

# CADIC

## Ein System zum hierarchischen Entwurf integrierter Schaltkreise

Bernd Becker, Günter Hotz, Reiner Kolla, Paul Molitor, Hans Georg Osthof  
Fachbereich 10, Universität des Saarlandes, 6600 Saarbrücken

### Zusammenfassung

In diesem Beitrag wird ein hierarchisches Entwurfssystem vorgestellt, das auf einem algebraischen Ansatz, einem Netzkalkül, basiert, der eine einfache und komfortable Behandlung logischer wie auch geometrischer Information zuläßt. Verschiedene Ausprägungen dieses Kalküls erlauben eine Anpassung an verschiedene Entwurfsebenen. Das System rankt sich um diesen Kalkül wie etwa Algol um die Numerik, wobei sich der augenblickliche Implementierungsstand auf die logisch-topologische Entwurfsebene und den Übergang zur topographischen Ebene konzentriert. Neben einer Schilderung des Kalküls der logisch topologischen Netze wird in diesem Aufsatz auch eine kurze Übersicht über hierarchische Syntheseverfahren, wie sie zur Zeit im System vorhanden sind, gegeben. Dabei zeigt es sich, daß der Kalkül vor allem platzeffiziente hierarchische Darstellung wie auch zeiteffiziente hierarchische Algorithmen in besonderer Weise unterstützt.

### 1. Einleitung

Der wachsende technologische Fortschritt bei der Höchstintegration elektronischer Schaltkreise ermöglicht heutzutage die Unterbringung von Systemen mit vielen hunderttausend Transistoren auf einem Plättchen Silizium, so daß infolgedessen selbst ein erfahrener IC-Entwickler kaum noch in der Lage ist, mit vernünftigem Zeitaufwand einen korrekt arbeitenden Schaltkreis zu entwickeln und zu verifizieren (seine Korrektheit gar zu beweisen). Hinzu kommt, daß mit dem technologischen Fortschritt auch die Anzahl der Anwendungen der Mikroelektronik in den verschiedensten Gebieten wächst, wobei in vielen Fällen der Schaltkreisentwickler nicht mit dem spezifischen Anwendungsproblem vertraut ist, während der Anwender nicht in der Lage ist, das komplexe Problem des IC-Entwurfs zu bewältigen. Um diese Schwierigkeiten zu überwinden, gehört zum Fortschritt im Gebiet der Mikroelektronik auch ein gründliches Studium der Möglichkeiten und Methoden zur Entwurfsautomatisierung, damit Systeme geschaffen werden, die es auch einem im Entwurf integrierter Schaltungen weniger erfahrenen Anwender erlauben, große Schaltungen zu entwickeln. Dies erfordert zum einen eine gewisse Abstraktionsebene, da Schaltkreisbeschreibungen auf niedriger (physikalischer) Ebene nicht nur mit technischen Details überladen und damit kaum zu übersehen sind, sondern auch sich gegenüber geringen Veränderungen in der Technologie als sehr unflexibel erweisen. Zum anderen muß es dann aber auch möglich sein, eine korrekte Spezifikation auf abstrakter Ebene in eine korrekte Implementierung auf niedriger Ebene automatisch übersetzen zu können, was eine präzise mathematische Grundlage für alle Bestandteile des Systems erfordert.

Ein wichtiger Schritt in diese Richtung wurde zu Beginn der achtziger Jahre von C.A. Mead und L.A. Conway ([14]) getan, gefolgt von verschiedenen anderen Entwurfssystemen wie beispielsweise [5],[6],[13],... um sehr verschiedene Ansätze zu nennen. Die Beschreibungsmittel dieser Systeme sind mehr oder weniger technologieunabhängig. Es bleibt jedoch schwierig, etwa die Korrektheit von Entwürfen zu beweisen oder zu übersehen, und zwar nicht zuletzt deswegen, weil die oben genannten Systeme eher pragmatischer Natur sind.

Es ist darüberhinaus wohl auch nicht möglich, ein universelles Entwurfssystem zu konstruieren, mit dem sich für alle denkbaren Klassen von Schaltkreisen optimale Resultate erzielen lassen. Wahrscheinlicher ist es, daß man häufig neue Systeme für spezielle Klassen von Schaltkreisen implementieren möchte, was eine problemorientierte Sprache, die dies in einfacher Weise erlaubt, erfordert. Die Datentypen einer solchen Sprache sollten auf einer algebraischen Struktur basieren, so daß Korrektheitsbeweise auf einer präzisen Grundlage möglich sind. Es gibt einige Arbeiten, die algebraische Methoden zur Definition von Hardwarekomponenten und deren Semantik einführen ([4],[9],[21]). Eine ähnliche Philosophie wird auch im CADIC System, dessen Grundlagen auf einen algebraischen Kalkül

von Hotz aus dem Jahre 1965 ([7]) zurückgehen, genutzt. Im Gegensatz jedoch zu anderen Projekten wird hier gezielt Logik und Geometrie in einem Kalkül erfaßt. Verschiedene Entwurfsebenen korrespondieren dabei zu verschiedenen Netzalgebren, wobei Übergänge zwischen diesen durch Homomorphismen beschrieben werden können. Eine genauere Darstellung des theoretischen Hintergrundes findet sich in [10],[16],[18]. Wir beschränken uns hier auf eine kurze zusammenfassende Darstellung des Kalküls sowie eine knappe Schilderung des darauf aufbauenden Systems und damit gewonnenen Erfahrungen. (siehe auch [2])

## 2. Der Netzkalkül

Die boolesche Algebra ist der klassische Kalkül zur Behandlung logischer Schaltkreise und war ausreichend, solange der Verdrahtungsaufwand der Schaltkreiskomponenten untereinander und damit deren geometrische Anordnung eine untergeordnete Rolle spielten. Dies gilt jedoch nicht mehr bei integrierten Schaltkreisen, wo die Kosten für die Verdrahtung durchaus mit den Kosten für Gatter konkurrieren, so daß die Entwicklung eines Kalküls, der neben der logischen Funktion auch die Behandlung geometrischer Informationen erlaubt, notwendig wird. Dabei sollte der geometrische Teil nicht zu genau sein, um nicht streng an einem Fabrikationsprozeß zu haften, ebenso wie der funktionelle Teil nicht an den tiefsten Beschreibungsebenen etwa der Transistorebene liegen, sondern vielmehr auf digitalen Werten aufbauen sollte. Eine erste Erweiterung der booleschen Algebra unter diesem Gesichtspunkt wurde 1965 mit der x-Kategorie (siehe [7]) gegeben, die zum Zwecke des VLSI-Entwurfs verallgemeinert wurde. Wir stellen diesen Kalkül im folgenden zusammenfassend vor.

Man betrachte einen Schaltkreis ausgelegt im Innern eines Rechtecks  $R$ , wobei die äußeren Anschlüsse auf dem Rand von  $R$  liegen. Das Innere von  $R$  bestehe aus Rechtecken verschiedener Formen und Ausdehnungen, die die Grundzellen des Schaltkreises darstellen, deren Seiten parallel zu den Seiten von  $R$  verlaufen, und deren Anschlüsse ebenfalls jeweils auf ihrem Rand liegen. Die Verbindungen zwischen diesen Zellen werden durch Leitungen einer gewissen Breite, die nach gewissen geometrischen Regeln in gewissen Schichten verlaufen, realisiert. Um nun die "wesentlichsten" Eigenschaften dieses Schaltkreises in einem hinreichend abstrakten Kalkül festzuhalten, vollziehe man zunächst folgende Abstraktion (siehe Abbildung 2.1): Man projiziere die Verbindungsleitungen in die Ebene und vergesse ihre Breite. Damit haben Leitungen weder Ausdehnung noch Schichten, so daß sie als einfache Polygonzüge in der Ebene angesehen werden können, die als untereinander kreuzungsfrei und nicht überlappend angenommen werden können, sofern man entstandene Überkreuzungen sowie Verzweigungen als Grundzellen betrachtet. (Wir betrachten im folgenden Kreuzungen, Verzweigungen und offene Enden von Leitungen als Zellen, zeichnen aber der Einfachheit wegen dafür keine Rechtecke sondern den Verdrahtungsverlauf, den sie realisieren.)

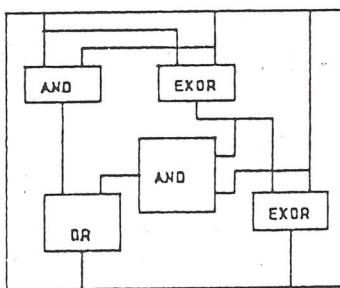


Abbildung 2.1

Jede Grundzelle des Schaltkreises hat einen Namen, der für den Typ der Zelle steht (, wie etwa 'AND' für eine Zelle die die Konjunktion zweier Eingabewerte durchführt). Darüberhinaus habe jeder Zellentyp definitionsgemäß eine nördliche, südliche, westliche und östliche Seite, die Information über die Anzahl (den Typ) und die Reihenfolge der Anschlüsse darauf trägt. So dürfen zwei Vorkommen desselben Zellentyps zwar verschiedene geometrische Ausdehnung haben, müssen aber die Anzahl, ggf. den Typ und die Reihenfolge der Anschlüsse auf den Seiten beibehalten. Dabei muß die nördliche

Seite einer Zelle nicht im Norden liegen (, die Zelle kann gedreht werden,) sondern dient nur zur formalen Unterscheidung der Seiten. Das Resultat dieser abstrakten Sicht ist also eine Formation von Linien und Rechtecken in der Ebene, die wir **logisch-topographisches Netz** nennen.

Seien  $N_1$  und  $N_2$  logisch-topographische Netze. Wir betrachten nun zwei einfache Operationen. Das Nebeneinandersetzen,  $N_1 \ominus N_2$ , (Übereinandersetzen,  $N_1 \oplus N_2$ ,) ist genau dann definiert, wenn die östliche (südliche) Seite von  $N_1$  auf die westliche (nördliche) Seite von  $N_2$  geometrisch paßt. Das Ergebnis des Zusammensetzen ist dabei das Netz, das man erhält, indem man die Operanden an den beteiligten Seiten übereinanderlegt, die Anschlüsse miteinander identifiziert, und die beteiligten Ränder löscht. Abbildung 2.2 verdeutlicht dies.

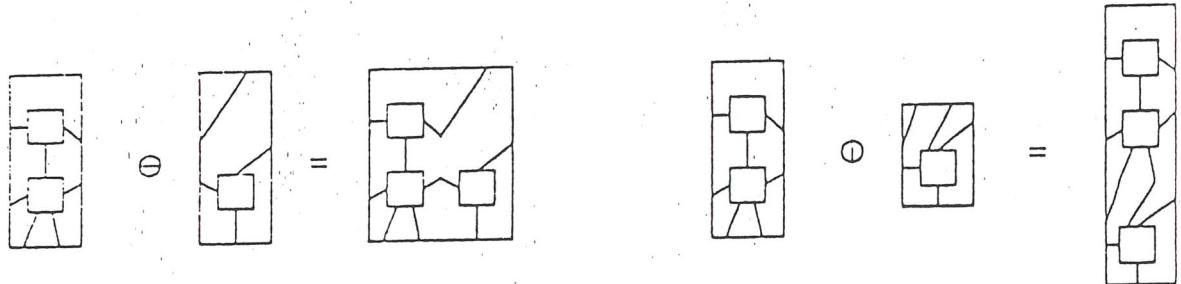


Abbildung 2.2

Ausgehend von einer Menge von Grundzellentypen einschließlich Netzen, die nur aus Leitungen einer gewissen Länge oder leeren Flächen bestehen, könnte man nun große logisch-topographische Netze mit diesen einfachen Operationen aufbauen. Ein großer Nachteil dieses Zugangs wäre jedoch, daß man dabei sehr genau über geometrische Details nachdenken müßte. Wir gehen daher noch einen Schritt weiter und betrachten zwei logisch-topographische Netze als äquivalent, wenn sie im wesentlichen die gleiche topologische Struktur haben, d.h. wenn sie sich durch eine Folge elementarer Deformationen ineinander überführen lassen. Elementare Deformationen seien dabei stetige Deformationen von Leitungen, Dehnungen, Verschieben und Drehen von Zellen, Verschieben aber nicht Vertauschen von äußeren Anschlüssen auf einer Seite eines Rechteckes und weitere. Wichtig ist dabei, daß keine der Deformationen eine weitere Kreuzung oder Überlappung von Leitungen oder Zellen produzieren darf. Eine präzisere Darstellung dieses Äquivalenzbegriffs findet sich in [16]. Abbildung 2.3 zeigt ein Beispiel zweier in diesem Sinne äquivalenter logisch-topographischer Netze.

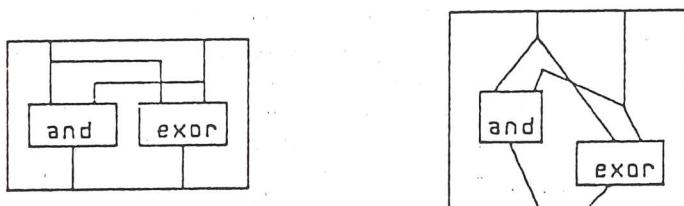


Abbildung 2.3: Zwei äquivalente logisch-topographische Netze

Diese Äquivalenzrelation erzeugt eine Klasseneinteilung auf der Menge der logisch-topographischen Netze. Betrachten wir diese Klassen, so unterscheiden wir nicht länger geometrische Details wie Länge oder Abstand, sondern nur noch Unterschiede im topologischen Sinn, wie beispielsweise die Reihenfolge, in der eine Leitung andere Leitungen kreuzt, oder wo sie verzweigt. Wir nennen eine solche Äquivalenzklasse ein **logisch-topologisches Netz** und ein Element einer Klasse einen (topographischen) Repräsentanten. In gleicher Weise, wie nun gängige Programmiersprachen etwa den Datentyp *real* für numerische Anwendungen vorsehen, sieht CADIC den Datentyp logisch-topologisches Netz für den Entwurf integrierter Schaltkreise vor. Dabei hängt die Qualität dieses Datentyps stark von den dafür verfügbaren Operationen und der algebraischen Struktur der Menge  $\text{NET}(A)$  der logisch-topologischen Netze über einer Grundzellenmenge  $A$  ab.

Zunächst kann man  $\Theta, \emptyset$  in natürlicher Weise auf logisch-topologische Netze erweitern, indem man sie auf geeigneten topographischen Repräsentanten (falls diese existieren,) ausführt und dann die Klasse des daraus entstandenen Gebildes betrachtet. Es ist leicht einzusehen, daß diese Operation wohldefiniert und genau dann definiert ist, wenn die Anzahl (bzw. die Folge der Typen) der Anschlüsse auf den beteiligten Seiten übereinstimmt. (siehe Abbildung 2.4)

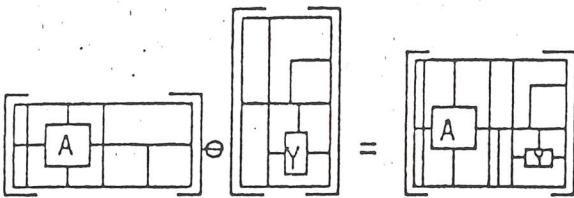
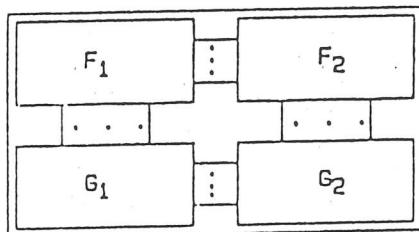


Abbildung 2.4

Diese Operationen ermöglichen es uns, ausgehend von einer Menge  $A$  von Grundzellen und Verdrahtungszellen, durch sukzessives Anwenden jedes Netz aus  $NET(A)$  zu erzeugen. Damit kann man Netze durch arithmetische Ausdrücke über Grundzellen und  $\Theta, \emptyset$  darstellen, wie man Zahlen durch Ausdrücke über  $+, *, \dots$  darstellen kann. Es ist dabei natürlich auch möglich, daß verschiedene Ausdrücke das gleiche Netz darstellen (siehe Abbildung 2.5). Aber diese Ausdrücke sind in dem Sinne gleich, daß sie sich durch Anwenden einer endlichen Menge von Rechenregeln, wie etwa Assoziativität, Distributivität (Abb. 2.5) und kompliziertere Regeln, ineinander überführen lassen. So kann etwa eine Eigenschaft für ein logisch topologisches Netz gezeigt werden, indem man sie für einen geeigneten Ausdruck zeigt, und die Invarianz der Eigenschaft unter Anwendung dieser Rechenregeln nachweist.  $NET(A)$  unter den Operationen  $\Theta, \emptyset$  wird auch die von  $A$  erzeugte freie Netzalgebra genannt ([16], [18]).



$$(F_1 \Theta F_2) \emptyset (G_1 \Theta G_2) = (F_1 \emptyset G_1) \Theta (F_2 \emptyset G_2), \text{ falls beide Seiten definiert sind.}$$

Abbildung 2.5

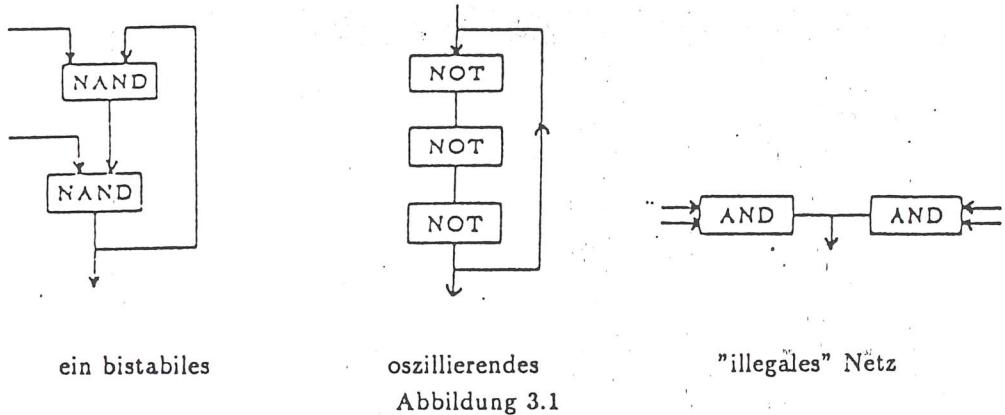
Die Fragen, ob und wie man entscheiden kann, ob zwei Netze gleich (zwei Ausdrücke äquivalent) oder das eine Teil des anderen ist, erwachsen unmittelbar aus dieser Betrachtungsweise. In [10] wird gezeigt, daß je nach Allgemeinheit diese Fragen effizient (in Linearzeit) bis schwierig (NP-vollständig) zu beantworten sind, wobei die praktisch interessanten Fälle zu den effizient lösbarsten gehören.

### 3. Die Semantik logisch-topologischer Netze

Wir haben uns bis hierhin hauptsächlich mit der geometrischen Bedeutung logisch-topologischer Netze beschäftigt und zusammenfassend dargestellt, in welcher Weise geometrische Information in unserem Kalkül gehandhabt werden kann. Wir wollen uns nun der funktionalen Bedeutung logisch-topologischer Netze zuwenden. Diese spielt dann eine herausragende Rolle, will man einen Entwurf analysieren und gewisse Eigenschaften verifizieren. Da die Kosten zum Beheben von Entwurfsfehlern um so schwerer wiegen, je später ein Fehler entdeckt wird, ist gerade die Korrektheit eines Entwurfs genauestens zu prüfen und wenn möglich gar zu beweisen. Dies erfordert vor allem ein präzises mathematisches Modell des Verhaltens. In [3], [11], [16] wird gezeigt, wie das Verhalten logisch-topologischer

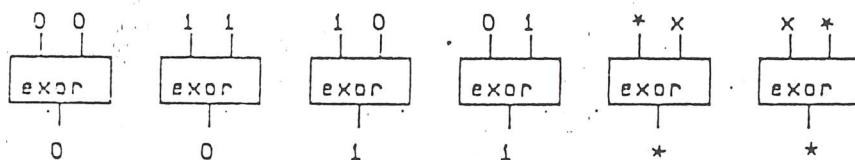
Netze durch Homomorphismen in ein Relationenkalkül extrahiert werden kann. Wir fassen hier nur die wichtigsten Punkte zusammen.

Der Einfachheit wegen beschränken wir uns hier auf Grundbausteinsysteme, die nur Verdrahtungsbausteine oder Zellen mit kombinatorischem Verhalten enthalten. Selbst mit dieser Einschränkung ist es möglich, beispielsweise bistabile, oszillierende oder "illegale" Schaltkreise (siehe Abbildung 3.1) zu definieren.



Somit kann das Verhalten in diesem Fall schon nicht mehr durch boolesche Funktionen vollständig definiert werden, sondern der Übergang zu einem Kalkül boolescher Relationen über einem dreiwertigen Wertebereich  $B = \{0, 1, *\}$  wird nötig. 0, 1 stehen dabei für die booleschen Werte, während  $*$  für die Menge  $\{0, 1\}$  (ein instabiler Wert) steht. Die grundlegende Idee bei der Definition der Semantik logisch-topologischer Netze ist nun die folgende:

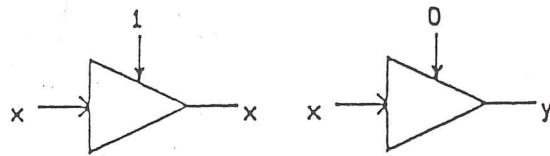
Man betrachte alle möglichen (dreiwertigen) Belegungen der äußeren Anschlüsse eines Netzes  $N$ . Eine solche Belegung kann durch Worte  $s_n, s_s, s_w, s_e \in B^*$ , die jeweils für die Belegung der nördlichen, südlichen, westlichen und östlichen Seite von  $N$  stehen, beschrieben werden. Dann ist das Verhalten eines Netzes  $N$  gegeben durch die Menge aller möglichen Konfigurationen der äußeren Anschlüsse, also eine Relation  $R(N) \subseteq B^{|s_n|} \times B^{|s_s|} \times B^{|s_w|} \times B^{|s_e|}$ , wobei etwa  $|s_n|$  die Länge einer Belegung der nördlichen Seite, also die Anzahl der Anschlüsse im Norden, von  $N$  bezeichnet. (Abbildung 3.2)



$$R(exor) = \{(00, 0, \epsilon, \epsilon), (11, 0, \epsilon, \epsilon), (10, 1, \epsilon, \epsilon), (01, 1, \epsilon, \epsilon), (*0, *, \epsilon, \epsilon), \\ (*1, *, \epsilon, \epsilon), (0*, *, \epsilon, \epsilon), (1*, *, \epsilon, \epsilon), (***, *, \epsilon, \epsilon)\}$$

Abbildung 3.2

Man kann nun ausgehend von den Verhalten von Grundbausteinen durch "Zusammensetzen" der Relationen das Verhalten von Netzen berechnen. Abbildung 3.3 gibt ein weiteres Beispiel, das zeigt, daß durch den Übergang auf Relationen, hochohmiges Verhalten wie auch der bidirektionale Charakter von Leitungen gut beschrieben werden kann.



$$\forall x, y \in \{0, 1, *\}$$

Abbildung 3.3: Semantik eines Tri-State-Treibers

In [3],[11] wird an Beispielen gezeigt, wie man mit diesem Modell die Korrektheit großer kombinatorischer Schaltkreise beweisen kann. Ein wichtiger Punkt dabei ist auch, daß diese Schaltkreise eine kurze rekursive Beschreibung haben. Diese Beschreibungstechnik bildet den Kern des zur Zeit implementierten Systems und hatte auch einen besonderen Einfluß auf die Entwicklung anderer Komponenten.

#### 4. Spezifikation großer Netze durch rekursive Gleichungen

In diesem Abschnitt wollen wir eine Methode zur Spezifikation logisch-topologischer Netze vorstellen, die zum einen sehr kurze, übersichtliche Definitionen für regelmäßige Schaltungen zuläßt, und zum anderen eine feine Beschreibungshierarchie liefert. Wir beschränken uns auch hier auf eine exemplarische Darstellung und verweisen für eine eingehendere Untersuchung auf [10].

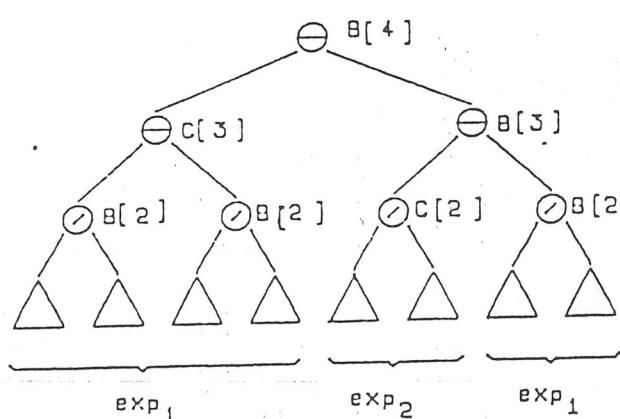
Seien  $A_1, A_2$  logisch-topologische Netze, gegeben durch Ausdrücke  $\exp_1, \exp_2$ , derart daß die Verknüpfungen  $A_i \ominus A_j$  sowie  $A_i \otimes A_j$  für  $i, j \in \{1, 2\}$  definiert sind. Man betrachte nun folgende Gleichungen:

$$\begin{aligned} B[k] &= C[k-1] \ominus B[k-1]; \quad (k \geq 3) \\ C[k] &= B[k-1] \ominus B[k-1]; \quad (k \geq 3) \\ B[2] &= A_1 \otimes A_1; \quad A_1 = \exp_1; \\ C[2] &= A_2 \otimes A_2; \quad A_2 = \exp_2; \end{aligned}$$

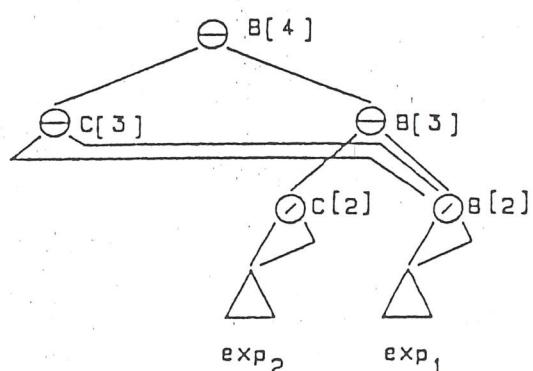
Diese Gleichungen definieren eine ganze Familie logisch-topologischer Netze. Das Netz  $B[4]$  erhält man etwa durch rekursives Anwenden obiger Gleichungen wie folgt

$$\begin{aligned} B[4] &= C[3] \ominus B[3] = (B[2] \ominus B[2]) \ominus (C[2] \ominus B[2]) \\ &= ((A_1 \otimes A_1) \ominus (A_1 \otimes A_1)) \ominus ((A_2 \otimes A_2) \ominus (A_1 \otimes A_1)) \\ &= ((\exp_1 \ominus \exp_1) \ominus (\exp_1 \ominus \exp_1)) \ominus ((\exp_2 \ominus \exp_2) \ominus (\exp_1 \ominus \exp_1)) \end{aligned}$$

Offensichtlich kann man zur Darstellung eines Netzes den Syntaxbaum eines Ausdruckes benutzen.



Ein Syntaxbaum zu  $B[4]$



Ein Syntaxgraph zu  $B[4]$

Abbildung 4

Expandiert man etwa  $B[4]$ , so stellt man fest, daß während der Berechnung die Gleichung zu  $B[2]$  dreimal und die Gleichung zu  $A_1$  gar sechsmal angewandt wird. Der Syntaxbaum zu  $B[4]$  enthält somit viele gleiche Teilbäume, die Darstellungen von Variablen ( $B[2], A_1$ ) im Gleichungssystem sind. Man kommt nun zu einer sehr kompakten Darstellung eines Netzes, indem man diese Teilbäume, die zu Variablen gehören, nur einmal speichert, d.h. von einem Baum zu einem Graphen übergeht. Dies führt nun zu sehr kompakten hierarchischen Darstellungen (, man bemerke, daß ein Syntaxbaum exponentiell größer sein kann als der zugehörige Syntaxgraph,) die auch schneller berechnet werden können als Darstellungen durch Bäume.

Der geringe Speicherbedarf hierarchischer Darstellungen ist schon ein beträchtlicher Vorteil und kann sich auch auf die Rechenzeit für weiterverarbeitende Verfahren auswirken, da unter Umständen auch bei großen Netzen die ganze Beschreibung noch im Hauptspeicher gehalten werden kann. Hierarchische Darstellungen werden aber vor allem dann interessant, wenn auch hierarchische Verfahren zur Weiterverarbeitung benutzt werden können, d.h. Verfahren, die hierarchische Darstellungen in andere hierarchische Darstellungen transformieren. Hier läßt sich im Vergleich zu nicht-hierarchischen Verfahren häufig ein großer Laufzeitgewinn erzielen.

Im CADIC System wird der Versuch unternommen, das System in allen Teilen auf hierarchische Verarbeitung auszulegen, d.h. ein durchgängig hierarchisches Entwurfssystem zu erstellen. Im folgenden Abschnitt geben wir eine kurze Übersicht über hierarchische Syntheseverfahren, soweit sie zur Zeit implementiert sind.

## 5. Hierarchische Syntheseverfahren

### 5.1 Schichtzuweisung

Bisher haben wir von der Verdrahtungsschicht abstrahiert. Dies wirft das Problem auf, im Verlauf der Synthese eines integrierten Schaltkreises eine Einbettung aller Leitungen in Schichten vornehmen zu müssen. In der Literatur wird dabei meist, durch die Technologie motiviert, das Kontaktminimierungsproblem für 2 Schichten untersucht ([17],[20]). Da der beste bekannte Algorithmus für 2 Schichten Zeitkomplexität  $O(n^3)$ , ( $n$  ist dabei die Schaltkreisgröße,) was für große Schaltkreise schon nicht mehr praktikabel ist, wird das Schichtzuweisungsproblem in CADIC hierarchisch zu lösen versucht. Dabei ist klar, daß eine hierarchische Lösung in Bezug auf eine vorgegebene hierarchische Darstellung, was dadurch charakterisiert ist, daß alle Vorkommen derselben Variablen die gleiche Schichtzuweisung haben müssen, schlechter sein kann als eine nicht-hierarchische Lösung. Betrachtet man also Optimierungsprobleme, so sind trade-offs zwischen der Kompaktheit der hierarchischen Darstellung und der Qualität des Resultats zu erwarten. Für die Schichtzuweisung heißt das, umso kompakter die Darstellung (und umso mehr Nebenbedingungen an die Schichtzuweisung gestellt werden), umso größer ist die Anzahl der benötigten Kontakte. In [19] wurde das Problem der hierarchischen Schichtzuweisung gerade unter diesem Aspekt an einigen Beispielen untersucht, wobei sich überraschend herstellte, daß meistens sehr kompakte Darstellungen existieren, die nicht wesentlich mehr Kontakte benötigen als die flache (nicht hierarchische) Darstellung, daß aber auch geringe Vergrößerungen einer vorgegebenen Hierarchie dramatische Verbesserungen in der Zahl der Kontakte bringen können. Da die durch eine Spezifikation durch rekursive Gleichungen gegebene Hierarchie möglicherweise schlecht, d.h. zu streng, für hierarchische Optimierungen ist, kann der Benutzer in CADIC diese Hierarchie vergröbern, um so Rechenzeit gegen Qualität auszuhandeln.

Zur Zeit wird ein probabilistischer Algorithmus zur hierarchischen Schichtzuweisung benutzt, der die Methode des simulated annealing nutzt ([15]). Ein deterministischer Algorithmus zur hierarchischen Kontaktminimierung, der das Optimum in Bezug auf die gegebene hierarchische Darstellung findet, ist nicht bekannt. Nach bisherigen Erfahrungen (siehe auch [15]) liefert die probabilistische Methode ziemlich gute Resultate.

### 5.2 Automatische Generierung der Versorgungsleitungen

Man kann natürlich die Verdrahtung für die Spannungsversorgung auf logisch-topologischer Ebene berücksichtigen, was allerdings in vielen Fällen eine Spezifikation schwerfälliger macht. Das CADIC System bietet daher auch die Möglichkeit, die Verdrahtung der Versorgungsleitungen automatisch

erzeugen zu lassen. Dies geschieht durch einen hierarchischen Algorithmus, dessen genaue Darstellung sich in [10] findet.

Ein Netz  $N$  kann nun in ein Netz  $N^{(\epsilon)}$  mit Versorgungsleitungen überführt werden, indem man top down durch einen Syntaxbaum für  $N$  Belegungsmuster für die Versorgungsanschlüsse propagiert. Die Belegungsmuster ergeben sich dabei in Abhängigkeit davon, welche Operation  $(\emptyset, \Theta)$  vorliegt, und ob einer oder beide Operanden reine Verdrahtungsnetze sind (, und demnach selbst keine Spannungsversorgung benötigen). Bei Zellvorkommen werden schließlich die entsprechenden Belegungsmuster durch Einfügen der benötigten Verdrahtung realisiert. Wir verzichten hier auf eine genaue Auflistung aller Fälle und illustrieren statt dessen in Abbildung 5.1 die Arbeitsweise dieses Verfahrens.

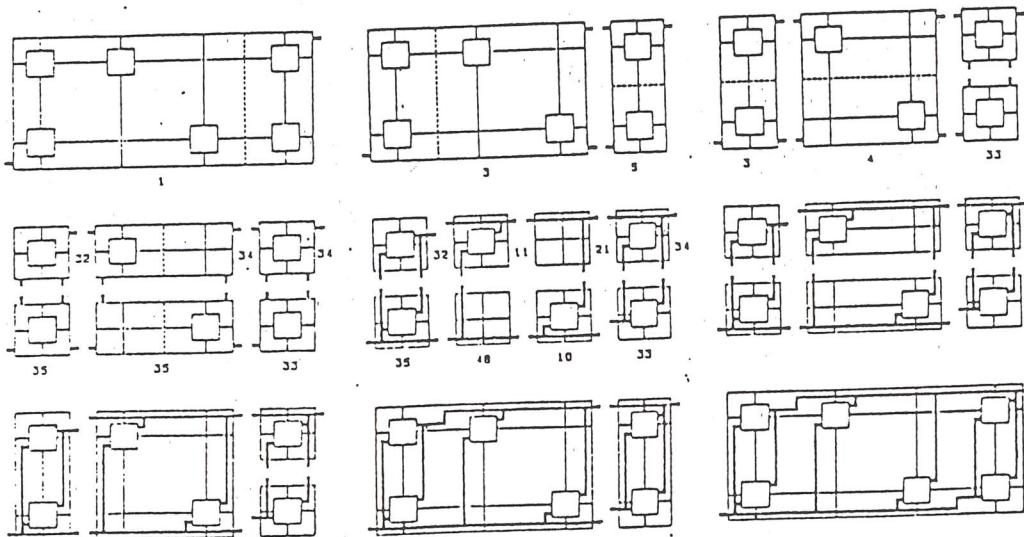


Abbildung 5.1

In [10] wird gezeigt, wie sich dieser Algorithmus zu einem hierarchischen Algorithmus verallgemeinern lässt, der mit entsprechender Vergrößerung der Hierarchie (um höchstens einen konstanten Faktor) das gleiche Ergebnis liefert wie das nicht-hierarchische Vorgehen.

### 5.3 Synthese topographischer Netze

In diesem Abschnitt wollen wir kurz auf das Problem eingehen, eine feste nach gewissen Maßen "gute" geometrische Einbettung eines logisch-topologischen Netzes zu erzeugen. Dies wird im CADIC System zur Zeit mit einem heuristischen Verfahren getan, das eine Einbettung eines logisch-topologischen Netzes auf einem rechtwinkligen Gitter liefert und dabei die resultierende Fläche zu minimieren versucht.

Das Verfahren kann durch folgenden divide&conquer Prozeß charakterisiert werden:  
Seien  $S_1, S_2$  logisch-topologische Netze, so daß  $S_1 \Theta S_2$  (bzw.  $S_1 \ominus S_2$ ) definiert sind, und seien Repräsentanten  $\Gamma(S_1), \Gamma(S_2)$  auf dem Gitter schon erzeugt. Dann erhält man einen Repräsentanten  $\Gamma(S)$  von  $S = S_1 \Theta S_2$ , indem man beide Repräsentanten nebeneinanderlegt und durch einen rechtwinkligen Verdrahtungskanal miteinander kreuzungsfrei verbindet. Gibt man zusätzlich den Repräsentanten vor, daß sie in gewisser einfacher Weise verformt (gestreckt) werden können, so kann das lokale Plazierungsproblem mit den in [12] vorgestellten Methoden gelöst werden. In [10] wurden diese noch dahingehend verbessert, daß im Gegensatz zu [12] nicht nur die Fläche des Verdrahtungskanals, sondern auch die Leitungslängen im Kanal minimiert werden. Wie in Abbildung 5.2 illustriert, führt dies häufig zu besseren Einbettungen.

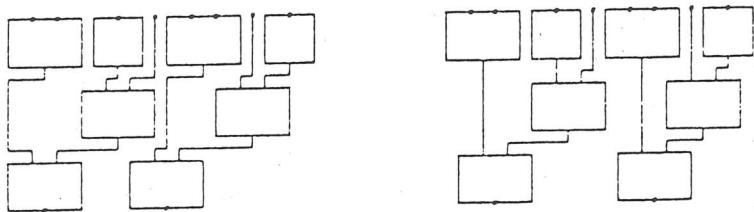


Abbildung 5.2: Resultat mit und ohne Leiterlängenminimierung

Dieser Algorithmus läßt sich leicht zu einem hierarchischen Algorithmus ausbauen, da er lokal auf den Rändern der Operanden arbeitet, und somit das Innere mehrfach vorkommender Teile nur einmal behandelt werden muß. Genaueres darüber findet sich ebenfalls in [10].

## 6. Schlußbemerkungen

Wir haben einen Kalkül vorgestellt, der nicht nur gleichzeitig den Umgang mit logischen und geometrischen Eigenschaften eines integrierten Schaltkreises erlaubt, sondern darüberhinaus auch in natürlicher Weise kompakte hierarchische Darstellungen von Schaltkreisen zuläßt. Die Frage, ob sich die Synthese eines integrierten Schaltkreises auf niedrigster Entwurfsebene ausgehend von der logisch-topologischen Ebene vollständig hierarchisch durchführen läßt, muß noch beantwortet werden. Dazu gehört auch die Frage, wie sich die Hierarchie auf abstrakter (z.B. logisch-topologischer) Ebene in eine Hierarchie auf konkreterer Ebene transformiert und was man für den Vorteil des Speicherplatz- und Laufzeitgewinns bei hierarchischer Synthese an Fläche und Zeitperformanz des Resultats bezahlen muß. Antworten darauf werden teilweise gegeben bei den oben vorgestellten Verfahren, die jedoch bei einer immer noch abstrakten Darstellung auf dem Gitter enden. Zwar würden sich auch geometrische Layouts mit verschiedenen Leiterbreiten und Abstandsregeln mit denselben Methoden erzeugen lassen, es fehlt jedoch ein Verfahren zur Dimensionierung der Schaltkreiskomponenten, was neben dem Studium der Hierarchie eine zukünftige Aufgabe sein wird. Zu der Synthese von Schaltungen aus abstrakteren Beschreibungen gesellt sich das weite Feld der Analyse. Dies beginnt bei Korrektheitsbeweisen auf hoher Beschreibungsebene und endet bei Testbarkeits- und Laufzeitanalysen, und der Frage, wie weit Hierarchie bei diesen Problemen hilft.

Zum Abschluß wollen wir die Kraft dieses Kalküls an einem Beispiel verdeutlichen. Abbildung 6.1 zeigt ein Gleichungssystem, das die vollständige Spezifikation eines schnellen  $2^k$ -bit Multiplizierers basierend auf einem modifizierten Wallace-tree (genaueres siehe [1]) für CADIC wiedergibt. (Untersucht wird zur Zeit die Frage, wie die Eingabe mit graphischen Methoden anstelle von Formeln ähnlich wie in [5] geschehen kann.) Das Layout eines 16-bit Multiplizierers, so wie es durch die im Abschnitt 5 vorgestellten Verfahren erzeugt wird, zeigt Abbildung 6.2. Für den Multiplizierer betrug dabei der Laufzeitgewinn durch hierarchische Synthesealgorithmen etwa den Faktor 50.

$$\begin{aligned}
 MU2 &= (+\Theta + \Theta \top \Theta +) \otimes (+\Theta \neg \Theta | \Theta |) \otimes \\
 &\quad (|\Theta AND \Theta \Gamma \Theta +) \otimes (\vdash \Theta \neg \Theta | \Theta |) \otimes \\
 &\quad (+\Theta + \Theta \top \Theta +) \otimes (+\Theta \neg \Theta | \Theta |) \otimes \\
 &\quad (|\Theta AND \Theta |) \otimes (|\Theta \Gamma \Theta \neg|) \otimes (|\Theta \Gamma \Theta +) \\
 CSA2 &= (|\Theta | \Theta \Gamma \Theta + \Theta \neg \Theta |) \otimes (+\Theta FA \Theta + \Theta \neg \Theta |) \otimes \\
 &\quad (|\Theta | \Theta \Gamma \Theta + \Theta + \Theta +) \otimes (+\Theta FA \Theta \neg \Theta | \Theta | \Theta \vdash) \otimes \\
 &\quad (\vdash \Theta \neg \Theta | \Theta | \Theta |)
 \end{aligned}$$

$$MULTREE[i, j] = MULTREE[i, j - 1] \Theta MULTREE[i, j - 1]$$

$$MULTREE[i, 0] = BMULTREE[i]$$

$$BMULTREE[i] = (|\Theta \Theta | \Theta \Theta |) \otimes$$

$$(+\_2^{i-1} \_2^{j-1}) \Theta BMULTREE[i - 1] \Theta +_{3,2^{i-1}-2,1} \otimes CSA2 \otimes$$

$$(+_{3 \cdot 2^{i-1} - 2,1} \ominus BMULTREE[i-1] \ominus +_{3 \cdot 2^{i-1} - 2,1}) \ominus (|\ominus\ominus| \ominus\ominus|)$$
$$BMULTREE[1] = MU2$$

Abbildung 6.1: Spezifikation einer Familie schneller Multiplizierer

## Literaturhinweise

- [1] B.Becker: "An easily testable optimal-time VLSI-multiplier", Preprints of EUROMICRO 85, Brüssel, S. 401-411, North-Holland.
- [2] B.Becker, G.Hotz, R.Kolla, P.Molitor, H.G.Osthof: "Hierarchical Design based on a Calculus of Nets", Proceedings of the 24<sup>th</sup> Design Automation Conference, S. 649-653, Miami 1987.
- [3] H.Bühler: "Korrekttheitsbeweise von rekursiv beschriebenen logisch-topologischen Netzen", Diplomarbeit, Saarbrücken 1986.
- [4] L.Cardelli: "An Algebraic Approach to Hardware Description and Verification", PH.D. Thesis, Edinburgh 1982.
- [5] E.Clarke, Y.Feng: "ESCHER - A geometrical Layout System for Recursively Defined Circuits", CMU-CS-85-150, Department of Computer Science, Carnegie-Mellon University, Pittsburgh 1985.
- [6] E.Hörbst, M.Nett, H.Schwärtzel: "VENUS - Ein Entwurf von VLSI-Schaltungen", Springer-Verlag Berlin Heidelberg New-York Tokyo 1986.
- [7] G.Hotz: "Eine Algebraisierung des Syntheseproblems für Schaltkreise", EIK 1, 1965, S.185-205,209-231.
- [8] G.Hotz, B.Becker, R.Kolla, P.Molitor: "Ein logisch-topologischer Kalkül zur Konstruktion von integrierten Schaltkreisen", Informatik: Forschung und Entwicklung, 1986, Heft 1 und 2, Springer Verlag 1986.
- [9] C.D.Kloos: "Towards a Formalization of Digital Circuit Design", TUM-I8604, February 1986, Technische Universität München.
- [10] R.Kolla: "Spezifikation und Expansion logisch-topologischer Netze", Dissertation, Saarbrücken 1986.
- [11] R.Kolla: "Proving correctness of logic topological nets", will appear as technical report, SFB124, Saarbrücken 1987.
- [12] C.E.Leiserson, R.Y.Pinter: "Optimal placement for river routing", SAM J. Comput. 12, S.447-462, 1983.
- [13] T.Lengauer, K.Mehlhorn: "The HILL-System: A Design Environment for the Hierarchical Specification, Compaction and Simulation of Integrated Circuit Layouts", MIT VLSI Conference 1984, S.139-149, Artech House.
- [14] C.A.Mead, L.A.Conway: "Introduction to VLSI-Systems", Addison Wesley 1980.
- [15] P.Molitor: "Layer Assignment by Simulated Annealing", Microprocesing and Microprogramming 16 (1985), S.345-350, North-Holland.
- [16] P.Molitor: "Über die Bikategorie der logisch-topologischen Netze und ihre Semantik", Dissertation, Saarbrücken 1986.
- [17] P.Molitor: "On the contact minimization problem", Proceedings of the STACS 87, Passau, S. 420-431, Lecture Notes in Computer Science, Springer Verlag.
- [18] P.Molitor: "Free Net Algebras in VLSI-Theory", TR10/1987, SFB124, Saarbrücken 1987.
- [19] P.Molitor, R.Kolla: "A note on hierarchical layer-assignment", TR08/1986, SFB124, Saarbrücken 1986.
- [20] R.Y.Pinter: "Optimal layer assignment for interconnect", ICCC 1982, S.398-401.
- [21] M.Sheeran: " $\mu$ FP - An algebraic VLSI design language", PH.D. Thesis, University of Oxford, England 1984.

