Neuroprozessor mit Echtzeitfähigkeit für optische Erkennungsaufgaben

Neuro Processor with Real Time Ability for Visual Recognition Tasks

F. Stüpmann, **G. Geske**, **A. Wego**, Rostock

Kurzfassung

Derzeit entwickelt die Firma Silicann Technologies ein künstliches neuronales Netz, das in analoger zeitkontinuierlicher Schaltungstechnik realisiert wird. Durch die voll-parallele Architektur lassen sich enorme Verarbeitungsgeschwindigkeiten im unteren Mikrosekundenbereich erreichen. Dieser IC mit dem Namen „Silimann“ besitzt für zeitkritische Assoziations-, Klassifikations- und Erkennungsanwendungen hervorragende Eigenschaften. Dargestellt wird das Training eines solchen Chips mit Off-Chip und „in the Loop“ Strategien, die für die Anwendung in Muster- und Farberkennungsaufgaben eingesetzt werden können.

Abstract

At present the company Silicann Technologies develops an artificial neural network which is designed in analog time continuous circuit technique. Very high processing speeds in the lower microsecond range can be achieved due to full parallel architecture. The integrated circuit named "Silimann" has excellent qualities for time-critical association-, classification- and detection applications. This paper shows how such a chip can be trained using off-chip and “in the loop”-strategies, which can be applied for pattern and color detection.

1. Einleitung

In der Industrie der Zukunft werden schnelle, autonome Erkennungssysteme immer mehr gefragt sein. Schnell bedeutet dabei Verarbeitungszeiten im Mikrosekundenbereich. Herkömmliche Berechnungssysteme stoßen hier an ihre Leistungsgrenzen. Deshalb wird weltweit nach alternativen Berechnungsmethoden gesucht. Eine ist die Methode der künstlichen neuronalen Netze. Hierbei wird kein fertiges Programm gespeichert, sondern das Programm, d. h. die Netzparameter, werden adaptiv erlernt. Die Daten werden parallel verarbeitet. Die parallele Verarbeitung führt zu einer enormen Geschwindigkeit der Netze.

Durch die Belehrbarkeit sind neuronale Netze in Fällen einsetzbar, in denen für die Aufgaben weder Regeln noch mathematischen Modelle existieren. Die biologischen Vorbilder stellen den Ausgangspunkt für die künstlichen neuronalen Netze dar. Wenn neuronale Netze auf sequentiellen Standardprozessoren simuliert werden, bleibt ein großer Vorteil --- die Parallelität --- ungenutzt. Deshalb wird an der Implementierung künstlicher neuronaler Netze in integrierter Hardware gearbeitet [6].

2. Netzwerkstruktur und Lernstrategien und -algorithmen

Die Struktur eines neuronalen Netzes ist in Bild 1 dargestellt. Hierbei handelt es sich um ein Netz mit 3 Neuronenschichten. Die Synapsen, die durch die Verbindungslinien zwischen den Neuronen repräsentiert sind, enthalten die in einem Trainingsprozess ermittelten Gewichtswerte. Aus dieser Sicht kann ein Netz mit einem Layer als zweischichtig (mit den Layer *j* und *k*) angesehen werden.

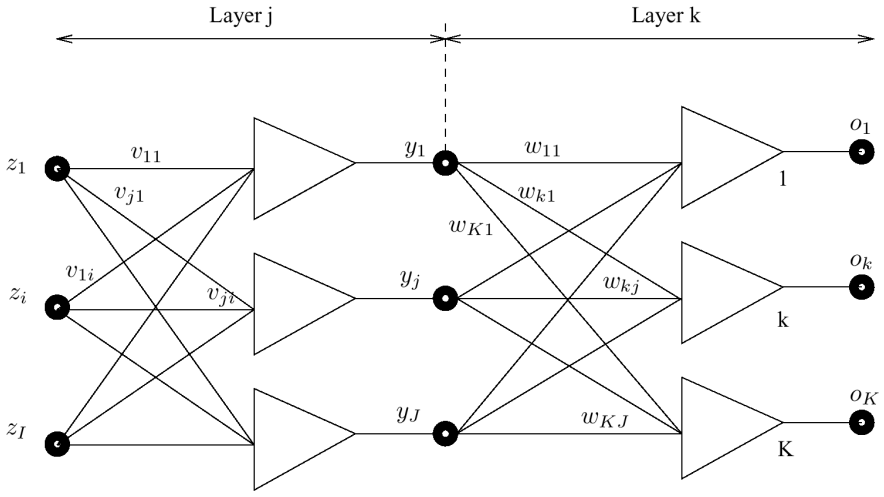

Bild 1: Neuronales Netz mit einem Hiddenlayer

Die mathematische Operation kann als zweistufige Vektor-Matrix-Operation aufgefasst werden, wobei die Vektoren *z*, *y*, und *o* und die Matrizen *V* und *W*

$$z = \begin{bmatrix} y_1 \\ z_i \\ \vdots \\ z_I \end{bmatrix};\ y = \begin{bmatrix} y_1 \\ y_j \\ \vdots \\ y_J \end{bmatrix};\ o = \begin{bmatrix} o_1 \\ o_k \\ \vdots \\ o_K \end{bmatrix};\ V = \begin{bmatrix} v_{11} & v_{22} & \cdots & v_{1I} \\ v_{21} & v_{22} & \cdots & v_{2I} \\ \vdots & \vdots & \ddots & \\ v_{J1} & v_{J2} & \cdots & v_{JI} \end{bmatrix};\ W = \begin{bmatrix} w_{11} & w_{12} & \cdots & w_{1J} \\ w_{21} & w_{22} & \cdots & W_{2J} \\ \vdots & \vdots & \ddots & \vdots \\ w_{K1} & w_{K2} & \cdots & w_{KJ} \end{bmatrix} \quad (1)$$

folgendermaßen miteinander verknüpft sind:

$$y = \Gamma[Vz] \qquad \text{und} \qquad o = \Gamma[Wy] \quad (2)$$

wobei Γ der Diagonaloperator der Neuronenaktivierungsfunktion f darstellt. Die Aktivierungsfunktion der Neuronen ist dabei eine sigmoide Übertragungsfunktion, wie z.B. die tanh-Funktion. Diese ist nötig, um linear nichtseparierbare Probleme zu lösen. Die Umsetzung der Vektor-Matrix-Multiplikationen geschieht mittels zweier Vier-Quadranten-Multiplizierer-Arrays, die in einem CMOS Chip, der in analoger VLSI-Technik gefertigt wird, integriert sind. Die Synapsengewichte V bzw. W, die mit den Eingangsvektoren des jeweiligen Arrays multipliziert werden, werden auf kapazitive Weise gespeichert. Die Erzeugung der sigmoiden Aktivierungsfunktion geschieht durch Ausnutzung der Transistorkennlinien.

2.1. Lernstrategie

Um ein analoges neuronales Netz (Hardware) zu trainieren gibt es grundsätzlich drei Möglichkeiten [2]:

1. On-Chip: das Training wird vollständig auf dem Chip ausgeführt
2. „in the Loop“: die Chipantwort wird für die Berechnung des Gewichtsupdates verwendet
3. Off-Chip: das Training wird komplett außerhalb des Chips durchgeführt (z.B. auf einem PC); die berechneten Gewichte werden anschließend in den Chip geladen

Die erste Variante ist die schnellste, jedoch hat dieses Verfahren Einbußen in der Genauigkeit der Berechnungen (siehe Abschnitt 2.2.1). Die zweite kann die Nichtidealitäten der analogen Schaltungstechnik (Kennlinienstreuungen, Offset) völlig kompensieren. Nachteilig wirkt sich hier der hohe Kommunikationsaufwand durch das ständige Neubeschreiben der Gewichtsspeicher aus. Die dritte Variante ist für kleine Netze ebenfalls schnell, kann jedoch die durch herstellungsbedingte Toleranzen hervorgerufenen Fehler im Chip nicht kompensieren.

2.2. Verwendete Algorithmen

2.2.1. Backpropagation-Algorithmus

Damit das neuronale Netz das gewünschte Übertragungsverhalten aufweist, müssen die entsprechenden Gewichtsmatrizen V und W berechnet werden. Dafür wird der sehr verbreitete Backpropagation-Algorithmus, mit dem der am Ausgang auftretenden Fehler

$$e = d - o \tag{3}$$

„zurückpropagiert“ wird, benutzt[1]. Dieser Fehler wird mit Hilfe dieses Gradientenabstiegsverfahrens minimiert. Die Änderung eines Gewichtes w_{kj} im Layer k berechnet sich wie folgt[2] [5]:

---

[1] d – gewünschter Trainingswert am Ausgang

$$\Delta w_{kj} = h(d_k - o_k) f'(o_k) y_j \tag{4}$$

Im Layer *j* sieht die Berechnung des Gewichtsänderung etwas komplexer aus:

$$\Delta v_{ji} = h f'(y_j) z_i \sum_{k=1}^{K} (d_k - o_k) f'(o_k) w_{kj} \tag{5}$$

Die in diesem Verfahren berechneten Gewichtsupdates verringern den Ausgangsfehler e. Die Gewichtänderungen können entweder pro Lernzyklus oder pro Lernepoche durchgeführt werden, wobei ein Zyklus das Anlegen eines einzelnen Musters und einer Gewichtskorrektur umfasst. Eine Epoche hingegen umfasst das einmalige Anlegen aller zu trainierenden Muster. Nach Unterschreiten eines bestimmten Fehlers ε wird das Lernen abgebrochen.
Beim Minimieren des Fehlers werden alle schaltungstechnischen Nichtidealitäten (wie z.B. Offset), die in dem vorwärts[3] betriebenen Netz enthalten sind, kompensiert. Die Berechnung des Gewichtsupdates kann unter Einbeziehung von Analog-Multplizierern und Subtrahierern erfolgen [3], was jedoch zu einem erheblichen Flächenaufwand und zu Ungenauigkeiten der Gewichtsupdateberechnung nach Gleichungen (4) und (5) durch Offsets führt. Demgegenüber steht allerdings eine hohe Trainingsgeschwindigkeit.

2.2.2. Weight-Pertubation

Ein weiteres Lernverfahren, dass für einen analogen Neuro-Chip angewandt werden kann ist der Weight-Pertubation-Algorithmus [2]. Mit diesem werden die schaltungsbedingten Fehler vermieden, da dieser in einer „in the Loop“ Strategie arbeitet. Hierbei wird eine endliche Differenz des Fehlergradienten in Bezug zu den Gewichten $\frac{\partial E}{\partial w}$ ermittelt. Dazu werden die Gewichte mit kleinen Störungen $\boldsymbol{p}$ überlagert und die Auswirkung auf den Ausgangsfehler beobachtet. Verringert sich der Fehler, kann das Vorzeichen der Gewichtsänderung (Störung) beibehalten werden. Die Änderung berechnet sich für ein einzelnes Gewicht nach[4]:

$$\Delta w_{kj} = -h \frac{e(w+p) - e(w)}{p_{kj}} \tag{6}$$

---

[2] h - Lernrate, die zum Einstellen der Lerngeschwindigkeit erforderlich ist

[3] es wird Vorwärts- und Rückwärtsphase unterschieden; in der Vorwärtsphase verarbeitet das Netz die am Eingang anliegenden Muster; in der Rückwärtsphase (Training) werden die Fehler rückwärts durch das propagiert

[4] $e(w)$ - Netzwerkfehler bei einem Gewichtsatz w; $e(w+p)$ - Netzwerkfehler für die Summe des aktuellen Gewichtssatzes mit einer Störung p; p - Gewichtsstörung

Bei einem sequentiellen Verfahren wird nur ein Gewicht pro Lernzyklus geändert, was jedoch auch parallelisiert werden kann, so dass die gesamte Gewichtsmatrix geändert wird:

$$\Delta w = -h(e(w+p) - e(w))p \tag{7}$$

Dieses Verfahren ist zwar zeitsparender, da ein geringerer Rechenaufwand erbracht werden muss, jedoch ist es auch nicht immer sehr effektiv, wenn man den Lernfortschritt pro Lernzyklus betrachtet. Daher werden Teilstörungen des Netzes vorgenommen, so dass beispielsweise nur alle Eingangs- (fan in) oder alle Ausgangssynapsen (fan out) eines Neurons gestört werden. Dieses ist für die Lernkonvergenz in vielen Fällen effektiver.

3. Modellierung

Die Modellierung des Neuro-Chips für das Off-Chip Training kann auf zwei Arten erfolgen: analytisch und numerisch.

3.1. Analytische Modellierung

3.1.1. Neuronen Schaltung

Das Neuron hat die Eingangsignale zu summieren und mit einer sigmoiden Übertragungsfunktion zu verknüpfen. In Softwarenetzen wird dafür oft die tanh- oder die logistische Funktion verwendet. Um ein sigmoides Übertragungsverhalten in einem Hardwareneuron zu erhalten wird eine sourcegekoppeltes MOSFET-Transistor Paar verwendet. Für die Schaltung ist eine gute Annäherung durch [4]

$$I_d(V_d) = kV_d\sqrt{\frac{2I_{ss}}{k} - V_d^{\,2}} \tag{8}$$

gegeben, wobei k durch den Herstellungsprozess und die Entwurfsparameter (W/L) bestimmt wird und m_0 die Ladungsträgerbeweglichkeit und C_{ox} die Gatekapazitätsdichte beschreibt.

$$k = m_0 C_{ox}\frac{W}{2L} \tag{9}$$

I_d ist der differentielle Ausgangsstrom, I_{ss} der Sourcestrom und V_d die differentielle Eingangsspannung.

3.1.2. Synapsenschaltung

Die Synapsenschaltung besteht aus einem Gilbert-Multiplizierer und einem analogen Gewichtspeicher. Die Funktion des Multiplizierers ist wie folgt gegeben [4]:

$$I_d(I_{in}, V_d) = k \cdot V_d \left(\sqrt{\frac{2(I_{ss} + I_{in})}{k} - {V_d}^2} - \sqrt{\frac{2(I_{ss} - I_{in})}{k} - {V_d}^2} \right) \quad (10)$$

Diese Gleichung nimmt gleiche Parameter für k in allen MOSFETs an. V_d (Spannung des Gewichtspeicherkondensators) moduliert den Eingangsstrom I_{in}, so dass der Ausgangsstrom ein Produkt aus diesen beiden Eingangsgrößen ist.
Der Gewichtsspeicher besteht aus einem Kondensator und einem als Spannungsfolger geschalteten Operationsverstärker. Die gespeicherte Spannung ist durch geeignete schaltungstechnische Maßnahmen bei Raumtemperatur mit einer durchschnittlichen Drift von nur 6µV/s relativ stabil und benötigt keine weitere Modellierung [1].

3.2. Numerische Modellierung

Bei der numerischen Modellierung werden die Übertragungsfunktionen der Einzelschaltungen aus Wertetabellen ermittelt, die durch Simulation in dem entsprechenden Simulationstool gewonnen werden. Die Genauigkeit hängt nur vom Simulationstool und den dahinter liegenden Bibliotheken und von der gewählten Schrittweite der Tabellenwerte ab. Mit dieser Methode werden die realen Verhältnisse recht gut wiedergegeben.

Der Vergleich der beiden Methoden der Modellierung hat ergeben, dass sich die numerische Methode besser eignet, da sie näher an der verwendeten Technologie arbeitet. Die analytische Methode müsste Modelle höherer Ordnung integrieren, um gleiches zu leisten, was jedoch aufwendig ist.

4. Anwendung Farb- und Mustererkennung

Eine mögliche Anwendung des Neuro-Chips ist die Farb- und Mustererkennung, die mit Einzelsensoren oder auch mit Sensorarrays durchgeführt werden kann. Damit kann der Chip in Verbindung mit optischen Sensoren zur Qualitätssicherung von Oberflächen, Prüfen und Sortieren von Bauteilen und für die Erkennung von Farbstrukturen, Farbverläufen und der Unterscheidung von Farbnuancen in vielfältigen industriellen Anwendungen eingesetzt werden. Aufgrund der begrenzten Anzahl von Eingängen (≤100) und der analogen Verarbeitung ist der Neuro-Chip für Aufgaben mit geringer Ortsauflösung und mittlerer spektraler Auflösung, die Verarbeitungsgeschwindigkeiten im unteren Mikrosekundenbereich erfordern, geeignet. Dazu können analoge Sensoren direkt angekoppelt werden. Der aktuelle

Chip besitzt 10 Ein- und Ausgangs-, sowie 6 Hiddenneuronen und erreicht eine Reaktionszeit von <20µs.

4.1. Off-Chip Training

Die mit diesem Chip unternommen RGB-Farbwertumsetzungen zeigen, dass das Training des Chips mittels der analogen Ausgangswerte des Sensors mit Hilfe des Off-Chip Modells recht gute Ergebnisse erzielt. Die Genauigkeit, die mit einem neuronalen Netzwerk erreicht werden kann, hängt jedoch von verschiedenen Einflussfaktoren ab: erstens von der Art und Qualität der Trainingsdaten, zweitens von der Topologie[5] des Netzes. Drittens begrenzen die Eigenheiten der analogen Schaltungstechnik die erreichbare Genauigkeit. Dazu zählen begrenzte Spannungsbereiche, Offsets etc. Der erste und der zweite Einflussfaktor zeigen typische Eigenschaften neuronaler Netze und treten gleichermaßen in Hardware, wie in Softwarenetzen auf. Durch die Modellierung (siehe Abschnitt 3) kann die Netzwerkcharakteristik gut nachgebildet werden, jedoch gilt dieses nicht für statistisch zufällig verteilte herstellungsbedingte Schwankungen einzelner Parameter wie z.B. Offsets.

4.2. Nachtraining mit Weight Pertubation

Der Fehler, der durch die Parameterstreuungen der Hardware hervorgerufen wird, kann durch ein „in the Loop“ Training mit dem Weight Pertubation Algorithmus wieder reduziert werden. Dieses wurde auf dem Chip nachgewiesen. Dabei bleiben jedoch Schranken, die durch die Netztopologie und die Art der Daten gegeben sind, bestehen.

5. Zusammenfassung

Der Einsatz eines analogen Neuro-Chips in der Farberkennung mit harten Echtzeitforderungen ist sehr vielversprechend. Verarbeitungszeiten im unteren Mikrosekundenbereich sind erreichbar. Das Training eines solchen Chips ist auf unterschiedliche Weise möglich. Die Kombination aus Off-Chip und „in the Loop“ Lernen ist recht effektiv, da die Nachteile der analogen Schaltungstechnik ausgeglichen werden können und ein relativ zügiges Training im Off-Chip Modus erreicht wird. Die Modellierung des Chips für das Off-Chip Training wurde auf analytische und numerische Weise durchgeführt. Unter Berücksichtigung des Aufwandes und des erzielbaren Ergebnisses ist die numerische Methode der analytischen Herangehensweise vorzuziehen.

---

[5] Topologieparameter sind beispielsweise die Anzahl der Hiddenneuronen oder die Anzahl der Hiddenlayer

6. Literatur

[1] G. Geske, F. Stüpmann, S. Rode: "Artificial Neural Network in analog VLSI-Technology", Baltic Electronic Conference, BEC, October 6.-9. 2002, Tallinn, Estonia

[2] M.A. Jabri: “Practical Performance and Credit Assignment Efficiency of Analog Multi-layer Perceptron Pertubation Based Training Algorithms”, SEDAL, Technical Report, 01.07.1994

[3] G. Geske, „Umsetzung und Analyse der analogen Komponenten eines neuronalen Klassifikators auf Basis der 0.6µm-AMS-CMOS-Technologie“, Universität Rostock, 2003, Dissertation

[4] P.R. Gray, R.G. Meyer: “Analysis and Design of Analog Integrated Circuits”, J. Wiley & Sons 1993

[5] Jacek M. Zurada. Introduction to artificial neural systems. West publishing Company, St. Paul, 1992.

[6] Frank Stüpmann: „Selbstständig lernende neuronale Struktur - ein Beitrag zur analogen Hardwarerealisierung neuronaler Netze“, Universität Rostock, 2001. Dissertation.