

Ein logisch-topologischer Kalkül zur Konstruktion integrierter Schaltkreise* **

Teil II

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

Universität des Saarlandes, Fachbereich für Informatik, Im Stadtwald, D-6600 Saarbrücken

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 Kommunikationsnetze 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, der Kontakte, minimal ist ([16]). Die Einschränkung auf nur 2 Schichten ist technologisch gesehen gerechtfertigt: Um verschiedene Verzögerungszeiten verschiedener Layers zu vermeiden, brettet man die Kommunikationsnetze in zwei Metallschichten ein.

In [27] wird das Problem der 2-Schichtenzuweisung auf das MAXCUT-Problem für planare Graphen reduziert, das in $O(n^3)$ lösbar ist ([15], [12], [21]). (n steht für die Anzahl der Überkreuzungen, Verzweigungen und Anschlüsse.) Würde man dieses Verfahren auf ein Kommunikationsnetz der Größe 100 000 ansetzen, so benötigte man im worst-case einige 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, den man begeht, und einem Steuerungsfaktor, der „Temperatur“.

Die Analogie zum Kristallisierungsprozeß findet man in [20]. In [20], [19] wurde diese Idee angewandt auf das Problem des Handelsreisenden, sowie auf Verdrahtungs- und Plazierungsprobleme bei Standardzellen.

Wir werden in dieser Arbeit 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. [27])

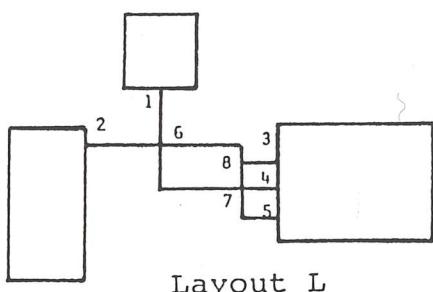
Sei L ein planarer Schaltkreis in der euklidischen Ebene. Ohne Beschränkung der Allgemeinheit sei L rechtwinklig verdrahtet.

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.

Eine Abbildung $f: V(L) \rightarrow \{\text{rot, blau}\}$ 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

* Diese Arbeiten werden von der Deutschen Forschungsgemeinschaft im Rahmen des SFB 124 „VLSI-Entwurfsmethoden und Parallelität“ gefördert.

** Heinz Zemanek zum 65. Geburtstag



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

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

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: -)

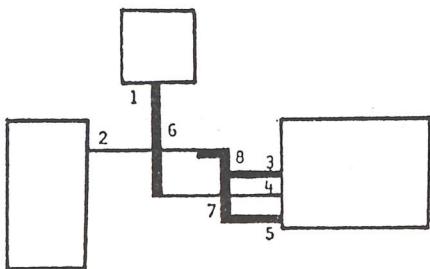


Abb. 24

Pfadsegment von i die Farbe f(i), das horizontale Pfadsegment die komplementäre Farbe $\bar{f}(i)$.

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 $k := \{e_1, e_2\} \in E(L)$ mit einem Paar $(g(k), u(k)) \in \{(0,1), (1,0)\}$: $g(k)$ soll genau dann 1 sein, wenn man einen Kontakt auf der Kante k benötigt, falls die Randecken e_1, e_2 von k gleich gefärbt sind. Im obigen Beispiel sieht die Gewichtung also wie folgt aus:

{1,6}	{2,6}	{3,8}	{4,7}	{5,7}	{6,7}	{6,8}	{7,8}
(0,1)	(1,0)	(0,1)	(1,0)	(0,1)	(1,0)	(1,0)	(0,1)

$u(k)$ ist also genau dann 1, wenn man einen Kontakt auf der Kante k benötigt, falls die Randknoten e_1, e_2 von k verschieden gefärbt sind.

Definieren wir für jede Färbung f von L und jede Kante $k := \{e_1, e_2\} \in E(L)$ das Prädikat $s(f, k) := (f(e_1) + f(e_2))$, 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 [25] vorgestellt:

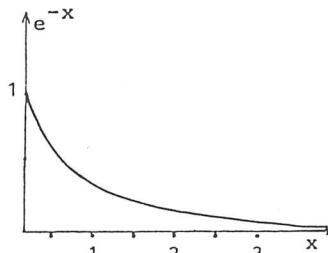


Abb. 25

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 $dE \leq 0$, so wird die Bewegung gemacht, ansonsten wird die Bewegung mit einer Wahrscheinlichkeit von $\exp(-dE/kT)$ (k Boltzmann-Konstante) ausgeführt. 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

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

(3) Temperatur = T_{max} ;
(4) **for** i from 1 to d (Temperatur)
/* d (Temperatur): Dauer, während der diese „Temperatur“ beibehalten wird */
loop Wähle zufällig eine Ecke $e' \in V(L)$;
Berechne $z_{loc}(e', f')$, wobei $f'(e) = \begin{cases} f(e) & \text{für } e \neq e' \\ \text{sonst;} & \end{cases}$
 $dz = z_{loc}(e', f') - z_{loc}(e', f)$;
if ($dz \leq 0$) or
(($dz \geq 1$) and ($random(0,1) \leq p(dz, Temperatur)$))
/* $random(0,1)$: zufällige Zahl zwischen 0 und 1.
 $p(dz, Temperatur)$: Wahrscheinlichkeit, mit der umgefärbt wird, falls $dz \geq 1$ */
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**;

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

Die Werte für $p(x, y)$, T_{max} , dT und $d(T_{max} - i * dT)$ ($i = 0, \dots, T/dT$) müssen experimentell ermittelt werden.

Statistische Betrachtungen

Für die Unbekannten des Verfahrens haben wir die Werte $T_{max} = 100$, $dT = 25$, $d(i) = 10$ $|V(L)|$ ($\forall i$) und $p(dz, t) = t/(100 * dz)$ gewählt. Das Verfahren wurde unter anderen auf folgende 4 Schaltkreise angesetzt:

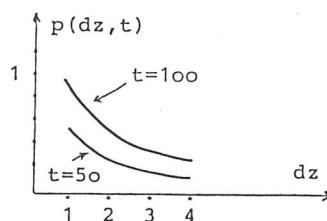


Abb. 26

- (1) 4-Bit-Multiplizierer ([23])
- (2) Testfreundlicher 4-Bit-Multiplizierer ([2])
- (3) 4-Bit-Conditional-Sum-Addierer ([4])
- (4) 8-Bit-Conditional-Sum-Addierer

Optimum	Anzahl der Kontakte bei obigem Verfahren (Durchschnittswert)	# $V(L)$	Rechenzeit (Siemens 7561)
(1)	15	30 (100%)	712
(2)	74	96 (30%)	734
(3)	18	20,3 (13%)	337
(4)	49	56 (14%)	861

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

Verfahren A: Wähle im vorgestellten Verfahren $T_{max} = 0$, $dT = 0$, $p(x, y) = 0$ und $d(i) = 50 |V(L)|$, d.h. 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 (147%)	87,3 (482%)
(2)	102 (38%)	178,1 (141%)
(3)	20,5 (14%)	29,5 (64%)
(4)	56,4 (15%)	79,8 (63%)

Es scheint also, daß das vorgestellte Verfahren ein schneller und (relativ) guter Algorithmus zur Einbettung der Verdrahtungsnetze in 2 Schichten ist. Der Vergleich des Verfahrens z.B. mit dem Verfahren B, das gerne als Heuristik verwendet wird, zeigt, daß unsere Vorgehensweise im Mittel um einen Faktor 1,5 bis 2 besser ist.

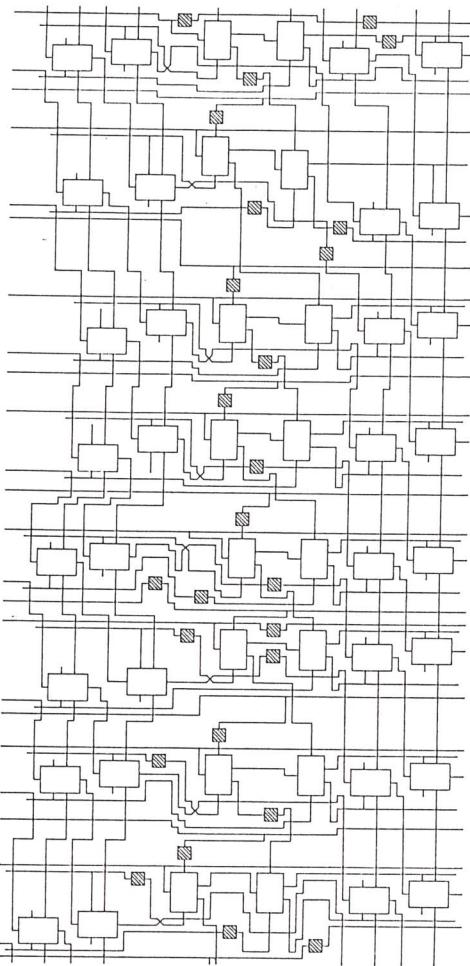


Abb.27. In 2 Schichten eingebetteter 4-Bit-Multiplizierer mit 26 Kontakten. Die schraffierten Kästchen repräsentieren die Kontakte.

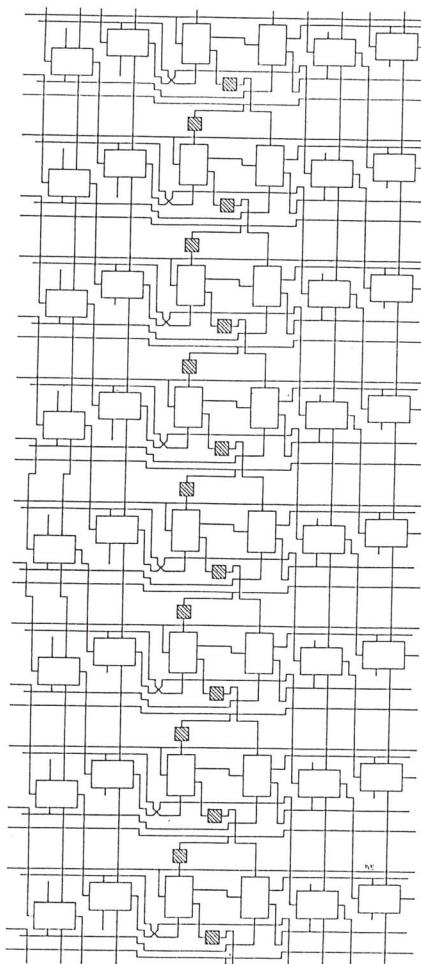


Abb.28. Optimal eingebetteter 4-Bit-Multiplizierer

Klar ist, daß man obiges Verfahren auf beliebig viele Schichten verallgemeinern kann.

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 Abb. 28 a 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.

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 Abb. 28 a, dann wird sie bei geeigneter Einstellung der freien Weglänge in dem absoluten Minimum landen und



Abb.28 a

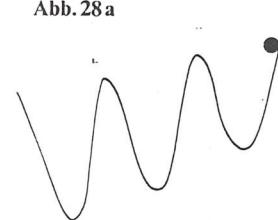


Abb.28 b

daraus auch nicht wieder entkommen. Sind benachbarte relative Minima aber etwa gleich tief, wie dies in Abb. 28 b angedeutet ist, dann ist die freie Weglänge schwer so einstellbar, daß dieser Effekt eintritt.

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 [6]). 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 [29], [9]). Will man trotzdem automatische Entwurfssysteme, die effiziente Algorithmen benutzen, entwickeln, 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 („rechteckige Verdrahtung“) nimmt, wurde bereits in Abschnitt 3 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

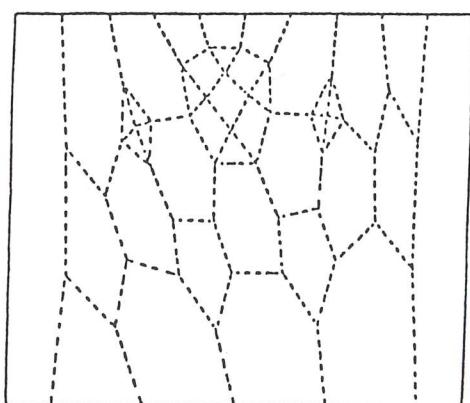


Abb. 29. $R(c)$ -beschränktes Layout von G

auf dem Rand der Chipfläche platziert 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 $e = (v_1, v_2)$ von $G(N)$ auf die Strecke $L(v_1)L(v_2) := L(e)$ 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 (s. Abb. 29).

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. [7]). (Daß dieser Ansatz Vorteile bieten kann, wird aus folgendem Beispiel klar: Nimmt man das ‚Quadratic Assignment Problem‘, das NP-hart für Gitter ist ([30]) 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, evtl. zu optimieren sind. Sie alle in einem Algorithmus zu berücksichtigen, scheint nicht möglich. Tatsächlich wird in der Regel eine Metrik benutzt, die durch die Verbindungen zwischen den Modulen definiert ist und dann in einem entsprechenden Verfahren optimiert wird. Eine Metrik, die erfahrungsgemäß viele der obigen Ziele gut berücksichtigt und deshalb häufig benutzt wird, ist die gewichtete Gesamtlänge der Verbindungen (siehe [6], [29]). Aus diesem Grunde definieren wir:

\mathbb{R}^+ bezeichne die positiven reellen Zahlen (einschl. 0) und $f: \mathbb{R}^+ \rightarrow \mathbb{R}^+$ sei eine stetige monoton wachsende Funktion.

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

L bzgl. f . ($|L(e)|$:= Euklidische Länge von $L(e)$), L' heißt optimales Layout von G bzgl. f , falls $\text{cost}(L', f) = \inf \{\text{cost}(L, f) \mid L \text{ ist } R(c)\text{-beschränktes Layout von } G\}$.

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 Euklidischen 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“ in [11]).

Optimierungen nach der p -ten Potenz ($p \geq 2$) der Euklidischen 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 usw.), 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 [3] und [5].

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 überkrezungsfrei, 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:

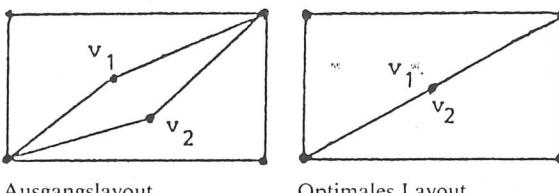


Abb. 30

Trotzdem läßt sich Satz 5.2 auf allgemeine $R(c)$ -planare Graphen verallgemeinern. Wir nennen ein Layout *quasi $R(c)$ -planar*, wenn das Layout, informell 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

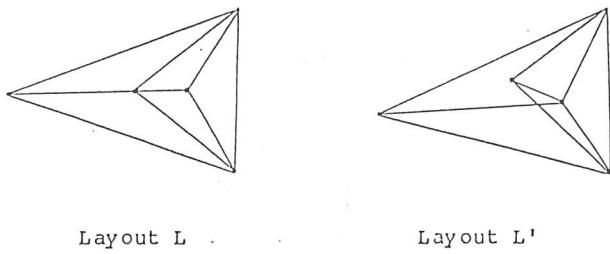


Abb. 31

Teil von [3] 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 $G \in R(c)$ -planar ist, existiert ein $R(c)$ -planares Layout L von G . Bezuglich 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“ (s. Abb. 31).

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 mit Hilfe 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. Abbildung 32 zeigt das optimale Layout des Multiplizierers aus Abschnitt 2, das mit

G. Hotz et al.: Kalkül zur Konstruktion integrierter Schaltkreise. II

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 [37] S. 257 ff.), 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. Allerdings 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 Programm listings sind in [14] nachzulesen.

Die Lösung der Optimierungsaufgabe für $p = 2$ in linearer Zeit ist möglich, wenn man Multigrid-Methoden benutzt, wie sie in [36] 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.

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 ([33], [26]).

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.

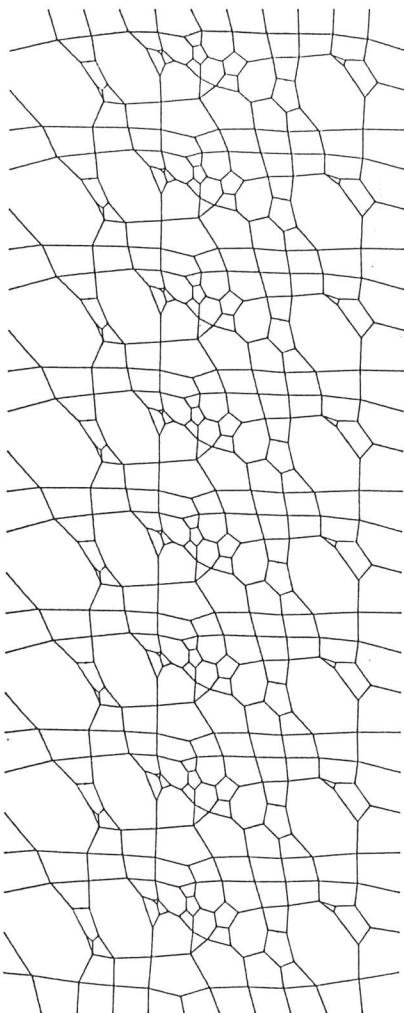


Abb.32. Optimales Layout bzgl. $f(x) = x^2$ für den Multiplizierer aus Abschnitt 2

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 [5]).

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 $S_k(i)$, $i = 1, \dots, k$; ein Gerüst $S_k(i)$ enthält genau die i -längsten Kanten von $L(G)$.

L wird klassifiziert durch eine *Bewertungszahl*
 $N(L) := (l(1), n(1), l(2), n(2), \dots, l(k), n(k))$; dabei ist
 $l(i)$ die Länge einer Kante in $S_k(i)$ und $n(i)$ die Anzahl der Kanten in $S_k(i)$.

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 $S_k(i)$ $i = 1, \dots, k$ gilt: $S_k(i)$ ist möglichst klein und mit möglichst geringer Länge der zugehörigen Kanten realisiert.

Ein minimales Element L von $<$ erfüllt also bei den 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 $p = 2, 3, \dots$ konvergiert gegen das optimal balancierte Layout.

Der Beweis ist im Detail in [5] 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. Abbil-

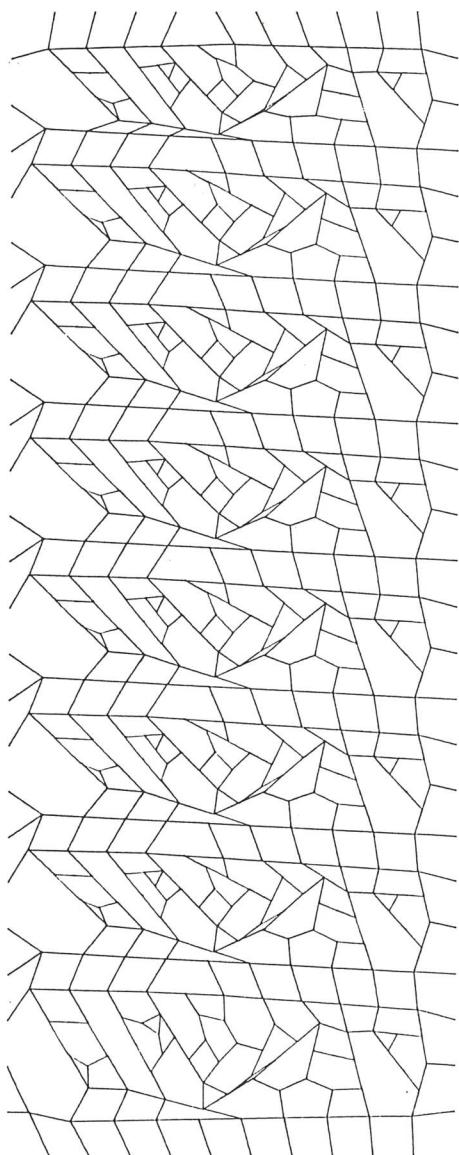


Abb.33. Optimal balanciertes Layout des Multiplizierers aus Abschnitt 2

dung 33 zeigt das optimal balancierte Layout des Multiplizierers aus Abschnitt 2.

Vergleicht man die Layouts aus den Abbildungen 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 Wert 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 [31]) 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 mit Hilfe eines ‚logarithmischen Pumpens‘. Eine genaue Ausarbeitung und Programmierung des Verfahrens wird in [32] vorgenommen.

In einem nächsten Schritt soll das erwähnte 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 [38] 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, Erfahrung mit Netzmanipulationen topologischer und topographischer Art zu sammeln.

Eine Programmiersprache für die logisch-topologische Ebene eines VLSI-Entwurfssystems müßte im

Kern die grundlegenden Objekte des vorgestellten Kalküls als Datentypen enthalten, insbesonders 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 ([24]) vorhanden sein, insbesonders also Konkatenation von Wörtern, Teilwortsuche und Teiltwortersetzung. Weiter sollen die Operationen \odot , \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 auf COMSKEE aufsetzen, das ausgezeichnet ist durch ihre 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 [4].

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 insofern, 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 Kollege 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 den Herren T. Lengauer, Universität Paderborn, K. Mehlhorn, Universität Saarbrücken, B. Schallenger, Siemens AG, München, G. Zimmermann, Universität Kaiserslautern für kritische Diskussionen danken, die für unsere Arbeiten nützlich waren.

Literatur

1. Artin, E.: Theorie der Zöpfe. Abh. Math. Semin. Univ. Hamb. 4, 47–72 (1925)
2. Becker, B.: An easily testable optimal-time VLSI-multiplier. EuroMicro 85, pp. 401–409. Amsterdam: North-Holland
3. Becker, B., Hotz, G.: On the optimal layout of planar graphs with fixed boundary. Techn. Bericht Nr. 03/1983 des SFB 124, Universität des Saarlandes
4. Becker, B., Hotz, G., Kolla, R., Molitor, P.: Ein CAD-System zum Entwurf integrierter Schaltungen. Techn. Bericht Nr. 16/1984 des SFB 124, Universität des Saarlandes
5. Becker, B., Osthof, H.G.: Layouts with wires of balanced length. STACS 85, Lecture Notes in Comput. Sci. 182, pp. 21–31. Berlin, Heidelberg, New York: Springer 1985
6. Breuer, M.A.: Design automation of digital systems. Vol. 1: Theory and techniques. Englewood Cliffs, NJ: Prentice-Hall 1972
7. Bryant, R. (ed.): Third Caltech Conference on Very Large Scale Integration. Berlin, Heidelberg: Springer 1983
8. Budach, L., Hoehnke, H.J.: Automaten und Funktoren, Bd. 35. Berlin: Akademie-Verlag 1975
9. Donath, W.E.: Complexity theory and design automation. Proc. 17th Design Autom. Conf. 1980, pp. 412–419. New York: IEEE Computer Soc. Press 1980
10. Ehresmann, C.: Categories et structures. Paris: Dunod 1965
11. Fisk, C.J., Caskey, D.L., West, L.L.: ACCL: Automated circuit card etching layout. Proc. IEEE, 55, No. 11, pp. 1971–1982, (1967)
12. Gabow, H.: Implementation of algorithms for maximum matching on non-bipartite graphs. Ph. Dissertation, Stanford University 1973
13. Gauß, K.F.: Zur mathematischen Theorie der elektrodynamischen Wirkungen. (1933), Königliche Gesellschaft der Wissenschaften zu Göttingen, (1877), Vol. 5, p. 605
14. Groh, U.: Optimale Einbettung von Graphen mit festem Rand. Diplomarbeit, Saarbrücken 1983
15. Hadlock, F.: Finding a maximum cut of a planar graph in polynomial time. SIAM J. on Comput. 4, 221–225 (1975)
16. Hashimoto, A., Stevens, J.: Wiring routing by optimizing channel assignment within large apertures. Proceedings of the Eight Design Automation Workshop, 1971, pp. 155–169. New York: IEEE Computer Soc. Press 1971
17. Hotz, G.: Eine Algebraisierung des Syntheseproblems für Schaltkreise. EIK 1, 185–205, 209–231 (1965)
18. Hotz, G.: Schaltkreistheorie, pp. 243–330. Berlin, New York: de Gruyter 1974
19. Sechen, C., Sangiovanni-Vincentelli, A.: The TimberWolf placement and routing package. IEEE J. Solid State Circuits SC-20, 510–522 (1985)
20. Kirkpatrick, S., Gelatt, C.D., Vecchi, M.P.: Optimization by simulated annealing. Science 220, 671–680 (1983)
21. Lawler, E.G.: Combinatorial optimization theory, Chap. 6, pp. 217–239. New York: Holt, Rinehart and Winston 1976
22. Leiserson, C.E., Pinter, R.Y.: Optimal placement for river routing. SIAM J. Comput. 12, 447–462 (1983)

23. Luk, W.K., Vuillemin, J.: Recursive implementation of optimal time VLSI integer multipliers. Proc. IFIP Congress 83, pp. 155–168. Amsterdam: North-Holland 1983
24. Messerschmidt, J.: Linguistische Datenverarbeitung mit Comskee. Stuttgart: B.G. Teubner 1984
25. Metropolis, N., Rosenbluth, W., Rosenbluth, N., Teller, A., Teller, E.: Equation of state calculations by fast computing machines. *J. Chem. Phys.* **21**, 1087–1092 (1953)
26. Osthof, H.G.: Der minimale Kreis um eine endliche Punktmenge. Diplomarbeit, Saarbrücken 1983
27. Pinter, R. Y.: Optimal layer assignment for interconnect. ICCC 1982, pp. 398–401. New York: IEEE Computer Soc. Press 1982
28. Pinter, R. Y.: River routing: Methodology and analysis. Third Caltech Conference on Very Large Scale Integration, pp. 141–163. Berlin, Heidelberg: Springer 1983
29. Sahni, S., Bhatt, A.: The complexity of design automation. Proc. 17th Design Autom. Conf. 1980, pp. 402–411. New York: IEEE Computer Soc. Press 1980
30. Sahni, S., Gonzalez, T.: P-complete approximation problems. *J. Assoc. Comput. Mach.* **23**, 555–565 (1976)
31. Schmitt, F.J.: Rechtwinklige Verdrahtung optimaler Layouts. Diplomarbeit, Saarbrücken 1985
32. Schworm, K.: Layouts unter Berücksichtigung von Leitungsbreiten und Modulgrößen. Diplomarbeit, Saarbrücken 1985
33. Shamos, M.I., Hoey, D.: Closest-point problems. Proc. 16th IEEE Symp. on Foundations of Comput. Sci., Oct. 1975, pp. 151–162. New York: IEEE Computer Soc. Press 1975
34. Sklansky, J.: Conditional-sum addition logic. *IRE-EC* **9**, 226–231 (1960)
35. Spaniol, O.: Arithmetik in Rechenanlagen. Teubner Studienreihe Informatik, Band 34, pp. 83–84. Stuttgart: B.G. Teubner 1976
36. Stüben, K.: Algebraic multigrid (AMG): Experiences and comparisons. Arbeitspapiere der GMD Bonn, Nr. 23, März 1983
37. Stoer, J., Bulirsch, R.: Einführung in die Numerische Mathematik II. Berlin, Heidelberg, New York: Springer 1978
38. Vecchi, M.P., Kirkpatrick, S.: Global wiring by simulated annealing. *IEEE Trans CAD-2*, 215–222 (1983)
39. Wallace, C.S.: A suggestion for a fast multiplier. *IEEE Trans. EC-13*, 14–17 (1964)
40. Weidner, W.: Der topologische und algebraische Abschluß freier \times -Kategorien. Dissertation, Saarbrücken 1977, pp. 1–77

Eingegangen am 4. September 1985
Angenommen am 14. November 1985



◀ *Günter Hotz*. Geboren am 16.11.1931. Studium: Mathematik und Physik in Frankfurt und Göttingen. Promotion 1958 in Göttingen. 1958–1962 Entwicklungsingenieur in der Firma Telefunken. Seit 1962 als Assistent, Dozent und Professor in Saarbrücken. Gründungspräsident der Gesellschaft für Informatik. Mitglied der Akademie der Wissenschaften und Literatur, Mainz

Reiner Kolla. Geboren 1957 in Differdange (Saarland). Von 1978–1982 Studium der Informatik an der Universität des Saarlandes mit Abschluß als Diplom-Informatiker. Seit 1982 wissenschaftlicher Mitarbeiter an dem Fachbereich Angewandte Mathematik und Informatik an der Universität des Saarlandes und in dem der Universität des Saarlandes und der Universität Kaiserslautern angegliederten Sonderforschungsbereich „VLSI-Entwurfsmethoden und Parallelität“.



Paul Molitor. Geboren 1959 in Luxemburg. Von 1978–1982 Studium der Informatik an der Universität des Saarlandes mit Abschluß als Diplom-Informatiker. Seit 1982 wissenschaftlicher Mitarbeiter an dem Fachbereich Angewandte Mathematik und Informatik an der Universität des Saarlandes und in dem der Universität des Saarlandes und der Universität Kaiserslautern angegliederten Sonderforschungsbereich „VLSI-Entwurfsmethoden und Parallelität“.



◀ *Bernd Becker*. Geboren 1954 in Hermeskeil (Rheinland-Pfalz), von 1973–82 Studium der Mathematik und Informatik in Saarbrücken, Diplom 1979, Promotion 1982, seit 1979 wiss. Mitarbeiter bei Prof. G. Hotz, seit 1983 Mitglied im SFB 124 „VLSI-Entwurfsmethoden und Parallelität“ in Saarbrücken