteil $c \cdot m \cdot n$ funktionstüchtig, mit $0<c<1$. Zunächst sollen die $(1-c) \cdot m \cdot n$ defekten Verbindungsgewichte keinen Einfluß mehr auf die Aktivierung ($w_{ij}=0$) haben. Die Aktivierung einer Verarbeitungseinheit ist wieder eine diskrete Zufalls-

variable mit dem Erwartungswert $E(a_1)$, wenn die Verarbeitungseinheit eine Eins erzeugen sollte, und $E(a_0)$ im anderen Fall [54,55]:

$$E(a_1) = 1 \cdot c \tag{3.2.5a}$$
$$E(a_0) = 1 \cdot c \cdot p_{on} \tag{3.2.5b}$$

Die Wahrscheinlichkeit p_{on}, mit der eine Verbindung gesetzt ist, ergibt sich entsprechend (3.1.3). Für eine binomial verteilte Aktivierung erhält man analog zu (3.2.2) die Wahrscheinlichkeit für das Auftreten einer fehlerhaften Eins:

$$P(a_j \geq Th,\ y_j=0) = \wp_{c1}(Th) \approx \sum_{i=Th}^{l} \binom{l}{i} \cdot (c \cdot p_{on})^i \cdot (1-c \cdot p_{on})^{l-i} \tag{3.2.6}$$

Die Wahrscheinlichkeit für das Fehlen einer geforderten Eins im Ausgabemuster ergibt sich zu:

$$P(a_j < Th,\ y_j=1) = \wp_{c0}(Th) \approx \sum_{i=0}^{Th-1} \binom{l}{i} \cdot c^i \cdot (1-c)^{l-i} \tag{3.2.7}$$

Die Anzahl der Muster, die sich für $Th=c \cdot l$ mit einer Bit-Fehlerwahrscheinlichkeit von kleiner als Eins assoziieren lassen, kann erneut numerisch aus der folgenden Bedingung bestimmt werden:

$$1 > (n-k) \cdot \wp_{c1} + k \cdot \wp_{c0} \tag{3.2.8}$$

In Tabelle 3.2.4 sind beispielhaft numerisch ermittelte Zahlenwerte für die maximal speicherbare Musteranzahl unter Einhaltung der Bedingung (3.2.8) aufgeführt. Für den Fall, daß defekte Verbindungsgewichte nicht nur ausfallen ($w_{ij}=0$), sondern auch konstant eingeschaltet sind ($w_{ij}=1$), geht man entsprechend vor. Die (1-c) defekten Verbindungselemente sind mit einer Wahrscheinlichkeit p_1 konstant eingeschaltet und mit der Wahrscheinlichkeit p_0 konstant abgeschaltet ($1=p_1+p_0$). Die Wahrscheinlichkeit, daß ein Verbindungselement nach dem Einspeichern (p_{on},(3.1.3)) unter Berücksichtigung der defekten Verbindungen eingeschaltet ist, ergibt sich zu:

$$\tilde{p}_{on} = p_{on} \cdot c + (1-c) \cdot p_1 \tag{3.2.9}$$

Für eine binomial verteilte Aktivierung erhält man analog zu (3.2.6) und (3.2.7) die Wahrscheinlichkeit für das Auftreten einer fehlerhaften bzw. für das Fehlen einer geforderten Eins im Ausgabemuster:

$$P(a_j \geq Th,\ y_j=0) = \wp^f_{c1}(Th) \approx \sum_{i=Th}^{l} \binom{l}{i} \cdot (\tilde{p}_{on})^i \cdot (1-\tilde{p}_{on})^{l-i} \qquad (3.2.10)$$

$$P(a_j < Th,\ y_j=1) = \wp^f_{c0}(Th) \approx \sum_{i=0}^{Th-1} \binom{l}{i} \cdot (c+f_1)^i \cdot (1-c-f_1)^{l-i} \qquad (3.2.11)$$

mit $f_1=p_1 \cdot (1-c)$.

Werden diese beiden Wahrscheinlichkeiten in (3.2.8) eingesetzt, kann man wieder auf die maximale Anzahl von speicherbaren Musterpaaren schließen (Tabelle 3.2.4):

$$1 > (n-k) \cdot \wp^f_{c1} + k \cdot \wp^f_{c0} \qquad (3.2.12)$$

In Abb. 3.2.4 ist für beide Fälle beispielhaft die Abhängigkeit der Musteranzahl von dem Anteil (1-c) defekter Verbindungen aufgezeigt. Die Abstufungen ergeben sich aus der Abhängigkeit des Schwellenwertes Th von dem Anteil der defekten Verbindungen. Je größer dieser Anteil ist, desto kleiner muß der Schwellenwert gewählt werden. Die Wahrscheinlichkeit $\wp_{c1}$ bzw. $\wp^f_{c1}$ für eine fehlerhafte '1' in der Ausgabe wird entsprechend größer.

z		k=3		k=6		k=9	
l	1-c	(3.2.8)	(3.2.12)	(3.2.8)	(3.2.12)	(3.1.8)	(3.1.12)
10	0	24238		12123		8086	
	0.05	16791	20565	7879	7843	4702	5155
15	0	23173		11590		7729	
	0.05	17365	17174	7159	8380	4690	5418

Tabelle 3.2.4: Maximale Anzahl der Musterzuordnungen entsprechend (3.2.8) bzw. (3.2.12) für verschiedene k,l-Werte und einen Anteil an defekten Verbindungsgewichten von 1-c=0.05 (m=n=1024).

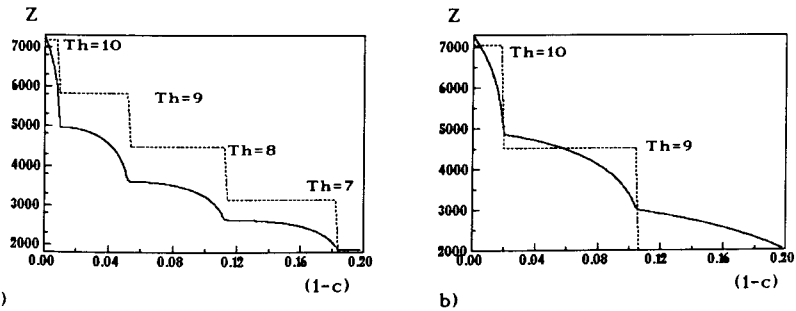


Abb. 3.2.4: Abhängigkeit der Musteranzahl z von dem Anteil (1-c) defekter Verbindungsgewichte unter der Bedingung (3.1.8) (a) und der Bedingung (3.1.12) (p_1=p_0) (b). Es ist n=m=1024, k=l=10.

Wie erwartet, ergibt sich eine beträchtliche Reduzierung der Anzahl von speicherbaren Musterzuordnungen in Abhängigkeit von dem Defektanteil (1-c). Eine ansehnliche Zahl von Musterzuordnungen mit einer Bit-Fehlerwahrscheinlichkeit $E(N_t)<1$ läßt sich aber dennoch speichern. Bei einem Defektanteil von 5% können in beiden Fällen noch mehr als 4000 Musterzuordnungen gespeichert werden.

Für das Hopfield-Netz gelten die gleichen Aussagen wie bereits unter Abschnitt 3.1.2 beschrieben. Ferner bleibt zu beachten, daß durch den Ausfall einzelner Gewichte die Symmetrie der Verbindungsmatrix verletzt wird und so die Konvergenz des Einschwingvorganges nicht mehr unbedingt gewährleistet ist. Bei der Pseudoinversen-Technik wirken sich Defekte in der Verbindungsmatrix aufgrund der linearen Abbildungseigenschaften unmittelbar auf den Ausgang aus. Es ist in diesem Fall angebrachter, mit der Änderung des Signal-Rausch-Abstandes zu argumentieren. Aus diesem Grund ist ein direkter Vergleich mit dem Hopfield-Netz und der assoziativen Matrix nicht möglich.

3.2.3 Berechnungsungenauigkeiten

Die Fehlertoleranz assoziativer Netzwerke gegenüber Ungenauigkeiten in der Berechnung der Dynamikfunktionen α,δ,λ (vgl. Abschnitt 2.2.1) ist hinsichtlich einer Realisierung in analoger Schaltungstechnik von Bedeutung. Die nicht zu verhindernde Parameterstreuung von Bauelementen in einer integrierten Schaltung sollten gerade in diesem Zusammenhang mit in die Systemüberlegungen einbezogen werden. Für das in der assoziativen

Matrix und im Hopfield-Netz verwendete Schwellenneuron läßt sich der Einfluß von Parametervariationen auf das Übertragungsverhalten anschaulich geometrisch interpretieren. Die Aktivierungsfunktion (2.2.1) mit der nachfolgenden Schwellenwertentscheidung trennt den Mustereingaberaum mittels einer Hyperebene in zwei Teilräume mit $\alpha(a)=1$ und $\alpha(a)=0$. Die Gleichung der Hyperebene ergibt sich aus:

$$\underline{x}^T \cdot \underline{w} - Th = 0 \qquad <==>$$

$$x_m = \frac{1}{w_{mj}} \left(Th - \sum_{i=1}^{m-1} x_i \cdot w_{ij} \right) \tag{3.2.13}$$

Die Achsenschnittpunkte dieser Hyperebene mit den jeweiligen x_i-Achsen ergeben sich zu:

$$x_i = \frac{Th}{w_{ij}}, \text{ mit } x_j = 0 \;\forall\; j \neq i \in \{ 1, \ldots, m \} \tag{3.2.14}$$

Die Variation der Werte in (3.2.13) bewirkt entweder eine Verschiebung der Eingabewerte, wenn die Eingabeleitungen gestört sind, oder eine Verschiebung der Trennebene, falls die Gewichte und der Schwellenwert gestört sind. Beide Effekte können zur Folge haben, daß Musterpunkte nahe der Trennebene in das andere Gebiet fallen können. Anschaulich versteht sich die maximal zulässige Abweichung Δ_{max} der Größen (x_i, w_{ij}, Th) als Abstand eines Punktes zur Trennebene (Abb. 3.2.5).

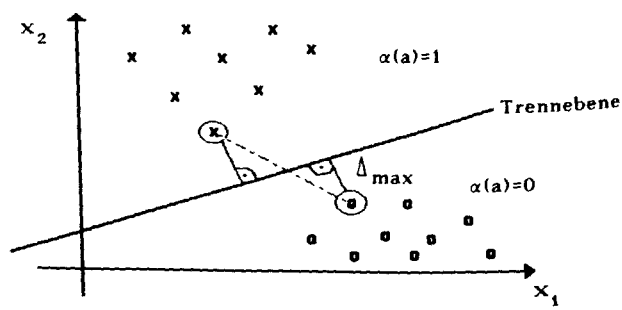


Abb. 3.2.5: Geometrische Anschauung der maximalen Parametervariation.

Die Störungen (Δ_i, Δ_{Th}) lassen sich für den allgemeinen Fall eines Schwellenneurons folgendermaßen ausdrücken:

$$a_j = \sum_{i=1}^{m} w_{ij} \cdot x_i^t \cdot (1+\Delta_i) - Th \cdot (1+\Delta_{Th})$$

$$= \left[\sum_{i=1}^{m} w_{ij} \cdot x_i^t - Th \right] + \left[\sum_{i=1}^{m} w_{ij} \cdot x_i^t \cdot \Delta_i - Th \cdot \Delta_{Th} \right] \qquad (3.2.15)$$

$$(t = 1, \ldots, z)$$

Es ergeben sich zwei Extremfälle :

1) Term 1 nimmt den kleinsten positiven und Term 2 den kleinsten negativen Wert an, d. h. aus einer Ausgabe $\alpha(a)=1$ wird eventuell $\alpha(a)=0$.

2) Term 1 nimmt den größten negativen und Term 2 den größten positiven Wert an, d. h. aus einer Ausgabe $\alpha(a)=0$ wird eventuell $\alpha(a)=1$.

Damit sich keine Änderung des Ausgabewertes ergibt, muß der Absolutwert von Term 1 größer als der Absolutwert von Term 2 sein:

$$\left| \sum_{i=1}^{m} w_{ij} \cdot x_i^t - Th \right| > \left| \sum_{i=1}^{m} w_{ij} \cdot x_i^t \cdot \Delta_i - Th \cdot \Delta_{Th} \right| \quad ; \quad \Delta_i, \Delta_{Th} \in [-1,+1] \qquad (3.2.16)$$

Ersetzt man Δ_i, Δ_{Th} durch die maximale Abweichung:

$$\Delta_{max} = \max \{ \Delta_{Th} , \max\{\Delta_i : i=1,\ldots,m\} \} \qquad (3.2.17)$$

und nimmt man den maximalen Wert von Term 2 ($\sum |w_{ij}| \cdot x_i^t + |Th|$), so folgt für Δ_{max}

$$\Delta_{max} < \min_t \left\{ \frac{\left| \sum_{i=1}^{m} w_{ij} \cdot x_i^t - Th \right|}{\sum_{i=1}^{m} |w_{ij}| \cdot x_i^t + |Th|} \right\} \qquad (3.2.18)$$

$$t = 1, \ldots, z .$$

Diese allgemeine Abschätzung erfaßt den ungünstigsten Fall, bei dem die Störungen in die entsprechend ungünstige Richtung wirken. Für eine Verarbeitungseinheit mit $\alpha(a)=1$ ($\alpha(a)=0$) wird die gewichtete Summe der Eingabewerte verringert (vergrößert) und die Schwelle Th erhöht (verringert). Aus (3.2.18) folgt für eine Realisierung einer assoziativen Matrix in analoger Schaltungstechnik, daß es ungünstig ist, den Schwellenwert Th der Anzahl der Einsen l im Eingabemuster gleichzusetzen. Es ist günstiger den Schwellenwert $Th \approx l-0.5$ zu wählen, damit der Abstand zu den möglichen Aktivierungen 1 und (l-1) in etwa gleich groß wird. Für eine assoziative Matrix mit binären Eingabewerten und Gewichten folgt aus (3.2.18):

$$\Delta_{max} < \frac{0.5}{2 \cdot l - 0.5} \approx \frac{1}{4 \cdot l} \qquad (3.2.19)$$

Für eine assoziative Matrix mit $m=n=10^6$ und $l=ld\ m=20$ muß, um eine fehlerfreie Funktion der Verarbeitungseinheiten zu gewährleisten, die maximale Abweichung Δ_{max} kleiner 0.0125 sein.

Für ein Hopfield-Netz mit Th=0 folgt:

$$\Delta_{max} < \min_{t} \left\{ \frac{\left| \sum_{i=1}^{m} w_{ij} \cdot x_i^t \right|}{\sum_{i=1}^{m} |w_{ij}|} \right\} \qquad (3.2.20)$$

$t = 1, \ldots, z$

Mit $x_i^t \in \{-1,0,+1\}$ und $w_{ij} \in \{-z,\ldots,+z\}$ ist der minimale Zähler gleich 1 und der maximale Nenner $(m-1) \cdot z$. Für zufällige Muster mit genau m/2 positiven ('+1') und negativen ('-1') Komponenten ($p(x_i^t='+1') = p(x_i^t='-1') = 0.5$) muß der Erwartungswert $E(w_{ij})$ und die Varianz $\sigma(w_{ij})$ für ein Verbindungsgewicht nach der Einspeicherung (2.3.10) von z Mustern abgeschätzt werden:

$$E(w_{ij}) = z \cdot [p(x_i=+1 \text{ und } x_j=+1) + p(x_i=-1 \text{ und } x_j=-1) - p(x_i=+1 \text{ und } x_j=-1) - p(x_i=-1 \text{ und } x_j=+1)] = 0 \qquad (3.2.21)$$

$$\sigma(w_{ij}) = \sqrt{z} \qquad (3.2.22)$$

Für den Nenner aus (3.2.20) folgt für $z \approx 0.15m$:

$$\sum_{i=1}^{m} |w_{ij}| = (m-1) \cdot \sqrt{z} \approx 0.4 \cdot m^{3/2} \qquad (3.2.23)$$

Der Zähler in (3.2.20) entspricht der Aktivierung der Verarbeitungseinheit eines Hopfield-Netzes für den Eingabevektor $\underline{x}^t$. Dieser läßt sich wie folgt umformen:

$$a_j = \sum_{i=1}^{m} x_i^t \cdot w_{ij} = \sum_{i=1}^{m} x_i^t \cdot \sum_{r=1}^{z} x_i^r \cdot x_j^r$$

$$= \sum_{i=1}^{m} x_i^t \cdot x_i^t \cdot x_i^t + \sum_{i=1}^{m} x_i^t \cdot \sum_{\substack{r=1 \\ r \neq k}}^{z} x_i^r \cdot x_j^r$$

$$= x_j^t \cdot |\underline{x}^t|^2 + \sum_{\substack{r=1 \\ r \neq k}}^{z} x_j^r \cdot (\underline{x}^t \cdot \underline{x}^r) \qquad (3.2.24)$$

Der erste Term in (3.2.24) entspricht dem gewünschten Signal, während der zweite Term die Überlagerungen aus anderen Musterkorrelationen darstellt. Der günstigste Fall für ein Hopfield-Netz ergibt sich für orthogonale Eingabemuster $\underline{x}^t$, für die der Summenterm in (3.2.24) zu Null und der Nenner in (3.2.20) maximal wird. In diesem Fall ist das Systemverhalten unempfindlich gegenüber Parameterschwankungen. Im ungünstigsten Fall ist der Nenner in (3.2.20) gleich Eins. Für eine Verarbeitungseinheit eines Hopfield-Netzes mit m=1000 folgt:

$$\Delta_{max} < 8 \cdot 10^{-5} \quad \text{mit } w_{ij} \in \{-z, \ldots, z\}$$

$$\text{bzw.} \quad \Delta_{max} < 0.001 \quad \text{mit } w_{ij} \in \{-1, 0, +1\} \qquad (3.2.25)$$

Aus diesen Abschätzungen kann man grundsätzlich die Anforderungen an die Analogtechnik erkennen, die besonders bei der Verwendung von mehrwertigen Gewichten nur schwer zu erfüllen sind. Auch die Realisierung von Verarbeitungseinheiten, bei denen mehr als 100 Eingaben gleichzeitig einen Beitrag zur Aktivierung leisten, wie z.B. beim Hopfield-Netz, wird in analoger Schaltungstechnik Probleme bereiten.

3.3 Vergleich mit inhaltsorientierten Zugriffsmethoden

Es bleibt zu überlegen, welche Vor- und Nachteile assoziative Matrizen gegenüber konventionellen inhaltsorientierten Zugriffsmethoden mit lokaler Speicherung haben. Die einfachste Lösung ist die Speicherung der z Musterpaare $(\underline{x}^i,\underline{y}^i)$ in eine adressenorganisierte Liste. Jeder Listeneintrag (Zeile) entspricht einem Musterpaar, wobei die Muster in diesem Fall nicht mehr spärlich kodiert sein müssen. Der Speicheraufwand SA in Bit für die Liste beträgt (Heteroassoziation):

$$SA = z\cdot\left(ld\binom{n}{k} + ld\binom{m}{l} \right) \quad [Bit] \tag{3.3.1}$$

Bei Eingabe eines Musters $\underline{x}^i$ wird die Zeile in der Liste gesucht, die $\underline{x}^i$ und somit auch die gesuchte Musterzuordnung $\underline{y}^i$ enthält. Die Speichereffektivität dieser Liste beträgt gemäß (3.1.2):

$$S_{eff}^{Liste} = \frac{z\cdot ld\binom{n}{k}}{z\cdot\left(ld\binom{n}{k} + ld\binom{m}{l} \right)} = \left(1 + \frac{ld\binom{m}{l}}{ld\binom{n}{k}} \right)^{-1} \tag{3.3.2}$$

Aus (3.3.2) folgt, daß ein solcher Listenspeicher hinsichtlich der Speichereffektivität nur dann günstiger als eine assoziative Matrix ist, wenn der Informationsgehalt der Eingabemuster kleiner ist als die der Ausgabemuster. Im anderen Fall ist die Speichereffektivität maximal 1/2 und damit nicht mehr größer als die einer assoziativen Matrix.

Ein großer Nachteil des Listenspeichers ist der hohe Suchaufwand für eine Musterzuordnung, denn es müssen sämtliche Listeneinträge mit dem Eingabemuster verglichen werden. Eine wesentliche Verbesserung erhält man mit einer sortierten Liste, bei der höchstens ld z Vergleichsoperationen benötigt werden.

Die Anzahl der Vergleichsoperationen läßt sich unter der Verwendung einer Hash-Funktion noch weiter reduzieren [S6]. Das Eingabemuster wird über eine vorgegebene Rechenvorschrift in eine Speicheradresse umgewandelt, unter der das Ausgabemuster gespeichert ist:

$$H : \{0,1\}^b \longrightarrow \{1, \ldots ,h\}, \quad 2^b \geq h \geq z, \quad b = ld\binom{m}{l}$$

$H(\underline{x}^i \in \{0,1\}^b) \equiv$ Adresse von $\underline{y}^i$ im Listenspeicher;

Der Berechnungsaufwand der Speicheradresse kann bei den heute üblichen Rechengeschwindigkeiten vernachlässigt werden. Im aufwendigsten Fall steht für jedes mögliche Eingabemuster ein separater Listeneintrag zur Verfügung ($h=2^b$), so daß eine Musterzuordnung mit nur einem Listenzugriff ausgeführt werden kann. Der Speicheraufwand für diese Liste ist aber unrealistisch hoch $[2^b \cdot ld \binom{n}{k}]$. Aus diesem Grund sollte die Anzahl der Listeneinträge h nur unwesentlich größer als die Anzahl der Musterpaare z sein. Dadurch steigt aber wiederum die mittlere Anzahl von Vergleichsoperationen, denn es können mit größer werdender Wahrscheinlichkeit Kollisionen auftreten, d.h. eine berechnete Speicheradresse bereits belegt ist. Es muß hier ein Kompromiß zwischen Zugriffszeit und Speicheraufwand gefunden werden. Wünscht man z.B. im Mittel nicht mehr als zwei Vergleichsoperationen pro Musterzuordnung, so sollte das Verhältnis z/h kleiner als 3/4 sein [56].

Den schnellsten Zugriff auf die gesuchte Musterzuordnung erhält man mit vollparallelen inhaltsadressierten Speichern (CAM- content addressable memories [56]), bei denen durch schaltungstechnische Maßnahmen in den einzelnen Speicherzellen alle Musterzuordnungen nur einen Listenzugriff benötigen. Beim Einspeichern neuer Musterpaare kommt es nicht wie beim Hashing zu Kollisionen. Die Anzahl der Listeneinträge kann somit genau der Anzahl z der Musterpaare entsprechen. Ferner müssen die Listeneinträge beim inhaltsadressierten Speicher in keiner Weise geordnet sein. Nachteilig ist der höhere schaltungstechnische Aufwand.

Für alle drei Methoden ist in den Listeneinträgen das Ausgabemuster und das dazugehörige Eingabemuster abzulegen, so daß man weder Speichereffektivität noch Zugriffszeit bezüglich einer assoziativen Matrix gewonnen hat. Der eigentliche Vorteil konventioneller Methoden liegt darin, daß sie garantiert fehlerfrei arbeiten. Bei der Anwendung assoziativer Netzwerke ist, wenn auch mit geringer Wahrscheinlichkeit, mit Fehlern in der Musterzuordnung zu rechnen. Auf der anderen Seite können assoziative Matrizen aber auch Fehler, sowohl in der Eingabe als auch in der Struktur kompensieren. Es handelt sich hier nicht nur um Ein-Bitfehler, die man auch mit fehlerkorrigierenden Kodierungen beheben kann, sondern es können, wie z.B. bei der Autoassoziation, Eingabemuster vervollständigt werden, die nur 60% der geforderten Information enthalten (Abb. 3.2.2). Den Grad der Fehlertoleranz kann man einfach dadurch erhöhen, indem weniger Muster gespeichert werden, d.h. die Redundanz erhöht wird. Die vorgestellten inhaltsadressierten Zugriffsmethoden können gestörte oder unvollständige Eingaben nur mit zusätzlichem schaltungstechnischen oder algorithmischen Aufwand verarbeiten.

Zusammenfassend ist die Leistung von assoziativen Matrizen hinsichtlich der Speichereffektivität und der Zugriffsgeschwindigkeit unter den erörterten Voraussetzungen besser als die der konventionellen Methoden des

assoziativen Zugriffs. Die Voraussetzungen für das Erreichen dieser Effektivität sind die geforderte spärliche Kodierung der Ein- und Ausgabemuster und eine geeignete technische Umsetzung in eine parallele Hardware-Struktur. Bei der Entwicklung von spärlichen Kodierungen ist nicht nur die Anzahl der Einsen wichtig, sondern die Kodierung sollte auch *ähnlichkeitserhaltend* sein. Bezüglich eines gewählten Ähnlichkeitskriteriums sollten ähnliche (unkodierte) Muster durch die Kodierung auch in ähnliche spärliche Muster umgewandelt werden. Originalmuster, die von ihrer Bedeutung bzw. Erscheinung her vergleichbar sind, sollten in spärliche Muster kodiert werden, die eine entsprechende Ähnlichkeit haben, z.B. hinsichtlich ihres Hamming-Abstandes.

Die Umsetzung der assoziativen Matrix in eine parallele Architektur ist aufgrund ihrer einfachen Arbeitsweise und der regulären Struktur naheliegend. Lösungsansätze werden in den folgenden Abschnitten diskutiert.

4. Grundschaltungen für eine Verarbeitungseinheit

Das Hauptmodul eines assoziativen Netzwerkes ist die Verarbeitungseinheit, deren Struktur und Aufgaben entsprechend Definition 1 vorgegeben sind. Da ein assoziatives Netzwerk aus möglichst vielen solcher Verarbeitungseinheiten bestehen sollte, ist eine effiziente Umsetzung in eine adäquate Schaltungsstruktur gefordert. Grundsätzlich kann man sich für eine Realisierung in digitaler, analoger oder gemischt digital-anloger Schaltungstechnik entscheiden. Welcher Lösungsansatz für ein bestimmtes Modell und eine bestimmte Anwendung der geeignete ist, hängt von der zur Verfügung stehenden Schaltungsfläche, der geforderten Zugriffsgeschwindigkeit und Zuverlässigkeit sowie von der Entwurfskomplexität ab. In den folgenden Abschnitten werden diese Kriterien anhand ausgewählter Realisierungsalternativen für die Verarbeitungseinheit einer assoziativen Matrix bzw. eines Hopfield-Netzes diskutiert.

4.1 Realisierung in digitaler Schaltungstechnik

Der Aufwand für eine Realisierung in digitaler Schaltungstechnik ist offensichtlich. Die m Gewichte werden in digitalen Registern gespeichert, deren Wortbreite b vom Wertebereich der Verbindungsgewichte abhängt. Für die assoziative Matrix wird nur ein Bit pro Gewicht benötigt. Das Hopfield-Netz benötigt entweder $b \geq ld(2z+1)$ Bit für $w_{ij} \in \{-z,\ldots,+z\}$ (2.3.10) oder zwei Bit für die Modellvarianten mit $w_{ij} \in \{-1,0,+1\}$. Komplexere Lernmodelle assoziativer Netzwerke benötigen eine Gleitkommadarstellung der Gewichte.

Die Aktivierungsfunktion wird seriell berechnet, weil eine gleichzeitige Berücksichtigung aller m Verbindungsgewichte mit vertretbarem Aufwand in der digitalen Schaltungstechnik nicht möglich ist. Für Eingabewerte $x_i \in \{0,1\}$ bzw. $x_i \in \{-1,1\}$ kann die gewichtete Summe der Eingabewerte (2.2.1) mit einem Addierer berechnet werden, für mehrwertige Eingabesignale benötigt man zusätzlich noch einen Multiplizierer. Der Schwellenwertvergleich wird von einem Komparator ausgeführt, dessen Wortbreite von dem berechneten Aktivierungswert abhängt.

Die Berechnung der Verbindungsgewichte kann für einfache Adaptationsregeln, z.B. die Lernregel der assoziativen Matrix oder des Hopfield-Netzes, ebenfalls von einem einfachen Addierer ausgeführt werden. Komplexe Lernverfahren benötigen in der Regel einen beträchtlich höheren

Berechnungs- und Entwurfsaufwand. Eine Verarbeitungseinheit kann dann die Komplexität eines RISC-Prozessors (reduced instruction set computer) mit einem Flächenaufwand in der Größenordnung eines Chips ($1cm^2$) erreichen. Der Mehraufwand für das Lernen multipliziert sich mit der Anzahl der Verarbeitungseinheiten eines Netzes und zahlt sich nur in der Lernphase aus. Für die Berechnung der Verbindungsgewichte erscheinen daher Emulatoren aus dem Bereich der Neurocomputer angebrachter, die ein großes Netzwerk durch wenige, parallel arbeitende Verarbeitungseinheiten berechnen lassen [3]. Der Zeitverlust ist bei den heute üblichen Rechengeschwindigkeiten gegenüber dem Gewinn an Kosten, Kompaktheit und Flexibilität zu vernachlässigen.

Die einfachste digitale Schaltungsstruktur erhält man für eine Verarbeitungseinheit einer assoziativen Matrix (Abb. 4.1.1). Die Verbindungen sind einfache 1-Bit-Speicherzellen mit wahlfreiem Zugriff. Die gewichtete Summe der Eingabewerte wird von einem Zähler berechnet und der Schwellenwert von einem Komparator ausgeführt. Die Wortbreite für den Zähler und den Komparator bleibt selbst für sehr große Verbindungsmatrizen kleiner als 8 Bit ($b = O(\log l) = O(\log(\log m))$, vgl. 3.1). Für die Berechnung der Lernregel (2.3.7) genügt ein ODER-Gatter, das nicht in Abb. 4.1.1 aufgeführt ist (vgl. Abschnitt 5.1).

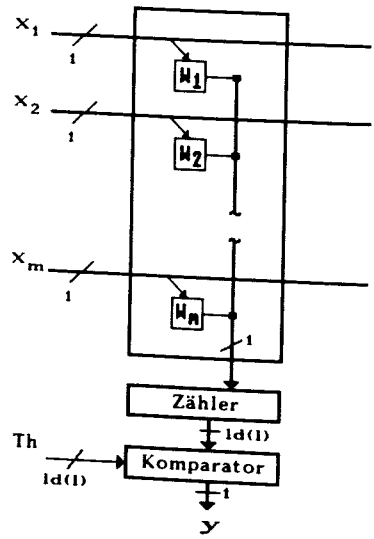


Abb. 4.1.1: Schaltungsschema einer Verarbeitungseinheit einer assoziativen Matrix in digitaler Schaltungstechnik.

Der Flächenaufwand für die Verarbeitungseinheit wird im wesentlichen von den m binären Speicherzellen bestimmt und ist bei der Verwendung von dynamischen Speicherzellen am geringsten. Die Fläche für einen 4-Bit-Zähler kann grob mit 0.121mm^2 (≈185 Transistoren) und für einen 4-Bit-Komparator mit 0.055mm^2 (≈150 Transistoren) abgeschätzt werden. Die Angaben beziehen sich auf die Zellen CTLCL1F und CMK4L1F der Standardzellenbibliothek VENUS3.3 [57] (2µm CMOS-Technologie). In Kapitel 5 wird ein VLSI-Entwurf einer assoziativen Matrix mit einer solchen Verarbeitungseinheit vorgestellt und diskutiert.

4.2 Berechnung der Aktivierungsfunktion in analoger Schaltungstechnik

Die Berechnung der Aktivierung einer Verarbeitungseinheit ist aufgrund der vielen Eingänge die zeitaufwendigste Funktion während eines Assoziationsvorganges. Diese Berechnung kann digital nur sequentiell erfolgen, so daß sich hier ein wesentlicher Vorteil einer analogen Implementierung ergibt. Voraussetzung ist allerdings, daß das gewählte Modell die mit einer analogen Implementierung verbundenen Ungenauigkeiten toleriert. In den folgenden Ausführungen steht die gewichtete Summe der Eingabesignale im Vordergrund, da sie die am häufigsten benutzte Aktivierungsfunktion ist.

4.2.1 Stromsummation

a) Statisches Verhalten

Die gewichtete Summe der Eingangssignale läßt sich in analoger Schaltungstechnik durch die Summation von Strömen mit einem Operationsverstärker (Abb. 4.2.1a) realisieren:

$$U_a = - R_a \cdot \sum_{i=1}^{m} x_i / R_i \qquad (4.2.1)$$

Für jeden hinzukommenden Eingabezweig x_k verändert sich das Aktivierungspotential U_a gemäß:

$$-\Delta U_a = \frac{R_a}{R_k} \cdot x_k \quad \rightarrow \quad -\frac{R_k}{R_a} = \frac{x_k}{\Delta U_a} = r \qquad (4.2.2)$$

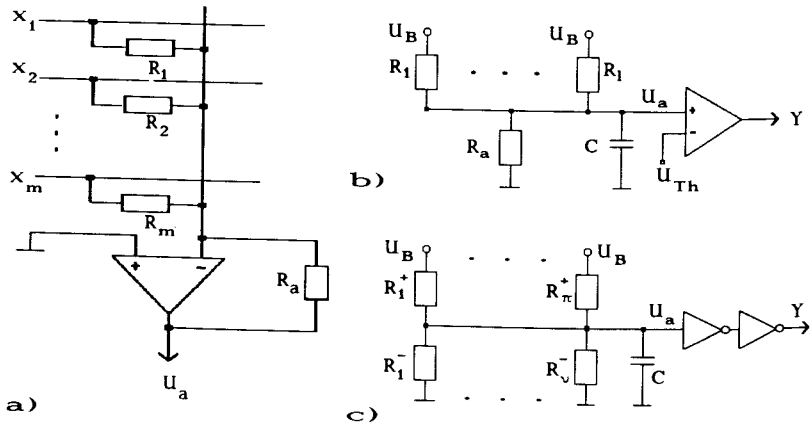


Abb. 4.2.1: Bestimmung der gewichteten Summe in analoger Schaltungstechnik: Summierer mit invertierendem Operationsverstärker (a); vereinfachte Realisierung mit einem Spannungsteilernetz für eine assoziative Matrix (b) und ein Hopfield-Netz (c).

Über das Widerstandsverhältnis r kann der Einfluß der einzelnen Eingabezweige auf das Aktivierungspotential U_a eingestellt werden. Die Realisierung von integrierten Operationsverstärkern ist in der MOS-Schaltungstechnik relativ aufwendig. Eine einfachere, angenäherte Lösung ergibt sich mit einer Spannungsteilerschaltung. Das Aktivierungspotential U_a der Verarbeitungseinheit einer assoziativen Matrix ergibt sich gemäß dem vereinfachten Ersatzschaltbild in Abb. 4.2.1b zu:

$$U_a = U_B \cdot \frac{1}{1 + \frac{R}{1 \cdot R_a}} \qquad R_i = R \ (\forall\, i=1,\ldots,m) \tag{4.2.3}$$

Im Fall der assoziativen Matrix haben nur die l Eingänge mit $x_i=1$ einen Einfluß auf die Aktivierung, alle anderen (m-l) mit $x_i=0$ bleiben unberücksichtigt. Demgegenüber müssen in jeder Verarbeitungseinheit eines Hopfield-Netzes immer (n-1) Eingänge berücksichtigt werden (Abb. 4.2.1c), da $y_i \in \{-1,+1\}$. Sei π die Anzahl der Summanden mit $(w_{ij} \cdot y_i) > 0$ und ν die Anzahl der Summanden mit $(w_{ij} \cdot y_i) < 0$. Das Aktivierungspotential einer

Verarbeitungseinheit eines Hopfield-Netzes ergibt sich entsprechend Abb. 3.2.1c zu:

$$U_a = U_B \cdot \frac{1}{1 + \frac{\nu}{\pi}} \qquad R_i^+ = R_i^- = R \ (\forall i=1, \ldots, n-1) \tag{4.2.4}$$

Betrachten wir zunächst die Verarbeitungseinheit einer assoziativen Matrix. Bei der Festlegung der Widerstandswerte R und R_A (4.2.3) sind zwei Randbedingungen zu beachten. Zum einen darf die Verlustleistung des gesamten Chips, die sich auf n Verarbeitungseinheiten aufteilt, nur einige Watt betragen. Bei einer Betriebsspannung von 5V und einem assoziativen Netzwerk von 1000 Verarbeitungseinheiten auf einem Chip sei die maximale Verlustleistung P_{max}=1W, die bei der Berechnung der Aktivierung verursacht werden darf. Der Querstrom durch das Widerstandsnetzwerk muß somit kleiner als 0.2 mA sein. Von den insgesamt n Verarbeitungseinheiten erhalten $k \approx \log(n)$ genau 1 aktivierte Eingaben, die verbleibenden (n-k) Verarbeitungseinheiten im Mittel $1 \cdot p_{on} \approx 1/2$ aktivierte Eingaben (vgl. Abschnitt 3.1). Im Mittel folgt für den Gesamtwiderstand einer Verarbeitungseinheit:

$$R_g = 2 \cdot R/1 + R_A \geq 25 \ k\Omega \tag{4.2.5}$$

Der Schwellenwertvergleich wird von einer analogen Differenzstufe durchgeführt, wobei der Schwellenwert Th als analoge Spannung U_{Th} allen Verarbeitungseinheiten zugeführt wird (Abb. 4.2.1b). Hieraus erwächst die zweite Randbedingung für die Wahl der Widerstandswerte. Der Spannungshub ΔU_a muß für jeden zusätzlich hinzugeschalteten Widerstandszweig einen Mindestwert ΔU_{min} überschreiten, damit die entsprechenden Aktivierungspotentiale noch sicher von der nachfolgenden Komparatorschaltung aufgelöst werden können:

$$\Delta U_{min} \leq \Delta U_a = U_a(1) - U_a(1-1) \tag{4.2.6}$$

Aus dieser Bedingung kann die maximale Anzahl aktivierter Eingabezweige l_{max} ermittelt werden, für die diese Bedingung gerade noch erfüllt ist:

$$\frac{\partial U_a}{\partial l} = \Delta U_{min} = U_B \cdot \frac{r}{(l + r)^2}, \qquad r = R/R_a$$

$$\rightarrow \quad l = \sqrt{r \cdot U_B/\Delta U_{min}} - r$$

$$\rightarrow \quad l_{max} = 1/4 \cdot U_B/\Delta U_{min} \tag{4.2.7}$$

$$\text{für } r = 1/4 \cdot U_B/\Delta U_{min}.$$

Der minimale Spannungshub ΔU_{min} wird durch die Auflösegenauigkeit der nachfolgenden Ausgabestufe und der technologiebedingten Parametervariationen der Bauelemente auf einem Chip bestimmt. Für ΔU_{min}=50mV (bzw. 10mV) ist die maximale Stufenanzahl l_{max}=25 (bzw. 125) mit einem Widerstandsverhältnis von r = 25 (bzw. 125). Mit (4.2.5) folgt für R_a ein Widerstand von 8.3kΩ und für R ein Wert von 208kΩ (bzw. 1MΩ). Das Aktivierungspotential $U_a(l_{max})$ beträgt $U_B/2$. Gegenüber einer Realisierung mit Operationsverstärker resultiert eine Verringerung der Stufenanzahl sowie eine Veränderung des Widerstandsverhältnisses r um den Faktor 1/4.

Die Bestimmung der Widerstandswerte im Fall des Hopfield-Netzes ist nicht ganz so einfach wie bei der assoziativen Matrix. In diesem Fall muß das Verhältnis der positiven und der negativen Beiträge der Aktivierungssumme abgeschätzt werden:

$$R^+/\pi + R^-/\nu \geq 25\ k\Omega$$

$$\rightarrow \qquad R\,\frac{\nu+\pi}{\nu\cdot\pi} \geq 25\ k\Omega \qquad \text{für } R^+=R^-=R \qquad (4.2.8)$$

Sind im Netzwerk nur orthogonale Vektoren gespeichert, kommen im stabilen Zustand des Netzwerkes entweder nur positive Summanden oder nur negative Summanden in jeder Verarbeitungseinheit vor. Wie bereits in Abschnitt 3.2.3 erläutert, entfällt dann der zweite Term in (3.2.24). Unter dieser Voraussetzung gibt es keinen statischen Querstrom und es fällt nur dynamische Verlustleistung an. Für beliebige Muster ist der zweite Term in (3.2.24) ungleich Null, so daß die Verarbeitungseinheit auch im stabilen Zustand Leistung verbraucht. Nehmen wir an, daß für y_j=+1 die Anzahl der positiven Summanden viel größer ist als die der negativen Summanden (bzw. für y_j=-1 $\nu \gg \pi$) und somit $\pi+\nu \approx \pi$ (bzw. $\pi+\nu \approx \nu$) gilt. Aus (4.2.8) folgt:

$$R \geq \nu \cdot 25\ k\Omega \qquad (4.2.9)$$

Dieses Ergebnis ist mit der Gleichung (4.2.5) für die assoziative Matrix vergleichbar, solange $\nu \leq \log(m)$ ist. Ansonsten wird der Aufwand für ein Hopfield-Netz größer. Eine genaue Bestimmung von ν kann nur für einen konkreten Mustersatz erfolgen. Theoretisch ergibt sich der Extremfall entsprechend den Abschätzungen aus Abschnitt 3.2.3 für $|\pi-\nu|=1$. Dieser Fall ist für m>100 in Analogtechnik nicht zuverlässig realisierbar, denn er erfordert eine Reproduzierbarkeit der Bauelementeparameter mit einer hohen Genauigkeit (Toleranz kleiner 1%, (3.2.20)).

Die Ausgabefunktion kann für ein Hopfield-Netz von zwei hintereinander geschalteten Invertern berechnet werden (Abb. 4.2.1c). Der logische Schwellenwert Th=0 einer Verarbeitungseinheit entspricht dem Spannungs-

wert $U_B/2$ (Abb. 4.2.1.c). Es muß folglich nur bewertet werden, ob das Aktivierungspotential U_a größer bzw. kleiner als $U_B/2$ ist. Diese Bewertung kann von einem einfachen CMOS-Inverter mit der symmetrischen Schaltschwelle $U_B/2$ ausgeführt werden.

Die Realisierung von hochohmigen Widerständen in MOS-Technik ist entweder flächenintensiv oder bedarf zusätzlicher Prozeßschritte zur Herstellung hochohmiger integrierter Schichten. Wenn derartige Prozeßschritte nicht zur Verfügung stehen, werden aktive Schaltungselemente als Widerstände verwendet. Eine einfache Realisierung in Einkanal-MOS-Technik verwendet NMOS-Transistoren vom Anreicherungstyp anstelle der Widerstände (Abb. 4.2.2a). Für das Aktivierungspotential U_a einer Verarbeitungseinheit der assoziativen Matrix folgt mit der einfachen Theorie für MOS-Transistoren unter Vernachlässigung der Kanallängenmodulation und der Beweglichkeitsreduktion der Ladungsträger [58]:

$$I_D = \frac{1}{2}\frac{W}{L} k_n (U_{GS} - U_T)^2 \qquad \text{(Sättigungsbereich)}$$

$$= \frac{1}{2}\frac{W}{L} k_n (U_B - U_a - U_T)^2$$

$$I_{Da} = \frac{W_a}{L_a} k_n (U_{DS}(U_{GS} - U_T) - \frac{1}{2} U_{DS})^2 \qquad \text{(Triodenbereich)}$$

$$= \frac{1}{2}\frac{W_a}{L_a} k_n (2U_a (U_B - U_T) - U_a^2)$$

$k_n = \mu_n \cdot \varepsilon_o \cdot \varepsilon_{ox}/d_{ox}$ Transistorsteilheitskonstante;

$x_i = '1' \equiv U_B$, $x_i = '0' \equiv 0V$, $U_{Ref} = U_B$;

Es gilt: $l \cdot I_D = I_{Da}$

$$\rightarrow \quad U_a = (U_B - U_T) \cdot \left(1 - \sqrt{\frac{1}{l \cdot \beta_r + 1}} \right) \qquad (4.2.10)$$

$$\beta_r = \frac{W \cdot L_a}{L \cdot W_a}$$

Unter Vorgabe eines minimalen Spannungshubes ΔU_{min} ergibt sich für die maximale Stufenanzahl l_{max}:

$$\frac{dU_a}{dl} = \frac{\beta_r}{2}(U_B - U_T) \cdot \left(\frac{1}{(l \cdot \beta_r + 1)^{3/2}}\right) = \Delta U_{min} \quad \rightarrow$$

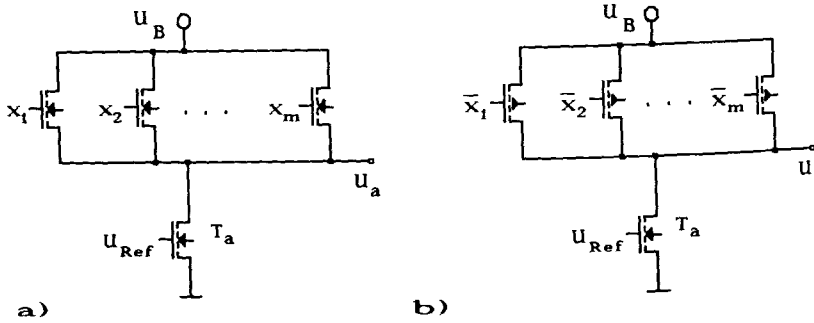


Abb. 4.2.2: Schaltungsalternativen für die Aktivierungsfunktion in MOS-Technik: einfache Realisierung mit NMOS-Transistoren (a) und mit PMOS-Transistoren (b) im Eingabezweig.

$$1 = \frac{1}{\beta_r}\left[\left(\frac{U_B - U_T}{2 \cdot \Delta U_{min}} \cdot \beta_r\right)^{3/2} - 1\right] \quad \rightarrow$$

$$1_{max} = 0.192 \cdot \frac{U_B - U_T}{\Delta U_{min}} \quad \text{für } \beta_r = 10.4 \cdot \frac{\Delta U_{min}}{U_B - U_T} \qquad (4.2.11)$$

Mit diesen Abschätzungen auf der Basis des einfachen Transistormodells erster Ordnung erhält man einen Zusammenhang zwischen der geforderten Auflösegenauigkeit ΔU_{min}, die von der nachfolgenden Schwellenwertschaltung (Differenzstufe) noch detektiert werden kann, der maximalen Stufenanzahl, bei der dieser Hub gerade noch gewährleistet ist, und dem dazu notwendigen Geometrieverhältnis β_r. Fordern wir z.B. mindestens 50mV für ΔU_a, so folgt eine Stufenanzahl von 15 und für das Verhältnis β_r ein Wert von 0.13. Das Aktivierungspotential $U_a(1_{max})$ beträgt ca. 1.7V. Gegenüber der Realisierung mit Operationsverstärker resultiert daraus eine Verringerung der Stufenanzahl um den Faktor 0.15.

Eine Erhöhung der Stufenanzahl ergibt die Verwendung von PMOS-Transistoren im Eingabezweig. Zum einen entfällt der Spannungsverlust $U_B - U_T$ (4.2.10) für die maximale Ausgangsspannung U_a. Zum anderen kommt es bei den PMOS-Transistoren, deren Source- und Substratanschluß auf dem gleichen Potential liegt, nicht zum Substrateffekt. Bei der Bestimmung des Aktionspotentials U_a sind drei Bereiche zu unter-

scheiden. Für $0 \leq U_a \lesssim 1V$ befinden sich die PMOS-Transistoren in Sättigung und der NMOS-Transistor im Anlaufstromgebiet. Wird das Aktivierungspotential größer als 1V, werden auch die Transistoren in den Eingabezweigen im Anlaufstromgebiet betrieben. Dieser zweite Bereich mit $1V \lesssim U_a \lesssim 3V$ ist durch eine größere Änderung ΔU_a beim Zuschalten weiterer Eingabezweige gekennzeichnet (Abb. 4.2.3a). Für $3V \lesssim U_a \lesssim U_B$ befindet sich der NMOS-Transistor in Sättigung und die PMOS-Transistoren im Anlaufstromgebiet. Das Aktivierungspotential läßt sich in diesem Bereich wie folgt berechnen:

$$1 \cdot I_{Dp} = I_{Da}$$

$$\rightarrow \quad U_a = U_T + (U_B - U_T)\sqrt{1 - \frac{1}{\beta_r \cdot 1}} \qquad (4.2.12)$$

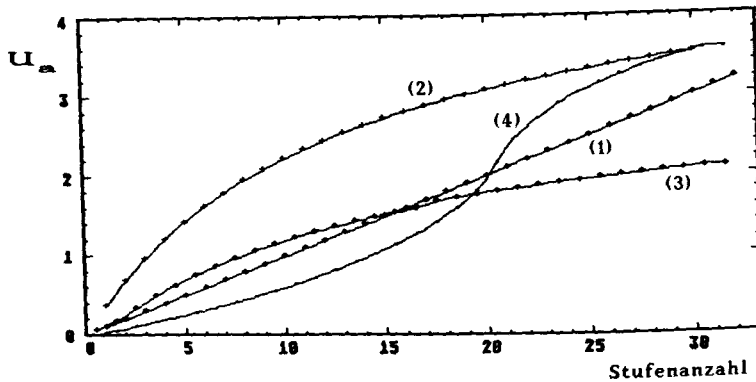
$$\beta_r = \frac{k_p \cdot W_p \cdot L_p}{k_n \cdot L_n \cdot W_n}$$

Aus dieser vereinfachten Abschätzung (4.2.12) folgt mit ΔU_{min}=50mV und β_r=0.1 eine Stufenanzahl von $1 < 25$ ($U_a \approx 4V$). Einen Vergleich der verschiedenen Schaltungsvarianten hinsichtlich der berechneten Aktivierung ist in Abb. 4.2.3 und Tabelle 4.2.1 zusammengefaßt.

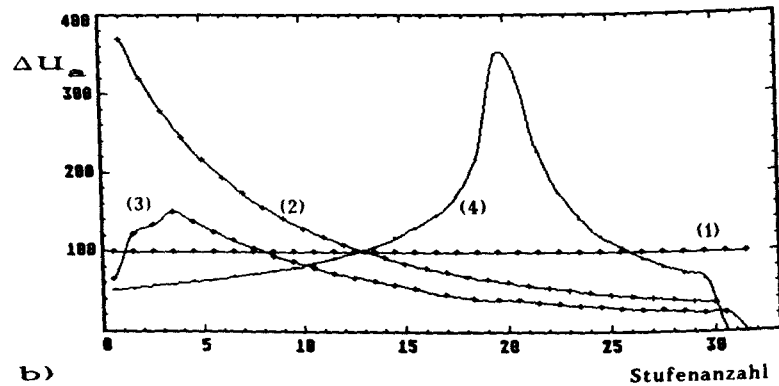
Die erzielten Ergebnisse für den Einsatz von MOS-Transistoren zur Bestimmung der Aktivierung lassen sich im wesentlichen auf das Hopfield-Netz übertragen. Die Dimensionierung der Transistoren in den Eingabezweigen eines Hopfield-Netzes wird von der Forderung bestimmt, daß das Aktivierungspotential U_a für die gleiche Anzahl von positiven und negativen Summanden (π=ν) $U_B/2$ betragen sollte.

	Operations-verstärker	Spannungs-teiler	NMOS	PMOS
I_{max}	50	< 13	< 8	< 30
r, β_r	r = 50	r = 12.5	$\beta_r \approx 0.25$	$\beta_r \approx 0.04$

*Tabelle 4.2.1: **Vergleich der vorgestellten Schaltungsalternativen hinsichtlich der maximalen Stufenanzahl und des geforderten Widerstandsverhältnisses r bzw. Geometrieverhältnisses β_r (ΔU_{min}=100mV).***



a)



b)

Abb. 4.2.3: Ergebnisse der Schaltungssimulation (BONSAI [59]) ausgewählter Realisierungsalternativen für die gewichtete Summe der Eingabesignale einer Verarbeitungseinheit: Aktivierungspotential U_a (a) und Aktivierungsänderung ΔU_a (b) in Abhängigkeit von der Stufenanzahl l.

(1) mit Widerständen und Operationsverstärker (Abb. 4.2.1a);
(2) einfaches Widerstandsnetzwerk (Abb. 4.2.1b);
(3) nur mit NMOS-Transistoren (Abb. 4.2.2.a)
(4) mit PMOS-Transistoren im Eingabezweig (Abb.4.2.2b).

Werden ausschließlich NMOS-Transistoren verwendet, dann folgt aus (4.2.10) für das Verhältnis β_r:

$$U_a(\pi=\nu) = U_B/2 \quad \rightarrow \quad \beta_r = \frac{W^+ L^-}{L^+ \cdot W^-} = 6 \qquad (4.2.13)$$

Aus entsprechenden Überlegungen folgt bei der Verwendung von PMOS-Transistoren für die positiven und NMOS-Transistoren für die negativen Summanden (Abb. 4.2.4a) ein Verhältnis von:

$$\beta_r = \frac{k_p \cdot W^+ \cdot L^-}{k_n \cdot L^+ \cdot W^-} = 1 \qquad (4.2.14)$$

Den Einfluß einer Variation von β_r auf die berechnete Aktivierung für unterschiedliche Verhältnisse π/ν verdeutlicht Abb. 4.2.4b. Für ein β_r=1.5 bekommen die positiven Summanden ($w_{ij} \cdot y_j > 0$) ein höheres Gewicht. Das Aktivierungspotential U_a ist bereits für $\pi/\nu > 0.5$ größer als $U_B/2$ und die Ausgabe y_j der Verarbeitungseinheit wird positiv ('+1'). Für β_r=0.75 haben die negativen Summanden ($w_{ij} \cdot y_j < 0$) ein höheres Gewicht und die Verarbeitungseinheit wird erst für $\pi/\nu > 1.3$ aktiv ('+1').

Die aufgeführten Abschätzungen beschreiben die einfachsten Realisierungsalternativen für eine Berechnung der Aktivierungsfunktion in analoger Schaltungstechnik. Durch schaltungstechnische Maßnahmen, wie z.B. Stromspiegelschaltungen [60,61], lassen sich Verbesserungen in der

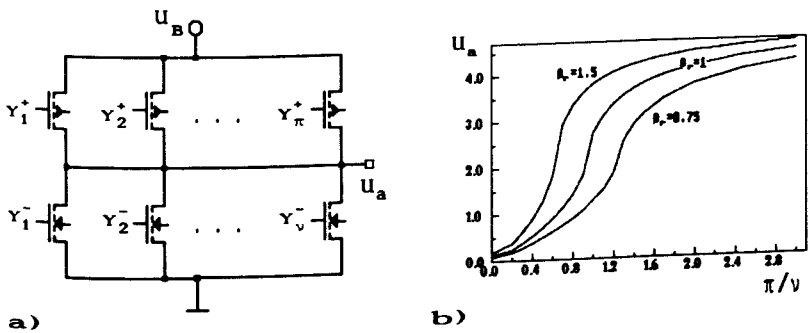


Abb. 4.2.4: Vereinfachte CMOS-Lösung für die Aktivierungsfunktion einer Verarbeitungseinheit des Hopfield-Netzes (a) und Abhängigkeit des Aktivierungspotentials U_a von dem Verhältnis π/ν für verschiedene β_r (b).

Berechnung erzielen. Der Aufwand pro Verbindungselement erhöht sich aber um weitere Transistoren und gegebenenfalls um zusätzliche Verbindungsleitungen und Referenzspannungen. Auch mit diesen Verbesserungen wird unter Berücksichtigung der Abschätzungen für die maximale prozentuale Abweichung Δ_{min} aus Abschnitt 3.2.3 deutlich, daß eine Realisierung der Aktivierungsfunktion in analoger Schaltungstechnik für l>100 Summanden und insbesondere für mehrwertige Gewichte sehr schwierig wird [62]. Im Fall einer assoziativen Matrix bedeutet das für spärlich kodierte Muster keine wesentliche Einschränkung, denn von den m Eingängen sind nur etwa ld(m) aktiv und die Gewichte nur zweiwertig {0,1}. Eine assoziative Matrix mit $m=10^5$ ist daher im Rahmen der analogen Schaltungstechnik möglich.

b) Dynamisches Verhalten

Das Aktivierungspotential U_a einer Verarbeitungseinheit der assoziativen Matrix ergibt sich im dynamischen Fall zu (Abb. 4.2.1b):

$$C \cdot \frac{\partial U_a}{\partial t} = \sum_{i=1}^{m} \frac{(x_i - U_a)}{R_i} - \frac{U_a}{R_a} \qquad (4.2.15)$$

Für die Zeitkonstante τ folgt:

$$\tau = \frac{C \cdot R \cdot R_a}{l \cdot R_a + R} \qquad (R_i=R,\ i=1,..,n) \qquad (4.2.16)$$

Die Kapazität C setzt sich aus der Leitungskapazität der Summationsleitung und der Eingangskapazität des Differenzverstärkers zusammen. Für eine 10µm breite und 1cm lange Aluminiumleitung kann man die Kapazität grob mit 5pF abschätzen. Mit R=210kΩ, R_a=8.3kΩ und l=20 errechnet sich eine Zeitkonstante τ von ca. 30ns. Die Komparationszeit einer einfachen Differenzstufe liegt in der Größenordnung von 100ns [63], so daß eine Verarbeitungseinheit einer assoziativen Matrix die Aktivierungs- und Ausgabefunktion in ca. 200ns berechnen kann.

Aufgrund der Rückkopplung wird die Dynamik eines Hopfield-Netzes in analoger Schaltungstechnik mit gekoppelten Differentialgleichungen beschrieben. Für die in Abb. 4.2.5 gezeigte Verschaltung der Verarbeitungseinheit folgt:

$$C \cdot \frac{\partial U_a^j}{\partial t} = \sum_{i=1}^{n} \frac{(y_i - U_a^j)}{R_{ij}} \qquad (4.2.17)$$

$$\text{mit } y_j = \alpha(U_a^j) \qquad j = 1, \ldots, n$$

Die Energie des aktuellen Zustandes des Netzwerkes im Sinne der energetischen Beschreibung nach (2.3.8) ergibt sich für kontinuierliche Ausgabewerte y_i zu:

$$E = -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} y_i \cdot y_j / R_{ij} + \sum_{i=1}^{n}\left(\sum_{j=1}^{n} 1/R_{ij}\right)\int_{0}^{Y_i}\alpha^{-1}(y)\,dy \qquad (4.2.18)$$

mit $\partial E/\partial t \leq 0$; $\partial E/\partial t = 0 \rightarrow \partial y_i/\partial t = 0$, $y_i \in [-1, +1] \subset \mathbb{R}$

Die Stabilität des Netzwerkes ist entsprechend Abschnitt 2.3.3 für eine symmetrische Verbindungsmatrix mit $w_{ii}=0$ garantiert. Mit einer genügend hohen Verstärkung der Ausgangsschaltung besitzt dieses Netzwerk die gleichen stabilen Zustände wie das in Abschnitt 2.3.3 definierte Hopfield-Netz mit statistischer Ausgabefunktion [39,64].

Eine genaue Bestimmung der Zeit bis zum Erreichen eines stabilen Zustands hängt wieder von den gespeicherten Mustern und dem Eingabemuster ab. Eine sehr grobe Abschätzung orientiert sich an der Differenz $|\pi-\nu|$ im stabilen Endzustand (vgl. (4.2.8)):

$$\tau = \frac{R \cdot C}{|\pi - \nu|} \qquad (R=R_{ij} \quad i,j=1,\ldots,n) \qquad (4.2.19)$$

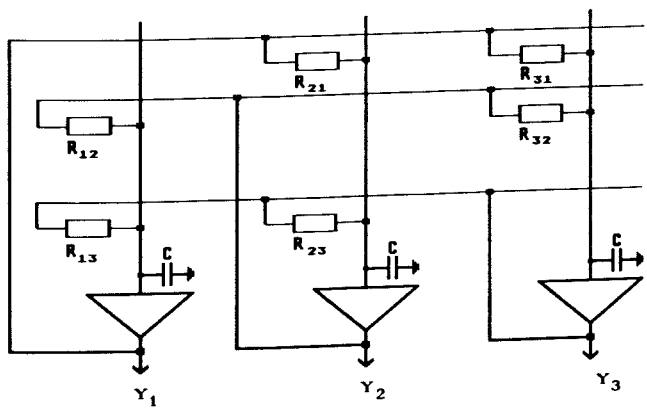


Abb. 4.2.5: Schematische Verschaltung für ein Hopfield-Netz mit Widerständen als Verbindungselemente.

Ist der zweite Term in (3.2.24) groß (π-ν klein), dann wird auch die Assoziationszeit länger sein. Im Mittel wird sie für ein Hopfield-Netz nur unwesentlich länger als die einer assoziativen Matrix sein. Die Assoziationszeiten divergieren allerdings für verschiedene Muster und können sich um den Faktor 10 unterscheiden [65].

Es bleibt ferner noch zu beachten, daß eine Verarbeitungseinheit nicht nur (n-1) Eingaben erhält, sondern auch mit n Verarbeitungseinheiten verbunden ist. Die kapazitive Last, die von einem Ausgang getrieben werden muß, ergibt sich aus einer ca. 1cm langen Aluminium-Leiterbahn und n Transistor-Eingängen (Gates). Entsprechendes gilt für die Eingabequellen einer assoziativen Matrix. Damit diese kapazitive Last so klein wie möglich gehalten wird, empfiehlt es sich, die Widerstände bzw. die Transistoren in den Eingabezweigen durch zusätzliche Transistoren mit Minimalmaßen zu schalten (Abb. 4.2.6). Der zusätzliche Flächenaufwand ist gering und die kapazitive Last in diesem Fall minimal. Sie kann für n=1000 und W/L=3µm/3µm grob mit 10pF abgeschätzt werden. In Abhängigkeit von der geforderten Zeitkonstanten τ muß eine geeignete Treiberschaltung aus der CMOS-Technik oder der BiCMOS-Technik ausgewählt werden [66].

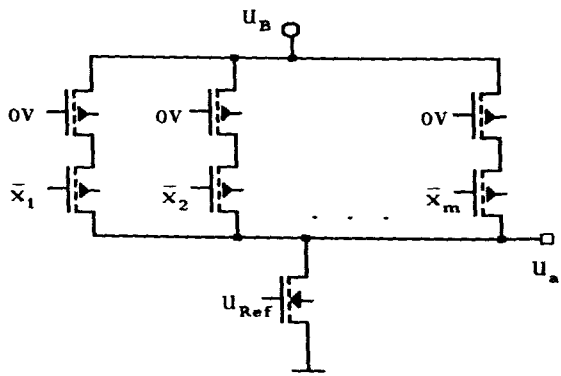


Abb. 4.2.6: Verwendung eines Schalttransistors mit Minimalmaßen im Eingabezweig zur Reduzierung der Eingangskapazität eines Verbindungselementes.

4.2.2 Geschaltete Kapazitäten

Anstelle der Widerstände können Kapazitäten zur Realisierung der Aktivierungsfunktion herangezogen werden. Eine dem Stromsummierer entsprechende Umsetzung ist die Verwendung von geschalteten Kapazitäten (switched capacitors). Das Schaltungsprinzip einer derartigen Verarbeitungseinheit ist in Abb. 4.2.7 gezeigt [67]. Die Berechnung der Aktivierung erfolgt zeitdiskret in drei sich nicht überlappenden Taktschritten. Mit dem Takt Φ_1 werden die aktuellen Eingabewerte $x_i(t)$ auf die Kapazitäten C_i, die den Gewichten w_i entsprechen, übertragen. Bevor die Auswertung der Eingaben erfolgt, wird mit dem Takt Φ_2 der Ausgang y(t) zurückgesetzt. Die Bestimmung der Aktivierung a(t) und des neuen Ausgabewertes y(t) vollzieht sich während des Taktes Φ_3 durch einen Ladungsausgleich der über den Knoten A (virtuelle Masse) gekoppelten Kapazitäten C_i und C_a. Die Ausgabefunktion wird im einfachsten Fall von einem Inverter, im aufwendigsten Fall von einem Operationsverstärker bestimmt.

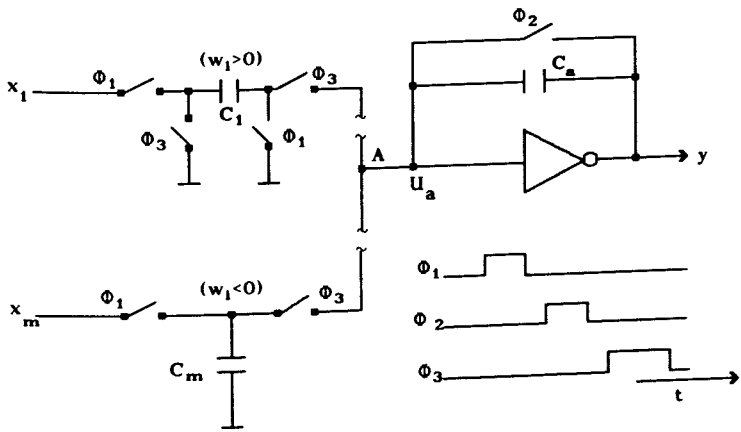


Abb. 4.2.7 Verarbeitungseinheit mit geschalteten Kapazitäten (nach Tsividis/Anastassiou [67]).

Die Bestimmung der Kapazitätsverhältnisse muß entsprechend den Überlegungen aus Abschnitt 4.2.1 zu den Widerstandsverhältnissen geschehen. Der Vorteil dieser Schaltungstechnik liegt darin, daß nur dynamische Verlustleistung anfällt. Eine Mindestgröße der Kapazitäten aufgrund von Verlustleistungsbeschränkungen (4.2.5) wie bei den Widerständen gibt es daher nicht. Eine untere Schranke für die Größe der Kapazitäten wird von

der Alpha-Teilchen-Empfindlichkeit [68] und der Auflösegenauigkeit der nachfolgenden Schaltungskomponenten bestimmt. Orientiert man sich an der Mindestladung einer dynamischen 1 Mega-Bit-Speicherzelle [69] von 200fC, so folgt für C_{min}:

$$C_{min} > Q_{min}/U_B = 200fC/5V = 40\ fF. \qquad (4.2.20)$$

Ein Nachteil dieser Schaltungsvariante ist der zusätzliche Aufwand für drei zeitlich festgelegte Takte Φ_1-Φ_3. Für ein Hopfield-Netz hat das zur Folge, daß die Rückkopplung nur zeitlich-diskret ausgeführt werden kann und somit bedeutend langsamer ist.

Eine interessante Perspektive für die Analogtechnik bietet die Realisierung von assoziativen Netzwerkmodellen mit Impulskodierung der Ein- und Ausgabesignale, die dem biologischen Vorbild näherkommen (vgl. 2.1). Einen schematischen Aufbau zeigt Abb. 4.2.8. Die synaptischen Verbindungen sind ähnlich aufgebaut wie die eines Widerstandsnetzwerkes. Das Aktivierungspotential entspricht der am Knoten A aufintegrierten Ladung, die durch Impulse an einem positiven (negativen) Gewicht erhöht (verringert) wird. Überschreitet das Aktivierungspotential den Schwellenwert U_{Th}, so wird vom Impulsgenerator ein Ausgabeimpuls erzeugt und gleichzeitig das Aktivierungspotential wieder auf das "Ruhepotential" (Masse) abgesenkt. Die Verarbeitungseinheit muß nun erneut aktiviert werden. Die Berechnung der Ausgabefunktion ist somit aufwendiger als die der bisherigen Schaltungskonzepte.

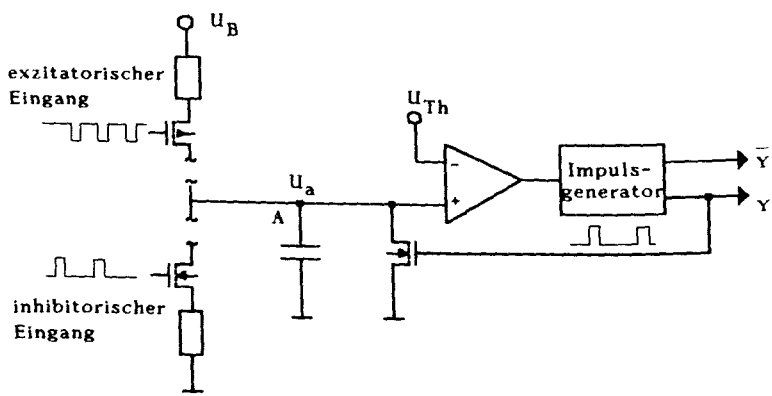


Abb. 4.2.8 : Asynchrone Verarbeitungseinheit mit Impulskodierung der Ein-/Ausgabesignale (nach Murray/Smith [70]).

An integrierten bzw. diskret aufgebauten Schaltungen ist die grundsätzliche Funktionsfähigkeit dieses Schaltungsprinzips nachgewiesen worden [71-73]. Der Vorteil dieser Konzepte liegt in der asynchronen Arbeitsweise, die die Voraussetzung für ein Ausnutzen der raum-zeitlichen Parallelität in neuronalen Netzen ist und keine globalen Steuertakte erfordert. Die raum-zeitliche Parallelität erscheint wichtig für die Verarbeitung von akustischen und visuellen Informationen wie Sprache oder Bilder.

Die Frage, welche Zeitkonstanten und wieviele Impulse notwendig sind, um eine Verarbeitungseinheit zu aktivieren, bleibt bisher noch unbeantwortet. Zur Zeit fehlen modelltheoretische Aussagen über diese charakteristischen Systemgrößen, die den Schaltungsentwurf eines größeren Netzwerkes konkretisieren könnten. Bezogen auf die Stromsummation ist eine genauere oder schnellere Berechnung der Aktivierungs- oder Ausgabefunktion sowie eine flächengünstigere Integration mit beiden hier behandelten Schaltungskonzepten derzeit sicherlich nicht möglich. Aus diesen Gründen wird bisher überwiegend die Stromsummation zur Berechnung der Aktivierungsfunktion in analoger Schaltungstechnik verwendet [74-78].

4.3 Realisierung der Verbindungselemente

Verbindungselemente kann man grob in festverdrahtete, programmierbare und adaptive Verbindungen einteilen (Abb. 4.3.1), die entweder für ein digitales oder analoges Schaltungskonzept bestimmt sind. Bei festverdrahteten Verbindungen wird die Verbindungsstruktur des Netzwerkes entweder bereits zur Herstellung oder während eines einmaligen, irreversiblen elektrischen Programmiervorganges festgelegt. Die Gewichtswerte von programmierbaren Verbindungselementen können dagegen jederzeit neu festgelegt und eingespeichert werden. Adaptive Verbindungselemente enthalten die Adaptationsregel und können in Abhängigkeit der anliegenden Signale selbständig ihren Gewichtswert anpassen und sind daher zwangsläufig mehrwertig.

Ausgehend von einfachen binären Verbindungselementen, die für die assoziative Matrix benötigt werden und deren Realisierungsalternativen sich aus den herkömmlichen Speichertechnologien (ROM, RAM, EEPROM) ergeben, werden Erweiterungen zu mehrwertigen und adaptiven Verbindungen aufgezeigt. Die Bewertung der verschiedenen Alternativen erfolgt hinsichtlich der möglichen *Verbindungsdichte*, die der Anzahl von

Verbindungselementen $*w_{ij}$ pro Fläche entspricht:

$$\text{Verbindungsdichte VD} = \frac{*w_{ij}}{cm^2} \qquad (4.3.1)$$

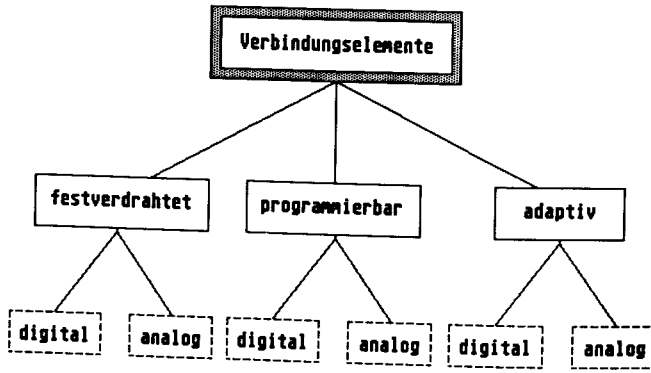


Abb. 4.3.1: Realisierungsalternativen für Verbindungselemente.

4.3.1 Festverdrahtete Verbindungselemente

Die größte Dichte an Verbindungselementen auf einer integrierten Schaltung erhält man mit festverdrahtetenVerbindungen. Es bieten sich Transistoren, Widerstände oder Kapazitäten an. Die Auswahl hängt von der zur Verfügung stehenden Technologie ab. Für den Standard-CMOS-Prozeß an der Universität Dortmund ist die Verwendung von Transistoren angebracht. Ein Beispiel für einen 2x2-Block von festverdrahteten Verbindungselementen für eine assoziative Matrix zeigt Abb. 4.3.2. Mit der Referenzspannung U_{Ref} kann in diesem Beispiel die Gewichtung der Verbindungstransistoren, die entsprechend Abschnitt 4.2.1 dimensioniert werden müssen, verändert werden. Der Kanalwiderstand eines Transistors im Triodenbereich kann grob abgeschätzt werden mit:

$$R = \frac{L}{W} \cdot \frac{1}{k_{p,n} \cdot (U_{GS} - U_T - U_{DS}/2)} \qquad (4.3.2)$$

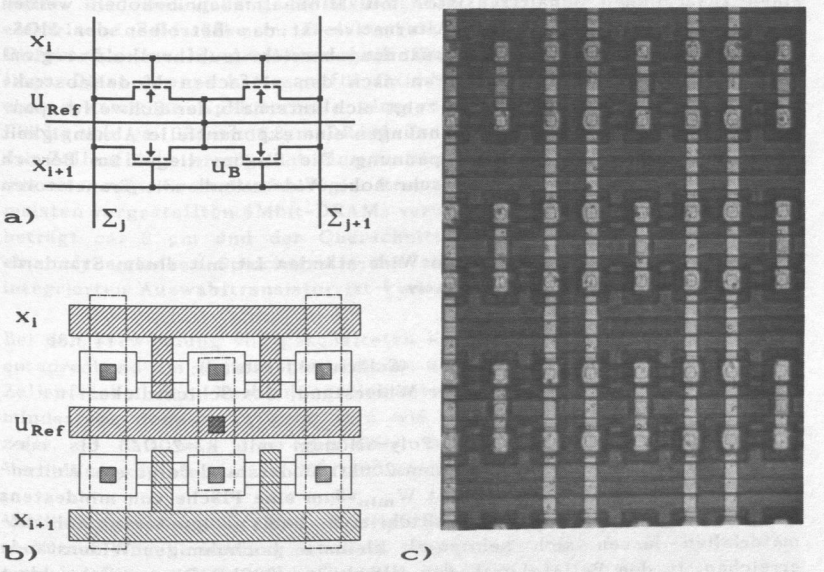


Abb. 4.3.2: Festverdrahtete Verbindungselemente mit NMOS-Transistoren: Schaltbild (a) und Layout (b) eines 2x2-Verbindungsblocks; c) Ausschnitt aus einer integrierten Verbindungsmatrix.

Für einen 200kΩ Widerstand errechnet sich mit U_{DS}=2V und U_{GS}=5V ein Geometrieverhältnis von W/L=1/24 für einen NMOS-Transistor, bzw. W/L=1/9 für einen PMOS-Transistor. Ein Transistor mit W/L=1 hat unter den gleichen Betriebsbedingungen einen Widerstand von 8.5kΩ (NMOS) bzw. 22.7kΩ (PMOS).

Für den Dortmunder 3µm CMOS-Prozeß läßt sich eine minimale Zellenfläche pro Verbindungselement von ca. 400µm² [63] und damit eine maximale Zellendichte von $25 \cdot 10^4/cm^2$ erzielen. Mit den sich in der Entwicklung befindenden Sub-µ-Technologien kann eine Zellenfläche von kleiner als 10µm² erreicht werden, entsprechend einer Zellendichte von mehr als $10^7/cm^2$.

Ein Nachteil dieser einfachen Verbindungsstruktur ist die höhere Gatekapazität von Verbindungstransistoren mit W/L≠1, die eine Erhöhung der ohnehin großen kapazitiven Belastung der Eingabequellen um eine Größenordnung bewirkt. Dieser Nachteil kann auf Kosten der Zellenfäche durch

einen zusätzlichen Schalttransistor mit Minimalmaßen behoben werden (Abb. 4.2.6). Eine interessante Alternative ist das Betreiben der MOS-Transistoren im Unterschwellenspannungsbereich (subthreshold region) [79], in dem die MOS-Transistoren nach den einfachen Modellabstraktionen gesperrt sind. Tatsächlich zeigt sich unterhalb der Schwellenspannung für genügend kleine Gatespannungen eine exponentielle Abhängigkeit des Drainstromes von der Gatespannung. Die Ströme liegen im Bereich von einigen Nanoampere, so daß sehr hohe Widerstände mit Transistoren erzielt werden können.

Die Realisierung von hochohmigen Widerständen ist mit einem Standard-CMOS-Prozeß relativ flächenintensiv:

$$R = R_{\square} \cdot L/W \qquad (4.3.3)$$

mit: $R_{\square} = \rho/d$ (Schichtwiderstand);
ρ = spezifischer Widerstand, d = Schichtdicke.

Schichtwiderstände können aus Poly-Silizium mit $R_{\square}=20\Omega/\square$ bis max. $10k\Omega/\square$ hergestellt werden. Für einen $200k\Omega$ Widerstand folgt ein Weiten-/Längenverhältnis W/L>20 und mit $W_{min}=3\mu m$ eine Fläche von mindestens $180\mu m^2$. Mit zusätzlichen Prozeßschritten und geeigneteren Schichtmaterialien lassen sich sehr viel kleinere hochohmige Widerstände erreichen. In den Bell-Laboratorien (USA) sind $200k\Omega$ Dünnschichtwiderstände aus amorphen Silizium (α-Si:H) mit einer Fläche von $4\mu m^2$ hergestellt worden [75]. Die getesteten Widerstände zeigten eine ausreichende Linearität und die Variation der Widerstandswerte auf einem Chip ist kleiner als 5%. Mit diesen Widerständen läßt sich eine Verbindungsdichte von ca. $10^7/cm^2$ erreichen. Für eine Sub-μ-Technologie läßt sich eine Verbindungsdichte von ca. $10^9/cm^2$ abschätzen [75].

Die Programmierung des Widerstandsnetzwerkes erfolgt am Ende der Herstellung mit einem Elektronenstrahlschreiber, d.h. die Verbindungsmatrix wird bei der Herstellung fest eingeschrieben. Eine Erweiterung zu elektrisch programmierbaren Verbindungen, bei denen die binären Verbindungen durch einen elektrischen Impuls irreversibel gesetzt werden können, wird in [80] beschrieben.

Kapazitäten müssen gemäß Gleichung (4.2.20) mindestens 40 fF betragen, woraus sich für eine Oxiddicke $d_{cox}=$ 40nm (bzw. 10nm) die benötigte Fläche A_C errechnen läßt:

$$A_C = \frac{C \cdot d_{cox}}{\varepsilon_o \cdot \varepsilon_{ox}} \approx 49\ \mu m^2 \quad (\text{bzw. } 12.3\ \mu m^2) \qquad (4.3.4)$$

Für Kapazitäten sind die Schichtdicke des Kondensatoroxids und das verwendete Dielektrikum die wesentlichen technologieabhängigen Para-

meter. Die Verringerung der Oxiddicke ist durch die maximal zulässige elektrische Feldstärke im Dielektrikum begrenzt, die für gebräuchliche Dielektrika der Integrationstechnik (SiO_2, Si_3N_4) bei ca. 8-10 MV/cm liegt. Diese Grenze hat man für Oxiddicken von ca. 10nm erreicht. Für eine dynamische Speicherzelle eines 1MBit-Speichers ergibt sich eine Fläche von ca. 35μm^2 (1.2μm DRAM-Technologie [81]). Eine Verringerung der Zellenfläche ermöglicht eine vertikale Integration der Kondensatoren mit der sog. Grabentechnologie (trench technology) [69], die von den meisten vorgestellten 4Mbit-DRAMs verwendet wird. Die Tiefe des Grabens beträgt ca. 8 μm und der Querschnitt ca. 1.5x1.5 μm^2. Die Zellfläche einer dynamischen Speicherzelle mit einem vertikal über der Kapazität integrierten Auswahltransistor ist kleiner als 10μm^2 [82].

Bei der Verwendung von geschalteten Kapazitäten erhöht sich die Fläche entsprechend um weitere Transistoren und Taktleitungen (Abb. 4.2.7). Die Zellenfläche der Verbindungselemente mit Kondensator wird somit mindestens doppelt so groß sein wie die einer dynamischen Speicherzelle, so daß man grob geschätzt mit einer Sub-μ-Technologie eine Verbindungsdichte von ca. $10^6/cm^2$ erreichen kann.

Verbindungselemente eines Hopfield-Netzes mit $w_{ij} \in \{-1,0,+1\}$ müssen nicht nur eine Verbindung schalten, sondern auch eine Vorzeichenauswertung ausführen. Haben der Eingang und das Gewicht das gleiche Vorzeichen, dann ist der Beitrag des Verbindungselementes positiv, im anderen Fall negativ. Abb. 4.3.3 zeigt zwei Realisierungsmöglichkeiten, bei denen die Gewichte über Kontaktstellen programmiert werden. Abhängig davon, welcher der beiden Kontakte vorhanden ist, erhält man ein positives, negatives oder neutrales Gewicht. Der Flächenaufwand ist etwa doppelt so groß wie der einer einfachen binären Verbindung. Im Vergleich zur assoziativen Matrix reduziert sich die Speichereffektivität pro Fläche eines Hopfield-Netzes um die Hälfte.

Die Integration von $10^7/cm^2$ festverdrahteten, binären Gewichten ist mit heutigen Halbleitertechnologien möglich. Die Verbindungsmatrix kann entweder bereits zur Herstellung mit einem der letzten Prozeßschritte (Maskenprogrammierung, Elektronenstrahlschreiben) oder elektrisch irreversibel programmiert werden (z.B. Fuse- und Anti-Fuse-Technik [68]). Die Realisierung von mehrwertigen festprogrammierten Gewichten kann man mit verschiedenen Ansätzen erreichen. Die Gewichtswerte können z.B. durch unterschiedliche Geometrien der Bauelemente verändert werden. Dieses Vorgehen ist für eine Auflösegenauigkeit größer 4 Bit nicht sinnvoll, da die Geometrie linear in (4.3.2-4) eingeht und zu einem großen Flächenaufwand führt. Am günstigsten sind Lösungen, bei denen die Gewichtsänderungen unabhängig von den Abmessungen des Bauelementes erreicht werden können. Beispiele derartiger Bauelemente sind

Floating-Gate-Transistoren und MNOS-Transistoren. Diese Bauelemente eignen sich auch zur Realisierung von programmierbaren und adaptiven Gewichten und werden in Abschnitt 4.3.3 ausführlicher behandelt.

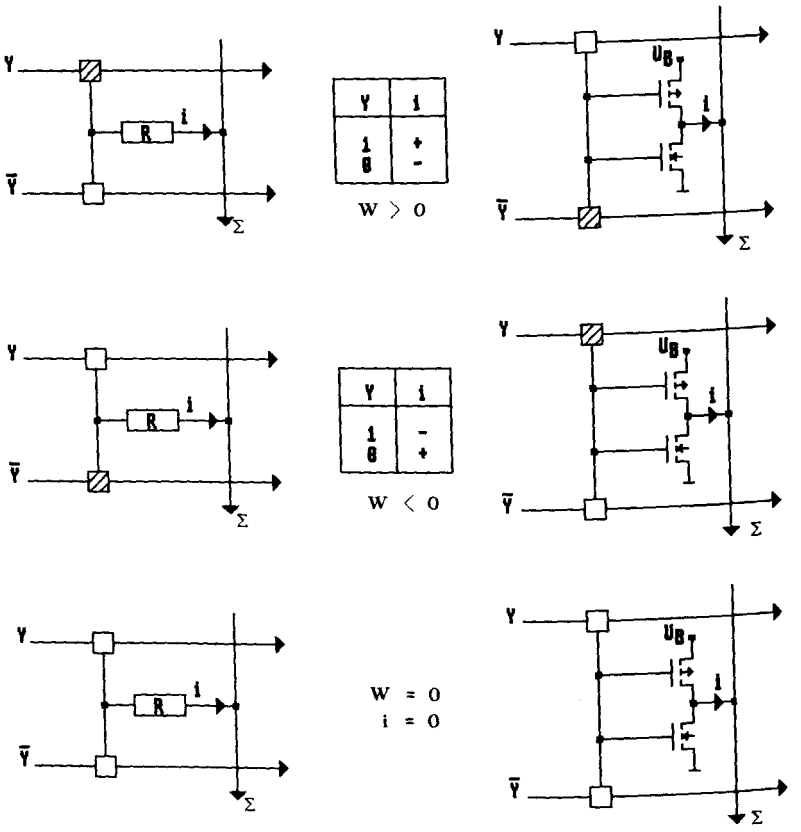


Abb. 4.3.3: Realisierungsalternativen für ternäre, festverdrahtete Verbindungselemente für ein Hopfield-Netz.

4.3.2 Programmierbare Verbindungselemente

Die Basis von programmierbaren Gewichten bilden statische und dynamische Speicherzellen. Eine statische 6-Transistor-Speicherzelle wird um einen Auswahltransistor T_A und einen Transistor T_W (bzw. Widerstand), der das eigentliche Gewicht darstellt, erweitert (Abb. 4.3.4). Will man die Programmierregel gemäß (2.3.7) in die Speicherzelle einbeziehen, werden mindestens zwei weitere Transistoren benötigt. Dieser Mehraufwand ist aber nicht immer gerechtfertigt, denn zur assoziativen Programmierung müssen alle z>m Musterpaare eingespeichert werden, während für das herkömmliche Einspeichern von n-Bit Worten nur m Schreibzyklen notwendig sind.

In Abb. 4.3.5 wird ein Layout einer derartigen Speicherzelle und eine integrierte Testschaltung mit einem 2x2-Speicherzellenblock gezeigt. Der Flächenaufwand dieser einfachen assoziativen Speicherzelle beträgt $100x130=13000\mu m^2$. Wenn anstelle der zusätzlichen Summenleitung Σ_j die Datenleitung D_j sowohl zur Programmierung als auch zur Summenbildung herangezogen wird, reduziert sich der Flächenaufwand auf $75x85=6375\mu m^2$ [63]. Mit dieser Speicherzelle läßt sich eine Verbindungsdichte von $1.6 \cdot 10^4/cm^2$ erreichen. Mit modernen Speichertechnologien werden derzeit statische 1MBit-Speicherbausteine produziert, 4MBit Speicherbausteine sind in der Entwicklung. Die Fläche einer statischen Speicherzelle beträgt für eine 1MBit-Speichertechnologie (0.7µm) zum Beispiel $5x12=60\mu m^2$ [83]. Assoziative Verbindungselemente sind ca. 1.3 mal größer, so daß sich mit einer derartigen Technologie eine Zellendichte von ca. $0.7 \cdot 10^6/cm^2$ erreichen läßt.

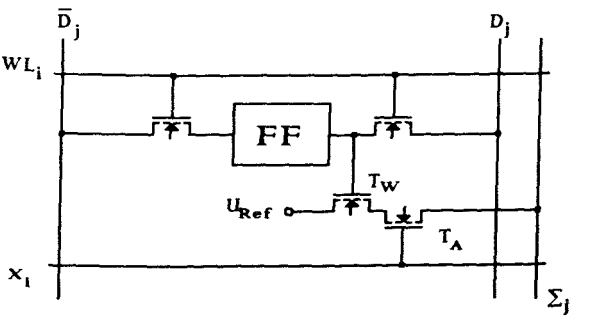


Abb. 4.3.4: Programmierbares Verbindungselement für eine assoziative Matrix mit statischer Speicherzelle (FF ≡ Speicher-Flip-Flop).

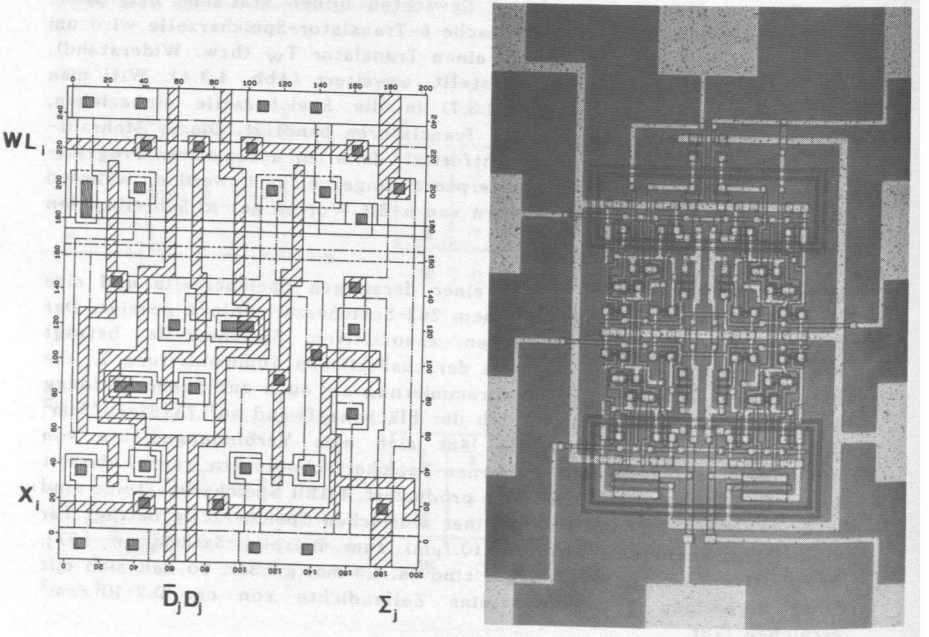


Abb. 4.3.5: Layout (a) eines binären assoziativen Verbindungselementes nach Abb. 4.3.4 und Photographie (b) einer integrierten Testschaltung mit einem 2x2 Speicherblock .

Mit einer dynamischen 3 Transistor-Speicherzelle (Abb. 4.3.6) läßt sich der Flächenaufwand beträchtlich reduzieren. Gegenüber einer festverdrahteten Verbindungszelle mit Auswahltransistor (Abb. 4.2.6) wird nur ein zusätzlicher Transistor zum Programmieren der Zelle benötigt. Für die Dortmunder 3µm CMOS Technologie ergibt sich eine Zellenfläche von ca. $2000\mu m^2$ und eine Verbindungsdichte von ca. $0.5 \cdot 10^5/cm^2$. Mit einer Sub-µ-Technologie kann man eine Zellenfläche von ca. $20\mu m^2$ und damit eine Verbindungsdichte von ca. $0.5 \cdot 10^7/cm^2$ erreichen.

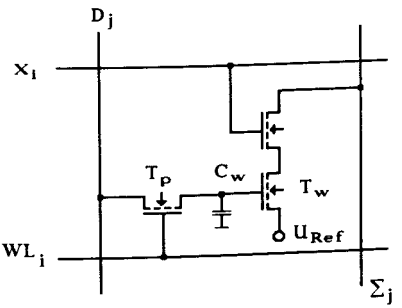


Abb. 4.3.6: Dynamische Drei-Transistor-Speicherzelle als Verbindungselement einer assoziativen Matrix.

Die Information wird in der Gatekapazität C_w des Transistors T_w gespeichert, deren Mindestgröße durch die Alphastrahlen-Empfindlichkeit bestimmt wird (4.3.4). Auf Grund eines parasitären Leckstromes am Transistor T_p wird die auf der Kapazität gespeicherte Ladung allmählich abgebaut. Alle Speicherzellen müssen daher in periodischen Abständen (1-100ms) durch einen Lese- und einen unmittelbar darauffolgenden Schreibzyklus regeneriert werden. Beim assoziativen Zugriff (X_i=1≡U_B) wird die gespeicherte Information zerstörungsfrei ausgelesen, d.h. innerhalb der Regenerationsperiode kann die Zelle beliebig oft benutzt werden. Eine dynamische Ein-Transistor-Speicherzelle ist nicht ohne weitere Vorkehrungen anwendbar, da der Auslesevorgang den Inhalt der Speicherzelle löscht. Eine Regeneration des Speicherzelleninhaltes durch sofortiges Zurückschreiben ist für eine verteilte Speicherung nicht möglich.

Zusammenfassend läßt sich für binäre programmierbare Verbindungselemente mit statischen Speicherzellen eine Verbindungsdichte von ca. $0.7 \cdot 10^6/cm^2$ erreichen. Mit dynamischen Speicherkonzepten ergibt sich eine Verbindungsdichte von etwa $0.5 \cdot 10^7/cm^2$, wobei sich der Verwaltungsaufwand und die Zugriffszeit durch die Regenerationsperioden erhöht. Für mehrwertige Gewichte reduziert sich die Zellendichte entsprechend der Bitanzahl b um mindestens den Faktor 1/b. Die Abbildung 4.3.7 zeigt beispielhaft zwei Realisierungsvorschläge für ternäre Verbindungselemente eines Hopfield-Netzes. Das von Graf/deVegvar [84] entwickelte Verbindungselement kann entweder exzitatorisch (RAM1=RAM2=0), inhibitorisch (RAM1=RAM2=1) oder abgeschaltet sein (RAM1=1, RAM2=0). Der Zustand (RAM1=0, RAM2=1) ist nicht zulässig (Abb. 4.3.7a). Das

Verbindungselement kann keine Vorzeichenauswertung vom Eingabesignal und dem gespeicherten Gewicht vornehmen. Das bedeutet, daß nur für $y_i=1\equiv U_B$ ein Verbindungselement Einfluß auf das Aktivierungspotential der Verarbeitungseinheit nimmt. Die Vorzeichenauswertung wird mit einem weiteren XOR-Gatter entsprechend dem Schaltungsvorschlag von Verleysen/Siriletti/Jespers [77] (Abb. 4.3.7b) realisiert. Der Zustand der Speicherzelle RAM2 bestimmt, ob das Verbindungselement ein Gewicht ungleich Null hat (RAM2=1). Mit der Speicherzelle RAM1 wird festgelegt, ob das Gewicht exzitatorisch (RAM1=1) oder inhibitorisch ist. Der Aufwand bei beiden Vorschlägen ist mindestens doppelt so groß gegenüber einem Verbindungselement der assoziativen Matrix, d.h die Speichereffektivität pro Fläche reduziert sich mindestens um den Faktor 1/2.

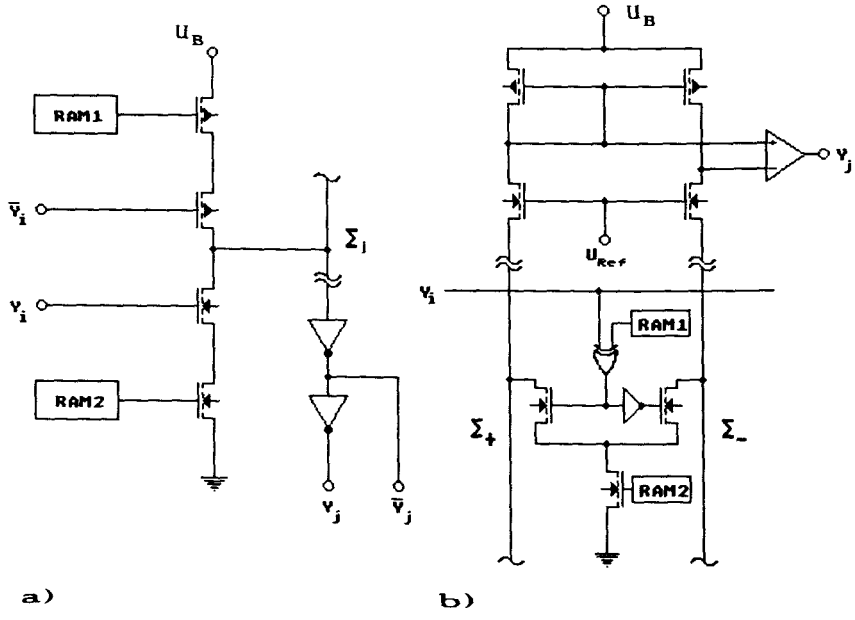


Abb. 4.3.7: Programmierbares Verbindungselement für ein Hopfield-Netz nach Graf/deVegvar [84] (a) und nach Verleysen/Jespers [77] (b).

4.3.3 Adaptive Verbindungselemente

Die Realisierung von *adaptiven* neuronalen ASICs erfordert für eine vollständig parallele Arbeitsweise die Integration der Adaptationsregel im Verbindungselement. Es kommen nur lokale Lernregeln in Frage (vgl. Abschnitt 2.2), denn eine Berücksichtigung von globalen Netzwerkzuständen führt zu einem großen Kommunikationsaufwand. In der Digitaltechnik bieten sich Zähler, Addierer und Schieberegister an, die aber nur eine geringe Verbindungsdichte von wenigen hundert Verarbeitungseinheiten pro cm^2 zulassen. Der beträchtliche Mehraufwand für das Lernen ist im Vergleich zu leistungsfähigen Emulatoren (vgl. Abschnitt 4.1) wiederum nicht gerechtfertigt.

a) Programmierbare nichtflüchtige Speicherzellen

Der Einsatz der analogen Schaltungstechnik bietet große Vorteile durch die Möglichkeit einer *funktionalen Integration* der Lernregel im Bauelement. Funktionale Integration bedeutet in diesem Zusammenhang das Ausnutzen von physikalischen Effekten im Bauelement zur Implementierung der geforderten Funktion. Bauelemente, deren physikalische Eigenschaften für eine funktionale Integration ausgenutzt werden können, sind z.B. elektrisch umprogrammierbare, nichtflüchtige Speicherzellen (EEPROM: electrically erasable and programmable read only memory). Nichtflüchtige, programmierbare Speicherzellen basieren auf der Grundstruktur eines MOS-Transistors, dessen Schwellenspannung U_T durch eine gezielte Änderung der elektrischen Ladung zwischen der Steuerelektrode (Gate) und dem Substrat beeinflußt werden kann. In der nichtflüchtigen Speichertechnik haben sich zwei Realisierungsformen herausgebildet. Bei einem Speichertransistor mit Doppelschichtisolator (MIOS: metal-isolator-oxide-semiconductor, MNOS: metal-nitride-oxide-semiconductor) wird die Ladung in Haftstellen im Gateisolator gespeichert [68]. Diese Haftstellen befinden sich in einer dünnen Schicht an der Grenzfläche zwischen den verwendeten Isolatorschichten (z.B. SiO_2 und Si_3N_4 bei einem MNOS-Transistor).

Der überwiegende Teil heutiger EEPROM-Speicher verwendet Speichertransistoren mit einer isolierten Gate-Elektrode (Floating-Gate, Abb. 4.3.8a). Zwischen der üblichen Gate-Elektrode und dem Substrat ist eine zusätzliche, durch SiO_2 vollständig isolierte Elektrode (i.a. aus Polysilizium) eingebracht (SIMOS: stacked-gate injection MOS, FLOTOX: floating gate tunnel oxide). Die auf dieser Elektrode gespeicherte Ladung verändert die Schwellenspannung und damit die Leitfähigkeit des Speichertransistors. Anhand des vereinfachten kapazitiven Ersatzschaltbildes (Abb. 4.3.8b) läßt sich der Einfluß der auf dem Floating-Gate (FG) gespeicherten Ladung Q_{FG} auf das Transistorverhalten bestimmen:

$$U_{FG} = \frac{Q_{FG}}{\Sigma C} + \frac{C_G}{\Sigma C} \cdot U_G + \frac{C_t + C_D}{\Sigma C} \cdot U_D + \frac{C_S}{\Sigma C} \cdot U_S \qquad (4.3.5)$$

mit $\Sigma C = C_G + C_D + C_t + C_S + C_{ox}$

$$\Rightarrow \quad I_D = \begin{cases} k_n \cdot \frac{W}{L} \cdot (U_{FG} - U_S - U_{To})^2 \\ k_n \cdot \frac{W}{L} \cdot (U_{DS} \cdot (U_{FG} - U_S - U_{To}) - 1/2 \cdot U_{DS}) \end{cases} \qquad (4.3.6)$$

Es ist U_{To} die Schwellenspannung des Transistors, wenn die Ladung auf der schwebenden Elektrode Null ist. Für die Schwellenspannung U_T, die an die Steuerelektrode angelegt werden muß, damit ein leitender Kanal im MOS-Transistor entsteht, folgt:

$$U_T = \frac{\Sigma C}{C_G} \cdot U_{TO} + \frac{\Sigma C - C_S}{C_G} \cdot U_S - \frac{C_D + C_t}{C_G} \cdot U_D - \frac{Q_{FG}}{\Sigma C} \qquad (4.3.7)$$

Das Aufladen der schwebenden Elektrode kann durch Elektronen mit hoher Energie (hot electrons) oder durch Ausnutzung des Tunneleffektes (Fowler-Nordheim-Tunneln) vorgenommen werden. Beide Verfahren finden ihre Anwendung, wobei die Injektion mit "heißen" Ladungsträgern bei

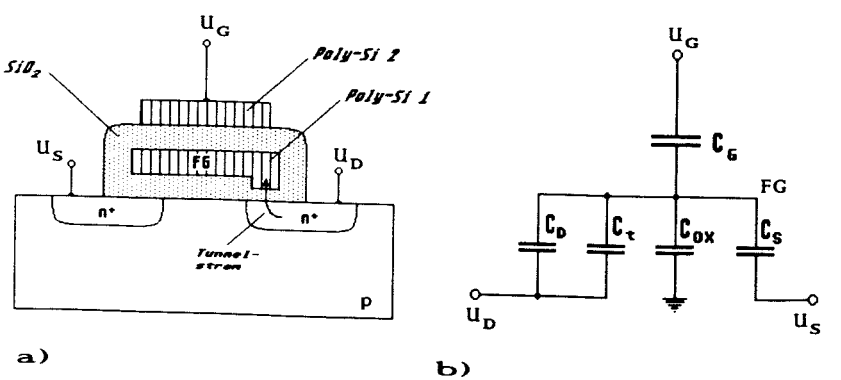


Abb. 4.3.8: Schematischer Aufbau eines nichtflüchtigen Speichertransistors mit einem Floating-Gate [68] (a) und ein vereinfachtes kapazitives Ersatzschaltbild (b).

heutigen EEPROMs an Bedeutung verloren hat. Realisierungen, die den Tunnelmechanismus ausnutzen, haben im allgemeinen einen kleineren Leistungsbedarf, eine höhere Programmiereffizienz und eine geringer ausfallende Schädigung der Transistorfunktionen [85]. Durch das Anlegen einer hohen positiven Spannung an die Steuerelektrode U_G bei an Masse liegenden Drain- und Source-Anschluß (U_S=U_D=0V) können Elektronen mit einer hohen Tunnelwahrscheinlichkeit durch die SiO_2-Potentialbarriere im Tunnelbereich auf die schwebende Elektrode gelangen. Diese negative Aufladung der schwebenden Elektrode bewirkt eine Erhöhung der Schwellenspannung U_T (4.3.7) und damit eine Verringerung der Leitfähigkeit. Eine Verringerung der Schwellenspannung U_T erreicht man durch eine positive Vorspannung des Drain-Anschlusses bei an Masse liegendem Source- und Steuergate-Anschluß, so daß eine hohe Tunnelwahrscheinlichkeit der Elektronen in das n^+-Diffusionsgebiet resultiert.

Die Höhe und Dauer des Programmierimpulses (U_p,t_p) bestimmt den Betrag der Schwellenspannungsänderung. Für die Ladung, die während einer Programmierdauer t_p auf die anfangs mit Q_o geladene schwebende Elektrode transferiert wird, gilt in erster Näherung die Fowler-Nordheim-Relation [86]:

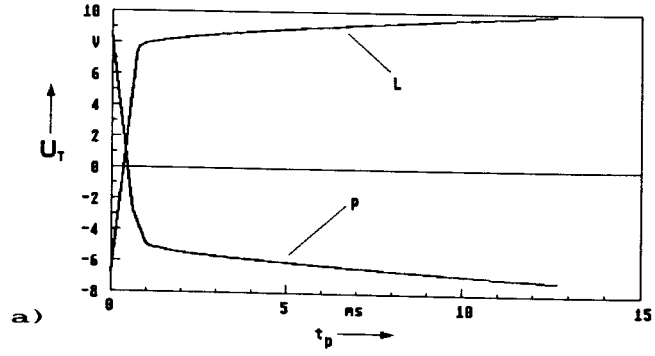
$$Q = Q_o + A_G \cdot c \cdot \int_0^{t_p} E_{ox}^2 \cdot \exp(-E_o/E_{ox})\, dt \qquad (4.3.8)$$

Mit A_G ist die wirksame Injektionsfläche für den Tunnelstrom bezeichnet, und E_{ox}= U_{FG}/d_{ox} ist das als homogen angenommene elektrische Feld zwischen dem Floating-Gate und dem Substrat (d_{ox} ist die Tunneloxiddicke). Für die konstanten Größen c und E_o gilt [87]:

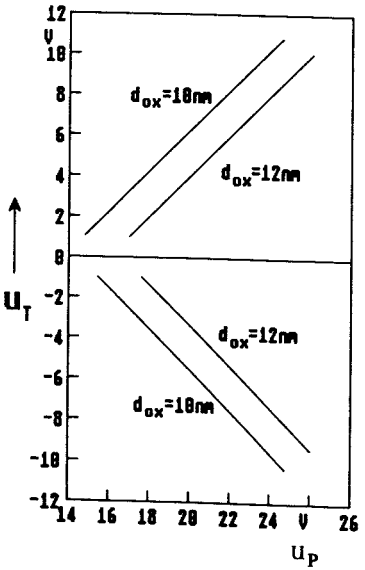
$$c = 2 \cdot 10^{-6}\ A/V, \quad E_o = 2.385 \cdot 10^8\ V/cm$$

Die qualitative Abhängigkeit der Schwellenspannungsverschiebung von der Impulshöhe und -dauer ist in Abb. 4.3.9 zusammengefaßt. Zur Speicherung der binären Zustände '0' und '1' will man eine möglichst schnelle und ausreichend hohe Schwellenspannungsverschiebung erreichen. Typische Werte für eine Schwellenspannungsverschiebung von 5V liegen bei U_p=20V und t_p=1ms. Für eine Speicherung von analogen Werten nutzt man kleinere Verschiebungen aus. Bei Verwendung eines konstanten Programmierimpulses (U_p=15V, t_p=0.1ms) haben Messungen an Floating-Gate-Zellen [88] eine exponentielle Programmierkurve für mehrere aufeinanderfolgende Programmierimpulse ergeben (Abb. 4.3.9c).

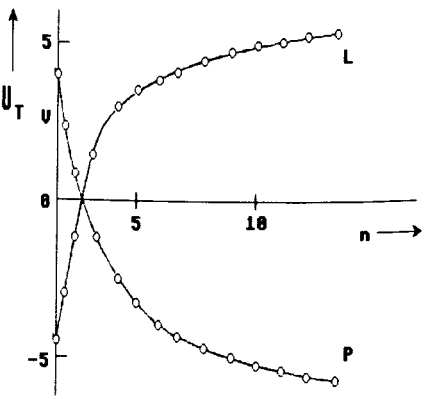
Die technologischen Schwierigkeiten bei der Realisierung von Floating-Gate-Transistoren liegen in der Erzeugung eines elektrisch stabilen dünnen Tunneloxides unterhalb der schwebenden Elektrode und einer spannungsfesten Isolationsschicht zwischen der schwebenden Elektrode und der Steuerelektrode [85].



a)



b)



c)

Abb. 4.3.9: Änderung der Schwellenspannung U_T eines Floating-Gate-Transistors in Abhängigkeit von der Programmierzeit t_p [87] (a), von der Programmierimpulshöhe U_p [87] (b) und von der Anzahl der Programmierimpulse n [88] (c) (L≡Löschen, P≡Programmieren).

Das elektrische Feld im Tunnelbereich muß etwa 10^7V/cm betragen, bevor die Elektronen mit einer genügend hohen Wahrscheinlichkeit die Potentialbarriere (3.2eV) durchtunneln. Um dies für vertretbare Programmierspannungen von ca. 15V zu ereichen, muß das Tunneloxid kleiner als 20nm dick sein. Die Tunnelwahrscheinlichkeit der Elektronen nimmt bei dünneren Gateoxiden zu (4.3.8), so daß kürzere Programmierzeiten und auch kleinere Programmierspannungen möglich sind. Herstellungsbedingte Variationen der geforderten Oxidschichtdicken führen zu einer Verschiebung der Programmierkennlinie (Abb. 4.3.9.b), so daß das Verhalten der Transistoren auf einem Chip um mehr als 10% voneinander abweichen kann. Messungen an SIMOS-Transistoren haben ferner ergeben, daß die Genauigkeit, mit der ein bestimmter Schwellenwert eingestellt werden kann, nur in einem Bereich von ±10% liegt [89]. Unerwünschte Degradationseffekte im dünnen Gateoxid begrenzen zudem die Anzahl der Programmierzyklen. Nach ca. 10^6 Schreib-/Löschzyklen verkleinert sich bei konstanten Programmierbedingungen die Schwellenspannungsverschiebung (Abb. 4.3.10). Die Degradation wird auf das Einfangen von Ladungsträgern im Oxid zurückgeführt, die das elektrische Feld im Tunnelbereich beeinflussen. Diese Eigenschaften sind beim Einsatz der Speichertransistoren mit schwebender Elektrode als Verbindungselement in assoziativen Netzwerken zu berücksichtigen.

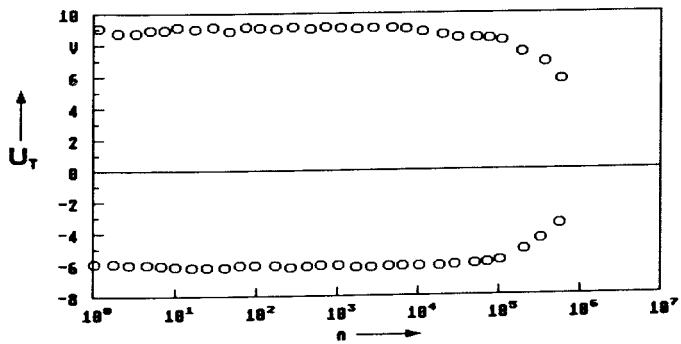
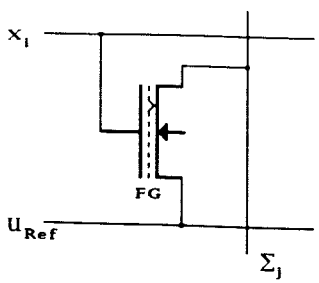


Abb. 4.3.10: Veränderung der Schwellenspannungsverschiebung eines Floating-Gate-Transistors in Abhängigkeit von der Anzahl der Programmier-und Löschzyklen (Degradationseffekt) [87].

b) Ein adaptives Verbindungselement mit Floating-Gate-Transistor

Mit einem Floating-Gate-Transistor läßt sich eine nichtflüchtige Ein-Transistor-Speicherzelle konstruieren, die sowohl als binäres, als auch als adaptives Verbindungselement genutzt werden kann (Abb. 4.3.11). Für eine assoziative Matrix werden nur binäre Gewichte benötigt. Der Vorteil eines Verbindungselementes mit Floating-Gate-Transistor gegenüber den in Abschnitt 4.3.2 vorgestellten programmierbaren Realisierungen liegt in der nichtflüchtigen Speicherung der Verbindungen. Nach dem Abschalten der Versorgungsspannung muß die Verbindungsmatrix nicht neu geladen werden, wie es bei den statischen bzw. dynamischen Speichertechniken der Fall ist. Der Degradationseffekt bedeutet für die assoziative Matrix keine Einschränkung, weil für jede Anwendung bzw. jeden Mustersatz die Verbindungselemente höchstens einmal gesetzt werden. Nachteile der Floating-Gate-Transistoren sind die größere Streuung der Leitwerte aktivierter Verbindungen sowie die relativ lange Programmierdauer. Aufgrund der nichtflüchtigen Speicherung ist die Matrix aber nur einmal zu programmieren. Neue Musterpaare lassen sich jederzeit hinzufügen. Die derzeit zu erreichende Verbindungsdichte mit Floating-Gate-Speicherzellen, die für eine 1µm EEPROM-Technologie eine Zellenfläche von ca. $13\mu m^2$ haben [90], liegt bei ca. $10^6/cm^2$.

Für die Anwendung der Floating-Gate-Speicherzelle als adaptives Verbindungselement wird die Aufladung der schwebenden Elektrode in kleineren Schritten durchgeführt (Abb. 4.3.9c). Eine negative Aufladung des Floating-Gates (Programmieren) entspricht einer Verringerung des Verbindungsgewichtes ($\Delta w_{ij} < 0$) und umgekehrt. Es bleibt zu beachten, daß die Schwellenspannung U_T des Speichertransistors gemäß (4.3.7) nicht kleiner als 0V wird. Der Transistor wird sonst selbstleitend, d.h.



Assoziieren: $U_{Ref} > 0V$.

x_i = '1' $\rightarrow$ $\Sigma_j = \Sigma_j + w_{ij}$

x_i = '0' $\rightarrow$ $\Sigma_j = \Sigma_j$

Programmieren: $U_{Ref} = 0V$

$x_i = U_P$, $\Sigma_j = 0V$ $\rightarrow$ $\Delta w_{ij} < 0$

$x_i = 0V$, $\Sigma_j = U_P$ $\rightarrow$ $\Delta w_{ij} > 0$

Abb. 4.3.11: Verbindungselement mit Floating-Gate-Speichertransistor.

er liefert auch für x_i='0'≡0V einen Beitrag zur Summenbildung. Mit einem zusätzlichen Auswahltransistor entsprechend Abb. 4.2.6 kann dieses Problem nicht auftreten und es verringert sich zudem die Eingangskapazität des Verbindungselementes (vgl. Abschnitt 4.2.1).

Die funktionalen Programmiereigenschaften der Floating-Gate-Transistoren (Abb. 4.3.9) können mit folgender Adaptationsregel nachgebildet werden ($t \in \mathbb{N}$):

$$w_{ij}^{t+1} = w_{ij}^{t} \cdot \exp(-f_1) + w_{max}(1 - \exp(-f_1)) \qquad \Delta w_{ij} > 0 \qquad (4.3.9a)$$

$$w_{ij}^{t+1} = w_{ij}^{t} \cdot \exp(-f_2) \qquad \Delta w_{ij} < 0 \qquad (4.3.9b)$$

Die Parameter f_1, f_2 beschreiben den Einfluß von Impulshöhe und -dauer des Programmierimpulses auf die Schwellenspannungsänderung. Sie bestimmen die Lerndynamik, mit der ein Verbindungselement sein Gewicht an vorgegebene Eingangssignale anpassen kann. Das maximale Gewicht, das ein Verbindungselement annehmen kann, ist w_{max} und entspricht der kleinsten einstellbaren Schwellenspannung U_T. Für $f_1=f_2>0.5$ erhält man den binären Fall, für $f_1=f_2<0.1$ werden bereits mehr als 20 Lernzyklen benötigt, um ein Gewicht $w_{ij}>0.9\,w_{max}$ zu erhalten (Abb. 4.3.12). Für kleine Werte der Parameter $f_1, f_2 < 0.1$ läßt sich die Adaptationsregel mit der Approximation $\exp(-x) \approx (1-x)$ wie folgt umformen:

$$w_{ij}^{t+1} = w_{ij}^{t} - f_1 \cdot w_{ij}^{t} + f_1 \cdot w_{max} \qquad (4.3.10)$$

$$\rightarrow \quad \Delta w_{ij} = f_1 \cdot (w_{max} - w_{ij}^{t}) \qquad \Delta w_{ij} > 0 \qquad (4.3.11a)$$

$$\text{bzw.} \quad \Delta w_{ij} = -f_2 \cdot w_{ij}^{t} \qquad \Delta w_{ij} < 0; \qquad (4.3.11b)$$

Die Herstellungstoleranzen der Bauelementeparameter können durch eine statistische Variation dieser Gewichtsänderungen berücksichtigt werden:

$$w_{ij}^{t+1} = w_{ij}^{t} + N(\Delta w_{ij}, \sigma^2) \qquad (4.3.12)$$

Die Gewichtsänderung $N(\Delta w_{ij}, \sigma^2)$ ist im allgemeinen eine normalverteilte Zufallsvariable mit dem Erwartungswert Δw_{ij} und der Standardabweichung σ. Mit den Gleichungen 4.3.9-12 läßt sich die Anwendbarkeit von Floating-Gate-Transistoren in einem adaptiven Speicherkonzept unter Berücksichtigung technologischer Randbedingungen überprüfen.

Grundsätzlich können alle Adaptationsregeln realisiert werden, die sich in Form von (4.3.9) ausdrücken lassen. Hinsichtlich der Programmierbedingungen für den Floating-Gate-Transistor bleibt zu beachten, daß sich

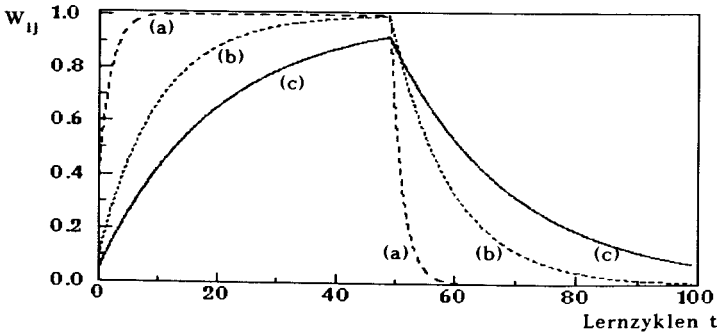


Abb. 4.3.12: Verlauf der Lernfunktion (4.3.9) für verschiedene Parameter f_1, f_2: (a) $f_1=f_2=0.5$; (b) $f_1=f_2=0.1$; (c) $f_1=f_2=0.05$; $w_{max}=1$.

die geforderten Gewichtsänderungen nicht unbedingt für alle Verbindungselemente parallel ausführen lassen. Mit der einfachen Hebb-Regel werden Verbindungselemente nur für $y_j=x_i='1'$ verstärkt und bleiben ansonsten unverändert (Abb. 4.3.13). Eine Anpassung der Gewichte kann mit dieser Regel für alle Verbindungen mit $x_i=0V$, $y_j=\Sigma_j=U_p$ und $U_{Ref}=0V$ (Abb. 4.3.11) parallel in einem Programmierzyklus ausgeführt werden. Für die Hopfield-Regel (Abb. 4.4.13b) ist das nicht der Fall. Zur Ausführung dieser Regel werden vier Programmierzyklen benötigt, weil die Änderungen der Verbindungselemente vom Vorzeichen der Eingabe x_i und der Ausgabe y_j abhängen. Für jede Kombination dieser Werte ist daher ein Programmierzyklus erfoderlich.

a)

X_i	Y_j	ΔW_{ij}
0	0	
0	1	
1	0	
1	1	+

b)

X_i	Y_j	ΔW_{ij}
-1	-1	+
-1	1	-
1	-1	-
1	1	+

c)

X_i	Y_j	ΔW_{ij}
0	0	
0	1	
1	0	-
1	1	+

Abb. 4.3.13: a) Einfache Hebb-Regel (assoziative Matrix);
b) Lernregel des Hopfield-Netzes;
c) Lernregel der adaptiven assoziativen Matrix.

Ein Beispiel für die Anwendung von adaptiven Verbindungselementen mit Floating-Gate-Transistoren ist die *adaptive assoziative Matrix* [91], die sich von der assoziativen Matrix (Def. 5) nur durch eine geänderte Lernregel und einen veränderten Wertebereich der Verbindungsgewichte unterscheidet (Abb. 4.3.13c):

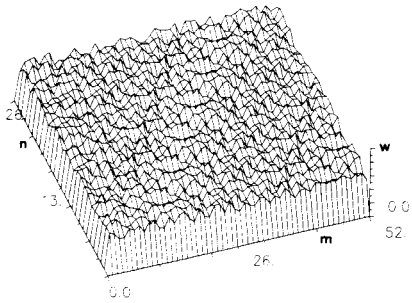
$$w_{ij}^{t+1} = \begin{cases} w_{ij}^{t} \cdot \exp(-f_1) + (1-\exp(-f_1) & \text{für } x_i=y_j=1; \\ w_{ij}^{t} \cdot \exp(-f_2) & \text{für } x_i=1 \text{ und } y_j=0; \\ w_{ij}^{t} & \text{sonst.} \end{cases}$$

$$w_{ij} \in [0,1] \subset \mathbb{R} \; ; \; w_{max}=1. \qquad (4.3.13)$$

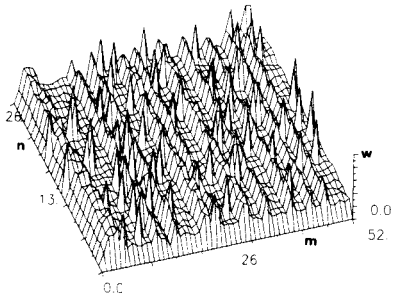
Die assoziative Matrix kann nur Verbindungen setzen, diese aber nicht wieder löschen. Das führt bei sehr vielen einzuspeichernden Mustern zu Überlagerungen in der Verbindungsmatrix und somit zu einer Abnahme der Speichereffektivität. Die kritische Grenze für den Füllungsgrad der Matrix liegt bei $p_{on}=0.5$ (vgl. Abschnitt 3.1). Mit der Lernregel gemäß (4.3.13) werden beim Einspeichern neuer Musterzuordnungen bestehende Verbindungen auch wieder "vergessen". Die alten Musterzuordnungen werden dabei nicht vollständig gelöscht, sondern "verblassen" allmählich mit jedem weiteren neu eingespeicherten Muster. Ein graduelles Verlernen kann dazu beitragen, daß die Verbindungsmatrix nicht überfüllt wird und dadurch die Anzahl der speicherbaren Musterzuordnungen im Mittel gleich bleibt.

Die Veränderung einer adaptiven Verbindungsmatrix beim Einspeichern von spärlich kodierten Musterpaaren ist in Abb. 4.3.14 zusammengefaßt. Durch ein mehrmaliges Anlernen der Musterpaare entwickelt sich eine Verbindungsmatrix, die die geforderten Musterzuordnungen erfüllt. Der Anfangszustand der Verbindungsmatrix ist für das Erlernen der Musterzuordnungen unbedeutend. Die Anzahl der notwendigen Lernschritte hängt von den Parametern f_1 und f_2 ab. Bei der Wahl dieser Parameter ist zu beachten, daß sich z.B. für spärlich kodierte Muster nur $1 \cdot k$ Gewichte pro Lernschritt erhöhen, während $1 \cdot (n-k)$ Gewichte verringert werden. Die Verbindungen verringern sich demnach viel häufiger, so daß in diesem Fall $f_1 > f_2$ gewählt werden sollte.

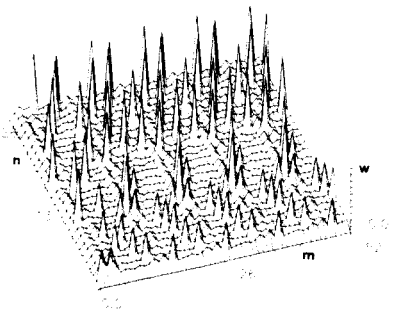
Die adaptive assoziative Matrix bietet weitere interessante Eigenschaften. Es können zum Beispiel Musterzuordnungen richtig erlernt werden, die während der Lernphase fehlerhaft eingegeben worden sind [92,93]. Sie ist ferner in der Lage, sich durch eine Fortführung der Lernphase an Änderungen der Musterzuordnungen anzupassen [93]. Die Verwendung von adaptiven Verbindungselementen führt aber im Vergleich zur assoziativen Matrix nicht zu einer höheren Anzahl von Musterpaaren, die sich mit einer geringen Bit-Fehlerwahrscheinlichkeit (3.1.10) zuordnen lassen.



a)



b)



c)

Abb. 4.3.14: Gewichtsmatrix einer adaptiven assoziativen Matrix (m=52, n=26) mit $w_{ij} \in [0,1]$:
a) Anfangszustand mit gleichverteilten Gewichten w_{ij} um 0.5;
b) Zustand nach 10 Lernschritten;
c) Endzustand nach der Lernphase.

Aufgrund der unterschiedlichen Gewichtswerte (Abb. 4.3.14c) muß der Schwellenwert Th beim Auslesen kleiner als die Anzahl der Einsen im Eingabemuster sein, damit alle geforderten Einsen des Ausgabemusters erzeugt werden können. Die Folge ist eine Erhöhung der Wahrscheinlichkeit für einen Bitfehler im Ausgabemuster und entsprechend verringert sich die Anzahl der maximal einzuspeichernden Muster (vgl. Abschitt 3.2.1 und 3.2.3).

Die adaptive assoziative Matrix ist ein einfaches Beispiel für ein lernfähiges Speicherkonzept auf der Grundlage eines Verbindungselementes mit Floating-Gate-Transistor. Für die Realisierung von adaptiven neuronalen ASICs sind adaptive Verbindungselemente in Form von einer Analogwertspeicherzelle ein Schlüsselelement. Die erste adaptive assoziative Speicherzelle mit nichtflüchtiger Informationsspeicherung ist von Goser/Fölster/Rückert vorgeschlagen worden und basiert auf der oben dargestellten Floating-Gate-Speicherzelle [92,94]. Andere Konzepte für Analogwertspeicher stammen zum Beispiel von Sage u.a. (MNOS/CCD [95]), Mackie u.a. (Kapazitäten bei niedrigen Temperaturen [76]), Spencer (programmierbare Widerstände aus Wismut-Sauerstoff-Verbindungen [96]) und Goser/Rückert (CCD-Ketten [97]). Allen gemeinsam ist der deutlich geringere Flächenaufwand gegenüber Realisierungen adaptiver Verbindungen in Digitaltechnik. Die Floating-Gate-Zelle zeichnet sich unter allen Vorschlägen durch ihren einfachen und kompakten Aufbau aus. Nur die von Spencer vorgeschlagenen programmierbaren Widerstände, deren Eigenschaften noch erforscht werden, könnten in Zukunft durch eine höhere Verbindungsdichte und kürzere Programmierzeiten eine interessante Alternative bieten.

Der Nachteil aller Konzepte ist die im Vergleich zu digitalen Realisierungen geringe Genauigkeit und Reproduzierbarkeit der gespeicherten Verbindungsgewichte. Eine zuverlässige Speicherung von analogen Werten mit einer geforderten Auflösegenauigkeit von mehr als 6 Bit ist mit den bekannten Konzepten nicht zu erreichen. Eine analoge Implementierung der Verbindungsgewichte ist somit nur dann anzustreben, wenn die geforderte Auflöse- und Rechengenauigkeit des assoziativen Netzwerkmodells gering ist.

Die theoretischen Forschungen auf dem Gebiet der Lernalgorithmen und Selbstorganisation orientieren sich weitgehend an mathematischen Strukturen und vernachlässigen das Problem der Berechnungsungenauigkeiten. Bei der Umsetzung der theoretischen Modelle in geeignete integrierte Schaltungsstrukturen muß man sich Klarheit darüber verschaffen, welche Auswirkungen die technologischen und schaltungstechnischen Randbedingungen auf die Eigenschaften des Modells haben. In diesem Zusammenhang ist es sicherlich sinnvoll, auch den entgegengesetzten Weg zu be-

schreiten, indem man vom physikalischen Bauelement ausgeht und die Adaptationsregel aus den Eigenschaften des Bauelementes ableitet. Das vorgestellte adaptive Verbindungselement mit Floating-Gate-Transistor ist der erste und bislang einzige Ansatz mit dieser Zielrichtung, der deutlich technologieorientiert ist und den genannten Randbedingungen für den Einsatz von Analogwertspeichern Rechnung trägt. Im Hinblick auf eine integrationsgerechte Umsetzung adaptiver neuronaler ASICs ist ein technologiegebundener Ansatz eine wesentliche Voraussetzung.

4.4 Integrierte Testschaltungen

Die in den vorangegangenen Abschnitten diskutierten Grundschaltungen für assoziative Netzwerke sind bis auf die in Abschnitt 4.2.2 erläuterten Konzepte mit geschalteten Kapazitäten in Form von integrierten Testschaltungen realisiert und ausgewertet worden [39,63,98-101]. Von den insgesamt 20 Testschaltungen ist die Hälfte zur Analyse der Eigenschaften einzelner Grundschaltungen, z.B. von verschiedenen Verbindungselementen (Abb. 4.3.2 und 4.3.5), entworfen worden. Die andere Hälfte umfaßt kleinere assoziative Netzwerke mit wenigen (5-10) einfachen Verarbeitungseinheiten. Zwei von diesen Testschaltungen werden in diesem Abschnitt beispielhaft erläutert.

Die Testschaltung in Abb. 4.4.1 enthält 8 einfache Verarbeitungseinheiten mit jeweils 32 Eingängen. Die Ansteuerung dieser Eingänge erfolgt über 8 externe Eingänge E_{1-8}, die den internen Eingängen entsprechend dem Verbindungsschema in Abb. 4.4.1 zugeordnet sind. Die Verbindungsmatrix besteht aus 8x32=256 festverdrahteten NMOS-Transistoren (W/L=1), von denen 39 durch fehlende Kontakte keinen Einfluß auf die Aktivierung einer Verarbeitungseinheit haben.

Mit dieser Testschaltung lassen sich in Abhängigkeit von der Wahl der Referenzspannungen zwei Aktivierungsfunktionen untersuchen. Für $U^{H}_{Ref} = U_B$ und $U^{A}_{Ref} = U^{W}_{Ref} > U_T$ erhält man die einfache Aktivierung mit NMOS-Transistoren (Abb. 4.2.2a). Mit $U^{A}_{Ref} = U^{W}_{Ref} = 0V$ und $U^{H}_{Ref} = 0V$ folgt eine zu der in Abb. 4.2.2b komplementäre Aktivierung mit einem NMOS-Transistor je Eingabezweig und einem PMOS-Transistor als Lastelement. Jeder hinzugeschaltete Eingabezweig bewirkt eine Reduzierung (*Inhibition*) des Aktivierungspotentials U_a. Die Transistoren sind für eine minimale Änderung des Aktivierungspotentials von $\Delta U_a \geq 40mV$ ($U_B = 7V$) bei einer maximalen Stufenanzahl von 30 dimensioniert worden (vgl. Abschnitt

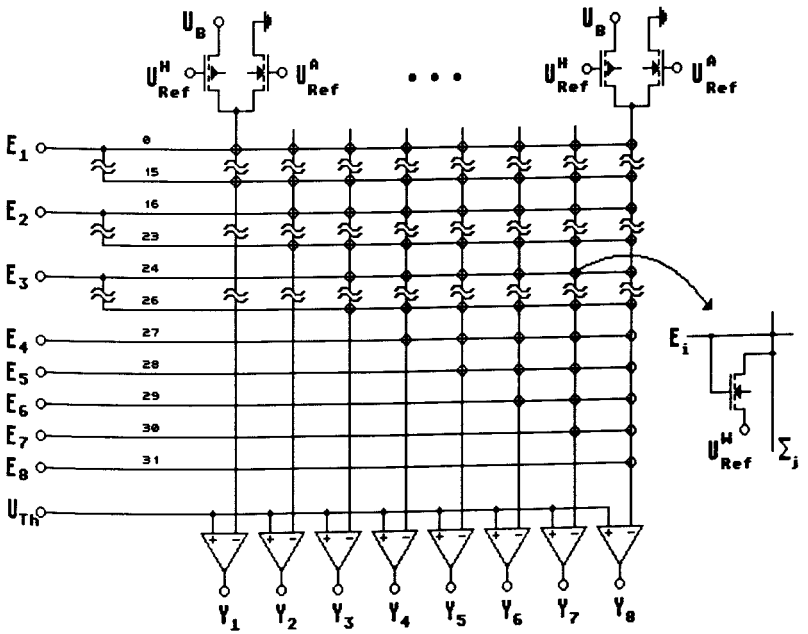
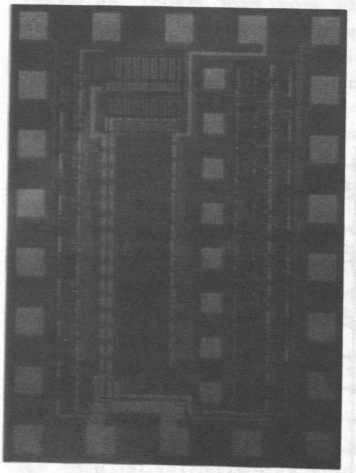
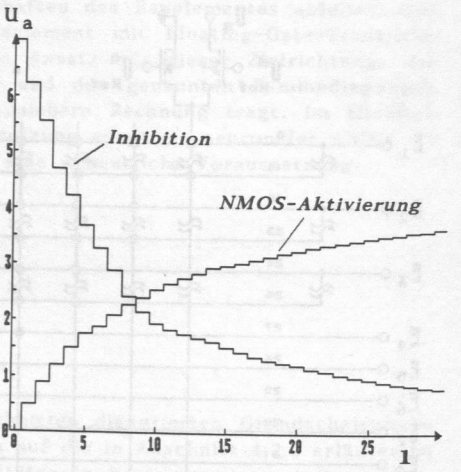


Abb. 4.4.1: Schematischer Aufbau einer Testschaltung für eine assoziative Matrix mit acht Verarbeitungseinheiten und festverdrahteten Verbindungselementen.

4.2.1). Bei der NMOS-Aktivierung beträgt das Verhältnis β_r=0.07 (4.2.10) und für die Inhibition ist β_r=6 (4.2.12). Die Ausgabe der Verarbeitungseinheit wird von einer einfachen Differenzstufe (125µmx125µm, 11 Transistoren [63]) und einem nachgeschalteten Ausgangstreiber erzeugt. Der Schwellenwert U_{Th} wird der Testschaltung über einen gesonderten Anschluß zugeführt.

Messungen an der integrierten Schaltung (Abb. 4.4.2) haben für die Aktivierung und die Inhibition eine maximale Stufenanzahl von 26 für $\Delta U_a \geq 40mV$ bzw. 30 Stufen für $\Delta U_a \geq 30mV$ ergeben (Abb. 4.4.2b). Mit den Differenzverstärkern lassen sich Aktionspotentiale mit $\Delta U_a \geq 40mV$ einwandfrei unterscheiden. Die Meßergebnisse stimmen somit mit den Entwurfsvorgaben zu dieser Testschaltung überein.





a) b)

Abb. 4.4.2: Photographie der integrierten Testschaltung (ca. 400 Transistoren, 1.5x1.0mm²) aus Abb. 4.4.1 (a) und gemessene Aktivierungswerte U_a in Abhängigkeit von der Anzahl aktivierter Eingänge 1 (b).

Mit der Testschaltung in Abb. 4.4.3 lassen sich Eigenschaften eines kleinen Hopfield-Netzes untersuchen. Die Schaltung besteht aus 5 Verarbeitungseinheiten, deren ternären festverdrahteten Verbindungslemente nach dem Vorschlag von Graf/deVegvar [84] entworfen worden sind (Abb. 4.3.7). In dem Netzwerk sind die Muster (++---) und (--+-+) gespeichert, die für das integrierte Netzwerk (Abb. 4.4.4) auch stabile Endzustände darstellen. Darüberhinaus ist das Muster (-----) ein stabiler Systemzustand, da die Verbindungselemente nach Graf/deVegvar keine Vorzeichenauswertung von Eingabe- und Gewichtswert ausführen (vgl. Abschnitt 4.3.2) und dadurch nur Gewichte w_{ij} mit y_j='+1' einen Beitrag zur Aktivierung leisten.

In Abb. 4.4.4b sind beispielhaft einige Musterzuordnungen des integrierten Hopfield-Netzes aufgeführt. Das Netzwerk kann unvollständige bzw. verfälschte Muster korrekt vervollständigen. Erfolgt eine Eingabe mit dem gleichen Hammingabstand zu den beiden gespeicherten Mustern,

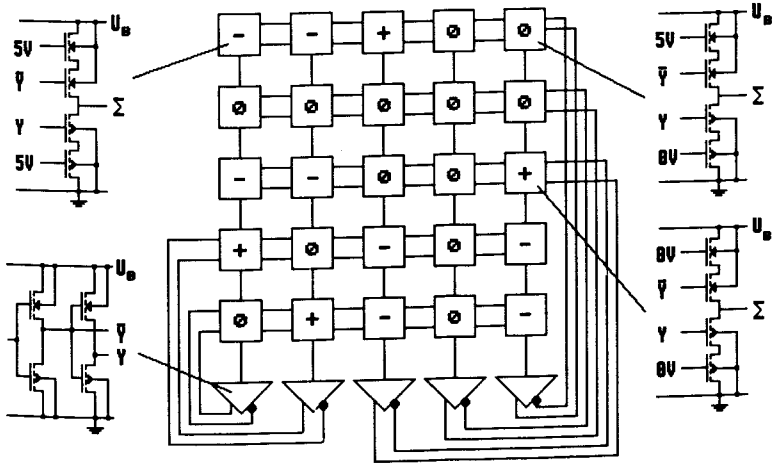
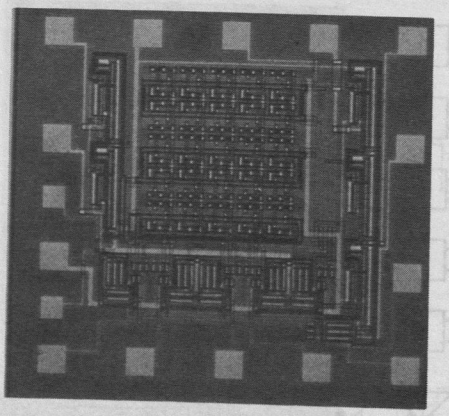


Abb. 4.4.3: Teststruktur eines Hopfield-Netzes mit 5 Verarbeitungseinheiten und festprogrammierten Verbindungselementen.

entscheidet sich das Netzwerk für das Muster (-----). Dieser Fall bedeutet für die einzelnen Verarbeitungseinheiten, daß die Anzahl der positiven und negativen Summanden gleich ist ($\pi=\nu$) und demzufolge das Aktivierungspotential $U_B/2$ betragen sollte. In der Testschaltung sind die Ausgänge y_j gleich Null ($\overline{y}_j=1$), d.h. es muß entweder das Aktivierungspotential U_a kleiner $U_B/2$ betragen oder die Schaltschwelle der Ausgabeschaltung größer als $U_B/2$ sein. Mit Hilfe von zusätzlichen Testanschlüssen (Abb. 4.4.4) konnte nachgewiesen werden, daß beide Effekte zur Geltung kommen.

Der Grund für die Asymmetrie der Aktivierung in diesem Fall ($\pi=\nu$) ist bereits in Abschnitt 4.2.1 (Abb. 4.2.4b) aufgezeigt worden. Durch die Verwendung von NMOS-Transistoren für die negativen und PMOS-Transistoren für die positiven Summanden ist das geforderte Verhältnis $\beta_r=1$ gemäß (4.2.14) aufgrund von herstellungsbedingten Parameterstreuungen bezüglich der Steilheitskonstanten k_n und k_p nicht exakt gleich Eins. In der Testschaltung ist für $\pi=\nu$ die Aktivierung der Verarbeitungseinheiten U_a kleiner als $U_B/2$, d.h. die PMOS-Transistoren haben einen größeren Widerstand und liefern demzufolge einen kleineren Beitrag zur gewichteten Summe als die NMOS-Verbindungen.



a)

Eingespeicherte Muster:

++−−−	−−+−+

Eingabe	Ausgabe
+−−−−	++−−−
+++−−	++−−−
++++−	++−−−
−−−−+	−−+−+
−−+++	−−+−+
+−+−+	−−+−+
+++−+	−−−−−
−++−−	−−−−−
−++++	−−−−−

b)

Abb. 4.4.4: (a) Integrierte CMOS-Schaltung der Teststruktur aus Abb. 4.4.3 (1x0.6mm², ca. 150 Transistoren);
(b) Beispiele für Mustervervollständigungen des integrierten Netzwerkes.

Der Fall $\pi=\nu$ tritt aber nicht nur bei Eingaben auf, die zu den gespeicherten Mustern den gleichen Hammingabstand haben, sondern z.B. auch für das Muster (-++++) (Abb. 4.4.4b). Für diese Eingabe wird nicht das Muster (--+-+) mit dem kleinsten Hamming-Abstand 2, sondern das Muster (-----) mit einem Hamming-Abstand 4 ausgegeben. Der auftretende Fehler ist somit gravierender als bei einer assoziativen Matrix, bei der sich Fehler in der Regel nur durch wenige zusätzliche Einsen auswirken.

Der Effekt weitet sich für größere Netzwerke auch auf die Fälle mit $|\pi-\nu|>0$ aus, da sich die Fehler in den einzelnen Summanden addieren. Eine Reduzierung dieses Effektes erreicht man mit dem Schaltungsvorschlag von Verleysen u.a. [77], in dem für die positiven und negativen Summanden Transistoren vom gleichen Typ verwendet werden und die Aktivierung über einen Differenzverstärker bewertet wird (vgl. Abb. 4.3.7). Nach Angaben der Autoren kann diese Schaltung den Fall $|\pi-\nu|=1$ bis zu einer Netzwerkgröße von n=512 noch sicher unterscheiden. Die Realisierung von größeren Hopfield-Netzen ist in analoger Schaltungstechnik unter Einhaltung der funktionalen Modelleigenschaften problematisch.

An ähnlichen integrierten Testschaltungen sind weitere Variationen kleinerer assoziativer Netzwerke in Analogtechnik analysiert worden. Die Messungen bestätigen ebenso wie die hier exemplarisch aufgeführten Ergebnisse die Überlegungen und Abschätzungen aus den vorangegangenen Abschnitten. Damit stehen getestete Grundschaltungen für den Entwurf großintegrierter assoziativer Netzwerke in analoger Schaltungstechnik zur Verfügung.

4.5 Diskussion

In diesem Kapitel sind Grundschaltungen für eine Verarbeitungseinheit einer assoziativen Matrix bzw. eines Hopfield-Netzes vorgestellt worden. Den Schwerpunkt bilden einfache Schaltungskonzepte aus der Analogtechnik, die sich gegenüber einer digitalen Schaltungsrealisierung durch eine flächengünstigere und schnellere Realisierung der Aktivierungs- und Ausgabefunktion einer Verarbeitungseinheit auszeichnen. Die wesentlichen Nachteile der analogen Schaltungskonzepte sind die begrenzte Berechnungs- bzw. Auflösegenauigkeit und die damit verbundene hohe Abhängigkeit von herstellungsbedingten Parameterstreuungen der Bauelemente. Diese Nachteile wirken sich erheblich auf assoziative Netzwerkmodelle aus, bei denen jeweils alle Eingänge einer Verarbeitungseinheit einen Beitrag zur Aktivierung leisten sowie bei Konzepten mit mehrwertigen Verbindungsgewichten.

Für die assoziative Matrix bedeuten diese Nachteile keine wesentliche Einschränkung, weil die Anzahl der Summanden bei der Aktivierung auch für große Matrizen sehr klein bleibt ($l \approx \log m$) und nur binäre Gewichte verwendet werden. Mit den in Abschnitt 4.2.1 diskutierten einfachen Schaltungskonzepten lassen sich bis zu 30 aktivierte Eingänge bei einer minimalen Differenz der Aktivierungspotentiale von 100mV unterscheiden. Eine Verarbeitungseinheit mit m=10000 ($l<14$) ist mit diesen Konzepten bereits möglich, ohne an die Grenzen der technischen Möglichkeiten zu gelangen.

Der Flächenaufwand einer einzelnen Verarbeitungseinheit wird von der Anzahl und Größe der Verbindungselemente bestimmt. Für binäre Verbindungselemente lassen sich derzeit Verbindungsdichten von ca. $10^8/cm^2$ für festprogrammierte und ca. $10^6/cm^2$ für programmierbare und adaptive Verbindungselemente erreichen. Die Weiterentwicklung der Sub-µ-Technologien wird zu einer Erhöhung der Verbindungsdichte um mindestens eine Größenordnung führen, so daß eine assoziative Matrix mit 10^6-10^7 programmierbaren Verbindungselementen auf einer Fläche von einem Quadratzentimeter integriert werden kann.

Die Verbindungsdichte für mehrwertige Gewichte reduziert sich entsprechend dem größeren Flächenaufwand der einzelnen Verbindungselemente. Für ein Hopfield-Netz mit ternären Gewichten resultiert beispielsweise eine Verringerung der Verbindungsdichte gegenüber der assoziativen Matrix um mindestens den Faktor 1/2. Über die Mehrwertigkeit hinaus bewirkt die Funktionalität der Verbindungsgewichte einen höheren Flächenaufwand. Wird, wie bei dem Verbindungselement von Verleysen u.a. (Abb. 4.3.7), die Vorzeichenauswertung des Eingabe- und Gewichtswertes in das Verbindungselement einbezogen, so erhöht sich der Flächenaufwand um ein XOR-Gatter und einen Inverter.

Eine beträchtliche Erhöhung des Flächenaufwandes ergibt sich für Verbindungselemente in digitaler Schaltungstechnik, die die Programmier- bzw. Lernregel enthalten. In Anbetracht des hohen Flächenbedarfs ist es oft sinnvoller, zumindest die Lernphase des assoziativen Netzwerkes auf einem Neurocomputer zu emulieren. Es bleibt kritisch abzuwägen, ob der Gewinn an Parallelität durch die spezielle schaltungstechnische Realisierung den Mehraufwand an Fläche und Entwurfskomplexität ausgleichen kann.

Eine interessante Perspektive bietet sich in diesem Zusammenhang für die Analogtechnik durch die funktionale Integration der Lernregel. Von den wenigen derzeit bekannten Lösungsvorschlägen zeichnet sich das in Abschnitt 4.3.3 vorgestellte adaptive Verbindungselement mit Floating-Gate-Transistor durch eine nichtflüchtige Speicherung und einen kompakten Aufbau aus. Die Anwendung von derartigen adaptiven Verbindungen beschränkt sich auf assoziative Netzwerkmodelle, die nur eine geringe Anforderung an die Rechengenauigkeit der Verarbeitungseinheit und die Reproduzierbarkeit der Gewichtswerte stellen.

Zusammengefaßt ergeben sich für eine assoziative Matrix die größte Verbindungsdichte und die einfachsten Schaltungsrealisierungen. Eine integrationsgerechte Umsetzung ist sowohl in digitaler als auch in analoger Schaltungstechnik möglich. Der Bedarf an eine spezielle Hardwarerealisierung ist in Kapitel 3 aufgezeigt worden. Aus diesen Gründen ist für eine assoziative Matrix die Realisierung eines neuronalen ASICs anzustreben.

Mit dem heutigen Stand der Großintegrationstechnik lassen sich assoziative Netzwerke mit tausend Verarbeitungseinheiten auf einem Chip integrieren. Für die Realisierung größerer Netzwerke , z.B. einer $10^4 \times 10^4$ Matrix mit programmierbaren nichtflüchtigen Gewichten, ist es notwendig, das Netzwerk in kleinere Module aufzuteilen. Aufgrund der ausgesprochen regulären und modularen Architektur assoziativer Netzwerke sind die Entwurfsaufgaben hinsichtlich der Partitionierung, Plazierung und Verdrahtung durch die Systemstruktur weitgehend vorgegeben. Für die Partitionierung der Verbindungsmatrix bieten sich eine Rasteraufteilung oder eine vertikale Aufteilung in "Scheiben" (Slices) an (Abb. 5.1). Eine Rasteraufteilung hat den Vorteil, daß sowohl die Anzahl der Zeilen als auch der Spalten der Verbindungsmatrix erweitert werden können. Bei einer Aufteilung in Scheiben kann nur die Anzahl der Spalten, also die der Verarbeitungseinheiten erhöht werden. Die Aufteilung einer Verarbeitungseinheit auf mehrere Chips hat den Nachteil, daß zur Bestimmung der Aktivierungsfunktion Teilsummen über mehrere Chips übertragen werden müssen. Das erhöht den Kommunikationsaufwand für digitale Implementierungen. In Analogtechnik bedarf es zusätzlicher schaltungstechnischer Maßnahmen.

Das Hauptproblem assoziativer Netzwerke hinsichtlich einer VLSI-Implementierung ist die sehr hohe Anzahl von Verbindungen. Die Verbindungstechnik beschränkt die Anzahl der Anschlüsse (Pads) einer integrierten Schaltung auf einige hundert, eine Erweiterung auf 1000 wird in

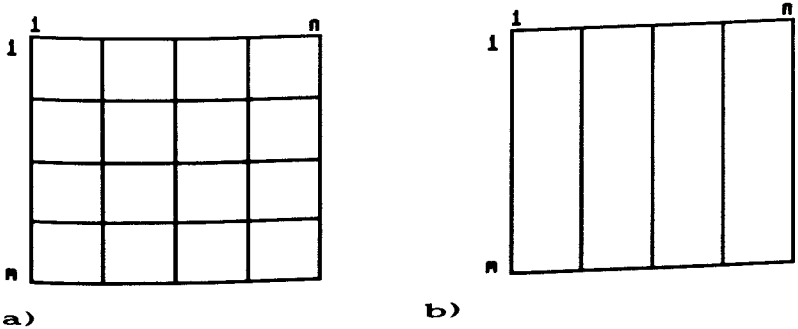


Abb. 5.1: Partitionierungsalternativen der Verbindungsmatrix: a) Rasteraufteilung; b) Aufteilung in Scheiben (Slices).

der Zukunft möglich sein [102]. Die parallele Übertragung aller Ein- und Ausgabesignale einer 1000x1000 Matrix ist damit nicht ohne weiteres möglich. Für synchrone Netzwerke ist es möglich, durch eine serielle Übertragung der Ein- und Ausgaben das Problem auf Kosten der Zugriffszeit zu umgehen. Die einfachste, aber langsamste Lösung ist die Verwendung von Schieberegistern (Abb. 5.2), so daß für die Ein- und Ausgabe jeweils nur noch ein Anschluß benötigt wird. In der Regel werden soviele Eingaben parallel übertragen, wie Anschlüsse für die Eingabe zur Verfügung stehen. Die Übertragung von m Eingaben benötigt dann bei p Anschlüssen m/p Taktzyklen. Nicht zu vernachlässigen sind der zusätzliche Schaltungs- und Verwaltungsaufwand zur Steuerung der Ein- und Ausgabe.

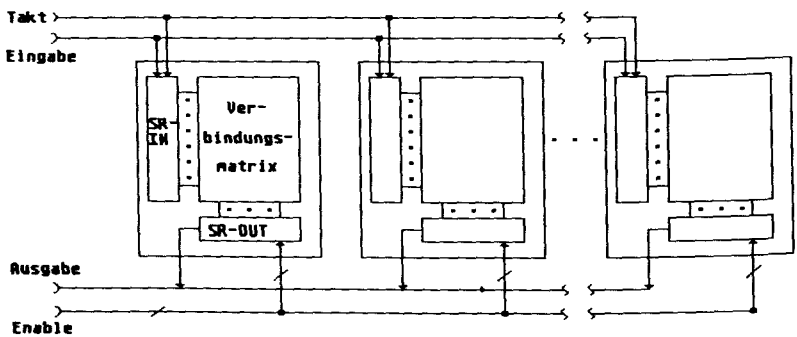


Abb. 5.2: Einsatz von Schieberegistern zur Einsparung von Anschlüssen.

Asynchrone Netzwerke benötigen für jeden Ein- und Ausgang des Netzwerkes eine separate Leitung. Stehen insgesamt p Anschlüsse für die Ein- und Ausgabe zur Verfügung, so können bei einer Rasteraufteilung der Verbindungsmatrix nur $(p/2)^2$ Verbindungselemente angesteuert werden. Für p=500 (1000) folgt eine maximale Anzahl von 62500 (250000) Verbindungen. Für asynchrone und vollparallele assoziative Netzwerke ist nicht die Integrationsdichte, sondern die Verbindungstechnik der einschränkende Faktor.

Die Verwendung von spärlichen Kodierungen schafft hier einen beachtlichen Vorteil. Da nur wenige Einsen (l=O(log m)) pro Muster vorhanden sind, ist es nicht sinnvoll das vollständige m-Bit Muster, sondern nur die l Adressen der Einsen innerhalb des Musters zu übertragen. Die

Adressen (ld m-Bit) der l Einsen im Eingabemuster können entweder einzeln oder auch parallel übertragen werden. Im ersten Fall benötigt man l-Taktzyklen und ld m Eingabeanschlüsse, im zweiten Fall einen Taktzyklus und l·ld m≈(ld m)^2 Eingabeanschlüsse. Selbst für große assoziative Matrizen und bei einer parallelen Übertragung der Adressen liegt die Anzahl p der notwendigen Anschlüsse im Rahmen der technologischen Möglichkeiten:

$$p \approx (\text{ld } m)^2 = 169 \quad \text{für } m=8192 \tag{5.1}$$

Für die Übertragung des Ausgabemusters gilt Entsprechendes. Nach diesen allgemeinen Vorbemerkungen zum Systementwurf wird in den folgenden Abschnitten der Entwurf einer assoziativen Matrix in Digitaltechnik und in einer digital-analogen Mischtechnik vorgestellt. Die Entwürfe werden abschließend diskutiert und mit anderen in der Literatur bekannten Vorschlägen verglichen.

5.1 Digitale Implementierung einer assoziativen Matrix

Eine Verarbeitungseinheit einer assoziativen Matrix in Digitaltechnik umfaßt m binäre Speicherzellen und eine Verarbeitungskomponente, die im wesentlichen aus einem Zähler und einem Komparator besteht. Für eine assoziative Matrix ergibt sich eine Unterteilung in einen n×m-Speicherblock und einen digitalen Verarbeitungsblock mit n Verarbeitungskomponenten. Es bietet sich an, den Speicherblock, für den man herkömmliche Speicherbausteine mit wahlfreiem Zugriff verwenden kann, und den Verarbeitungsblock getrennt auszuführen (Abb. 5.1.1). Diese Aufteilung hat den Vorteil, daß die Größe der Verbindungsmatrix variabel bleibt. Die Verwaltung der assoziativen Matrix wird von einem Steuerprozessor (Kontroller) übernommen, der die entsprechenden Adressen der Komponenten im Eingabemuster mit x_i=1 an die Speichermatrix anlegt und das Assoziationsergebnis abruft.

Selbst mit der Trennung von Speicher- und Verarbeitungsbereich wird es nicht möglich sein, den Verarbeitungsblock einer assoziativen Matrix auf einer integrierten Schaltung (Chip) unterzubringen. Die günstigste Partitionierung der Verbindungsmatrix ist eine vertikale Aufteilung in einzelne "Scheiben" gleicher Größe (Abb. 5.1.2). Jede dieser Scheiben wird von einem einzelnen VLSI-Baustein (Slice-Prozessor) verwaltet, der die notwendigen Rechenoperationen (Aufsummieren, Vergleichen) parallel für alle Spalten dieser Scheibe ausführt. Die Anzahl der Speicherspalten pro Scheibe richtet sich nach der maximalen Anzahl von Verarbeitungskompo-

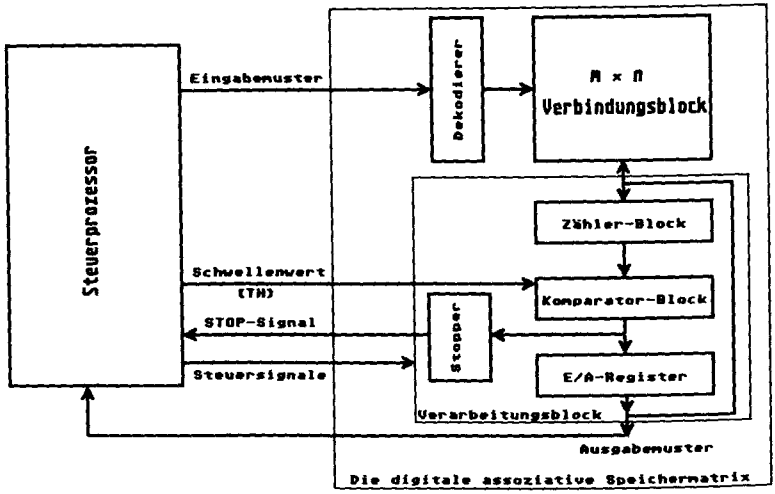


Abb. 5.1.1: Systemüberblick einer assoziativen Matrix in Digitaltechnik.

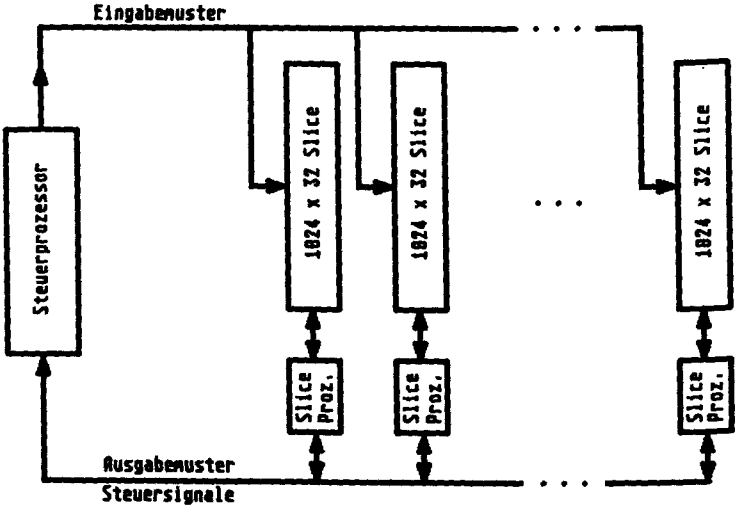
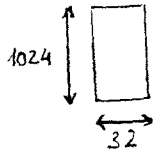


Abb. 5.1.2: Partitionierung der Systemarchitektur einer assoziativen Matrix in Digitaltechnik (m=1024).



nenten, die auf einem Chip integriert werden können. Das Beispiel in Abb. 5.1.2 geht von 32 Verarbeitungskomponenten pro Chip aus, so daß eine 1024x1024 Matrix aus 32 derartigen Chips besteht.

Der Entwurf des Slice-Prozessor-Bausteins wird durch die Verfügbarkeit einer rechnerunterstützten Entwurfsumgebung wesentlich erleichtert. Am Lehrstuhl Bauelemente der Elektrotechnik der Universität Dortmund steht das VENUS-Standardzellen-Entwurfssystem [57] zur Verfügung. Mit diesem Entwurfssystem ist ein Slice-Prozessor mit 32 Verarbeitungskomponenten entworfen worden, dessen Blockschaltbild in Abb. 5.1.3 gezeigt ist. Die 32 Zähler und Komparatoren werden jeweils zu einem Block zusammengefaßt. Der Baustein enthält ferner einen 32-Bit-Zwischenspeicher und einen 32-Bit-Oder-Block zur Ausführung der Programmierregel einer assoziativen Matrix. Eingaben des Bausteins sind die Daten aus der Speichermatrix (32 Bit), der digitale Schwellenwert (6-10 Bit) und verschiedene Steuersignale (Beschreibung siehe [103]). Ausgegeben wird von dem Baustein das assoziierte Teilmuster (32 Bit) und ein zusätzliches STOP-Signal, das anzeigt, ob das erzeugte Teilmuster eine Eins enthält. Die 32 Ein- bzw. Ausgabeleitungen für die Teilmuster können zu bidirektionalen Ein- bzw. Ausgabeleitungen zusammengefaßt werden.

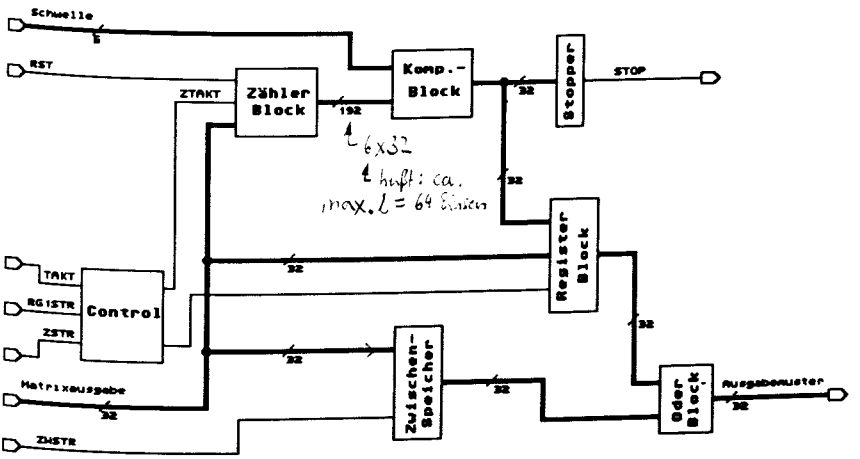


Abb. 5.1.3: Vereinfachtes Blockschaltbild eines Slice-Prozessors.

In der ersten Version ist mit dem VENUS1.4 Entwurfssystem ein Slice-Prozessor mit 6-Bit-Zählern und Komparatoren entworfen worden (Abb. 5.1.4). Die Standardzellen-Bibliothek von VENUS1.4 basierte auf einem 3µm CMOS–Prozeß mit einer Aluminium- und einer Polysilizium-Verdrahtungsebene. Die Anzahl der Transistoren des Chips beträgt ca. 22500. Mit der Verfügbarkeit der Standardzellen-Bibliothek von VENUS3.3, die auf einem 2µm-CMOS-Prozeß mit zwei Aluminium- und einer Polysilizium-Verdrahtungsebene basiert, ist dieser Baustein auf 8- und 10-Bit-Zähler und Komparatoren erweitert worden [103]. In Tabelle 5.1.1 sind die charakteristischen Daten der Entwürfe zusammengefaßt.

	Zellen	Zeilen	Abmessungen [mm]	Fläche [mm^2]	Pads	Taktrate MHz
VENUS1.4 6 Bit Zähler	696	16	7.47x8.2	61.254	53	10
VENUS3.3 6 Bit Zähler	861	13	5.59x5.52	30.857	53	10
VENUS3.3 8 Bit Zähler	871	14	5.71x5.48	31.29	55	10
VENUS3.3 10 Bit Zähler	1141	16	6.45x6.04	38.96	57	10

Tabelle 5.1.1: Kenngrößen der mit VENUS entworfenen Slice-Prozessor-Bausteine [103].

Der Assoziationsvorgang wird für alle Verarbeitungseinheiten parallel ausgeführt, wobei die Eingabemuster aber nur seriell abgearbeitet werden. Für jede Eins im Eingabemuster wird von dem Kontroller die entsprechende Adresse über einen gemeinsamen Bus an alle Speicherblöcke übergeben, die dann innerhalb einer Speicherzugriffszeit (t_{spzg}) die Daten den Slice-Prozessoren zur Verfügung stellen. Der Slice-Prozessor benötigt einen Taktzyklus (t_{ass}) für das Aktualisieren der Zähler.

Nach dem Abarbeiten der 1 Einsen im Eingabemuster wird der Schwellenwert eingelesen und der Vergleich mit dem Komparator ausgeführt (2 Taktzyklen t_{ass}).

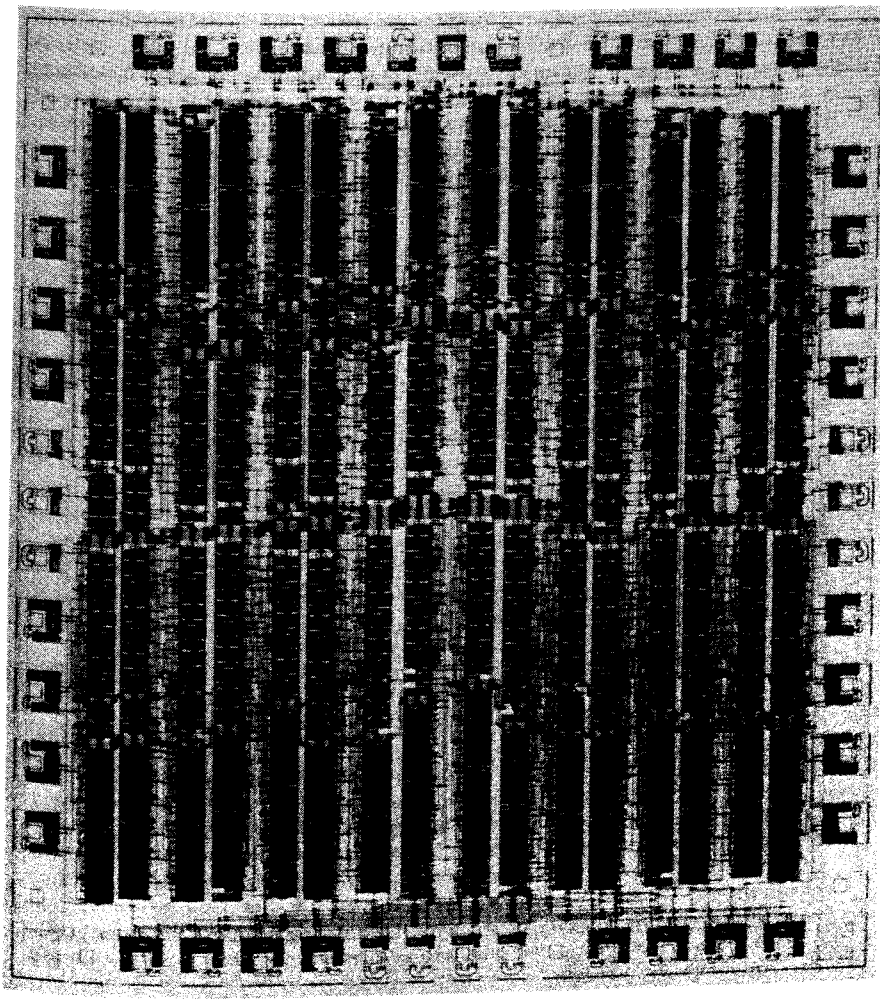


Abb. 5.1.4: Layout eines VENUS1.4-Standardzellen-Entwurfs von einem Slice-Prozessor-Baustein mit 32 Verarbeitungskomponenten (ca. 22500 Transitoren, 7.47x8.2mm^2, [104]).

Abschließend werden nur die Teilmuster der Slice-Prozessoren an den Kontroller übertragen, die mindestens eine Eins enthalten. Die Auslesereihenfolge wird mit Hilfe des Stop-Signals von einer Prioritätsschaltung festgelegt, wie man sie auch bei inhaltsadressierten Speichern verwendet [103]. Die Übertragung eines Teilmusters vollzieht sich in zwei Taktzyklen, indem zuerst die Kennung des Bausteins übertragen wird und dann das Teilmuster selbst. Das ergibt im Mittel 2·k Taktzyklen (t_{ass}). Zusammengefaßt folgt für die Assoziationszeit:

$$T_A = 1\cdot(t_{spgz}+t_{ass}) + 2\cdot t_{ass} + 2\cdot k\cdot t_{ass} \qquad (5.1.1)$$

Für den Lernvorgang werden erst die Zwischenspeicher aller Slice-Prozessoren mit dem entsprechenden Teil des Ausgabemusters belegt. Dazu sind 2·k Taktzyklen notwendig. Anschließend wird für jede der l Einsen im Eingabemuster die entsprechende Zeile aus der Speichermatrix in den Registerblock des Slice-Prozessors gelesen und der Inhalt des Register- und Zwischenspeicherblocks über den ODER-Block miteinander verknüpft. Das Ergebnis wird in die Speichermatrix zurückgeschrieben. Insgesamt benötigt man für jede Eins im Eingabemuster einen Lesezugriff auf die Speichermatrix, drei Slice-Prozessor-Taktzyklen und einen Schreibzyklus für die Speichermatrix. Zusammenfassend ergibt sich für z Musterpaare die folgende Lernzeit:

$$T_L = z\cdot(2\cdot t_{ass} + 1\cdot(3\cdot t_{ass}+ 2\cdot t_{spzg})) \qquad (5.1.2)$$

Ein weiterer Schwellenwertvergleich mit geänderter Schwelle vollzieht sich in drei Schritten. Die neue Schwelle wird eingelesen, der Vergleich durchgeführt und die k Teilmuster werden wieder ausgelesen:

$$T_{Th}= 2\cdot t_{ass} + 2\cdot k\cdot t_{ass} \qquad (5.1.3)$$

Ein mit VENUS3.3 entwickelter Slice-Prozessor kann auf einer Chipfläche von einem Quadratzentimeter 64 Verarbeitungskomponenten enthalten. Die Anzahl der Verarbeitungskomponenten $\varkappa$ pro Baustein erhöht sich mit zunehmender Integrationsdichte, so daß man für eine Sub-μ-Technologie mit 256 Verarbeitungsblöcken pro Baustein rechnen kann. Die Anzahl der Slice-Prozessoren CH_{ass} für eine assoziative m×n-Matrix ergibt sich aus der einfachen Beziehung:

$$CH_{ass}= n/\varkappa \qquad (5.1.4)$$

Eine assoziative Matrix mit m=n=8192 besteht demnach für $\varkappa$=64 aus 128 Slice-Prozessor-Bausteinen und einer Verbindungsmatrix mit 64MBit. In dieser Matrix können mehr als eine Million spärlich kodierte Musterpaare (k=3,l=10) mit einer Bit-Fehlerwahrscheinlichkeit von kleiner als Eins assoziativ gespeichert werden. Das entspricht einer Speichereffek-

tivität von ca. 0.61. Die Assoziationszeit beträgt bei einer Taktrate von 10MHz ca. 4.7µs und das Programmieren aller Musterpaare ca. 11s. In Tabelle 5.1.2 sind weitere Daten für verschiedene assoziative Matrizen beispielhaft aufgeführt.

n = m	Z (3.1.10)	S_{eff} (3.2.1)	T_A (5.1.1)	T_L (5.1.2)	RAM (MBit)	Ch_{ass} x=64	256
4096					16	64	16
l=12 k=3	318353	0.604	4.4µs	2.74s			
l=11 k=10	95395	0.540	5.5µs	0.75s			
8192					64	128	32
l=13 k=3	1182378	0.613	4.7µs	11s			
k=10	354750	0.554	6.1µs	3.3s			
16384					256	256	64
l=14 k=3	4407707	0.621	5.0µs	44.1s			
k=10	1322288	0.566	6.4µs	13.2s			

Tabelle 5.1.2: Leistungsdaten assoziativer Netzwerke in Digitaltechnik (Taktrate 10 MHz).

Für eine vollparallele Arbeitsweise der Slice-Architektur werden Speicherbausteine mit einer mxb-Speicherorganisation benötigt, d.h. im obigen Beispiel sind 512K-RAMs mit einer 8Kx64- oder 256K-RAMs mit einer 8Kx32- Speicherorganisation erforderlich. Die geforderte Wortbreite b ist größer als die von handelsüblichen Speicherbausteinen mit b=4Bit bzw. b=8Bit. Solange keine Speicherbausteine mit einer geeigneten Speicherorganisation zur Verfügung stehen, wird eine vollparallele Arbeitsweise mit einem relativ hohen Hardwareaufwand verbunden sein.

Soll unter Verwendung von handelsüblichen Speicherbausteinen kein Speicherplatz verschwendet werden, dann ist eine vollparallele Arbeitsweise im allgemeinen nicht zu erreichen. Eine Auslastung der Speichergröße S eines vorgegebenen Speicherbausteins mit der Wortbreite b ist nur möglich, wenn mehr Spalten der Verbindungmatrix gespeichert werden als parallel zugreifbar sind. Bei der Verwendung von 256K-RAM-Bausteinen mit einer 64Kx4-Speicherorganisation werden z.B. zur Speicherung einer 8192x8192-Verbindungsmatrix 256 Speicherbausteine benötigt. Jeder Bau-

stein speichert 32 Spalten der Verbindungsmatrix, von denen aber nur 4 parallel zugreifbar sind. Für den Zugriff auf eine vollständige Zeile der Verbindungsmatrix sind 8 Lesezyklen erforderlich. Allgemein erhöht sich die Anzahl der Lesezugriffe um den Faktor f_A:

$$f_A = \frac{S}{m \cdot b} \qquad (5.1.5)$$

Der Faktor f_A sollte für einen schnellen Zugriff klein gehalten werden. Bei den heutigen Speicherentwicklungen vergrößert sich aber zunehmend das Verhältnis S/b. Mit jeder neuen Speichergeneration vervierfacht sich die Speichergröße, während die Wortbreite im allgemeinen nicht größer als 8Bit ist. Um den Faktor f_A konstant zu halten, könnte die Matrixgröße m entsprechend erhöht werden. Das führt aber zu einer sehr hohen Anzahl von Speicherbausteinen.

Für größere Faktoren f_A wird es hinsichtlich des Hardwareaufwandes zunehmend sinnvoller, anstatt der Slice-Prozessoren herkömmliche Mikroprozessoren für die Verwaltung einer Matrixscheibe zu verwenden. Es ergibt sich eine einfache Mehrprozessor-Architektur zur Emulation assoziativer Matrizen. Für eine derartige Architektur, die gemäß Kapitel 1 einen einfachen Neurocomputer darstellt, sind bereits zwei Prototypen am Max-Planck-Institut für biologische Kybernetik in Tübingen entwickelt worden [105,106]. Mit der Programmierbarkeit der Prozessoren erhält die Mehrprozessor-Architektur den Vorteil der Flexibilität. Nachteilig erweist sich die größere Assoziationszeit und der zusätzliche Programmieraufwand.

Zusammenfassend zeichnet sich das Slice-Prozessor-Konzept für kleine Faktoren f_A ($\lesssim 10$) durch einen sehr schnellen assoziativen Zugriff sowie einen einfachen und modularen Systemaufbau aus. Das Systemkonzept nutzt die Vorteile einer spärlichen Kodierung und die systeminhärente Parallelität assoziativer Netzwerke weitgehend aus. Das Slice-Prozessor-Konzept kann dadurch für assoziative Matrizen mit mehr als 10^4 Verarbeitungseinheiten eine Musterzuordnung in der Größenordnung von 10µs bestimmen.

5.2 Digital-analoge Implementierung einer assoziativen Matrix

Der Entwurf einer Verarbeitungseinheit in Analogtechnik wird, wie in Abschnitt 4 erläutert, im wesentlichen von der zur Verfügung stehenden Technologie bestimmt. Der Lehrstuhl Bauelemente der Elektrotechnik an der Universität Dortmund verfügt über einen 3µm CMOS-Prozeß mit einer Aluminium- und einer Polysilizium-Verdrahtungsebene [6]. Hochohmige Widerstände und Floating-Gate-Transistoren werden entwickelt, stehen aber derzeit noch nicht zur Verfügung [85]. Für die Realisierung der Verbindungselemente verbleiben Kapazitäten und Transistoren. Die Nachteile von Kapazitäten sind der höhere Flächenaufwand und die größere herstellungsbedingte Streuung der absoluten Kapazitätswerte. Aus diesen Überlegungen heraus ist für die vorgegebene Technologie die Verwendung der Stromsummation als Aktivierungsfunktion naheliegend. Um eine im Rahmen der Möglichkeiten flexible, aber auch einfache Architektur zu erhalten, wird die assoziative Matrix mit programmierbaren statischen Verbindungselementen realisiert.

Ein wesentlicher Vorteil der digitalen Schaltungstechnik gegenüber der Analogtechnik ist die Verfügbarkeit von ausgereiften und automatisierten Entwurfstechniken. Für den Entwurf von assoziativen Netzwerken mit analogen und digitalen Schaltungskomponenten kann auf eine automatisierte Entwurfsumgebung derzeit noch nicht zurückgegriffen werden. Aus diesem Grunde waren alle Schaltungskomponenten der assoziativen Matrix in Analogtechnik einzeln zu entwerfen. Zur Entwurfsunterstützung der Schaltungskomponenten stehen am Lehrstuhl Bauelemente der Elektrotechnik der Universität Dortmund der Schaltkreissimulator BONSAI [59] sowie Zeichnungseditoren und Verifikationsprogramme der Firma Silvar-Lisco [107-109] zur Verfügung.

Zur Vereinfachung des Entwurfs von verschiedenen Systemkonzepten ist im Rahmen dieser Arbeit eine Zellenbibliothek für assoziative Netzwerke entwickelt worden. Die Zellenbibliothek (Abb. 5.2.1) enthält Realisierungsalternativen für die einzelnen Funktionsblöcke einer Verarbeitungseinheit eines assoziativen Netzwerkes (vgl. Definition 1) und verschiedene Peripherieschaltungen. Die einzelnen Zellen sind gemäß den Überlegungen aus Kapitel 4 entworfen und das elektrische Verhalten ist anhand von Testschaltungen (vgl. 4.4) ermittelt worden.

Der Entwurf einer assoziativen Matrix mit analogen Schaltungskomponenten erfordert nun eine Auswahl der geeigneten Zellen und deren Zusammenfügen mit Hilfe eines Layout-Editors. Die Plazierung und Verdrahtung der Zellen ist durch den regulären Aufbau der assoziativen Matrix weitgehend vorgegeben. Im folgenden werden zwei Bausteinentwürfe einer assoziativen Matrix vorgestellt, die mit Zellen der Bibliothek aufgebaut und an der Universität Dortmund gefertigt worden sind.

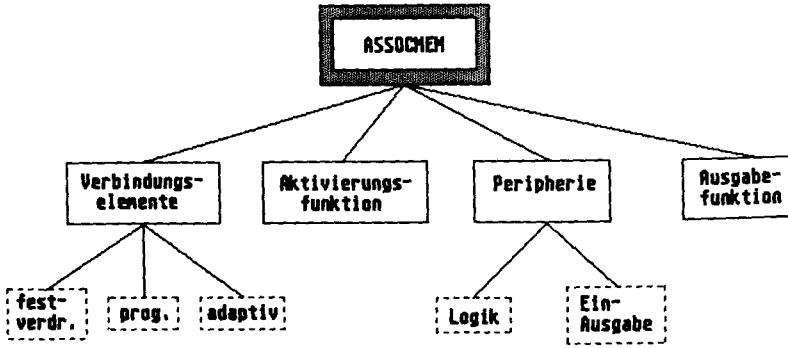
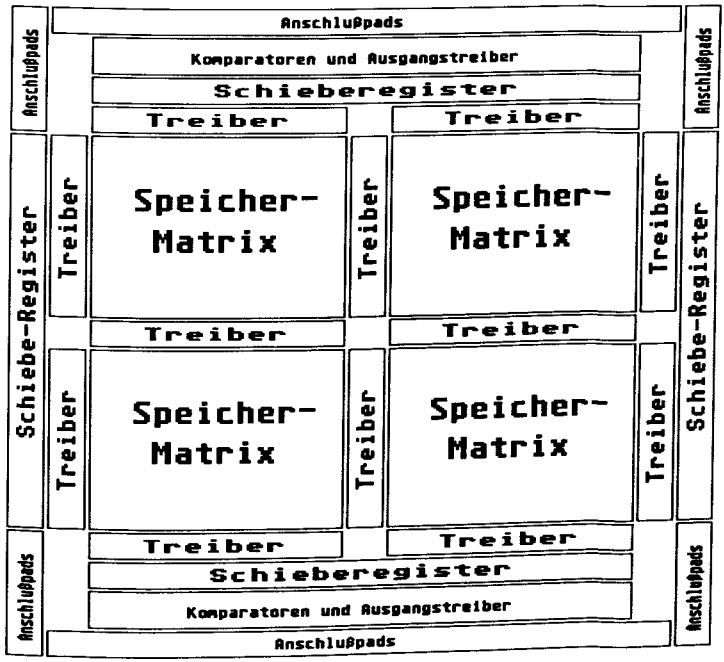


Abb. 5.2.1: Schematischer Aufbau der Zellenbibliothek ASSOCMEM für assoziative Netzwerke.

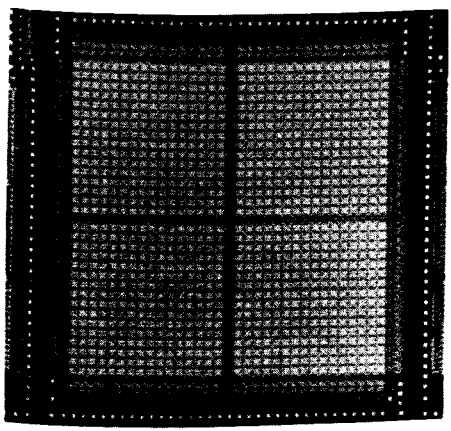
5.2.1 Ein Testbaustein für die assoziative Matrix

Die in Abschnitt 4.4 erläuterten Testschaltungen konnten die grundsätzliche Funktionsweise von einzelnen Schaltungskomponenten und kleinen assoziativen Netzwerken nachweisen. Der nächste Schritt ist die Untersuchung einer größeren integrierten assoziativen Matrix. Zu diesem Zweck ist der Testbaustein ARAM1 (associative random access memory) entworfen und realisiert worden.

Der Baustein ist schaltungstechnisch sehr einfach aufgebaut (Abb. 5.2.2). Er umfaßt 64 Verarbeitungseinheiten mit jeweils 64 programmierbaren statischen Verbindungen, d.h. insgesamt enthält der Baustein 4096 Verbindungselemente. Die Daten zur Programmierung sowie die Eingabemuster für eine Musterzuordnung werden über Schieberegister eingegeben. Für eine Assoziation wird das rechte Schieberegister im Blockschaltbild (Abb. 5.2.2a) genutzt, während die verbleibenden Schieberegister zur Programmierung (Wortleitung, Datenleitungen D,$\overline{D}$) benötigt werden. Die Ausgaben der 64 Verarbeitungseinheiten werden einzeln herausgeführt, die Ausgänge y_1-y_{63} nach unten und die Ausgänge y_2-y_{64} nach oben. Insgesamt hat der Baustein ARAM1 84 Anschlußpads (Beschreibung siehe [63]). Die Kenndaten von ARAM1 sind in Abb. 5.2.2c zusammengefaßt.



a)



b)

Verbindungs-elemente:	4096 prog.
Matrixgröße:	64x64
Chipgröße:	7.05x7.95 mm
Fläche:	ca. 56 mm^2
Transistoren:	ca. 40 000
Anschlüsse:	84

c)

Abb. 5.2.2: Die Testschaltung ARAM1: Blockschaltbild (a), Schaltungsfoto (b), Kenndaten (c) und Layout (d) (siehe folgende Seite).

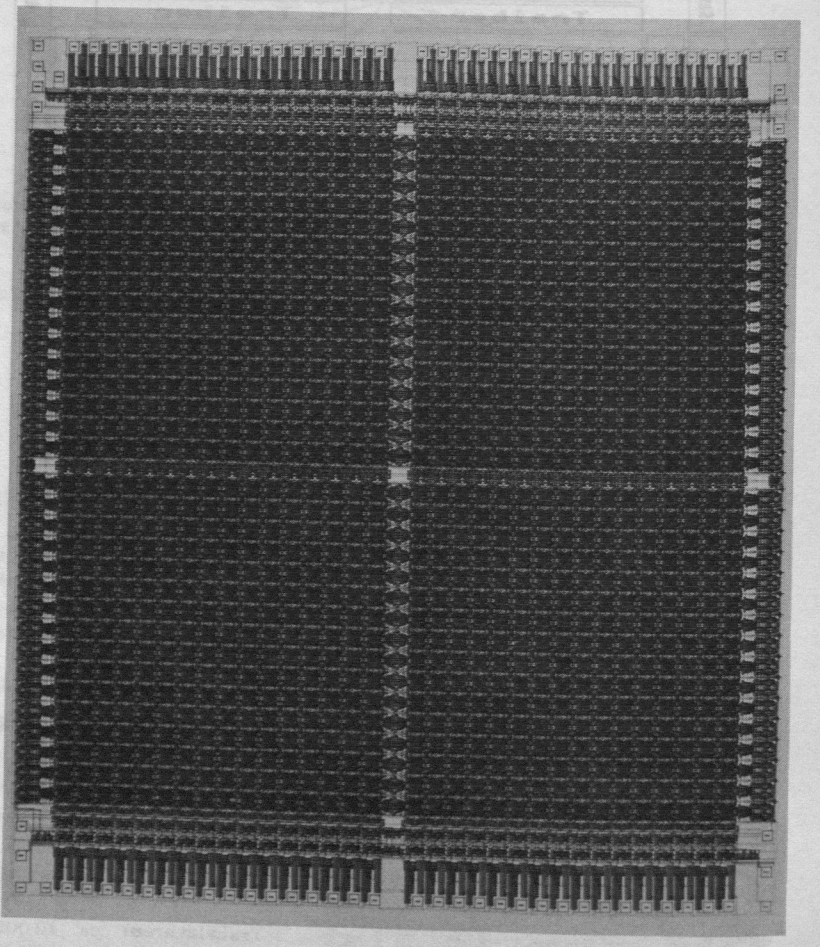


Abb. 5.2.2d: Layout der Testschaltung ARAM1.

Der Aufbau der einzelnen Verarbeitungseinheiten ist in Abb. 5.2.3 zusammengefaßt. Die berechnete Aktivierungsfunktion entspricht der in Abschnitt 4.4 skizzierten "*Inhibition*" mit einem PMOS-Transistor als Lastelement (W/L=90) und zwei hintereinandergeschalteten NMOS-Transistoren (W/L=2) in den Eingabezweigen, die mit Masse verbunden sind. Die Transistoren sind für eine minimale Änderung des Aktivierungspotentials von $\Delta U_a \geq 100mV$ (U_B=5V) bei einer Stufenanzahl von 20 dimensioniert worden. Der Schwellenwertvergleich wird von einer einfachen Differenzstufe durchgeführt und das Ergebnis über eine Treiberstufe auf das Ausgabepad gegeben. In Abb. 5.2.4 wird ein Ausschnitt der integrierten Verbindungsmatrix und der Ausgabeschaltung gezeigt.

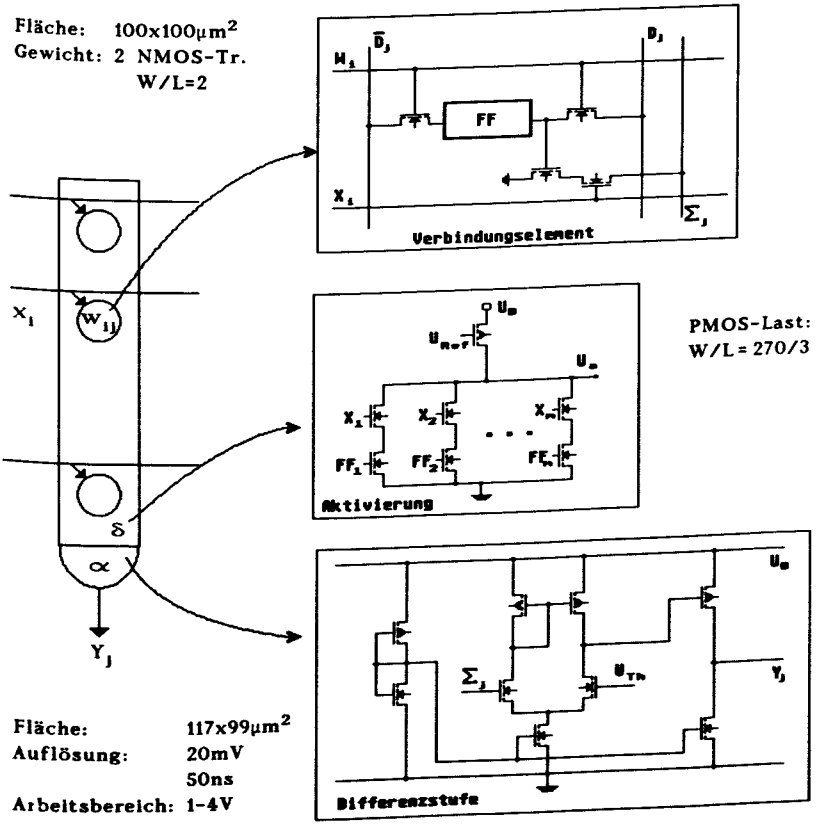
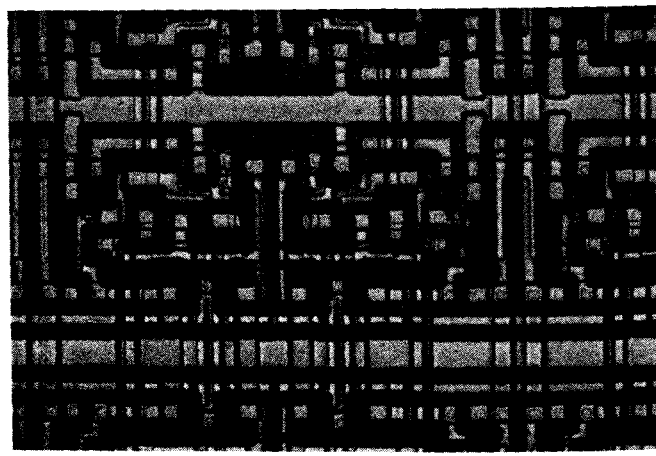
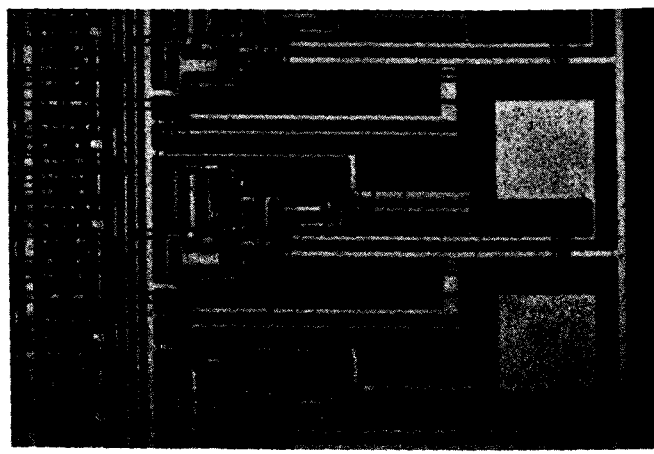


Abb. 5.2.3: Aufbau einer Verarbeitungseinheit in Analogtechnik (ARAM1).



a)



b)

Abb. 5.2.4: Ausschnitt aus der integrierten Verbindungsmatrix (a) und der Ausgabeschaltung (b) vom Testbaustein ARAM1.

Der Baustein ARAM1 dient ausschließlich Testzwecken, um Aussagen über die zu erwartenden technologiebedingten Streuungen der Bauelemente innerhalb einer komplexeren integrierten Schaltung zu erhalten. Derartige Aussagen gibt es für die Dortmunder CMOS-Technologie noch nicht, weil integrierte Schaltungen dieser Größenordnung bisher nicht gefertigt worden sind. Um zusätzliche Fehlerquellen weitgehend auszuschließen, werden neben den zu testenden Verarbeitungseinheiten nur Schieberegister verwendet. Diese lassen sich relativ einfach testen, zusätzlich reduzieren sie die Anzahl der Anschlußpads.

Der Test eines Bausteins erfolgt mit regelmäßigen Belegungsmustern der Verbindungsmatrix (Abb. 5.2.5), die mit Hilfe der Schieberegister einfach und schnell programmiert werden können. Die Programmierung des Belegungsmusters in Abb. 5.2.5a erfolgt z.B. in 128 und die des Musters in Abb. 5.2.5c in 96 Schieberegistertakten. Mit diesen Belegungsmustern können die Auflösegenauigkeit der Verarbeitungseinheiten und ein einfacher Funktionstest der Verbindungselemente durchgeführt werden. Die Auswertung der Ergebnisse gibt Aufschluß über die Auswirkungen von technologiebedingten Streuungen der Bauelementeparameter auf das Systemverhalten. Sie bildet die Grundlage für einen technologieorientierten Entwurf einer hochintegrierten assoziativen Matrix.

Ein weiteres Kriterium für die Auswahl der Architektur von ARAM1 ist die Möglichkeit, einzelne Verarbeitungseinheiten mit einfachen Mitteln auf einem Spitzenmeßplatz untersuchen zu können. Diese Untersuchungen wurden stichprobenartig an verschiedenen ARAM1-Bausteinen durchgeführt. Mit Hilfe dieser Messungen konnte die generelle Funktionsfähigkeit des Testbausteins nachgewiesen werden. Für die Durchführung weiterer Tests wird derzeit am Lehrstuhl Bauelemente der Elektrotechnik der Universität Dortmund eine rechnergestützte Testumgebung entwickelt.

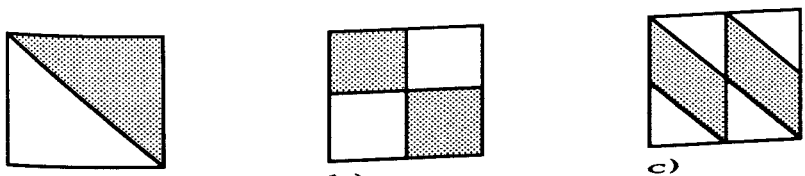


Abb. 5.2.5: Beispiele für Belegungsmuster der Verbindungsmatrix von ARAM1.

5.2.2 Realisierung einer 96x16-Slice einer assoziativen Matrix

Die Organisation der Ein- und Ausgaben beim Baustein ARAM1 ist für die Anwendung einer assoziativen Matrix ungeeignet, weil sie die spärliche Kodierung der Muster nicht ausnutzt. Wie bereits zu Beginn dieses Kapitels erwähnt, ist es sinnvoll nur die Adressen der Komponenten der Ein- bzw. Ausgabemuster zu übertragen, die eine Eins enthalten (x_i=1). Im einfachsten Fall werden die Adressen seriell übertragen, dekodiert und in einem internen Register das Eingabemuster erzeugt (Abb. 5.2.6a). Nach der Übertragung aller Adressen, wird in einem parallelen Assoziationsschritt das Ausgabemuster bestimmt. Die Einsen des Ausgabemusters werden mit Hilfe einer Prioritätsschaltung einzeln kodiert und ausgegeben. Insgesamt sind für die Ein- bzw. Ausgabe der Muster nur (ld m + ld n)-Anschlüsse notwendig. Die Assoziationszeit ergibt sich zu:

$$T_A = (l+k)\cdot t_{adr} + t_{AM} \qquad (5.2.1)$$

Es ist t_{adr} die Zeit zur Übertragung einer Adresse und t_{AM} die Assoziationszeit der assoziativen Matrix in Analogtechnik, die in der Größenordnung der Zugriffszeit von dynamischen Speicherbausteinen liegt (vgl. 4.2.1b).

Eine parallele Übertragung der Ein- bzw. Ausgabemuster läßt sich erreichen, indem die Muster einer zusätzlichen Einschränkung unterworfen werden. Sie müssen sich in festgelegte Abschnitte unterteilen lassen, in denen maximal eine Eins enthalten sein darf. Bei einer gleichmäßigen Unterteilung der Muster in g Abschnitte mit m/g Komponenten lassen

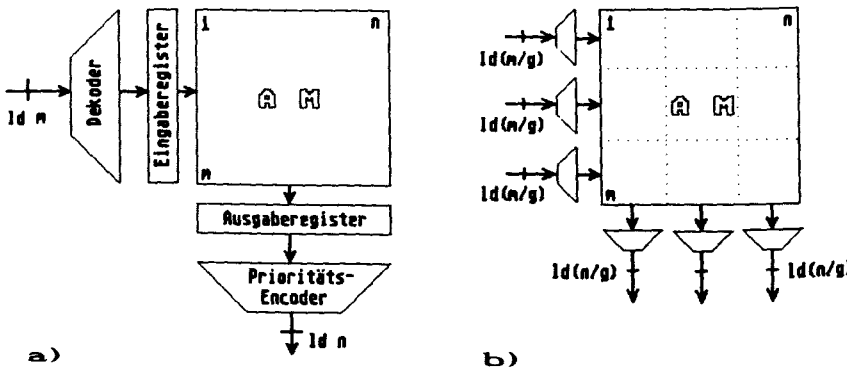


Abb. 5.2.6: Organisation der Ein-/Ausgabe für eine serielle (a) und parallele (b) Übertragung der Muster.

sich die Muster mit l Einsen (l≤g) durch g Adressen mit jeweils ld(m/g) Bit darstellen. Diese Adressen werden parallel an den Baustein übertragen. Die Anzahl p der Ein- bzw. Ausgänge ergibt sich demnach zu:

$$p \geq g \cdot \left(ld(m/g) + ld(n/g) \right) \qquad (5.2.2)$$

Mit l=k=g=ld m folgt für m=n=8192 (bzw. 16384) eine Anzahl p von 242 (bzw. 286) Anschlüssen. Bei der Verwendung von bidirektionalen Ein- bzw.Ausgabeanschlüssen reduziert sich diese Anzahl auf p/2. Die Anzahl der Anschlußpads bleibt folglich auch für sehr große assoziative Matrizen im Rahmen der technologischen Möglichkeiten.

Mit diesem Organisationsprinzip läßt sich die vollparallele Übertragung der Eingabemuster sehr einfach realisieren, indem für jeden Abschnitt ein Dekoder bereitgestellt wird (Abb. 5.2.6b). Es bleibt zu beachten, daß der Informationsgehalt der verwendeten Muster kleiner ist als für die Muster, bei denen die l Einsen überall verteilt sein können:

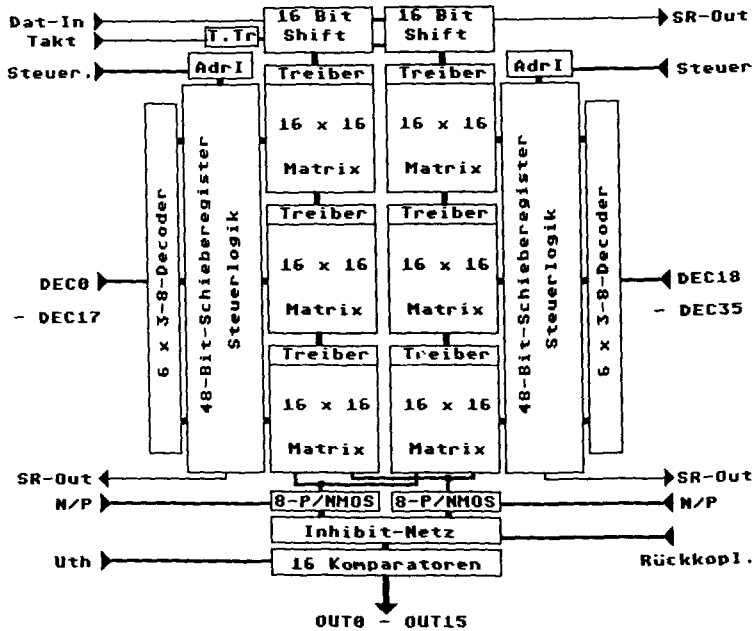
$$ld\binom{n}{k} > ld\left((n/k)^k\right) \qquad (5.2.3)$$

Die Korrektur von Fehlern ist in diesem Fall einfacher, da sie sich besser lokalisieren lassen.

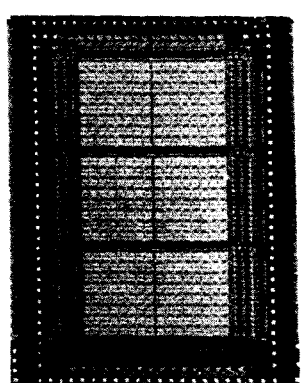
Der zweite im Rahmen dieser Arbeit realisierte Baustein (ARAM2) nutzt das Organisationsprinzip der parallelen Übertragung der Eingabemuster [110]. Der Baustein ARAM2 enthält eine 96x16-Slice (1.5kBit) einer assoziativen Matrix (Abb. 5.2.7). Die Eingabemuster sind in zwölf Abschnitte mit jeweils acht Komponenten unterteilt, in denen maximal nur eine Komponente gleich Eins sein kann. Die Muster werden parallel über zwölf 3-zu-8-Dekoder an die Matrix angelegt. Die 16 Ausgänge werden bei diesem Baustein einzeln ausgeführt, damit die Funktionsfähigkeit der einzelnen Verarbeitungseinheiten unabhängig voneinander überprüft werden kann.

Die Verbindungsmatrix unterteilt sich in sechs Blöcke mit jeweils 16x16-Verbindungen. Die Blöcke sind in zwei Spalten angeordnet, von denen die linke Spalte den Zeilen 0-47 und die rechte Spalte den verbleibenden Zeilen 48-95 entspricht. Beide Spalten können über separate Schieberegister unabhängig voneinander programmiert werden. Die Schieberegister links und rechts von der Verbindungsmatrix können außerdem zur Eingabe der Muster oder zur Überprüfung der Dekoder-Ausgänge verwendet werden. Die Anwendung der Schieberegister erfolgt über Steuereingänge (Beschreibung siehe [111]).

Der Aufbau einer Verarbeitungseinheit entspricht weitgehend dem in Abb. 5.2.3 gezeigten Aufbau einer Verarbeitungseinheit von ARAM1. Die Verbindungselemente sind für ARAM2 um eine zusätzliche Zeilenleitung für eine Referenzspannung U_{Ref} erweitert worden (Abb. 5.2.8).



a)



b)

Verbindungs-elemente:	1536 prog.
Matrixgröße:	96x16
Chipgröße:	7.05x4.95 mm
Fläche:	ca. 34.9 mm^2
Transistoren:	ca. 19000
Anschlüsse:	86

c)

Abb. 5.2.7: Der Baustein ARAM2: Blockschaltbild (a), Schaltungsfoto (b), Kenndaten (c) und Layout (d) (siehe folgende Seite).

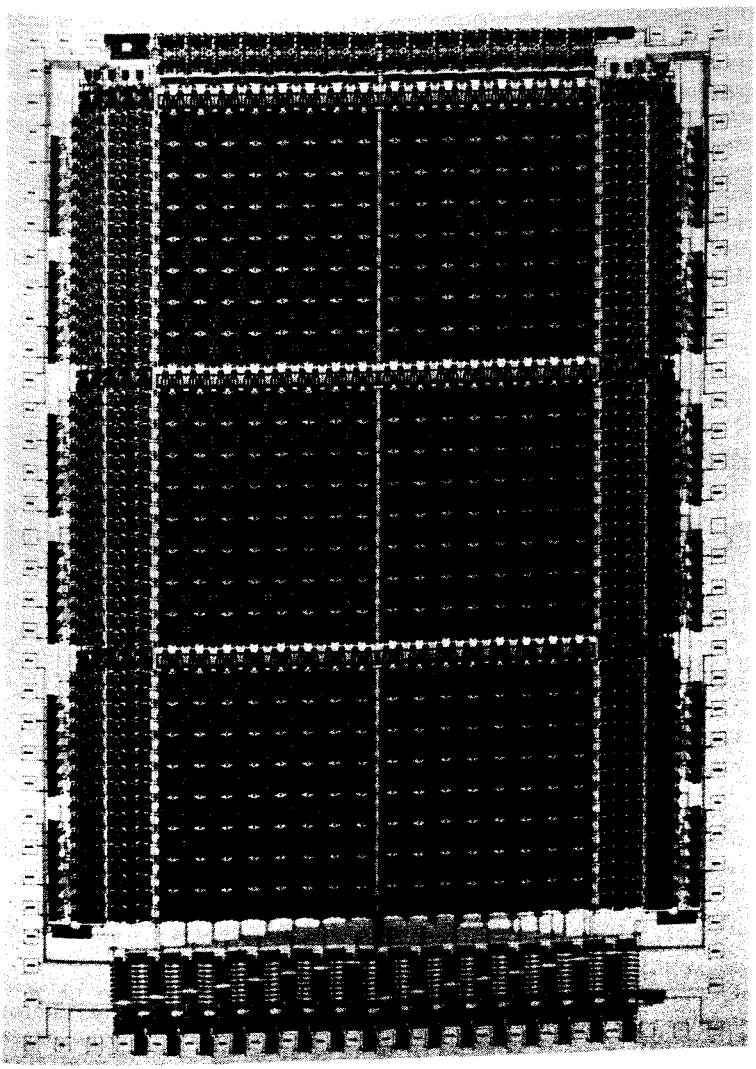
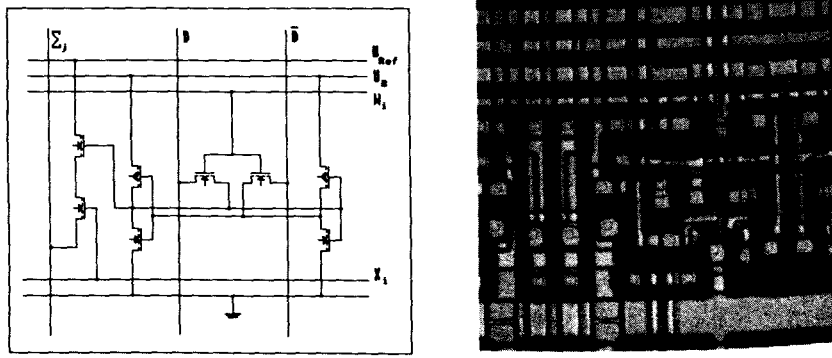


Abb. 5.2.7d: Layout der integrierten Schaltung ARAM2.

In Abhängigkeit von der Wahl der Referenzspannung lassen sich zwei Aktivierungsfunktionen berechnen (vgl. 4.4). Für $U_{Ref}=U_B$ erhält man die einfache Aktivierung mit NMOS-Transistoren und für $U_{Ref}=0V$ eine entsprechende Aktivierung wie bei ARAM1 (Inhibition). Im ersten Fall wird ein weiterer NMOS-Transistor (W/L=40/3) und im zweiten Fall ein PMOS-Lastelement (W/L=80/5) benötigt, die über die Eingänge N/P ausgewählt werden. Es ist möglich, die Referenzspannung U_{Ref} für beide Spalten der Verbindungsmatrix unterschiedlich zu wählen, so daß z.B. die Verbindungen der linken Spalte nur positive ($U_{Ref}=U_B$) und die der rechten Spalte nur negative ($U_{Ref}=0V$) Beiträge zur Aktivierung liefern. Dadurch ist es möglich, verschiedene Varianten der Stromsummation als Aktivierungsfunktion zu untersuchen.



a) b)

Abb. 5.2.8: Schaltbild (a) und Schaltungsfoto (b) eines programmierbaren Verbindungselementes von ARAM2.

Es besteht bei ARAM2 die Möglichkeit, jeweils die ersten (V_0-V_7) und die letzten (V_8-V_{15}) acht Verarbeitungseinheiten über eine direkte, interne Rückkopplung der Ausgänge zu verbinden. Die Rückkopplungsverbindungen sind festprogrammiert (NMOS-Trans. W/L=70/3) und wirken grundsätzlich hemmend (inhibitorisch) auf die Aktivierung einer Verarbeitungseinheit (Inhibit-Netz, Abb. 5.2.7a). Die Rückkopplung einer Verarbeitungseinheit auf sich selbst kann zugelassen werden, ist dann aber nur positiv (exzitatorisch, NMOS-Trans. W/L=6/3).

Mit der internen Rückkopplung läßt sich erreichen, daß von den acht Verarbeitungseinheiten einer Gruppe nur jeweils eine aktiv ist (y_j=1) ("*laterale Inhibition*" [25]). Die Verarbeitungseinheit, die die größte Aktivierung erhält, wird als erste eine Ausgabe y_j=1 erzeugen und damit alle anderen Verarbeitungseinheiten über die inhibitorische Rückkopplung hemmen (winner-takes-all [27]). Aufgrund der Streuungen der Bauelementeparameter ist eine absolute Gleichheit der Aktivierungspotentiale der Verarbeitungseinheiten nicht zu erwarten, so daß sich immer eine Verarbeitungseinheit durchsetzen wird. Diese Erwartung bestätigten sich anhand der ersten Messungen an integrierten ARAM2-Schaltungen.

Die Verwendung der lateralen Rückkopplung ist optional und hier nur als ein Beispiel für ein alternatives Konzept aus dem Bereich der assoziativen Netzwerke gedacht. Unter Umständen kann sie zur Einhaltung der Randbedingung genutzt werden, nach der maximal eine Eins in jedem Abschnitt eines Ausgabemusters enthalten sein darf. Dieses Prinzip ist sicherlich nur für kleine Abschnittsgrößen geeignet, da die Anzahl der lateralen Verbindungen quadratisch mit der Größe des Abschnittes wächst. Aufschluß über die Anwendbarkeit werden weiterführende Untersuchungen liefern.

Die Funktionsfähigkeit der einzelnen Komponenten von ARAM2 ist auf einem Spitzenmeßplatz an verschiedenen integrierten Schaltungen nachgewiesen worden. Die beiden Bausteine ARAM1 und ARAM2 belegen somit, daß eine integrationsgerechte Umsetzung von assoziativen Matrizen mit Schaltungskomponenten in Analogtechnik möglich ist. Die Anzahl der Bausteine CH_{AM}, die zur Realisierung einer vorgegebenen assoziativen Matrix notwendig sind, wird von der erreichbaren Verbindungsdichte (4.3.1) bestimmt. Der Flächenbedarf eines Verbindungselementes wird im allgemeinen größer sein als der einer vergleichbaren binären Speicherzelle. Für einen Vergleich mit dem Slice-Prozessor-Konzept, das auf konventionelle Speicherbausteine zurückgreifen kann, ist es sinnvoll, die Anzahl der notwendigen Bausteine in Abhängigkeit von dem Mehraufwand pro Verbindungselement auszudrücken. Es sei SD entsprechend (4.3.1) die Anzahl von binären Speicherzellen pro cm^2 für die im Slice-Prozessor-Konzept verwendeten Speicherbausteine mit der Speichergröße S. Für den Mehraufwand γ pro Verbindungselement folgt:

$$\gamma = \frac{SD}{VD} \geq 1 \qquad (5.2.4)$$

Unter der Vorraussetzung gleicher Bausteingrößen ergibt sich die Anzahl der Bausteine für das digital-analoge Konzept damit zu:

$$CH_{AM} = \gamma \cdot \frac{m \cdot n}{S} \qquad (5.2.5)$$

Soll die Anzahl CH_{AM} kleiner bleiben als die Anzahl der Bausteine beim Slice-Prozessor-Konzept, dann muß für γ die folgende Bedingung eingehalten werden:

$$\frac{n}{\varkappa} + \frac{m \cdot n}{S} > \frac{m \cdot n}{S} \cdot \gamma$$

$$\Rightarrow \qquad \gamma < 1 + \frac{S}{\varkappa \cdot m} \qquad (5.2.6)$$

Zahlenwerte für das Verhältnis γ sind beispielhaft in Tabelle 5.2.1 aufgeführt. Nach diesen einfachen Abschätzungen benötigt das digital-analoge Konzept immer dann weniger Bausteine, wenn der Flächenaufwand eines Verbindungselementes höchstens um den Faktor γ größer ist als für eine vergleichbare binäre Speicherzelle. Für nichtflüchtige Verbindungselemente (ROM,EEPROM, vgl. 4.3) ist der zusätzliche Flächenaufwand eines Verbindungselementes nur unwesentlich größer als bei festprogrammierten bzw. elektrisch programmierbaren Speicherzellen. In diesem Fall ergibt sich grundsätzlich ein Vorteil für das digital-analoge Konzept. Bei der Verwendung von statischen Speicherzellen erhöht sich der Aufwand pro Verbindungselement um zwei weitere Transistoren und eine Zeilenleitung (vgl. 4.3.2). Der Mehraufwand kann in diesem Fall grob mit 1.5 abgeschätzt werden, so daß bis zu einer Matrixgröße von m=8192 das digital-analoge Konzept günstiger ist.

S \ m	1024	2048	4096	8192	16384
256K: $\varkappa$= 64	9	5	3	2	1.5
1M: $\varkappa$=256	5	3	2	1.5	1.25

Tabelle 5.2.1: Maximal zulässiger Mehraufwand γ (5.2.6) pro Verbindungselement einer assoziativen Matrix in Abhängigkeit von verschiedenen Parametern S,m,$\varkappa$.

Vorteile für das Slice-Prozessor-Konzept ergeben sich bei der Verwendung von dynamischen Speicherzellen, da dynamische Verbindungselemente für eine verteilte Speicherung mit parallelem Zugriff ungünstig sind (vgl. 4.3.2). Der Mehraufwand pro Verbindungselement wird mindestens doppelt so groß sein wie bei einer dynamischen Speicherzelle. Das digital-analoge Konzept bietet hier nur Vorteile für assoziative Matrizen mit weniger als 4096 Verarbeitungseinheiten.

Zusammenfassend liegen die Vorteile des digital-analogen Konzeptes gegenüber dem Slice-Prozessor-Konzept in der flächengünstigeren Realisierung "kleinerer" assoziativer Matrizen ($m \leq 4096$) und in der Möglichkeit einer vollparallelen Arbeitsweise. Eine assoziative Matrix mit bis zu 10^6 programmierbaren Verbindungselementen läßt sich mit heutigen Speichertechnologien auf einem Chip integrieren. In dieser Matrix können ca. 23000 ($k=3, l=10$) Musterzuordnungen mit einer geringen Bit-Fehlerwahrscheinlichkeit (3.1.10) gespeichert werden ($S_{eff}=0.574$). Die Zugriffszeit liegt bei einer seriellen Übertragung der Muster in der Größenordnung von 10µs und bei einer parallelen Übertragung in der Größenordnung von 200ns. Ein derartiger Baustein zeichnet sich durch eine effiziente Informationsspeicherung und einen sehr schnellen assoziativen Zugriff auf die gespeicherten Informationen aus.

5.3 Diskussion

Mit dem Slice-Prozessor-Konzept und dem digital-analogen Konzept sind in diesem Kapitel zwei integrationsgerechte Umsetzungen einer assoziativen Matrix aufgezeigt worden. Die Vorteile der Slice-Prozessor-Architektur gegenüber dem ARAM-Konzept sind die genaue und zuverlässige Berechnung der Aktivierungs- und Ausgabefunktion der Verarbeitungseinheiten sowie die relativ hohe Unempfindlichkeit gegenüber technologiebedingten Streuungen der Bauelementeparameter. Die Verbindungsmatrix wird mit konventionellen Speicherbausteinen realisiert, so daß sich der Entwicklungsaufwand auf den Entwurf der Slice-Prozessoren konzentriert. Die Nachteile der digitalen Implementierung sind der relativ hohe Flächenaufwand einer Verarbeitungskomponente (vgl. 5.1), die semiparallele Arbeitsweise und die Abhängigkeit von der Speicherorganisation handelsüblicher Speicherbausteine.

Die Vorteile des digital-analogen Konzeptes ergeben sich aus den Nachteilen der digitalen Umsetzung und die Nachteile entsprechend aus den Vorteilen. Der Entwurfsaufwand für ein ARAM-Chip ist durch die Verfügbarkeit einer Zellenbibliothek (ASSOCMEM) für assoziative Netzwerke auch ohne eine automatische Entwurfsumgebung vergleichbar mit dem des Slice-Prozessors. Die Zeit zur Bestimmung einer Musterzuordnung ist im Fall der seriellen Übertragung der Muster bei beiden Konzepten vergleichbar. Ein Vorteil der digital-analogen Architektur ergibt sich durch die Möglichkeit einer parallelen Übertragung der Ein- bzw. Ausgabemuster. In Tabelle 5.3.1 sind die Charakteristika der beiden Konzepte noch einmal zusammengefaßt.

Konzept:	T_A	CH
Slice-Prozessor:	$\sim(1 \cdot f_A + k)$	$\frac{n}{x} + \frac{m \cdot n}{S}$
digital/analog :	$\sim(1 + k)$	$\gamma \cdot \frac{m \cdot n}{S}$
parallel :	konst.	

Tabelle 5.3.1: Assoziationszeit T_A und Anzahl der Bausteine CH für die vorgestellten Realisierungskonzepte einer assoziativen Matrix.

Die in dieser Arbeit vorgestellten Systemkonzepte lassen sich in das zur Zeit noch sehr junge Entwicklungs- und Forschungsgebiet der neuronalen ASICs einordnen (vgl. Kapitel 1). Es gibt bisher nur sehr wenige vergleichbare Schaltungsrealisierungen für assoziative Netzwerke, die sich überwiegend dem Hopfield-Netz [74,77,84] widmen (Tabelle 5.3.2).

	Größe [mm]	Ein-/ Ausgabe	Gewichte (prog.)	Technologie
Sivilotti/ Mead [77]	6.7x5.7	parallel	22x22= 484	4µm NMOS
Graf/ [84] deVegvar	6.7x6.7	seriell	54x54= 2916	2.5µm CMOS
Graf/ [112] Hubbard	6.7x6.7	seriell	96x46= 4416	2.5µm CMOS
Verleysen/ Jespers [77]	3x3	parallel	14x14= 196	3µm CMOS
ARAM1	7x8	seriell	64x64= 4096	3µm CMOS
ARAM2	7x5	parallel	96x16= 1536	3µm CMOS

Tabelle 5.3.2: Übersicht ausgewählter Schaltungsrealisierungen assoziativer Netzwerke mit programmierbaren Verbindungselementen und analoger Berechnung der Aktivierungsfunktion.

In Tabelle 5.3.2 sind die bekanntesten Realisierungen aufgeführt, die sich wie das ARAM-Konzept einer analogen Berechnung der Aktivierungsfunktion bedienen und programmierbare Verbindungselemente aufweisen. Im Vergleich zu diesen Systemkonzepten verfügt die entwickelte ARAM2-Architektur über einen einfacheren Aufbau, eine höhere Speichereffektivität und einen schnelleren assoziativen Datenzugriff. Entsprechende Aussagen gelten für digitale Realisierungsvorschläge im Vergleich zur Slice-Prozessor-Architektur [113,114]. Das ARAM- und das Slice-Prozessor-Konzept lassen sich zudem relativ einfach auf Netzwerkgrößen mit 10^3-10^4 Verarbeitungseinheiten ausweiten, die für Anwendungen interessant werden. Für die anderen bekannten Systementwürfe neuronaler ASICs ist eine Erweiterung zu derart großen Netzwerken nicht ohne weiteres möglich.

Das ARAM- und das Slice-Prozessor-Konzept erlangen ihre Leistungsfähigkeit durch einen hohen Grad an Parallelität und der Ausnutzung modellspezifischer Eigenschaften der assoziativen Matrix. Die Ausnutzung der spärlichen Kodierung reduziert die Anzahl der Anschlußpads einer integrierten assoziativen Matrix auf eine technologisch beherrschbare Größenordnung, ohne den assoziativen Zugriff auf gespeicherte Muster nennenswert zu verzögern. Die Beschränkung auf binäre Ein- bzw. Ausgabemuster und Verbindungsgewichte führt zu einfachen Berechnungen innerhalb einer Verarbeitungseinheit sowie zu einer sehr hohen Verbindungsdichte. Beide Eigenschaften ermöglichen eine Realisierung in analoger Schaltungstechnik und damit auch eine vollparallele Arbeitsweise für große assoziative Matrizen ($m,n>10^3$). Das verteilte Speicherprinzip gewährleistet den fehlertoleranten und assoziativen Datenzugriff.

Eine Zusammenstellung wichtiger Systemgrößen einer assoziativen Matrix und ihrer Realisierungskonzepte findet sich in Tabelle 5.3.3. Die Anzahl der Musterpaare z, die mit einer geringen Bit-Fehlerwahrscheinlichkeit gespeichert werden können, ist mit der Bedingung (3.1.10) ermittelt worden. Die mit '*' gekennzeichneten Zeilen beziehen sich auf unvollständige Eingabemuster (3.2.1a). Die mit '**' gekennzeichneten Zeilen entsprechen dem Fall, daß nur ein Anteil von $c=0.95$ der Verbindungsgewichte korrekt funktioniert und die defekten Verbindungselemente keinen Einfluß auf das Systemverhalten haben (3.2.8). Die Berechnung der Anzahl der notwendigen Bausteine CH (5.2.6) und des Faktors f_A (5.1.15) geht von einer 128Kx8-Speicherorganisation der beim Slice-Prozessor-Konzept verwendeten RAM-Bausteine aus.

Ein wesentliches Merkmal der assoziativen Matrix ist, daß sehr viel mehr Musterzuordnungen gespeichert werden können als Verarbeitungseinheiten vorhanden sind (3.1.12). Bezogen auf Tabelle 5.3.3 haben die Ausgabemuster je nach Größe einen Informationsgehalt von 20-40 Bit ($k=3$) und die zu erreichende Speichereffektivität liegt bei ca. 0.6. Die Zugriffszeit ist für alle Daten gleich und insbesondere unabhängig von der Anzahl der ge-

speicherten Musterzuordnungen. Weitere Musterpaare können jederzeit hinzugefügt werden. Eine Überprüfung, ob ein Musterpaar bereits gespeichert ist, ist nicht erforderlich.

Die assoziative Matrix ist darüber hinaus in der Lage, auch unvollständige oder fehlerhafte Eingabemuster richtig abzubilden. Es handelt sich nicht nur um Ein-Bit-Fehler, sondern es können je nach Auslastung der Matrix mehrere Einsen in der Eingabe fehlen (vgl. 3.2.1). Zum Beispiel lassen sich in eine assoziative Matrix mit n=m=8192 noch 930175 Muster mit einer geringen Bit-Fehlerwahrscheinlichkeit zuordnen, wenn die Eingabemuster mindestens 16 der geforderten 20 Einsen enthalten.

m=n	l	Z	S_{eff}	MBit	CH			
					Slice-Prozessor			ARAM
					x=64	x=256	f_A	γ=1.5
1024:	10	23313	0.565	1	17	5	128	1-2
l'=8 *	10	18296	0.450					
**	13	17482	0.431					
2048:	11	86012	0.591	4	36	12	64	6
l'=8 *	10	66475	0.457					
**	14	64430	0.443					
4096:	12	318353	0.604	16	70	32	32	24
l'=16 *	20	246628	0.468					
**	15	238785	0.453					
8192:	13	1182378	0.613	64	192	96	16	96
l'=16 *	20	930175	0.483					
**	16	888947	0.461					
16384:	14	4407707	0.621	256	512	320	8	384
l'=16 *	20	3500175	0.493					
**	16	3325531	0.469					

Tabelle 5.3.3: Zusammenstellung wichtiger Systemgrößen einer assoziativen Matrix und ihrer Realisierungskonzepte (Heteroassoziation, k=3).

Die Speichereffektivität und die Fehlertoleranz verbessern sich für größere assoziative Matrizen (Tabelle 5.5.3). Darin begründet sich noch einmal die Forderung nach einer geeigneten Hardwareimplementierung, damit auch für sehr große Matrizen der schnelle assoziative Datenzugriff gewährleistet bleibt (Echtzeit). Mit den in dieser Arbeit entwickelten Architekturen lassen sich Zugriffszeiten im Mikrosekundenbereich erzielen. Unter Berücksichtigung der Speichereffektivität, Fehlertoleranz und Zugriffszeit ergibt sich ein sehr leistungsfähiges Systemkonzept, das im Bereich der assoziativen Informationsverarbeitung gemäß Kapitel 2 seine Anwendung findet und dort konventionellen Lösungen überlegen ist.

Neben dem schnellen assoziativen Datenzugriff ist für Anwendungen auch eine kompakte Realisierung entscheidend. Die Verfügbarkeit von Sub-µm-Technologien [115] ermöglicht die Integration von Ein-Chip–Lösungen für assoziative Matrizen mit mehreren tausend Verarbeitungseinheiten, die für einen mobilen Einsatz (Roboter, Telefon) wichtig sind. Ein assoziativer Baustein mit m=n=2048 kann bei einer Speichereffektivität von ca. 0.6 mehr als 86000 Musterzuordnungen unter Einhaltung der Bedingung (3.1.10) speichern (Tabelle 5.3.3). Für den Fall, daß 5% der Verbindungselemente defekt sind, ist der Baustein noch in der Lage ca. 64000 Musterzuordnungen (S_{eff}=0.443) mit einer geringen Bit-Fehlerwahrscheinlichkeit auszuführen, sofern diese Defekte nicht systematisch auftreten.

Mit der Wafer-Scale-Integration [68] eröffnet sich die Möglichkeit, eine assoziative Matrix mit m=n=16384 inklusive notwendiger Steuer- und Peripherieschaltungen auf einer Silizium-Scheibe zu integrieren [78]. Eine derart kompakte Realisierung, die mehr als 4 Millionen Musterzuordnungen (S_{eff}=0.621) speichern und einen assoziativen Datenzugriff in Echtzeit ausführen kann, könnte Arbeitsplatzrechner um eine wertvolle Komponente zur assoziativen Informationsverarbeitung bereichern.

Die Forderung nach großen assoziativen Matrizen verdeutlicht, daß die Leistungsfähigkeit dieses Systemkonzeptes erst mit den Fortschritten der Mikroelektronik vollständig zum Tragen kommt. Auf der anderen Seite sind die erarbeiteten Systemarchitekturen sehr gut für die Großintegration geeignet, weil sie aufgrund ihres sehr einfachen Aufbaus die Möglichkeiten einer Standardtechnologie weitgehend ausnutzen können und die Entwurfskomplexität auf ein beherrschbares Maß reduzieren.

6. Zusammenfassung und Ausblick

Neuronale Netzwerkmodelle gewinnen derzeit in verschiedenen Wissenschaftsgebieten zunehmend an Bedeutung. Ihre hochgradig parallele und kollektive Arbeitsweise lassen sie zu einem vielversprechenden alternativen Lösungsansatz für Problemstellungen zum Beispiel im Bereich der Mustererkennung werden. Ihre ausgesprochen modulare und reguläre Architektur sowie die systeminhärente Fehlertoleranz machen sie ebenfalls attraktiv für die Großintegrationstechnik. Diese Eigenschaften sind notwendige Voraussetzungen für die Beherrschung der Entwurfs- und Testkomplexität hochintegrierter Systeme.

Die vollständige Leistungsfähigkeit erreichen neuronale Netzwerkmodelle nur dann, wenn ihre Parallelität von einer angepaßten Rechner- oder Schaltungsstruktur unterstützt wird. Die integrationsgerechte Umsetzung von neuronalen Netzwerken in eine anwendungs- und modellspezifische VLSI-Architektur (neuronales ASIC) war zentrales Thema dieser Arbeit.

Im Mittelpunkt standen assoziative Netzwerke, die eine Untermenge der neuronalen Netze bilden und deren grundlegende Aufgabenstellung die Musterabbildung bzw. -vervollständigung von binären Mustern ist. Für diese Netzwerke wurde eine einheitliche Beschreibungsform angegeben, mit der sich viele der bekannten Modelle spezifizieren und strukturell vergleichen lassen. Der neue und für den Systementwickler wichtige Aspekt dieser Beschreibungsform ist, daß der strukturelle Aufbau assoziativer Netzwerke deutlicher zum Ausdruck kommt. Diese Eigenschaft ist für eine systematische Umsetzung in eine integrationsgerechte Architektur wichtig.

Anhand dieser Beschreibungsform wurden der allgemeine Aufbau assoziativer Netzwerke dargestellt und zwei grundlegende assoziative Modelle spezifiziert (assoziative Matrix, Hopfield-Netz). Das verteilte Speicherprinzip und das Schwellenwertverhalten der Verarbeitungseinheiten sind Eigenschaften dieser Modelle, die dem Bereich der Neurophysiologie entlehnt sind. Die assoziative Matrix nutzt ferner die Annahme, daß im Gehirn nur relativ wenige Neuronen (Verarbeitungseinheiten) zur gleichen Zeit aktiv sind (spärliche Kodierung).

Auf der Grundlage von Angaben aus der Literatur wurden, in dieser geschlossenen Form erstmalig, die für die Mikroelektronik relevanten Eigenschaften der Speichereffektivität und Fehlertoleranz ausgewählter Modelle quantitativ erfaßt. Das wesentliche Ergebnis dieser Untersuchung ist, daß hinsichtlich der betrachteten Anwendung die assoziative Matrix als einfachste Netzwerkstruktur die günstigste Speichereffektivität und Fehler-

toleranz aufweist. Unter Verwendung einer spärlichen Kodierung und der Hebb-Regel als Adaptationsregel kann eine maximale Speichereffektivität von 0.69 erreicht werden. Fehler können auf Kosten der Speichereffektivität sowohl in der Eingabe als auch in der Systemstruktur kompensiert werden, ohne Änderungen an der Architektur vornehmen zu müssen.

Die Leistungsfähigkeit assoziativer Netzwerke steigt mit der Anzahl der Verarbeitungseinheiten, d.h. es ist eine effiziente Umsetzung dieser Systeme in eine angepaßte Schaltungsstruktur gefordert. Die Mikroelektronik bietet verschiedene Realisierungsalternativen, von denen die wichtigsten in dieser Arbeit diskutiert wurden. Den Schwerpunkt bildeten analoge Schaltungskonzepte für die Bestimmung der Aktivierungsfunktion und Realisierungsalternativen für die Verbindungselemente. Die Funktionsweise ausgewählter Grundschaltungen wurde anhand von eigens zu diesem Zweck entworfenen und realisierten Testschaltungen überprüft.

Der neue Gesichtspunkt dieser Untersuchungen lag in der Bewertung der verschiedenen Schaltungskonzepte hinsichtlich ihrer Auswirkungen auf die Speichereigenschaften assoziativer Netzwerke. Es zeigte sich, daß für die assoziative Matrix die Nachteile der Analogtechnik (Berechnungs- und Auflöseungenauigkeiten, höhere Technologieabhängigkeit) aufgrund der spärlichen Kodierung und der Verwendung von binären Verbindungselementen keine wesentlichen Einschränkungen bedeuten. Gravierend wirken sich dagegen die Nachteile auf assoziative Netzwerkmodelle aus, bei denen jeweils alle Eingänge einen Beitrag zur Aktivierung leisten sowie mehrwertige Verbindungsgewichte verwendet werden. Die einfachsten Schaltungsrealisierungen und die größte Verbindungsdichte ergeben sich für die assoziative Matrix.

Erstmalig wurde im Rahmen dieser Arbeit die Eignung von Floating-Gate-Transistoren für die Realisierung adaptiver Verbindungselemente untersucht und bewertet. Das hier vorgestellte adaptive Verbindungselement mit Floating-Gate-Transistor ist ein geeignetes Beispiel für die funktionale Integration einer Systemfunktion (Adaptationsregel) unter Ausnutzung physikalischer Eigenschaften eines einzelnen Bauelementes. In dieser Hinsicht eröffnen sich neue Perspektiven für die Analogtechnik, da digitale Realisierungen adaptiver Verbindungselemente einen wesentlich höheren Flächenaufwand aufweisen.

Auf der Grundlage der erarbeiteten theoretischen und praktischen Ergebnisse wurden schließlich zwei neue VLSI-Architekturen entworfen. Es handelt sich um ein rein digitales und ein digital-analoges Systemkonzept für eine assoziative Matrix. Beide Konzepte zeichnen sich durch einen modularen und regulären Systemaufbau, eine effiziente Informationsspeicherung und einen sehr schnellen assoziativen Zugriff auf gespeicherte Daten aus. Die integrationsgerechte Realisierung einer assoziativen Matrix

wurde mit den Bausteinen ARAM1 und ARAM2 erfolgreich durchgeführt, die zu den derzeit größten realisierten neuronalen ASICs gehören. Beide Bausteine wurden am Lehrstuhl Bauelemente der Elektrotechnik der Universität Dortmund gefertigt und hinsichtlich ihrer generellen Funktionsfähigkeit getestet.

Gegenüber vergleichbaren Realisierungen neuronaler ASICs verfügen die in dieser Arbeit entwickelten Architekturen über den wesentlichen Vorteil, daß sie sich einfach auf Netzwerkgrößen mit mehr als 10^3 Verarbeitungseinheiten erweitern lassen und damit für Anwendungen interessant werden. Mit der Verfügbarkeit einer Sub-µm-Technologie und den weiteren Fortschritten in der Mikroelektronik wird die Leistungsfähigkeit der entwickelten Systemkonzepte damit vollends zum Tragen kommen.

Assoziative und neuronale Netzwerke werden die herkömmliche Informationsverarbeitung nicht verdrängen, sondern in bestimmten Anwendungen ergänzen, wie etwa ein Gleitkomma-Prozessor die Rechengeschwindigkeit von Datenverarbeitungsanlagen beschleunigt. Dabei ist es denkbar, daß solche mikroelektronischen Bauelemente bei einem elektronischen Notizbuch oder bei der Robotersteuerung erste Anwendungen finden. Man darf gespannt sein, welche Bedeutung diese neuen Systemkonzepte bei den zu erwartenden Technologie-Entwicklungen neben den heute dominierenden digitalen Rechnerarchitekturen erlangen werden.

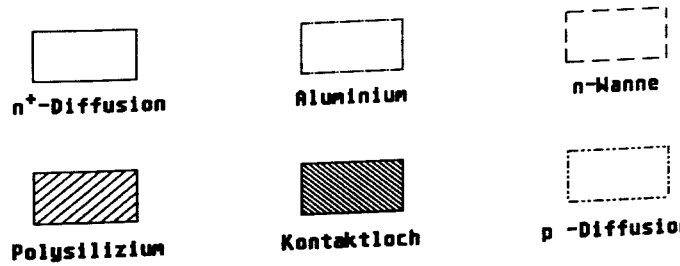
7. Verzeichnis der verwendeten Symbole

A	Menge der Aktivierungswerte
A_C	Kondensatorfläche
A_G	wirksame Injektionsfläche für den Tunnelstrom
AM	Assoziative Matrix
AN	Assoziatives Netzwerk
AS	Verarbeitungseinheit einer assoziativen Matrix
a	Aktivierungswert einer Verarbeitungseinheit
b	Wortbreite, Bitanzahl
C	Kapazität
CH	Anzahl der Bausteine für eine assoziative Matrix
CH_{AM}	Anzahl der ARAM-Slice-Bausteine
CH_{ASS}	Anzahl der Slice-Prozessoren
$\overline{D}$,D	Datenleitungen
d	Schichtdicke
d_{ox}	Dicke der SiO_2-Schicht
d_{cox}	Schichtdicke des Kodensatoroxides
E	Energie
E_{ox}	elektrisches Feld zwischen Floating-Gate und Substrat
$E(N_t)$	Erwartungswert für die Bitfehleranzahl
$E(I_H)$	Erwartungswert für den Informationsgehalt einer assoziativen Matrix (Heteroassoziation)
FG	Floating-Gate
g	Blockgröße
H	Hash-Funktion
HN	Hopfield-Netz
HOP	Verarbeitungseinheit eines Hopfield-Netzes
I	Menge der Eingabevektoren
I_D	Drainstrom eines MOS-Transistors
I^i	Informationsgehalt des i-ten Musters
I^i_{Fehler}	Information zur Behebung eines Fehlers im Ausgabemuster
k	Anzahl der Einsen im Ausgabemuster
k_n	Transistorsteilheitskonstante für NMOS-Transistoren
k_p	Transistorsteilheitskonstante für PMOS-Transistoren
L	Kanallänge eines MOS-Transistors
l	Anzahl der Einsen im Eingabevektor
l'	Anzahl der Einsen eines unvollständigen Eingabevektors
l^f	Anzahl der fehlerhaften Einsen im Eingabevektor
l^r	Anzahl der korrekten Einsen im Eingabevektor
ld	Logarithmus-Dualis
ln	natürlicher Logarithmus
log	Logarithmus allgemein
m	Anzahl der Eingabeleitungen/-werte einer Verarbeitungseinheit
max{..}	Maximum einer Menge

$\min\{...\}$	Minimum einer Menge
$\mathbb{N}$	Menge der natürlichen Zahlen {0,1, ...}
N_t	Bitfehleranzahl
n	Anzahl der Verarbeitungseinheiten eines AN
O	Menge der Ausgabevektoren
$O(x)$	in der Größenordnung von x
P	Verlustleistung
$p(x)$	Wahrscheinlichkeit von x
p_{on}	Wahrscheinlichkeit, mit der eine Verbindung einer AM Eins ist
Q_{FG}	Ladung auf dem Floating-Gate
$\mathbb{R}$	Menge der reellen Zahlen
R	Widerstand
$R_{[]}$	Schichtwiderstand
r	Widerstandverhältnis R_a/R_l
S_{eff}	Speichereffektivität
SA	Speicheraufwand
SD	Anzahl digitaler Speicherzellen pro cm^2
t	diskrete Zeitvariable
Th	Schwellenwert
t_{SPZG}	Speicherzugriffszeit
t_{ASS}	Taktzyklus des Slice-Prozessor-Bausteins
t_{adr}	Übertragungszeit einer Adresse
t_{AM}	Assoziationszeit eines ARAM-Bausteins
U	Spannung allgemein
U_a	Aktivierungs- bzw. Ausgangsspannung
U_B	Versorgungspannung (5V)
U_{Ref}	Referenzspannung
U_T	Schwellenspannung einen MOS-Transistor
V	Verarbeitungseinheit
VD	Verbindungsdichte
$\mathbf{W}$	Verbindungsmatrix
W	Kanalweite eines MOS-Transistors/Menge der Gewichtswerte
$\underline{\mathbf{w}}$	Gewichtsvektor
w	Verbindungsgewicht
X	Menge der Eingabewerte
$\mathbf{X}$	Matrix der Eingabevektoren
$\underline{\mathbf{x}}$	Eingabevektor einer Verarbeitungseinheit
x	Eingabewert bzw. Eingabeleitung einer Verarbeitungseinheit
$\mathbf{Y}$	Matrix der Ausgabevektoren
Y	Menge der Ausgabewerte
$\underline{\mathbf{y}}$	Ausgabevektor aller Verarbeitungseinheiten
y	Ausgabewert bzw. Ausgabeleitung einer Verarbeitungseinheit
y_l	Lehrersignal
$\underline{\mathbf{z}}$	Vektor der externen Ausgaben eines ANs
z	Anzahl der gespeicherten Musterpaare

α	Ausgabefunktion einer Verarbeitungseinheit
β_r	Geometrieverhältnis von MOS-Transistoren
γ	SD/VD
Δ	Abweichung allgemein
ΔU	Spannungsänderung
δ	Aktivierungsfunktion einer Verarbeitungseinheit
ε_0	elektrische Feldkonstante
ε_{ox}	Dielektrizitätszahl des SiO_2
$\varkappa$	Anzahl der Verarbeitungskomponenten eines Slice-Prozessors
λ	Adaptationsfunktion einer Verarbeitungseinheit
μ_n	Elektronenbeweglichkeit
μ_p	Löcherbeweglichkeit
ν	Anzahl der negativen Summanden
π	Anzahl der positiven Summanden
ρ	spezifischer Widerstand
Σ	Summenleitung
σ	Varianz
τ	Zeitkonstante
Φ	Taktsignal
$\mathfrak{E}$	endliche Menge der Eingabequellen
ϵ	Abbildung, die die Verbindungen zwischen externen Eingaben und Verarbeitungseinheiten festlegt
$\mathfrak{p}$	Bit-Fehlerwahrscheinlichkeit
$\mathfrak{V}$	endliche Menge von Verarbeitungseinheiten
$\mathfrak{v}$	Abbildung, die die interne Verbindungsstruktur festlegt
$\mathfrak{z}$	Abbildung, die die Ausgabeeinheiten eines Netzwerkes festlegt

Legende zu den Layouts:



8. Literaturverzeichnis

[1] Goser, K., Rückert, U.: "Künstliche Intelligenz - eine Herausforderung an die Großintegrationstechnik", Nachrichtentechnische Zeitschrift, Heft 11 (1986), S. 748-752.

[2] Hecht-Nielsen, R.: "The ANZA Neurocomputing system", Hecht-Nielsen Neurocomputer Corporation, San Diego, 1987.

[3] Ramacher, U., Pandel, J.: "Eine Hardware-Architektur für die schnelle Emulation von künstlichen neuronalen Netzen beliebiger Größe", Interner Bericht der Firma Siemens, ZFE ME22, 1988.

[4] Hillis, W.D.: "The Connection Machine", Cambridge, Mass.: MIT Press, 1985.

[5] Recce, M., Treleaven, P.C.: "Parallel Architectures for Neural Computers", in: *Eckmiller, R., v.d. Malsberg, Ch. (Hrsg.)*: "Neural Computers" NATO ASI Series, Serie F, Vol. 41, Berlin: Springer Verlag 1988, S. 487-495.

[6] Hilleringmann, U.: "Laserrekristallisation von Silizium: Integration von CMOS-Schaltungen auf isolierendem Substrat", Dissertation Universität Dortmund, Bauelemente der Elektrotechnik, 1988.

[7] Kandel, E.R., Schwarz, J.H.: "Prinicples of Neural Science", 2 Aufl., New York: Elsevier, 1985.

[8] Churchland, P.S.: "Neurophilosophie", Cambridge, Mass.: MIT Press, 1986.

[9] Barr, M.L., Kiernan, J.A.: "The Human Nervous System", 5.Aufl., Philadelphia: Lippincott Company, 1988.

[10] Hebb, D.O.: "The Organization of Behaviour", John Wiley, New York, 1949.

[11] Koch, C., Poggio, T., Torre, V.: "Micronetworks in Nerve Cells", in: Lecture Notes in Biomathematics, *Amari, S., Arbib, M.A. (Hrsg.)*, Vol. 45, Berlin: Springer Verlag (1982), S. 105-110.

[12] Braitenberg, V.: "Cell Assemblies in the Cerebral Cortex", in: *Heim, R., Palm, G. (Hrsg.)*: "Theoretical approaches to complex systems", Berlin: Springer Verlag 1978, S. 171-188.

[13] Rosenblatt, F.: "Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms", Washington: Spartan Books, 1961.

[14] Uttely, A.M.: "The Design of Conditional Probability Computers", Information and Control 2 (1959), S. 1-24.

[15] Steinbuch, K.: "Die Lernmatrix", Kybernetik, Bd. 1, Heft 1 (1961), S. 36-45.

[16] Widrow, B.: "Generalization and Information Storage in Networks of Adaline Neurons", in: *Yovits, M.C., u.a. (Hrsg.)*: "Self-Organizing Systems", Washington: Spartan Books, 1962, S. 435-462.

[17] McCulloch, W.S., Pitts, W.A.: "A logical calcalus of the ideas immanent in nerveous activity", Bull. Math. Biophys. 5 (1943), S. 115-133.

[18] Kleene, S.C.: " Representation of events in nerve nets and finite Automata", in: *Shannon, McCarthy (Hrsg.)*: "Automata Studies", Princeton 1956.

[19] Minsky, M., Papert, S.: "Perceptrons: An Introduction to Computational Geometry", 2. Auflg., Cambridge Mass.: MIT Press 1972.

[20] Shaw, G., Palm, G. (Hrsg.): "Brain Theory - Reprint Volume". London: World Scientific Publ., 1988.

[21] Anderson, J.A., Rosenfeld, E. (Hrsg.): "Neurocomputing", Cambridge: Mass.: MIT Press, 1988.

[22] Amit, D.J., Gutfreund, H.: "Storing Infinite Numbers of Patterns in a Spin-Glass Model of Neural Networks", Phys. Review Letters, Vol. 55, No. 14 (1985), S. 1530-1533.

[23] Hopfield, J.J.: "Neural Networks and physical systems with emergent collective computational abilities", Proc. Nat. Acad. Sci. USA, Vol. 79 (1982), S. 2554-2558.

[24] Ackley, D.H., Hinton, G.E., Sejnowski, T.J.: "A learning algorithm for Boltzmann machines", Cognitive Science 9 (1985), S.147-169.

[25] Kohonen, T.: "Self-Organization and Associative Memory", Berlin: Springer-Verlag, 1984.

[26] Grossberg, S.: "The adaptive Brain, I: Cognition, learning, reinforcement, and rhythm", Amsterdam: Elsevier/North-Holland, 1987.

[27] Rumelhart, D.E., McClelland, J.L.: "Parallel Distributed Processing", Vol. 1 : Foundations, Cambridge, Mass.: MIT Press, 1986.

[28] Goser, K.: "Mikroelektronik neuronaler Netze", Mikroelektronik, Bd. 3, Heft 3 (1989), S. 104-108.

[29] Kemke, C.: "Modelling Neural Networks by Means of Networks of Finite Automata", Proc. First Int. Conf. on Neural Networks, San Diego, (1987), S. 21-24.

[30] Palm, G.: "Towards a Theory of Cell Assemblies", Biol. Cybern. 39 (1981), S. 181-194.

[31] Kinzel, W.: "Learning and Pattern Recognition in Spin Glass Models", Z. Physik, B60 (1985), S. 205-213.

[32] Surmann, H.: "Untersuchung der Hardware-Fehlertoleranz ausgewählter adaptiver assoziativer Speicherkonzepte basierend auf neuronalen Netzen, Diplomarbeit, Universität Dortmund, Bauelemente der Elektrotechnik, 1989.

[33] Ben-Israel, A.,Greville, T.N.E.: "Generalized Inverses: Theory and Applications", New York: Wiley, 1974.

[34] Willshaw, D.J., Bunemann, O.P., Longuet-Higgins, H.C.: "A Non-Holographic Model of Associative Memory", Nature 222, Nr. 5197 (1969), S. 960-962.

[35] Palm, G.: "Neural Assemblies", Berlin: Springer-Verlag, 1982.

[36] Arbib, M.A.: "Brains, Machines, and Mathematics", 2.Aufl., New York: Springer-Verlag, 1987.

[37] Fogelman Soulié, F.: "Lyapunov functions and their use in automata networks", in: "Disordered Systems and Biological Organization", *Bienenstock, E., u.a. (Hrsg.)*, NATO ASI Series F, Vol. 20, Berlin : Springer, 1986, S. 85-100.

[38] Barroca da Silva, J.: "Anwendung von adaptiven Assoziativspeichern zur Speicherung und Verarbeitung von einfachen digitalen Bildern", Diplomarbeit, Universität Dortmund, Lehrstuhl Bauelemente der Elekrotechnik, 1987.

[39] Dittrich, M.: "Simulation und CMOS-VLSI-Entwurf eines assoziativen Speichers nach dem Prinzip der "Collective Computation"", Diplomarbeit, Universität Dortmund, Lehrstuhl Bauelemente der Elekrotechnik, 1987.

[40] Dittrich, M.: "Simulation eines digitalen Assoziativspeichers auf einem Personal Computer", Studienarbeit, Universität Dortmund, Lehrstuhl Bauelemente der Elektrotechnik, 1986.

[41] Hopfield, J.J., Tank, D.W.: "Neural Computation of Decisions in Optimization Problems", Biol. Cybern. 52 (1985), S. 141-157.

[42] Garey, M.R., Johnson, D.S.: "Computers and intractability", New York : Freemann, W.H., 1979.

[43] Sejnowski, T.J., Rosenberg, C.R.: "NETtalk: A Parallel Network that Learns to Read Aloud", Technical Report JHU/EECS-86/01, Johns Hopkins University, Baltimore, 1986.

[44] Hecht-Nielsen, R.: "Neurocomputer applications", AFIPS Conference Proceedings, National Computer Conference, Vol.56, (1987), S. 239-244.

[45] Goser, K., Rückert, U.: "Intelligent VLSI-Memories For Robotics", Cognitiva 85, Paris (1985), Tagungsband S. 425-430.

[46] Shannon, C.E., Weaver, W.: "The mathematical theory of communication", Illinois: University of Illinois Press, 1949.

[47] Palm, G.: "On Associative Memory", Biol. Cybern., 36 (1980), S. 19-31.

[48] McEliece, R.J., u.a.: "The Capacity of the Hopfield Associative Memory", IEEE Trans. of Inf. Theory, Vol IT-33, No. 4 (1987), S. 461-482.

[49] Amari, S., Magina, K.: "Statistical Neurodynamics of Associative Memory", Neural Networks, Vol. 1 (1988), S. 63-73.

[50] Forshaw, M.R.B.: "Pattern Storage and Associative Memory in Quasi-Neural Networks", Report No. 8615, Image Processing Group, University College London, 1986.

[51] Montgomery, B.L., Vijaya Kumar, B.V.K.: "Evaluation of the use of the Hopfield neural network model as a nearest-neighbor algorithm", Applied Optics, Vol. 25, No. 20 (1986), S. 3759-3766.

[52] Stiles, G.S., Deng, D.-L.: "On the Effect of Noise on the Moore-Penrose Generalized Inverse Associative Memory", IEEE Trans. on Pattern Anal. and Mach. Int., Vol. PAMI-7, No. 3 (1985), S. 358-360.

[53] Stiles, G.S., Deng, D.-L.: "A Qualitative Comparison of the Performance of Three Discrete Distributed Associative Memory Models", IEEE Trans. on Comp., Vol. C-36, No. 3 (1987), S. 257-263.

[54] Willshaw, D.J., Longuet-Higgins, H.C.: "Associative memory models", Machine Intelligence 5, Edindurgh: University Press (1970), S. 351-359.

[55] Palm, G.: "On the Storage Capacity of an Associative Memory with Randomly Distributed Storage Elements", Biol. Cybern., 39 (1981), S. 125-127.

[56] Kohonen, T.: "Content-Addressable Memories", Berlin: Springer Verlag 1980.

[57] Hörbst, E., Nett, M., Schwärtzel, H.: " VENUS-Entwurf von VLSI-Schaltungen", Berlin: Springer Verlag 1986.

[58] Schumacher, K.: "Integrationsgerechter Entwurf analoger MOS-Schaltungen", München: Oldenbourg Verlag, 1987.

[59] Sibbert, H.: "BONSAI - Ein Programm zur Analyse und Optimierung von integrierten Schaltungen", Benutzeranleitung, Vers. 1/2, Robert Bosch GmbH, Reutlingen, 1986.

[60] Heite, Ch.: "Entwurf und Entwicklung von Methoden der Dynamikreduktion analoger Signale in CMOS-Technik zur Systemintegration", Diplomarbeit, Universität Dortmund, Lehrstuhl Bauelemente der Elekrotechnik, 1988.

[61] Schmidt, M.: "Entwicklung spezieller OTA-Strukturen für ein Filterkonzept in CMOS-Technik", Studienarbeit, Universität Dortmund, Lehrstuhl Bauelemente der Elekrotechnik, 1988.

[62] Rückert, U., Kreuzer, I., Tryba, V., Goser, K.: "Fault-Tolerance Of Associative Memories Based On Neural Networks", COMPEURO, Hamburg (1989), Tagungsband S. 1.52-1.55.

[63] Stiebler, G.: "Entwurf von CMOS-Grundschaltungen für assoziative Netzwerke", Diplomarbeit, Universität Dortmund, Lehrstuhl Bauelemente der Elekrotechnik, 1988.

[64] Hopfield, J.J.: "Neurons with graded response have collective computational properties like those of two-state neurons", Proc. Nat. Acad. Sci., Vol. 81 (1984), S. 3088-3092.

[65] Howard, R.E., u.a.: "An Associative Memory based on an Electronic Neural Network Architecture", IEEE Trans. on Elect. Dev., Vol. ED-34 (1987), S. 1553-1556.

[66] Henning, G.: "Treiber- und Empfängerschaltungen zur Signalübertragung in VLSI-Systemen", Dissertation Universität Dortmund, Bauelemente der Elektrotechnik, 1989.

[67] Tsividis, Y., Anastassiou, D.: "Swichted-Capacitor Neural Networks", Electronics Letters, Vol. 23 (1987), No. 18, S. 958-959.

[68] Goser, K.: "Großintegrationstechnik I und II", Studienkurs der Fernuniversität Hagen, Fachbereich Elektrotechnik 1989.

[69] Götzlich, J.: "Die dritte Dimension in der Mikroelektronik", Phys. Bl. 44, Nr.10 (1988), S. 391-395.

[70] Murray, A.F., Smith, A.V.W.: "Asynchronous Arithmetic for VLSI Neural Systems", Electronics Letters Vol. 23 No. 12 (1987), S. 642-643.

[71] MacGregor, R.J., Oliver, R.M.: "A General-purpose Electronic Model for Arbitrary Configurations of Neurons", J. theor. Biol. 38 (1973), S. 527-538.

[72] Coon, D.D., Perera, A.G.U.: "Semiconductor electronic concepts for neural network emulation", Int. J. Electronics, Vol. 63, No. 1 (1987), S. 61-69.

[73] Murray, A. u.a.: "Fully-Programmable Analogue VLSI Devices for the Implementation of Neural Networks", in: *Delgado-Frias, J.G., Moore, W.R. (Hrsg.)*: "VLSI for Artificial Intelligence", Boston: Kluwer Academic Publ., 1989, S. 236-244.

[74] Sivilotti, M.A., u.a.: "VLSI Architectures for Implementation of Neural Networks", in *Denker, J.S. (Hrsg.)*: "Neural Networks for Computing", AIP Conf. Proc. 151, New York: American Inst. Phys., 1986, S. 408-413.

[75] Graf, H.P., u.a.: "VLSI implementation of a neural network memory with several hundreds of neurons", in *Denker, J.S. (Hrsg.)*: "Neural Networks for Computing", AIP Conf. Proc. 151, New York: American Inst. Phys., 1986, S. 182-187.

[76] Mackie, S., u.a.: "Microelectronic Implementations Of Connectionist Neural Networks", IEEE Conference on Neural Information Processing Systems, Denver (1987).

[77] Verleysen, M., u.a.: "A New CMOS Architecture For Neural Networks", in *Delgado-Frias, J.G., Moore, W.R. (Hrsg.)*: "VLSI for Artificial Intelligence", Boston: Kluwer Academic Publ., 1989, S. 209-217.

[78] Rückert, U., Kreuzer, I., Goser, K.: "A VLSI Concept For An Associative Matrix Based On Neural Networks", COMPEURO, Hamburg (1987), Tagungsband S. 31-34.

[79] Sivilotti, M.A., u.a.: " Real-Time Visual Computations Using Analog CMOS Processing Arrays", in *P. Losleben (Hrsg.)*: "Advanced Research in VLSI", Cambridge, Mass.: MIT Press, 1987, S. 295-312.

[80] Thakoor, A.P., u.a.: "Binary Synaptic Connections Based On Memory Switching in a-Si:H", in *Denker, J.S. (Hrsg.)*: "Neural Networks for Computing", AIP Conf. Proc. 151, New York: American Inst. Phys., 1986, S. 426-431.

[81] Saito, S. u.a.: "A 1-MBit CMOS DRAM with Fast Page Mode and Static Column Mode", IEEE Jour. of Solid State Circuits, Vol. SC-20, No.5 (1985), S. 903-908.

[82] Shah, A.H.: "A 4 Bit dRAM with Trench Transitor Cell", IEEE Journal of Solid State Circuits, Vol. Sc21 (1986), S. 618-626.

[83] Burgs, A.L.J.: "Aktueller Stand der SRAM-MOS Technologie" COMPEURO 89, Hamburg (1989) Tagungsband S. 35-42.

[84] Graf, H.P., u.a.: "VLSI-Implementation of neural network memory with serveral hundrets of neurons", in: *Denker, J.S. (Hrsg.)*: "Neural Networks for Computing", AIP Conf. Proc. 151, New York: American Inst. Phys. (1986), S. 182-187.

[85] Soennecken, A.: "Entwicklung von Speichertransistoren mit dünnen Oxidschichten", Diplomarbeit, Universität Dortmund, Bauelemente der Elektrotechnik, 1988.

[86] Lenzinger, M., Snow, E.H.: "Fowler-Nordheim Tunneling into Thermally Grown SiO_2", Journal of Applied Physics, Vol. 40 No.1 (1969), S. 278-283.

[87] Longo, H.-E.: "Family of Characteristics of Floating-Gate memory Cells (EEPROM) and Degradation Due to Cycling", Siemens Forsch.-und Entwickl.-Ber.Bd. 16 (1987) Nr.5, Springer-Verlag 1987, S. 184-191.

[88] Fölster, C.: "Untersuchungen der G329-Zellen von Siemens", Interner Bericht, Fernuniversität Hagen, Bauelemente der Elektrotechnik 1983.

[89] Sommer, K., Schneider, H.-J.: "Untersuchung der Analogwert-Speichereigenschaften von Floating-Gate-MOS Transistoren", Diplomarbeit, FH Iserlohn, 1982.

[90] Beinvogel, W.: "16-MBit und 64-MBIt-Speicher im kommen", Mikroelektronik, Bd. 3, Heft 3 (1989), S. 131.

[91] Rückert, U., Goser, K.: "VLSI-Architectures for Associative Networks", ISCAS '80, Tagungsband S. 755-758.

[92] Fölster, C.: "Adaptive and Associative Systems", Teil 1-3, Interner Bericht , Fernuniversität Hagen, Bauelemente der Elektrotechnik 1983.

[93] Rückert, U., Goser, K.: "Adaptive Associative Systems For VLSI", in: *Becker, J.D., Eisele, I. (Hrsg.)*: "WOPPLOT 86-Parallel Processing: Logic, Organization, and Technology", Berlin: Springer-Verlag, 1987, S. 166-184.

[94] Goser, K., Fölster, C., Rückert, U.: "Intelligent Memories in VLSI", Information Sciences 34 (1984), S. 61-82.

[95] Sage, J.P., u.a.: "An Artificial Neural Network Integrated Circuit Based On MNOS/CCD Principles", in: *Denker, J.S. (Hrsg.)*: "Neural Networks for Computing", AIP Conf. Proc. 151, New York: American Inst. Phys., (1986). S. 381-385.

[96] Spencer, E.G.: "Programmable Bistable Switches and Resistors for Neural Networks", in: *Denker, J.S. (Hrsg.)*: "Neural Networks for Computing", AIP Conf. Proc. 151, New York: American Inst. Phys., (1986). S. 414-419.

[97] Rückert, U., Goser, K.: "VLSI-Design Of Associative Networks", in: *Delgado-Frias, J.G. , Moore, W.R. (Hrsg.).:* "VLSI for Artificial Intelligence", Boston: Kluwer Academic Publishers, 1989. S. 227-235.

[98] Elbracht, B.: "Analyse, Simulation und Entwurf eines CMOS-VLSI-Speicherkonzeptes basierend auf dem neuronalen Netzwerk nach Hopfield", Diplomarbeit, Universität Dortmund, Lehrstuhl Bauelemente der Elektrotechnik, 1987.

[99] Kleerbaum, C.: "VLSI–Entwurf eines assoziativen Speicherkonzeptes basierend auf dem neuronalen Netzwerkmodell nach Hopfield", Studienarbeit, Universität Dortmund, Lehrstuhl Bauelemente der Elektrotechnik, 1988.

[100] Oberländer, J.: "Untersuchung von integrierten assoziativen Speicherzellen", Studienarbeit, Universität Dortmund, Lehrstuhl Bauelemente der Elektrotechnik, 1989.

[101] Melis, J.: "Funktionsanalyse von integrierten assoziativen Speicher-Teststrukturen", Studienarbeit, Universität Dortmund, Lehrstuhl Bauelemente der Elektrotechnik, 1989.

[102] Reichel, H.: "Packaging of VLSI Devices", COMPEURO, Hamburg (1989), Tagungsband S. 5.63-5.67.

[103] Häbel, B.: "CMOS-Standardzellenentwurf eines digitalen assoziativen Netzwerk-Akzelerators mit dem VENUS-Entwurfssystem", Diplomarbeit, Universität Dortmund, Lehrstuhl Bauelemente der Elekrotechnik, 1988.

[104] Darianian, M.: "VLSI-Entwurf einer digitalen assoziativen Speichermatrix auf einer Siemens-VENUS-Workstation", Diplomarbeit, Universität Dortmund, Lehrstuhl Bauelemente der Elekrotechnik, 1987.

[105] Palm, G., Bonhoeffer, T.: "Parallel Processing for Associative and Neural Networks", Biol. Cybern. 51 (1984), S. 201-204.

[106] Erb, M., Palm, G.: "Lernen und Informationsspeicherung in Neuronalen Netzen", Berlin: VDE-Verlag, ITG-Fachbericht 102 (1988), S. 379-390.

[107] "SL2000 - Structured Design System", SILVAR–LISCO, Comand Ref. Manual, Vol. 1, 1986.

[108] "PRINCESS Graphics Editor", SILVAR-LISCO, Command Ref. Manual, Vol. 1, 1986.

[109] "Design Verification Language", SILVAR-LISCO, User's Guide, Vers. 2.0, 1986.

[110] Rückert, U., Goser, K.: "Ein digital/analoges Assoziativspeicherkonzept basierend auf neuronalen Strukturen", GMD-Studie Nr. 155 zum 4. E.I.S.-Workshop 1989, S. 201-210.

[111] Groß, H.-J.: "Architekturanalyse eines digital/analogen Assoziativspeicherkonzeptes", Teststrukturen", Studienarbeit, Universität Dortmund, Lehrstuhl Bauelemente der Elektrotechnik, 1989.

[112] Graf, H.P., Hubbard, W.: "VLSI Neural Network for Fast Pattern Matching", in: *Personnaz, L., Dreyfus, G. (Hrsg.)*: "Neural Networks from Models to Applications", Paris: I.D.S.E.T., 1989, S. 725-732.

[113] Personnaz, L. u.a.: "Towards a Neural Network Chip: A Performance Assessment and a simple Example", in: *Personnaz, L., Dreyfus, G. (Hrsg.)*: "Neural Networks from Models to Applications", Paris: I.D.S.E.T., 1989, S. 682-691.

[114] Knauer, K., Ramacher, U., Pandel, J.: "Digitale Realisierung eines neuronalen Netzes auf einem Chip", ZFE ME 22, Erfindungsmeldung 1988.

[115] Klar, H., Ramacher, U.: "Microelectronics for Artificial Neural Nets", Düsseldorf: VDI-Verlag, Reihe 21, Nr. 42, 1989.

Lebenslauf

29.11.1957	geboren in Soest, Westfalen
01.04.1964- *28.05.1973*	Besuch der Pauli- und Thomä-Hauptschule in Soest, Mittlere Reife
01.08.1973- *16.06.1976*	Besuch des Archigymnasiums in Soest, Allgemeine Hochschulreife
04.10.1976- *30.09.1978*	2-jährige Dienstzeit bei der Bundeswehr, Ausbildung und Ernnenung zum Reserveoffizier
01.10.1978- *12.10.1984*	Studium der Informatik mit Nebenfach Physik an der Universität Dortmund, Diplom Informatiker
01.11.1984- *31.03.1985*	wissenschaftlicher Mitarbeiter an der FernUniversität Hagen, Lehrstuhl Bauelemente der Elektrotechnik
01.04.1985- *dto.*	wissenschaftlicher Mitarbeiter an der Universität Dortmund, Lehrstuhl Bauelemente der Elektrotechnik