

II. Formalismus assoziativer Netze für VLSI - Implementierung

Um die Eigenschaften und die Verwendungsmöglichkeiten neuronaler Netzwerkmodelle beschreiben und vergleichen zu können, bedarf es einer allgemeinen, formalen Beschreibungsform. Diese Beschreibungsform sollte nach D. Rumelhart, G. Hinton und J. McClelland (RMC86) acht grundlegende Komponenten enthalten:

- eine Menge von Prozessorelementen,
- einen Aktivierungszustand,
- eine Ausgabefunktion für jedes Prozessorelement,
- eine Verbindungsbeschreibung zwischen den Elementen,
- eine Propagierungsfunktion, um Aktivitätsmuster durch das Netz zu verbreiten,
- eine Aktivierungsfunktion, die mittels der Eingabe in ein Element und des derzeitigen Zustands der Aktivierung einen neuen Aktivierungszustand berechnet,
- eine Lernregel, wodurch die Verbindungsgewichte der Elemente verändert werden, und
- eine Umgebung, in der das System arbeiten muß.

Denkt man weiterhin an die Hardware-Realisierung neuronaler Netzwerkmodelle, so sollte mit Hilfe der Beschreibungsform eine Umsetzung in spezielle VLSI-Architekturen möglich sein, denn der wesentliche Gesichtspunkt einer formalen Beschreibung im Zusammenhang mit dem Entwurf hochintegrierter Systeme ist der Einsatz als Spezifikationssprache. U. Rückert (RÜC89) definiert einen formalen Rahmen, der die Basis für modellunabhängige Simulatoren bildet, die grundsätzlich zur funktionalen Simulation von Systemarchitekturen herangezogen werden können (siehe Kapitel VII).

Das grundlegende Systemelement eines neuronalen Netzwerkes bildet bei U. Rückert, wie auch in anderen Beschreibungsformen (Rumelhart s.o.), ein einfaches Prozessorelement (Verarbeitungseinheit, Neuron). Die bei einem Prozessorelement eintreffenden Nachrichten und Signale (≥ 1000) werden auf einen Aktivitätszustand abgebildet. In Abhängigkeit von diesem Zustand berechnet das Prozessorelement einen Ausgabewert, der bei einem Neuron nach dem Prinzip der Impuls-Frequenzkodierung an die Außenwelt oder an weitere Neuronen gesendet wird. Die Mehrheit der neuronalen Netzwerkmodelle verwendet allerdings nicht die Impuls-Frequenzkodierung, da die Analyse und Simulation der Netzwerke sehr aufwendig ist. Die Modelle kodieren zeitlich gemittelte Größen, wie etwa die Impulsrate oder aber binäre Zustände und benutzen diese zur Approximation der Aktivität.

II.1 Definition des formalen Prozessorelementes

Das formale Prozessorelement nach U. RÜckert (RÜC89) ist angelehnt an die Beschreibung endlicher Automaten - wie auch bei C. Kemke (KEM84) - und wird beschrieben durch:

Definition II.1

Ein **Prozessorelement** ist ein Acht-Tupel $P = (n, X, A, Y, W, a, S, X, TH)$ mit:

- n : Anzahl der Eingabeleitungen.
- X : Menge der Eingabewerte,
- A : Menge der Aktivierungswerte,
- Y : Menge der Ausgabewerte.
- W : Menge der Gewichte.
- a : Aktivierungsfunktion
 $\alpha : X^n \times W^n \times A \longrightarrow A$
- S : Ausgabe- oder Begrenzungsfunktion
 $S : A \longrightarrow Y$,
- X : Adaptions- oder Lernfunktion
 $X : W^n \times X^n \times A \times Y \longrightarrow W$
- TH : Schwelle oder Schwellenwert

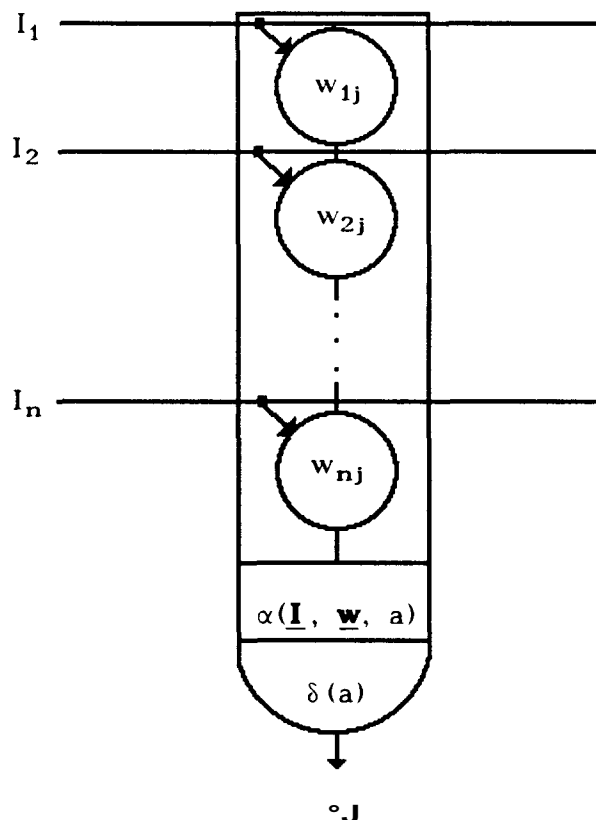


Abbildung II.1: Darstellung des j-ten Prozessorelementes.

Ein Prozessorelement wird somit charakterisiert durch die n Eingabeleitungen, auf denen die Eingabewerte $I_i \in X$ übertragen werden. Jeder Eingabeleitung $i \in \{1..n\}$ ist eineindeutig eines der n Gewichte $w_i \in W$ zugeordnet, wobei $W_j > 0$ excitatorische (erregende), $W_j < 0$ inhibitorische (hemmende) Verbindung heißt. Die Mengen X, A, Y, W umfassen häufig die reellen Zahlen \mathbb{R} oder Teilmengen von \mathbb{R} (z.B. $\mathbb{Z}, \{-1,0,1\}$ oder $\{0,1\}$), wobei die gleichzeitig auf den Eingabeleitungen übertragenen Werte zu einem n -dimensionalen Vektor \underline{I} und die Gewichte zu einem Gewichtsvektor \underline{w} zusammengefaßt werden. Der vom Prozessorelement berechnete Ausgabewert $O_j \in Y$ wird auf der einen Ausgabeleitung ausgegeben. Abbildung II.1 zeigt das Modell eines Prozessorelementes.

In der Neurophysiologie gibt es Gehirn-Modelle, bei denen das oben definierte Prozessorelement einem Neuron und die Verbindungsgewichte den Synapsen entsprechen. (PAL82, PAL88). In anderen Fällen heißen die Prozessorelemente auch Hypothesen und die Verbindungen Bedingungen (constraints) zwischen den Hypothesen. Excitatorische Verbindungen entsprechen sich gegenseitig stützenden Hypothesen, während inhibitorische Verbindungen sich gegenseitig ausschließende Hypothesen darstellen. In der Festkörperphysik existieren Modelle (Spingläser-Modelle), in denen die Prozessorelemente und die Verbindungsgewichte Atomen und Molekülen mit ihren Wechselwirkungen entsprechen (KIN85).

Die Dynamik eines Prozessorelementes wird durch die drei Funktionen a, δ und S bestimmt. Die Aktivierungsfunktion a berechnet einen neuen Aktivierungswert $a \in A$ in Abhängigkeit vom aktuellen Aktivierungswert, den aktuellen Gewichten und der anliegenden Eingabe. Die am häufigsten verwendete Aktivierungsfunktion a basiert auf der gewichteten Summe der Eingabewerte, d.h:

$$a = \sum_{i=1}^n I_i * w_i = \underline{I}^T * \underline{w} \quad (\text{II.1})$$

Die Ausgabefunktion S bestimmt in Abhängigkeit von der momentanen Aktivierung $a_j \in A$ des Prozessorelementes j den Ausgabewert O_j . Neben einfachen, linearen Ausgabefunktionen (KOH77) findet man häufig Schwellenfunktionen der Art

$$\delta(a) = \begin{cases} 1, & \text{wenn } a \geq TH, \\ 0 & \text{sonst,} \end{cases} \quad (\text{II.2})$$

oder semilineare (logistische, sigmoide) Ausgabefunktionen

$$S(a) = \frac{1}{1 + e^{-a - TH}}, \quad (\text{II.3})$$

wobei der Exponent $-a - TH$ durch einen Parameter T skaliert sein kann. Abbildung II.2 zeigt die beiden Ausgabefunktionen. Die sigmoide Funktion ist im Gegensatz zur Schwellenfunktion differenzierbar und streng monoton wachsend. Sie approximiert die Schwellenfunktion für sehr kleine T .

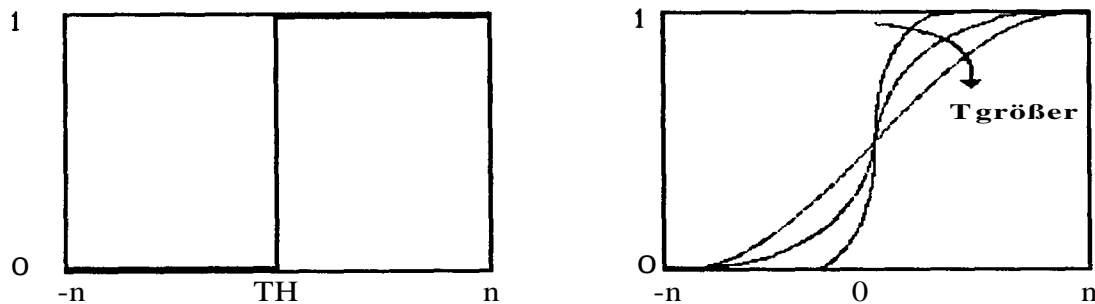


Abbildung II.2: Schwellen- und Sigmoidausgabefunktion

Grundlegend für die sich ändernde Arbeitsweise eines Prozessorelements und damit des gesamten assoziativen Netzes ist die Lern- oder Adaptionsfunktion. Sie ist eine Vorschrift zur zeitlichen Änderung der Verbindungsgewichte $W(t) = W(t-1) + \Delta W$, wodurch sich eine *zeitvariante* Arbeitsweise ergibt.

Die Lernverfahren lassen sich dabei in *unüberwachte* und *überwachte* Verfahren unterteilen. Die *unüberwachten* Lernverfahren benutzen statistische Eigenschaften der Eingabemuster (Traningsmenge), um diese zu klassifizieren. Zu den *unüberwachten* Verfahren gehört die weit verbreitete Hebb-Regel. Sie basiert auf der Hebb'schen Hypothese, daß die Verbindung W_j verstärkt wird, wenn die Ausgabeleitung des Prozessorelements j und die Eingabeleitung X_j aktiviert sind:

$$\Delta w_{ij} = T_j * I_i * O_j, \quad (II.4)$$

wobei T_j die konstante Lernrate ist. Die Hebb-Regel wird beim einfachen Assoziativspeicher in Kapitel IV verwendet.

Zu den *überwachten* Lernverfahren zählt die *Widrow-Hoff-oder Delta-Regel*:

$$\Delta w_{ij} = \eta * I_i * \delta_j, \quad \delta_j = (O_j^z - O_j). \quad (II.5)$$

O_j^z ist die gewünschte (Ziel) und O_j die vom Netz berechnete Ausgabe, womit sich δ_j als Abweichung der gewünschten zur tatsächlich berechneten Ausgabe ergibt. Die Delta-Regel sorgt also dafür, daß die be-

rechnete Ausgabe mit der vorgegebenen Ausgabe übereinstimmt (es gilt: $\delta_j=0 \Rightarrow \Delta w_{ij}=0 \Rightarrow W$ konstant). Überwachte Lernverfahren werden im Kapitel V untersucht, wobei im Hinblick auf eine schaltungstechnische Realisierung zwischen *lokalen Regeln* (Delta-Regel), bei denen die Größen lokal an jedem Prozessorelement zur Verfügung stehen, und *globalen Regeln* (Pseudoinverse) unterschieden wird.

Die Leistungsfähigkeit eines neuronalen Netzes entsteht nun durch das Zusammenspiel von sehr vielen (≥ 1000) Prozessorelementen. Das Zusammenspiel wird geregelt durch die Dynamikfunktionen jedes einzelnen Prozessorelementes und durch die Netzwerkstruktur, die im folgenden definiert wird.

x_i	y_j	Δw_{ij}
0	0	} $\eta \cdot \delta \cdot I_i$
0	1	
1	0	
1	1	

a)

x_i	y_j	Δw_{ij}
0	0	}
0	1	
1	0	
1	1	

b)

x_i	y_j	Δw_{ij}
0	0	+
0	1	—
1	0	—
1	1	+

c)

Abbildung II.3: a) Delta-Regel für binäre Eingabevektoren
 b) Vereinfachte Hebb-Regel
 c) Hebb-Regel für Hopfield-Netze

II.2 Definition des assoziativen Netzwerkes

Definition II.2

Ein *assoziatives Netzwerk* ist ein Quintupel $AN = (\mathfrak{B}, (r, u, c, \psi)$ mit einer endlichen Menge \mathfrak{B} von m Prozessorelementen, einer endlichen Menge \mathfrak{E} von n externen Eingaben sowie drei Abbildungen $D: \mathfrak{B} \times \mathfrak{B} \rightarrow \{0,1\}$, $C: \mathfrak{E} \times \mathfrak{B} \rightarrow \{0,1\}$ und $\psi: \mathfrak{B} \rightarrow \{0,1\}$.

Für alle $P_i, P_j \in \mathfrak{B}$ bestimmt $v(P_i, P_j)$, ob eine Verbindung von P_i nach P_j besteht. Die Abbildung $e(e_j, P_j)$ legt für alle $e_j \in \mathfrak{E}$ fest, ob das Prozessorelement $P_j \in \mathfrak{B}$ mit der externen Signalquelle e_j verbunden ist, und die Abbildung $\psi(P_j)$ bestimmt, ob die Ausgabe von P_j eine externe Ausgabe an die Umwelt ist.

Die Prozessorelemente und die Netzwerktopologie charakterisieren somit ein assoziatives Netzwerk. Die Dynamikgleichungen der Prozessorelemente bestimmen das zeitliche Verhalten des Netzwerkes, das entweder bezüglich einer zeitdiskreten oder einer kontinuierlichen Zeit-

skala arbeiten kann. Im kontinuierlichen Fall wird die Dynamik des Netzes durch gekoppelte Differentialgleichungen beschrieben, die der physikalischen Realität der Nervenzellen näher kommen, aber analytisch nur schwer handhabbar sind. Der zeitdiskrete Fall ist immer dann zwingend, wenn man die vielen Prozesselemente auf einem Rechner simulieren möchte. Die in dieser Arbeit verwendeten Modelle beziehen sich daher auf eine diskrete Zeitskala, so daß im allgemeinen $t \in \mathbb{N}$ gilt. Ändern alle Prozesselemente $P_j \in \mathfrak{B}$ gleichzeitig ihren Aktivitätszustand, so spricht man von einer *synchronen*, andernfalls von einer *asynchronen* Arbeitsweise des Netzes. Ein Netzwerk arbeitet *seriell*, wenn nur ein Prozesselement seinen Aktivitätszustand zu einem Zeitpunkt t neu berechnet; falls mehr als ein Prozesselement seinen Aktivitätszustand zum Zeitpunkt t neu ermittelt, so arbeitet das Netzwerk *parallel*. Werden die Prozesselemente, die ihren Aktivitätszustand ändern, zufällig ausgewählt, so spricht man von einer *nicht-deterministischen*, andernfalls von einer *deterministischen* Update-Strategie. Nichtdeterministisch arbeitende Netze wurden von J.J. Hopfield (HOP82) eingeführt.

Die drei Abbildungen D , c , ψ bestimmen die Netzwerktopologie, die durch Matrizen dargestellt wird. Bei theoretischen Untersuchungen steht häufig die Topologie der Netze im Vordergrund, weshalb Definition II.2 als gewichteter Graph $G(V,E)$ mit der Knotenmenge $V = \mathfrak{B} \cup \mathfrak{C}$ und der Kantenmenge $E = \{w_{ij} | i = 1..n, j = 1..m\}$ interpretiert wird. Die Abbildungen erlauben die Spezifikation von beliebigen Verbindungsstrukturen, z. B. Mehrebenen- oder Rückkopplungsstrukturen. Das assoziative Netz kommuniziert mit der Umwelt über externe Signalquellen. In der Literatur findet sich häufig eine Partitionierung des Netzes nach dieser Kommunikationsstruktur.

Definition II.3

Ein Prozesselement $P \in \mathfrak{C}$ eines assoziativen Netzwerkes $AN = (\mathfrak{B}, \mathfrak{C}, 0, c, \psi)$ mit $|\mathfrak{C}| = n$ heißt

- **Eingabelement** (input unit) $:\Leftrightarrow \exists i \in \{1..n\}$ mit $c(e_i, P) = 1$,
- **Verstecktes Element** (hidden unit) $:\Leftrightarrow \forall i \in \{1..n\}$ gilt $c(e_i, P) = 0$ und $\psi(P) = 0$,
- **Ausgabelement** $:\Leftrightarrow \psi(P) = 1$.

Das Netz arbeitet im '*spreading activation*'-Arbeitsmodus, wenn in dem nach Definition II.3 partitionierten Netz die Netzwerkeingabe von den Eingabelementen über die versteckten Elemente zu den Ausgabelementen propagiert wird, von denen dann das Ergebnis abgelesen werden kann. Die Netzwerktopologie darf dabei keine Rückkopplungsschleifen enthalten.

Ein Netzwerk arbeitet im '*relaxation*' (Fixpunkt)-Modus, wenn von einem Anfangsaktivitätszustand A_0 (bestehend aus den Aktivitätswerten der m Prozessorelemente) durch Rückkopplung ein stabiler Aktivitätszustand A_f erreicht wird. Ein Spezialfall des Relaxations-Modus ist das 'simulated annealing' (gleichzeitiges Ausglühen), bei dem die sigmoide Ausgabefunktion durch langsame Abnahme des Faktors T (Temperatur) der Schwellenfunktion angenähert wird, wodurch ein stabiler Zustand im globalen Minimum erreicht wird.

Ein weiteres Merkmal das assoziative Netze in der Literatur charakterisiert ist die Art der Wissensrepräsentation. Für eine Menge E von abstrakten Eigenschaften (Fähigkeiten, Dingen) spricht man von *lokaler Repräsentation* (**local representation**), wenn jedem Prozessorelement $P \in 25$ genau eine Eigenschaft $e \in E$ zugeordnet wird. Netzwerke, die ihre Information lokal repräsentieren, sind einfach zu verstehen und zu entwerfen, da die Topologie und Gewichtung des Netzes einem genauen Abbild der Struktur des Wissens über die Eigenschaften und deren Zusammenhänge entspricht. Man spricht deshalb auch von einer 'Eins-zu-Eins-Korrespondenz' (one-unit-one-concept) zwischen dem Wissen und den Prozessorelementen.

Wird dagegen jede Eigenschaft $e \in E$ durch ein Aktivitätsmuster, verteilt über alle Prozessorelemente, dargestellt, und ist jedes Prozessorelement an der Repräsentation von mehreren, verschiedenen Eigenschaften beteiligt, so spricht man von einer *verteilten Repräsentation* (distributed representation) des Wissens. Die verteilte Wissensrepräsentation der konnektionistischen Modelle ist vollkommen anders als die Art der Wissensrepräsentation von herkömmlichen KI-Systemen. Ein Vorteil der verteilten Wissensrepräsentation liegt in der *Fehlertoleranz* der Modelle, denn fällt z. B. ein Verbindungselement aus, so können die Muster vollständig oder zum größten Teil noch ausgelesen werden, da viele Verbindungselemente an der Speicherung beteiligt sind (siehe Kapitel IV). Daneben entspricht die verteilte Wissensrepräsentation eher der Wissensrepräsentation im menschlichen Gehirn als die lokale Wissensrepräsentation.

III. Zuverlässigkeit und Fehler von Bauelementen

Die Integrationstechnik benötigt fehlertolerante und fehlerkorrigierende Speicherarchitekturen. Konventionelle Speicher funktionieren nur dann einwandfrei, wenn alle Funktionseinheiten korrekt arbeiten. Zusätzliche Speicher- und Logik-Elemente sind erforderlich, um eine fehlerhafte Einheit zu rekonfigurieren, was einen Overhead von bis zu 20% bedeuten kann. Ein Ausfall einer Funktionseinheit eines Chips kann heute ganze Produktionsstraßen stilllegen und damit sehr hohe Kosten verursachen.

Zur Vermeidung bzw. zum Erkennen von Fehlern werden heute auf allen Ebenen des Hardware-Entwurfs diverse Methoden getroffen (modularer Aufbau, **Built-In-Test**, Scan Path Methode, usw.). Mit weiterer Miniaturisierung nimmt die Komplexität der Schaltwerke noch weiter zu, so daß Funktionseinheiten nur ausschnittsweise getestet werden können und daher möglicherweise fehlerbehaftet sind.

Im folgenden werden die für diese Arbeit notwendigen Kenngrößen, nämlich die Zuverlässigkeit von Bauelementen und die Arten der zu untersuchenden Fehler, beschrieben. Weitere Ausführungen finden sich bei K. Goser (GOS84, GOS87), A. Deixler und E. Metschi (DME74) sowie bei W. Görke (GÖR74).

III.1 Das Auftreten von fehlerhaften Elementen

Das Auftreten von fehlerhaften Elementen im Fertigungslos läßt sich bei der Massenproduktion von Bauelementen praktisch nicht vermeiden. Zudem verlieren die Bauelemente im Laufe ihrer Lebensdauer ihre zugesicherten Eigenschaften.

Definition III.1

Die **Zuverlässigkeit** ist ein Maß dafür, wie ein Bauelement im Laufe seines Lebens die zugesicherten Eigenschaften beibehält.

Um die Zuverlässigkeit von Bauelementen richtig einschätzen zu können, sind Kenntnisse über Zuverlässigkeits-Kenngrößen erforderlich.

Definition III.2

Es sei $N(t)$ die Anzahl der funktionsfähigen Bauelemente zum diskreten Zeitpunkt t und N_0 die Anzahl der funktionsfähigen Einheiten zum Zeitpunkt t_0 (Beginn).

Die **Bestandsfunktion** $R: \mathbb{N} \rightarrow \mathbb{R}$ wird definiert durch:

$$R(t) = \frac{N(t)}{N_0}$$

und die **Ausfallfunktion** $F: \mathbb{N} \rightarrow \mathbb{R}$ durch

$$F(t) = \frac{(N_0 - N(t))}{N_0} = \frac{AN}{N_0}$$

$X = -AN / (N_0 * \Delta t)$ heißt **Ausfallrate** und gibt die Anzahl der Ausfälle AN im Zeitraum Δt an. Die Maßeinheit für die Ausfallrate ist h^{-1} oder $\text{Fit} = 10^{-9} h^{-1}$. $1/\lambda = \text{MTBF}$ (mean time between failures) ist der mittlere Abstand zwischen zwei Ausfällen.

Der Wert X ist in der Regel nicht konstant, sondern ändert sich mit der Lebenszeit der Bauelemente. Trägt man X über der Zeit t auf, so erhält man die sogenannte **Badewannenkurve**.

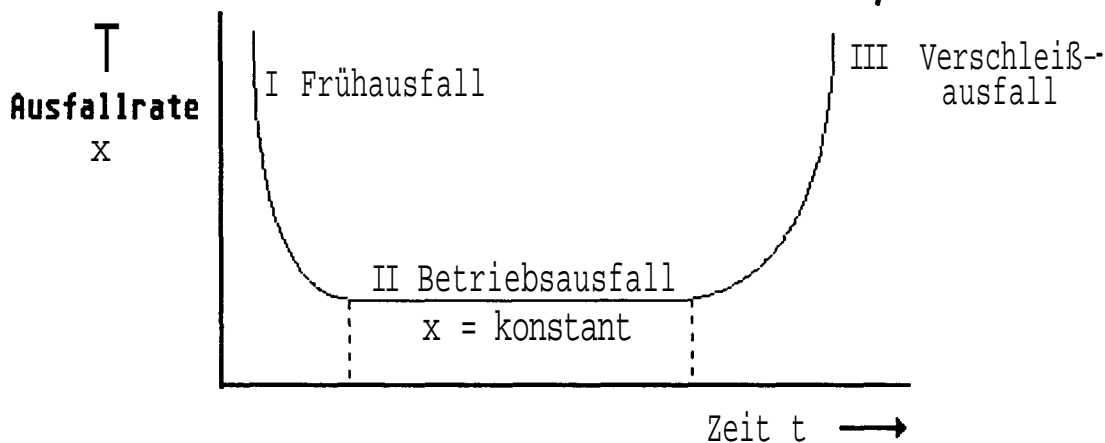


Abbildung III.1: Badewannenkurve

- Der Bereich I ist gekennzeichnet durch eine mit der Zeit abnehmende Ausfallrate X . Zu erklären ist diese Abnahme damit, daß Bauelemente, die mit herstellungsbedingten "Kinderkrankheiten" behaftet waren, bereits am Anfang ihres Einsatzes ausfallen. Um diesen Abschnitt bei wichtigen Anwendungen auszuschalten, können Bauelemente künstlich 'vorgealtert' werden, denn die Phase mit relativ hohen Ausfallraten wird in die Zeit vor dem Einbau in ein Gerät verlegt. Die am häufigsten verwendete beschleunigte Alterungs-

Methode ist die Lagerung des Bauelements bei einer erhöhten Umgebungstemperatur (burn-in). Nach Arrhenius gilt:

$X = K_A e^{-E_A / kT}$, wobei E_A die Aktivierungsenergie ist.

- Im Bereich II ist die Ausfallrate X annähernd konstant. In diesem Zeitraum fallen Bauelemente nach statistisch bedingten Ursachen aus. Dieser Abschnitt umfaßt in der Regel die normale Betriebsdauer.
- Die Ausfallrate X steigt im Bereich III wieder an. Dieser Anstieg ist bedingt durch den verstärkten Ausfall wegen "Alterserscheinungen". Heutzutage jedoch sind Halbleiterbauelemente bereits so zuverlässig, daß bei normaler Anwendung dieser Bereich nicht mehr erreicht wird.

III.2 Fehlerarten

Die verschiedenen Fehler, die in der Produktion oder während des Betriebs einer Funktionseinheit entstehen, werden in vier Kategorien eingeordnet:

- a) Katastrophenfehler (*catastrophic failures*),
- b) harte Fehler (*hard errors*),
- c) weiche Fehler (*soft errors*) und
- d) Parametervariation.

Zur Phänomengattung der Katastrophenfehler gehören die Ausfallereignisse "Kurzschluß" und "Unterbrechung" von Bauelementen. Fehler dieser Art verursachen in jedem Falle den Ausfall des nichtredundanten Bereiches einer Schaltung, in dem die Bauelemente eingesetzt werden. Derartige Fehler werden in dieser Arbeit nicht weiter betrachtet.

Defekte Speicherzellen aufgrund physikalischer Fehler - z.B. Kontaktfehler - zählen zu der Klasse der harten Fehler (*permanent errors*, MOO88). Das Bauelement liefert keinen oder einen konstanten Beitrag. Harte Fehler stehen im Mittelpunkt dieser Untersuchung.

Weiche Fehler (*transient errors*) treten nur sporadisch auf, z.B. durch α -Strahlung, Temperaturschwankungen, mechanische Beanspruchung oder Radio-Interferenzen. Die zeitliche Zuverlässigkeit (*temporal redundancy*) der Bauelemente erlaubt die Behebung dieses Fehlertyps, in dem die Berechnung erneut durchgeführt wird. Die weichen Fehler sind relativ selten und meistens in ihrer Ursache / Wirkung nicht nachzuweisen, weshalb sie nicht weiter untersucht werden.

Bei der Fertigstellung von Bauelementen kommt es aufgrund von herstellungsbedingten Unzulänglichkeiten (Maskenjustierung, **Unterdiffusion**, Unterätzen, Lichtbrechung usw.) zu einer Streuung von Parametern (z.B. der **Schwellenspannung**). Schwankungsbreiten von 10% und mehr über einen Wafer sind durchaus normal; die Schwankungen von Parametern über einen Chip oder gar über benachbarte Schaltungselemente sind wesentlich kleiner. Abbildung III.2 zeigt die Verteilung der Schwellenspannung (V_T) über einen Wafer. Das Histogramm folgt nicht unbedingt einer Normalverteilung. Es können ebenfalls sogenannte Ausreißer, also Werte weit weg vom Mittelwert, auftreten. In den meisten praktischen Fällen elektronischer Schaltungen stellt die Exponentialverteilung eine ausreichende Annäherung an das Verhalten der Bauelemente dar. (GOR74).

In den nachfolgenden Kapiteln wird die Fähigkeit von ausgewählten Repräsentanten von Assoziativspeichern bezüglich der harten Fehler und der Parametervariation untersucht.

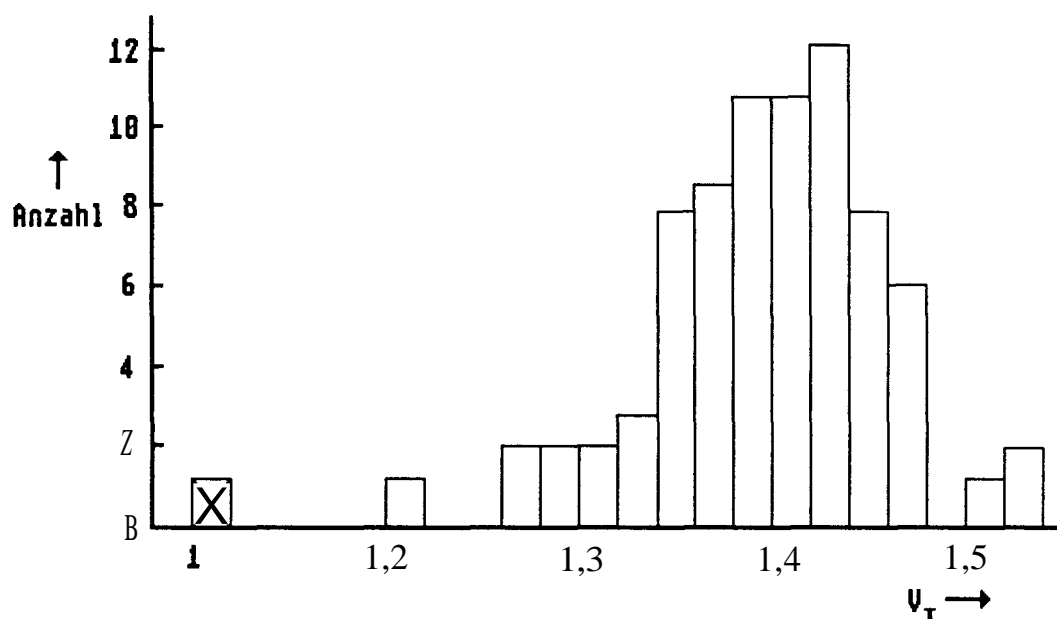


Abbildung III.2: Verteilung der Schwellenspannung (V_T) über einen Wafer.

IV. Fehlertoleranz des einfachen Assoziativspeichers

IV.1 Einführung

In diesem Kapitel werden Gleichungen abgeleitet, mit deren Hilfe sich die Fehlertoleranz des einfachen Assoziativspeichers abschätzen läßt. Sie erlauben es, unter Vorgabe gewisser Fehler, die in der Chip-Herstellung auftreten können, Speicherkapazitätswerte festzulegen, hauptsächlich für sehr große Speicher, die im Augenblick noch nicht simulierbar sind. Fehler können dabei in der Eingabesowie bei den Verbindungselementen auftreten. Der Einfluß der Parametervariation von Verbindungselementen auf die Speicherkapazität wird nach der Darstellung von Simulationsergebnissen für kleinere Netze diskutiert. Den Abschluß des Kapitels bildet die Simulation eines Assoziativspeichers, der ASCII-Muster speichert.

Der einfache Assoziativspeicher wurde schon von verschiedenen Wissenschaftlern untersucht, unter anderem von D. J. Willshaw (WIL70), von T. Kohonen (KOH72, KOH77), von G. Palm (PAL80, PAL81), von A. Lansner und Ö. Ekeberg (LAE 85) sowie von J.J. McClland (RMC86). Als Einführung in die assoziative Speicherung wird ein Vergleich mit der aufzählenden Speicherung von Nachrichten angestellt, danach folgt die Definition des einfachen Assoziativspeichers und die hier verwendete Lernregel.

IV.2 Assoziativspeicher kontra aufzählender Speicher

Um Information mathematisch zu beschreiben, wird sie als Wert einer Kodierung von Nachrichten, d.h. der Spezifikation einer Abbildung $I: N \rightarrow \mathbb{R}$ aufgefaßt. Dabei wird eine Nachricht n aus einer Menge von Nachrichten N einer Zahl $I(n) \in \mathbb{R}$ zugeordnet. Entscheidend bei der Konstruktion solcher Abbildungen ist die **Homomorphie-Eigenschaft** der Abbildung bezüglich der Addition, d.h. die Information der Nachricht $(n + n')$ beträgt die Summe der Informationen beider Nachrichten.

$$I(n + n') = I(n) + I(n').$$

Wenn die Nachricht als Sequenz (s_1, \dots, s_n) definiert ist, wobei die s_j aus einem festen, endlichen Alphabet A stammen, so läßt sich die Menge von Nachrichten ausdrücken als:

$$N(A) := \{(s_1, \dots, s_n) \mid n \in \mathbb{N}, s_1, \dots, s_n \in A\},$$

Es ist klar, daß sich mit nur zwei Elementen, z.B. $A = \{0, 1\}$, jede Nachricht kodieren läßt. Die Addition zweier unabhängiger Nachrichten $n = (s_1, \dots, s_n)$ und $n' = (s'_1, \dots, s'_k)$ wird dann einfach definiert durch:

$$(s_1, \dots, s_n) + (s'_1, \dots, s'_k) = (s_1, \dots, s_n, s'_1, \dots, s'_k)$$

Der Aufbau der Menge $N(A)$ korrespondiert mit der herkömmlichen Darstellung von Nachrichten. Die Struktur der Menge $N(A)$ wird mit *aufzählender Struktur* bezeichnet.

Eine andere Möglichkeit, Strukturen zu beschreiben, besteht in der Abbildung von Nachrichten auf Nachrichten. Eine korrekte Antwort auf eine Reihe von Fragen enthält auch Information, d.h., Elemente aus einer Menge F von 'Fragen' werden auf Elemente einer Menge A von 'Antworten' abgebildet. Man erhält eine Funktion $n: F \rightarrow A$, die Information beinhaltet (siehe Abbildung IV.1). Die Menge der Nachrichten läßt sich folgendermaßen definieren: Eine Nachricht ist eine Funktion $n: F \rightarrow A$, wobei A eine feste Menge von Antworten und F eine beliebige endliche Teilmenge aus einer Menge P von 'möglichen Fragen' ist.

$$N(P, A) := \{n: F \rightarrow A, F \subseteq P, F \text{ endlich}\}.$$

Es werden zwei Elemente in der endlichen Menge A ausgezeichnet, z.B. 'Ja' und 'Nein'. Die Kombination von zwei unabhängigen Nachrichten $n: F \rightarrow A$ und $n': F' \rightarrow A$ wird in der Menge $N(P, A)$ für $F \cap F' = \emptyset$ und $n + n': F \cup F' \rightarrow A$ definiert durch:

$$(n + n')(f) := \begin{cases} n(f), & \text{falls } f \in F \\ n'(f), & \text{falls } f \in F'. \end{cases}$$

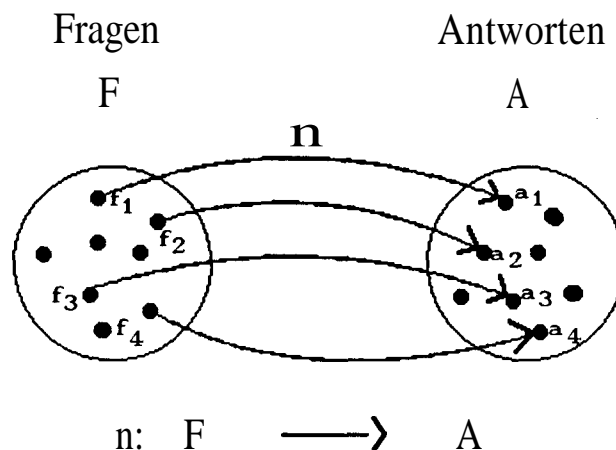


Abbildung IV.1: Abbildung von Fragen auf Antworten (PAL80).

Palm definiert nun einen Speicher als Maschine mit 2 Zuständen, dem 'Einlese'- und dem 'Auslese'- Zustand. Im 'Einlese'- Zustand werden der Maschine Nachrichten n_1, \dots, n_z , $z \in \mathbb{N}$ eingegeben. Ist der 'Auslese'-Zustand aktiv, dann können die vorher eingespeicherten Nachrichten n_1, \dots, n_z reproduziert werden.

Hat die Menge von Nachrichten, auf denen der so definierte Speicher arbeitet, die Form einer aufzählenden Struktur, so wird der Speicher als *Listenspeicher* bezeichnet. Entspricht die Menge der Nachrichten einer Abbildung, so heißt der darauf arbeitende Speicher *Assoziativspeicher*. Ein Videorecorder ist z.B ein Listenspeicher.

Assoziativspeicher sind in ihrer Arbeitsweise schwerer zu verstehen. Anfangs sind keine Nachrichten gespeichert, d.h. der Speicher ist leer. Im 'Einlese'- Modus werden dann Fragen mit den entsprechenden Antworten gespeichert. Im 'Auslese'- Modus kann anschließend durch Eingabe einer Frage die korrespondierende Antwort berechnet werden. Bei der Implementierung des Assoziativspeichers ist zu beachten, daß man diesen nicht als Listenspeicher konstruiert. Dies würde geschehen, falls eine Liste aller Fragen mit den entsprechenden Antworten eingespeichert wird. Es ist einfach einzusehen, daß Listenspeicher in Assoziativspeicher und Assoziativspeicher in Listenspeicher umgewandelt werden können (PAL80).

Im folgenden werden zwei Möglichkeiten aufgezeigt, Assoziativspeicher zu konstruieren. Zum einen kann die Abbildung n als lineare Abbildung $\mu: \mathbb{R}^n \rightarrow \mathbb{R}^m$ implementiert werden. Diese Konstruktion wird im Kapitel V weiter verfolgt. Die andere, approximative Möglichkeit wird nun dargestellt.

IV.3 Definition des einfachen Assoziativspeichers

Definition IV.1

Ein **einfacher Assoziativspeicher** ist ein assoziatives Netzwerk $EA = (\mathfrak{B}, \mathcal{C}, v, c, \psi)$ mit folgenden Eigenschaften:

- $\mathfrak{B} = \{P_j \mid P_j = (n, X, A, Y, W, \alpha_j, S_j, \lambda_j, TH_j), j = 1..m\}$ mit
 $X = Y = W = \{0,1\};$
 $A = [0..n] \in \mathbb{R};$
 $n, m \in \mathbb{N};$

$$\alpha_j(\underline{I}, \underline{w}_j) = \underline{I}^T * \underline{w}_j = \sum_{i=0}^n I_i * w_{ij},$$

wobei $I_i, w_{jj} \in \{0,1\};$

$$S_j(\mathbf{a}) = \begin{cases} 1, & \text{falls } a \geq TH_j \\ 0, & \text{sonst} \end{cases}$$

$$a = \alpha_j(\mathbf{I}, \mathbf{w}_j) \in A;$$

$TH_j \in A$ ist der Schwellenwert (s. II.1);

$$\lambda_j(\mathbf{I}^t, \mathbf{w}_j, \mathbf{O}^t) = \mathbf{w}_j, \text{ mit}$$

$$w_{ij}^t = \begin{cases} 1, & \text{falls } (\mathbf{O}_j^t - A I_i^t) > \sum_{j=1}^m w_{ij}^t \cdot 1 \\ 0, & \text{sonst} \end{cases} \quad I_i^t, \quad \mathbf{O}_j^t, \quad w_{ij}^t \in \{0,1\}$$

wobei $i = 1..n$ und $t = 1..z$, z : Anzahl der Muster.

- $\mathcal{E} = \{e_i \mid i = 1..n\}$.
- $v(P_i, P_j) = 0 \quad \forall i, j \in \{1..n\}, P_i, P_j \in \mathcal{E}$.
- $c(e_i, P_j) = 1, \quad \forall i \in \{1..n\}, j \in \{1..m\}, P_j \in \mathcal{E}, e_i \in \mathcal{E}$.
- $\psi(P_j) = 1, \quad \forall j \in \{1..m\}, P_j \in \mathcal{E}$.

Der einfache Assoziativspeicher besteht aus m Prozessorelementen, die alle denselben binären Eingabevektor \mathbf{I} aus den externen Eingabequellen e_i empfangen. Jedes Prozessorelement p_j sendet als Ausgabe einen Wert aus $\{0,1\}$ an die externe Ausgabe. Der Aktivierungswert eines Prozessorelementes ergibt sich durch Summierung der gewichteten Eingabe und liegt bei korrekter Funktionsweise in der Menge $\{1..n\}$. Falls diese Summe größer wird als der zugehörige Schwellenwert TH , so wird eine 1, andernfalls eine 0 auf die Ausgabeleitung ausgegeben. Gelernt wird mit einer vereinfachten Form der Hebb-Regel. Die Prozessorelemente arbeiten zeitdiskret und parallel deterministisch, d.h. sie 'sehen' eine globale Uhr und bestimmen gemeinsam ihren neuen Zustand durch Anwendung der Dynamik-Funktionen, wenn die globale Uhr einen neuen Taktzyklus beginnt. Abbildung IV.2 zeigt ein Modell des einfachen Assoziativspeichers.

Zur weiteren Ableitung von Eigenschaften werden die Ausgabeleitungen der m Prozessoren zum Ausgabevektor \mathbf{O} zusammengefaßt. Es sind dann:

- n die Dimension des Eingabevektors \mathbf{I} [$n = \dim(\mathbf{I})$] und
- m die Dimension des Ausgabevektors \mathbf{O} [$m = \dim(\mathbf{O})$],

w_{ij} ist dann der Wert des i -ten Verbindungsgewichtes vom Prozessorelement j . Der Ausgabevektor entspricht dem vom Netz berechneten bzw. gespeicherten Muster. Mit I_i^t wird der Wert des t -ten Eingabemusters an der Position $i = 1..n$ bezeichnet, wobei $t = 1..z$ und z die Anzahl der im Netz gespeicherten bzw. einzuspeichernden Muster ist. Entsprechend bezeichnet O_j^t den Wert des t -ten Ausgabemusters an der Position $j = 1..m$. Weiterhin seien

l = Anzahl der Einsen im Eingabevektor \underline{I} und
 k = Anzahl der Einsen im Ausgabevektor O.

Wie in der Einleitung angedeutet, werden die Muster nicht unter Angabe der Adresse, unter der sie gespeichert wurden - wie im herkömmlichen Speicher - , sondern durch Angabe des zu dem Ausgabemuster korrespondierenden Eingabemuster ausgelesen. Das Eingabemuster \underline{I}^t kann dabei unvollständig, d.h. $\mathbf{I}^t \subset \underline{I}^t$ oder verrauscht, d.h. $\mathbf{I}^t \approx \underline{I}^t$ sein (Musterabbildung und Mustervervollständigung). Bei unvollständigem oder verrauschtem Eingabemuster wird versucht, mit Hilfe einer Schwellenwert-Auswahl-Logik das Ausgabemuster auszulesen.

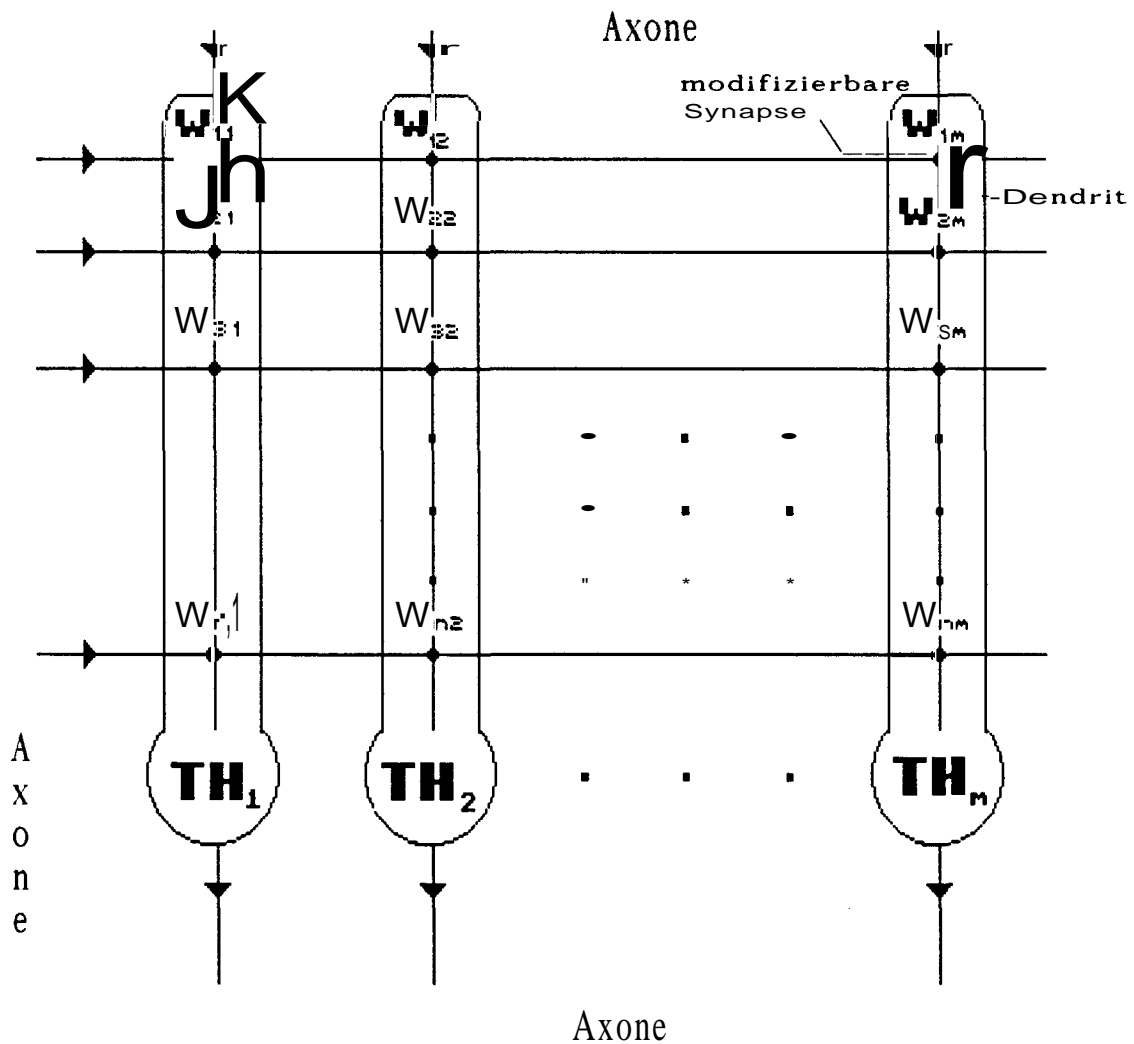


Abbildung IV.2: Der einfache Assoziativspeicher dargestellt als assoziatives Netz. Die Elemente in der vertikalen entsprechen den Prozessorelementen P_j mit dem Schwellenwert TH_j . Faßt man den Assoziativspeicher als Modell des menschlichen Gehirns auf, so entsprechen die dünnen Linien in der horizontalen den Axonen und die Leitungen in der vertikalen den Dendriten. Die Verbindungen fungieren als Synapsen. Sie sind variabel (PAL88).

IV.4 Das Einspeichern von Mustern im Assoziativspeicher (Lernen)

Um Muster in einen nach Definition IV.1 definierten Assoziativspeicher einzuspeichern, wird eine vereinfachte Form der Hebb'schen Lernregel verwendet. Dabei wird die Ausgabeleitung des j -ten Prozessorelements mit dem Wert des Ausgabemusters O_j^t , $t = 1..z$ belegt. Das Eingabemuster I_i^t wird auf die Eingabeleitung eines jeden Prozessorelementes geschrieben. Sind nun Ausgabeleitung O_j und Eingabeleitung I_i aktiv, so wird das Verbindungsgewicht w_{ij} auf 1 gesetzt, andernfalls wird w_{ij} nicht verändert. Folglich muß es nur ein Korrespondenzpaar (I_i^t, O_j) unter allen Paaren geben, damit $w_{ij} = 1$ ist. Existieren weitere Korrespondenzen (I_i^s, O_j^s) mit $s \neq t$, so bleibt $w_{ij} = 1$ erhalten.

Abbildung IV.3-6 illustriert die Speicherung und den Abruf von Mustern. Zur Speicherung eines Musters (3) wird die Ausgabebitfolge an die horizontalen und die Eingabebitfolge an die vertikalen Leitungen angelegt. Dabei werden die Leitungen aktiviert (dicke Punkte), an denen die beiden Bitfolgen korrespondieren, d.h. beide gleich 1 sind. In (4) wird ein weiteres

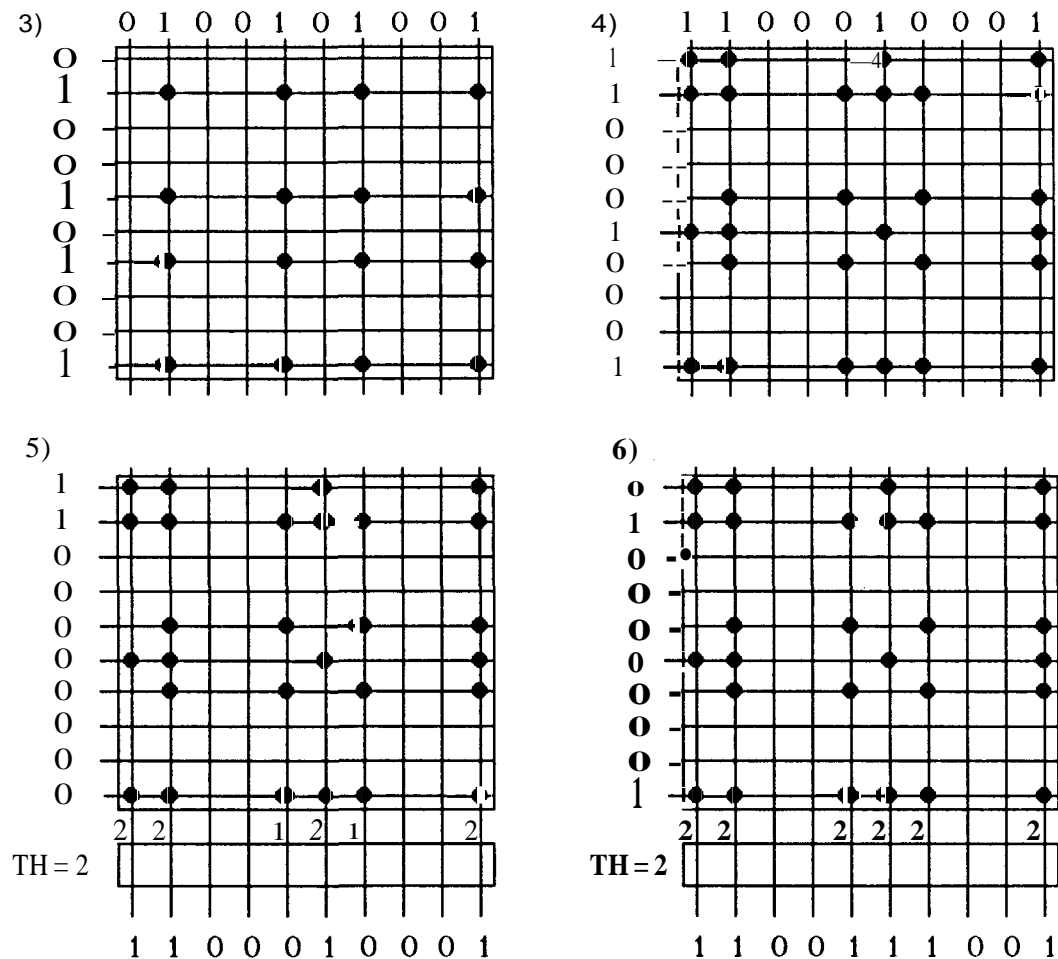


Abbildung IV.3 - 6 : Speicherung und Abruf von Mustern in einem Assoziativspeicher. Vertikal sind die Gewichtsvektoren der m Prozessorelemente dargestellt.

Muster eingespeichert. Um ein Muster wieder auszulesen, genügt es, einen Teil des Musters anzulegen (5). Das Teilmuster erzeugt über die aktivierten **Verknüpfungen** ein Ausgangssignal. Die Ausgabefunktion ergänzt mit Hilfe des Schwellenwertes TH das korrekte Ausgabemuster. Die Ausgabe kann auch eine Überlagerung der gespeicherten Muster darstellen (6), wenn eine Kombination aus beiden Mustern als Eingabe vorliegt. Faßt man die Verbindungsgewichtsvektoren der m Prozessorelemente als Spaltenvektoren einer Matrix W auf, so läßt sich die Wirkung der Lernphase auf die Matrix ebenfalls **ausdrücken** durch:

$$w_{ij} = \bigvee_{t=1}^z I_i^t A O_j^t, \text{ wobei } \bigvee \text{ die Konjunktion von Bits ist, d.h. } 0 \vee 0 = 0, \\ 0 \vee 1 = 1 \vee 0 = 1 \vee 1 = 1.$$

IV.5 Gesamt- und Bitfehlerwahrscheinlichkeit beim Assoziieren eines Musters

Um die Fehlertoleranz des Assoziativspeichers zu untersuchen, benötigt man charakteristische Leistungsgrößen, mit deren Hilfe sich Eigenschaften des Speichers beschreiben lassen. Ein Kriterium für die Leistungsfähigkeit des Speichers ist die Anzahl der Muster z, die in ihm gespeichert werden können. Ein weiteres Kriterium ist die Wahrscheinlichkeit für einen Bitfehler im Ausgabemuster bei der Assoziierung und ein drittes ist die Informationsspeicherkapazität (s.u.), die eine Kombination der ersten beiden Kriterien beinhaltet.

Die fehlerhafte Assoziierung von Eingabe- mit den Ausgabebitfolgen wächst, wenn versucht wird, sehr viele Muster in den Verbindungsgewichten des Assoziativspeichers abzulegen. Es kommt zu Überlappungen von Mustern, da die Gewichtsvektoren der Prozessorelemente sehr viele Einsen enthalten (Abbildung IV.6). Die Bitfehlerwahrscheinlichkeit ist entscheidend für die Fehlertoleranz-Untersuchungen. Ausführungen finden sich in Kohonen (KOH72), Palm (PAL80) und Willshaw (WIL71). Ein Muster wird fehlerhaft ausgelesen, wenn ein Ausgabeelement in den aktiven Zustand übergeht, obwohl das Zielmuster dort nicht aktiviert ist, oder wenn das Ausgabemuster nicht aktiviert ist, obwohl das Zielmuster an der Position aktiviert ist, d.h.

- a) $a_j \geq TH_j$, aber $O_j^t = 0$ und
- b) $a_j < TH_j$, aber $O_j^t = 1$.

Der Fehler b) tritt auf, wenn das Eingabemuster oder die Verbindungselemente gestört sind, z.B. wenn das Eingabemuster weniger als 1 Einsen enthält oder die Verbindungselemente eines Prozessorelementes ausfallen.

Betrachten wir zunächst einen Assoziativspeicher, der keine gestörten Verbindungselemente enthält und bei dem das Eingabemuster korrekt vorliegt. Sind die l Einsen des Eingabe- bzw. die k Einsen des Ausgabevektors zufällig in den Vektoren verteilt, so ist es einfach, die Wahrscheinlichkeit P_{on} zu bestimmen, mit der ein Verbindungsgewicht (Knoten) aktiv ist, nachdem alle z Muster angelernt wurden. Wird ein Muster eingespeichert, so werden $l * k$ der insgesamt $m * n$ Verbindungen aktiviert. Damit ist ein Verbindungsgewicht mit der Wahrscheinlichkeit $l*k/(m*n)$ eingeschaltet, und $1 - l*k/(m*n)$ ausgeschaltet. Nach dem Anlernen von z Mustern gilt:

$$1 - P_{on} = \left(1 - \frac{k*l}{m*n}\right)^z$$

und damit:

$$P_{on} = 1 - \left(1 - \frac{k*l}{m*n}\right)^z \quad (IV.1)$$

Abbildung IV.7 zeigt die Belegungswahrscheinlichkeit oder den Füllungsgrad p_{on} für $z = 10, 20$ und 30 Muster mit $l = k = 5$ und $m = n = 35$. Mit steigender Musteranzahl z steigt auch die Belegungswahrscheinlichkeit für zufällige Muster ($p_{on,10} = 0.19, p_{on,20} = 0.34$ und $p_{on,30} = 0.45$).

Ein Fehler in der Position j des Ausgabevektors tritt also dann auf, wenn alle l Einsen der Eingabe auf eine aktivierte Verbindung des Prozessorelementes j treffen und dies ist genau dann der Fall, wenn für jede Eins an der Position i des Eingabevektors \underline{I}^t ein Musterpaar $(\underline{I}^s, \underline{O}^s)$ unter allen Mustern existiert, so daß in \underline{I}^s an der Stelle i und \underline{O}^s an der Stelle j eine Eins vorkommt ($s, t = 1..z$). Diese Bedingung wird nach Palm (PAL80) mit $C(t, j)$ bezeichnet. Die Anzahl der fehlerhaften Einsen N_{t1} im Ausgabevektor des Musterpaares $(\underline{I}^t, \underline{O}^t)$ ergibt sich zu:

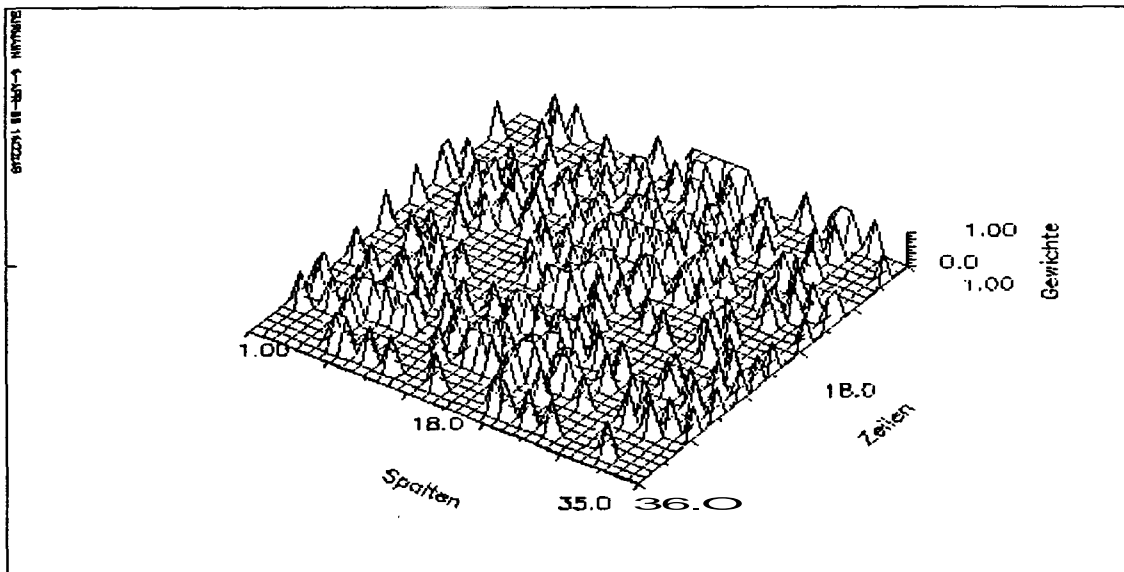
$$N_{t1} = \sum_{j \notin \underline{O}^t} 1[C(t, j) \text{ trifft zu}], \quad (IV.2)$$

wobei $1[C(t, j) \text{ trifft zu}]$ eine Zufallsvariable ist, die eine Eins ergibt, falls der Fehler auftritt, andernfalls eine 0.

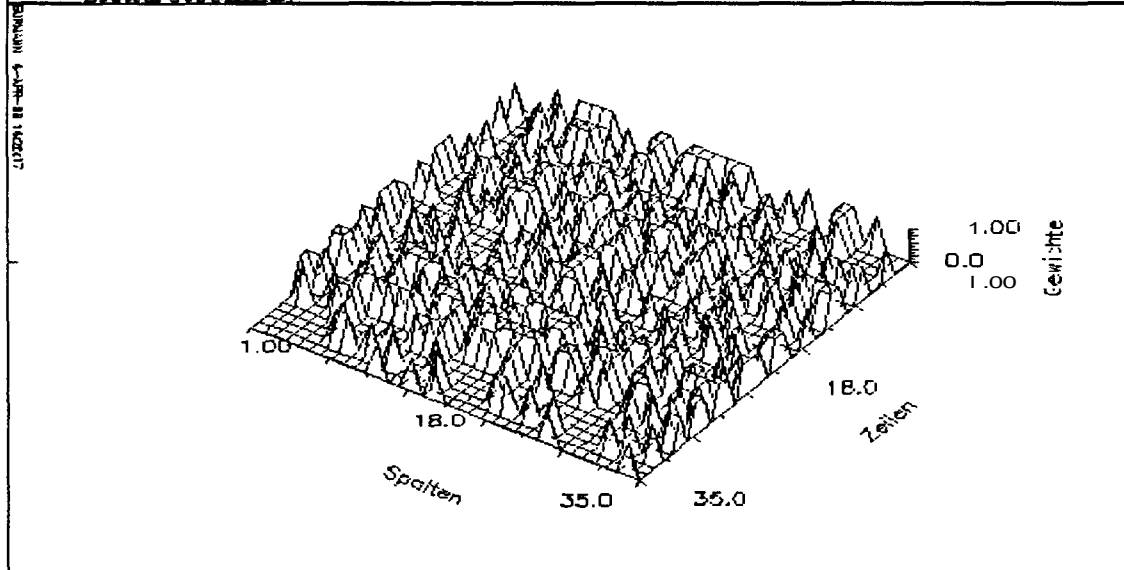
Es berechnet sich $\mathfrak{p} := P[1[C(t, j) \text{ trifft zu}]]$, die Wahrscheinlichkeit für einen Fehler a) im Falle von Hetero-Assoziation, zu:

$$\mathfrak{p} = P[\forall i \in \{1..n\}: I_i^t = 1 \exists s \in \{1..z\}: I_i^s = O_j^s = 1]$$

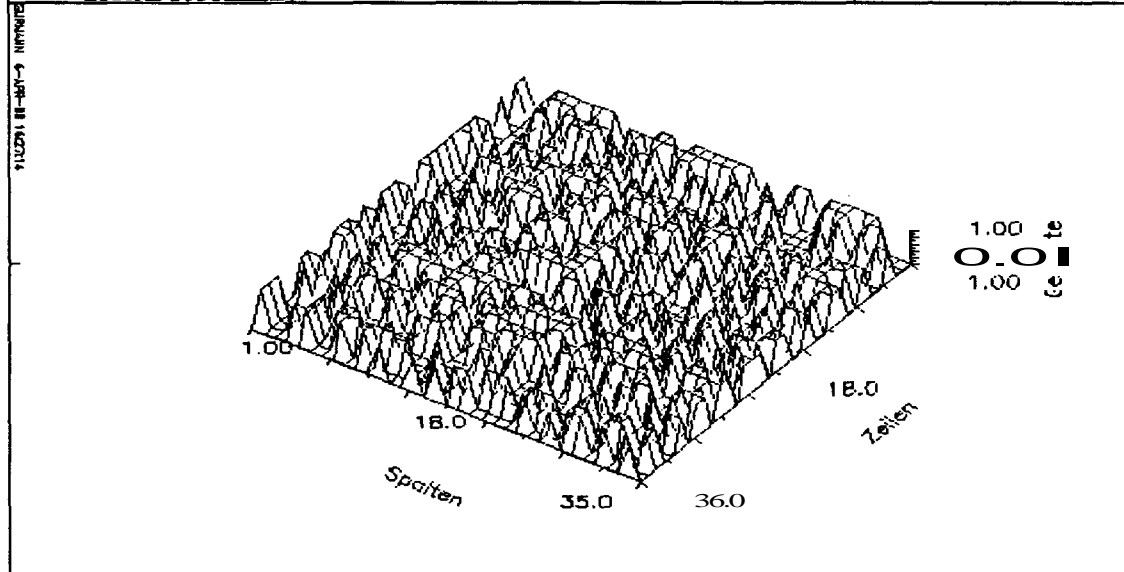
$$= 1 + \sum_{h=1}^l (-1)^h \binom{l}{h} \left(1 - \frac{k}{n}(1 - T(m, l, h))\right)^{z-1}, \quad (IV.3)$$



BE Bauelemente der Elektrotechnik $p - 0.19$ 3 D-Plot



BE Bauelemente der Elektrotechnik $p - 0.339$ 3 D-Plot



BE Bauelemente der Elektrotechnik $p - 0.45$ 3 D-Plot

Abbildung IV.7 : Belegungswahrscheinlichkeit p_{on} für $z = 10, 20$ und 30 Muster

$$\text{mit } T(a,b,c) = \prod_{i=0}^{c-1} \frac{a-b-i}{a-i} = T(a,c,b). \quad (\text{PAL80})$$

Die Bitfehlerwahrscheinlichkeit läßt sich auch über einen anderen Ansatz berechnen. Dazu wird angenommen, daß die Aktivität $a \in A$ eines Prozessorelementes eine Zufallsvariable ist, die einer Binomial-Verteilung folgt. Die Binomial-Verteilung $P(X = k) = \binom{n}{k} p^k (1-p)^{n-k}$ gibt an, mit welcher Wahrscheinlichkeit in einer Folge von n unabhängigen Versuchen das Ereignis $X = k$ auftritt. Sie ist abhängig von der Anzahl der Versuche n und der Wahrscheinlichkeit p , mit der das Ereignis bei einem Versuch eintritt. Auf den Assoziativspeicher übertragen bedeutet dies, daß unter den l Einsen in der Eingabe ($l = n$) sich $a = k$ befinden, die auf eine aktivierte Verbindung treffen, und zwar mit der Wahrscheinlichkeit p_{0n} ($p_{0n} = p$). Es berechnet sich die Wahrscheinlichkeit für den Fehler a) durch Aufsummieren der einzelnen Wahrscheinlichkeiten, d.h.

$$P(a \geq TH = l, O_j^t = 0) = \sum_{i=1}^l \binom{l}{i} * p_{0n}^i * (1 - p_{0n})^{l-i} \quad (\text{IV.4})$$

$$= \binom{l}{1} * p_{0n}^1 * (1 - p_{0n})^0 = p_{0n}^1.$$

Die Wahrscheinlichkeit für einen Fehler a) bei Verwendung von kleineren Schwellenwerten $TH = l' < l$, erhält man durch Einsetzen von l' in Gleichung (IV.4). Allerdings ist Gleichung IV.4 nur gültig für die Annahme, daß die Einsen zufällig in die Matrix gestreut werden. Palm zeigt jedoch, daß die korrekte Wahrscheinlichkeit lp asymptotisch gegen p_{0n}^1 geht ($(p - p_{0n}^1)/p_{0n}^1 \rightarrow 0$, PAL80, Appendix II).

Nach der Wahrscheinlichkeitsrechnung läßt sich für asymptotische Abschätzungen die Binomial-Verteilung durch die standardisierte Normal-Verteilung $N(0,1)$ approximieren, wenn:

$$l * p_{0n} * (1 - p_{0n}) > 9 \quad (\text{IV.5})$$

ist (BOW80) mit Erwartungswert $E(A) = p_{0n} * l$ und Standardabweichung $\beta(A) = p_{0n} * l * (1 - p_{0n})$.

$N(t)$ ist die Gauß-Verteilung mit:

$$N(t) = \frac{1}{\sqrt{2 * \pi}} \int_{-\infty}^t e^{-0.5 * x^2} dx. \quad (\text{IV.6})$$

Für die Anzahl l der Einsen im Eingabevektor folgt mit Gleichung IV.5, daß l mindestens größer 36 sein muß. Da häufig spärlich besetzte

Muster angenommen werden ($I=O(\log(n))$), verwende ich die Binomial-Verteilung, die für spärlich besetzte Muster numerisch besser ausgewertet werden kann. Der Ausgabevektor O besitzt $(m - k)$ Positionen für einen Fehler a) und k Positionen für einen Fehler b), d.h. der Erwartungswert für die Anzahl der Fehler $E(N_t) = E(N_{t1}) + E(N_{t0})$ errechnet sich allgemein durch:

$$E(N_t) = P(a \geq TH_i, o^{\dagger} = 0) * (m-k) + P(a < 1, O_j^{\dagger} = 1) * k. \quad (IV.7)$$

$$\begin{aligned} &= (m - k) * p + k * P(a < 1, o^{\dagger} = 1) \\ &= (m - k) * p_{on}^1 + k * P(a < 1, O_j^{\dagger} = 1) \text{ [asymptotisch]} \end{aligned}$$

Für den hier betrachteten Fall ist der Fehler b) ($P(a < TH_j, O_j^{\dagger} = 1)$) gleich Null, da keine Verbindungselemente gestört sind, das Eingabemuster korrekt vorliegt und durch die Lern- und Aktivierungsfunktion gesichert ist, daß für $O_j^{\dagger} = 1, a_j = 1$ ist, d.h.

$$E(N_t) = (m - k) * p \approx (m - k) * p_{on}^1 \quad (IV.8)$$

Werden alle z Muster, die im Assoziativspeicher gespeichert sind, nacheinander ausgelesen, so ergibt sich der Erwartungswert für die Gesamtanzahl $E(N_A)$ aller Fehler durch $z * E(N_t)$, d.h.

$$E(N_A) = z * (m - k) * p \approx z * (m - k) * p_{on}^1 \quad (IV.9)$$

Im folgenden werden unterschiedliche Ansätze diskutiert, um mit Hilfe von Gleichung (IV.9) zu einer Abschätzung der Musteranzahl z zu gelangen.

IV.5.1 Musteranzahl des fehlerfrei arbeitenden Assoziativspeichers

Sollen alle Muster, die im Assoziativspeicher gespeichert sind, korrekt ausgelesen werden, so muß $E(N_A) < 1$ sein, d.h.

$$1 > z * (m - k) * p \text{ bzw.} \quad (IV.10)$$

$$1 > z * (m - k) * p_{on}^1 \quad (IV.11)$$

Beide Ungleichungen lassen sich durch numerische Verfahren, z.B. mit Hilfe der Intervallhalbierungsmethode, lösen. Als Startwerte eignen sich $z_{\min,1} = 1$ und $z_{\max,1} = (m*n)/(1*k)$. Tabelle IV.1 zeigt ein Beispiel für $m = n = 1000$ und $k = 1 = 3$.

Iteration	Gleichung IV.10		Gleichung IV.11	
	Z _{min}	Z _{max}	Z _{min}	Z _{max}
1	1	111111	1	111111
2	1	55556	1	55556
3	1	27778	1	27778
4	1	13889	1	13889
5	1	6945	1	6945
6	1	3473	1	3473
7	1	1737	1	1737
8	869	1737	869	1737
9	869	1303	869	1303
10	869	1086	1086	1303
11	869	977	1086	1194
12	923	977	1086	1140
13	923	950	1086	1113
14	936	950	1086	1099
15	943	950	1086	1092
16	946	950	1086	1089
17	948	950	1087	1089

Tabelle IV.1: Numerische Lösung der Gleichungen IV.10 und IV.11 für $m = n = 1000$ und $k = l = 3$.

IV.5.2 Musteranzahl des Assoziativspeichers bei höchstens einem Fehler pro Muster

Einen anderen Ansatz, der von Willshaw (WIL70) untersucht wurde, ergibt sich durch die Forderung, daß $E(N_t) < 1$ sein soll, d.h.

$$1 \geq (m - k) * p \text{ oder asymptotisch } 1 \geq (m-k) * p_{on}^1 \quad (IV.12)$$

Die erste Ungleichung läßt sich wieder nur numerisch lösen, aber mit Hilfe der zweiten Ungleichung besteht die Möglichkeit, z analytisch zu berechnen.

$$1 \geq (m-k) * p_{on}^1$$

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

und damit gilt:

$$z \leq \frac{\log \left(1 - \left(\frac{1}{m-k} \right)^{1/l} \right)}{\log \left(1 - \left(\frac{k \cdot l}{m \cdot n} \right) \right)} \quad (\text{IV.13})$$

McClelland (RMC86) erhält mit einer stärkeren Einschränkung in (IV.12), der Abschätzung $\log(1-x) = -x$ und der Annahme spärlich besetzter Eingabe- und Ausgabevektoren ($k = O(\log(m))$), den linearen Zusammenhang:

$$z \leq 0.69 \frac{n \cdot m}{k \cdot l} \quad (\text{IV.14})$$

Als Ergebnis wird festgehalten, daß die Speicherkapazität des Assoziativspeichers mit der Anzahl der Verbindungen wächst, und daß sie mit wachsender Anzahl von Einsen in der Eingabe fällt, was intuitiv klar ist. Es ist

$$z_{\max} = O\left(\frac{m \cdot n}{\log(n) \cdot \log(m)}\right) \quad (\text{IV.15})$$

für spärlich besetzte Eingabe- und Ausgabevektoren.

IV.6 Informationsspeicherkapazität des Assoziativspeichers

Im Gegensatz zum herkömmlichen Speicher, in dem alle Musterpaare hintereinander abgelegt werden, kommt es beim Assoziativspeicher zu Überlagerungen in der Matrix der Verbindungsgewichte, wenn nicht alle Musterpaare orthogonal zueinander stehen. Speichert man sehr viele, zufällig gewählte Muster in den Assoziativspeicher (s.o), so treten Bitfehler in den Ausgabevektoren auf. Der falsche Ausgabevektor enthält weniger Information als der korrekte. Hat der binäre Vektor die Dimension m , dann existieren 2^m mögliche binäre Vektoren. Nach Shannon's Informationstheorie (SHW49) beträgt der Informationsgehalt des binären Vektors $m = \text{ld}(2^m)$. Hat der binäre Vektor die Dimension m und enthält genau k Einsen, dann existieren $\binom{m}{k}$ Vektoren mit dem Informationsgehalt $\text{ld}\left(\binom{m}{k}\right)$. Mit Hilfe der Zufallsvariablen N_t , der Anzahl der fehlerhaften Bits im Ausgabemuster \mathbf{O}^t läßt sich der Informationsverlust für jedes Musterpaar $(\mathbf{I}^t, \mathbf{O}^t)$ bestimmen durch $\text{ld}\left(\binom{m}{N_t+k}\right)$. Es ergibt sich die Information, die in einem Assoziativspeicher gespeichert ist, durch:

$$I = \sum_{t=1}^t I_t = \sum_{t=1}^t \left[\text{ld}\left(\binom{m}{k}\right) - \text{ld}\left(\binom{N_t+k}{k}\right) \right] \quad (\text{IV.16})$$

Durch **Umformung** und Einsetzen von $E(N_t)$ erhält man den Erwartungswert $E(I)$ für die Informationsspeicherkapazität für ungestörte Verbindungsgewichte und Eingaben (PAL80):

$$EU \geq -z \sum_{i=0}^{k-1} \text{ld} \left(\frac{(m-k) * p^{k-i}}{m - i} \right) \quad (IV.17)$$

$$\approx -z \sum_{i=0}^{k-1} \text{ld} \left(\frac{(m-k) * p_{0n}^{k-i}}{m - i} \right) \quad (IV.18)$$

Abbildung IV.8 zeigt den Verlauf der Informationsspeicherkapazität $E(I)$ in Abhängigkeit von der Musteranzahl z . Im Gegensatz zum aufzählenden Speicher ist die Kapazität nicht auf einen festen Wert beschränkt, sondern variabel. Werden zu viele Muster in den Verbindungsgewichten gespeichert, so erhöht sich die Anzahl der Bitfehler N_t . Im Maximum der Informationsspeicherkapazität treten pro Muster im Mittel mehr als 69 Fehler auf (Tabelle IV.2).

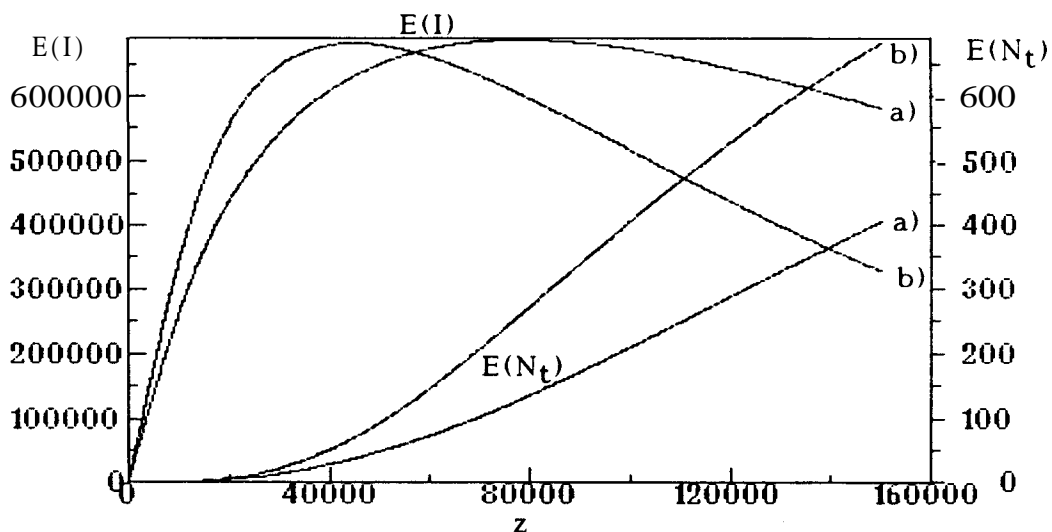


Abbildung IV.8: Informationsspeicherkapazität $E(I)$ und Erwartungswert für die Anzahl der Bitfehler $E(N_t)$ in Abhängigkeit von der Musteranzahl z , $m = n = 1000$, a) $l = k = 3$, b) $l = k = 4$.

Abbildung IV.9 zeigt den Verlauf der Wahrscheinlichkeit p_{0n} , mit der ein Knoten aktiviert ist, in Abhängigkeit von der Musteranzahl. Es zeigt sich, daß, je voller die Verbindungsgewichtsmatrix ist, die Wahrscheinlichkeit p für einen Bitfehler vom Typ a) steigt.

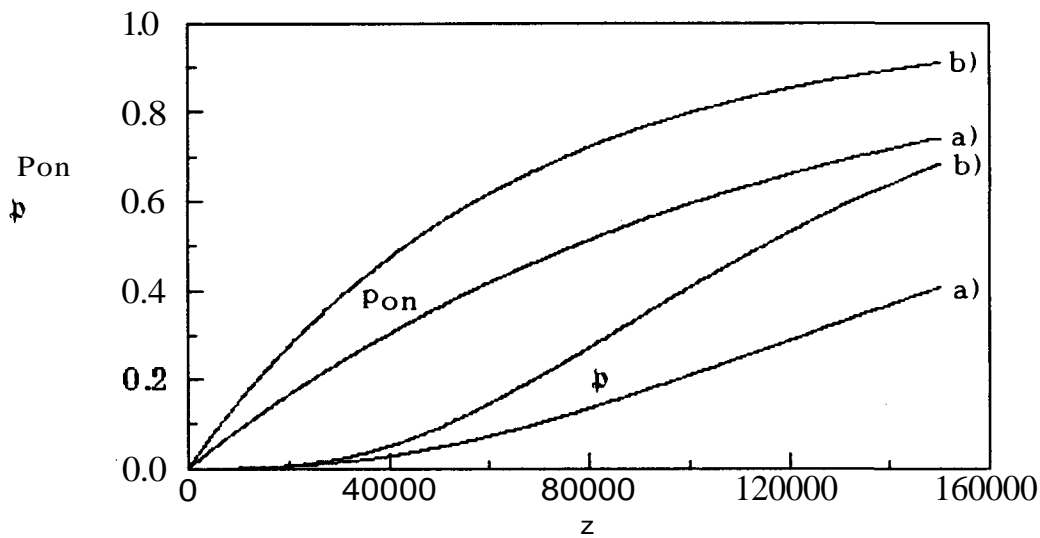


Abbildung IV.9: Ein-Bit-Fehlerwahrscheinlichkeit p und Belegungswahrscheinlichkeit p_{On} der Verbindungsgewichtsmatrix in Abhängigkeit von der Musteranzahl z . $m = n = 1000$, a) $l = k = 3$, b) $l = k = 4$.

Tabelle IV.2 zeigt die Werte der Informationsspeicherkapazität, Gesamt- und Bitfehleranzahl, Bitfehlerwahrscheinlichkeit und Belegungswahrscheinlichkeit für die Musteranzahlen aus den Gleichungen IV.10 - IV.14 ($z_1 - z_5$); z_6 bezeichnet die Musteranzahl bei maximaler Informationsspeicherkapazität.

	z_1	z_2	z_3	z_4	z_5	z_6
z	1555	1745	12003	12251	43125	45000
p	$6.46 \cdot 10^{-7}$	$9.65 \cdot 10^{-7}$	0.001	0.00108	0.0625	0.0702
p_{On}	$3.65 \cdot 10^{-7}$	$5.75 \cdot 10^{-7}$	0.0009	0.001	0.0617	0.0694
P_{On}	0.0246	0.0275	0.1747	0.1780	0.498	0.513
$E(N_t)_1$	0.006	0.010	1.0	1.075	62.235	69.913
$E(N_t)_2$	0.004	0.006	0.9285	.9999	61.470	69.116
$E(N_A)_1$	1.0009	1.6768	12002.9	13170	2683903	3146088
$E(N_A)_2$	0.5647	0.9989	11144.5	12249	2650902	3110208
$E(I)_1$	54841	61540	395470	401961	680918	681410
$E(I)$	54842	61542	397080	403643	683878	684286

Tabelle IV.2: Ausgewählte Werte der Informationsspeicherkapazität, Gesamt- und Bitfehleranzahl, Bitfehlerwahrscheinlichkeit und Belegungswahrscheinlichkeit ($m = n = 1000$, $k = l = 4$).

Tabelle IV.3 zeigt die maximale Informationsspeicherkapazität $E(I)$ für verschiedene Werte von l und k bei $m = n = 1000$ (aus PAL80). Es zeigt sich, daß die Informationsspeicherkapazität mit der Anzahl der Einsen in den Eingabe- bzw. Ausgabemustern fällt.

$l \backslash k$	2	3	4
2	690.6	690	689
3	688	687	686
4	685	683	681

Tabelle IV.3: Maximale Information (in 10^3 bits) für verschiedene Werte von l und k ($m = n = 1000$).

IV.7 Ableitungen für **zerstörte** Verbindungselemente

Als nächstes wird der Fall betrachtet, daß bei der Fertigung oder während des Betriebs des Assoziativspeichers ein harter Fehler bei den Verbindungsgewichten auftritt, d.h. eine Verbindung ausfällt und damit $w_{ij}(t) = 0, \forall t \in \{0..∞\}$ wird. Das Prozessorelement j erhält keine Eingabe an der Position i . Im folgenden wird angenommen, daß alle Verbindungselemente mit der gleichen Wahrscheinlichkeit ausfallen können, d.h. $b := P(w_{ij} \text{ fällt aus})$ für alle i und j identisch ist. Wir erhalten einen Assoziativspeicher, der nur noch $n * m * (1 - b)$ Verbindungen enthält. In der Literatur heißen solche Netze 'Zufällig verbundene Netze' (RMC86) oder auch 'Assoziativspeicher mit zufällig verteilten Speicherelementen' (PAL81). Nach dem Einspeichern von z Mustern beträgt die Wahrscheinlichkeit dafür, daß ein Knoten durch das Lernen aktiviert wird, nach Gleichung IV.1:

$$P_{on} = 1 - \left(1 - \frac{k * l}{m * n}\right)^z$$

und zwar unabhängig vom Grad der Zerstörung.

Es sei $j := 1 - b$ die Wahrscheinlichkeit für die Existenz einer Verbindung, dann berechnet sich die Wahrscheinlichkeit dafür, daß ein Knoten aktiviert wird, durch:

$$P_{on, j} = \left(1 - \left(1 - \frac{k * l}{m * n}\right)^z\right) * j. \quad (IV.19)$$

Für eine binomial verteilte Aktivität a ; (analog Abschnitt IV.5) erhält man den Fehler vom Typ a) durch:

$$P(a \geq TH_j, O_j^t = 0) = \sum_{i=TH_j}^l \binom{l}{i} * (p_{on} * \delta)^i * (1 - p_{on} * \delta)^{l-i}. \quad (IV.20)$$

Beispiel IV.1:

Für $p_{on} = 0.5$, $b = 0.05 = 5\%$, $\delta = 1 - b = 0.95$, $l = 10$, ergibt sich

$$P(a \geq 10, O_j^t = 0) = 5.85 * 10^{-4} \text{ und } P(a \geq 9, O_j^t = 0) = 7.047 * 10^{-3}$$

Für den Fehler vom Typ b) gilt bei binomial verteilter Aktivität:

$$P(a < TH_j, O_j^t = 1) = \sum_{i=0}^{TH_j-1} \binom{l}{i} * \delta^i * (1 - \delta)^{l-i}, \quad (IV.21)$$

da durch das Lernverfahren gesichert ist, daß die Verbindung aktiviert ist. Für das Beispiel IV.1 ergibt sich $P(a < 10, O_j^t = 1) = 0.401$ und $P(a < 9, O_j^t = 1) = 0.0861$.

Setzt man Gleichung IV.20 und Gleichung IV.21 in Gleichung IV.7 ein, so erhält man :

$$E(N_t)_\delta = (m-k) * \sum_{i=TH_j}^l \binom{l}{i} * (p_{on} * \delta)^i * (1 - p_{on} * \delta)^{l-i} + k * \sum_{i=0}^{TH_j-1} \binom{l}{i} * \delta^i * (1 - \delta)^{l-i} \quad (IV.22)$$

$E(N_t)_\delta$ ist eine Funktion von TH . Für $TH; \approx 1$ ist $P(a < l_0, O_j^t = D)$ am größten, d.h. eine eigentlich aktivierte Ausgabeleitung wird nicht aktiviert. Hingegen ist $P(a \geq l_0, O_j^t = 0)$ am kleinsten, was bedeutet, daß keine zusätzlichen Einsen erzeugt werden.

Minimiert man $E(N_t)_\delta$ bezüglich TH für ein festes δ und z mit Hilfe eines numerischen Verfahrens, so ergibt sich eine **untere Grenze** für die Anzahl der zu erwartenden Fehler $E(A)_\delta$ bei zerstörten Verbindungen durch:

$$E(N_A)_\delta = \min(E(N_t)_\delta) * z. \quad (IV.23)$$

Für das Beispiel IV.1 mit $k = l = 10$ und $m = n = 1000$ und $z = 6932$ berechnet sich $\min(E(N_t)_\delta) = 5.158$ bei $l_0 = 10$ und $E(N_A)_\delta = 35757$. In den Ausgabemustern werden dabei im Mittel 4.5 Einsen weggelassen und 0.5 zusätzliche Einsen erzeugt.

Für die Informationsspeicherkapazität läßt sich eine **obere Grenze** bestimmen, indem $\min(E(N_t)_\delta)$ in Gleichung IV.17 eingesetzt wird, d.h.

$$E(I)_\delta \leq -z \sum_{i=0}^{k-1} \text{ld} \left(\frac{\min(E(N_t)_\delta) + k - i}{m - i} \right). \quad (IV.24)$$