# Effizientes Layout von neuronalen Netzen

Robert Albin Legenstein

20. September 1999

# Vorwort

Die Theorie der künstlichen Neuronalen Netze ist in den bisherigen fast 60 Jahren ihres Bestehens zu einem wichtigen Zweig der Theoretischen Informatik geworden. Seit es diese Theorie gibt wurde auch versucht, solche lernende Maschinen zu realisieren. Der erste Neurocomputer, *Snark*, wurde bereits 1954 von Marvin Minsky entwickelt. *Snark* verwendete motorgesteuerte Potentiometer als Gewichte. Zum Glück gibt es heute etwas ausgereiftere Techniken.

Die VLSI-Technologie (Very Large Scale Integration) bietet die Möglichkeit, auch größere Netzwerke zu implementieren. Vergleicht man jedoch die Leistung solcher auf Silizium basierender Netzwerke mit der von organischen/biologischen, so muß man feststellen, daß diese unerreichbar zu sein scheinen.

Diese Arbeit setzt in einem Punkt zwischen den drei Spannungsfeldern von Theorie, technischer Realisierung und Biologie an. Mein Ausgangspunkt ist die Theorie von Schwellenschaltkreisen (threshold circuits). Diese wird auf ihre Tauglichkeit in Bezug auf VLSI diskutiert und mögliche Anpassungen der Theorie werden vorgeschlagen. An einigen Beispielen wird das Design von effizienten Schaltkreisen nach VLSI-freundlichen Kriterien demonstriert.

Die Forschungen an biologischen und künstlichen Neuronalen Netzen sind miteinander eng verknüpft. Nicht nur bietet die Biologie Inspiration für die Informatik, auch die Informatik trägt zum besseren Verständnis der biologischen Vorgänge bei. Diese Arbeit beschäftigt sich auch mit einigen biologischen Aspekten. Einige quantitativen Abschätzungen über die Anatomie der Kortex dienen als Vorlage für ein Modell, an dem hauptsächlich die Verbindungsstruktur zwischen den Neuronen untersucht wird.

## Neuerungen

Im Rahmen dieser Arbeit konnten einige neue Ergebnisse erzielt werden. Diese sollen hier kurz angeführt werden.

Ein neuer *Schaltkreis für COMPARISON* wird vorgestellt. Die Konstruktion benötigt Schwellengatter, hat variablen Fan-in und ist gut für VLSI-Implementation geeignet. Siehe dazu Abschnitt 3.2.2 auf Seite 22 bzw. Lemma 3.2.2 auf Seite 23 und Abbildung 3.3 auf Seite 23.

Ein neuer *AND/OR-Schaltkreis für COMPARISON* hat ebenfalls variablen Fan-in und gute VLSI-Eigenschaften. Zu finden ist diese Konstruktion in Abschnitt 3.2.2 auf Seite 25 (Modifikation der Konstruktion von Siu et al.(LEG)) bzw. Abbildung 3.5 auf Seite 27.

Die *Lokalität von Schaltkreisen* wird im Abschnitt 3.2.3 auf Seite 33 anhand der Kantenkreuzungen (crossings) abgeschätzt.

Ein Schaltkreis mit beschränktem Fan-in zur Berechnung von *Schwellenfunktionen* wird angegeben. Theorem 1 auf Seite 43 macht Aussagen über dessen Eigenschaften.

Selbiges wird für *c-symmetrische Funktionen*, eine Funktionsklasse die in Definition 3.3.3 auf Seite 44 definiert ist, angegeben. Die Eigenschaften des Schaltkreises sind in Theorem 2 auf Seite 45 angegeben.

In Kapitel 4 auf Seite 47 wird ein Schaltkreismodell eingeführt, das Verbindungsstrukturen untersuchen soll (*Grid-Modell*). Dabei sind vor allem beschränkte Gatterdichte sowie beschränktes Fan-in und beschränkte Leitungslänge vorgegeben.

In diesem Modell wird die ein-Layer-Architektur ($K_{n,m}$) untersucht. Siehe dazu Kapitel 4.1 auf Seite 49 bzw. Lemma 4.1.1 auf Seite 51

In [SRK95] wurde ein Schaltkreis zur effizienten *Präfixberechnung* angegeben. Lemma 3.1.2 auf Seite 19 erweitert das Ergebnis auf kleinen Fan-in $o(\log n)$.

Eine wichtige Eigenschaft der *COMPARISON-Funktion* wurde in Lemma 3.2.1 auf Seite 22 angegeben. Diese Eingenschaft ist sicher bekannt, konnte in der Literatur aber nicht gefunden werden.

## Danksagung

Vor allem möchte ich meinem Betreuer Dr. Wolfgang Maass danken, für sein großes Interesse, seinen Einsatz und seine Ideen zu dieser Arbeit. Weiters will ich allen Mitarbeitern des Instituts für Grundlagen der Informationsverarbeitung, wissenschaftlich oder nicht, für die freundliche Aufnahme in den Institutsbetrieb meinen Dank aussprechen.

Dann gibt es noch all jene die eigentlich nicht direkt etwas mit dieser Arbeit zu tun haben, aber sonst ganz nett sind. Nicht zuletzt danke ich dem Staat Österreich dafür, daß er mich die ganze Zeit ausgehalten hat.

# Inhaltsverzeichnis

# Abbildungsverzeichnis

# Kapitel 1

# Definitionen und Bemerkungen

## 1.1 Aufbau der Arbeit

Der Titel von Kapitel 1 spricht für sich selbst. Hier werden zunächst die meisten allgemein anerkannten und in dieser Arbeit benützten Modelle und Paradigmen eingeführt. Dies sind vor allem das Schaltkreismodell, Komplexitätsmaße und Gattertypen. Neben den Definitionen findet man hier aber auch einige wichtige Anmerkungen und bekannte Ergebnisse.

Im Kapitel 2 werden die Anforderungen und Limitationen von Schaltkreisen in VLSI und in biologischen Netzen besprochen. Es wird auch gezeigt welche Auswirkungen dies auf die Theorie von Schaltkreisen hat. Im Kapitel 2.2.1 wird außerdem ganz kurz der Aufbau von biologischen Schaltkreisen skizziert.

Einen wichtigen Punkt der vorangegangenen Betrachtungen behandelt Kapitel 3 genauer. Hier wird anhand einiger Beispiele gezeigt, wie man effiziente Schaltkreise mit beschränktem Fan-in erhalten kann. Die Auswirkungen eines solchen Design werden detailliert in Kapitel 3.2 anhand der Funktion COMPARISON erläutert.

Kapitel 4 ist eher biologisch motiviert und versucht bestimmte Verbindungsstrukturen an Beschränkungen anzupassen. Dazu wird ein neues Schaltkreismodell eingeführt.

Im Anhang findet man einige Beweise die aus verschiedenen Gründen nicht im Text eingebettet wurden.

## 1.2 Allgemeine Mathematik

### 1.2.1 Eine fast konstante Funktion

Eine nicht so gebräuchliche aber in dieser Arbeit öfters verwendete Funktion ist die $\log^*$ Funktion. Diese Funktion ist extrem schwach ansteigend. Aus diesem Grund wird ein $\log^*$-Faktor als "fast konstanter" Faktor angesehen. $\log^* n$ gibt an, wie oft man $n$ logarithmieren muß bis $\log\log\log\ldots\log n$ kleiner oder gleich 1 ist. Die Werte von $\log^* n$ für $n = 2, 4, 16, 65536$ sind jeweils $1, 2, 3, 4$.

**Definition 1.2.1** ($\log^*$)
Für ein $k \geq 1$ sei $\log^{(k+1)} n = \log(\log^{(k)} n)$ und $\log^{(1)} n = \log n$. Dann ist $\log^* n = min\{k : \log^{(k)} n \leq 1\}$.

□

### 1.2.2 Binäre Darstellung von Zahlen

Wir werden vor allem mit booleschen Funktionen $f : \{0,1\}^n \to 0,1^m$ arbeiten. Bestimmte Funktionen wie etwa COMPARISON oder ADDITION haben eine inherente Verbindung zu natürlichen oder ganzen Zahlen.

Wir müssen also die Zahlen binär kodieren. Gegeben sei ein boolescher Vektor $X = (x_0, \ldots, x_{n-1}) \in \{0,1\}^n$. Dieser Vektor stellt eine natürliche Zahl $x_{nat} \in \{0, \ldots, 2^n - 1\}$ dar. Diese Zahl kann folgendermaßen bestimmt werden:

$$x_{nat} = \sum_{i=0}^{n-1} x_i 2^i$$

Um die Notation zu vereinfachen wird in dieser Arbeit der Vektor $X$ mit der von ihm dargestellten Zahl identifiziert. So bedeuted etwa $X > Y$ eigentlich $x_{nat} > y_{nat}$.

Für ADDITION benötigen wir Ganzzahlen. Auf die Behandlung von negativen Zahlen wird nicht weiter eingegangen. Jedoch bilden die angegebenen Schaltkreise jedenfalls die passende Summe wenn negative Zahlen im 2er-Komplement dargestellt werden.

### 1.2.3 Graphen

Ein gerichteter Graph ist ein Tupel $(V, E)$, wobei $V$ (Knotenmenge) eine endliche Menge und $E$ (Kantenmenge) eine binäre Relation auf V ist ($E \subseteq V \times V$).

- Ein *ungerichteter Graph* $G = (V, E)$ ist ein gerichteter Graph, wobei $E$ symmetrisch ist, d.h.:$(v, w) \in E \Leftrightarrow (w, v) \in E, \forall v, w \in V$
- Ein *Pfad* von $v$ nach $w$ $(v, w \in V)$ ist eine Folge von Knoten $v = v_0, v_1, \ldots, v_k = w, v_i \in V; i = 0 \ldots k$, wobei $(v_i, v_{i+1}) \in E, i = 0 \ldots k - 1$, gilt. Ein Pfad ist einfach, wenn $v_i \neq v_j, 0 \leq i, j \leq k; i \neq j$ gilt.
- Ein *Kreis* ist ein Pfad von $v$ nach $v$. Ist dieser Pfad mit Ausnahme von $v_0 = v_k$ einfach, so ist der Kreis einfach.
- Die *Länge eines Pfades (Kreises)* ist die Anzahl seiner Kanten ($k$).
- Ein Graph heißt *azyklisch* wenn er keine nichttrivialen Kreise enthält.
- Ein Graph heißt *zusammenhängend*, wenn es für jedes Paar $(v, w)$ mit $v, w \in V$ einen Pfad in $V$ gibt.
- Ein azyklischer, ungerichteter, zusammenhängender Graph heißt *Baum*.
- Die In-Valenz (Out-Valenz) eines Knotens $v$ ist die Anzahl von Kanten, die in $v$ enden (beginnen)

$$\begin{aligned} \text{In-Valenz}(v) &:= |\{w | (w, v) \in E\}| \\ \text{Out-Valenz}(v) &:= |\{w | (v, w) \in E\}| \end{aligned}$$

Vergleiche hierzu Kapitel 9 in [HA96].

## 1.3 Schaltkreise

Der Großteil dieser Arbeit wird sich mit feedforward Netzwerken auseinandersetzen. Besonders Kapitel 3 baut auf dieses Konzept auf. Die folgende Definition gibt eine Beschreibung des feedforward Konzeptes an.

**Definition 1.3.1 (Feedforward Netzwerk)**
Ein feedforward Netzwerk ist ein gerichteter *azyklischer* Graph. Die Knoten des Graphen können in drei Mengen unterteilt werden:

1. *Inputknoten* - haben keine eintretenden Kanten

2. *interne Knoten* - haben sowohl ein- als auch austretende Kanten

3. *Outputknoten* - haben keine austretenden Kanten

Ein Netzwerk mit $n$ Inputknoten und $m$ Outputknoten berechnet eine $m$-stellige Funktion $f(x_1, \ldots, x_n)$. Dabei werden den Inputknoten die Werte $x_1, \ldots, x_n$ zugeordnet. Jeder interne Knoten und jeder Outputknoten $g_i$(diese Knoten werden *Gatter* genannt) berechnet eine bestimmte Funktion seiner Inputs. Diese Inputs sind die Funktionswerte aller Gatter $g_j$ für die gilt: $(j \to i)$ ist eine Kante des Netzwerkes. Der Betrag der Netzwerkfunktion $f(x_1, \ldots, x_n)$ ist dann der Vektor der Funktionswerte der Outputknoten.

□

Statt des Begriffes *"feedforward Netzwerk"* kann man auch *"Schaltkreis"* verwenden. Meist will man eine bestimmte Funktion für beliebig viele Inputs berechnen. Ein Schaltkreis $C_n$ berechnet die Funktion aber immer nur für konstant viele (nämlich $n$) Inputs. Kann man Schaltkreise für beliebige $n$ konstruieren, so spricht man von einer *Schaltkreisfamilie* $\{C_n\}$. Man hat also eine Folge von Funktionen $\{f_n\}$ und eine Schaltkreisfamilie $\{C_n\}$ sodaß jede Funktion $f_n$ durch den Schaltkreis $C_n$ in $\{C_n\}$ berechnet wird. In dieser Arbeit wird "Schaltkreis" mit "Schaltkreisfamilie" gleichgesetzt, um umständliche Notation zu vermeiden.

**Definition 1.3.2 (Boolescher Schaltkreis)**
Ein boolescher Schaltkreis ist ein feedforward Netzwerk dessen Inputs und Outputs binär sind.

□

In dieser Arbeit werden Schaltkreise oft graphisch dargestellt. Dabei werden folgende Konventionen verwendet:

- Der Informationsfluß erfolgt meist von oben nach unten. Um eine Überladung der Darstellungen zu vermeiden wurden deshalb Pfeile an den Kanten vermieden, sofern die Richtung eindeutig ist.

- Die Inputknoten sind in der obersten Reihe angeordnet

- Kreise symbolisieren Gatter. Meist sind dies Schwellengatter. AND bzw. OR-Gatter sind durch $\wedge$ bzw. $\vee$ angedeutet.

- Eingebettete Schaltkreise sind durch Rechtecke angedeutet.
- Die Outputknoten sind die untersten Knoten

Die nächste Definition führt die gängigen Komplexitätsmaße von Schaltkreisen ein.

**Definition 1.3.3 (size / depth)**
Die *size* eines Schaltkreises ist die Anzahl der Gatter im Schaltkreis. Die *depth* eines Gatters ist die maximale Anzahl von Kanten aller möglichen gerichteten Pfade von einem Inputknoten zu diesem Gatter. Die *depth* eines Schaltkreises ist das Maximum der depth aller Gatter.

□

Statt *size* und *depth* kann man auch *Größe* und *Tiefe* verwenden. Die englischen Begriffe sind aber schon so gebräuchlich daß es klüger erscheint, dieselben zu verwenden, auch wenn das einige Probleme mit der deutschen Grammatik ergibt. Ich werde deutsche und englische Begriffe je nach Bedarf verwenden, bei Formeln aber ausnahmslos auf englische zurückgreifen. Dies gilt auch für andere Begriffe wie "*Fan-in*" für die es kein passendes deutsches Wort gibt.
Die folgende Definition wird vor allem im Kapitel 3 von Bedeutung sein.

**Definition 1.3.4 (Fan-in / Fan-out)**
Der *Fan-in* eines Gatters ist die Anzahl der eintretenden Kanten (In-Valenz).
Der *Fan-out* eines Knotens ist die Anzahl der austretenden Kanten (Out-Valenz). Der maximale Fan-in bzw. Fan-out über alle Knoten im Schaltkreis bestimmt den Fan-in bzw. Fan-out des Schaltkreises.

□

Ein Schaltkreis wird als *geschichtet (layerd)* bezeichnet wenn jedes Gatter mit Tiefe $d$ seine Inputs nur von Knoten der Tiefe $d-1$ bezieht. In diesem Fall bezeichnet man die Menge aller Gatter der Tiefe $d$ als *Schicht d (layer d)*. Nicht geschichtete Schaltkreise lassen sich einfach in geschichtete überführen indem man Knoten einführt die Funktionswerte weiterleiten.

## 1.4 Schwellengatter

Bis jetzt wurde noch nicht angegeben welche Funktionen den einzelnen Gattern zugeordnet sind. Der wichtigste Gattertyp in dieser Arbeit wird das

*Schwellengatter* sein. Aber auch die traditionellen AND/OR Gatter werden benötigt. Bevor wir uns mit Schwellengatter beschäftigen, noch die Definition von AND-OR Schaltkreisen.

**Definition 1.4.1 (AND/OR Schaltkreis)**
Ein AND/OR Schaltkreis ist ein boolescher Schaltkreis in dem jedes Gatter ein AND Gatter, OR Gatter oder ein NOT Gatter ist.

□

Schwellengatter (TGatter) werden in der englischen Literatur als "threshold gates" bezeichnet. Ein Schwellengatter berechnet auf seinen Inputs eine *lineare Schwellenfunktion* (oder linear threshold function).

**Definition 1.4.2 (lineare Schwellenfunktion)**
Eine lineare Schwellenfunktion $f(X)$ ist eine binäre Funktion sodaß

$$f(x_1, \ldots, x_n) = \begin{cases} 1 & \text{falls} \qquad \sum_{i=1}^{n} w_i x_i \geq \Theta \\ 0 & \text{sonst} \end{cases}$$

□

Die Koeffizienten $w_i$ werden *Gewichte* oder *weights* genannt, $\Theta$ wird allgemein als *Schwelle* oder *threshold* bezeichnet.

**Definition 1.4.3 (Schwellenschaltkreis)**
Ein Schwellenschaltkreis ist ein feedforward Netzwerk in dem jedes Gatter eine lineare Schwellenfunktion berechnet.

□

In einem Schwellenschaltkreis werden die Gewichte oft global betrachtet, sodaß $w_{i,j}$ ein Gewicht ist das im Gatter $g_i$ den Output des Gatters $g_j$ gewichtet.

### Gewichte in Schwellenschaltkreisen

Die Gewichte der Schwellengatter spielen eine bedeutende Rolle, da sie die Berechnungskraft der einzelnen Gatter bestimmen. Die allgemeine Definition der linearen Schwellenfunktion erlaubt reelle Gewichte. Man kann aber zeigen, daß für binäre Inputs auf $2^{O(n \log n)}$ beschränkte ganzzahlige Gewichte ausreichen (das Ergebnis findet man in vielen Publikationen, z.B. in [GsR]). Schwellenschaltkreise werden außerdem unterteilt in Schaltkreise mit polynomiellen Gewichten (die Gewichte sind beschränkt durch $O(n^c)$ für ein konstantes $c$) und in Schaltkreise mit exponentiellen Gewichten (d.h. es gibt Gewichte die nach unten durch $\Omega(2^{n^\epsilon})$ beschränkt sind für ein $\epsilon > 0$). Man kann etwa COMPARISON (wird im Kapitel 3.2 behandelt) durch ein Gatter mit exponentiellen Gewichten berechnen, nicht aber durch ein Gatter mit polynomiellen Gewichten(siehe [SRK95] Kapitel 6.2).

## 1.5 PL-Gatter

Ein weiterer benötigter Gattertyp ist das sogenannte *PL-Gatter*. Das PI-Gatter berechnet ähnlich wie das Schwellengatter eine gewichtete Summe. Es ist also ein lineares Gatter. Die Aktivierungsfunktion unterscheidet sich allerdings von der des Schwellengatters. In einem bestimmten Bereich wird die Summe linear an den Output weitergegeben. Wird allerdings dieser Bereich überschritten, so wird die Summe "abgeschnitten". Die Aktivierungsfunktion ist also stückweise linear (piecewise linear *PL*). Sie ist somit nicht binär. Die Tatsache daß die Summe weitergegeben werden kann, ermöglicht eine einfache Kaskadierung und erleichtert die Arbeit bei beschränkten Fan-in.

**Definition 1.5.1 (PL-Gatter)**
Sei $S(x) = \sum_{i=1}^{n} w_i x_i + \Theta$. Ein PL-Gatter berechnet auf $n$ Inputs $x = (x_1, \ldots, x_n)$ die Funktion $f : \mathrm{R}^n \rightarrow [-\alpha, +\alpha]$

$$f(x_1, \ldots, x_n) = \begin{cases} S(x) & \text{falls} \quad |S(x)| \leq \alpha \\ \alpha & \text{falls} \quad S(x) > \alpha \\ -\alpha & \text{falls} \quad S(x) < -\alpha \end{cases}$$

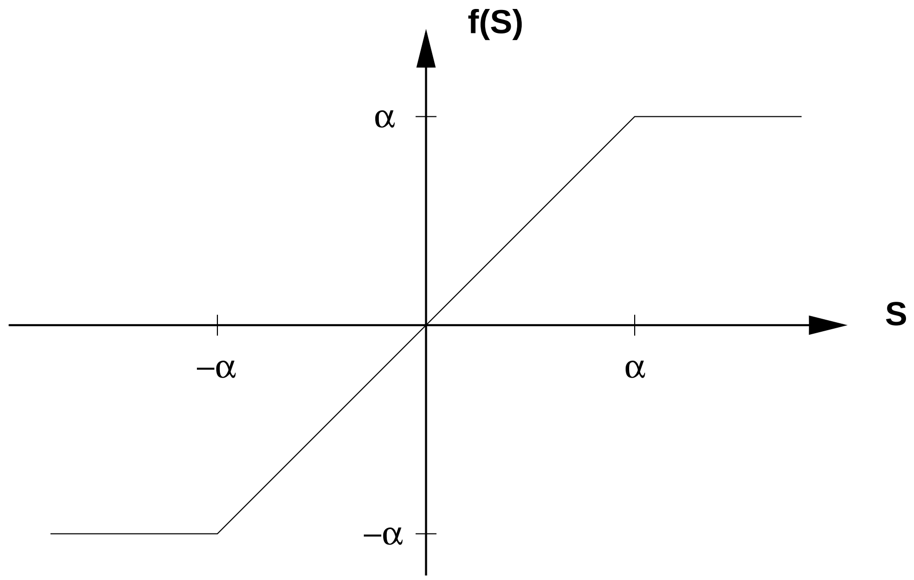


Abbildung 1.1: Aktivierungsfunktion des PL-Gatters

# Kapitel 2

# Biologische und technische Anforderungen

Die theoretische Arbeit mit Schaltkreisen hat neben rein theoretischen Fragen auch praktische Aspekte. Zum Einen will man natürlich Fortschritte in Bezug auf VLSI-Implementation von Schaltkreisen erzielen. Als Beispiel seien etwa Additions-Schaltkreise angeführt. Schnelle Additions-Schaltkreise sind für gute Rechenleistungen in CPUs unerlässlich. Andere Beispiele wären etwa die Paritätsberechnung für digitale Datenübertragung oder die Fast Fourier Transformation für eine Unzahl von technischen Anwendungen. Gerade in solchen zeitkritischen Anwendungen ist extreme Parallelität wichtig. Beim Schaltkreisentwurf darf in solchen Fällen aber natürlich nicht auf die Anforderungen der Implementation vergessen werden, denn diese sind in der Theorie meist nicht berücksichtigt.

Zum Anderen versucht man mithilfe künstlicher Neuronaler Netze das Verhalten von biologischen Schaltkreisen zu simulieren oder zu erklären. Mit der Computational Neuroscience ist ein eigener Forschungszweig entstanden, der dem Geheimnis der unglaublichen Leistungen des Gehirns auf die Spur kommen will.

Bei der VLSI-Implementation von Neuronalen Netzen stößt man allerdings, verglichen mit biologischen Netzen, sehr bald an Grenzen, die wahrscheinlich in der starken Vernetzung der Neuronen begründet sind.

Im Folgenden will ich kurz auf die technischen Anforderungen an Schaltkreise eingehen.

## 2.1 VLSI-Implementationen von Neuronalen Netzen

Die Hauptbeschränkungen von VLSI entstehen aus der 2-Dimensionalen Abbildung der Rechenelemente und vor allem deren Vernetzung in einer beschränkte Anzahl von Layern. Dies führt zu *beschränkter Vernetzung* und *beschränkter Präzision* der Gewichte. Hohe Präzision benötigt mehr Platz, einerseits um die Gewichte zu speichern, und andererseits um das Ergebnis zu berechnen . Biologische Netze haben wenige Layers, zumindest bestimmte Teile der Kortex. So kann man etwa eine bekannte Person in einem verrauschten Bild innerhalb weniger 100 msek. erkennen, wobei ein Neuron eine Verzögerung von etwa 10 msek. hervorruft. Aufgrund der beschränkten Konnektivität muß man in VLSI tiefere Netze mit schnelleren Einheiten bevorzugen.

### 2.1.1 Schaltkreise und Area

Bei VLSI-Implementationen in Silizium hat man nur beschränkten Platz (*area*) zur Verfügung. Eine Abschätzung des Platzverbrauches durch die Anzahl der Gatter (In der Theorie wird meist diese *size* des Netzwerkes abgeschätzt) reicht hier keinesfalls aus. Dies resultiert aus mehreren Überlegungen. Einerseits ist der Platzverbrauch der Verbindungen (*wires*) sehr wichtig (*"the area of the connections counts"* [Bei98]), andererseits hängt der Platzverbrauch eines Neurons von dessen Gewichten ab (*"comparing the number of nodes is inadequate for comparing the complexity of NNs as the nodes themselves could implement quite complex functions"* [Wil90]).

Bei solchen Überlegungen spielt der Fan-in eine bedeutende Rolle. Durch beschränkten Fan-in kann man sowohl die Gewichte als auch die Konnektivität beschränken. Beiu nimmt an daß man sowohl Präzision als auch Fan-in für praktische digitale Implementationen (sub-)logarithmisch beschränken muß [Bei96]. Man kann den Fan-in sogar direkt in Verbindung mit der Konnektivität eines Netzwerkes setzen: *the "area required for inter-node connectivity grows like the cube of a node's fan-in"* [Ham88]. Des weiteren gibt es sowohl für analoge als auch für digitale Implementationen technologische Schranken an den maximalen Fan-in und der Präzision der Gewichte. In verschiedenen Arbeiten wurden bereits einige Ansätze gewählt, die Komplexität von Netzen alternativ abzuschätzen:

- Die Anzahl der Verbindungen(*edges*) soll die Konnektivität des Netzwerkes erfassen.

- Die Anzahl der benötigten Bits um alle Gewichte und Schwellen des Netzwerkes darzustellen. Dieses Maß stellt auch eine Verbindung zur minimum description length und der Entropie von Datenmengen her.
- Die Summe der Gewichte und Schwellen des Netzwerkes (*weightsum*).

Das Beschränken das Fan-in hat aber auch noch andere Effekte. Ein Schaltkreis mit Fan-in $k$ und Größe $s$ kann von maximal $s(k-1)+1$ Inputs abhängen, und bei Tiefe $d$ maximal von $k^d$ Inputs. Daher benötigt man bei konstantem Fan-in $k$ mindestens $(n-1)/(k-1) = \Omega(n)$ Gatter und eine Tiefe von mindestens $d = \log n / \log k = \Omega(\log n)$.

Weiters ist bekannt daß ein Schwellengatter mit Fan-in $k$ maximal $2^{k^2}$ Funktionen berechnen kann. Daher hat man für sehr kleines konstantes $k$ viel Berechnungskraft gegenüber AND-OR Schaltkreisen eingebüßt. Schwellenschaltkreise mit beschränktem Fan-in $k$ können immer auch durch AND-OR Schaltkreise implementiert werden, wobei sich Tiefe und Größe des Schaltkreises um einen konstanten Faktor (abhängig von $k$) erhöhen.

### 2.1.2 Schaltkreise und Delay

Die Zeit die ein Schaltkreis benötigt um eine Funktion zu berechnen (*delay*) ist natürlich ein wichtiger Faktor im Schaltkreisentwurf. Man geht bei VLSI-Modellen meist davon aus, daß die Berechnung getaktet ist. Die Zeit wird also in diskrete Intervalle unterteilt. Innerhalb eines Intervalls haben die Gatter Zeit um das Ergebnis bereitzustellen. Außerdem muß in dieser Zeitspanne das Ergebnis über die Leitungen zu den Nachfolgegattern geleitet werden.

In der Theorie wird meist nur die Tiefe des Schaltkreises zur Bestimmung der Berechnungszeit herangezogen. Dies ist korrekt, wenn die Schaltzeit der Elemente und die Verzögerung durch die Leitungen unabhängig von der Inputgröße ist. Dies ist in VLSI natürlich nicht der Fall [Ull84]. Das delay von Leitungsbahnen ist abhängig vom Produkt der Kapazität und des Widerstandes, welche wiederum beide von der Leitungslänge abhängen. Somit steigt das delay der Leitungen quadratisch mit deren Länge.

Bezieht man die Kapazität und den Widerstand auf die Fläche, so haben die Gatter (Transistoren) bedeutend höhere Werte. Es hängt von der Länge der Leitungen ab, wovon das delay dominiert wird. Bei kurzen Leitungen aber wirkt sich vor allem die Eingangskapazität der Gatter aus, welche linear vom Fan-in abhängt.

Die Verzögerung durch die Leitungen können durch verschiedene Maßnahmen verbessert werden. Lange Leitungen können in Metall geführt werden (z.B. bei NMOS), wodurch man wesentlich bessere Leitungseigenschaften erhält. Das ergibt eine ähnliche Strategie (wenn auch nicht so drastisch) wie jene im Kortex. Dort gibt es viele lokale Verbindungen zwischen "benachbarten" Neuronen und wenige sehr lange, welche in der weißen Masse geführt werden.

Modelle die solche Parameter berücksichtigen wurden bereits verwendet:

- Ein Modell nimmt an daß die Berechnungszeit eines Gatters von dessen Eingangskapazität bestimmt ist, sodaß jedes Gatter ein *delay* proportional zu dessen Fan-in aufweist (*FIdelay*).

- Ein anderes Modell geht von der Dominanz der Leitungskapazitäten aus. Damit ist das *delay* bestimmt durch die Leitungslänge zwischen den Gattern.

## 2.2 Biologische Neuronale Netze

Die Daten dieses Kapitels stammen hauptsächlich aus dem Buch "Anatomy of the Cortex" von Braitenberg und Schüz [BS91]. Darin wurden Statistische Untersuchungen am Kortex von weißen Mäusen publiziert. Die Ergebnisse lassen sich größtenteils auf andere Wirbeltiere und den Menschen übertragen.

### 2.2.1 Kurze Beschreibung der Struktur

Die informationsverarbeitenden Elemente im Nervensystem sind die *Neuronen* . Die Neuronen bestehen aus dem *Dendritenbaum*, der den Großteil der Information aufnimmt , dem *Zellkern*, der Berechnungen ausführt, und dem *Axon*, das die berechnete Information weiterleitet. Zwei Neuronen sind durch *Synapsen* verbunden. Diese Synapsen sind intelligente Verbindungen, die für das Lernen verantwortlich sind. Synapsen treten meist zwischen dem Axon des präsynaptischen Neurons (liefert Information) und dem Dendritenbaum des postsynaptischen Neurons (empfängt Information) auf.

Eine Vereinfachte Sichtweise der Ereignisse zwischen zwei Neuronen sieht so aus: Ein Neuron wird *aktiviert*, es feuert und gibt ein gewisses Aktionspotential ab. Das Axon leitet dieses Aktionspotential zu den Synapsen. Diese geben die Information gewichtet weiter an den Dendritenbaum des postsynaptischen Neurons, der leitet es weiter an den Zellkern. Der Zellkern wird aktiviert wenn das durch viele solcher Synapsen erreichte Potential eine gewisse Schwelle überschreitet.

Man sieht sofort die Übereinstimmungen mit dem Modell des Schwellenschaltkreises. Der Funktionswert eines Schwellengatters sagt uns ob dieses "aktiviert" wurde. Der Dendritenbaum wird auf eintretende Kanten projiziert, die Gewichtung durch die Synapsen wird durch die Gewichte des Neurons bewerkstelligt und austretende Kanten entsprechen dem Axon.

Es soll gesagt werden daß dies nur eine extrem vereinfachte Darstellung der Vorgänge in Nervenzellen ist.

### 2.2.2 Quantitative Abschätzungen

Im Kortex der Maus hat man im Durchschnitt etwa $9.2 \times 10^4$ Neuronen pro $mm^3$ ([BS91] Kap. 4). Dieser Wert variiert über verschiedene Tiere und Gehirnareale. Die totale Anzahl von Neuronen im Gehirn des Menschen wird auf etwa $10^{10}$ geschätzt (1000 mal mehr als bei der Maus).

Im Durchschnitt hat man etwa $7.2 \times 10^8$ Synapsen pro $mm^3$ ([BS91] Kap. 5). Einige Korrekturen auf Grund der Messung führen zu einem Wert von etwa 8000 Synapsen pro Neuron. Auch dieser Wert ist bei verschiedenen Tieren und Arealen verschieden und liegt zwischen 2000 und 10000. Diesen Wert könnte man mit dem Fan-in von Gattern gleichsetzen. Treten allerdings mehrere Synapsen zwischen zwei Neuronen auf, so stimmt diese Rechnung nicht mehr. Nach Braitenberg und Schüz ist die Wahrscheinlichkeit von multiplen Verbindungen im Kortex allerdings als gering einzuschätzen. Wie dem auch sei, die obere Schranke an den Fan-in kann man durchaus mit $10^4$ angeben. Aus verschiedenen Gründen wird der Fan-out mit nur etwa einem Zehntel des Fan-in beziffert, wäre also $10^3$. Beim Menschen kommt man auf eine Gesamtzahl von etwa $10^{14}$ Synapsen im Gehirn.

Allein an diesen Zahlen sieht man daß eine Verbindungsstruktur "jeder mit jedem", wie dies bei Hopfieldnetzen angenommen wird, unmöglich zu realisieren ist. Jedenfalls nicht mit vielen Neuronen und in der einfachen Architektur eines vollständigen Graphen.

Es gibt zwei verschiedene Arten von Verbindungen (Axone). Zum Einen gibt es kurze lokale Verbindungen mit wenigen Millimetern Reichweite. Dazu gibt es noch lange Verbindungen, die aus der grauen Masse (hier befinden sich die Neuronen) austreten um in der weißen Masse weite Strecken zurückzulegen. Von diesen langen Verbindungen gibt es relativ wenige. Man schätzt 20% extrakortikale Verbindungen in den oberen Layers von primären sensorischen Arealen der Kortex. In anderen Areas und Layers sind es wahrscheinlich weit weniger ([BS91] Kap. 8).

Betrachtet man die Summe der Reichweiten von Dendritenbaum und Axon so kann man die Reichweite lokaler Verbindungen auf $5mm$ beschränken.

### 2.2.3 Excitatorische und inhibitorische Synapsen

Von den Synapsen selbst gibt es zwei Arten. Die sogenannten *excitatorischen* Synapsen regen das postsynaptische Neuron an, dies wird im Modell durch positive Gewichte ausgedrückt. Im Gegensatz dazu wirken die *inhibitorischen* Synapsen hemmend und im Modell werden negative Gewichte verwendet.

Diese zwei Arten kommen allerdings keineswegs zu gleichen Teilen vor. Inhibitorische Synapsen treten gerne an Zellkörpern auf (d.h. sie wirken nicht über den Dendritenbaum) und machen nur etwa 10% aller Synapsen aus ([BS91] Kap. 12). Dazu kommt noch daß die sogenannten Pyramidenzellen (Neuronen werden nach äußeren Gesichtspunkten unterteilt. Im Kortex treten vor allem Pyramidenzellen, Sternzellen und Martinottizellen auf) nur excitatorische Synapsen zu postsynaptischen Neuronen bilden, ihre Aktivierung also nur excitatorisch wirkt ([BS91] Kap. 15). Da die überwiegende Mehrheit der Neuronen im Kortex Pyramidenzellen sind, haben also die Mehrheit der Rechenelemente nur excitatorische Wirkung (sprich positive Gewichte zu den Nachfolgern). Des weiteren nimmt man an daß nur excitatorische Synapsen lernfähig sind.

### 2.2.4 Zusammenfassung

Ein biologisch motiviertes Rechenmodell muß somit einige Einschränkungen beachten. Die grundsätzlichsten sind in der folgenden Liste zusammengefaßt:

- beschränkte Neuronendichte (im Kortex $\leq 10^5/mm^3$)
- beschränktes Fan-in ($\leq 8000$)
- beschränkte Verbindungslänge (im Kortex $\leq 5mm$)

Schaltkreise die solchen Bedingungen genügen werden im Kapitel 4 betrachtet. Weitere Einschränkungen und Erweiterungen könnten so aussehen:

- wenige Verbindungen ohne Beschränkung der Länge
- nur etwa 10% der Gewichte sind negativ
- viele Neuronen haben nur positive *austretende* Kanten

Diese Annahmen wirken sich nur auf die geometrische Struktur des Netzwerkes aus. Es wird hier nicht auf die Qualität der Neuronen selbst eingegangen. Dazu gibt es eine ganze Anzahl von möglichen Ansatzpunkten auf die in dieser Arbeit nicht weiter eingegangen wird. Solche Ansatzpunkte wären etwa probabilistische Synapsen und damit verbunden Populationskodierung, spiking neurons oder dynamische Synapsen.

# Kapitel 3

# Schaltkreise mit beschränktem Fan-in

## 3.1 Parallele Präfixberechnung bei beschränktem Fan-in

Die Berechnung von Präfixprodukten wurde in Bezug auf Addition von Chandra, Fortune und Lipton angewandt [CFL85]. Diese Ideen sind auch in ([SRK95], Kapitel 5.4) zu finden. Ebenfalls für Addition, aber mit logarithmisch beschränktem Fan-in. In diesem Kapitel werden diese Ergebnisse auf konstant beschränkten Fan-in verallgemeinert und später im Kapitel 3.2.2 auf Comparison angewendet.

Man betrachte folgendes Problem. Für einen Vektor $X = (x_0, \ldots, x_{n-1})$ sollen die Werte von $x_0, x_0 \wedge x_1, x_0 \wedge x_1 \wedge x_2, x_0 \wedge x_1 \wedge x_2 \wedge x_3, \ldots$ berechnet werden. Eine solche Berechnung nennt man eine Präfixberechnung.

Allgemeine Definition: Sei $A$ eine Menge von Elementen mit einer assoziativen binären Operation $\odot$ (z.B. genannte AND-Operation, aber auch OR oder Parität). Man habe Gatter die diese Operation auf den Inputvariablen berechnen können. Bei $n$ Inputs $x_0, \ldots, x_{n-1}$ soll ein effizienter Schaltkreis alle Präfixprodukte $x_0, x_0 \odot x_1, \ldots, x_0 \odot x_1 \odot \cdots \odot x_{n-1}$ berechnen.

Dazu reicht trivialerweise ein Layer von $n$ Gattern und $O(n^2)$ Kanten. Die Anzahl der Kanten kann aber noch stark verbessert werden.

**Lemma 3.1.1 (Variation von Lemma 5.3 in [SRK95])** Eine Präfixberechnung auf $n$ Variablen kann durch einen Schaltkreis der Tiefe $depth = O(\log n / \log \Delta)$ mit $size = O((n/\Delta) \log n)$, $edges = O(n \log n)$ und Fan-in

$2 \leq \Delta \leq \log n$ durchgeführt werden. [1]

$\square$

Für die Berechnung der Präfixe der Funktion $f$ wird $f$ zuerst auf bestimmten Teilmengen der Inputmenge berechnet. Die Teilmengen (*Intervalle*) haben die Größen 2,4,8,16 usw. Die Struktur der Intervalle ist auf Bild 3.1 auf der nächsten Seite für 16 Inputs zu sehen. Der dargestellte Baum dient aber nur dem besseren Verständnis. Tatsächlich werden alle Intervalle parallel berechnet. Für ein beliebiges Präfix benötigt man maximal ein Intervall aus jeder Schicht das Baumes.

**Beweis:** Man betrachte die Indizes der Inputs als Menge $\{0, \ldots, n-1\}$. Ein *Intervall* $I$ ist eine geordnete Untermenge von aufeinanderfolgenden Indizes dieser Menge, also $I = (i, i+1, \ldots, j)$. Die *Hauptintervalle* der Indexmenge kann man folgendermaßen ermitteln. Man baut einen binären Baum mit den InputIndizes als Blätter (geordnet von links nach rechts). Die Hauptintervalle sind dann die inneren Knoten des Baumes und ein Hauptintervall enthält alle Indizes seiner Vorgängerknoten. Dann gibt es $O(n)$ Hauptintervalle mit $O(\log n)$ verschieden Größen $n, n/2, n/4, \ldots, 1$. Um aus den Hauptintervallen ein Intervall zu bilden das ein beliebiges Präfix der Indizes umfaßt benötigt man von allen Hauptintervallen einer Größe maximal eines, also $O(\log n)$ insgesamt.

Im ersten Layer des Schaltkreises werden diese Hauptintervalle berechnet. Dazu benötigt man $O(n)$ Gatter (vorläufig mit unbeschränktem Fan-in). Für die Anzahl der Kanten ist die Größe des Intervalls wichtig. Wir summieren die Kanten auf. Dabei wird für jede Intervallgröße die Anzahl der Kanten für ein Intervall mit der Anzahl der Intervalle dieser Größe multipliziert.

$$edges = \sum_{i=0}^{\log n} O(2^i) O(\frac{n}{2^i}) = O(n \log n).$$

Im zweiten Layer werden diese Hauptintervalle zu den Präfixes zusammengesetzt. Dazu benötigt man ebenfalls $O(n)$ Gatter mit $O(n \log n)$ Kanten.

Nun soll der Fan-in der Gatter beschränkt werden. Nachdem die Funktion die ein Gatter berechnet assoziativ ist, kann sie auch durch einen $\Delta$-Baum dieser Gatter berechnet werden. Dadurch erhöht sich die Tiefe auf

[1] Bei $\Delta > \log n$ gilt: $size = O(n)$

$O(\log n/\log\Delta)$. Die Anzahl der Kanten bleibt in der gleichen Ordnung. Für die Anzahl der Gatter gilt im ersten Layer:

$$size = \sum_{i=0}^{\log n} O(2^i) O(\frac{n}{\Delta 2^i}) = O(\frac{n}{\Delta}\log n)$$

Dies gilt aber nur für $\Delta = O(\log n)$, denn es müssen hier jedenfalls $n$ Intervalle betrachtet werden, sodaß bei größerem $\Delta$ gilt: $size = O(n)$.
Da die Gatter des zweiten Layers (berechnen die Präfixe) nur Fan-in $\log n$ haben, wird hier Gatter- und Kantenanzahl in der Ordnung nicht weiter beeinflußt.

□

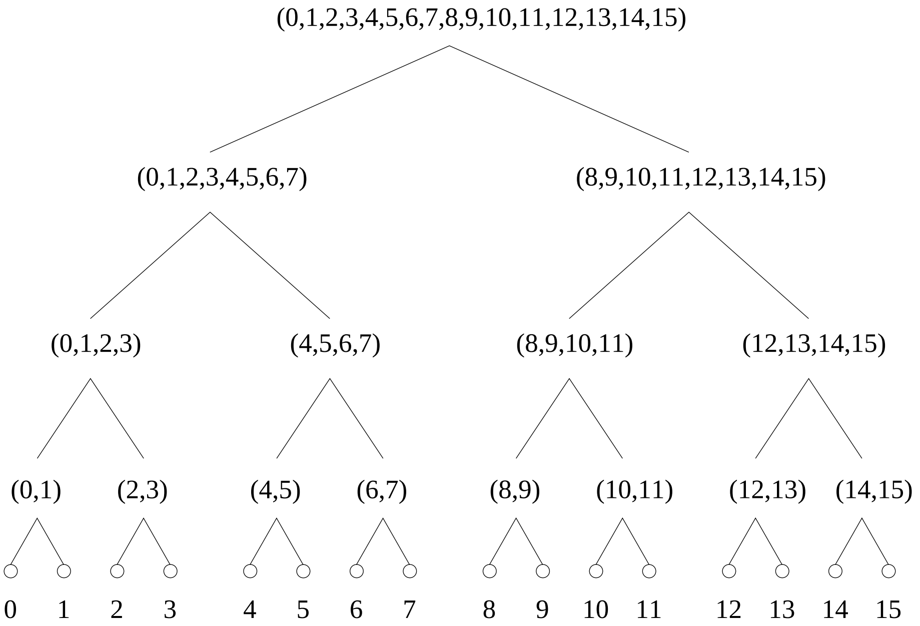


Abbildung 3.1: Hauptintervalle für $n = 16$

Man kann die Kantenkomplexität aber noch weiter verbessern, und zwar auf "fast linear". Die erzielte Komplexität ist dann $n \log^* n$, wobei $\log^*$ eine extrem schwach ansteigende Funktion ist (siehe dazu Definition 1.2.1 auf Seite 2).

**Lemma 3.1.2 (Variation von Lemma 5.4 in [SRK95])** Eine Präfixberechnung auf $n$ Variablen kann durch einen Schaltkreis der Tiefe $depth = O(\log n / \log \Delta)$ mit $size = O((n/\Delta) \log^* n)$, $edges = O(n \log^* n)$ und Fan-in $2 \leq \Delta \leq \log^* n$ durchgeführt werden. [2]

Auch dieses Konstruktion arbeitet mit Intervallen, diesmal aber mit logarithmisch großen. Die Intervalle für $n = 16$ sind in Bild 3.2 zu sehen.

**Beweis:** Dabei geht man zunächst ähnlich vor wie beim Beweis von Lemma 3.1.1. Die Intervallgrößen werden jetzt aber logarithmisch als $\log n, \log\log n, \log\log\log n, \ldots$ gewählt. Es gibt also $\log^* n$ verschiedene Intervallgrößen. Im ersten Schritt wird auf diese Intervalle die assoziative Funktion berechnet. Dazu benötigt man $O((n/\Delta) \log^* n)$ viele Gatter, $O(n \log^* n)$ Kanten und eine Tiefe von maximal $(\log n / \log \Delta)$. Damit kann man die Präfixprodukte aber noch nicht effizient berechnen.

Stellt man diese Intervalle wiederum als logarithmischen Baum dar, sieht man daß ein Intervall $I$ in logarithmisch viele ($= k$) Teilintervalle $I_1, I_2, \ldots, I_k$ unterteilt wird. Von diesen wird eine Präfixberechnung $I_1, I_1 \odot I_2, I_1 \odot I_2 \odot I_3, \ldots$ durchgeführt.

Ein Beispiel: Das Intervall $(0, \ldots, 15)$ ist unterteilt in Intervalle $(0,1,2,3)$, $(4,5,6,7)$, $(8,9,10,11)$, und $(12,13,14,15)$. Dann wird $(0,1,2,3)$, $(0,1,2,3) \odot (4,5,6,7)$, $(0,1,2,3) \odot (4,5,6,7) \odot (8,9,10,11)$ usw. berechnet.

Dazu kann man die Schaltkreise aus Lemma 3.1.1 verwenden. Für ein Intervall $I$ der Größe $s$ benötigt man dann $O((s/\Delta \log s) \log s) = O(s/\Delta)$ Gatter und $O((s/\log s) \log s) = O(s)$ Kanten bei einer Tiefe von $O(\log s / \log \Delta)$. Die Summe über alle Intervalle ergibt eine Gatteranzahl von $O((n/\Delta) \log^* n)$, $O((n \log^* n)$ Kanten und eine Tiefe von maximal $(\log n / \log \Delta)$.

Da jetzt jedes Präfixprodukt eine Verknüpfung von maximal $(\log^* n)$ Intervallen der letzten Schicht ist, bleibt auch der letzte Schritt der Berechnung in den gegebenen Grenzen.

$\square$

Mit dieser Konstruktion hat man ein Kanten und Fan-in-effektives Werkzeug zur Berechnung von Präfixprodukten in Schaltkreisen. Viele boolesche Funktionen wie etwa *Addition* oder *Comparison* (siehe dazu Abschnitt 3.2.2 auf Seite 25) können dadurch effizient implementiert werden.

[2] Bei $\Delta > \log^* n$ gilt: $size = O(n)$

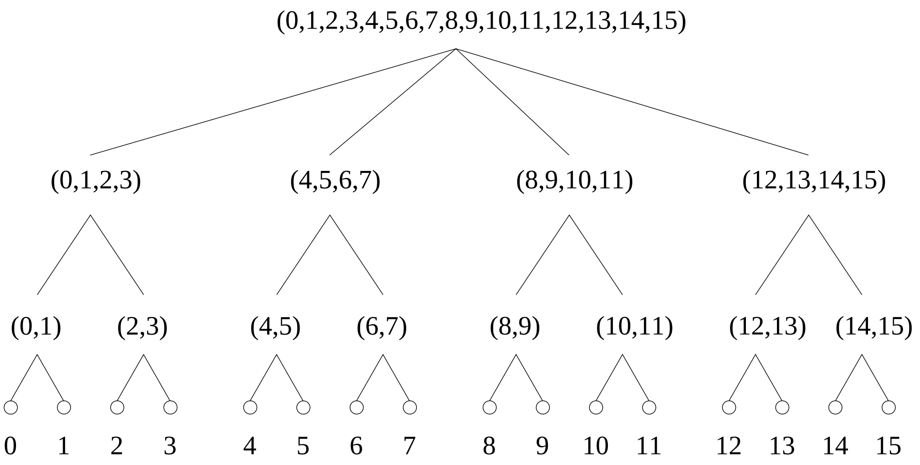


Abbildung 3.2: Logarithmische Intervalle für $n = 16$

## 3.2 Fallbeispiel COMPARISON

Dieser Abschnitt beschäftigt sich mit verschiedenen Schaltkreisen zur Berechnung von COMPARISON (Definition siehe 3.2.1). Diese werden vor allem in Hinblick auf beschränkten Fan-in und diverse VLSI-freundliche Komplexitätsmaße untersucht.
Dabei stütze ich mich vor allem auf Arbeiten von Valeriu Beiu (siehe dazu [Bei98]). Ich habe versucht seine Arbeiten zu ergänzen und erweitern.
Im Abschnitt 3.2.2 auf der nächsten Seite werde ich einige Schaltkreise für COMPARISON anführen und analysieren. Darauf folgt eine Diskussion der Eigenschaften in Hinblick auf VLSI-Komplexität im Abschnitt 3.2.3 auf Seite 28.

### 3.2.1 Definitionen und Bemerkungen

Die COMPARISON-Funktion entspricht dem Vergleich zweier natürlicher Zahlen (siehe dazu auch Kapitel 1.2.2).

**Definition 3.2.1 (COMPARISON)**
Sei $X = (x_0, \dots, x_{n-1}) \in \{0,1\}^n$ und $Y = (y_0, \dots, y_{n-1}) \in \{0,1\}^n$.

$$C_n : \{0,1\}^{2n} \rightarrow \{0,1\}$$
$$C_n = C_n(X,Y) = \begin{cases} 1 & \text{falls} \quad \sum_{i=0}^{n-1} 2^i (x_i - y_i) \geq 0 \\ 0 & \text{sonst} \end{cases}$$

$\square$

Wie leicht aus Definition 3.2.1 zu sehen ist kann man COMPARISON mittels einem Schwellengatters mit exponentiellen Gewichten berechnen. Es kann aber nicht mittels Schwellengatter mit polynomiell beschränkten Gewichten berechnet werden. Außerdem gilt $COMPARISON \in \widehat{LT}_2$.

Zur Vereinfachung der Notation wird im Folgenden $X$ bzw. $Y$ mit der durch sie definierten natürlichen Zahl gleichgesetzt wenn keine Mehrdeutigkeiten entstehen können.

Der folgende Hilfssatz sagt etwas über die Struktur der Zahlen aus wenn $X \geq Y$ bzw. $X < Y$ ist. Man kann ihn leicht so deuten: Ist $X = Y$, dann müssen alle Stellen identisch sein. Ist $X > Y$ so muß eine Stelle $x_i$ in $X$ größer als $y_i$ sein und alle höherwertigen Stellen müssen gleich sein.

**Lemma 3.2.1**

$$\begin{aligned} C_n(X,Y) = 1 \quad \Leftrightarrow \quad & \{\exists_{0 \leq i \leq n-1} \forall_{i < j \leq n-1} : [(x_i > y_i) \wedge (x_j = y_j)]\} \\ & \vee \quad \forall_{0 \leq i \leq n-1} : (x_i = y_i) \end{aligned}$$

Der Beweis dazu ist im Anhang A.1 auf Seite 63 zu finden.

### 3.2.2 Einige Schaltkreise für COMPARISON

In diesem Abschnitt stelle ich vier Schaltkreise für $C_n$ vor. Die erste Lösung (genannt *LNB*, Legenstein nach Beiu) stammt von mir und besitzt ähnliche Eigenschaften wie eine von Beiu, Peperstraete, Vandewalle und Lauwereins vorgestellte [BPVL93b, BPVL93a], scheint mir aber strukturell einleuchtender.

#### Konstruktion von Legenstein nach Beiu (LNB)

Die Idee der Konstruktion ist, COMPARISON der zwei $n$-Bit Zahlen $X$ bzw. $Y$ auf COMPARISON von zwei $2n/\Delta$-Bit Zahlen $X'$,$Y'$ zu reduzieren. Der Schaltkreis ist für $n = 8$ in Abbildung 3.3 auf der nächsten Seite zu sehen. Für $Z = (Z_0, \ldots, Z_{n-1})$ und ein $\Delta$ mit $4 \leq \Delta \leq n$ sei

$$Z^{i,\Delta} = (z_{i\Delta}, \ldots, z_{(i+1)\Delta-1}) \quad \text{für} \quad 0 \leq i \leq \frac{n}{\Delta} - 1$$

(Wir nehmen hier zur Vereinfachung der Ausdrücke an daß $n/\Delta$ ganzzahlig ist.)

**Lemma 3.2.2**

$$\begin{aligned}
\text{Sei:}\quad & X' = (C_{\frac{\Delta}{2}}(X^{0,\frac{\Delta}{2}}, Y^{0,\frac{\Delta}{2}}), \ldots, C_{\frac{\Delta}{2}}(X^{\frac{2n}{\Delta}-1,\frac{\Delta}{2}}, Y^{\frac{2n}{\Delta}-1,\frac{\Delta}{2}})) \\
& Y' = (C_{\frac{\Delta}{2}}(Y^{0,\frac{\Delta}{2}}, X^{0,\frac{\Delta}{2}}), \ldots, C_{\frac{\Delta}{2}}(Y^{\frac{2n}{\Delta}-1,\frac{\Delta}{2}}, X^{\frac{2n}{\Delta}-1,\frac{\Delta}{2}})) \\
\text{Dann gilt:}\quad & C_n(X,Y) = C_{\frac{2n}{\Delta}}(X', Y')
\end{aligned}$$

Das Lemma sagt aus daß man die $n$ Bits von $X$ und $Y$ jeweils in Gruppen von $\Delta/2$ Bits aufteilt, auf die erhaltenen $2n/\Delta$ Zahlenpaare $X^{i,\Delta/2}$, $Y^{i,\Delta/2}$ zwei COMPARISON Operationen durchführt und das Problem somit auf COMPARISON von zwei $2n/\Delta$ Bit Zahlen $X'$ und $Y'$ reduziert. Im Anhang A.2 auf Seite 63 ist der Beweis zu finden.

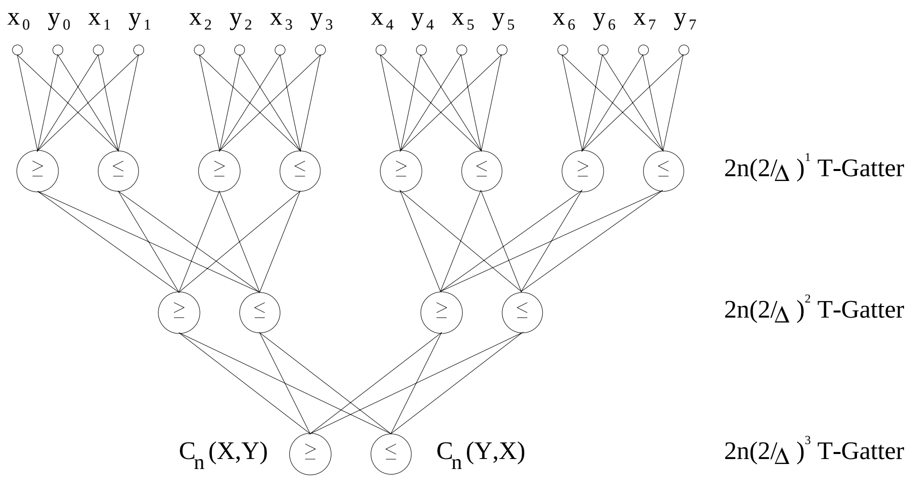


Abbildung 3.3: COMPARISSON, Schaltkreis LNB für $n = 8$. $(\geq)$ berechnen Bits von X', $(\leq)$ berechnen Bits von Y'.

Die eingeschränkte COMPARISON-Operation wird nach Definition 3.2.1 mit einem Schwellengatter mit Fan-in $\Delta$ berechnet. Die rekursive Anwendung dieser Reduktion ergibt in der $i$-ten Schicht zwei Zahlen zu je $\frac{n}{(\frac{\Delta}{2})^i}$ Bits. Das ergibt eine Schaltkreistiefe von

$$depth_{LNB} = \frac{log(n)}{log(\Delta) - 1} = O\Big(\frac{log(n)}{log\Delta}\Big) \tag{3.1}$$

Die Gewichte der Gatter im Schaltkreis sind beschränkt durch $2^{\frac{\Delta}{2}}$. Die Schwelle ist jeweils 0. Für die Summe der Gewichte eines Gatters gilt

$$\sum_{i=0}^{\Delta-1} |w_i| + |\Theta| < 2^{\frac{\Delta}{2}+1} = O\Big(2^{\frac{\Delta}{2}}\Big) \tag{3.2}$$

Man sieht daß für konstantes Fan-in $\Delta$ auch die Gewichte durch eine Konstante beschränkt sind, bei logarithmischen Fan-in sind sie polynomiell und bei linearem Fan-in exponentiell.
Betrachten wir nun die Anzahl der benötigten Gatter $size_{LNB}$.

$$\text{In Schicht } i \text{ benötigt man } 2n\left(\frac{1}{\frac{\Delta}{2}}\right)^i \text{ Gatter}$$

$$size_{LNB} = \sum_{i=1}^{depth_{LNB}} 2n\left(\frac{1}{\frac{\Delta}{2}}\right)^i$$

Mit der Formel

$$S_k = \sum_{i=1}^{k} a_1 q^{i-1} = \frac{a_1 - a_k q}{1-q}$$

erhält man

$$size_{LNB} = \frac{4(n-1)}{\Delta - 2} = O\Big(\frac{n}{\Delta}\Big) \tag{3.3}$$

*Anmerkung:*
Bei der Reduktion der Zahlen $X$ und $Y$ muß jeweils $C_n(Y^{0,\Delta}, X^{0,\Delta})$ nicht berechnet werden, da dieser Wert zur Nachverarbeitung nicht mehr benötigt wird. Ebenso wird dieser Wert bei der Berechnung des Ergebnisses in der letzten Schicht nicht benötigt. Daher reduziert sich die Größe des Schaltkreises auf

$$size_{LNB} = \frac{4(n-1)}{\Delta-2} - depth_{LNB} = \frac{4(n-1)}{\Delta-2} - \frac{log(n)}{log(\Delta)-1}$$

### Konstruktion von Siu et al.(SRK)

Diese Konstruktion wurde von Siu et al. in [SRK91] vorgeschlagen. Es handelt sich um einen AND-OR Schaltkreis der die Eigenschaften von Comparison (Lemma 3.2.1 auf Seite 22) ausnützt.
Nach Lemma 3.2.1 kann man Comparison rekursiv berechnen:

$$C_1(X,Y) = x_1 \vee \bar{y}_1$$
$$C_n(X,Y) = (x_n \wedge \bar{y}_n) \bigvee [(x_n \vee \bar{y}_n) \wedge C_{n-1}(X,Y)]$$

Nun kann man folgende booleschen Ausdrücke definieren und damit den Schaltkreis angeben:

$$B_0 = \bigwedge_{j=0}^{n-1} (x_j \vee \bar{y}_j)$$
$$B_k = (x_k \wedge \bar{y}_k) \wedge \{ \bigwedge_{j=k+1}^{n-1} (x_j \vee \bar{y}_j)\} \quad \text{für} \quad 1 \leq k \leq n-2$$
$$C_n = (x_{n-1} \wedge \bar{y}_{n-1}) \vee \bigvee_{k=0}^{n-2} B_k$$

Der Schaltkreis sieht folgendermaßen aus:

**1. Layer:** $n-1$ AND-Gatter berechnen $(x_i \wedge \bar{y}_i)$ und $n$ OR-Gatter berechnen $(x_i \vee \bar{y}_i)$

**2. Layer:** $n-1$ AND-Gatter berechnen $B_k$

**3. Layer:** Ein OR-Gatter berechnet das Resultat $C_n$

Wie leicht zu sehen ist hat dieser Schaltkreis folgende Eigenschaften:

$$size_{SRK} = 3n - 1 = O(n) \qquad depth_{SRK} = 3$$
$$FanIn_{SRK} \leq n$$
$$w_{i,j} = \pm 1 \qquad \Theta \leq n$$

### Modifikation der Konstruktion von Siu et al.(LEG)

In Bezug auf Schaltkreistiefe ist die Konstruktion von SRK bei polynomieller Größe mit AND/OR-Gattern optimal (siehe [SRK95], Theorem 5.5). Man sieht jedoch daß die Berechnung der $B_k$ ineffizient ist.

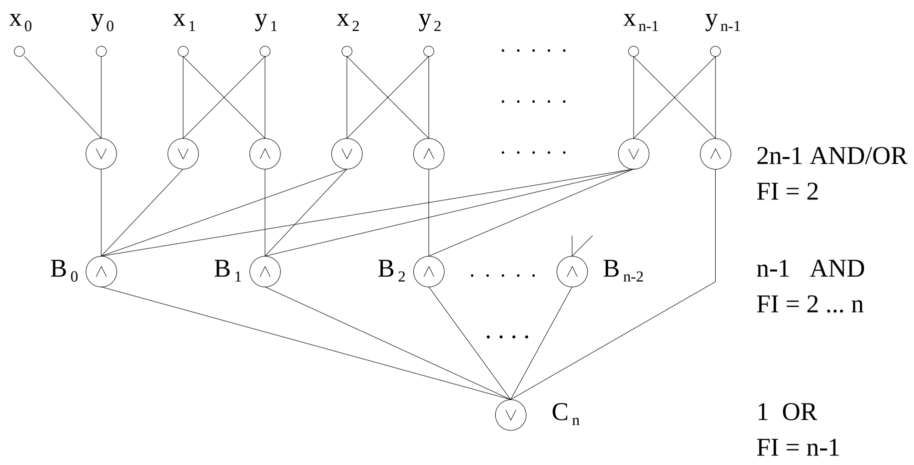


Abbildung 3.4: COMPARISSON, Schaltkreis SRK

Die laufenden AND-Verknüpfungen $\{\bigwedge_{j=k}^{n-1}(x_j \vee \bar{y}_j)\}$ für $0 \leq k \leq n-2$ können mittels paralleler Präfixberechnung (siehe Abschnitt 3.1 auf Seite 17) berechnet werden. Eine weitere Schicht von AND-Gattern mit $FanIn = 2$ kann aus diesen Präfixen und $(x_k \wedge \bar{y}_k)$ die $B_k$ berechnen. Dabei kann auch im Sinne einer VLSI-Implementation der Fan-in der Gatter beschränkt werden. Das OR-Gatter des Outputs kann man ebenfalls leicht als OR-Baum implementieren.

Damit hat der resultierende Schaltkreis folgende Eigenschaften:

$$\begin{aligned} size_{LEG} &= O(n + \frac{n \log^* n}{\Delta}) & edges_{LEG} &= O(n \log^* n) \\ depth_{LEG} &= O(\frac{\log n}{\log \Delta}) & FanIn_{LEG} &\leq \Delta \\ w_{i,j} &= \pm 1 & \Theta &\leq \Delta \end{aligned}$$

### Konstruktion von Roychowdhury et al. (ROS)

Zur Vollständigkeit sei noch eine Idee von V.P. Roychowdhury, A. Orlitsky und K.-Y. Siu angeführt [ROS94]. Sie behandeln COMPARISON mithilfe von allgemeinsymmetrischen Funktionen (asF, siehe dazu Definition 3.3.2 auf Seite 44). Bei der Konstruktion wurde darauf geachtet, daß die Gewichte der Schwellengatter polynomiell beschränkt sind, und daher asF berechnen. Aus dieser Intention wird die Konstruktion schnell klar. Man bekommt

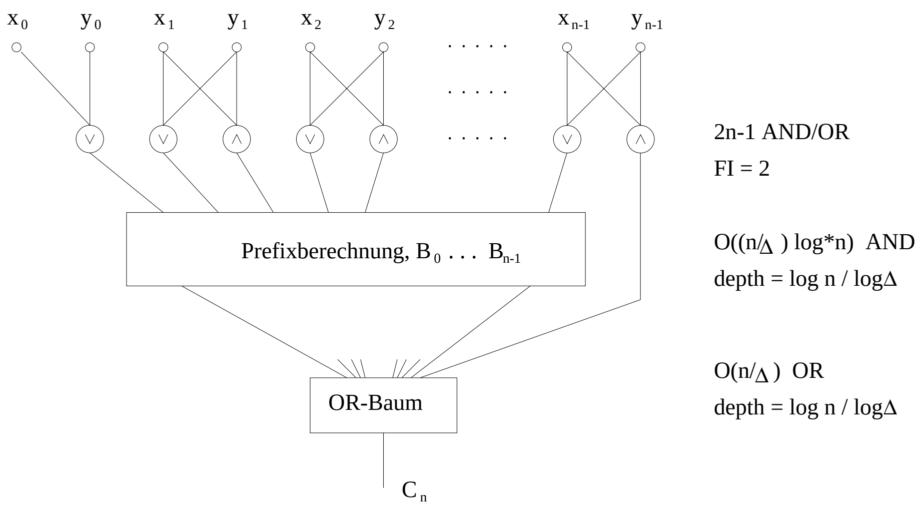


Abbildung 3.5: COMPARISSON, Schaltkreis LEG

einen Schaltkreis für COMPARISON in optimaler Größe $O(n/\log n)$ (bei Verwendung von asF).

Der Aufbau des Schaltkreises ist sehr ähnlich zu den bereits besprochenen. Im ersten Layer werden COMPARISONS von jeweils $\log n$ Variablen durchgeführt, ähnlich wie bei LNB. Im ersten Layer wird für $m = \lceil \log n \rceil + 1$ $C_i = X^{i,m} > Y^{i,m}$ und $\tilde{C}_i = X^{i,m} \geq Y^{i,m}$ mittels T-Gattern berechnet. Somit hat der erste Layer $2\lceil n/m \rceil - 1$ TGatter mit Fan-in $2m$. Im 2. Layer hat man $\lceil n/m \rceil - 1$ AND-Gatter mit $FanIn = 2, 3, \ldots, \lceil n/m \rceil$

$$B_1 = \bigwedge_{j=1}^{\lceil n/m \rceil} \widetilde{C}_j$$

$$B_k = C_k \wedge ( \bigwedge_{j=k+1}^{\lceil n/m \rceil} \widetilde{C}_j) \quad k \geq 2$$

Ein Gatter im dritter Layer verbindet wiederum diese Ergebnisse zu:

$$C_n(X,Y) = \bigvee_{k=1}^{\lceil n/m \rceil} B_k$$

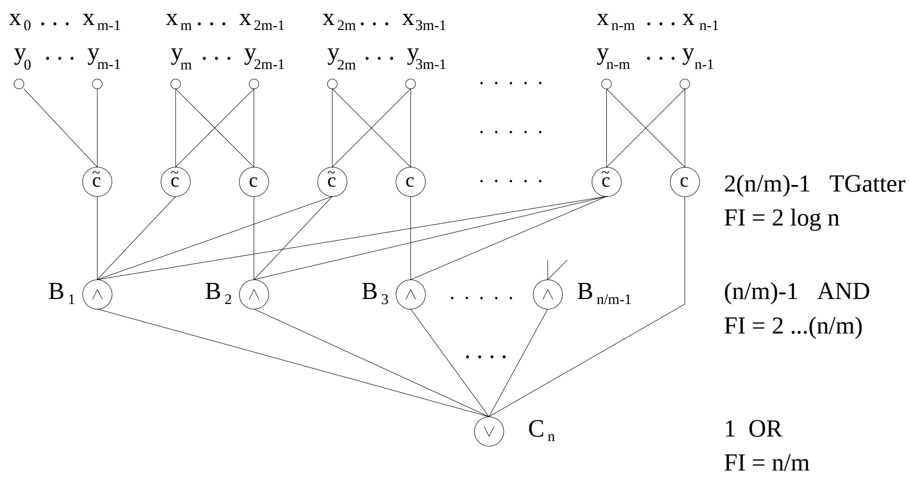


Abbildung 3.6: COMPARISSON, Schaltkreis ROS

Damit gelten folgende Schranken für den Schaltkreis:

$$size_{ROS} = 3\lceil\frac{n}{\lceil\log n\rceil + 1}\rceil - 1 \qquad depth_{ROS} = 3$$

$$FanIn_{ROS} \leq \lceil\frac{n}{\lceil\log n\rceil + 1}\rceil \qquad w_{i,j}, \Theta \leq 2^{\lceil\log n\rceil}$$

Es wurden vier Möglichkeiten beschrieben, COMPARISON zu berechnen.

**Kurze Diskussion**

Alle angegebenen Lösungen haben polynomiell beschränkte Gewichte oder sind mit AND-OR Schaltkreisen realisierbar. Die Lösungen LNB und LEG haben beschränkten Fan-in bei logarithmischer Tiefe. SRK und ROS haben konstante Tiefe 3. Dabei ist ROS sizeoptimal. Bei genügend großem $\Delta$ hat LNB noch bessere size, hat dann aber superpolynomionelle Gewichte.

Im folgenden Kapitel werden die Schaltkreise auf VLSI-Komplexität untersucht. Es zeigt sich daß die Schaltkreise mit beschränktem Fan-in durchwegs besser abschneiden.

### 3.2.3 COMPARISON und VLSI

**Kantenkomplexität**

Die Kantenkomplexität gibt an, wieviele Kanten in einem Schaltkreis auftreten. Am Chip ist die Anzahl der Verbindungen zwischen den Neuronen besonders kritisch (siehe auch Abschnitt 2.1.1). Viele Verbindungen sind nur schwer zu implementieren. Eine geringe, lokale Kommunikation zwischen den Gattern ist für ein Layout anzustreben. Unter bestimmten Bedingungen kann mit der Kantenkomplexität der Platzverbrauch am Chip gut abgeschätzt werden. Etwa bei analogen Schaltkreisen mit konstantem Platzverbrauch der Gewichte.

Die Kantenkomplexität *edges* kann auch einfach über die Summe der Fan-ins der Gatter berechnet werden. $edges = \sum_{NN} FanIn$
Für die Kantenkomplexität ergeben sich folgende Abschätzungen:

$$edges_{LNB} = O(n) \tag{3.4}$$
$$edges_{SRK} = O(n^2) \tag{3.5}$$
$$edges_{LEG} = O(n \log^* n) \tag{3.6}$$
$$edges_{ROS} = O\Big(\frac{n^2}{\log^2 n}\Big) \tag{3.7}$$

**Beweis:** Die Lösung von **LNB** hat $O(n/\Delta)$ Gatter mit $Fan - in \leq \Delta$. Das ergibt insgesamt $O(n)$ Kanten.
Die Lösung von **SRK** hat im ersten Layer $2n$ Gatter mit Fan-in 2. Im kritischen 2. Layer gibt es $n-1$ Gatter mit $Fan - in = 2, 3, \ldots, n$. Also insgesamt $O(n^2)$ Kanten. Der 3. Layer hat nur noch ein Gatter mit $Fan - in = n - 1$.
Die Kantenanzahl für **LEG** wurde bereits in der Konstruktion angegeben.
Bei der Konstruktion von **ROS** ist ebenfalls der 2. Layer kritisch. Dort gibt es $O(n/\log n)$ Gatter mit $Fan - in = 2, 3, \cdots = (n/\log n)$. Dies führt zur angegebenen Kantenkomplexität. Wie leicht zu sehen ist wird diese Kompexität im 1. und 3. Layer nicht überschritten.

□

### Kurze Diskussion

Die Lösung LNB hat also die beste Komplexität in diesem Fall. Dies ist offensichtlich auch optimal, denn schließlich muß jeder Input mit dem Output verbunden sein, was zu $edges(C_n) = \Omega(n)$ führt. Mit der Präfixberechnung konnte die Kantenkomplexität von SRK von $O(n^2)$ auf $O(n \log^* n)$ gedrückt werden. Eine deutliche Verbesserung. Der Unterschied zu LNB ist nur minimal ("fast" konstant). Man darf aber nicht vergessen, daß LEG gar keine Threshold-Gatter benötigt, sondern mit AND/OR-Gattern implementiert werden kann. Dies ist ein entscheidender Vorteil.

### Gewichtssummenkomplexität

Die Gewichtssumme eines Netzwerks (*weightsum*) ist die Summe über die Beträge aller Gewichte und Thresholds in den Gattern desselben:

$$weightsum = \sum_{NN} \left( \sum_i |w_i| + |\Theta| \right) \tag{3.8}$$

In der Realisierung von Netzwerken kommt der Speicherung der Gewichte eine bedeutende Rolle zu. Dabei ist nicht nur wichtig daß das Maximum der Gewichte klein ist, exponentielle Gewichte sind für großes $n$ kaum noch zu realisieren, sondern auch die Summe der Gewichte soll klein gehalten werden.

Für AND/OR-Schaltkreise macht dieses Maß wenig Sinn. Wir führen es trotzdem ein und denken uns dabei die AND/OR-Gatter als Thresholdgatter implementiert. Das T-Gatter hat dann Gewichte von $\pm 1$ (je nachdem ob die Variable negiert oder nicht negiert auftritt), und der Threshold ist kleiner gleich dem Fan-in des Gatters. Es ist leicht zu zeigen daß ein T-Gatter mit solchen Gewichten sowohl AND- also auch OR-Funktionen berechnen kann.

Zur Abschätzung der Gewichtskomplexität solcher Schaltkreise ist folgender Hilfssatz hilfreich:

**Lemma 3.2.3** Sei $S$ ein AND/OR-Schaltkreis mit einer Kantenkomplexität $edges(S)$. Dann gilt: $weightsum(S) \leq 2edges(S)$

Der Ausgangspunkt des Beweises ist, daß pro Kante nur ein Gewicht vom Betrag 1 benötigt wird, und ebenso pro Kante die Thresholdsumme um maximal 1 erhöht wird.

**Beweis:** Man betrachte die Gatter $g_i$ im Netzwerk ($1 \leq i \leq size(S)$) mit Fan-in $FanIn_i$. Es gilt: $\sum_{i=1}^{size(S)} FanIn_i = edges(S)$. Für jedes Gatter $g_i$ ist die Gewichts- und Thresholdsumme $weightsum(g_i) \leq 2FanIn_i$. Aufsummiert auf alle Gatter folgt die Behauptung.

□

Alle diskutierten Schaltkreise haben polynomielle Gewichte (LNB hat polynomielle Gewichte für $\Delta = O(\log n)$ ). Trotzdem zeigt sich, daß auch in diesem Komplexitätsmodell große Unterschiede zwischen den Schaltkreisen auftreten.

Die Gewichtssummenkomplexität der besprochenen Schaltkreise beträgt:

$$\begin{aligned}
weigntsum_{LNB} &= O\Big(\frac{n}{\Delta}2^{\frac{\Delta}{2}}\Big)\\
weightsum_{SRK} &= O(n^2)\\
weightsum_{LEG} &= O(n\log^* n)\\
weightsum_{ROS} &= O\Big(\frac{n^2}{\log n}\Big)
\end{aligned}$$

**Beweis:** Für die Konstruktion von **LNB**: Nach Formel 3.2 auf Seite 24 hat jedes Gatter im Schaltkreis eine Gewichts-Thresholdsumme von $O\Big(2^{\frac{\Delta}{2}}\Big)$. Die Anzahl der Gatter im Netzwerk ist $O(n/\Delta)$. Das ergibt eine Gewichtssumme über alle Gatter von $O\Big(\frac{n}{\Delta}2^{\frac{\Delta}{2}}\Big)$.
Für die Konstruktion von **SRK**: Der Schaltkreis hat $O(n^2)$ Kanten. Aus Lemma 3.2.3 auf der vorherigen Seite ergibt sich ebenfalls eine Gewichtssumme von $O(n^2)$.
Für die Konstruktion von **LEG**: Auch diese Konstruktion ist ein AND-OR Schaltkreis. Diesmal mit Kantenkonmplexität von $O(n\log^* n)$. Aus Lemma 3.2.3 folgt, daß die Gewichtssumme in der selben Ordnung bleibt.
Für die Konstruktion von **ROS**: Im 1. Layer haben $O(n/\log n)$ T-Gatter jeweils eine Gewichtssumme von O(n). Das ergibt in Summe $weightsum = O(n^2/\log n)$. Die restlichen Layers sind ein AND-OR Schaltkreis mit Kantenkomplexität $edges = O(n^2/\log^2 n)$. Nach Lemma 3.2.3 verbleibt die Gesamtkomplexität des Schaltkreises bei $weightsum = O(n^2/\log n)$.

□

**Kurze Diskussion**

Wiederum erzielen LNB und LEG die besten Ergebnisse. Freilich darf bei LNB der Fan-in nicht zu groß werden, denn die Gewichtssummenkomplexität steigt bei dieser Konstruktion exponentiell mit diesem. So würde man etwa mit logarithmischen Fan-in etwa in den Bereich von ROS kommen. Bei $FanIn = n$ haben wir den Extremfall daß nur noch ein Gatter da ist, das auf die gewohnte Weise COMPARISON berechnet, natürlich mit exponentiellen Gewichten. Bei konstantem $\Delta$ allerdings ist diese Konstruktion zumindest von der Ordnung her mit $weightsum = O(n)$ nicht zu schlagen.

Aber es zeigen sich auch Vorteile von LEG. Diese Verbesserung von SRK ist in der Gewichtssummenkomplexität vom Fan-in unabhängig. Es werden eben nur AND/OR-Gatter benötigt, und das schlägt sich hier nieder.

Während man bei LNB also möglichst kleinen Fan-in sucht, kann dieser hier beliebig groß gewählt werden, je nach Hardwareanforderung. Wenn $n$ in nicht allzu kleinen Grenzen gehalten wird, wird der multiplikative Faktor von $\log^* n$ bei LEG durch den $\Delta$-abhängigen Faktor bei LNB aufgewogen.

### Delay nach Eingangskapazität

Die Berechnungszeit(*delay*) eines Schaltkreises wird in der Theorie meist durch die Schaltkreistiefe abgeschätzt. Wenn man davon ausgeht daß in jedem Layer die Eingangskapazität die Verzögerung eines Gatters bestimmt, so ergibt sich das delay in jedem Layer durch das Gatter mit dem größten Fan-in im Layer.

Das *FIdelay* wird dann berechnet, indem man für jeden Layer den maximalen Fan-in bestimmt, und diese aufsummiert.

$$FIdelay(S) = \sum_{i=1}^{depth(S)} \text{maximaler Fan-in in Layer } i \text{ von } S \tag{3.9}$$

Mit dieser Schätzung ergibt sich das delay der Schaltkreise zu

$$\begin{aligned}
FIdelay_{LNB} &= O\Big(\frac{\Delta \log n}{\log \Delta}\Big)\\
FIdelay_{SRK} &= O(n)\\
FIdelay_{LEG} &= O\Big(\frac{\Delta \log n}{\log \Delta}\Big)\\
FIdelay_{ROS} &= O\Big(\frac{n}{\log n}\Big)
\end{aligned}$$

**Beweis:** Da der Fan-in bei **LNB** und **LEG** durch $\Delta$ beschränkt ist, ergibt sich die Summe über die Layer automatisch als $\Delta depth$, was zu den Ergebnissen führt.
Bei **SRK** besitzen sowohl Layer zwei als auch Layer drei linearen Fan-in. Der erste Layer hat konstanten Fan-in von zwei. Demnach gilt in diesem Fall für das delay $FIdelay_{SRK} = O(n)$.
Bei **ROS** hat der erste Layer Fan-in von $O(\log n)$. Der zweite Layer hat Fan-in von $2, 3, \ldots, O(n/\log n)$. Der maximale Fan-in bestimmt das delay des zweiten Layers. Der dritte Layer besteht aus einem Gatter mit Fan-in $O(n/\log n)$. Daraus folgt die Behauptung.

□

### Lokalitätsbetrachtungen

Für das Chiplayout ist auch die Lokalität der Gatterfunktionen von Bedeutung. Eine optimale Strategie wäre etwa eine baumartige Berechnung bei der nur Ergebnisse von lokal leicht zugänglichen Vorgängergattern in einem Gatter benötigt werden. Eine solche Berechnung ist durch einen planaren Graphen darstellbar und somit kreuzungsfrei. Als grobes Maß für Lokalität kann die Anzahl der Kreuzungen im Berechnungsgraphen verwendet werden.

Der Umkreis aus dem ein Gatter seine Inputwerte bezieht und die summierte Wirelength ist auch eine Betrachtung wert. Diese Maße sind allerdings nur schwer anzugeben, da hier die geometrische Anordnung der Gatter eine bedeutende Rolle spielt. Ich gehe hier davon aus, daß sich die Inputs des Schaltkreises auf einer Seite des Chips befinden und daß der Schaltkreis in gewohnter Manier gelayert abgebildet wird. Betrachtet man aber z.B. ein Gatter, das Verbindungen zu $n$ Inputs besitzt, und berechnet man die maximale Wirelength zum Gatter, so kommt man bei angegebenen Voraussetzungen zu $n/2$. Würde man die Inputs im Kreis anordnen und das Gatter in den Mittelpunkt setzen, so wäre die maximale Wirelength nur noch $\frac{n}{2\Pi}$.

Um die folgenden Berechnungen nicht unnötig zu verkomplizieren, nehme ich im Folgenden nicht die exakten Werte der zuvor berechneten Größen. Im Endeffekt interessieren mich nur die Ordnungen der Ergebnisse, sodaß ich z.B. *size* oder *edges* auf deren wesentlichen Teil beschränke, d.h. Konstanten meist weglasse. Die sich dadurch ergebenden Zwischenergebnisse sind natürlich nicht exakt.

Betrachten wir die Kreuzungen(*crossings*) der Schaltkreise, so kommen wir zu folgenden Ergebnissen:

$$\begin{aligned} crossings_{LNB} &= O(n\Delta) \\ crossings_{SRK} &= O(n^4) \\ crossings_{LEG} &= O(n^2(log^* n)^2) \\ crossings_{ROS} &= O\Big(\frac{n^4}{\log^4 n}\Big) \end{aligned}$$

**Beweis:** Der Schaltkreis von **LNB** ist eindeutig der lokalste. Dieser Schaltkreis hat für kleines $\Delta$ eine sehr baumähnliche Struktur. Werden die Bits von $X$ und $Y$ alternierend angeboten (d.h. $x_0, y_0, x_1, y_1, x_2, \ldots, x_{n-1}, y_{n-1}$), so ergeben sich lokale Kreuzungen nur zwischen jeweils zwei Knoten. Da die Bits für den nächsten Layer $X'$ und $Y'$ somit ebenfalls alternierend berechnet werden, ist diese Voraussetzung auf dem ganzen Schaltkreis erfüllt. Die

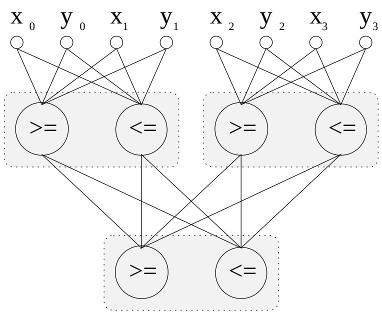


Abbildung 3.7: *"Knoten"* im Schaltkreis LNB für $\Delta = 4$

einzigen Unregelmäßigkeiten ergeben sich dadurch, daß für Untergruppen $A = X^{i,\Delta/2}$ und $B = Y^{i,\Delta/2}$ jeweils $A \geq B$ und $A \leq B$ berechnet werden muß. Betrachtet man diese zwei Funktionen als lokale Operation und weist ihnen einen Knoten im Berechnungsgraphen zu (siehe Abbildung 3.7), so ist der Berechungsgraph ein kreuzungsfreier Baum. Die *crossings* des Schaltkreises ergeben sich also als crossings in einem solchen Knoten summiert über alle Knoten. Die Anzahl der Kreuzungen in diesem lokalen Knoten ist dann:

$$crossings_{node} = \sum_{i=1}^{\Delta-1} i = \frac{\Delta(\Delta-1)}{2} = O(\Delta^2)$$

Da für jeden solchen Knoten zwei Gatter verwendet werden, ergibt sich nach Formel 3.3 auf Seite 24

$$\begin{aligned} crossings_{LNB} &= \frac{size_{LNB}}{2} crossings_{node} \\ &= \frac{2(n-1)}{\Delta-2} \frac{\Delta(\Delta-1)}{2} \\ &= O(n\Delta) \end{aligned}$$

Bei **SRK** ist der erste Layer sehr lokal. Es werden nur Funktionen von zwei Inputs berechnet, wobei jeder Input nur zwei mal benötigt wird. Bei alternierender Anordnung der Inputs (wie bei LNB) hat man hier weniger als $2n$ Kreuzungen. Es gibt allerdings sehr viele Kanten zwischen erstem und zweitem Layer.

Man betrachte ein $B_k$. Wie viele Kanten von Gattern $B_j$ mit $j < k$ (davon gibt es $k$) schneiden die Kanten von $B_k$ ? Das Gatter $B_k$ selbst hat

$n-k$ Inputkanten. Wir stellen uns den Schaltkreis so vor: Die $B_i$ sind von links nach rechts aufsteigend sortiert. Ebenso sind die Gatter des ersten Layers aufsteigend sortiert (siehe Abbildung 3.8 auf der nächsten Seite). Jedes $B_j$ hat eine Kante zu jedem der $n-k-1$ nachfolgenden $(x_l \vee \bar{y}_l)$. Daher wird die linkeste Kante von $B_k$ $k(n-k-1)$ mal geschnitten, die nächste $k(n-k-2)$ mal usw.. Für ein Gatter ergibt sich somit folgende Anzahl von crossings$(\alpha = n-k)$ :

$$\begin{aligned} crossings_k &= \sum_{i=1}^{n-k} k(n-k-i) = k \sum_{i=1}^{\alpha} \alpha - i \\ &= k \sum_{i=1}^{\alpha-1} i = k\frac{\alpha(\alpha-1)}{2} \\ &= k\frac{(n-k)(n-k-1)}{2} \end{aligned}$$

Dies über alle $B_i$ summiert führt nach einigen Berechnungen zu

$$\begin{aligned} crossings_{SRK} &= \sum_{i=0}^{n-1} i\frac{(n-i)(n-i-1)}{2} \\ &= \frac{1}{2}\sum_{i=1}^{n} i(n-i)(n-i-1) \\ &= \frac{1}{24}(n^4 - 2n^3 - n^2 + 2n) \\ &= O(n^4) \end{aligned}$$

Im dritten Layer(eine OR-Verknüpfung) gibt es keine Kreuzungen.

Betrachten wir nun die *crossings* bei **ROS**. Dieser Schaltkreis ist im ersten Layer strukturell gleich aufgebaut wie der von LNB mit $\Delta = O(\log n)$. Berechnet man $C_i$ und $\bar{C}_i$ im Layer alternierend und betrachtet $C_i$ und $\bar{C}_i$ als einen Knoten , so gibt es wiederum nur Kreuzungen innerhalb der Knoten. Ein solcher Knoten hat dann folgende *crossings*:

$$crossings_{node} = \sum_{i=1}^{(\log n)-1} i = \frac{\log n(\log n - 1)}{2} = O(\log^2 n)$$

Und aufsummiert auf alle $O(n/\log n)$ Knoten des ersten Layers ergibt sich

$$\begin{aligned} crossings_{Layer1} &= \frac{n}{\log n}\frac{\log n(\log n - 1)}{2} \\ &= O(n \log n) \end{aligned}$$

Wirklich aufwendig ist allerdings der zweite Layer, der strukturell gleich ist wie bei SRK, also sehr ähnlich einer Präfixberechnung, nur diesmal mit $n/\log n$ Inputs statt mit $n$. Die Ergebnisse von SRK kann man also übernehmen, wobei jeweils $n$ durch $n/\log n$ zu substituieren ist. Da es im dritten Layer wiederum keine Kantenüberkreuzungen gibt, ergibt sich die Anzahl der *crossings* zu

$$\begin{aligned} crossings_{ROS} &= crossings_{Layer1} + crossings_{Layer2} \\ &= O\Big(\frac{n^4}{\log^4 n}\Big) \end{aligned}$$

Die Berechnung der *crossings* bei **LEG** stellt etwas aufwendig dar. Dies vor allem wegen der strukturell aufwendigen Präfixberechnung. Es gibt jedoch eine einfache Möglichkeit, die *crossings* nach oben abzuschätzen. Nach einfachen geometrischen Gesetzen können sich $n$ Geraden nur $O(n^2)$ mal schneiden. Da es in LEG nach Formel 3.6 auf Seite 29 $O(n \log^* n)$ Kanten gibt, sind die *crossings* beschränkt durch

$$crossings_{LEG} = O(n^2 (\log^* n)^2)$$

□

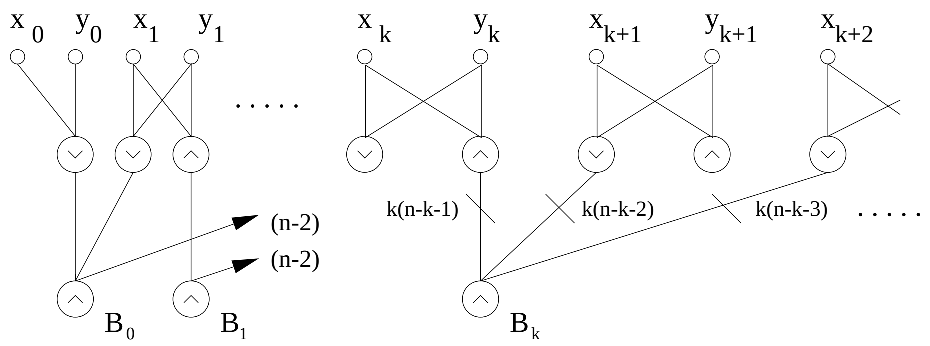


Abbildung 3.8: Kreuzungen im Schaltkreis SRK

### Kurze Diskussion

Daß **LNB** sehr lokal ist, drückt sich in den sehr wenigen *crossings* aus. LNB ist hier eindeutig der beste Schaltkreis. **SRK** und **ROS** haben durch den aufwendigen zweiten Layer, der nicht effizient implementiert ist, sehr

| | LNB | SRK | LEG | ROS |
|---|---|---|---|---|
| **depth** | $O\left(\frac{\log n}{\log \Delta}\right)$ | 3 | $O\left(\frac{\log n}{\log \Delta}\right)$ | 3 |
| **size** | $O\left(\frac{n}{\Delta}\right)$ | $O(n)$ | $O\left(n + \frac{n \log^* n}{\Delta}\right)$ | $O\left(\frac{n}{\log n}\right)$ |
| **max. Fan-in** | $\Delta$ | $n$ | $\Delta$ | $\frac{n}{\log n}$ |
| **max. Gewicht** | $2^{\frac{\Delta}{2}}$ | 1 (AND/OR) | 1 (AND/OR) | $n$ |
| **edges** | $O(n)$ | $O(n^2)$ | $O(n \log^* n)$ | $O\left(\frac{n^2}{\log^2 n}\right)$ |
| **weightsum** | $O\left(\frac{n}{\Delta} 2^{\frac{\Delta}{2}}\right)$ | $O(n^2)$ | $O(n \log^* n)$ | $O\left(\frac{n^2}{\log n}\right)$ |
| **FIdelay** | $O\left(\frac{\Delta \log n}{\log \Delta}\right)$ | $O(n)$ | $O\left(\frac{\Delta \log n}{\log \Delta}\right)$ | $O\left(\frac{n}{\log n}\right)$ |
| **crossings** | $O(n\Delta)$ | $O(n^4)$ | $O(n^2 (log^* n)^2)$ | $O\left(\frac{n^4}{\log^4 n}\right)$ |

Tabelle 3.1: Verschiedene Schaltkreiskomplexitäten für COMPARISON

schlechte Lokalitätseigenschaften. Hier hat man jeweils $edges^2$ viele Kreuzungen, was die schlechteste Komplexität in diesem Bereich darstellt. Auch **LEG** hat mit $edges^2$ Kreuzungen eine schlechte Komplexität, da es aber wenige Kanten gibt, ist LEG auch hier den genannten Schaltkreisen überlegen. Allerdings schneidet LEG im Vergleich zu LNB diesmal ungewohnt schlecht ab. Die Präfixberechnung mittels Intervallen ist eben nicht lokal(siehe Kapitel 3.1 auf Seite 17).

### 3.2.4 Konklusionen

Eine Zusammenstellung der erzielten Ergebnisse ist in der Tabelle 3.2.4 zu sehen.

Depth-optimale Schaltkreise sind nur mit unbeschränktem Fan-in erzielbar. Es ist noch keine explizite Funktion bekannt, dessen Tiefe größer als 3 sein müßte. Konstantes Fan-in impliziert aber mindestens logarithmische Tiefe und lineare size. So haben die zwei Schaltkreise mit linearem Fan-in, *SRK* und *ROS*, konstante Tiefe. *ROS* hat optimale sublineare Größe, ebenfalls bei konstantem Fan-in nicht erreichbar.

Nun sind diese Maße aber nicht sehr aussagekräftig wenn es um Realisierungen solcher Schaltkreise geht, und es ist interessant zu sehen wie sich eine Einbuße in der Tiefe durch Vorteile in allen hier besprochenen VLSI-freundlichen Komplexitätsmaßen auswirkt. Wenn man die Tiefe nicht als

Maßstab für das delay heranzieht (was durchaus realistisch ist), so ergeben sich eigentlich keine Vorteile von Tiefenoptimalen Schaltkreisen. Ein Schaltkreis mit konstantem Fan-in und logarithmischer Tiefe hat immer nur ein logarithmisches **FIdelay**, bei einem Schaltkreis mit maximalem Fan-in $f(n)$ ist das FIdelay zumindest $f(n)$. Bei den Leitungslängen zeigen sich übrigens auch Nachteile von unbeschränkten Fan-in. Über Umwege, denn ausschlaggebend ist hier die Lokalität. Ein Gatter mit großem Fan-in muß seine Inputs auch von entfernten Neuronen beziehen, was die Leitungslänge erhöht. Natürlich handelt es sich hierbei meist um konstante Faktoren, da ja ohnehin alle Inputs zu dem Outputgatter geführt werden müssen.

Das Untersuchen von Schaltungen mit beschränktem Fan-in macht jedenfalls Sinn. Nicht nur da in VLSI Grenzen an das Fan-in vorgegeben sind. Man kann erwarten, die Konnektivität zu verringern, und weiters die Lokalität zu erhöhen. Dies zeigt sich bei den Abschätzungen der **edges** und **crossings**. Bei letzterer Komplexität sind große Unterschiede auffallend. Beiu hat bei einem ähnlich wie *LNB* strukturierten Schaltkreis den optimalen Fan-in im $AT^2$-Maß approximiert, wobei er $A = weightsum$ und $T = depth$ setzte ([Bei97]). Er kam auf kleinen konstanten Fan-in (6-9) als Optimum. Ich würde nicht so weit gehen. Immerhin hängt bei *weightsum* die Area exponentiell vom Fan-in ab, kleine optimale Werte waren also zu erwarten. Vielmehr scheint dieser Schaltkreis sehr gut geeignet zu sein um den Fan-in auf die Ansprüche der Implementierung anzupassen. Berücksichtigt man, daß AND/OR-Schaltkreise einfacher und billiger zu implementieren sind, so stellt auch *LEG* eine Alternative dar.

Natürlich bietet nicht jede Funktion so gute Struktureigenschaften wie COMPARISON um gut lokal implementiert zu werden. Das nächste Kapitel aber zeigt, wie man auch eine relativ komplexe Funktion, die Summe von $n$ $N$-Bit Zahlen, sehr gut mit lokalen Operationen berechnen kann.

Abschließend können folgende Konklusionen gezogen werden:

- VLSI-Optimalität entspricht nicht der üblichen *size*- bzw. *depth*-Optimalität.

- Beim Entwurf von VLSI-optimalen Schaltkreisen ist der Fan-in ein wichtiger Faktor.

- Für COMPARISON gibt es für verschiedene Komplexitätsmaße (*size*, *depth*, *edges)*) bei beschränktem Fan-in optimale Lösungen.

Die angesprochene Lokalität von Schaltkreisen wäre eine weitere Betrachtung wert.

## 3.3 Funktionsklassen und beschränktes Fan-in

Die Menge der booleschen Funktionen kann man in verschiedene *Funktionsklassen* unterteilen. Speziell für Schwellenschaltkreise kann man aus der Kenntnis der Klasse Schaltkreise für jede Funktion in der Klasse nach einem einheitlichen Schema entwerfen. Die einfachste Klasse ist die der *linearen Schwellenfunktionen* (linear separierbare Funktionen). Weitere Klassen sind die *symmetrischen* und die *allgemeinsymmetrischen* Funktionen (*symmetric* bzw. *generalized symmetric functions*). Schwellenfunktionen lassen sich natürlich mit einem Schwellengatter berechnen. Dieser Abschnitt will Schaltkreise mit beschränktem Fan-in Funktionen solcher Klassen berechnen.

### 3.3.1 Berechnung von Schwellenfunktionen

Übliche Schwellenschaltkreise bestehen aus Gattern mit unbeschränktem Fan-in. Das Ziel dieses Kapitels ist es, äquivalente Schaltkreise mit beschränktem Fan-in anzugeben. Dazu ist es nötig, die Schwellengatter durch Netzwerke von Gattern mit beschränktem Fan-in zu ersetzen. Im Folgenden wird ein solches Netzwerk angegeben.

Ein Schwellengatter $g$ berechnet eine Funktion $f(x_1, \ldots, x_n)$ auf seinen Inputs $x_1, \ldots, x_n$

$$f : \{0,1\}^n \to \{0,1\}$$

$$f(x_1, \ldots, x_n) = \begin{cases} 1 & \text{falls} \quad \sum_{i=1}^n w_i x_i \geq \Theta \\ 0 & \text{sonst} \end{cases}$$

Man sieht daß eine Strategie zur Berechnung der linearen Schwellenfunktion wäre, die Summe der gewichteten Inputs zu berechnen. Dies entspricht einer Addition von $n$ Ganzzahlen, deren Größe durch die $w_i$ bestimmt wird. Dann ist es einfach, einen Schaltkreis für COMPARISON aus dem Abschnitt 3.2.2 so zu verändern, daß er die Summe mit $\Theta$ vergleicht. Man nimmt dazu etwa die Konstruktion von LNB, ersetzt $Y$ durch die binäre Repräsentation von $\Theta$ und nimmt die sich dadurch ergebenden konstanten Werte in den Schwellengattern in die Schwelle auf.

Im Folgenden beschränken wir uns darauf, die Summe von $n$ $N$-Bit Ganzzahlen durch Schwellengatter mit $FanIn \leq m$ zu berechnen. Dabei gibt $N$

die Größe der binären Repräsentation des maximalen Gewichtes der linearen Schwellenfunktion an.

Dazu benötigen wir zuerst ein paar Hilfssätze, die es uns erlauben die Anzahl der zu summierenden Zahlen auf zwei zu reduzieren und dann zu addieren. Zur Vereinfachung der Notation definieren wir eine Funktion $f_d(m) = 2^{-d/2} m^{1+1/(2^d-1)} \sqrt{\log m}$

Lemma 3.3.1 sagt aus, daß man zwei $n$-Bit Ganzzahlen effizient addieren kann. Der Schaltkreis mit unbeschränktem Fan-in und konstanter Tiefe stammt aus [CFL85] und wurde in ([SRK95], Theorem 5.9) auf beschränkten Fan-in erweitert. Die Autoren haben bei diesem Schaltkreis die Stärken der parallelen Präfixberechnung (siehe Abschnitt 3.1 auf Seite 17) ausgenützt. Der Beweis wird hier nicht angegeben.

**Lemma 3.3.1 (Theorem 5.9 in [SRK95])** Die Summe zweier $n$-Bit Ganzzahlen kann mittels AND/OR Schaltkreis der Tiefe $O(\log n / \log m)$, Kantenkomplexität $O(n(\log^* n)^2)$ und $FanIn \leq m$ berechnet werden.

Das folgende Lemma gibt uns die Möglichkeit, die Summe von $m$ Inputbits effizient zu berechnen. Es wird später im Beweis zu Lemma 3.3.3 verwendet (Der Beweis ist im Anhang zu finden). Das Lemma ist ein Spezialfall eines Ergebnisses von Beame, Brisson und Ladner in [BBL92].

**Lemma 3.3.2 (Spezialfall in [BBL92])** Die $\log m$-Bit Summe $\sum_{i=1}^{m} x_i$ von $m$ Inputs $(x_1, \ldots, x_m)$ kann durch einen Schwellenschaltkreis der Tiefe $7d$ mit Kantenkompexität $O(2^{-d/2} m^{1+1/(2^d-1)} \sqrt{\log m})$ und Fan-in beschränkt durch $m$ für jede Konstante $d > 0$ berechnet werden.

Lemma 3.3.3 benützt ein Block partitions Argument um die Summe von $\log m$ Zahlen auf die Summe von nur zwei Zahlen zu reduzieren. Diese zwei Zahlen können dann addiert werden. Das angegebene Lemma wurde aus dem Beweis von Lemma 5.8 in [SRK95] extrahiert und etwas variiert. Da doch einige Schritte angepaßt wurden, ist der Beweis in Anhang A.3 auf Seite 64 zu finden.

**Lemma 3.3.3** Die Summe von $\log m$ $O(N)$-Bit Ganzzahlen kann mittels Schwellenschaltkreis auf die Summe von zwei $O(N + \log m)$-Bit Ganzzahlen reduziert werden. Der Schaltkreis hat bei einem konstantem $d > 0$ eine Tiefe von $7d$, $O(N f_d(\log^2 m))$ Kanten und $FanIn \leq \log^2 m$.

Das folgende Lemma ist ein weiteres Design für die Reduktion von Summanden. Es ist aus [SRK95] entnommen und dort als Lemma 5.7 zu finden.

**Lemma 3.3.4 (Lemma 5.7 in [SRK95])** Gegeben seien $m$ $N$-Bit Ganzzahlen. Die Summe der Zahlen kann reduziert werden auf eine Summe von $\log m$ $(N + \log m)$-Bit Ganzzahlen. Dabei benötigt man bei einem konstanten $d > 0$ einen Schwellenschaltkreis der Tiefe $7d$ mit Kantenkomplexität $O(N2^{-d/2}m^{1+1/(2^d-1)}\sqrt{\log m})$ und $FanIn \leq m$.

Die vorangegangenen Lemmas stellen schon einige Hilfsmittel zur Verfügung. Nun muß man diese Hilfsmittel nur noch richtig anwenden, um zum gewünschten Ergebnis zu gelangen. Lemma 3.3.5 gibt uns das Instrument zur Berechnung der gewichteten Summe der linearen Schwellenfunktion.

**Lemma 3.3.5** Die Summe von $n$ $N$-Bit Ganzzahlen mit $N = \Omega(\log n)$ kann bei konstantem $d > 0$ durch einen Schwellenschaltkreis mit der Tiefe $O(d \log n / \log m)$ mit Kantenkomplexität $O(nN2^{-d/2}m^{1/(2^d-1)}\sqrt{\log m})$ und $FanIn \leq m$ berechnet werden.

Der Beweis des Lemmas erfolgt konstruktiv. Die Konstruktion unterteilt sich in vier Schritte. Im ersten Schritt wird die Summe der $n$ $N$-Bit Zahlen auf eine Summe von $n \log m/m$ $(N + \log m)$-Bit Zahlen reduziert. Dies erfolgt durch Anwendung von Lemma 3.3.4. Im zweiten Schritt wird Lemma 3.3.4 weiter iteriert angewendet bis man eine Summe von weniger als $\log m$ $O(n)$-Bit Zahlen erhält. In einem dritten Schritt kann durch Lemma 3.3.3 diese Summe nochmals auf eine Summe von zwei $O(N)$-Bit Zahlen reduziert werden. Ein letzter Schritt addiert diese zwei Zahlen letztendlich. Die Struktur der Reduktionen ist in Abbildung 3.3.1 auf der nächsten Seite zu sehen.

**Beweis:**
Gegeben sei die binäre Repräsentation $X_1, \ldots, X_n$ der $N$-Bit Ganzzahlen.

*Schritt 1:* Wir unterteilen die $n$ Zahlen in $n/m$ Gruppen zu je $m$ Zahlen. Dann wenden wir die Reduktion von Lemma 3.3.4 parallel für jede Gruppe an. Nach der Reduktion sind in jeder Gruppe $\log m$ Zahlen zu je $N + \log m$ Bits. Dieser Schritt benötigt für jede Gruppe einen Schaltkreis der Tiefe $7d$ mit Kantenkomplexität $O(Nf_d(m))$. Da es $n/m$ Gruppen gibt, verbleiben nach diesem Schritt $n \log m/m$ Ganzzahlen, und insgesamt benötigt man $O(nNf_d(m))$ Kanten.

*Schritt 2:* Jetzt wenden wir Schritt 1 solange an, bis wir nur noch weniger als $m$ Ganzzahlen zu summieren haben. Wir sehen daß in der $k$-ten Anwendung $O(n(\log^k m)/m^k)$ Zahlen zu maximal $(N + k \log m)$

$$
\begin{array}{cc}
X_1, \ldots, X_n & (N) \text{ Bits} \\
\Big\downarrow \text{Reduktion} & \\
X'_1, \ldots, X'_{n \log m/m} & (N + \log m) \text{ Bits} \\
\Big\downarrow \text{iterierteReduktion} & \\
X''_1, \ldots, X''_{\log m} & (N + \log n) \text{ Bits} \\
\Big\downarrow \text{BlockPartition} & \\
X'''_1, X'''_2 & O(N + \log n) \text{ Bits} \\
\Big\downarrow \text{Addition} & \\
\sum_{i=1}^{n} X_i &
\end{array}
$$

Abbildung 3.9: Summe von $n$ $N$-Bit Ganzzahlen

Bits verbleiben. Man benötigt also $k = O(\log n / \log m)$ Reduktionen und die Kantenkomplexität in der $k$-ten Reduktion ist $O((N + k \log m).n(\log m)^{k-1}/m^k.f_d(m))$. Wird die Reduktion von Lemma 3.3.4 nun nochmals angewand, so verbleiben weniger als $\log m$ Zahlen zu je $O(N + logn) = O(N)$ Bits. Wobei man einen Schaltkreis der Tiefe $7d$ mit $O(Nf_d(m))$ benötigt.

*Schritt 3:* Wir wenden Lemma 3.3.3 an und erhalten zwei $O(N)$-Bit Zahlen. Dazu benötigen wir einen Schaltkreis der Tiefe $7d$ mit einer Kantenkomplexität von $O(Nf_d(\log^2 m))$.

*Schritt 4:* Im letzten Schritt können diese zwei $O(N)$-Bit Zahlen mit dem Schaltkreis aus Lemma 3.3.1 addiert werden. Dieser Schaltkreis hat Tiefe $O(\log n / \log m)$ und $O(N(\log^* N)^2)$ Kanten.

Die Kantenkomplexität des Schaltkreises wird vom ersten Schritt dominiert, ist also $O(nNf_d(m)) = O(nN2^{-d/2}m^{1/(2^d-1)}\sqrt{\log m})$. Da jede Reduktion von *Schritt 2* $7d$ Layers benötigt, berechnet sich die Tiefe des Schaltkreises zu $O(d \log n / \log m)$.

□

Damit haben wir die Möglichkeit, beliebige boolesche lineare Schwellenfunktionen zu berechnen. Aus Kapitel 1.4 wissen wir daß die Gewichte ganzzahlig sind und $N$ beschränkt durch $O(n \log n)$. Das Ergebnis der Bemühungen ist das folgende Theorem.

**Theorem 1** *Jede lineare Schwellenfunktion auf $n$ Variablen kann bei einem konstanten $d > 0$ mittels Schwellenschaltkreis der Tiefe $O(d \log n / \log m)$ mit $O(n^2 \log n . 2^{-d/2} m^{1/(2^d-1)} \sqrt{\log m})$ Kanten und $Fan-in \leq m$ berechnet werden.*

**Beweis:** Die zu berechnenede lineare Schwellenfunktion sei gegeben durch

$$f(x_1, \ldots, x_n) = sign(\sum_{i=1}^{n} w_i . x_i - \Theta)$$

. Die Gewichte $w_i$ kann man binär kodieren als $w_i = \sum_{j=0}^{O(n \log n)} (2^j w_{i_j})$ mit $w_{i_j} \in \{0, 1\}$. Negative Gewichte können im Zweierkomplement dargestellt werden. Man nehme den Summations-Schaltkreis aus Lemma 3.3.5, der $n$ $O(n \log n)$-Bit Zahlen $X_1, \ldots X_n$ addiert. Jedes $x_i$ wird mit der $j$-ten Stelle von $X_i$ verbunden falls $w_{i_j} = 1$. Die restlichen Inputs des Schaltkreises bleiben konstant 0. Dann berechnet der Summations-Schaltkreis die gewichtete Summe $\sum_{i=1}^{n} w_i . x_i$. Ein folgender Schaltkreis vergleicht diese Summe mit $\Theta$. Benutzt man dazu eine Variation des Schaltkreises LNB aus Kapitel 3.2.2 mit $Y = \Theta$, so benötigt dieser Schaltkreis $O(n \log n)$ Kanten bei einer Tiefe von $O(\log n / \log m)$. Die Komplexität wird also vom Summations-Schaltkreis bestimmt, wobei $N = n \log n$ zu setzten ist.

□

Bei konstantem $d$ und $m$ benötigt der Schaltkreis $O(n^2 \log n)$ Gatter mit beschränktem Fan-in. Benötigt die lineare Schwellenfunktion nur polynomielle Gewichte, so reduziert sich die Kantenkomplexität des Schaltkreises auf $O(n \log n)$ bei gleichbleibender Tiefe.

### 3.3.2 Symmetrische und allgemeinsymmetrische Funktionen

Zwei wichtige Klassen von booleschen Funktionen sind symmetrische und allgemeinsymmetrische Funktionen. Eine symmetrische Funktion hängt nur von der Summe der Inputbits ab, liefert daher für alle Permutationen der Inputs das selbe Ergebnis(daher wird sie symmetrisch genannt).

**Definition 3.3.1 (symmetrische Funktion)**
Eine boolesche Funktion $f : \{0, 1\}^n \to \{0, 1\}$ ist *symmetrisch* falls es eine Funktion $\hat{f} : Z \to \{0, 1\}$ gibt, sodaß gilt:

$$f(x_1, \ldots, x_n) = \hat{f}\Big(\sum_{i=1}^{n} x_i\Big)$$

□

Allgemeinsymmetrische Funktionen erweitern diesen Ansatz auf gewichtete Summen, wobei die Summe polynomiell beschränkt ist.

**Definition 3.3.2 (allgemeinsymmetrische Funktion)**
Eine boolesche Funktion $f : \{0,1\}^n \rightarrow \{0,1\}$ ist *allgemeinsymmetrisch* falls es eine Funktion $\hat{f} : Z \rightarrow \{0,1\}$ und positive Ganzzahlen $w_i$ mit $\sum_{i=1}^n w_i = O(n^c)$ für eine Konstante $c > 0$ gibt sodaß gilt:

$$f(x_1, \ldots, x_n) = \hat{f}\Big(\sum_{i=1}^{n} w_i x_i\Big)$$

□

Diese Definitionen sind oft in der Literatur zu finden. Wir wollen hier ermitteln, mit welchem Aufwand solche Funktionen mit beschränktem Fan-in zu realisieren sind. Dazu benötigen wir noch eine Definition die die Schreibarbeit erleichtert, aber nicht in der Literatur zu finden ist.

**Definition 3.3.3 ($c$-symmetrische Funktion)**
Eine boolesche Funktion $f : \{0,1\}^n \rightarrow \{0,1\}$ ist *$c$-symmetrisch* falls es eine Funktion $\hat{f} : Z \rightarrow \{0,1\}$ und positive Ganzzahlen $w_i$ mit $\sum_{i=1}^n w_i = O(n^c)$ gibt sodaß gilt:

$$f(x_1, \ldots, x_n) = \hat{f}\Big(\sum_{i=1}^{n} w_i x_i\Big)$$

Bei der $c$-symmetrischen Funktion wurde also nur die Ordnung der Summe festgelegt, von der sie abhängt. Die Vereinigung über alle $c$ ergibt die Menge der allgemeinsymmetrischen Funktionen. Weiters sind die symmetrischen Funktionen eine Teilmenge der 1-symmetrischen Funktionen.

Bei $c$-symmetrischen Funktionen sind die Gewichte der Summe auf $O(n^c)$ beschränkt. Man kann den Schaltkreis zur Addition von $n$ $N$-Bit Zahlen aus Lemma 3.3.5 auf Seite 41 verwenden um die Summe zu berechnen. Das folgende Lemma gibt uns die Möglichkeit, aus dieser Summe den Funktionswert zu erhalten.

**Lemma 3.3.6** Sei $f(x_1, \ldots, x_n)$ eine $c$-symmetrische Funktion. Wenn die $\log(O(n^c))$-Bit Gewichtssumme $\sum_{i=1}^n w_i x_i$ gegeben ist, kann diese Funktion durch einen AND-OR Schaltkreis der Tiefe $O(c \log n / \log m)$ mit Kantenkomplexität $O(cn^c \log n)$ und $FanIn \leq m$ berechnet werden.

Der Beweis stützt sich auf die Tatsache daß man eine Funktion von nur wenigen Bits berechnen muß. Man berechnet die Funktion durch die bekannte "Sum of Products"-Architektur.

**Beweis:** Wenn die Summe gegeben ist, muß man eine Funktion von $\log(O(n^c))$ Variablen berechnen. Der nötige Schaltkreis hat eine "Sum of Products"-Architektur. Dazu benötigt man weniger als $2^{\log(O(n^c))} = O(n^c)$ Summen(AND-Gatter mit je $O(c \log n)$ Inputs) und ein OR-Gatter mit $O(n^c)$ Inputs. Für beschränkten Fan-in kann man diese Gatter wieder als Bäume implementieren. Man erhält einen Schaltkreis mit $O(cn^c \log n)$ Kanten der Tiefe $O(c \log n / \log m)$.

□

Zur Berechnung der Summe muß man $n$ $O(c \log n)$-Bit Zahlen addieren. Mit Lemma 3.3.5 hat der Summations-Schaltkreis Tiefe $O(d \log n / \log m)$ und $O(cn \log n . 2^{-d/2} m^{1/(2^d-1)} \sqrt{\log m})$ Kanten. Je nach $c, m$ und $d$ dominiert die Kantenkomplexität des Summations- oder SOP-Schaltkreises.

**Theorem 2** *Jede c-symmetrische Funktion von $n$ Variablen kann für ein konstantes $d > 0$ durch einen Schwellenschaltkreis mit $FanIn \leq m$ und Tiefe $O((c+d) \log n / \log m)$ berechnet werden. Die Kantenkomplexität ist*

$$\begin{array}{ll} O(c.n^c \log n) & \textit{für} \quad m^{1/(2^d-1)} \sqrt{\log m} = o(n^{c-1}) \\ O(cn \log n . 2^{-d/2} m^{1/(2^d-1)} \sqrt{\log m}) & \textit{sonst} \end{array}$$

Obwohl Theorem 2 auch für $c = 1$, und damit für symmetrische Funktionen, gilt kann man für diese noch eine Verbesserung erzielen. Dies rührt daher, daß symmetrische Funktionen eine echte Teilmenge von 1-symmetrischen Funktionen sind. Bei der Summenberechnung muß nur die Summe von $n$ Zahlen zu je *einem* Bit berechnet werden. Dazu benutzt man einen Schaltkreis um $n$ Bits zu addieren. Dieser Schaltkreis hat eine Kantenkomplexität von $O(n.2^{-d/2} m^{1/(2^d-1)} \sqrt{\log m})$. Das führt zu folgendem Ergebnis, das in [SRK95] als Theorem 5.11 zu finden ist. Dort ist auch der Beweis angegeben.

**Theorem 3 (Theorem 5.11 in [SRK95])** *Jede symmetrische Funktion von $n$ Variablen kann für ein konstantes $d > 0$ durch einen Schwellenschaltkreis mit $FanIn \leq m$ und Tiefe $O(d \log n / \log m)$ berechnet werden. Die Kantenkomplexität ist*

$$\begin{array}{ll} O(n \log n) & \textit{für} \quad m^{1/(2^d-1)\sqrt{\log m}} = o(\log n) \\ O(n.2^{-d/2} m^{1/(2^d-1)} \sqrt{\log m}) & \textit{sonst} \end{array}$$

### 3.3.3 Konklusionen

Auch die Summe von $n$ $N$-Bit Zahlen kann in sehr lokaler Weise mit beschränktem Fan-in berechnet werden. Die Methode hierbei ist eine ständige Reduktion der $n$ Zahlen auf zuletzt nur noch zwei. Dieser Schaltkreis führt auch zu Schaltkreisen für Schwellen- und allgemeinsymmetrischen Funktion, welche in ähnlich lokaler Weise berechenbar sind. Die Anzahl der Kanten entspricht in ihrer Ordnung der Anzahl der Bits die für die Darstellung der Gewichte nötig sind.

Das Ergebnis für die Schwellenfunktionen bietet praktisch gesehen die Möglichkeit, beliebige Schwellenschaltkreise auch mit beschränktem Fan-in zu implementieren. Der Informationsverlust durch die Schwellenbildung ist allerdings kritisch. Einfacher erscheint es, die Summe durch spezielle Summengatter zu berechnen anstatt hier auf ungeeignete Schwellengatter zurückzugreifen. Dann würde ein einfacher Baum als Berechnungsgraph ausreichen.

Rein Theoretisch ist das Ergebnis aber schon von Bedeutung, da es eine Verbindung zwischen Schaltkreisen mit unbeschränktem und beschränktem Fan-in herstellt. Ähnliches gilt für allgemeinsymmetrische Funktionen.

# Kapitel 4

# Geometrisches Layout

Die Voraussetzungen im Nervensystem von Wirbeltieren ermöglichen es nicht, große Inputmengen auf eine große Anzahl von Neuronen direkt weiterzugeben. Dieses Kapitel untersucht die Möglichkeit, dies indirekt durch Weiterleitung und Verzweigung der Daten zu erreichen. Dazu werden konkrete geometrische Anordnungen und Strukturen herangezogen. Es entsteht ein Neuronen-Layout, bei dem die Verbindungen allerdings nicht unbedingt den Anforderungen eines VLSI-Layouts entsprechen müssen.

Als Gatter verwenden wir PL-Gatter (siehe dazu Abschnitt 1.5). Das PL-Gatter ist folgendermaßen definiert:

Sei $S(x) = \sum_{i=1}^{n} w_i x_i + \Theta$. Ein PL-Gatter berechnet auf $n$ Inputs $x = (x_1, \ldots, x_n)$ die Funktion $f : \mathrm{R}^n \to [-\alpha, +\alpha]$

$$f(x_1, \ldots, x_n) = \begin{cases} S(x) & \text{falls} \quad |S(x)| \leq \alpha \\ \alpha & \text{falls} \quad S(x) > \alpha \\ -\alpha & \text{falls} \quad S(x) < -\alpha \end{cases}$$

Diese Gatter haben den Vorteil daß ein Gatter sehr einfach als Baum von PL-Gattern mit kleinem Fan-in und großem $\alpha$ aufgebaut werden kann.

Das folgende Modell ist größtenteils biologisch motiviert. Im Kortex hat man eine bestimmte Dichte von Neuronen (etwa $10^5$ Neuronen pro $mm^3$), ein Fan-in von etwa $10^4$ und lokale Verbindungen mit wenigen Millimetern (maximal 5 mm für Axon+Dendritenbaum) Länge (siehe Kapitel 2.2). Dazu gibt es noch relativ wenige weitreichende Verbindungen, die hier aber außer Acht gelassen werden.

Eine erstes Modell (nur dieses wird hier behandelt) projiziert die Berechnungen auf einen 2D-Grid. Ein Schaltkreis im Grid-Modell ist charakterisiert durch folgende Eigenschaften:

- Alle Gatter befinden sich auf einer zweidimensionalen Ebene
- In einem achsenparallelen Quadrat mit Einheitskantenlänge ($a$) befindet sich maximal ein Gatter.
- Die Gatter haben einen maximalen Fan-in von $\Delta$
- Die Verbindungslänge ist beschränkt durch $B$. $B$ wird auf $a$ bezogen, d.h. $B = Kantenlänge/a$

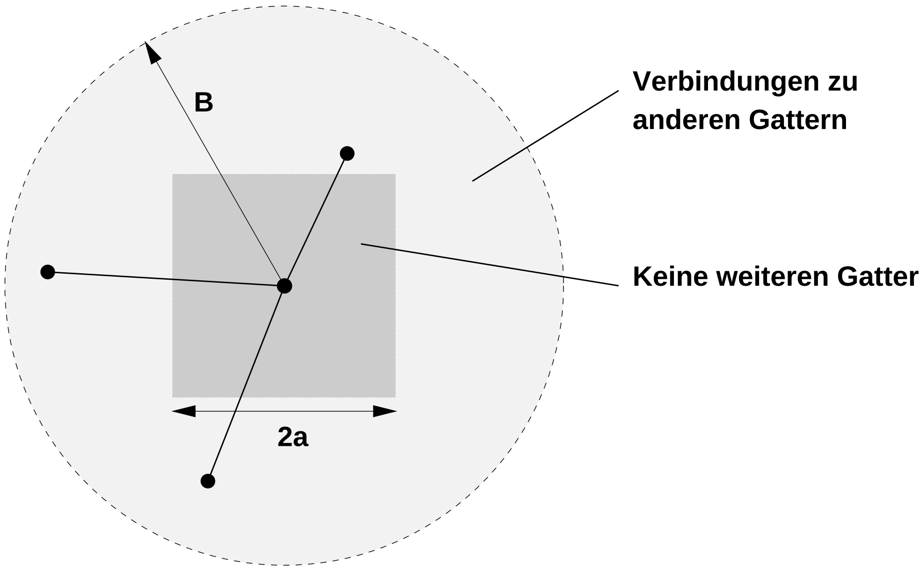


Abbildung 4.1: Gridmodell - beschränkte Gatterdichte und beschränkte Verbindungslänge

Der zweite Punkt führt bei optimaler Flächennutzung zu einer Anordnung der Gatter auf einem Grid. Man kann also die Ebene durch horizontale und vertikale Geraden mit Abstand $a$ unterteilen. In jeden Kreuzungspunkt kann man ein Gatter setzen (siehe Abbildung 4.2 auf der nächsten Seite). Wir wollen dieses Modell im Weiteren das *Gridmodell* nennen.

Im Folgenden wird die Möglichkeit untersucht, gewisse Verdrahtungsstrukturen auf so ein Modell abzubilden.

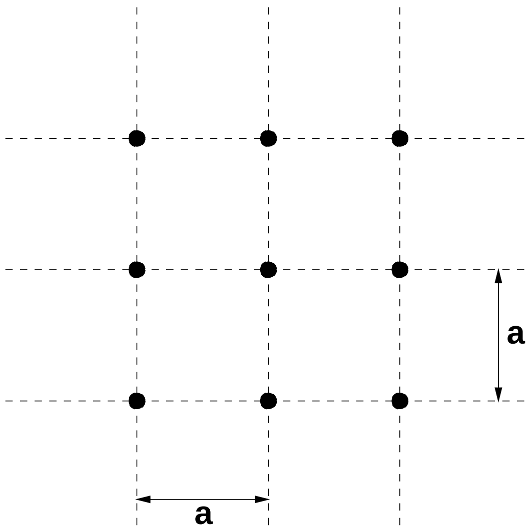


Abbildung 4.2: Gridmodell - Die Gatter können an einem Grid ausgerichtet werden

Wir bezeichnen Gatter im allgemeinen Schaltkreismodell als *A-Gatter* und den Schaltkreis als *A-Schaltkreis*. Im Gridmodell werden die Gatter als *R-Gatter* (Relaygatter) bzw. der Schaltkreis als *R-Schaltkreis* bezeichnet. Die Gatter selbst unterscheiden sich in ihrem Gattertyp natürlich nicht voneinander, die Nomenklatur soll aber die Diskussion erleichtern.

Eine wichtige Struktur wäre etwa die Vernetzung zwischen zwei Layers eines Netzwerkes, wobei alle Gatter der ersten Schicht mit allen Gattern der zweiten Schicht verbunden sind.

## 4.1 $K_{n,m}$-Struktur im Gridmodell

Der $K_{n,m}$ ist ein bipartiter Graph $G = (V, E)$. Die Knotenmenge $V$ kann man in zwei Teilmengen $V_1$, $V_2$ unterteilen sodaß $|V_1| = n$, $|V_2| = m$, $V_1 \cap V_2 = \{\}$, $V_1 \cup V_2 = V$ und keine Kanten innerhalb der Knotenmengen auftreten, aber jeder Knoten in $V_1$ mit jedem in $V_2$ verbunden ist, also

$$\begin{aligned} \forall_{a,b \in V_1} &: (a,b) \notin E, \\ \forall_{a,b \in V_2} &: (a,b) \notin E, \\ \forall_{a \in V_1} \forall_{b \in V_2} &: (a,b) \in E \end{aligned}$$

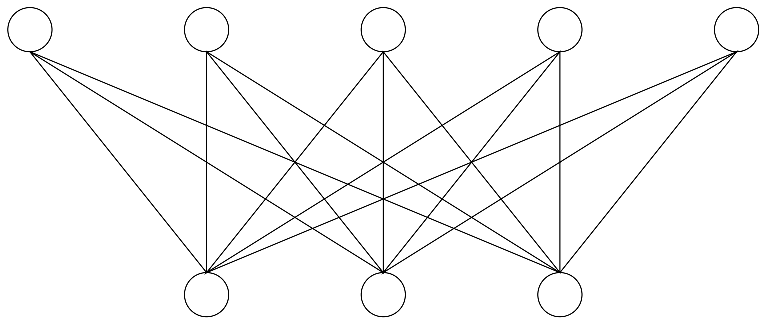

Abbildung 4.3: Der bipartite Graph $K_{5,3}$

In unserem Fall betrachten wir also einen Schaltkreis bei dem $V_1$ die $n$ Inputknoten $x_1, \ldots x_n$ und $V_2$ die $m$ Gatter des ersten Layers $y_1, \ldots, y_m$ darstellt, wobei jedes Gatter ein PL-Gatter ist. Damit sind die Kanten gewichtet. Die Gewichte der Gatter werden folgendermaßen indiziert:

$w_{j,i}$ ist das Gewicht das im
$j$-ten Outputgatter $(1 \leq j \leq m)$ das
$i$-te Inputgatter gewichtet $(1 \leq i \leq m)$

Durch die Struktur des Schaltkreises soll es möglich sein, alle Gewichtskonstellationen in den A-Gattern durch Anpassung der Gewichte in den R-Gattern zu simulieren.

Für die Lösung im Gridmodell ist allerdings nur die gewichtete Summe von Bedeutung da das Outputgatter einfach aus der Summe die PL-Funktion berechnen kann. Dieses Problem entspricht einer Matrix-Vektor Multiplikation $y = W.x$, wobei $x$ der Inputvektor, $y$ der Outputvektor (Spaltenvektoren) und $W$ die $m \times n$ Gewichtsmatrix bestehend aus den Gewichten $w_{j,i}$ ist.

Die Frage wäre nun, ob und mit welchem Aufwand diese Struktur bei sehr kleinem $B$ zu implementieren ist.

### 4.1.1 $K_{n,m}$-Struktur für kurze Verbindungen

$B = 1$ ist natürlich die minimal nötige Verbindungslänge um überhaupt einen sinnvollen Schaltkreis zu erhalten. Jedes Gatter kann demnach mit seinen horizontalen und vertikalen Nachbarn verbunden werden, wenn die Gatter genau am Grid liegen. $\Delta$ ist in diesem Fall automatisch auf vier beschränkt. Es wird sich zeigen daß bereits bei diesem minimalem $B$ eine einfache und effiziente Lösung des Problems existiert.

Bei einem $B$ von $\sqrt{2}$ und einem $\Delta$ von 8 könnte ein Gatter mit seinen acht Nachbarn in allen Himmelsrichtungen und den Diagonalen verbunden werden. Wir werden im Folgenden die Lösung für dieses Problem betrachten. Die Lösung für $B = 1$ kann mit geringem Aufwand daraus konstruiert werden.

**Lemma 4.1.1** Die $K_{n,m}$-Struktur kann im Gridmodell mit $B = \sqrt{2}$, $\Delta = 3$, $(2nm + m - n)$ Gatter ,$(6nm - 6m - 3n + 3)$ kreuzungsfreien Kanten und Tiefe $(2m + n - 1)$ realisiert werden.

Zum Beweis wird der Schaltkreis konstruiert. Die Idee ist hierbei, vertikal die Summen "aufzubauen". Mit jedem vertikalen Gatter wird ein Input gewichtet und zur Summe addiert. In jeder Spalte wird dies für ein $y_j$ gemacht. Dazwischen gibt es Spalten die die Summen wieder aufspalten um den Inputwert zu erhalten.

Der Schaltkreis besteht aus $n$ Zeilen zu je $2m - 1$ Gattern und eine weitere Zeile mit $m$ Outputgattern. Am linken Rand werden die Inputknoten vertikal aufeinanderfolgend angeordnet. Wir bezeichnen die R-Gatter mit $g_{j,i}$ und $r_{j,i}$ für $j = 1, \ldots, m$ und $i = 1, \ldots n$. Die $g_{j,i}$ sollen die Summen bilden und die $r_{j,i}$ sollen aus Summen die Inputwerte $x_i$ berechnen. Die Funktionen der Gatter sind definiert als:

$$r_{j,i} = \begin{cases} x_i & \text{für } j = 1,\ 1 \le i \le n \\ g_{j-1,i} & \text{für } 2 \le j \le m,\ i = 1 \\ g_{j-1,i} - (w_{j-1,i-1})r_{j,i-1} & \text{für } 2 \le j \le m,\ i = 2 \\ g_{j-1,i} - g_{j-1,i-1} + (1 - w_{j-1,i-1})r_{j,i-1} & \text{für } 2 \le j \le m,\ 3 \le i \le n \end{cases} \tag{4.1}$$

$$g_{j,i} = \begin{cases} r_{j,i} & \text{für } i = 1 \\ r_{j,i} + (w_{j,i-1})g_{j,i-1} & \text{für } i = 2 \\ r_{j,i} + (w_{j,i-1} - 1)r_{j,i-1} + g_{j,i-1} & \text{für } 1 \le j \le m,\ 3 \le i \le n \end{cases} \tag{4.2}$$

Die Outputgatter $y_1, \ldots y_m$ sind dann so definiert:

$$y_j = (w_{j,n} - 1)r_{j,n} + g_{j,n} \tag{4.3}$$

Es ist leicht zu sehen daß die Gatter leicht in einem planaren Graphen mit kurzen Verbindungen anordenbar sind. Die $r_{1,i}$ benötigt man physisch nicht, sie sind die Inputs, welche die erste "Spalte" einnehmen. Darauf folgt eine Spalte von $g_{1,i}$, dann eine Spalte $r_{2,i}$, $g_{2,i}$, ... . Der Schaltkreis ist für $n = 4$ und $m = 3$ in Abbildung 4.4 zu sehen. Einen Ausschnitt aus einem solchen Schaltkreis mit Gewichten findet man in Abbildung 4.5 auf der nächsten Seite.

Es wird nun behauptet daß für alle Gatter gilt:

$$r_{j,i} = x_i \quad g_{j,i} = \sum_{k=1}^{i-1} w_{j,k} x_k + x_i$$

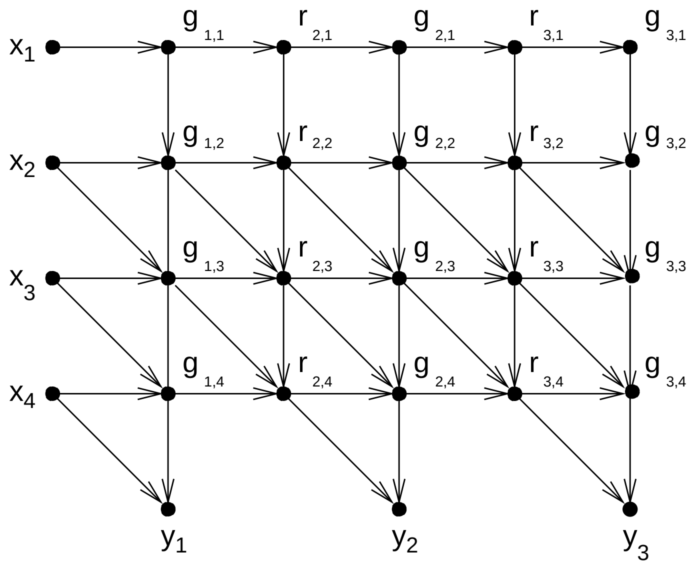


Abbildung 4.4: $K_{4,3}$-Struktur im Gridmodell

Der Beweis erfolgt mittels Induktion über $i, j$. Nachdem Verbindungen nur aufsteigend sind (d.h. Kanten nur zu Gattern mit größerem $i$ oder $j$ gehen), genügt es, zuerst die Randfälle $i = 1$ bzw. $j = 1, 2$ zu zeigen, und dann den allgemeinen Fall zu betrachten.

**Beweis:**

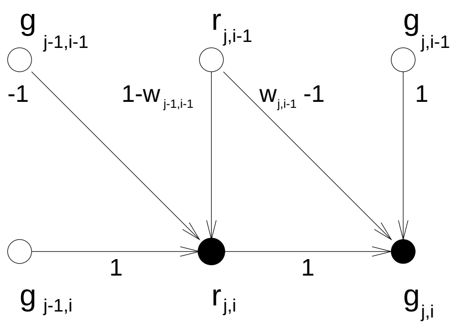


Abbildung 4.5: $K_{n,m}$-Struktur im Gridmodell. Ausschnitt mit Gewichten

Randfälle:

$r_{1,i} = x_i$ laut Definition.
$r_{j,1} = g_{j,1} = x_i$ ist offensichtlich.
$r_{j,2} = g_{j-1,2} - w_{j-1,1}r_{j,1} = x_2 + w_{j-1,1}x_1 - w_{j-1,1}x_1 = x_2$

$$g_{j,2} = r_{j,2} + (w_{j,1})g_{j,1} = x_2 + (w_{j,1})x_1 = \sum_{k=1}^{1} w_{j,k}x_k + x_i$$

Allgemeiner Fall: $3 \leq j \leq m$, $2 \leq i \leq n$

$$\begin{aligned}
r_{j,i} &= g_{j-1,i} - g_{j-1,i-1} + (1 - w_{j-1,i-1})r_{j,i-1} \\
&= \sum_{k=1}^{i-1} w_{j-1,k}x_k + x_i - \sum_{k=1}^{i-2} w_{j-1,k}x_k - x_{i-1} + x_{i-1} - w_{j-1,i-1}x_{i-1} \\
&= x_i + w_{j-1,i-1}x_{i-1} - x_{i-1} + x_{i-1} - w_{j-1,i-1}x_{i-1} \\
&= x_i \\
g_{j,i} &= r_{j,i} + (w_{j,i-1} - 1)r_{j,i-1} + g_{j,i-1} \\
&= x_i + w_{j,i-1}x_{i-1} - x_{i-1} + \sum_{k=1}^{i-2} w_{j,k}x_k + x_{i-1} \\
&= \sum_{k=1}^{i-1} w_{j,k}x_k + x_i
\end{aligned}$$

□

Damit gilt für die Outputgatter nach Formel 4.3 auf Seite 51

$$\begin{aligned} y_j &= (w_{j,n} - 1) r_{j,n} + g_{j,n} \\ &= w_{j,n} x_n - x_n + \sum_{i=1}^{n-1} w_{j,i} x_i + x_n \\ &= \sum_{i=1}^{n} w_{j,i} x_i \end{aligned}$$

und der Schaltkreis berechnet die gewünschte Funktion.

Die Größe des Schaltkreises ist leicht zu ermitteln. Man hat $2m - 1$ Spalten zu je $n$ Gatter und weitere $m$ Outputgatter, also $2mn + m - n$ Gatter. Der Platzbedarf entspricht in etwa der Größe des Schaltkreises und ist rechteckig $(n-1) \times (2m-1)$.

$(2m - 1)$ Gatter haben einen Fan-in von 1, $(2m - 1) + m$ einen Fan-in von 2 und $(2m-1)(n-2)$ einen Fan-in von 3. Das ergibt $6mn - 6m - 3n + 3$ Kanten bei einem Fan-in und Fan-out beschränkt durch 3. Die Tiefe ist durch den Weg von $x_1$ nach $y_m$ beschränkt und ist somit $2m + n - 1$.

Wie bereits erwähnt kann man daraus leicht einen Schaltkreis mit $B = 1$ konstruieren. Dazu müssen die Querverbindungen der Länge $\sqrt{2}$ durch “Treppen” ersetzt werden. Abbildung 4.6 zeigt eine solche Treppe. Statt einem Gatter benötigt man dann vier wodurch size und area um den Faktor vier steigen. Die Kantenanzahl verdoppelt sich.

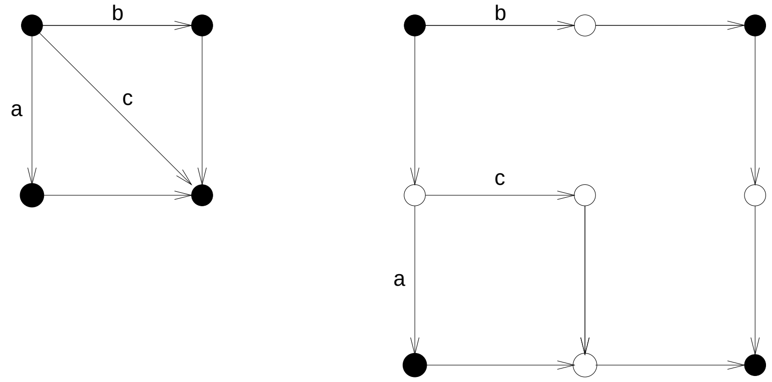


Abbildung 4.6: Gridmodell - Diagonale Verbindungen können durch “Treppen” ersetzt werden

*Bemerkung:* Die erste Zeile des Schaltkreises kann verbessert werden, der

Gewinn ist aber nur minimal. Es ist auch möglich einen Schaltkreis mit nur $mn$ Gattern und weniger als $3mn$ Kanten zu konstruieren. Dieser ist auch kreuzungsfrei, die Gewichte der R-Gatter sind aber Brüche von Gewichten der A-Gatter, was etwas lästig ist. Außerdem darf kein Gewicht Null sein.

### 4.1.2 $K_{n,m}$-Struktur für lange Verbindungen

Die vorangegangene Konstruktion scheint sich nicht auf größere Werte von $B$ verallgemeinern zu lassen. Für großes $B$ wird deshalb ein natürlicher Ansatz gewählt. Dabei werden wiederum die Inputs vertikal angeordnet, diesmal aber horizontal weitergeführt. Im Abstand von $B$ sind somit Kopien der Inputs horizontal angeordnet. Diese Gatter werden im Folgenden "Quasi-Inputs" genannt. Zwischen diesen Spalten liegen die "Summengatter". Siehe dazu Abbildung 4.7 auf der nächsten Seite.

Die Summengatter bilden von den Inputs und Quasi-Inputs in ihrer Reichweite gewichtete Summen. Die Summengatter bezeichnen wir mit $g_{j,i}$, wobei $g_{j,i}$ eine Summe für $y_j$ bildet. Wird in $g_{j,i}$ $x_k$ aufsummiert, so wird dieses $x_k$ mit $w_{j,k}$ gewichtet. Das Gatter $g_{j,i}$ gibt seinen Output an $g_{j,i+1}$ weiter. Diese Gatter sind vertikal im Abstand $B$ voneinander angeordnet. Die Summengatter zwischen zwei Quasi-Inputreihen fassen wir zu einer Blockspalte zusammen. Zusammenfassend kann man sagen:

- Horizontal werden die Inputs im Abstand $B$ weitergeführt (Quasi-Inputs).
- Zwischen den Quasi-Inputs werden gewichtete Summen gebildet (Summengatter).
- Die Teilsummen für einen Output werden vertikal aufsummiert. Zwei Teilsummen für ein $y_i$ haben jeweils Abstand $B$.
- Ein Summengatter das von allen Inputs abhängt ist Outputgatter.

Ein solcher Schaltkreis ist ausschnittsweise in Abbildung 4.8 auf Seite 57 zu sehen. Dort sieht man auch daß alle $g_{1,j}$, $g_{2,j}$ etc. innerhalb einer Blockspalte jeweils in einem "Block" zusammengefaßt sind. Ein Block besteht somit aus $B$ Zeilen und $B-1$ Spalten mit $B(B-1)$ Summengattern. Ein (Quasi-)Input gibt seinen Wert an alle Summengatter mit maximalen horizontalen und vertikalen Abstand $B/2$ weiter. Die längste Verbindung ist in diesem Fall $B/\sqrt{2}$ und der unterste Block in der Blockspalte besteht aus Outputgattern.

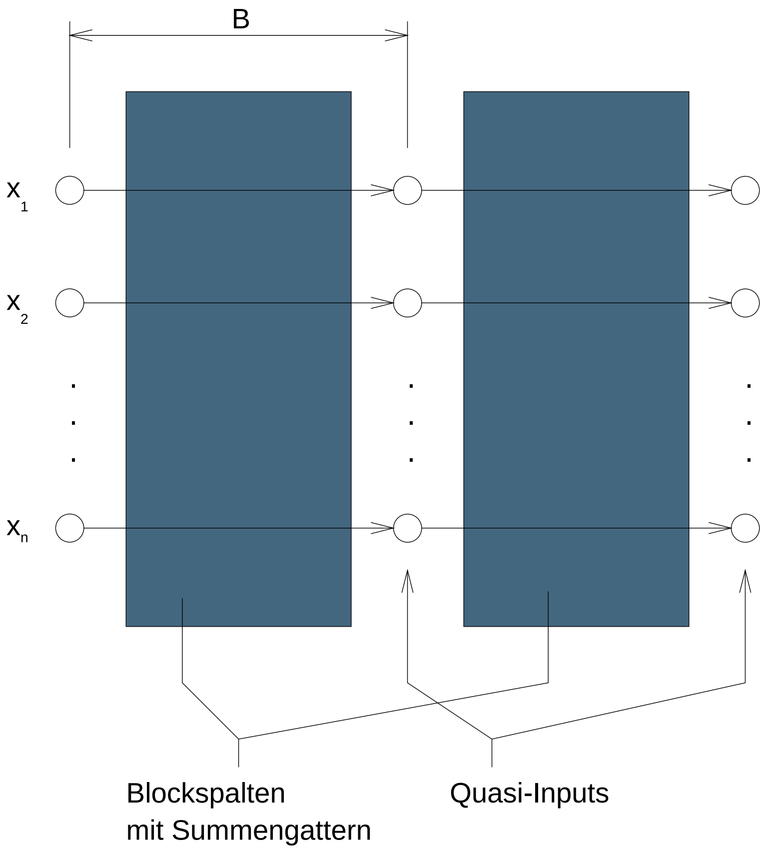


Abbildung 4.7: $K_{n,m}$-Struktur. Die Inputs werden weitergeleitet. Zwischen den Inputreihen liegen Summengatter.

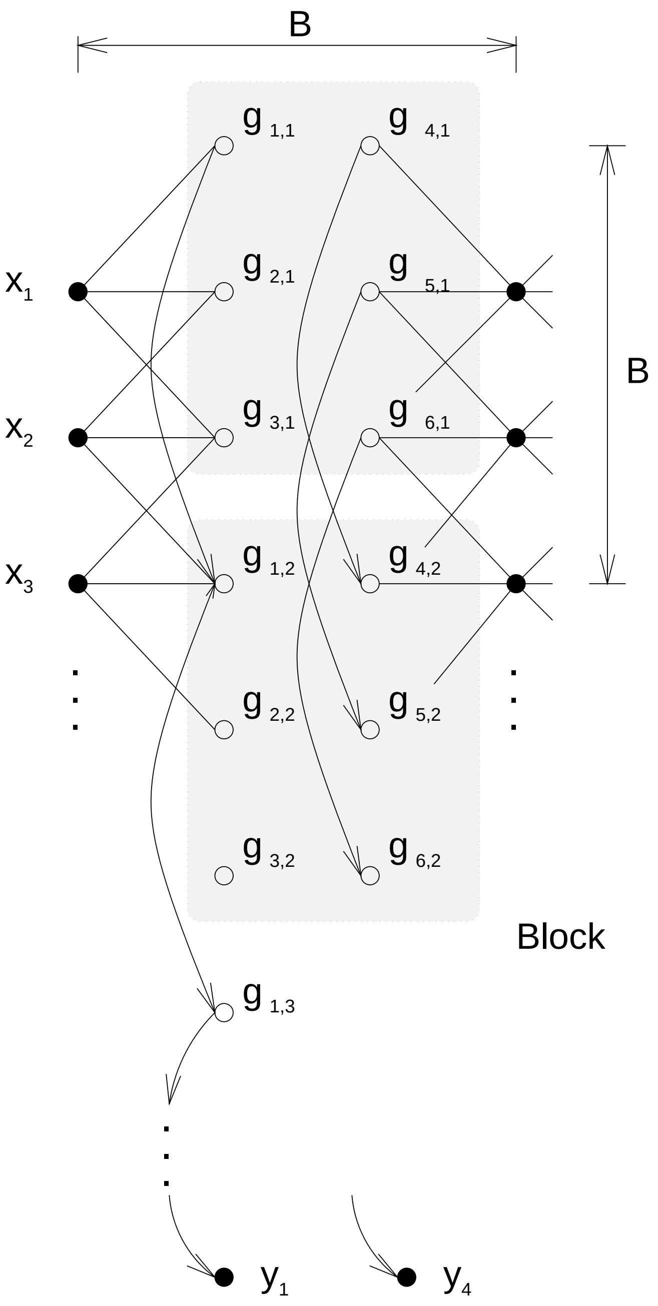


Abbildung 4.8: Ausschnitt $K_{n,m}$-Struktur. Nicht alle Verbindungen sind eingezeichnet.

Ein Summengatter berechnet also für ein $j \geq 2$

$$g_{j,i} = g_{j,i-1} + \sum_{k \in A \subseteq \{1,n\}} w_{j,k} x_k$$

Ein Summengatter bekommt Inputs von $B$ (Quasi-)Inputgatter. Nur an den Rändern (der oberste und unterste Block) ist dies nicht möglich.

*Betrachtung der Summengatter:*
Pro Blockspalte werden Summengatter für $B(B-1)$ Outputs verwendet. Die Anzahl der benötigten Blockspalten bei $m$ Outputs ist somit $\frac{m}{B(B-1)}$.

Wieviele Blöcke sind in einer Blockspalte? Wegen der Ränder (oberster bzw. unterster Block), bei denen nicht alle $B$ Inputs genützt werden, benötigt man einen zusätzlichen Block. Es gibt also $\frac{n}{B}+1$ solche Blöcke in einer Blockspalte. Sollte $B$ größer als $2n$ sein, so benötigt man nur einen Block. In jedem Block sind $B(B-1)$ Gatter.

$$size(\text{SumGatter}) = \left(\frac{n}{B}+1\right) \times B(B-1) \times \frac{m}{B(B-1)} = \frac{nm}{B} + m$$

*Betrachtung der Quasi-Inputgatter:*
Wieviele Kopien der Inputs benötigt man? Rundet man die Anzahl der Blockspalten, so bekommt man die Anzahl der Kopien der Inputsspalte. Benötigt man nur eine halbe zusätzliche Blockspalte, so sind alle Gatter von der letzten Inputkopie aus erreichbar.

$$size(\text{Quasi-Inputs}) = n \times \text{round}\left(\frac{m}{B(B-1)}\right) \leq \frac{nm}{B(B-1)} + \frac{n}{2}$$

*Größe des Schaltkreises:*

$$\begin{aligned} size \quad &= size(SumGatter) + size(Quasi-Inputs) \\ &\leq \frac{nm}{B} + \text{round}\left(\frac{m}{B(B-1)}\right) n + m \\ &= O\left(\frac{nm}{B}\right) \end{aligned}$$

Die Tiefe des Schaltkreises setzt sich zusammen aus dem Pfad zur letzten Inputkopie und dem Pfad der längsten Summationskette.

$$depth \leq \frac{m}{B(B-1)} + \frac{n}{B} + 1 = O\left(\frac{m}{B^2} + \frac{n}{B}\right)$$

Die Kanten setzen sich zusammen aus Kanten der (Quasi-)Inputs zu den Summengattern ($nm$), den horizontalen Kanten um die Inputs zu kopieren ($\leq \frac{nm}{B(B-1)} + \frac{n}{2}$) und den vertikalen um die Summen weiterzugeben ($\leq \frac{mn}{B} + m$).

$$edges \leq mn + \frac{mn}{B} + \frac{nm}{B(B-1)} + \frac{n}{2} + m = O(mn)$$

Der Fan-in ist beschränkt durch $B+1$, und falls $B$ größer als $n$ ist, durch $n+1$.

### 4.1.3 $K_{n,m}$-Struktur bei kleinem Fan-in

Im Allgemeinen soll auch der Fan-in $\Delta$ unabhängig von $B$ betrachtet werden können. Auch der Fan-out ist eine wichtige Konponente, wird hier allerdings nicht betrachtet. Bei der letzten Konstruktion hatte man einen maximalen Fan-in von $B+1$. Soll der Fan-in $\Delta$ auf kleinere Werte beschränkt bleiben, so können die Summengatter jedenfalls nur die Summe von maximal $\Delta$ Variablen berechnen.

*Betrachtung der Summengatter:*
Bei unbeschränktem Fan-in war genau ein Gatter in einem Block für einen Output zuständig. Nun benötigen wir statt einem Gatter $\lceil \frac{B}{\Delta-1} \rceil$ Gatter mit Fan-in $\Delta$. Der Schaltkreis hat also $\lceil \frac{B}{\Delta-1} \rceil$ mal mehr Summengatter, bzw. $\lceil \frac{min\{B,n\}}{\Delta-1} \rceil$ mal mehr wenn $B$ größer als $n$ sein kann.

Auch hier gilt, daß jedes Gatter in einem Block an ein Gatter im folgenden Block weiterleitet. Wurden alle Inputs von Gattern gewichtet, so wird die Summe der $\lceil \frac{B}{\Delta-1} \rceil$ Gatter mittels $\Delta$-Baum berechnet. Dieser Baum für einen Output benötigt weniger als $\frac{2}{\Delta} \lceil \frac{B}{\Delta-1} \rceil$ Gatter bei einer Tiefe von $\frac{\log \lceil \frac{B}{\Delta-1} \rceil}{\log \Delta}$.

*Betrachtung der Quasi-Inputgatter:*
In einer Blockspalte werden nun statt $B(B-1)$ Outputs nur noch $\frac{B(B-1)}{\frac{B}{\Delta-1}} = (B-1)(\Delta-1)$ Outputs berechnet. Die Anzahl der Blockspalten ist somit beschränkt durch $\frac{m}{(B-1)(\Delta-1)}$ und die Anzahl der Quasiinputgatter ist $n\text{round}\left(\frac{m}{(B-1)(\Delta-1)}\right)$.

Insgesamt ergibt sich die Größe des Schaltkreises zu

$$\begin{aligned}
size &= size(SumGatter) + size(Quasi - Inputs) + size(Baum)\\
&\leq \lceil \frac{B}{\Delta - 1} \rceil \Big( \frac{nm}{B} + m \Big) + \mathrm{round}\Big( \frac{m}{(B-1)(\Delta-1)} \Big) n + \frac{2m}{\Delta} \lceil \frac{B}{\Delta - 1} \rceil\\
&= O\Big( \frac{nm}{\Delta} \Big)
\end{aligned}$$

Die Tiefe des Schaltkreises erhöht sich ebenfalls. Es gibt mehr Blockspalten und Quasi-Inputs und einen Baum am Ende der Berechnung.

$$depth \leq \frac{m}{(B-1)(\Delta-1)} + \frac{n}{B} + \frac{\log B}{\log \Delta} = O\Big( \frac{m}{B\Delta} + \frac{n}{B} \Big)$$

### 4.1.4 Werte für Verbindungslänge und Fan-in

Dieser kurze Abschnitt soll biologisch plausible Werte für den Fan-in $\Delta$ und unsere Verbindungslänge $B$ angeben.

Aus Kapitel 2.2.2 wissen wir, daß der Fan-in in biologischen Nezten auf etwa $10^5$ beschränkt ist.

Für die Verbindungslänge soll die Anzahl der erreichbaren Neuronen herangezogen werden. Bei einer Verbindungslänge von 5mm im Kortex wird ein Fläche von $523mm^3$ erschlossen. Bei einer Neuronendichte von $9.2 \times 10^4$ sind das $4.8 \times 10^7$ ansprechbare Neuronen, sodaß bei unserer zweidimensionalen Abbildung ein $B$ von etwa $4 \times 10^3$ ausreichend sein müßte um diese Anzahl von Neuronen anzusprechen.

## 4.2 Konklusionen

Ein Modell zur Untersuchung von Verbindungsstrukturen in Neuronalen Netzen wurde vorgestellt. Dieses sollte noch erweitert werden. Um dreidimensionale Strukturen besser abbilden zu können sollte die Gatterdichte nicht so streng aufteilend definiert werden. Eine Definition einer physikalischen Dicht (Neuronen pro Fläche) wäre eine Möglichkeit. Weiters sollten wenige lange Verbindungen möglich sein.Ein dreidimensionales Modell würde der Realität sehr nahe kommen, ist aber schwer zu behandeln.

Ein indirektes Layout der wichtigsten Struktur zum Verteilen von Informationen (verbinde alle Inputs mit allen Gattern) ist möglich. Dabei bekommt man schon bei den Minimalanforderungen an Fan-in und Verbindungslänge einen Schaltkreis mit ,im Wesentlichen, gleich vielen Kanten wie bei einer direkten Verdrahtung in Form eines vollständigen bipartiten Graphen. Allerdings steigt die Anzahl der Gatter beträchtlich und vor allem die Tiefe des Schaltkreises steigt von konstant auf linear. Gerade diese lineare Tiefe dürfte sich in Bezug auf Verzögerungszeiten in biologischen Netzwerken als problematisch herausstellen. Durch erhöhte Verbindungslänge $B$ kann man die Größe und Tiefe des Netzwerkes um den Faktor $1/B$ drücken. Da $B$ eigentlich eine Fläche von etwa $B^2$ bestimmt, konnte man hoffen daß B zu quadratischen Verbesserungen führen würde. Dies wird bei dieser Konstruktion nur teilweise erfüllt. Asymptotisch verbessert sich der Schaltkreis mit steigendem $B$ nur linear.

Weiters fällt bei kleinem Fan-in $\Delta$ (kleiner als $B$) asymptotisch $B$ ganz weg. Kleines Fan-in kann bei dieser Konstruktion also nur sehr wenig durch längere Verbindungen kompensiert werden.

# Anhang A

# Beweise

## A.1 Beweis von Lemma 3.2.1

Der Beweis erfolgt für drei Fälle. Es wird gezeigt daß der zweite Ausdruck der Disjunktion genau dann erfüllt ist wenn $X = Y$ gilt (Fall a). Des weiteren ist der erste Ausdruck dann erfüllt wenn $X > Y$ (Fall b) und nicht erfüllt wenn $X \leq Y$ (Fall c). Daraus folgt die Behauptung.

**Beweis:**

Fall a) $\forall_{0 \leq i \leq n-1} : (x_i = y_i) \Leftrightarrow X = Y \Rightarrow C_n(X,Y) = 1$

Fall b) $\exists_{0 \leq i \leq n-1} \forall_{i < j \leq n-1} : [(x_i > y_i) \wedge (x_j = y_j)]$

$$\Rightarrow \sum_{k=0}^{i-1} 2^k (x_k - y_k) + 2^i (x_i - y_i) + \sum_{j=i+1}^{n-1} 2^j (x_j - y_j) > 0$$

$$\Rightarrow X > Y \Rightarrow C_n(X,Y) = 1$$

Fall c) $\neg\{\exists_{0 \leq i \leq n-1} \forall_{i < j \leq n-1} : [(x_i > y_i) \wedge (x_j = y_j)]\}$

$$\Rightarrow \sum_{i=0}^{n-1} 2^i (x_i - y_i) \leq 0$$

$$\Rightarrow X \leq Y \Rightarrow C_n(X,Y) = (X = Y)$$

□

## A.2 Beweis von Lemma 3.2.2

Wir beweisen die Korrektheit von Lemma 3.2.2 auf Seite 23 für die Fälle $X = Y$, $X > Y$ und $X < Y$(Wobei $X$ bzw. $Y$ der Einfachheit halber mit der durch sie dargestellten natürlichen Zahl gleichgesetzt werden). Damit ist für alle $X$,$Y$ die Korrektheit bewiesen.

**Beweis:**

*Fall 1:* $X = Y, \quad C_n(X,Y) = 1$

$x_i = y_i$ für $i = 0 \dots n$

$\Rightarrow C_{\Delta/2}(X^{j,\Delta}, Y^{j,\Delta}) = C_{\Delta/2}(Y^{j,\Delta}, X^{j,\Delta}) = 1 \quad j = 1 \dots n/\Delta$

Daher ist $X' = Y' = (1, \dots, 1)$ und es gilt $C_n(X,Y) = C_{\frac{2n}{\Delta}}(X', Y') = 1$

*Fall 2:* $X > Y, \quad C_n(X,Y) = 1$

$$\sum_{i=0}^{n-1} 2^i x_i > \sum_{i=0}^{n-1} 2^i x_i$$

$\Rightarrow \exists_{0 \leq a \leq n-1} \forall_{a < b \leq n-1} : (x_a > y_a) \wedge (x_b = y_b)$

Sei $X^{k,\Delta}$ der Block in dem $x_a$ vorkommt. Dann gilt

$X^{l,\Delta} = Y^{l,\Delta} \Rightarrow C(X^{k,\Delta}, Y^{k,\Delta}) = C(Y^{k,\Delta}, X^{k,\Delta}) = 1 \quad$ für $l > k$

Weiters gilt $X^{k,\Delta} > Y^{k,\Delta}$ d.h. $C(X^{k,\Delta}, Y^{k,\Delta}) = 1$ und $C(Y^{k,\Delta}, X^{k,\Delta}) = 0$. Betrachtet man die Beschaffenheit von $X'$ und $Y'$ so sieht man daß $x'_k = 1$, $y'_k = 0$ und alle höherwertigen Stellen identisch sind. Daraus ergibt sich $X' > Y'$ und $C(X', Y') = C(X, Y) = 1$.

*Fall 3* $X < Y$ ist analog zu Fall 2.

□

## A.3 Beweis von Lemma 3.3.3

Der Beweis verwendet ein Block partitions Argument. Man unterteilt die Zahlen in Blöcke gleicher Größe. Für jeden Block wird die Summe über alle Zahlen bestimmt und gezeigt, daß sich die Summe eines Blocks mit der Summe des zweitnächsten Blocks nicht überschneiden.

Dabei wird wieder in mehreren Schritten vorgegangen. Schritt 1 unterteilt jede Zahl in sehr kleine Blöcke. Schritt 2 summiert Blöcke. Schritt 3 zeigt nur noch, wie die vorherigen Summen das Ergebnis der Berechnung (zwei binär kodierte Zahlen) ergeben.

**Beweis:**

Gegeben sei die binäre Representation $X_1, \dots, X_{\log m}$ der $N$-Bit Zahlen.

*Schritt 1:* Wir unterteilen jede $N$-Bit Zahl $X_i$ in $O(N/\log\log m)$ aufeinanderfolgende Blöcke $\tilde{s}_{i,j}$ zu je $\log\log m$ Bits, für $i = 1, \ldots, \log m$ und $j = 0, \ldots, O(N/\log\log m)$. Jeder Block ist jetzt in seinem Wertebereich beschränkt durch $2^{\log\log m} - 1 = \log m - 1$. Die Summe aller j-ten Blöcke $\tilde{s}_j = \sum_{i=1}^{\log m} \tilde{s}_{i,j}$ ist somit beschränkt durch $(\log m)(\log m - 1) = O(\log^2 m)$.

*Schritt 2:* Wir stellen den Wert des j-ten Blocks (fast) unär kodiert dar. Dies geht einfach, indem man die $l$-te Stelle des Blocks ($l = 0, \ldots, \log\log m$; $l = 0$ ist das least siginificant bit) $2^l$ mal anbietet. Die Summe der $j$-ten Blöcke $\tilde{s}_j = \sum_{i=1}^{\log m} \tilde{s}_{i,j}$ ist dann die Summe von $O(\log^2 m)$ Bits. Nach Lemma 3.3.2 auf Seite 40 kann diese Summe durch einen Schwellenschaltkreis der Tiefe $7d$ mit Kantenkomplexität $O(f_d(\log^2 m))$ mit $FanIn \leq \log^2 m$ berechnet werden. Für alle $O(N/\log\log m)$ Blocksummen benötigt man also $O(f_d(\log^2 m)N/\log\log m)$ Kanten.

*Schritt 3:* Jede Blocksumme $\tilde{s}_j$ kann durch $(2\log m)$ Bits dargestellt werden, sodaß es in der binären Darstellung von $\tilde{s}_j 2^{j\log m}$ und $\tilde{s}_{j+2} 2^{(j+2)\log m}$ keine Überschneidungen gibt. Seien nun

$$\tilde{s}_{gerade} = \sum_{j \text{ gerade}} \tilde{s}_j . 2^{j\log m}$$
$$\tilde{s}_{ungerade} = \sum_{j \text{ ungerade}} \tilde{s}_j . 2^{j\log m}$$

Dann ist die binäre Darstellung von $\tilde{s}_{gerade}$ einfach die Folge der Bits in $\tilde{s}_0, \tilde{s}_2, \tilde{s}_4, \ldots$ und analoges gilt auch für $\tilde{s}_{ungerade}$. Damit gilt für die Summe der $\log m$ Zahlen $\sum_{i=1}^{logm} X_i = \tilde{s}_{gerade} + \tilde{s}_{gerade}$.

Schritt 1 und Schritt 3 benötigen keine Berechnung. Sie zeigen uns nur, wie man die Inputs aufteilen muß, bzw. wie man das Ergebnis aus der Berechnung des zweiten Schrittes erhält. Damit gelten die Komplexitäten des Schrittes 2.

□

# Literaturverzeichnis


[BBL92] P. Beame, E. Brisson, and R. Ladner. The complexity of computing symetric functions using threshold circuits. *Theoretical Computer Science*, 100(1):253–265, Juni 1992.

[Bei96] Valeriu Beiu. Digital integrated circuit implementations. In E. Fiesler and R. Beale, editors, *Hanbook of Neural Computation*, volume E1, chapter E1.4. Oxford University Press and the Institute of Physics Publishing, Oxford, 1996.

[Bei97] Valeriu Beiu. *Mathematics of Neural Networks*, chapter 12: Constant Fan-In Digital Neural Networks are VLSI-Optimal. Kluwer Academic Publishers, Boston, 1997.

[Bei98] Valeriu Beiu. On the circuit and VLSI complexity of threshold gate comparison. *Neurocomputing*, 19:77–98, 1998.

[BPVL93a] V. Beiu, J.A. Peperstraete, J. Vandewalle, and R. Lauwereins. Comparison and threshold gate decomposition. In D.J. Myers and A.F. Murray, editors, *Proc. Intl. Conf. on Microelectronics for Neural Networks*, pages 83–90. UnivEd Tech. Ltd., 1993.

[BPVL93b] V. Beiu, J.A. Peperstraete, J. Vandewalle, and R. Lauwereins. Efficient decomposition of comparison and its applications. In M. Verleysen, editor, *Proc. European Symp. on Artificial Neural Networks*, pages 45–50, 1993.

[BS91] V. Braitenberg and A. Schüz. *Anatomy of the Cortex*. Studies of Brain Function vol.18. Springer-Verlag, Berlin Heidelberg, 1991.

[CFL85] A.K. Chandra, S. Fortune, and R. Lipton. Unbounded fan-in circuits and assiociative functions. *Journal of Computer and System Sciences*, 30(2):222–234, April 1985.

[GsR] M. Goldmann, J. Håstad, and A. Razborov. Majority gates vs. general weighted threshold gates. In *Proceedings of the 7th Annual Structure in Complexity Theory Conference.*

[HA96] T. Hagauer and O. Aichholzer. Skriptum zu Entwurf und Analyse von Algorithmen. Hochschülerschaft TU-Graz, 1996.

[Ham88] D. Hammerstrom. The connectivity analysis of simple association -or- how many connections do you need. *Neural Information Processing Systems, American Inst. of Physics, New York*, pages 338–347, 1988.

[ROS94] V.P. Roychowdhury, A. Orlitsky, and K.-Y. Siu. Lower bounds on threshold and related circuits via communication complexity. *IEEE Trans. Inform. Theory*, 40:467–474, 1994.

[SRK91] K.-Y. Siu, V.P. Roychowdhury, and T. Kailath. Depth-size tradeoffs for neural computations. *IEEE Trans. Comput.*, 40:1402–1412, 1991.

[SRK95] K-Y. Siu, V. Roychodhury, and T. Kailath. *Discrete Neural Computation.* Prentice Hall, 1995.

[Ull84] Jeffery D. Ullman. *Computational Aspects of VLSI.* Principles of Computer Science Series. Computer Science Press, 1984.

[Wil90] R.C. Williamson. $\epsilon$-entropy and the complexity of feedforward neural networks. *Advances in Neural Information Processing Systems*, 3:946–952, 1990.

# Index