Ein logisch-topologischer Kalkül zur Konstruktion von integrierten Schaltkreisen.

G. Hotz B. Becker R. Kolla P. Molitor

Zusammenfassung: Es wird ein CAD-System 'CAD-IC' vorgestellt, das den Entwurf integrierter Schaltungen unterstützen soll. Der Kern des Systems besteht in einem Netzkalkül, der es erlaubt, neben der logischen Information auch geometrische Informationen zu handhaben. Dieser Kalkül besitzt verschiedene Ausprägungen, die den Entwurf auf verschiedenen Entwurfsebenen unterstützen. Das System ist um den Kalkül herum entwickelt, wie etwa ALGOL um die Numerik. Soweit das System bis jetzt entwickelt ist, betrifft es die logisch-topologische Entwurfsebene und den Übergang zur topographischen Entwurfsebene. Wir stellen hier das Konzept des Kalküls vor und erläutern an Beispielen einige Grundlagenuntersuchungen zu diesem Thema.

Abstract: We present a CAD-system 'CAD-IC', supporting the automatic design of integrated circuits. The kernel of the system is based on a "calculus of nets", which allows both, the handling of logical and geometrical information. Depending on the design-level different versions of this calculus may be adopted. The system itself is built around this calculus, as f.e. ALGOL around numerics. As far as the system is developed at the moment, it mainly deals with the logical-topological design level and the transition to the topographical level. We give the main ideas of the calculus and illustrate some basic investigations with help of examples.

Schlüsselworte

VLSI

logic design

placement

routing

layout

network topology

recursion schemes

layer assignment

geometrical problems and computations

Ein logisch-topologischer Kalkül zur Konstruktion von integrierten Schaltkreisen.

G. Hotz B. Becker R. Kolla P. Molitor

1. Einleitung:

Die Bedeutung der booleschen Algebra für den Entwurf logischer Schaltkreise ist bekannt. Sie bildet die Basis für einen Kalkül, der es erlaubt, die Logik von Schaltkreisen zu beschreiben und diese als Objekte für Berechnungen zu erschließen. Dieser Kalkül war so lange ausreichend, als die Anordnung der elementaren Bausteine und deren Verbindungsleitungen im Rechner eine untergeordnete Rolle spielten. Bei dem Aufbau komplexer Schaltungen in der Technik der Integration von vielen Halbleiterfunktionen auf einem Chip spielt die geometrische Anordnung der Leiterbahnen auf einem Chip eine hervorragende Rolle. Diese Geometrie bestimmt die Funktion des Chips. Deformationen von Leitungen vermögen mitunter die Funktion des Chips zu ändern. Aus diesem Grund ist es wünschenswert, einen Kalkül zur Verfügung zu haben, der sowohl die logische Funktion der geplanten Schaltung repräsentiert als auch geometrische Informationen über die Anordnung der Schaltung auf einem Chip zu handhaben erlaubt. Eine entsprechende Erweiterung der booleschen Algebra wurde erstmals 1965 in [Ho1] beschrieben. Es handelt sich hierbei um eine Algebra von Schaltnetzen, die in ihrem Aufbau viel an die Theorie der Zöpfe [A] erinnert. Sie unterscheidet sich von der Theorie der Zöpfe dadurch, daß die Elemente der Algebra nicht nur Geflechte von Fäden sind (Fig.1), sondern eben Netze, und daß wir nicht nur eine Operation verwenden, sondern zwei Operationen. Eine Einführung in diese Algebra der Netze findet sich auch in [Ho2] und [Bu-Hoc] und wird dort als x-Kategorie bzw. monoidale oder Kronecker-Kategorie bezeichnet.

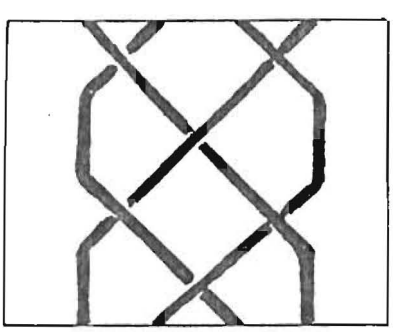


Fig.1

Die Verknüpfungen der x-Kategorie sind zwei Operationen "O" und "x", die durch Fig.2 und Fig.3 erläutert werden.

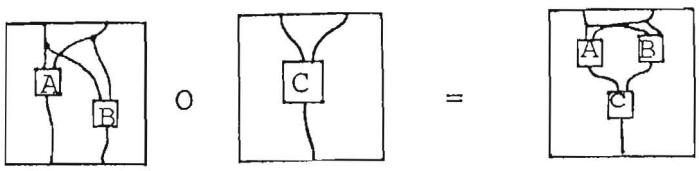


Fig.2

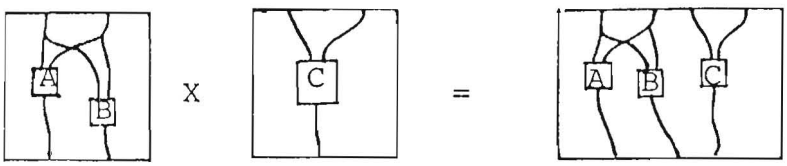


Fig.3

Das Produkt "O" ist nur dann definiert, wenn die entsprechenden Netze zusammenpassen. Das Produkt "x" ist stets definiert. Für die Zwecke des Schaltungsentwurfes in VLSI erweist sich die Operation "x" als zu speziell. Man muß auch "Verschaltungen in horizontaler Richtung" verarbeiten können. Diese Forderung führt uns zur Ersetzung von "x" durch eine Operation "Θ", die sich wie"O"verhält, nur daß "Θ" Netze in horizontaler Richtung verschaltet. Aus Symmetriegründen schreiben wir für "O" nun "Φ" und gelangen zu einer Algebra, deren Elemente in Rechtecke eingeschlossene Schaltnetze sind, mit Anschlußmöglichkeiten an den Kanten der Rechtecke. Sind F und G solche Netze, dann sind die Operationen F Θ G und F Φ G genau dann erklärt, wenn die Netze zusammenpassen. Diese Eigenschaft beschreiben wir durch vier Funktionen O(sten), N(orden), W(esten) und S(üden). Diese Funktionen geben die Anzahl und den Typ der Anschlußkontakte im Osten, Norden, Westen bzw. Süden von F an.

Diese Funktionen können z. B. die Lage der einzelnen Anschlüsse auf der entsprechenden Seite von F angeben und auch für welche Signaltypen diese Anschlüsse zugelassen sind. Mathematisch ist z.B. N(F) ein Wort über einem Typen-Alphabet T, das es uns erlaubt, diese "Anschlußspezifikationen" zum Ausdruck zu bringen.

F $\ominus$ G soll also genau dann definiert sein, wenn O(F) = W(G) bzw. F $\oplus$ G genau dann, wenn S(F) = N(G) gilt. Beide Operationen sind assoziativ, wenn sie definiert sind und es gilt

$$W(F \ominus G) = W(F),\ O(F \ominus G) = O(G),$$
$$N(F \oplus G) = N(F),\ S(F \oplus G) = S(G),$$
$$N(F \ominus G) = N(F) \cdot N(G),\ S(F \ominus G) = S(F) \cdot S(G),$$
$$O(F \oplus G) = O(F) \cdot O(G),\ W(F \oplus G) = W(F) \cdot W(G).$$

Weiter gilt

$$(F_1 \ominus F_2) \oplus (G_1 \ominus G_2) = (F_1 \oplus G_1) \ominus (F_2 \ominus G_2),$$

falls die Ausdrücke auf beiden Seiten definiert sind.

Die Operation "." ist das Monoidprodukt in T*, der Menge der Folgen über dem Typenalphabet T.

Es stellt sich nun natürlich die Frage, was unsere Netze überhaupt sind. Auf dem fertigen Chip sind sie physikalische Gebilde, das heißt geometrische mit Materie gefüllte Komplexe. In einer höheren Abstraktionsebene ist uns die Materie gleichgültig und wir haben es nur mit geometrischen Objekten im Sinne der euklidischen Geometrie zu tun. Wir sprechen in diesem Zusammenhang von der topographischen Entwurfsebene oder Beschreibungsebene. In einer nächsten Stufe abstrahieren wir von der euklidischen Geometrie und interessieren uns nur für die relative Lage der Bausteine und Leitungswege zueinander.

Wir sprechen in diesem Zusammenhang von der logisch-topologischen Entwurfs- oder Beschreibungsebene.

In dieser Darstellung unseres Konzeptes für ein Chip-Entwurfssystem beschränken wir uns weitgehend auf die logisch-topologische Ebene und den Übergang von dieser Ebene zur topographischen Ebene.

Der bis hierhin skizzierte Kalkül ist eine Bi-Kategorie [E]. Der Übergang von der topographischen zur topologischen Ebene ergibt sich in natürlicher Weise. Beim Schaltungsentwurf hat man aber den Übergang in entgegengesetzter Richtung, der mit Optimierungsfragen verbunden ist. Die logisch-topologische Bikategorie läßt auch verschiedene Ausprägungen zu, je nachdem, ob man für die Netze kontinuierliche Deformationen zuläßt, oder sich auf ein Gitter beschränkt und die Deformationen entsprechend einschränkt.

Wir geben in dieser Übersicht keine strengen Definitionen der Grundbegriffe, sondern wir versuchen, dem Leser anhand von Beispielen eine Vorstellung von unserem Ansatz zu geben.

In dem folgenden Abschnitt zeigen wir die Kraft des Kalküls, indem wir einige Schaltnetze durch rekursive Gleichungen in unserem Kalkül definieren. Danach behandeln wir die Expansion dieser Gleichungen zu Lösungen und die Auswahl von topographischen Repräsentaten unserer logisch-topologischen Netze. In einem weiteren Abschnitt wird die Verteilung der Netze auf verschiedene Layoutebenen behandelt. Im Abschnitt 5 betrachten wir Optimierungsprobleme für unsere Netze. Im abschließenden Abschnitt gehen wir auf unseren Formelübersetzer für bikategorielle Ausdrücke ein und erörtern die für die logisch-topologische Ebene eines VLSI-Entwurfssystems erwünschten Datentypen und Kontrollstrukturen eines Programmsystems zur Unterstützung des VLSI-Entwurfes.

2. Anwendungsbeispiele

Zunächst betrachten wir einige Permutationsnetzwerke, die uns für die folgenden Beispiele nützlich sein werden.

Elementare Permutationsnetzwerke sind:

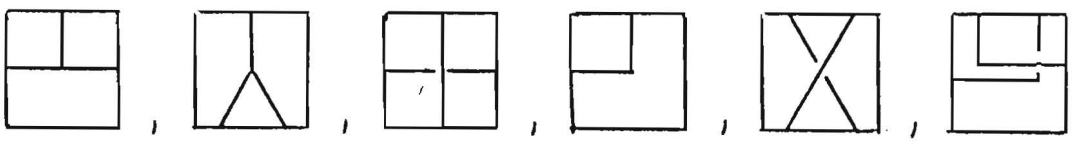


Fig. 4

Die Funktion dieser Netze geht aus der Darstellung hervor.

Wir führen als Operatoren Spiegelungen und Drehungen von Netzen ein. Ist F ein Netz, dann ist F^i die Drehung von F um 90^o gegen den Uhrzeigersinn. i ist hier das Symbol für $\sqrt{-1}$. Wir haben dann: Die Netze

$$(F^i)^i = F^{-1},\ (F^{-1})^i = F^{-i},\ (F^{-i})^i = F$$

sind Drehungen von F gegen den Uhrzeiger um 90^o, 180^o, 270^o und 360^o. Mit F^σ bezeichnen wir die Spiegelung von F an der Vertikalen. Wir haben also $(F^\sigma)^\sigma = F$. So ist $((F^i)^\sigma)^{-i}$ die Spiegelung von F an der Horizontalen.

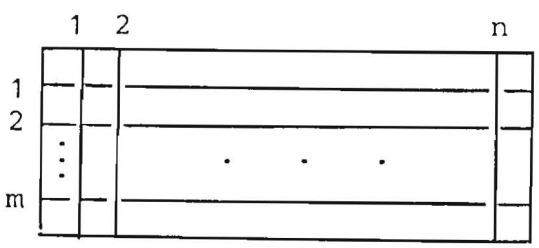


Fig. 5

Das in Figur 5 beschriebene Netz ergibt sich als

wobei wir definieren

$$\overset{n}{\Theta} F = F \Theta F \Theta \ldots \Theta F,\quad \overset{m}{\Phi} F = F \Phi \ldots \Phi F$$

(n-mal bzw. m-mal)

falls $F \Theta F$, bzw. $F \Phi F$ definiert ist.

Für $\overset{n}{\Theta}$ schreiben wir kürzer $n|$ und für $\overset{n}{\Phi}$ kürzer $\underline{n}$ für $n \in \mathbb{N}$.

Als nächstes beschreiben wir ein Netzwerk μ_n, das zwei n-Tupel von Werten mischt, wie es rekursiv durch Figur 6 definiert wird. Das Netzwerk stellt also 1 neben 1', 2 neben 2' ... und n neben n'. Wir haben

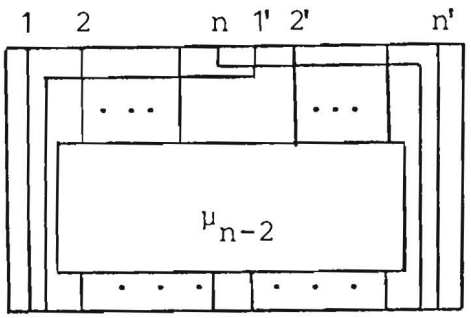


Fig.6

$\mu_1 =$ $= 2|$, $\mu_2 = 1|\ominus$ $\ominus\ 1|$,

und aus Figur 6 entnimmt man

$\mu_n = (1|\ \ominus$ $\ominus\ (\ominus^{n-2}$ $)\ \ominus$ $\ominus\ (\ominus^{n-2}$ $)\ \ominus$ $\ominus\ 1|)\ \mathbin{@}\ (2|\ \ominus\ \mu_{n-2}\ \ominus\ 2|)$

Als nächstes beschreiben wir induktiv einen Dekoder, wie er z.B. bei Speicheransteuerungen verwendet wird. Der Dekoder D_j mit j Eingängen wird verwendet, um ein Element aus 2^j Elementen auszuwählen. Dies geschieht hier so, daß j-1 Eingänge verwendet werden, je ein Element aus zwei Mengen zu 2^{j-1} Elementen auszuwählen. Die endgültige Auswahl zwischen den beiden ausgewählten Elementen trifft die j-te Leitung. (Fig. 7).

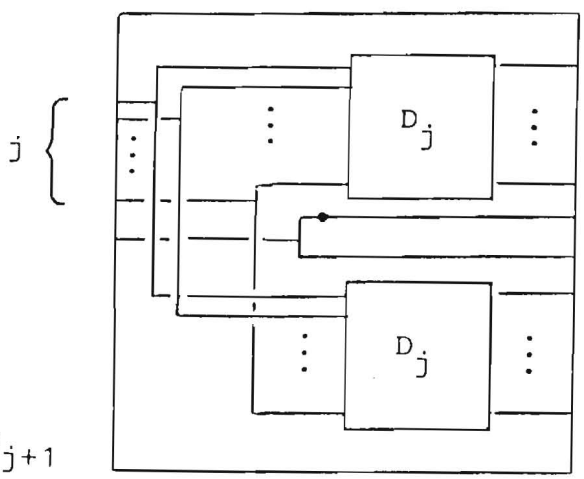


Fig. 7

Wir setzen $D_1 =$  , worin der Baustein  die Negation C bezeichnet. Wir können also schreiben

$$D_1 = \text{  }^{i} \ominus (C^i \mathbin{@} \underline{1}).$$

Nun definieren wir induktiv D_{j+1}, indem wir der Reihe nach setzen

$$V_n = (\text{} \mathbin{@} \text{} \mathbin{@} \ldots^{n-1} \ldots \mathbin{@} \text{} \mathbin{@} \text{})$$

$$V_{n,o} = V_n$$

$$V_{n,j+1} = V_n \ominus (\underline{1} \mathbin{@} V_{n,j})$$

$$D_{j+1} = V_{j+1,j-1} \ominus (\underline{j-1} \mathbin{@} D_1 \mathbin{@} \underline{j-1}) \ominus (D_j \mathbin{@} \underline{2} \mathbin{@} D_j)$$

Diese Rekursionen lassen sich leicht als korrekt beweisen und automatisch auflösen. Es scheint möglich, daß sich diese Rekursionen sogar graphisch z.B. als Figur 7 eingeben und automatisch ablesen lassen.

Wir betrachten ein n-stelliges Addierwerk für Dualzahlen der Tiefe log n, das unter dem Namen "conditional sum adder" seit 1960 ([Sp],[Sk]) bekannt ist. Hierzu erweitern wir zunächst unser Bausteinsystem, indem wir je einen Baustein für Konjunktion (k) und Disjunktion (d) zu den bereits vorhandenen hinzunehmen. Den Baustein (c) für Negation haben wir

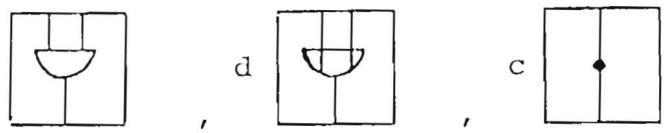


Fig. 8

bereits oben eingeführt. Diese Bausteine sind die "booleschen" Bausteine. Wir schildern zunächst die Idee des Verfahrens. Wir konstruieren ein Addierwerk A_i für Summanden aus 2^i Binärstellen, das etwas mehr leistet als ein normales Addierwerk. Es bildet nämlich zu Zahlen $a=\alpha_{m-1}\ldots\alpha_o$ und $b=\beta_{m-1}\ldots\beta_o$, $m=2^i$ die beiden Summen $c=a+b$ und $d=a+b+1$. Die Voraussetzung hierzu ist, daß die Ziffern

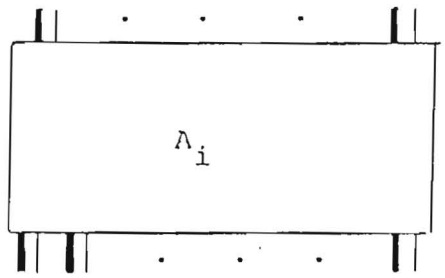


Fig.9

gemischt in der Reihenfolge α_{m-1}, β_{m-1}, α_{m-2}, $\beta_{m-2}\ldots\alpha_o\beta_o$ eingegeben werden und die Summen $c=\gamma_m\gamma_{m-1}\ldots\gamma_o$ und $d=\delta_m\delta_{m-1}\ldots\delta_o$ gemischt in der Reihenfolge $\delta_m\gamma_m\delta_{m-1}\gamma_{m-1}\ldots\delta_o\gamma_o$ ausgegeben werden.

In Figur 9 entsprechen die stark ausgezogenen Eingänge den α_j und die stark ausgezogenen Ausgänge den δ_j. Nun kann man auf folgende Weise aus den A_i ein Addierwerk A_{i+1} bauen. Man teilt die Eingänge von A_{i+1} in zwei gleich große Gruppen, die man so, wie es in Figur 11 erläutert wird, an zwei Addierwerke A_i übergibt. Dann sind die 2^i hinteren Ausgänge des rechten Addierwerkes bereits korrekt. In Abhängigkeit von den Überträgen δ_m und γ_m des rechten Addierwerkes errechnet man nun aus den Resultaten des linken Addierwerkes die richtigen Resultate für die entsprechenden Ausgänge von A_{i+1}. Dies geschieht, wie man sich leicht überlegt, für jedes Paar δ_j, γ_j des linken Addierwerkes A_i durch den Baustein S, der durch Figur 10 beschrieben wird. In dieser Figur trägt die mit σ_2 bezeichnete Leitung den Wert δ_m, die Leitung σ_1 den Wert γ_m des rechten Addierwerkes A_i. Die Leitungen σ_2 und σ_1 laufen durch den Baustein S hindurch und treten rechts wieder aus.

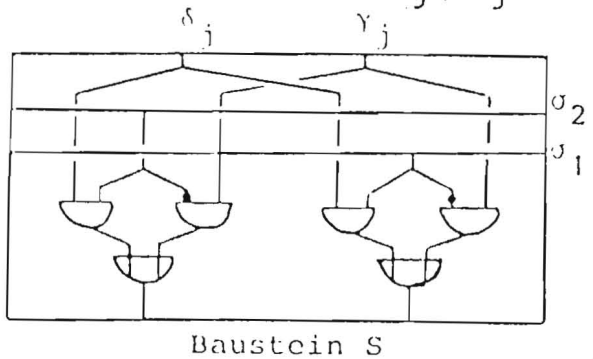
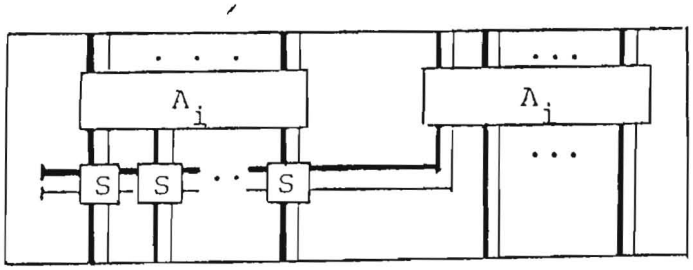


Fig.10



Hierin ist ⊟ eine Versorgung der freien Enden, d.h. mathematisch eine Projektion, die wir mit Π_2^{-i} bezeichnen.

Fig.11 Baustein A_{i+1}

Man entnimmt aus Figur 11 die Rekursion

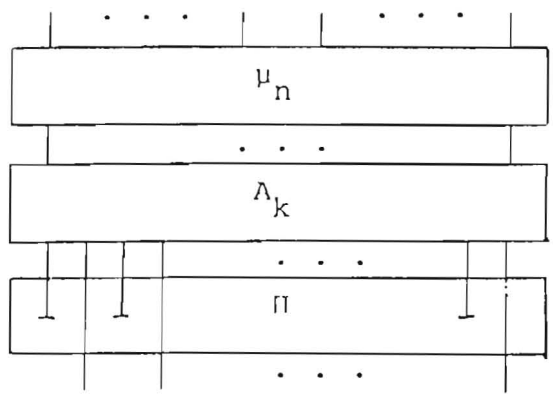
$$A_{i+1} = (A_i \ominus A_i) \oplus (\Pi_2^{-i} \ominus (\overset{m+1}{\ominus} S) \ominus \text{⊟} \ominus 2m|) \quad (m=2^i)$$

Weiter sieht man leicht, daß folgende Gleichungen gelten:

$$S = (\text{⋀} \ominus \text{⋀}) \oplus (1| \ominus \text{X} \ominus 1|) \oplus (\text{⊞} \ominus \text{⊞} \ominus (\overset{3}{\ominus} \text{⊞})) \oplus$$
$$(\overset{4}{\ominus} \text{⊞} \ominus \text{⊞} \ominus \text{⊞}) \oplus (1| \ominus \text{⋀} \ominus 2| \ominus \text{⋀} \ominus 1|) \oplus$$
$$(2| \ominus C \ominus 3| \ominus C \ominus 1|) \oplus (\overset{4}{\ominus} k) \oplus (d \ominus d)$$

$$A_o = (\text{⋀} \ominus \text{⋀}) \oplus (1| \ominus \text{X} \ominus 1|) \oplus (\text{⋀} \ominus \text{⋀} \ominus \oplus) \oplus (1| \ominus \text{X} \ominus 2|) \oplus$$
$$(d \ominus k \ominus \text{⋀}) \oplus (2| \ominus C \ominus 1|)$$

Nun können wir das gesamte Addierwerk $(AD)_k$ definieren.



Sei $n = 2^k$. Wir haben das Netzwerk zum Mischen bereits definiert. Am Ausgang benötigen wir nur die Ausgänge für a+b. Deshalb "projezieren wir die Ausgänge für a+b+1 weg. Dies geschieht durch den Bau stein Π (Figur 12).
Wir haben

Addierwerk (conditional sum) $(AD)_k = \mu_n \oplus A_k \oplus \Pi$

Fig. 12

Wir sehen wieder, daß wir durch einfache Ausdrücke, deren Umfang nicht von der Größe des Addierwerkes abhängen, das logisch-topologische Schaltnetz für nicht mehr ganz einfache Verfahren hinschreiben konnten.

Als letztes Beispiel geben wir die Beschreibung eines Speichers an, der matrixförmig angelegt ist und durch zwei Dekoder von links und von oben

angesteuert wird. Der Kürze halber beschreiben wir nur einen ROM. Wir geben hier nicht mehr die Begründung im einzelnen, sondern nur die Figuren und die rekursiven Ausdrücke, die der Leser aus den Figuren leicht gewinnt. Zunächst betrachten wir die Ansteuerung des Speichers aus 4 Zellen (Fig. 13).

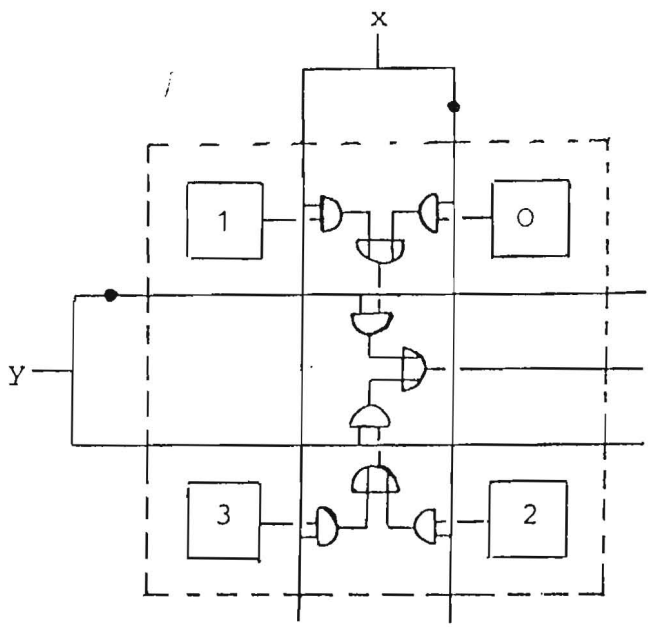


Fig. 13

Die Speicherzellen 1 und 3 bezeichnen wir mit S und die Zellen 0 und 2 durch S^{-1}.

In Fig. 13 kommt die in Fig.14 dargestellte Zelle C wiederholt vor. Die Signale 1 und 2 gehen unverändert durch die Zelle.

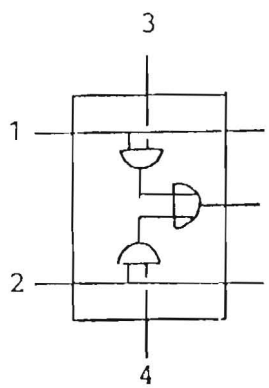
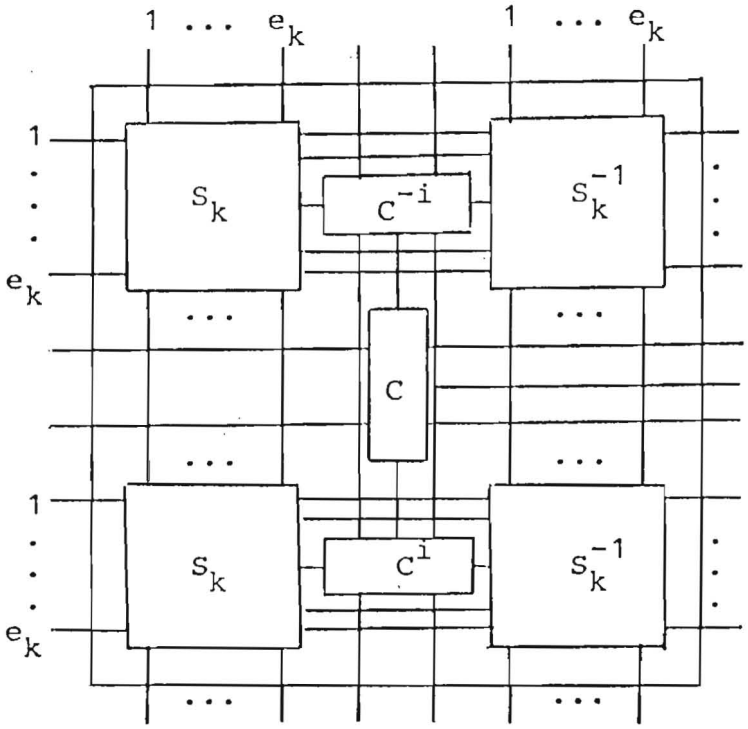


Fig. 14

Nun definieren wir induktiv einen Speicher mit 4^k Speicherzellen. Dies geschieht durch Figur 15.



Speicher S_{k+1}, $e_{k+1} = 2e_k + 2$, $e_o = o$

Fig. 15

Die Dekodierung der Adressen besorgt unser Dekodierer D_k. Wir entnehmen den Figuren

$$S_1 = (S \ominus C^{-i} \ominus S^{-1}) \oplus (\Gamma_{2,1} \ominus C \ominus \Gamma_{3,1}) \oplus (S \ominus C^{i} \ominus S^{-1}),$$

$$S_{k+1} = (S_k \ominus (\Gamma_{e_{k-1}+1,2} \oplus C^{-i} \oplus \Gamma_{e_{k-1}+1,3}) \ominus S_k^{-1}) \oplus$$
$$(\Gamma_{2,e_k+1} \ominus C \ominus \Gamma_{3,e_k+1}) \oplus$$
$$(S_k \ominus (\Gamma_{e_{k-1}+1,3} \oplus C^{i} \oplus \Gamma_{e_{k-1}+1,2}) \ominus S_k^{-1}), \text{ wobei}$$

$$\Gamma_{i,k} = \bigoplus_{i}^{k} \ominus \boxplus$$

Damit erhalten wir für den vollständigen Speicher 4^n Speicherzellen die Formel

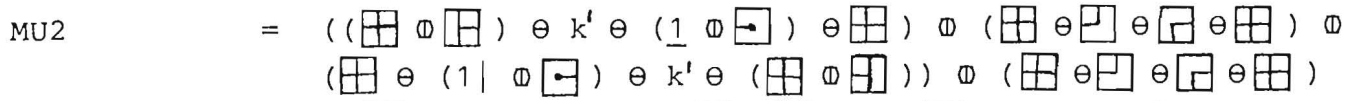
$$M_n = D_n^{-i} \oplus (D_n \ominus S_n)$$

Man hat Tiefe $(M_n) = 4 \cdot n$. Die Tiefe des Speichers entspricht also 2 · Adressenlänge.

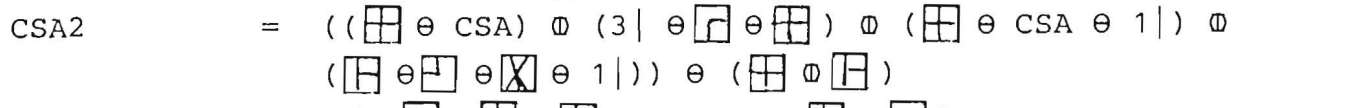
Als drittes Beispiel betrachten wir ein Multiplikationsnetzwerk. Wir wählen das schon in diesem Zusammenhang in [LuVu] betrachtete Netzwerk aus, das eine einfache Variante des Multiplizierens nach Wallace [Wa] darstellt. Die diesem Netzwerk zugrunde liegende Idee besteht in einer baumartigen Zusammenfassung von je drei Summanden zu zweien. Wir geben hier nur ein rekursives Gleichungssystem an, das das Netz definiert. Der an der Herleitung des Systems interessierte Leser sei auf [BHKM] verwiesen. Das Netz von Wallace findet man auch in [Sp].

Basiszellen:

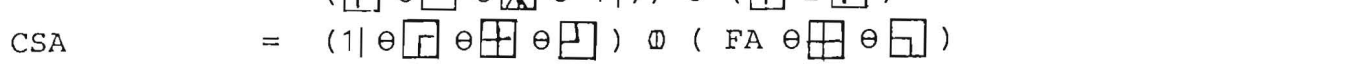
MU2 =



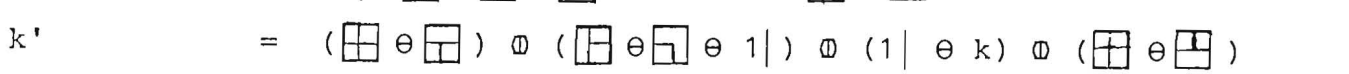
CSA2 =



CSA =



k' =

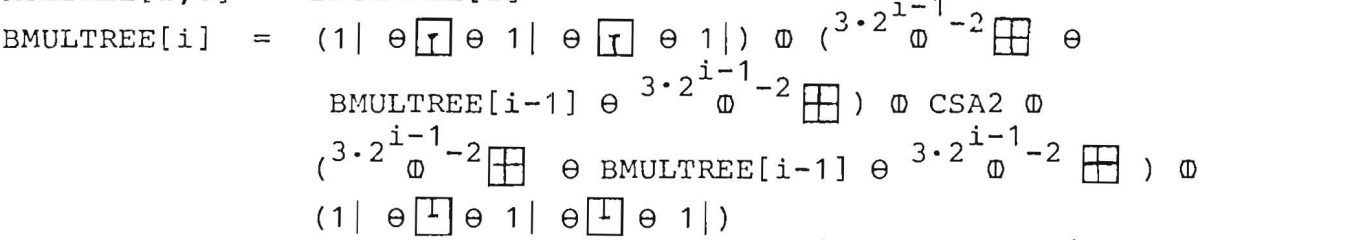


Rekursive Gleichungen:

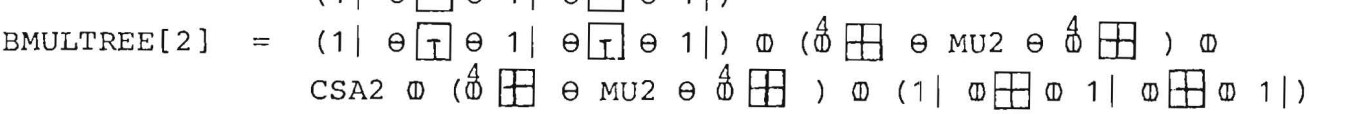
MULTREE[i,j] = MULTREE[i,j-1] ⊖ MULTREE[i,j-1]

MULTREE[i,0] = BMULTREE[i]

BMULTREE[i] =



BMULTREE[2] =



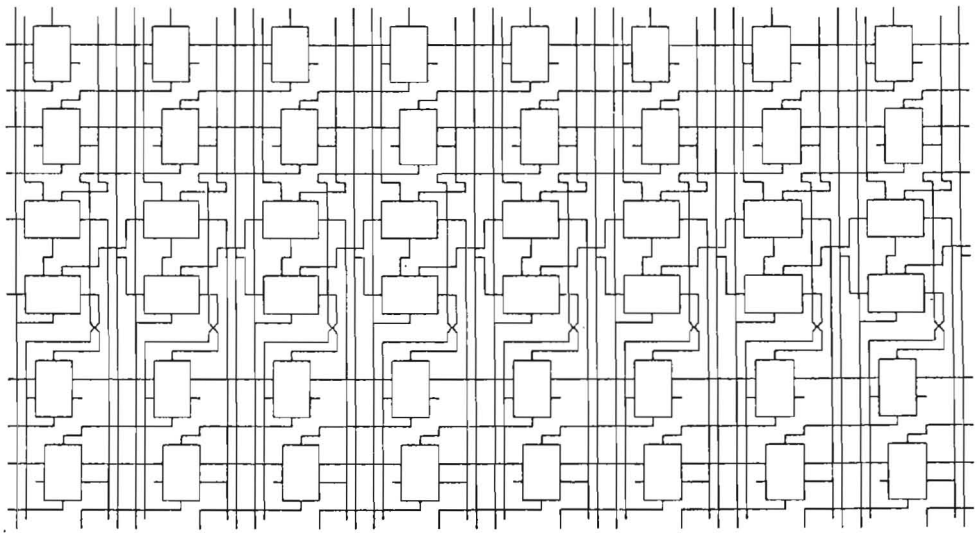


Fig.16 MULTREE[2,3] (4-Bit Multiplizierer)

Im folgenden Abschnitt behandeln wir die Expansion dieser Gleichungen und die Auswahl eines topographischen Repräsentanten in der Klasse der Lösungen in der logisch-topologischen Ebene.

3. Expansion rekursiver Netzgleichungen

Wir haben im letzten Kapitel Beispiele für die Definition von Netzen durch Gleichungen folgender Art gesehen:

$$(1)\ F_i = \exp(i); \quad (2)\ F_1 = \exp';$$

wobei exp(i) ein Ausdruck für ein Netz war, in dem nur Indizierungen mit Konstanten oder Ausdrücken über i vorkamen.
Wir benutzen also eine Menge

$$M = \{ F_i, G_i, H_i \ldots \mid i \in N_0 \}$$

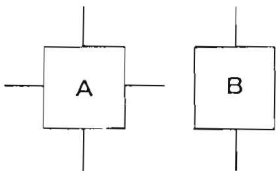
von indizierten Bausteinen, die für komplexe Netze stehen, eine Menge ME von einfachen Bausteinen wie z.B. Gatter und eine Menge GL von Gleichungen, die definieren, wie ein Baustein aus M zu ersetzen ist.
Gleichungen der Form (2) gelten explizit für ein festes F aus M, während Gleichungen der Form (1) für solche F benutzt werden, für die keine explizite Gleichung vorgegeben ist. Die Expansion eines speziellen Netzes N (etwa eines 32-bit Addierers) über Bausteinen aus M und ME geschieht nun folgendermassen: Sei e ein Ausdruck für N. Ersetze in e alle $F \in M$ durch einen in GL für F definierten Ausdruck. Man erhält so durch sukzessive Anwendung der Gleichungen immer größere Ausdrücke, bis wir schließlich (unter bestimmten Bedingungen an GL) einen Ausdruck über ME erhalten, der das Netz über Grundbausteinen darstellt. Es ist klar, daß es im allgemeinen nicht immer eine Expansion über ME geben muß. Man betrachte dazu etwa die folgende Menge von Gleichungen:

$$A_i = A_{i+1} \ominus B, \quad A_1 = k;$$

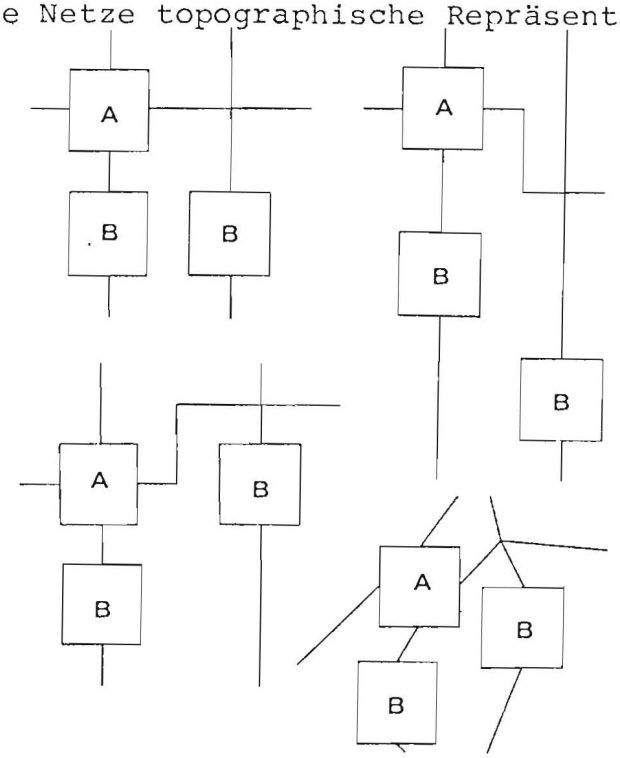
Versucht man nun $e = A_6$ zu expandieren, so wird man A_1 nie erreichen, und damit kein Netz über ME.
Weiterhin ist es vorstellbar, daß man für $F \in M$ mehrere Gleichungen hat, so daß eine Expansion nicht immer eindeutig bestimmt ist.

Wir wollen uns an dieser Stelle aber nicht weiter mit dieser Problematik auseinandersetzen, sondern davon ausgehen, daß eine eindeutig bestimmte Expansion über ME existiert und konstruiert wurde. Nun stellt sich das Problem der Auswahl eines topographischen Repräsentanten. Wir illustrieren diese Problematik anhand eines einfachen Beispiels:

Man betrachte die Bausteine



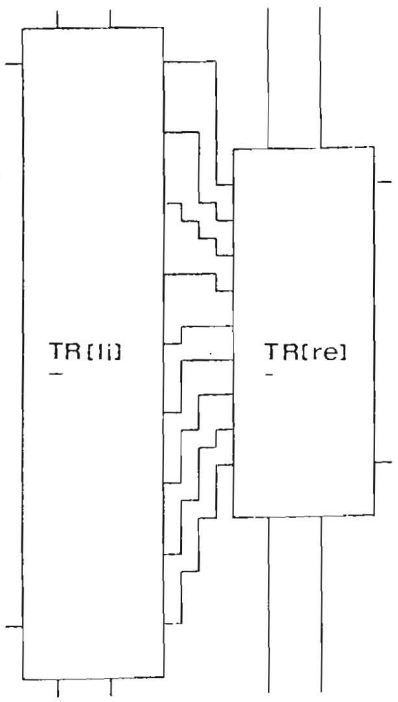
und das Netz $N = (A \ominus +) \oplus (B \ominus B)$.
Da wir nicht nur rechtwinklige Leitungsführung betrachten, sind folgende Netze topographische Repräsentanten:



Das einzige, was diese Repräsentaten gemeinsam haben, ist, daß sie sich durch gewisse Deformationen ineinander überführen lassen, ohne daß dabei neue Kreuzungen entstehen oder alte wegfallen.

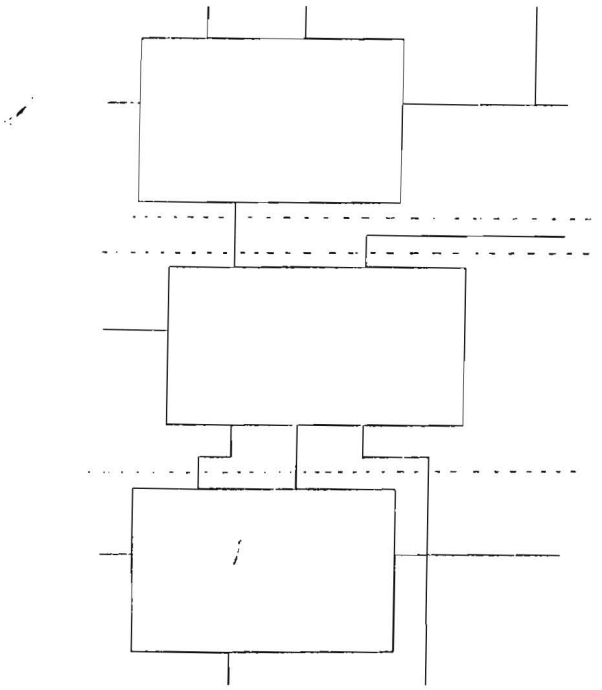
Bei der Erzeugung von Repräsentanten verfolgen wir zwei Ansätze. Der eine beschäftigt sich mit dem Problem, die Netze mit einem fest vorgegebenen Rand in der euklidischen Ebene so einzubetten, daß die Planarität erhalten bleibt und bzgl. gewisser Kriterien über Leiterlängen optimiert werden. (s. Abschnitt 5) In einem anderen Ansatz wird versucht, ausgehend von einem Ausdruck für ein logisch topologisches Netz einen Repräsentanten zu erzeugen, bei dem die Leitungen alle rechtwinklig laufen und die Größe der Grundbausteine vorgegeben werden kann. Wir werden diesen zweiten Ansatz insbesondere für Netze schildern, die durch Expansion rekursiver Netzgleichungen entstanden sind.

Eine erste Beobachtung ist, daß man mit unserem Kalkül auch topographische Netze erfassen kann, indem man die Beschreibung des Randes (N,S,W,E) nicht nur auf eine Folge von Signaltypen beschränkt, sondern auch die geometrischen Abstände der Anschlüsse erfaßt. Ein Nachteil ist, daß man bei dieser Beschreibungsart nicht nur über Funktion und Leitungsführung, sondern auch detailliert über die Geometrie nachdenken muß.
Andererseits legt einem diese Beobachtung aber auch eine Vorgehensweise zur Erzeugung eines topographischen Repräsentanten nahe: Sei e ein Ausdruck für ein logisch topologisches Netz N und sei li ⊖ re (li ⊕ re) eine Zerlegung dieses Ausdrucks in Teilausdrücke. Seien TR(li) und TR(re) topographische Repräsentanten für die durch li und re beschriebenen Netze. Dann konstruiere einen Repräsentanten für N, indem man die Operation ⊖ (⊕) geometrisch nachvollzieht. Dies kann durch planares rechtwinkliges Verdrahten in einem Verdrahtungskanal ('river routing',s.[PI83]) ereicht werden.



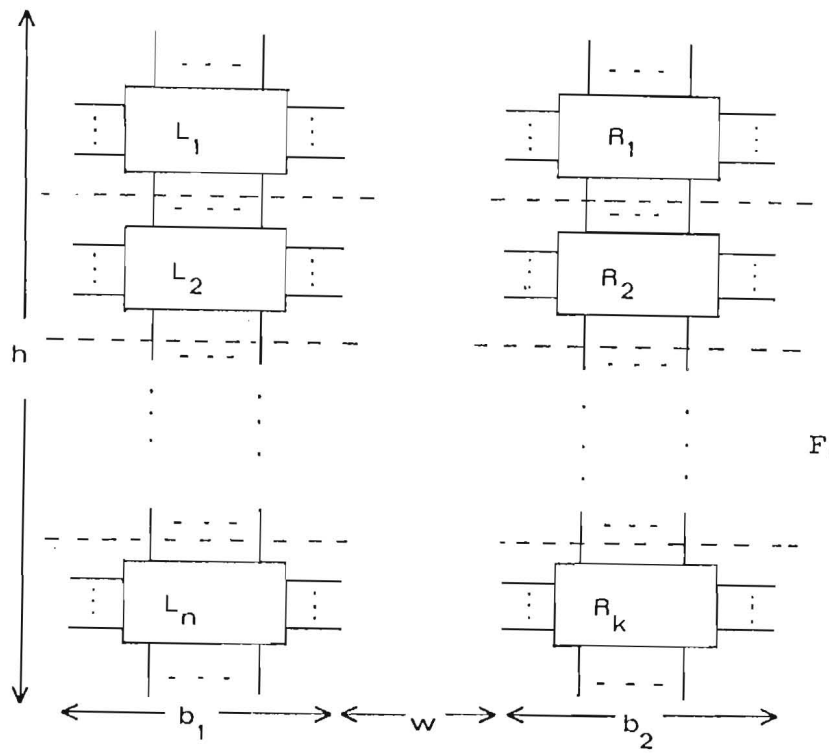
Figur 19

Wie man in Figur 19 sieht, ist dieses Vorgehen unbefriedigend. Halten wir an der vorgegebenen Größe der Grundbausteine fest, so gibt es dennoch in einem Netz gewisse Linien, entlang denen man das Netz in einfacher Weise deformieren kann. Dies sind insbesondere alte Verdrahtungskanäle. (siehe Figur 20).



Figur 2o

Man betrachte das Netz aus Figur 20. Bei diesem Netz können die Abstände zwischen zwei Anschlüssen um d Einheiten vergrößert werden, indem man alle Strecken, die von der gestrichelten Linie zwischen den Anschlüssen geschnitten werden, um d verlängert. Merkt man sich solche Linien, so stellt sich beim geometrischen Verknüpfen von TR(li) mit TR(re) folgendes Problem: (s. Figur 21)



Figur 21

Gegeben seien zwei Folgen von Blöcken

$$L_1,\ldots,L_n,\ R_1,\ldots,R_k$$

mit gewissen Mindestbreiten und festen Anschlußpositionen, sowie eine Breite w für den Verdrahtungskanal. Die Abstände zwischen den Blöcken können durch Deformation von TR(li) bzw. TR(re) vergrößert werden, was eine Vergrößerung der Mindesthöhe h zur Folge hat.
Entscheide, ob es eine Plazierung der Blöcke durch Deformation von TR(li) bzw. TR(re) gibt, so daß eine planare rechtwinklige Verdrahtung in einem Kanal der Breite w möglich ist, und, falls ja, bestimme eine solche Plazierung, die minimale Höhe h hat.
Ein solches Verfahren wird in [LeiPi] angegeben. Mithilfe dieses Verfahrens kann man nun folgende Einbettungsverfahren realisieren:

(1) Bestimme w minimal, für das es eine Plazierung gibt, so daß eine Verdrahtung möglich ist.

(2) Bestimme w so, daß, wenn h'(w) die resultierende Höhe einer optimalen Plazierung ist, die Größe

$$(b_1 + b_2 + w) * h'(w),$$

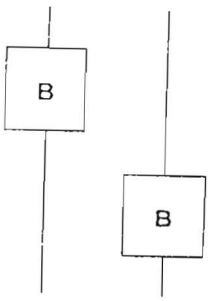
d.h. die resultierende Fläche, minimal ist.

Konstruiere dann die Einbettung mit einer optimale Plazierung für die durch (1) oder (2) errechnete Kanalbreite w.

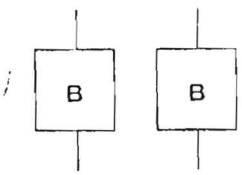
Ist das einzubettende Netz N durch Expansion rekursiver Netzgleichungen entstanden, so läßt sich das obige Einbettungsverfahren sehr zeiteffizient durchführen, da in den Gleichungen Teilnetze oft mehrfach benutzt werden. So lautet die Rekursion für den Addierer etwa: $A_{i+1} = (A_i \ominus A_i) \oplus \ldots$. Hat man einen Repräsentanten $TR(A_i)$ bestimmt, so muß man diesen bei weiteren Vorkommen nur noch kopieren und nicht mehr durch das aufwendigere Einbettungsverfahren erzeugen.

Ein Nachteil dieses Verfahrens ist es, daß der erzeugte topographische Repräsentant stark von dem Ausdruck e abhängt, d.h. es gibt für dasselbe Netz Ausdrücke, die gute und schlechte Resultate liefern.
So liefert etwa

$$(B \ominus |) \oplus (| \ominus B)$$

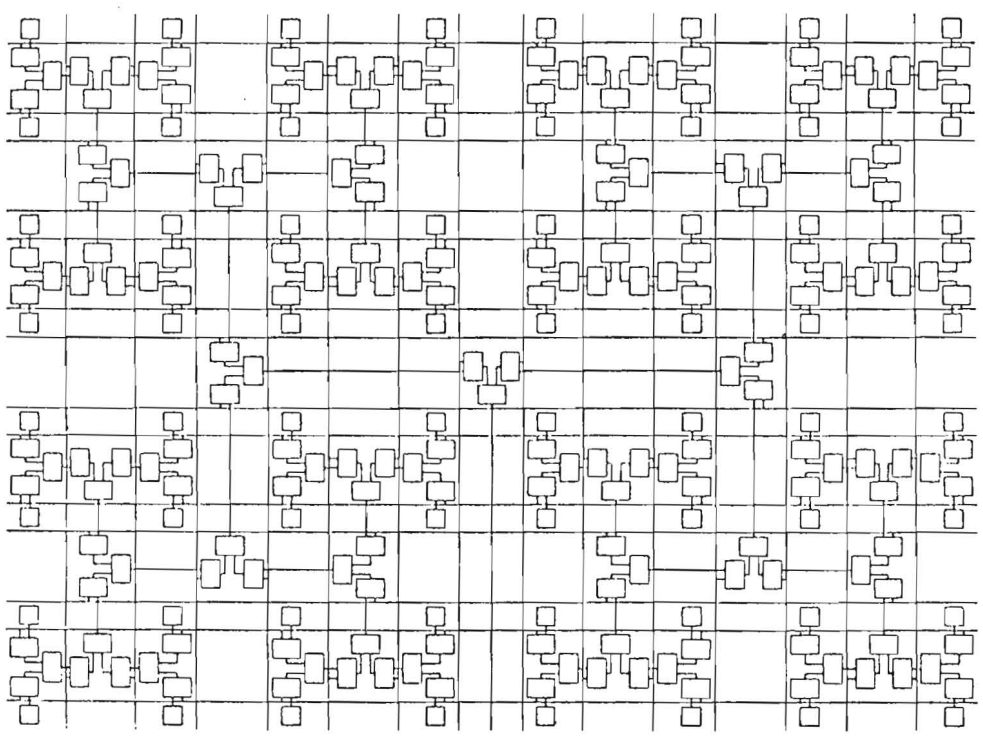


aber (B ⊖ B) = (B ⊖ |) ⊘ (| ⊖ B) liefert

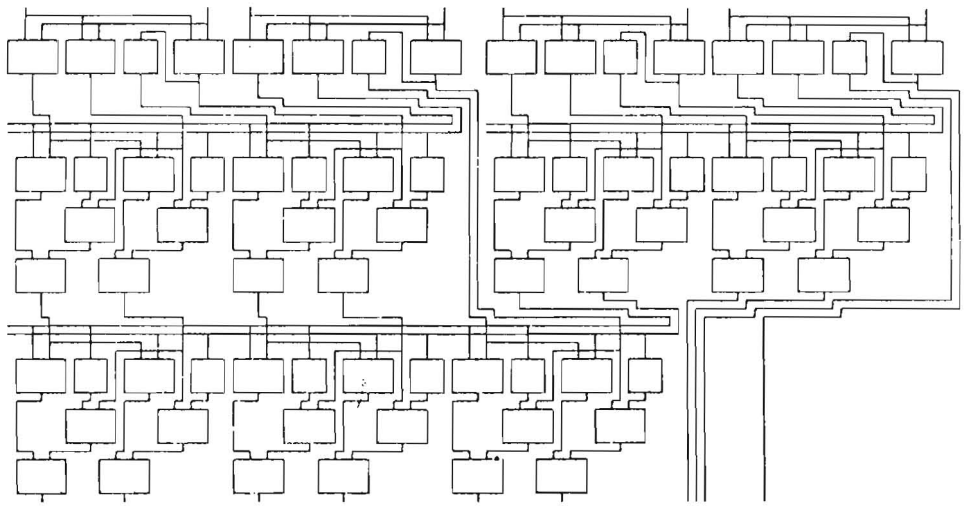


Die Frage, ob man ein effizientes Einbettungsverfahren finden kann, das solche Effekte umgeht, ist Gegenstand augenblicklicher Untersuchungen.

Zum Abschluß dieses Abschnittes wollen wir für einige Beispiele, die durch rekursive Netzgleichungen entwickelt wurden, topographische Repräsentanten vorstellen, die nach dem obigen Verfahren erzeugt wurden. In den meisten Fällen lieferte das Verfahren ohne Änderung der Rekursionsformeln zufriedenstellende Resultate. (s. Figur 16, 22, 23



Figur 22: 64-bit Speicher (erzeugt nach Verfahren (1))



Figur 23: 4-bit Addierer (erzeugt nach Verfahren (2))

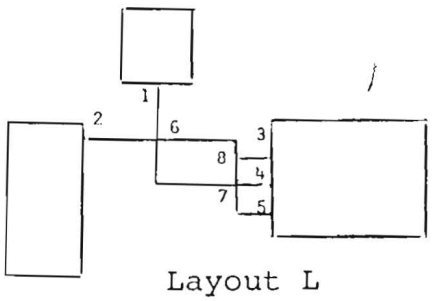
4. Einbettung der Verdrahtungsnetze in Schichten nach dem Prinzip der physikalischen Kristallisierung

Beim Übergang von der logisch-topologischen zur logisch-topographischen bzw. logisch-physikalischen Entwurfsebene müssen die Verdrahtungsnetze in Schichten (Layers) eingebettet werden. Das Ziel bei der Schichtzuweisung ist, die Leistung der Schaltkreise zu optimieren, sowie die Fabrikationskosten zu minimieren.
Durch die Einbettung der Verdrahtungsnetze in Schichten darf die Semantik des Schaltkreises nicht verändert werden: es dürfen keine Kurzschlüsse generiert werden; die Verzögerungszeiten der verschiedenen Schichten dürfen die Funktion des Schaltkreises nicht beeinflussen.
Ein Spezialfall dieses Problems ist die Aufgabe, die Leitungen in 2 Schichten einzubetten, so daß die Anzahl der Schichtwechsel, d.h. der Kontakte, minimal ist ([HaSt71]). Die Einschränkung auf nur 2 Schichten ist technologisch gesehen gerechtfertigt: um unterschiedliche Verzögerungszeiten verschiedener Layers zu vermeiden, bettet man die Verdrahtungsnetze in zwei Metallschichten ein.
In [Pi82] wird das Problem der 2-Schichtenzuweisung auf das MAXCUT-Problem für planare Graphen reduziert, das in $O(n^3)$ lösbar ist ([Ha75], [Ga73], [La76]). (n steht für die Anzahl der Überkreuzungen, Verzweigungen und Anschlüsse.) Würde man dieses Verfahren auf ein Verdrahtungsnetz der Grösse 100000 ansetzen, so benötigte man im schlechtesten Fall 30-50 Jahre, um das Ergebnis zu erhalten.
In dieser Arbeit stellen wir ein lineares probabilistisches Verfahren für das Problem der 2-Schichtzuweisung vor, das in seiner Vorgehensweise an den Kristallisierungsprozeß erinnert. Die Idee dieses Verfahrens besteht darin, daß man, ausgehend von einer beliebigen Anfangskonfiguration, sich nicht darauf beschränkt, nur lokale Verbesserungen vorzunehmen, sondern daß man mit einer gewissen Wahrscheinlichkeit die Konfiguration verschlechtern darf. Diese Wahrscheinlichkeit ist abhängig von der Grösse des Fehlers, der entsteht, und einem Steuerungsfaktor, der "Temperatur".
Die Analogie zum Kristallisierungsprozeß findet man in [KiGeVe].
In [KiGeVe], [JeGe83] wurde diese Idee angewandt auf das Problem

des Handelsreisenden, sowie auf Verdrahtungs- und Plazierungsprobleme bei Standardzellen.
Wir werden in diesem Abschnitt zuerst das Problem der Schichtzuweisung formulieren und stellen dann unser Verfahren vor und vergleichen es anhand einiger Beispiele mit anderen Verfahren.

Formulierung des Problems (vgl. [Pi82])

Sei L ein planarer Schaltkreis in der euklidischen Ebene. O.B.d.A sei L rechtwinklig verdrahtet.



Layout L

Zu L konstruieren wir in kanonischer Weise ein Graphlayout G(L):=(V(L),E(L)), wobei V(L) die Menge der in L vorkommenden Überkreuzungen, Abzweigungen und Anschlüsse ist und E(L) die entsprechenden Verbindungen.

V(L):={1,2,3,4,5,6,7,8}
E(L):={{1,6},{2,6},{3,8},
{4,7},{5,7},{6,7},
{6,8},{7,8}}

Eine Abbildung f:V(L)--->{rot,blau} nennen wir Färbung von L. Eine solche Färbung f wird wie folgt interpretiert: Ist i ein Anschlußknoten (bzw. ein Abzweigungsknoten), so erhalten die Pfadsegmente, die von i ausgehen, unter der Färbung f die Farbe f(i). Ist i ein Überkreuzungsknoten, so erhält das vertikale Pfadsegment von i die Farbe f(i), das horizontale Pfadsegment die komplementäre Farbe $\overline{f(i)}$ ($\overline{\text{blau}}$:= rot, $\overline{\text{rot}}$:= blau).

i	1	2	3	4	5	6	7	8
f(i)	r	b	r	b	r	r	r	r

Einbettung von L gemäß f:
(rot: ▬, blau: ——)

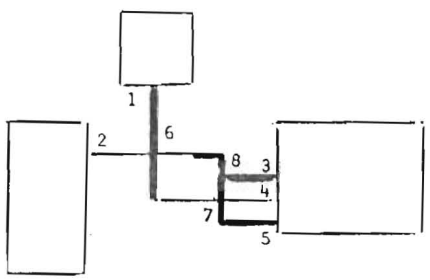


Fig.24

Wir stellen fest, daß eine Färbung f von L eine gewisse Anzahl z(L,f) von Konflikten (Schichtwechsel) zur Folge hat. In unserem Beispiel gilt z(L,f)=2.

Zur Berechnung von z(L,f) gewichten wir jede Kante k:={e1,e2}∈E(L) mit einem Paar (g(k),u(k))∈{(0,1),(1,0)}; g(k) soll genau dann 1 sein, wenn man einen Kontakt auf der Kante k benötigt, falls die Randecken e1, e2 von k gleich gefärbt sind.

Im obigen Beispiel sieht die Gewichtung wie folgt aus:

{1,6}	{2,6}	{3,8}	{4,7}	{5,7}	{6,7}	{6,8}	{7,8}
(o,1)	(1,o)	(o,1)	(1,o)	(o,1)	(1,o)	(1,o)	(o,1)

u(k) ist also genau dann 1, wenn man einen Kontakt auf der Kante k benötigt, falls die Randknoten e1, e2 von k verschieden gefärbt sind.
Definieren wir für jede Färbung f von L und jede Kante $k:=\{e1,e2\}\in E(L)$ das Prädikat $s(f,k):=(f(e1)\neq f(e2))$, dann gilt:

$$z(L,f) = \sum_{k\in E(L)} (g(k)(1-s(f,k)) + u(k)s(f,k))$$

Um also die Anzahl der Kontakte zu minimieren, muß man eine Färbung f' von L finden, so daß

$$z(L,f') = \min_{f} \{ \sum_{k\in E(L)} (g(k)(1-s(f,k)) + u(k)s(f,k))\}$$

Der probabilistische Algorithmus

Wir verwenden zur Lösung des gestellten Problems einen Algorithmus, der in seiner Vorgehensweise an den Kristallisierungsprozeß erinnert. Um aus einer saturierten Lösung einen grossen Kristall (Struktur mit niedrigster Energie) herzustellen, senkt man die Temperatur ein bißchen, wartet dann, bis Kräfteausgleich herrscht, senkt dann wieder die Temperatur usw.. Um diesen Kristallisierungsprozeß zu simulieren, muß man also die thermische Bewegung der Moleküle bei konstanter Temperatur T simulieren. Eine solche Simulation wird in [MRRTT] vorgestellt:

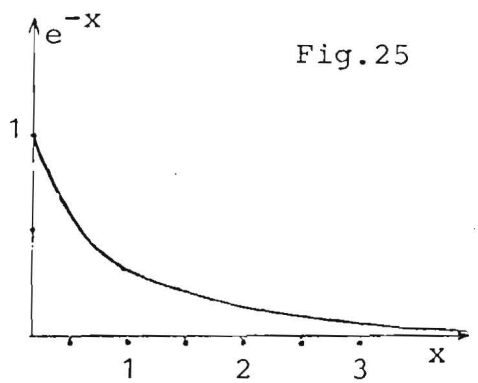


Fig.25

In jedem Schritt wird zufällig ein Molekül ausgewählt, ebenso eine Bewegung für dieses Molekül. Man berechnet dann die Energieänderung dE, falls diese Bewegung ausgeführt werden würde. Ist $dE<0$, so wird die Bewegung gemacht, ansonsten wird die Bewegung mit einer Wahrscheinlichkeit von exp(-dE/kT) ausgeführt (k Boltzmann Konstante).

Wird das Verfahren lang genug angewendet, so genügt das Verfahren der Boltzmann Verteilung, die durch exp(-E/kT) gegeben ist. Die

Simulation berechnet also mit grosser Wahrscheinlichkeit eine Konfiguration mit niedriger thermischer Energie.

Diesen Algorithmus übernehmen wir für unser Problem der Schichtzuweisung. Folgende Tabelle zeigt die Analogie:

thermische Energie	Anzahl der Kontakte
Position der Moleküle	Färbung
Bewegen eines Moleküls	Umfärben eines Knotens
Temperatur	"Steuermechanismus"

Algorithmus

(1) Berechne eine Anfangsfärbung f von L

(2) Für jede Ecke $e' \in V(L)$ berechne

$$z_{loc}(e',f) := \sum_{k:=\{e',e\} \in E(L)} (u(k)s(f,k)+g(k)(1-s(f,k)))$$

(3) Temperatur:=Tmax;

(4) for i from 1 to d(Temperatur)

/* d(Temperatur): Zeitintervall, während dem diese "Temperatur" beibehalten wird */

loop Wähle zufällig eine Ecke $e' \in V(L)$;

Berechne $z_{loc}(e',f')$, wobei $f'(e)=f(e)$ für $e \neq e'$ und $f'(e') \neq f(e')$;

$dz := z_{loc}(e',f') - z_{loc}(e',f)$;

if $(dz \leq 0)$ or $((dz>0)$ and $(random(0,1)<p(dz,Temperatur)))$

/* random(0,1): zufällige Zahl zwischen 0 und 1.
p(dz,Temperatur): Wahrscheinlichkeit mit der trotzdem umgefärbt wird. */

then f:=f'; $z_{loc}(.,f) := z_{loc}(.,f')$; fi;

pool;

(5) if Temperatur=0 then goto 6;
else Temperatur:=Temperatur-dT; goto 4;
fi;

(6) end;

Laufzeit : $O(n + \sum_{i=0}^{t/dT} d(Tmax-i*dT))$

Die Werte für p(x,y), Tmax, dT und d(Tmax-i*dT) (i=0,...,T/dT) müssen experimentell ermittelt werden.

Statistische Betrachtungen

Für die Unbekannten des Verfahrens haben wir die Werte Tmax=100, dT=25, d(i)=10·|V(L)| (∀i) und p(dz,t)=t/(100*dz) gewählt. Die Wahrscheinlichkeit, daß sich eine Konfiguration verschlechtert, ist also monoton steigend in der Temperatur, monoton fallend in dz.

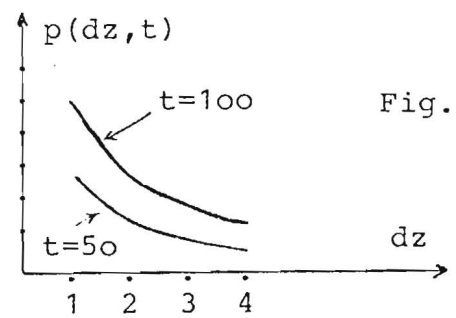


Fig.26

Das Verfahren wurde unter anderem auf die 4 folgenden Schaltkreise angesetzt:

(1) 4-Bit Multiplizierer ([LuVu])
(2) Testfreundlicher 4-Bit Multiplizierer ([Be85])
(3) 4-Bit Conditional-Sum Addierer ([BHKM])
(4) 8-Bit Conditional-Sum Addierer

	Optimum	Anzahl der Kontakte bei obigem Verfahren (Durchschnittswert)	#V(L)	Rechenzeit (SIEMENS 7561)
(1)	15	3o (Faktor 2)	712	4.5 sec
(2)	74	96 (Faktor 1.3)	734	4.9 sec
(3)	18	2o.3 (Faktor 1.1)	337	1.7 sec
(4)	49	56 (Faktor 1.1)	861	4.5 sec

Um die Güte des Verfahrens abschätzen zu können, haben wir das Verfahren mit zwei anderen Algorithmen verglichen:

Verfahren A: Wähle im vorgestellten Verfahren Tmax=0, dT=0, p(x,y)=0, d(i)=5o·|V(L)| (∀i), das Umfärben eines Knotens verschlechtert die Konfiguration nicht.

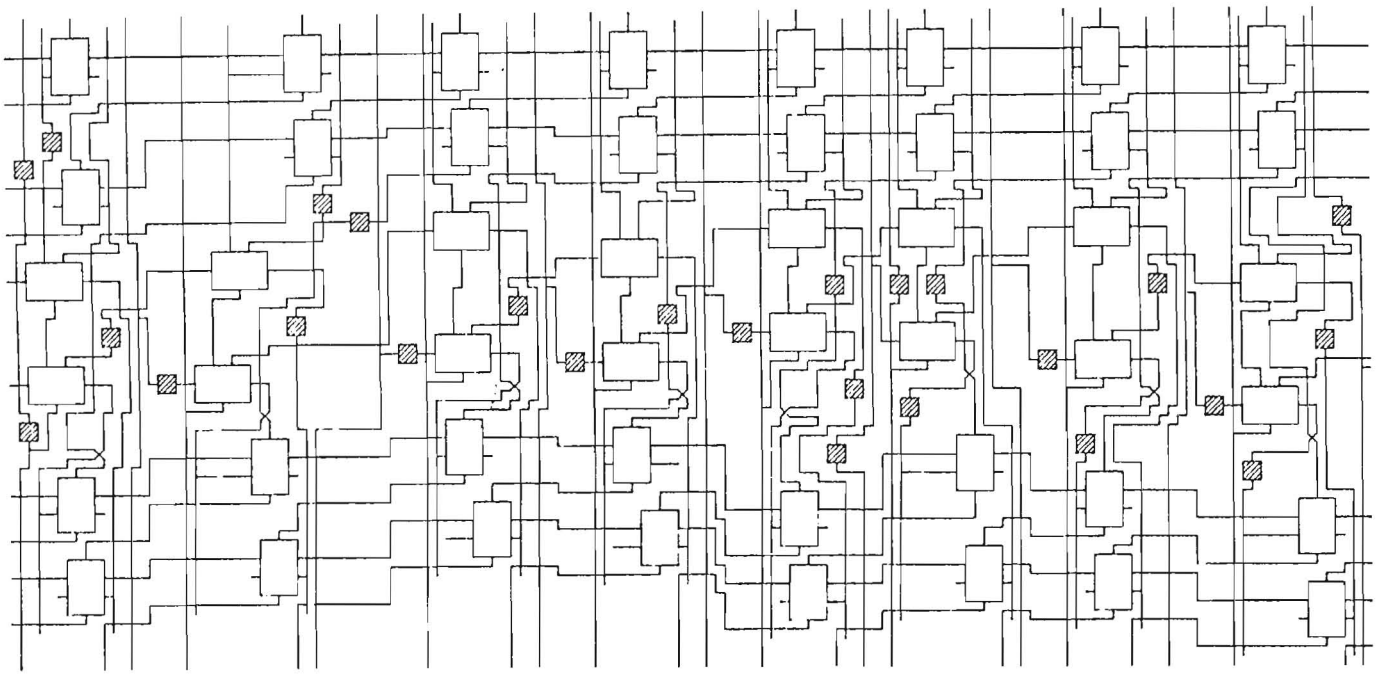
Verfahren B: ("iterative improvement")
Wähle eine zufällige Anfangsfärbung und färbe einen Knoten nur dann um, falls sich dadurch die Anzahl der Kontakte echt verringert. Tue dies solange, bis ein lokales Minimum erreicht ist.

	Verfahren A		Verfahren B	
(1)	37	(2.5)	87.3	(5.8)
(2)	102	(1.4)	178.1	(2.4)
(3)	20.5	(1.1)	29.5	(1.6)
(4)	56.4	(1.2)	79.8	(1.6)

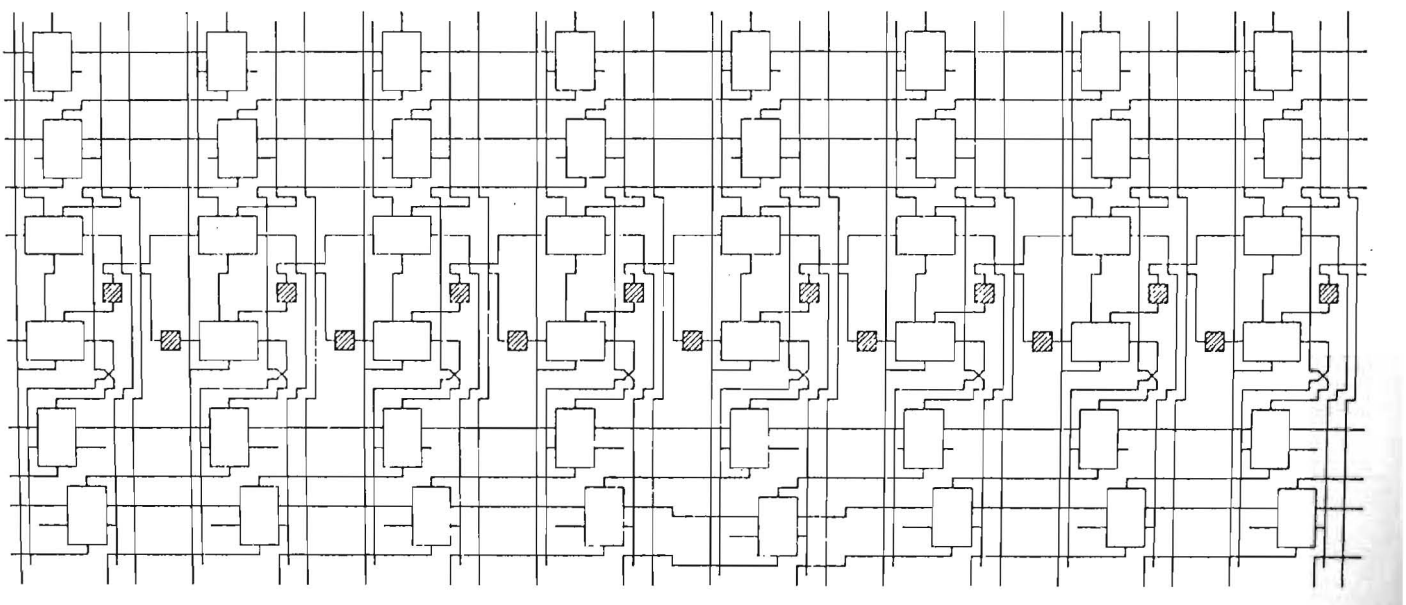
Es scheint also, daß das vorgestellte Verfahren ein schneller und (relativ) guter Algorithmus zur Einbettung der Verdrahtungsnetze in 2 Schichten ist. Vergleicht man das Verfahren z.B. mit Verfahren B, das gerne als Heuristik verwendet wird, so sehen wir, daß unsere Vorgehensweise im Mittel um einen Faktor 1.5 bis 2 besser ist.

Figur 27 und 28 zeigen zwei Beispiele. Die Schichtzuweisung in Figur 27 wurde durch den hier vorgestellten heuristischen "Kochalgorithmus" gewonnen. Figur 28 zeigt eine optimale Schichtzuweisung, wie sich an Hand eines einfachen Kriteriums nachweisen läßt. Man sieht, daß in diesem Beispiel der"Kochalgorithmus" ein recht brauchbares Resultat liefert. Leider sind wir nicht in der Lage, eine im mathematischen Sinne strenge Angabe darüber zu machen, wie gut das Verfahren ist. Experimente haben gezeigt, daß die Resultate in unseren Beispielen stark von den für den Prozeß gewählten Parametern abhängen. Wir wissen auch nicht, ob die hier recht glücklich getroffene Parameterwahl für alle übrigen Beispiele gut ist. Die Resultate rechtfertigen aber einen gewissen Optimismus in dieser Hinsicht.

4-Bit Multiplizierer von [LuVu]



(Fig.27: in 2 Schichten eingebetteter 4-Bit Multiplizierer mit 26 Kontakten. Die schraffierten Kästchen repräsentieren die Kontakte.)



(Fig.28: optimal eingebetteter 4-Bit Multiplizierer)

Das Verfahren erscheint stets dann vielversprechend, wenn die nicht optimalen relativen Optima im Verhältnis zu dem Optimum nicht zu ausgeprägt sind. Die nebenstehende Kurve in Figur 28a soll dies erläutern. Man denke sich eine Kugel auf der Kurve, die der Schwerkraft folgt. Diese Kraft würde die Kugel in ein relatives Optimum führen und dort festhalten. Nun überlagern wir diesen Vorgang mit einem stochastischen Prozeß.

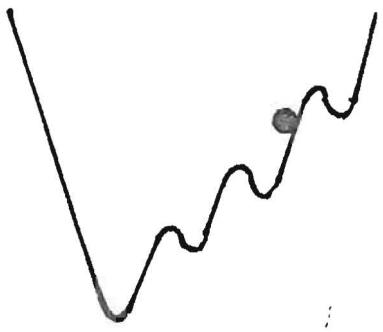


Fig. 28a

Die Kugel folgt zwar vorzüglich der Schwerkraft, aber mit einer gewissen Wahrscheinlichkeit geht sie auch die Wand hoch. Entsprechend ihrer mittleren freien Weglänge wird diese Kugel also aus gewissen "Fallen" entkommen können. Hat die Kurve, auf der die Kugel sich bewegt, eine Form wie in Figur 28a, dann wird sie bei geeigneter Einstellung der freien Weglänge in dem absoluten Minimum landen und daraus auch nicht wieder entkommen. Sind benachbarte relative Minima aber etwa gleich tief, wie dies in Figur 28b angedeutet ist, dann ist die freie Weglänge schwer so einstellbar, daß dieser Effekt eintritt.

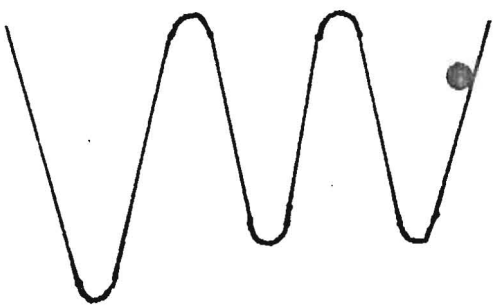


Fig. 28b

Diese Bemerkungen mögen genügen, um anzudeuten, in welcher Richtung sich Versuche bewegen mögen, um die Brauchbarkeit des"Kochverfahrens"für gewisse Problemkreise in mathematischer Präzision zu fassen.

5. Optimierungsprobleme beim Übergang von der logisch-topologischen Ebene in die topographische Ebene

Die physikalische Realisierung einer Schaltung auf einem Chip erfordert, nachdem eine logische Aufteilung der Gesamtschaltung in einzelne Teile (=Module) und eine topologische Anordnung dieser Teile in der logisch-topologischen Ebene des Systems vorgenommen wurde, die Konstruktion eines 'Layouts'; d.h. die einzelnen Module müssen geometrisch auf dem Chip plaziert und die verbindenden Leitungen müssen gemäß gegebenen Optimierungskriterien und Designregeln "verlegt" werden. Eine erste "Annäherung" an eine physikalische Realisierung soll in CAD-IC in der topographischen Ebene erfolgen.

Probleme, die hierbei auftreten, sind im Bereich der 'Design-Automation' als Plazierungs- und Routing-Probleme bekannt (siehe [Br]). Eine exakte Formulierung eines speziellen Plazierungs- oder Verdrahtungsproblems hängt zwar stark von der gewählten Technologie und den Optimierungskriterien des Designers ab; trotzdem hat sich in den letzten Jahren herausgestellt, daß es viele grundlegende Probleme (sowohl für Plazierung als auch für Verdrahtung) gibt, die NP-hart sind; d.h. schnelle Algorithmen zur Lösung sind vermutlich nicht zu erwarten (siehe [SaBh],[Do]). Will man trotzdem automatische Entwurfssysteme entwickeln, die effiziente Algorithmen benutzen, muß man entweder heuristische Verfahren entwerfen oder die Design-Methoden so einschränken, daß man für die auftretenden Probleme schnelle Lösungsverfahren angeben kann.

Eine mögliche Plazierungs- und Verdrahtungsstrategie für logisch-topologische Netze, die Rücksicht auf klassische Verdrahtungsrichtlinien ('rechtwinklige Verdrahtung') nimmt, wurde bereits in Abschnitt 2 vorgestellt.

Im folgenden wird ein weiterer Ansatz entwickelt, dem ein idealisiertes Modell des Schaltnetzes zugrunde liegt:
Wie auch in klassischen Verfahren üblich, ordnen wir einem (logisch-topologischen) Schaltnetz N einen Graphen zu, indem wir die

Bausteine des Netzes als Knoten des Graphen und Leitungen zwischen den Bausteinen als Kanten interpretieren. Soll das Schaltnetz auf einem Chip realisiert werden, gibt es einige Bausteine, die Input- und Output-Ports, die auf dem Rand der Chipfläche plaziert werden müssen. In unserem Modell nehmen wir deshalb an, daß ein Zykel c des Graphen ausgezeichnet ist und bereits als konvexes Polygon R(c) in die Ebene eingebettet ist. R(c) entspricht dann dem Chiprand, die Knoten von c entsprechen den Input- und Output-Pins. Die entstehende Struktur bezeichnen wir als 'der zu N gehörige Graph G(N) mit festem Rand R(c)'.

Ein Layout L von G(N) wird im folgenden durch eine Abbildung L der Menge der Knoten des Graphen in die reelle Zahlenebene definiert. L muß auf den Knoten von c mit der Abbildung R übereinstimmen. Implizit werden dabei die Kanten e=(v1,v2) von G(N) auf die Strecke L(v1)L(v2) := L(e) abgebildet.
Interessant sind nur solche Layouts, die alle Knoten des Graphen auf Punkte der Ebene, die innerhalb R(c) (also auf der Chip-Fläche) liegen, abbilden. Solche Layouts bezeichnen wir fortan als R(c)-beschränkt (siehe Bild 29).

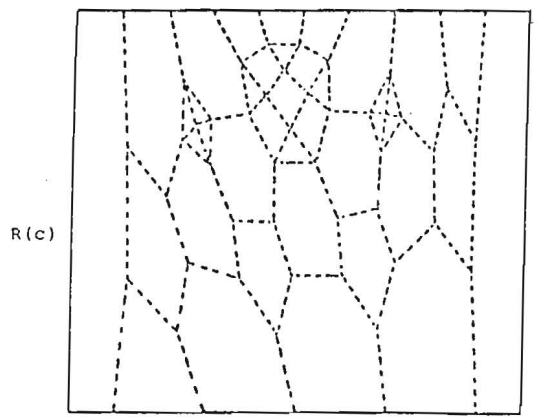


Bild 29: R(c)-beschränktes Layout von G

Im folgenden werden also Layouts konstruiert, bei denen die Positionen der Knoten (=Module) kontinuierlich auf der Chip-Fläche beweglich sind; wir haben kein festes vorgegebenes Gitter und die Kanten (=Leitungen) verlaufen in beliebige Richtungen. Dies ist zwar bei den heutigen Technologien noch nicht in voller Allgemeinheit möglich, aber Ansätze in diese Richtung sind zu beobachten (sogenannte 'Boston geometry', siehe z.B. [Bry]). (Daß

dieser Ansatz Vorteile bieten kann, wird aus folgendem Beispiel klar: Nimmt man das 'Qudratic Assignment Problem', das NP-hart für Gitter ist ([SaGo]) und "übersetzt" es in das Modell 'Graph mit festem Rand', dann zeigt sich, daß das optimale Layout in weniger als kubischer Zeit (in Abhängigkeit von der Anzahl der Knoten des Graphen) berechnet werden kann.)

Bei der Konstruktion einer Einbettung gibt es viele verschiedene (sich unter Umständen widersprechende) Ziele, die zu berücksichtigen sind. Sie alle in einem Algorithmus zu berücksichtigen, scheint nicht möglich. Tatsächlich wird in der Regel eine Matrikbenutzt, die durch die Verbindungen zwischen den Modulen definiert ist und dann in einem entsprechenden Verfahren optimiert wird. Eine Matrik, die erfahrungsgemäß viele der obigen Ziele gut berücksichtigt und deshalb häufig benutzt wird, ist die gewichtete Gesamtlänge der Verbindungen (siehe [Br],[SaBh]).

Aus diesem Grunde definieren wir:

R+ bezeichne die positiven reellen Zahlen (einschließlich 0) und f: R+ -> R+ sei eine stetige monoton wachsende Funktion.

Dann sind $cost(L,f) := \sum_{e \in E} f(|L(e)|)$ die Kosten von L bzgl. f. (|L(e)| := euklidische Länge von L(e))

L' heißt optimales Layout von G bzgl. f, falls
cost(L',f) =
inf{cost(L,f)|L ist R(c)-beschränktes Layout von G}

Der Einfachheit halber beschränken wir uns hier auf die Kostenfunktionen f(x)=x**p. Ziel der Optimierung ist dabei eine gleichmäßige balancierte Verteilung aller Module auf dem Chip. Eine Optimierung bzgl. dem Quadrat der Euklidschen Länge ist z.B. an folgender Vorstellung orientiert: die Module üben Kräfte aufeinander aus und sind über die Kanten wie durch Gummibänder miteinander verbunden. Optimal ist nun ein Layout genau dann, wenn die Summe der Kräfte, die auf ein Modul (,das nicht Randknoten ist,) wirkt, gleich 0 ist. In existierenden (automatischen) Entwurfssystemen werden häufig Plazierungsheuristiken benutzt, die auf ähnlichen Ideen basieren (siehe z.B. 'force-directed placement'

Optimierungen nach der p-ten Potenz (p>2) der Euklidschen Länge bewirken eine noch stärker balancierte Verteilung der Module und werden uns im "Grenzfall" ein 'optimal balanciertes' Layout liefern. Neben dem Optimierungsziel 'balancierte Verteilung' soll das Layout aber gleichzeitig die topologische Information (über relative Lage der Drähte zueinander u.s.w.), die aus der logisch-topologischen Ebene des Systems stammt, nicht zerstören. Inwieweit das alles gewährleistet ist, erläutern die nun folgenden Ergebnisse. Eine detaillierte Darstellung der Sachverhalte findet man in [BeHo] und [BeOs].

Daß optimale Layouts existieren, und unter welchen Bedingungen sie eindeutig sind, erhält man aus

Satz 5.1: (Existenz und Eindeutigkeit)
i) Sei G ein Graph mit festem Rand R(c), dann existiert ein optimales Layout L von G bzgl. f(x)=x**p, und L ist R(c)-beschränkt.
ii) Ist G zusammenhängend, dann ist das optimale Layout von G bzgl. f(x)=x**p eindeutig bestimmt.

Die Beweise setzen nur elementare Analysiskenntnisse voraus, auf sie soll hier ganz verzichtet werden.

Beide Sätze sind noch 'unabhängig' von der speziellen Gestalt des Graphen, den wir aus der logisch-topologischen Ebene erhalten haben. Wie oben erwähnt, bestand unser Ziel aber u.a. darin, eine Optimierung zu entwickeln, die die topologischen Eigenschaften nicht zerstört. Dazu definieren wir den Begriff R(c)-planar. Ein Graph G heißt R(c)-planar, falls es ein R(c)-beschränktes Layout L gibt, bei dem die Kanten überkreuzungsfrei in die Ebene eingebettet sind. Graphen, die aus der logisch-topologischen Ebene zur Optimierung übergeben werden, sind in der Regel R(c)-planar. Es stellt sich nun die Frage, ob das optimale Layout eines R(c)-planaren Graphen wieder überkreuzungsfrei, d.h. R(c)-planar, ist. Daß dies tatsächlich unter gewissen Voraussetzungen der Fall ist, zeigt folgender Satz.

Satz 5.2:
Sei G ein 3-fach zusammenhängender R(c)-planarer Graph mit festem Rand,
dann ist das optimale Layout von G bzgl. f(x)=x**p R(c)-planar.

Daß ein optimales Layout eines 2-fach zusammenhängenden Graphen nicht R(c)-planar sein muß, wird aus folgendem Beispiel deutlich:

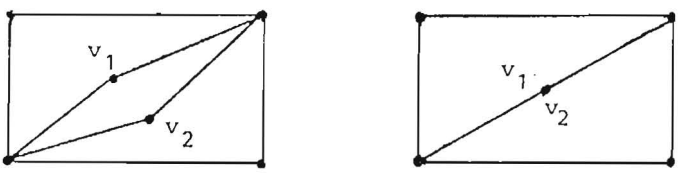


Bild 30: Ausgangslayout optimales Layout

Trotzdem läßt sich Satz 5.2 auf allgemeine R(c)-planare Graphen verallgemeinern. Wir nennen ein Layout quasi R(c)-planar, wenn das Layout, informal gesprochen, 'planar aussieht'. (Siehe obiges Beispiel!) Damit erhalten wir

Satz 5.3:
Sei G ein zusammenhängender Graph mit festem Rand R(c),
dann ist das optimale Layout von G bzgl. f quasi R(c)-planar.

Der exakte Beweis beider Theoreme ist sehr aufwendig und schwierig. Er nimmt den wesentlichen Teil von [BeHo] in Anspruch und erfordert topologische Hilfsmittel. Wir geben hier die intuitive Beweisidee, die trotz der Kompliziertheit des Beweises die Aussage der Sätze einleuchtend macht.
Zunächst wird der Beweis für triangulierte Graphen mit festem Rand geführt. Da G R(c)-planar ist, existiert ein R(c)-planares Layout L von G. Bezüglich L können wir uns also die Innenfläche von R(c) als "glattes Zeltdach" bestehend aus dreieckigen "Facetten" vorstellen. Nehmen wir an, daß ein optimales Layout L' nicht planar ist. Dann führen wir folgendes Verfahren durch: L wird stetig in L' deformiert. Wegen der Nichtplanarität von L' wird bei der Deformation mindestens eine "Facette des Zeltdaches" über

eine andere gezogen. Da das gesamte "Zeltdach" aber aus einer Fläche besteht, muß die deformierte Fläche Spitzen aufweisen, an denen alle Kanten "in eine Richtung laufen" (siehe Bild 31).

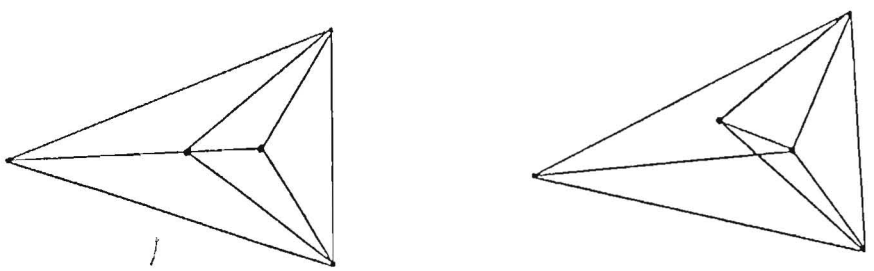


Bild 31: Layout L Layout L'

Solch ein Layout kann aber nicht optimal sein, da ein Verkürzen der betreffenden Kanten die Kosten des gesamten Layouts verringert. Auf Grund dieser Idee läßt sich ein formaler Beweis von Satz 5.2 und 5.3 auch für allgemeine Graphen mit festem Rand führen.

Zusammenfassend läßt sich also festhalten: Falls das topologische Netz, das wir als "Eingabe" für unsere Plazierung erhalten, "hinreichend zusammenhängend" ist, existiert ein optimales Layout (z.B. bzgl. der Quadrate der Kantenlängen), das planar ist und die topologische Anordnung im Netz unverändert übernimmt.

Neben der Existenz und den Eigenschaften optimaler Layouts bzgl. f(x)=x**p ist natürlich ihre Konstruktion von Interesse. Dies geschieht mithilfe von Approximationsverfahren. Dabei ist zu unterscheiden zwischen f(x)=x**2 und f(x)=x**p für p>2.
Im Falle p=2 entspricht die Lösung der Optimierungsaufgabe der Lösung von zwei (dünn besetzten) linearen Gleichungssystemen. Wendet man das Jacobi- oder Gauß-Seidel-Verfahren an, so läßt sich die Konvergenz beider Verfahren gegen das optimale Layout zeigen. Bild 32 zeigt das optimale Layout des Multiplizierers aus Abschnitt 2, das mit Hilfe des Gauß-Seidel-Verfahrens gewonnen wurde. Die Komplexität des Verfahrens hängt ab vom Spektralradius der betrachteten Matrix. Es gibt eine Klasse von Graphen (siehe [StBu] S.257ff), bei der man bei vorgegebener Genauigkeit das optimale Layout in quadratischer Zeit (in der Anzahl der Kanten des Graphen) approximieren kann. Experimentelle Ergebnisse lassen bei den betrachteten Graphen auf lineare Zeit schließen. Aller-

dings ist die genaue Berechnung des Spektralradius' für diese Fälle noch nicht gelungen.
Im Falle p>2 wurden Methoden des stärksten Abstiegs untersucht. Genauere Ausführungen für beide Fälle (p=2 und p>2) und zugehörige Programmlistings sind in [Gr] nachzulesen.
Die Lösung der Optimierungsaufgabe für p=2 in linearer Zeit ist möglich, wenn man Multigrid-Methoden benutzt, wie sie in [St] eingeführt werden. Eine Programmierung des Verfahrens (für das hier vorliegende Problem) steht zur Zeit noch aus. Inwieweit sich die Multigrid-Methoden auf den Fall p>2 anwenden lassen, ist zur Zeit noch offen und Gegenstand weiterer Untersuchungen.

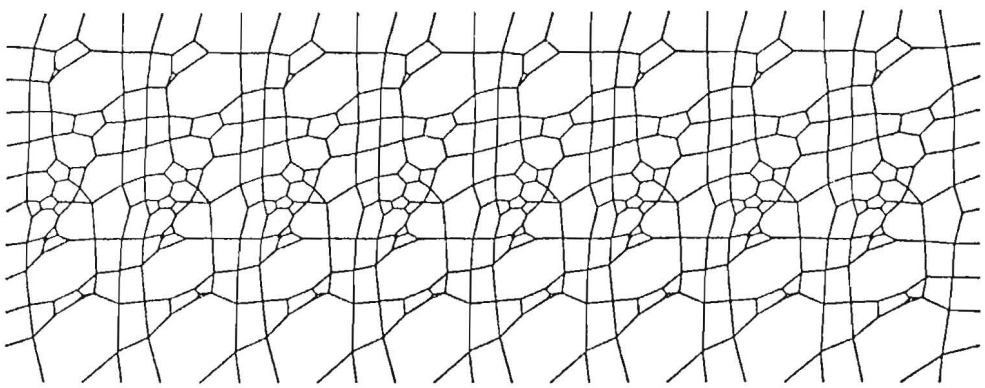


Bild 32: Optimales Layout bzgl. f(x)=x**2
für den Multiplizierer aus Abschnitt 2

Im letzten Teil dieses Abschnitts gehen wir etwas näher auf folgende Beobachtung ein: Die längste Kante eines Layouts fällt bei der Kostenberechnung bzgl. f(x)=x**p um so stärker ins Gewicht, je höher die Zahl p ist, mit der die Kantenlängen potenziert werden.
Es besteht also Grund zu der Annahme, daß das optimale Layout eines Graphen bzgl. f(x)=x**p bei wachsendem p dazu neigt, die längste Kante zu minimieren. Eine solche Minimierung ist von Interesse, da die Signallaufzeit über eine Leitung proportional zum Quadrat der Länge der Leitung ist, d.h. eine Optimierung der maximalen Leitungslängen (= Optimierung nach den maximalen quadrierten Leitungslängen) ist sinnvoll.

Die "lokale Version" dieses Problems ("Suche den Punkt s in der

Ebene, der den maximalen Abstand zu einer endlichen vorgegebenen Punktemenge S in der Ebene minimiert!") ist eine klassische Aufgabe der algorithmischen Geometrie. Effiziente Lösungsmethoden sind bekannt. ([ShHo],[Os])

Ein natürlicher Ansatz zur Lösung der oben gestellten Frage ist der folgende: Suche ein Layout, so daß jeder Nicht-Randknoten die maximale Entfernung zu seinen Nachbarknoten minimiert, d.h. obige "lokale Version" des Problems ist für jeden Nicht-Randknoten gelöst. Ist dies für ein Layout L der Fall, so heißt L balanciert. (Entsprechend heißt ein Knoten v balanciert, falls er in einem Layout die maximale Entfernung zu seinen Nachbarknoten minimiert.)
Die Existenz eines balancierten Layouts für den allgemeinen Fall ist aber nicht direkt einleuchtend. Es liegt nahe, ein Einzelschrittverfahren zu versuchen, das einen effizienten Algorithmus für das lokale Problem benutzt; unklar ist aber, ob dieses Verfahren konvergiert. Außerdem stellt sich (leider) heraus, daß es u.U. kein eindeutiges balanciertes Layout gibt und daß ein balanciertes Layout nicht einmal die gewünschte Eigenschaft (Maximum über Kantenlängen ist minimal) haben muß (genaueres siehe [BeOs]).

Trotzdem läßt sich das gewünschte Layout konstruieren. Ausgangspunkt sind die optimalen Layouts bzgl. f(x)=x**p. Im folgenden beantworten wir also die Frage: Existiert ein 'Limes' dieser optimalen Layouts und welche Eigenschaften hat er? Wir benötigen folgende Definitionen:

Sei L ein Layout eines Graphen G mit festem Rand.
Dann zerfällt die Kantenmenge von G in k Gerüste Sk(i), i=1,...,k; ein Gerüst Sk(i) enthält genau die i-tlängsten Kanten von L(G).
L wird klassifiziert durch eine Bewertungszahl
N(L):=(l(1),n(1),l(2),n(2),...,l(k),n(k)); dabei ist l(i) die Länge einer Kante in Sk(i) und n(i) die Anzahl der Kanten in Sk(i).
Damit läßt sich eine Ordnung '<' auf der Menge der Layouts von G definieren: L < L' genau dann, wenn N(L) < N(L') bzgl. der lexikographischen Ordnung in den Komponenten von N.

Es ist leicht zu zeigen, daß ein minimales Element L dieser Ordnung, falls es existiert, folgende Eigenschaften besitzt: es ist balanciert und für die Gerüste Sk(i) i=1,...,k gilt: Sk(i) ist möglichst klein und mit möglichst geringer Länge der zugehörigen Kanten realisiert.
Ein minimales Element L von '<' erfüllt also beide vorher angegebenen Optimierungskriterien ('Minimierung der maximalen Kantenlänge' und 'Balancierung der Knoten'). Wir nennen L deshalb optimal balanciert.

Existenz und Eindeutigkeit des optimal balancierten Layouts werden durch die folgenden Sätze gewährleistet.

Satz 5.4: (Existenz und Eindeutigkeit)
i) Sei G ein Graph mit festem Rand. Falls ein optimal balanciertes Layout von G existiert, ist es eindeutig bestimmt.
ii) Sei G ein Graph mit festem Rand. Die Folge (L(p)) der optimalen Layouts von G bzgl. f(x)=x**p für p=2,3,... konvergiert gegen das optimal balancierte Layout.

Der Beweis ist im Detail in [BeOs] nachzulesen, wir geben hier nur einen Überblick:
Die Eindeutigkeitsaussage wird aus der Minimalität bzgl. '<' gewonnen, indem wir folgendes tun: wir nehmen an, es existieren zwei minimale Elemente, dann konstruieren wir aus beiden ein Layout, dessen Bewertungszahl noch kleiner ist und erhalten damit einen Widerspruch.
Zum Beweis der Existenzaussage schließen wir zunächst auf die Existenz einer konvergenten Teilfolge der (L(p)). Von einer konvergenten Teilfolge zeigen wir ebenfalls mit Widerspruchsbeweis, daß sie gegen das minimale Element von '<' konvergiert. Mit Hilfe von Satz 5.4.i) schließt man jetzt sofort 5.4.ii).

Da der Limes von quasi R(c)-planaren Layouts wieder quasi R(c)-planar ist, erhält man mit Satz 5.3:

Satz 5.5:
Das optimal balancierte Layout eines R(c)-planaren Graphen mit festem Rand ist quasi R(c)-planar.

Nach Satz 5.4 lassen sich optimal balancierte Layouts approximieren. Ein effektives Verfahren erfordert eine schnelle Berechnung der Gerüste. Wie dies möglich ist, wird zur Zeit untersucht. Bild 33 zeigt das optimal balancierte Layout des Multiplizierers aus Abschnitt 2

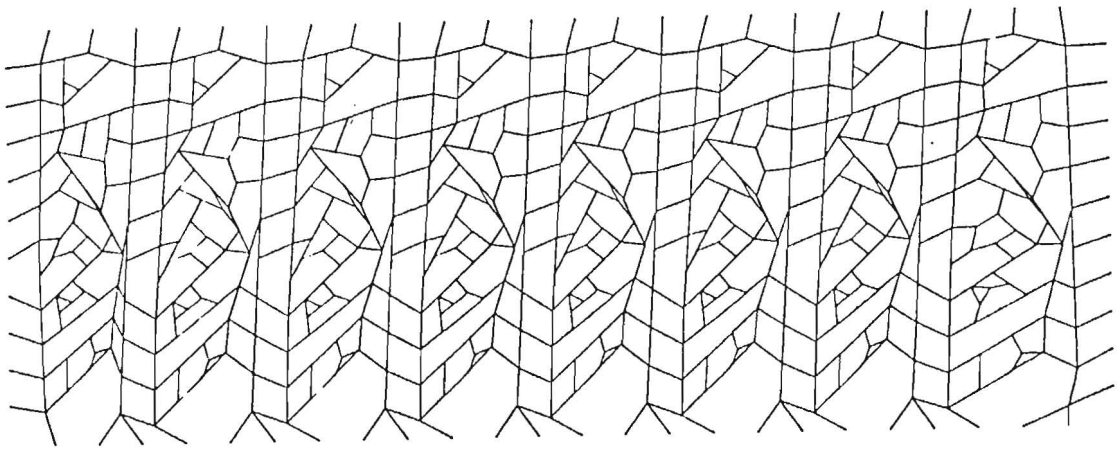


Bild 33:
Optimal balanciertes Layout des Multiplizieres aus Abschnitt 2

Vergleicht man die Layouts aus den Bildern 32 und 33, dann läßt sich feststellen, daß schon eine Optimierung nach dem Quadrat der Kantenlänge zu einer gleichmäßigen Verteilung der Knoten auf dem zur Verfügung stehenden Gebiet führt. Die Länge der einzelnen Kanten ist allerdings noch recht unterschiedlich. Beim optimalen Layout bzgl. höheren Werten von p zeigt sich, daß die Kantenlängen einander möglichst angeglichen werden, ohne daß die gleichmäßige Verteilung der Knoten leidet. Diese Tendenz verstärkt sich beim optimal balancierten Layout.

Wir schließen mit einigen kritischen Bemerkungen:
- bei den bisher behandelten Optimierungen können die Leitungen in beliebiger Richtung verlaufen. Die heutigen Technologien erlauben dies in der Regel nicht. Deshalb wurde im Rahmen einer Diplomarbeit (siehe [Schm]) ein Verfahren angegeben, das ein gegebenes optimales Layout unter Beibehaltung der Lage der Knoten in ein rechtwinkliges umwandelt.

- Bisher wurden Layouts von Graphen mit festem Rand betrachtet. Diese Layouts spiegeln ein physikalisches Schaltnetz nur in sehr idealisiertem Maße wider. Auf dem Weg zu einem 'physikalischen' Layout ist dies nur ein erster Schritt. Eine mögliche Weiterentwicklung von Cadic in diese Richtung soll abschließend kurz skizziert werden:

Wir erhalten ein realistischeres Bild des Schaltnetzes, indem wir die Knoten und Kanten des Graphen im Layout mit einem "Schlauch" umgeben, der der Breite der Leiterbahnen bzw. der Größe der Module entspricht. Das Layout des "Schlauchgraphen" soll unter Beibehaltung der Leitungsbreiten und Modulgrößen so umgeformt werden, daß es überkreuzungsfrei ist (und somit die Entwurfsbedingungen erfüllt) und gleichzeitig möglichst wenig Fläche einnimmt. Dies erfolgt mithilfe eines 'logarithmischen Pumpens'. Eine genaue Ausarbeitung und Programmierung des Verfahrens wird in [Schw] vorgenommen.

In einem nächsten Schritt soll das erhaltene Layout des Schlauchgraphen weiter kompaktifiziert werden. Es ist daran gedacht, Methoden zu verwenden, die dem Kristallisationsprozeß der Physik nachgebildet sind (siehe Abschnitt 4). Dabei soll die Wirkung von Federkräften, die nun nicht mehr Knoten, sondern räumlich ausgedehnte Module bewegen, mit Zufallsprozessen überlagert werden. Ähnliche Experimente werden für globale Verdrahtung in [VeKi] diskutiert. Hier befinden sich die Überlegungen aber noch in einem frühen Stadium.

6. Eine Programmiersprache zur Unterstützung des automatischen Entwurfs von Schaltkreisen auf der logisch topologischen Ebene

Aus den Ausführungen der Kapitel 2 und 3 ergibt sich, daß unser CAD-System rekursive Ausdrücke verstehen und manipulieren sollte. Ein von uns geschriebener Formelübersetzer für solche Ausdrücke, der die Rekursionen der hier vorgestellten Beispiele aufgelöst und ausgezeichnet hat, dient uns dazu, Erfahrung mit Netzmanipulationen topologischer und topographischer Art zu sammeln.

Eine Programmiersprache für die logisch-topologische Ebene eines VLSI-Entwurfssystem müßte im Kern die grundlegenden Objekte des vorgestellten Kalküls als Datentypen enthalten, insbesondere also den Datentyp string über beliebigen Alphabeten, sowie den Datentyp net über beliebigen Grundbausteinen. In naheliegender Weise machen diese Objekte auch einen erweiterten Typ set notwendig. Als Operationen sollten die Stringoperationen aus COMSKEE ([Me84]) vorhanden sein, insbesonders also Konkatenation von Wörtern, Teilwortsuche und Teilwortersetzung. Weiter sollten die Operationen $\oplus$, $\ominus$, Rotation und Spiegelung von Netzen zur Verfügung stehen.

Homomorphismen und Funktoren nehmen wir als weitere Objekte auf. Natürlich könnte man diese leicht durch Prozeduren ausprogrammieren. Da sie aber eine zentrale Rolle spielen, sollen sie spezifizierbar sein. Die Funktoren verwenden wir, um Netze top-down zu beschreiben, bzw. die rekursiven Gleichungen aufzulösen.

Aus dem vorgestellten Kalkül, insbesonders aus der Forderung nach der rekursiven Definition von Netzen ergibt sich die Notwendigkeit, die Datenstrukturen dynamisch anzulegen. Um dieser Forderung gerecht zu werden, wollen wir COMSKEE erweitern, das ausgezeichnet ist durch seine effizienten und dynamischen Datenstrukturen. Zu nennen sind hier die Typen string, set und array[string] (Wörterbuch). Für COMSKEE liegt eine sehr effiziente Implementierung für das Betriebssystem BS2000 von SIEMENS vor. Implementierungen auf einer VAX unter UNIX und auf einem SIRIUS-Mikrocomputer sind weit fortgeschritten. Aus diesen Gründen planen wir unsere Programmiersprache CAD-IC als Erweiterung von COMSKEE.

Eine vorläufige Definition der Programmiersprache, soweit sie durch die Ausführungen im ersten Teil dieser Arbeit begründet wird, findet man in [BHKM].

Schlußbemerkung: Die Arbeiten, über die wir hier berichten, werden von der Deutschen Forschungsgemeinschaft im Projekt B des Sonderforschungsbereiches 124 "VLSI-Entwurfsmethoden und Parallelität" seit dem 1.1.1983 gefördert. Im gleichen Projekt werden auch die Arbeiten an HILL gefördert. Beide Teilprojekte überlappen sich in ihren Verfahren etwas und ergänzen sich in sofern, als der Schwerpunkt von CADIC, d.h. unserer Arbeiten in den höheren Entwurfsebenen liegt, und HILL vorzüglich untere Schichten des Entwurfes bearbeitet. Es ist in beiden Systemen eine Schnittstelle geplant, die es erlaubt, Entwürfe von CADIC an HILL weiterzureichen. Weiter planen wir eine Schnittstelle zu VENUS von Siemens, um in CADIC entworfene Layouts auf hoher Ebene in Chips umsetzen zu können.

Unter SFB 124 B wird weiter das Projekt von Herrn Kollegen Zimmermann in Kaiserslautern seit 1984 gefördert, das besonders Plazierungsfragen behandelt. Diese Arbeiten ergänzen unsere Arbeiten insofern, als sie eine Orientierung im "Großen" für die Plazierung von Modulen liefern, für die unsere Optimierungsverfahren nicht greifen.

An den Arbeiten waren auch beteiligt U. Becker, U. Fissgus, H.G. Osthof. Große programmiertechnische Unterstützung erhielten wir aus dem SFB 100, Projekt E, in dem die Programmiersprache COMSKEE entwickelt wurde.

Zum Schluß möchten wir Herrn T. Lengauer, Universität Paderborn
K. Mehlhorn, Universität Saarbrücken
B. Schallenberger, Siemens AG, München
G. Zimmermann, Universität Kaiserslautern
für kritische Diskussionen danken, die für unsere Arbeiten nützlich waren.

References:

[A] E. Artin: "Theory der Zöpfe", Abhandlung des Mathem. Seminars der Universität Hamburg, Vol.4, pp. 47-72, 1925

[Be85] B. Becker: "An easily testable optimal-time VLSI-Multiplier", Jan. 1985, Universität des Saarlandes

[BeHo] B. Becker, G. Hotz: "On the Optimal Layout of Planar Graphs with Fixed Boundary", Techn. Bericht Nr. 03/1983 des SFB 124, Universität des Saarlandes

[BMKM] B. Becker, G. Hotz, R. Kolla, P. Molitor: "Ein CAD-System zum Entwurf integrierter Schaltungen", Techn. Bericht Nr. 16/1984 des SFB 124, Universität des Saarlandes

[BeOs] B. Becker, H.G. Osthof: "Layouts with Wires of Balanced Length", Techn. Bericht Nr. 07/1984 des SFB 124, Universität des Saarlandes

[Br] M.A. Breuer: "Design Automation of Digital Systems", Vol. 1, Theory and Techniques, Prentice Hall 1972

[Bry] R. Bryant: "Third Caltech Conference on VLSI", Springer Verlag Berlin 1983

[BuHoe] L. Budach, H.J. Hoehnke: "Automaten und Funktoren", Akademie-Verlag, Berlin, Bd. 35

[Do] W.E. Donath: "Complexity Theory and Design Automation", Proc. 17th Design Autom. Conf. 1980, pp 412-419

[Eh] C. Ehresmann: "Categories et Structures DUNOD, Paris 65"

[F..] C.J. Fisk, D.L. Caskey, L.L. West: "ACCL: Automated Circuit Card Etching Layout", Proc. IEEE, Vol. 55, No. 11 (1967), pp. 1971-1982

[Ga73] H. Gabow: "Implementation of Algorithms for Maximum Matching on Non-Bipartite Graphs", Ph.Dissertation, Stanford University 1973

[Ga] K.F. Gauß: "Zur mathematischen Theorie der elektrodynamischen Wirkungen", (1933), Königliche Gesellschaft der Wissenschaften zu Göttingen, (1877), Vol. 5, p 605

[Gr] U. Groh: "Optimale Einbettung von Graphen mit festem Rand", Diplomarbeit, Saarbrücken 1983,

[Ha75] F. Hadlock: "Finding a Maximum Cut of a Planar Graph in Polynomial Time", SIAM Journal on Computing, Vol.4, No. 3 (1975) pp. 221-225

[HaSt71] A. Hashimoto, J. Stevens: "Wiring Routing by Optimizing Channel Assignment within Large Apertures", Proceedings of the Eight Design Automation Workshop, IEEE, 1971, pp. 155-169

[Ho1] G. Hotz: "Eine Algebraisierung des Syntheseproblemes für Schaltkreise", EIK 1965, pp. 185-205, 209-231,

[Ho2] G. Hotz: "Schaltkreistheorie", de Gryter, Berlin, New York 1974, pp. 243-330

[JeGe83] D.W. Jepsen, C.D. Gelatt: "Macro Placement by Monte Carlo Annealing"

[KiGeVe] S. Kirkpatrick, C.D. Gelatt, M.P. Vecchi: "Optimization by Simulated Annealing", SCIENCE, Vol. 220, No. 4598, May 1983, pp. 671-680

[La76] E.G. Lawler: "Combinatorial Optimization Theory", Holt, Rinehart and Winston 1976, Chapter 6, pp. 217-239

[LeiPi] C.E. Leiserson, R.Y. Pinter: "Optimal Placement for River Routing", SIAM, J. Comput. Vol. 12, No. 3, Aug. 83, S. 447-462

[LuVu] W.K. Luk, J. Vuillemin: "Recursive Implementation of Optimal Time VLSI Integer Multiplier", IFIP 1983, pp 155-168

[Me84] J. Messerschmidt: "Linguistische Datenverarbeitung mit Comskee", B.G. Teubner Stuttgart 1984,

[MRRTT] N. Metropolis, W. Rosenbluth, N. Rosenbluth, A. Teller, E. Teller: "Equation of State Calculations by Fast Computing Machines", Journal of Chemical Physics, Vol.21, No. 6 (1953), pp. 1087-1092

[Os] H.G. Osthof: "Der minimale Kreis um eine endliche Punktmenge", Diplomarbeit, Saarbrücken 1983

[Pi82] R.Y. Pinter: "Optimal Layer Assignment for Interconnect", ICCC 1982, pp. 398-401

[Pi83] R.Y. Pinter: "River Routing: Methodology and Analysis Proceedings of the 3rd Caltech Conference on VLSI", März 83, S. 141-163

[SaBh] S. Sahni, A. Blatt: "The Complexity of Design Automation", Proc. 17th, Design Autom. Conf. 1980, pp. 402-411

[SaGo] S. Sahni, T. Gonzalez: "P-complete Approximation Problems", JACM 23, 1976, pp. 555-565

[Schm] F.J. Schmitt: "Rechtwinklige Verdrahtung optimaler Layouts", Diplomarbeit, Saarbrücken 1985

[Schw] K. Schworm: "Layouts unter Berücksichtigung von Leitungsbreiten und Modulgrößen", Diplomarbeit, Saarbrücken, erscheint 1985

[ShHo] M.I. Shamos, D. Hoey: "Closest-Point Problems", Proc. 16th IEEE Symp. on Foundations of Comput. Sci., Oct. 1975, pp. 151-162

[Sk] J. Sklansky: "Conditional-sum Addition Logik", IRE-EC 9 (1960), pp 226-231

[Sp] O. Spaniol: "Arithmetik in Rechenanlagen", Teubner Studienreihe Informatik, Stuttgart 1976, pp. 83-84

[St] K. Stüben: "Algebraic Multigrid (AMG): Eperiences and Comparisons", Arbeitspapiere der GMD Bonn, Nr. 23, März 1983

[StBu] J. Stoer, R. Bulirsch: "Einführung in die Numerische Mathematik II", Springer Verlag Berlin, 1978

[VeKi] M.P. Vecchi, S. Kirkpatrick: "Global Wiring by Simulated Annealing", IEEE Transactions on CAD, Vol. CAD-2, Nr. 4, October 1983

[Wa] C.S. Wallace: "A suggestion for a fast multiplier IEEE 13, 1964, pp. 14-17

[Wei] W. Weidner: "Der topologische und algebraische Abschluß freier x-Kategorien", Dissertation, Saarbrücken 1977, pp. 1-77

Universität des Saarlandes
Fachbereich Informatik
Im Stadtwald
6600 Saarbrücken