Automatisierung des Entwurfs vollständig testbarer Schaltungen

Sybille Hellebrand, Hans-Joachim Wunderlich
Universität Karlsruhe
Institut für Rechnerentwurf und Fehlertoleranz
(Prof. Dr. D. Schmid)
D-7500 Karlsruhe 1, Haid-und-Neu-Straße 7

Kurzfassung

Die Kosten für die Testvorbereitung, Testerzeugung und Testdurchführung wachsen überproportional mit der Komplexität anwendungsspezifischer Schaltungen, und die Teststrategie sollte daher bereits in einer sehr frühen Phase des Schaltungsentwurfs festgelegt und berücksichtig werden. In diesem Artikel werden logische Grundzellen und Algorithmen zur Unterstützung des pseudo-erschöpfenden Tests vorgestellt. Diese Teststrategie hat den Vorteil, daß die äußerst rechenzeitaufwendige Testmustererzeugung entfällt und zugleich eine vollständige Fehlererfassung auf Gatterebene garantiert ist. Die vorgestellten Grundzellen dienen der Zerlegung der Gesamtschaltung in erschöpfend testbare Teile, die präsentierten Algorithmen sollen diese Segmentierungszellen so plazieren, daß der Mehraufwand an Silizium gering bleibt. Hierzu wurden Varianten sogenannter "Hill-Climbing" und "Simulated-Annealing"-Verfahren entwickelt.

1. Einleitung

Komplexe digitale Schaltungen enthalten zumeist eine umfangreiche Zusatzausstattung, die für die wirtschaftliche Durchführung des Produktionstests notwendig ist. Diesem Test muß jeder gefertigte Chip unterworfen werden, da besonders bei anwendungsspezifischen Schaltungen mit einer nur geringen Ausbeute gerechnet werden muß und hier Fehlertoleranzverfahren nicht geeignet sind. Daher haben die Kosten dieses Tests großen Einfluß auf die Gesamtkosten der Schaltung.

Die Kosten für die Testdurchführung können durch den Entwurf selbsttestbarer Schaltungen drastisch reduziert werden. Ein erster Schritt, auch die Kosten für die Testmusterbestimmung zu vermindern, war die Integration eines Prüfpfades [AnWi73], um im Testbetrieb alle Speicherelemente unmittelbar zu setzen und zu beobachten. In diesem Falle sind nur für das verbleibende Schaltnetz Testmuster zu bestimmen, aber auch dann ist die Testerzeugung höchst rechenaufwendig. Sie kann durch den Einsatz universeller Testmengen, durch den Zufallstest oder durch den pseudo-erschöpfenden Test umgangen werden. Universelle Testmengen erfordern oft großen Mehraufwand beim Entwurf. Beim Zufallstest muß für die konkrete Schaltungsstruktur die Zahl der Muster bestimmt werden, auf die die Schaltung korrekt reagieren muß, damit Fehlerfreiheit angenommen werden kann (vergl. [Wu87a]). Diese Zahl kann durch die Verwendung ungleichverteilter Zufallsmuster reduziert werden [Wu87b]. Die Integration entsprechender Zusatzhardware auf dem Chip ermöglicht die Testdurchführung als Selbsttest [KOEN79], [Wu87c]. Mit dem Zufallstest wird jedoch zumeist keine vollständige Fehlererfassung garantiert, so daß bei sehr hohen Anforderungen an die Produktqualität der Chips andere Teststrategien notwendig werden.

Eine vollständige Fehlererfassung garantiert der erschöpfende Test, bei dem alle möglichen Eingangsbelegungen aufgezählt werden, wobei jedoch die Testlänge exponentiell mit der Zahl der primären Eingänge wächst. Dieses Problem kann in vielen Fällen durch den pseudo-erschöpfenden Test [McBo81] gelöst werden. Hier wird für jeden Ausgang o des Schaltnetzes das minimale Teilschaltnetz K_o bestimmt, das alle Knoten enthält, von denen o abhängt, und das dann durch vollständige Aufzählung aller Muster-

kombinationen erschöpfend getestet wird. Die Teilschaltnetze K_o heißen die "Abhängigkeitskegel" der Ausgänge o (vgl. Bild 1).

Wie beim erschöpfenden Test entfällt beim pseudo-erschöpfenden Test der Aufwand zur Testmusterbestimmung, die Zahl t der benötigten Testmuster ist ferner zumeist erheblich reduziert. Sie kann durch die Ungleichung $2^{\ell} \leq t \leq m \cdot 2^{\ell}$ abgeschätzt werden, wobei ℓ die maximale Anzahl an Eingängen der einzelnen Kegel und m die Anzahl der primären Ausgänge bezeichnet. Dabei hängt t nicht nur von der Zahl der primären Ausgänge, sondern auch von der Auswahl geeigneter Kompaktierungstechniken ab [McCl84], [Aker85].

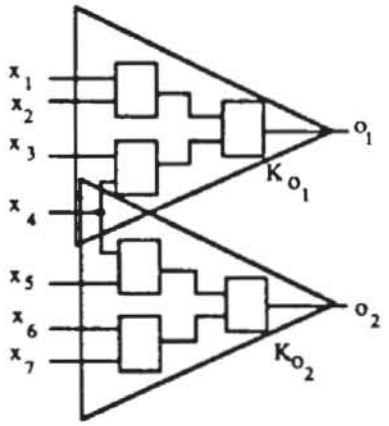


Bild 1: Abhängigkeitskegel

Auch der pseudo-erschöpfende Test garantiert eine hohe Fehlererfassung, innerhalb der einzelnen Kegel werden alle kombinatorischen Fehlfunktionen erfaßt. Eine besondere Behandlung erfordern jedoch Fehler, die ein sequentielles Verhalten hervorrufen können, wie zum Beispiel unterbrochene Leitungen in CMOS-Schaltungen. Mit geeigneter Testhardware zur Erzeugung von Musterfolgen können auch alle einfachen stuck-open-Fehler innerhalb der Kegel erkannt werden [WuHe88]. Für die Gesamtschaltung bedeutet dies die Erkennung von entsprechenden Mehrfachfehlern. Durch Verwendung spezieller linear rückgekoppelter Schieberegister [WaMc86], deren Rückkopplungsfunktionen mit Mitteln der Kodierungstheorie bestimmt werden können, läßt sich der pseudo-erschöpfende Test eines Schaltnetzes auch als Selbsttest durchführen.

Der pseudo-erschöpfende Test eines Schaltnetzes ist sinnvoll, wenn die Anzahl der Eingänge der einzelnen Kegel ein gewisses Limit ℓ nicht überschreitet. Denn während in der oben erwähnten Ungleichung die Zahl der Kegel und damit die Schaltungsgröße höchstens linear eingehen, wächst die Testmenge exponentiell mit ℓ. Um bei einem Hochgeschwindigkeitstest die Durchführungszeit im Sekundenbereich zu halten, sollte $\ell \approx 20$ nicht wesentlich überschritten werden. Beispiel einer pseudo-erschöpfend testbaren Schaltung ist der Paritätsgenerator TI SN54/74LS630 mit 23 Eingängen und 12 Ausgängen. Jeder Ausgang hängt von 10 Eingängen ab, je zwei Abhängigkeitskegel haben dieselben primären Eingänge und können somit gleichzeitig getestet werden. Ein pseudo-erschöpfender Test kann mit $6 \cdot 2^{10} = 6144$ Mustern durchgeführt werden, während für einen erschöpfenden Test $2^{23} \approx 8{,}39 \cdot 10^6$ Muster benötigt würden. Bei Einsatz der erwähnten Kompaktierungstechniken kann die Musterzahl für den pseudo-erschöpfenden Test auf 1024 reduziert werden [McCl84].

Bei beliebigen Schaltnetzen ist nicht gewährleistet, daß die Ausgänge von einer ausreichend kleinen Teilmenge der Eingänge abhängen. Die Vorteile des pseudo-erschöpfenden Tests können jedoch für beliebige Schaltungen genutzt werden, wenn diese zu Testzwecken segmentiert werden [McBo81]. Dabei werden durch Segmentierungszellen im Testbetrieb weitere Knoten innerhalb der Schaltung als primäre Ein- und Ausgänge zugänglich gemacht. Dies sei am Beispiel der kleinen Schaltung aus Bild 2 erläutert. Ohne die eingezeichneten Segmentierungszellen besitzt Knoten 13 einen Abhängigkeitskegel mit 6

Eingängen. Mit den Segmentierungszellen werden die Knoten 7 und 12 zu primären Ausgängen und k_1 und k_2 zu neuen Eingängen. Der Kegel von Knoten 7 besitzt die Eingänge {1, 2, 3}, Knoten 12 besitzt $\{k_1, 4, 5, 6\}$ als Eingänge und Knoten 13 hängt von $\{1, 2, 3, 4, k_2\}$ ab.

Um die Vorteile des pseudo-erschöpfenden Tests zu erhalten, müssen an die Segmentierungszellen folgende Anforderungen gestellt werden:

1) Auch sämtliche Fehler in der zusätzlichen Testausstattung müssen während des pseudo-erschöpfenden Tests erkannt werden.
2) Der zusätzliche Flächenbedarf für die Segmentierungszellen soll möglichst gering sein.
3) Das Systemverhalten sollte durch die Segmentierungszellen möglichst wenig beeinträchtigt werden.

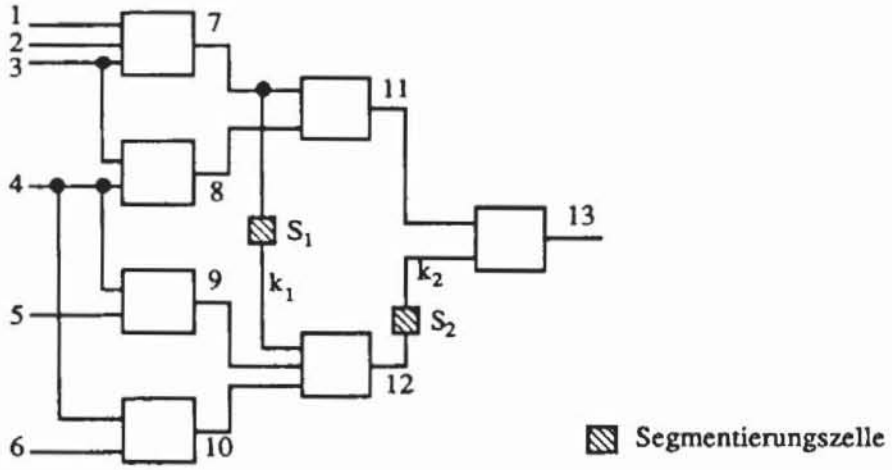


Bild 2: Schaltnetz mit Segmentierungszellen

Die bekannten Segmentierungstechniken erfüllen manche dieser Anforderungen nur unzureichend und andere gar nicht. Es werden daher im folgenden neue Segmentierungszellen vorgestellt, die auf dem bekannten LSSD-Prinzip beruhen. Diese Zellen können in den Logikentwurf automatisch eingefügt werden, so daß garantiert ist, daß die korrekte Systemfunktion der jetzt pseudo-erschöpfend testbaren Schaltung beibehalten wird. Um die dadurch entstehenden Zusatzkosten an Chipfläche zu reduzieren, sollten möglichst wenig Segmentierungszellen benötigt werden. In Abschnitt 3 werden die bekannten Verfahren zur Plazierung solcher Zellen diskutiert, und in Abschnitt 4 wird ein effizienter "Hill-Climbing" Algorithmus zur Segmentierung vorgestellt, der zu deutlich günstigeren Lösungen gelangt. Jedoch besteht bei diesem Verfahren die Gefahr suboptimaler Lösungen, die bei erhöhtem Rechenaufwand durch probabilistische Methoden reduziert werden kann. Diese werden in Abschnitt 5 dargestellt. Insgesamt ergibt sich damit ein Programmsystem, das einen LSSD-basierten Entwurf mit möglichst geringen Kosten so ergänzt, daß die Gesamtschaltung auf alle kombinatorischen Funktionsfehler vollständig testbar ist und die Korrektheit der Funktion dabei weiter gewährleistet bleibt.

2. Die logischen Grundzellen für den pseudo-erschöpfenden Test

Beim pseudo-erschöpfenden Test werden im wesentlichen für zwei Grundfunktionen Zellen benötigt. Zum einen müssen die Speicherzellen prüfpfadfähig sein, so daß nur das verbleibende Schaltnetz getestet werden muß, und zum anderen muß die Schaltung im Testbetrieb segmentierbar sein. Die klassischen Verfahren hierfür werden in der englischsprachigen Literatur "Level-sensitive Scan-Design" und "Multiplexer-Partitioning" genannt. Diese werden im nächsten Abschnitt skizziert und bewertet. Darauf folgend wird eine neue Architektur vorgestellt, die einige signifikante Nachteile der klassischen Segmentierungsmethoden vermeidet.

2.1. Klassische Partitionierungstechniken

Die zahlreichen klassischen Verfahren des Scan Design finden sich bereits in Übersichtsartikeln [McC184] und Lehrbüchern [Fuji86], [McC186], so daß wir uns im folgenden nur auf die Prinzipien des LSSD beschränken, die für das Verständnis der später vorgestellten Grundzellen wichtig sind. Das wesentliche Bauelement hierfür ist das sogenannte Schieberegister-Latch, dessen Realisierung mit Transfergattern Bild 3 zeigt.

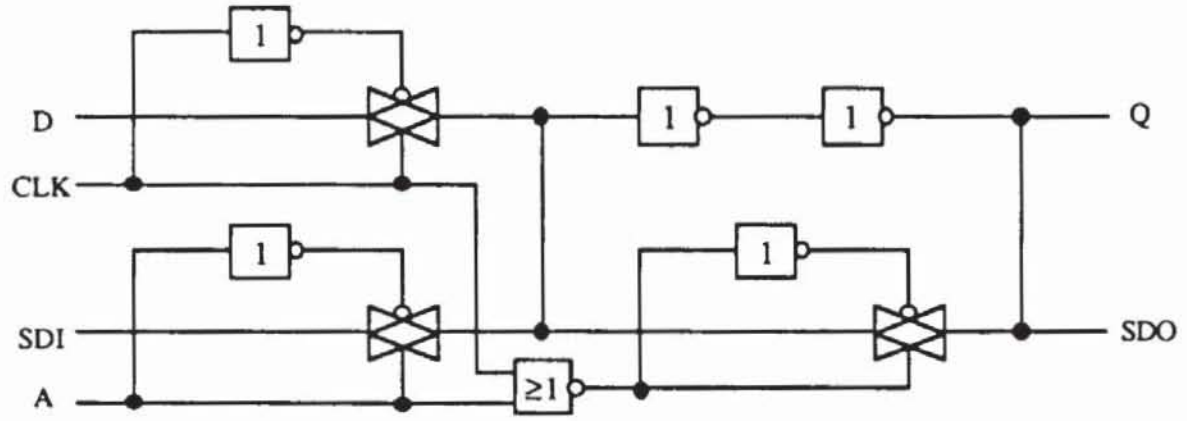


Bild 3: Latch mit Schiebedateneingang

Ist der Systemtakt CLK aktiv, wird an Q der Wert von D geladen, bei aktivem Testtakt A der Schiebedateneingang SDI. CLK und A dürfen sich nicht überlappen. Schaltet man hinter ein solches Latch ein weiteres ohne Schiebedateneingang, das mit CLK $\vee$ A getaktet wird, erhält man ein Master-Slave Flip-Flop. Will man im Systembetrieb nur die Latchfunktion nutzen, so fügt man ein Latch mit einem weiteren Schiebetakteingang B an (Bild 4).

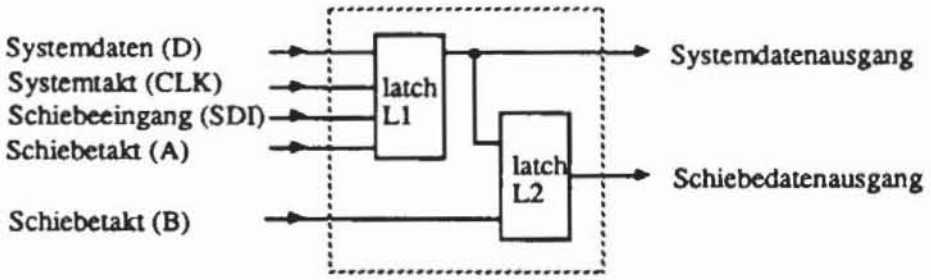


Bild 4: Pegelgesteuertes Schieberegister-Latch (SRL)

Falls im Systembetrieb Latches verwendet werden, muß man mit besonderen Entwurfsregeln garantieren, daß Signalwettläufe und das individuelle Zeitverhalten der verwendeten Zellen die korrekte Funktion nicht stören. Dazu werden neben den beiden Schiebetakten A und B noch zwei Systemtakte CLK(1), CLK(2) benötigt. Die Schaltung muß in zwei disjunkte Schaltnetze N(1) und N(2) und zwei Mengen von Latches zerlegt werden (Bild 5), so daß gelten:

1) Die mit CLK(1) getakteten Latches erhalten ihre Daten aus N(1).
2) Die mit CLK(2) getakteten Latches erhalten ihre Daten aus N(2).
3) Das Netz N(1) erhält keine Daten von Latches, die mit CLK(1) getaktet werden.
4) Das Netz N(2) erhält keine Daten von Latches, die mit CLK(2) getaktet werden.

Es fällt auf, daß die Latches L2 nur zu Testzwecken verwirklicht sind und keine Systemfunktion haben. Dieser Mehraufwand entfällt bei einer L1/L2*-Architektur, bei der auch die L2*-Latches einen Dateneingang und einen Eingang für den zugehörigen Systemtakt erhalten. Ein L2*-Latch ist also mit der

in Bild 3 gezeigten Schaltung identisch. Bild 6 zeigt, wie diese Zellen eingesetzt werden. Offensichtlich müssen bei Einsatz dieser Zellen noch folgende Regeln zusätzlich erfüllt werden:

5) CLK(1) und Schiebetakt A steuern die L1-Latches.
6) CLK(2) und Schiebetakt B steuern die L2-Latches.

Die Suche nach einer Partitionierung der Schaltung, die Regel 1) bis Regel 6) erfüllt, entspricht einem einfachen graphentheoretischen Problem. Dabei bilden die benötigten Zustandsvariablen der Latches die Knoten V, und von einer Variablen a ist zur Variablen b eine Kante gerichtet, wenn b von a abhängt. Bild 7 verdeutlicht diese Umformung.

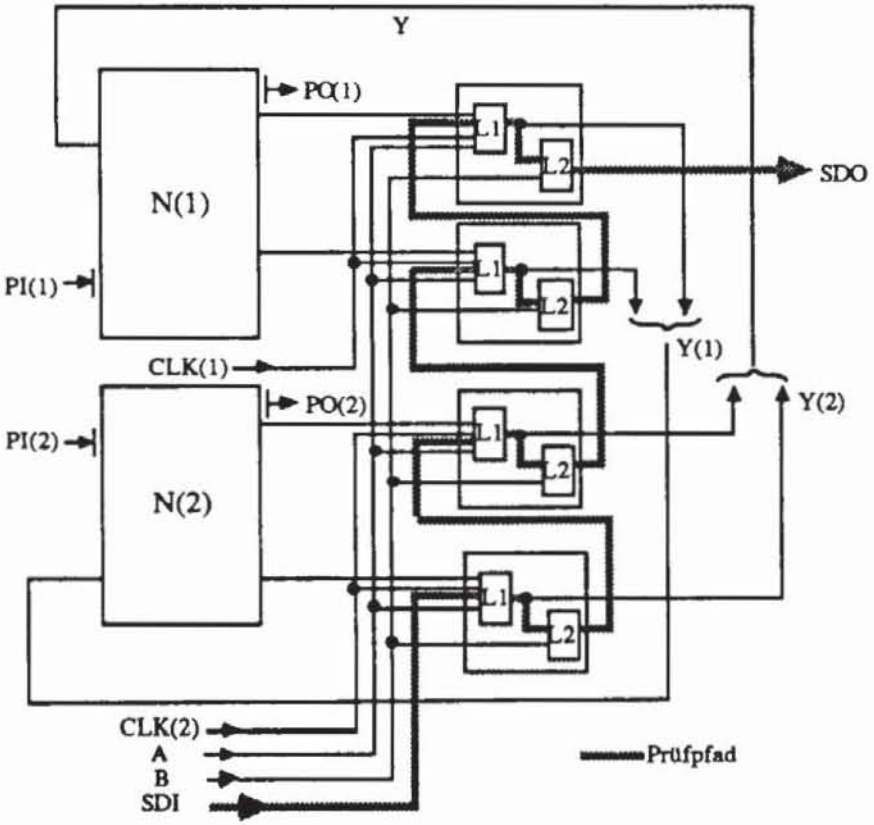


Bild 5: Einfach-Latch Konfiguration

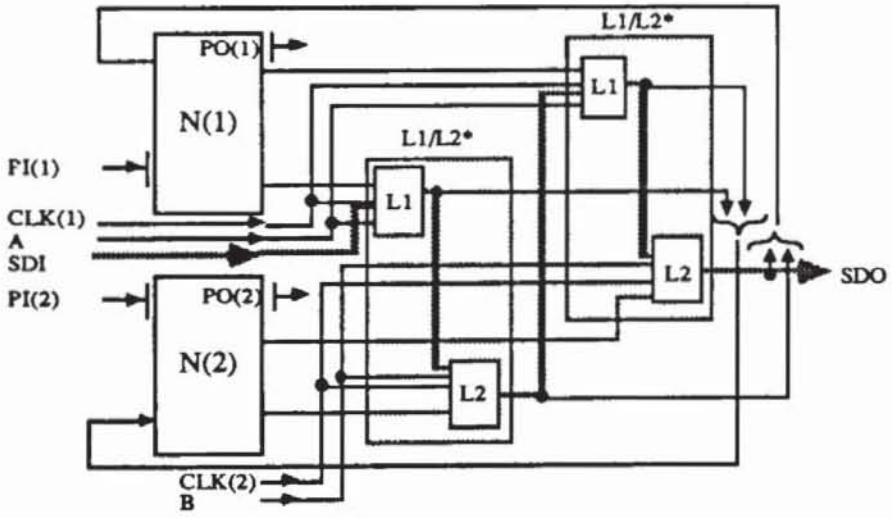


Bild 6: LSSD mit L1/L2*-Zellen

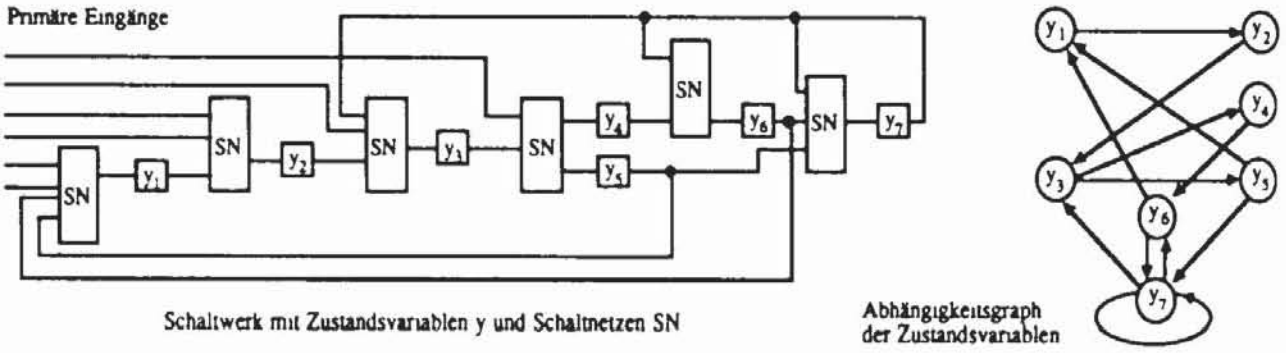


Bild 7: Schaltwerk und zugehöriger Abhängigkeitsgraph der Zustandsvariablen

Nun ist eine Partitionierung der Knoten in zwei gleich große Mengen V_1 und V_2 gesucht, so daß alle Kanten von V_1 zu Knoten aus V_2 oder umgekehrt gehen, aber innerhalb von V_1 und V_2 keine Kanten verlaufen. Leider ist eine solche Partitionierung nicht immer möglich. In diesem Fall müssen zusätzliche Latches eingeführt werden, wobei manche L1- oder L2*-Latches durch Doppel-Latches nach Bild 4 oder durch Master-Slave Flipflops ersetzt werden. Dies vergrößert jedoch den Flächenbedarf der Schaltung und kann zu Leistungseinbußen führen. Aus diesem Grunde sollte eine Schaltungssegmentierung zur Unterstützung des pseudo-erschöpfenden Tests keine zusätzlichen Abhängigkeiten zwischen den Zustandsvariablen einführen.

Als Segmentierungshardware wurden Multiplexer nach Bild 8 vorgeschlagen [McBo81]. Der Einfachheit wegen beschränkt sich dieses Beispiel auf die vollständige Partitionierung in zwei Teile. Im Systembetrieb schalten sämtliche Multiplexer den 1-Eingang durch und die Teilschaltnetze G1 und G2 sind verbunden. Soll G2 getestet werden, schalten M1 und M3 den 0-Eingang und M2 und M4 den 1-Eingang durch. Entsprechend kann der Test für G1 durchgeführt werden.

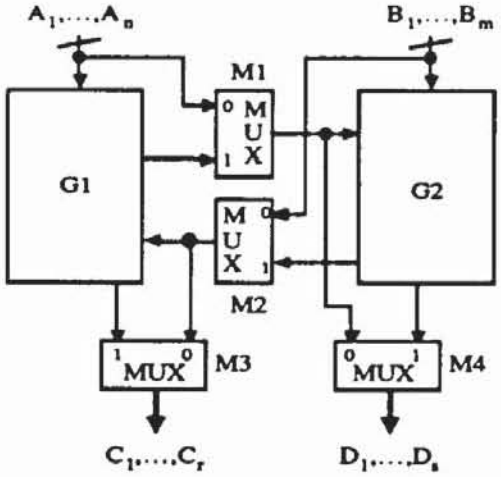


Bild 8: Struktur einer Multiplexersegmentierung

Dieser Vorschlag hat jedoch auch gewichtige Nachteile:

1) Es wird ein sehr hoher zusätzlicher Verdrahtungsaufwand erforderlich, da zu einem Multiplexer eine Signalleitung von einem Schaltnetzeingang hingeführt und eine andere zu einem Schaltnetzausgang weggeführt werden muß.

2) An jedem aufzutrennenden Schaltungsknoten sind zwei Multiplexer nötig, je einer zum Setzen und zum Beobachten. Beide liegen hintereinander im Datenpfad und beeinträchtigen das Schaltungsverhalten.

3) Insbesondere bei einer Partitionierung in eine größere Zahl von Schaltungen ist eine umfangreiche Steuerung für die Multiplexer erforderlich, die nicht pseudo-erschöpfend getestet werden kann.

4) Ob die korrekte Systemfunktion der Multiplexer M1 und M2 geprüft wird, ist layoutabhängig.

5) Die Eingänge A, B und die Ausgänge C, D sind im allgemeinen Latches, zwischen denen durch die Multiplexer neue Abhängigkeiten eingeführt werden und deren LSSD-gemäße Aufteilung gestört wird.

Im nächsten Abschnitt beschreiben wir eine Segmentierungstechnik, die diese Nachteile umgeht.

2.2. Entwurf der Segmentierungszellen

Bereits die traditionelle LSSD-Technik verfolgt das Ziel, Knoten in der Schaltung direkt zugänglich zu machen. Anstatt mit der Multiplexer-Technik dasselbe Ziel mit einem völlig neuen Verfahren anzustreben, versuchen wir, die LSSD-Technik so zu erweitern, daß auch Knoten im Schaltnetz direkt zugänglich sind, ohne daß dort im Systembetrieb Zustände gespeichert werden. Es sollen also die Segmentierungszellen S1 und S2 aus Bild 2 in den bereits vorhandenen Prüfpfad integriert werden. Eine entsprechende Zelle zeigt Bild 9.

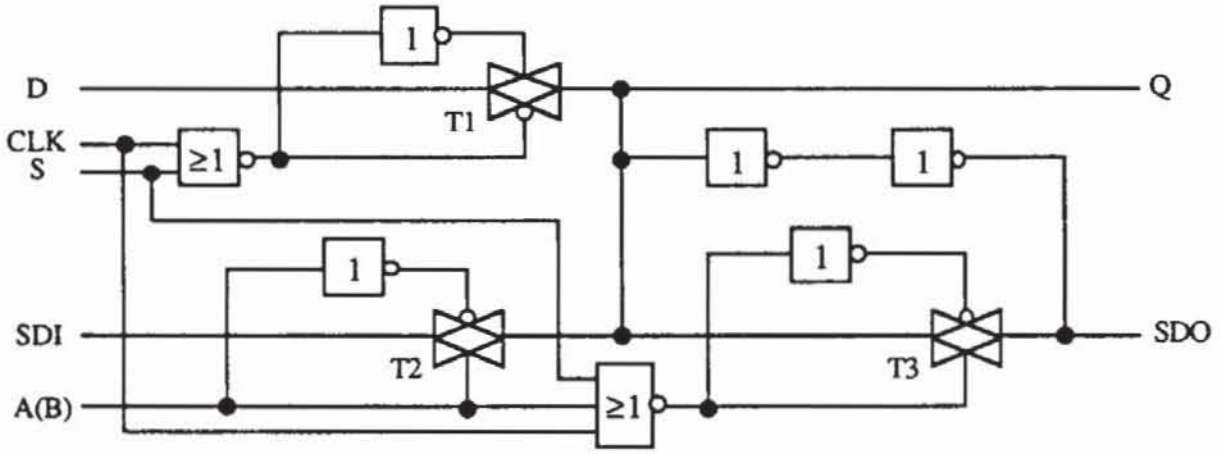


Bild 9: Segmentierungszelle

Diese Zelle entspricht im wesentlichen einem L1-Latch, und jeweils zwei von ihnen ergeben zusammen ein L1/L2*-Latch. Es wird in entsprechender Weise in den Prüfpfad geschaltet.

Das Eingangssignal S legt fest, ob die Zelle im Systembetrieb den Dateneingang D direkt nach Q durchschaltet (S=1), oder ob sie als Latch arbeitend an Q den vorhergehenden Wert speichert (S=0, T1 sperrt und T3 leitet). Für S=1 bleibt auch bei eingebauten Segmentierungszellen die Funktion des Schaltnetzes unverändert. Ist jedoch S=0, so konfiguriert sich die Zelle als normales Prüfpfad-Latch, dessen Inhalt genau wie der aller anderen Speicherelemente direkt zugänglich ist.

Die Zelle erfüllt also die geforderte Segmentierungsfunktion wie die klassische Multiplexer-Lösung. Die geschilderten Nachteile treten jedoch nicht auf:

1) Verdrahtung:
Die Segmentierungszellen werden Teil des Prüfpfads, in dem die Zellen in beliebiger Reihenfolge angeordnet sein können. Es bestehen daher bei der Verdrahtung weit höhere Freiheitsgrade als bei der Verschaltung der Datenleitungen nach Bild 8.

2) Verzögerung:
Die Segmentierungszelle fügt nur das Transfergatter T1 zusätzlich in den Datenpfad ein, während beim klassischen Verfahren zwei Multiplexer benötigt werden.

3) Steuerung:
Nur das Signal S wird zu Segmentierung auf 0 gesetzt. Im Gegensatz zur Multiplexer-Lösung werden alle Zellen mit demselben Signal gesteuert.

4) Fehlererfassung:
Jeder Teil des Datenpfades von D nach Q im Systembetrieb wird auch im Testmodus durchlaufen. Ein erschöpfender Test der Gesamtschaltung testet auch die Segmentierungszelle erschöpfend.

5) Eignung für LSSD:
Wie aus Bild 8 deutlich wird, führt die Multiplexer-Lösung zahlreiche neue Abhängigkeiten zwischen den vorhandenen Zustandsvariablen ein und kann so eine Partitionierung der Latches entsprechend der Regeln 1) bis 6) ungültig machen. Dagegen treten beim Einbau der Segmentierungszellen keine neuen Abhängigkeiten zwischen den vorhandenen Latches auf, sondern nur zwischen den Segmentierungszellen selbst. Bestehende Partitionierungen bleiben somit gültig.

Aber dennoch entstehen durch den Einbau dieser Zellen Zusatzkosten, und ihre Anzahl sollte daher minimiert werden. Entsprechende Algorithmen werden in den nächsten Abschnitten vorgestellt.

3. Schaltungssegmentierung

Mit den beschriebenen Segmentierungszellen kann der Aufwand an Zusatzhardware direkt durch die Zahl der benötigten Latches gemessen werden. Damit ist folgendes Problem zu lösen:

Problem OSS(Optimale Schaltungssegmentierung): Gegeben sei ein Schaltnetz S und eine natürliche Zahl $\ell \in N$. Transformiere S durch den Einbau von Segmentierungszellen in ein Schaltnetz S', bei dem jeder Ausgang von höchstens ℓ Eingängen abhängt. Dabei soll die Anzahl der Segmentierungszellen minimal sein.

Für den sukzessiven Test der Abhängigkeitskegel der transformierten Schaltung werden höchstens $m \cdot 2^{\ell}$ Testmuster benötigt, wenn m die Anzahl der primären Ausgänge bezeichnet. Die Testdauer für das transformierte Schaltnetz hängt also im wesentlichen von der Wahl von ℓ ab.

OSS läßt sich als graphentheoretisches Problem auffassen, wobei der Graph G = (V, E) als Knotenmenge V alle primären Eingänge und alle Gatterausgänge besitzt. Zwei Knoten bilden eine gerichtete Kante (v, w) $\in$ E, wenn Knoten v an einem Eingang und w am Ausgang desselben Gatters angeschlossen sind. Damit bilden die primären Eingänge von S die Menge $\mathfrak{Q}(G)$ der Quellknoten von G. Die Knoten von G seien gemäß dem Signalfluß durchnumeriert. Der Einbau von Segmentierungszellen in S kann als "Schneiden" von Knoten und Kanten in G modelliert werden. Bild 10 zeigt den Graphen, der zu dem Schaltnetz aus Bild 2 korrespondiert, mitsamt den Schnitten.

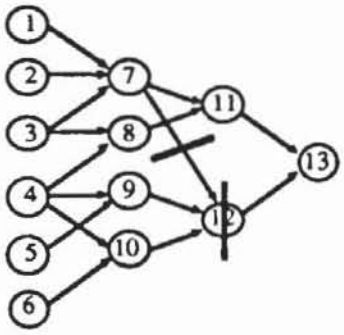


Bild 10: Schaltnetzgraph zu Bild 2

Wir ordnen jedem Knoten v $\in$ V die Anzahl der Quellknoten, von denen ein Pfad zu v führt, als *Abhängigkeitswert* a(v) zu. Das Schneiden eines Knotens v $\in$ V entspricht dem Einbau einer Segmentierungszelle in einen Fanout-Stamm. Dabei wird v durch zwei Knoten v' und v" ersetzt. Der Knoten v' hat keine Nachfolger und seine Vorgänger sind die Vorgänger von v, der Knoten v" ist ein zusätzlicher Quellknoten, dessen Nachfolger die Nachfolger von v sind (vgl. Bild 11).

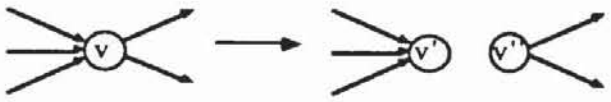


Bild 11: Schnitt eines Knotens

Das Schneiden einer Kante entspricht dem Auftrennen eines Fanout-Zweigs durch eine Segmentierungszelle und kann auf entsprechende Weise definiert werden. Falls das Schaltnetz an jedem Fanout-Zweig einen Treiberbaustein besitzt, sind nur Knoten zu schneiden. Da jeder Schaltnetzgraph G für Rechenzwecke entsprechend ergänzt werden kann, beschränken wir uns aus Gründen der Übersichtlichkeit der Darstellung auf das Schneiden von Knoten. Damit läßt sich das Problem OSS wie folgt formulieren:

> Gegeben sei ein gerichteter, zyklenfreier Graph G = (V, E), $\ell \in N$. Transformiere G durch Schneiden von Knoten in einen Graph G' = (V', E'), so daß $a(v) \leq \ell$ für alle $v \in V'$ gilt. Die Anzahl der Schnitte soll minimal sein.

Eine Transformation von G wird durch die Angabe der Menge der zu schneidenden Knoten vollständig beschrieben und OSS läßt sich durch vollständige Aufzählung aller $2^{|V|}$ möglichen Transformationen von G exakt lösen. Um diesen exponentiellen Aufwand zu vermeiden, haben Roberts und Lala ein heuristisches Vorgehen vorgeschlagen [RoLa84]. Ihre Heuristik garantiert aber nicht, daß die Abhängigkeitswerte sämtlicher Knoten aus V' unter der Konstanten ℓ liegen, sondern es soll nur der größte Abhängigkeitswert $a^* := \max\{a(v) | v \in V'\}$ für einen Knoten in G' möglichst wenig von ℓ abweichen. Ihr Verfahren wurde an das Problem OSS angepaßt und zusätzlich zu den in Abschnitt 4 und 5 entwickelten Methoden als Vergleichsalgorithmus implementiert, dessen Ergebnisse Tabelle 1 zeigt.

Patashnik modifiziert das Problem OSS leicht, indem er anstelle einer minimalen Zahl von Schnitten eine minimale Zahl von Testmustern für den pseudo-erschöpfenden Test der resultierenden Schaltung fordert (OSS'). In [Pata83] zeigt er, daß OSS' NP-vollständig ist. Daher wurden für seine Behandlung als Heuristiken "Iterative-Improvement" [Arch85] und "Simulated-Annealing" [McSh87] vorgeschlagen und auch unser Ziel war die Entwicklung effizienter Heuristiken zur Bestimmung suboptimaler Lösungen für OSS.

4. Ein Hill-Climbing-Verfahren zur Lösung von OSS

Das Problem OSS läßt sich als ein kombinatorisches Optimierungsproblem folgender Art auffassen:

Kombinatorische Optimierung: Gegeben sei eine Menge $\mathfrak{Z}$ von Zuständen, eine Teilmenge $\mathfrak{Z}^* \subset \mathfrak{Z}$ zulässiger Zustände und eine Kostenfunktion $k: \mathfrak{Z} \rightarrow R$ auf $\mathfrak{Z}$ Gesucht ist ein zulässiger Zustand $Z \in \mathfrak{Z}^*$ mit minimalen Kosten, d. h. $k(Z) = \min\{k(X) | X \in \mathfrak{Z}^*\}$.

Bei derartigen Optimierungsproblemen haben sich sogenannte "Hill-Climbing"-Verfahren bewährt [Rich83]. Hierbei wird ein Suchbaum konstruiert, dessen Knoten den Zuständen Z entsprechen. Die Wurzel ist ein Anfangszustand $Z_0 \in \mathfrak{Z}$, und für jeden Knoten werden alle unmittelbaren Nachfolger gemäß einer Erzeugungsregel als Folgezustände erzeugt. Mit einer heuristischen Funktion $h: \mathfrak{Z} \rightarrow R$ wird entschieden, zu welchem der unmittelbaren Nachfolger weiterverzweigt wird. Das Verfahren wird fortgesetzt, bis ein zulässiger Zustand erzeugt worden ist.

Um das Problem OSS entsprechend behandeln zu können, sind die Menge der Zustände $\mathfrak{Z}$, der zulässigen Zustände $\mathfrak{Z}^*$, die Kostenfunktion k, die Heuristik h, die Erzeugungsregel und damit auch der Suchbaum festzulegen.

Menge der Zustände $\mathfrak{Z}$ und $\mathfrak{Z}^*$: Wie bereits erläutert wurde, wird eine Transformation des Graphen G = (V, E) durch Angabe der Knoten, die geschnitten werden sollen, eindeutig beschrieben. Die Potenzmenge $\mathcal{P}(V)$ der Knotenmenge V wird deshalb als Zustandsmenge $\mathfrak{Z}$ gewählt. Der Graph,

der aus G durch Schneiden der Knoten in einer Teilmenge $W \subset V$ entsteht, wird im folgenden stets mit $G_W = (V_W, E_W)$ bezeichnet. Zulässig sind solche Teilmengen $W \subset V$, denen eine Transformation in einen Graphen G_W entspricht, so daß $a(v) \leq \ell$ für alle $v \in V_W$ gilt, d. h. $\mathfrak{Z}^* := \{W \subset V \mid \forall v \in V_W: a(v) \leq \ell\}$.

Kostenfunktion k: $\mathfrak{Z} \rightarrow R$: Die Kostenfunkton für OSS spiegelt direkt den Hardware-Mehraufwand wider, für jede Teilmenge $W \subset V$ ist $k(W) := |W|$ die Zahl der einzubauenden Segmentierungszellen.

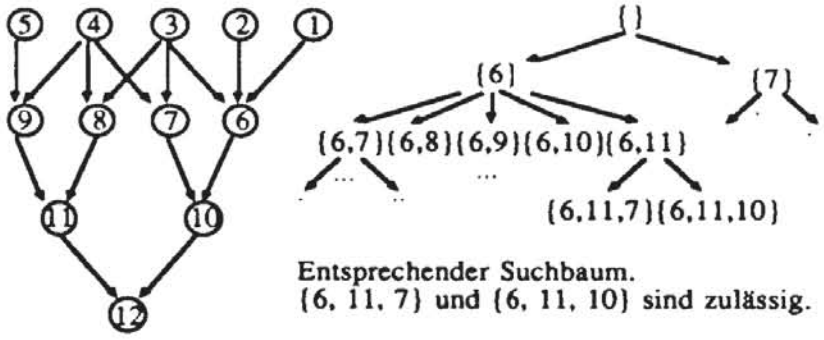
Heuristische Funktion h: $\mathfrak{Z} = \mathcal{P}(V) \rightarrow R$: Zur Bewertung der Zustände wurde folgende heuristische Funktion h gewählt:

$$\begin{aligned} h: \mathcal{P}(V) &\rightarrow R \\ W &\rightarrow \sum_{\substack{v \in V_W \\ a(v) > \ell}} \ln(a(v)) \end{aligned}$$

Gilt $a(v) \leq \ell$ für alle $v \in V_W$, so ist W ein zulässiger Zustand und h wird Null gesetzt. Diese Funktion kann als eine Schätzung der Zahl der Knoten interpretiert werden, die zusätzlich zu W noch geschnitten werden müssen, um einen zulässigen Zustand zu erreichen.

Erzeugungsregel und Suchbaum: Es sei $W \subset V$ eine Menge von Knoten, d. h. $W \in \mathfrak{Z}$. Die unmittelbaren Nachfolger von W werden wie folgt bestimmt: Sei v_{min} der Knoten mit der kleinsten Nummer in G_W, dessen Abhängigkeitswert das Limit ℓ überschreitet. $\mathcal{V}(v_{min})$ sei die Menge der Vorgängerknoten von v_{min} in G_W. $U \subset V$ ist Folgezustand von W, genau dann, wenn $U = W \cup \{v\}$, $v \in \mathcal{V}(v_{min}) \wedge v \notin \mathfrak{Q}(G_W)$ gilt.

Damit haben also nur unzulässige Zustände einen Nachfolger im Suchbaum. Weiter schränkt diese Erzeugungsregel die Zahl der unmittelbaren Nachfolger sehr stark ein. Aber dennoch können mit dieser Erzeugungsregel und $\{\}$ als Anfangszustand im Suchbaum alle kostenminimalen Zustände erzeugt werden. Bild 12 zeigt das Beispiel eines Schaltungsgraphen mit zugehörigem Suchbaum.



Schaltungsgraph G = (V, E)

Entsprechender Suchbaum.
{6, 11, 7} und {6, 11, 10} sind zulässig.

Bild 12: Schaltungsgraph und zugehöriger Segmentierungssuchbaum für $\ell = 3$

Zur Überwindung lokaler Minima der heuristischen Funktion wurde der Hill-Climbing Algorithmus erweitert. Falls nach einer bestimmten Anzahl von Zustandsübergängen keine Verbesserung der heuristischen Funktion erzielt werden kann, wird im Suchbaum um einige Schritte zurückgegangen ("Back-Tracking"). Falls sich der Wert der heuristischen Funktion auch dadurch nicht verbessern läßt, wird eine zweite Erzeugungsregel (Hilfsregel) angewandt.

Hilfsregel: Sei $W \subset V$ eine Menge von Knoten, $v_{min} \in V_W$ der Knoten mit der kleinsten Nummer in G_W, dessen Abhängigkeitswert das Limit ℓ überschreitet. A_{max} bezeichne die Menge der Vorgängerknoten von v_{min} mit maximalem Abhängigkeitswert. $U \subset V$ ist Nachfolger von W genau dann, wenn $U = W \cup \{v\}$, $v \in A_{max}$ gilt.

Der Aufwand des Verfahrens ist im schlimmsten Fall $O(|V|^2)$. Für eine sinnvolle Schaltung beschränkt die gewählte Erzeugungsregel die Zahl der Folgezustände für jeden Knoten des Suchbaums auf höchstens $(2^{2\ell}-1)(\ell-1)$. Für jeden Folgezustand kann die heuristische Funktion mit einem Aufwand von $O(|V|)$ berechnet werden und nach höchstens $|V|$ Schritten ist eine zulässige Lösung erreicht.

Der beschriebene Segmentierungsalgorithmus und ein Vergleichsalgorithmus basierend auf [RoLa84] wurden auf einer SUN 3/50 in PASCAL implementiert. Tabelle 1 zeigt die Ergebnisse für einige Benchmark-Schaltungen [Brgl85].

	$\ell = 16$		$\ell = 20$		$\ell = 24$	
Schaltung	Hill-Climbing	Vergleichs-algorithmus	Hill-Climbing	Vergleichs-algorithmus	Hill-Climbing	Vergleichs-algorithmus
tn.c432	27	56	21	44	17	43
tn.c499	8	20	9	32	7	28
tn.c880	17	34	15	48	10	26
tn.c1355	8	20	9	32	7	28
tn.c1908	22	51	17	53	19	48
tn.c2670	33	108	29	77	28	70
tn.c3570	90	138	68	105	49	94
tn.c5315	62	96	46	82	35	61
tn.c6288	96	115	65	74	36	36
tn.c7552	117	118	87	75	75	57

Tabelle 1: Benötigte Schnitte beim Hill-Climbing Algorithmus und bei einem Vergleichsalgorithmus basierend auf [RoLa84].

5. Ein Simulated-Annealing Verfahren zur Lösung von OSS

Eine Alternative bei der Behandlung kombinatorischer Optimierungsprobleme sind probabilistische Methoden. Das sogenannte Simulated-Annealing Verfahren kommt ursprünglich aus der statistischen Physik und wurde erstmals von Kirkpatrick [Kirk83] auf allgemeine kombinatorische Optimierungsprobleme des Schaltungsentwurfs angewandt.

Auch bei der Simulation des Abkühlungsprozesses von erhitzten Festkörpern (z.B. beim Züchten von Kristallen) ist ein kombinatorisches Optimierungsproblem zu lösen. Ausgangspunkt ist ein physikalisches System, das eine Menge $\mathfrak{Z}$ von Zuständen einnehmen kann, die durch die Position der Teilchen charakterisiert sind. Jedem Zustand $Z \in \mathfrak{Z}$ wird die Energie des Systems $E(Z)$ zugeordnet. Bei konstanter Temperatur θ entspricht die Verteilung der Zustände im thermischen Gleichgewicht der Boltzmann-Verteilung, d. h. ein Zustand Z hat das statistische Gewicht $\exp(-E(Z)/(k_B \cdot \theta))$, wobei k_B die Boltzmann-Konstante bezeichnet. Bei abnehmender Temperatur strebt das System gegen den Zustand mit der niedrigsten Energie (z. B. Kristallgitter).

Bei konstanter Temperatur kann man die Entwicklung des thermischen Gleichgewichts des Systems simulieren, indem von einem Systemzustand Z ausgehend ein Zustand Z' zufällig erzeugt wird. Falls für diesen Zustand $E(Z') < E(Z)$ gilt, wird Z' zum neuen Systemzustand. Für $E(Z') \geq E(Z)$ wird Z' jedoch nur mit der Wahrscheinlichkeit $\exp(-(E(Z') - E(Z))/(k_B \cdot \theta))$ angenommen. Die Verteilung der Zustände im n-ten Simulationsschritt konvergiert für $n \to \infty$ gegen die Boltzmann-Verteilung [Metr53]. Unter "Simulated Annealing" versteht man die Simulation des gesamten Abkühlungsprozesses mit einer Folge $(\theta \to 0)$ solcher "Metropolisalgorithmen".

Bei der Anwendung auf ein beliebiges Optimierungsproblem $(\mathfrak{Z}, k)$ tritt die Kostenfunktion $k: \mathfrak{Z} \to \boldsymbol{R}$

anstelle der Energie E und statt der Temperatur θ wird ein Kontrollparameter c eingeführt. Für konstantes c wird der Übergang von einem Zustand i zu einem Zustand j mit Wahrscheinlichkeit 1 akzeptiert, falls damit eine Kostenverbesserung verbunden ist, und mit Wahrscheinlichkeit exp(-(k(j) - k(i))/c), falls $k(j) \geq k(i)$ gilt.

Es muß jedoch gezeigt werden, daß bei dem jeweiligen kombinatorischen Optimierungsproblem die Folge der Metropolisalgorithmen tatsächlich zu einem globalen Optimum konvergiert. Dazu präzisieren wir den Algorithmus anhand eines mathematischen Modells und stellen im folgenden einige Kriterien auf, um entscheiden zu können, ob für einen fest vorgegebenen Kontrollparameter c ein Metropolisalgorithmus konvergiert und ob für variables c auch eine Folge solcher Algorithmen gegen ein Optimum strebt. Die Folge der Zustandsübergänge für konstantes c läßt sich als homogene Markoffkette $(X_t)_{t\in N}$ über dem endlichen Zustandsraum $\mathfrak{Z}$ mit Übergangsmatrix $P = (p_{ij})$ modellieren [Fell65]. Dabei bezeichne p_{ij} die von t unabhängige bedingte Wahrscheinlichkeit $p(X_t=j \mid X_{t-1}=i)$, daß sich das System zum Zeitpunkt t im Zustand j befindet, unter der Bedingung, daß es zum Zeitpunkt t-1 im Zustand i war. Sei g_{ij} die Wahrscheinlichkeit, daß Zustand j aus Zustand i erzeugt wird und $a_{ij} := \min\{1, \exp(-(k(j)-k(i))/c)\}$ die oben definierte Wahrscheinlichkeit, mit der der Übergang akzeptiert wird. Dann gilt für die Übergangswahrscheinlichkeiten $p_{ij} = g_{ij} \cdot a_{ij}$.

In der statistischen Physik ist die Konvergenz zum thermischen Gleichgewicht garantiert. Diesem Gleichgewicht entspricht in unserem Modell eine "ergodische" Verteilung der Zustände.

Definition: Sei $(X_t)_{t\in N}$ eine homogene Markoffkette über dem endlichen Zustandsraum $\mathfrak{Z}$. Ein Vektor $q \in [0,1]^{|\mathfrak{Z}|}$ heißt ergodische Verteilung von $(X_t)_{t\in N}$, wenn unabhängig von der Anfangsverteilung

$$\forall\, i \in \mathfrak{Z} : \lim_{t\to\infty} p(X_t=i) = q_i$$

gilt.

Eine hinreichende Bedingung für die Existenz einer ergodischen Verteilung gibt der folgende Satz.

Satz (1): Sei $(X_t)_{t\in N}$ eine homogene Markoffkette über dem endlichen Zustandsraum $\mathfrak{Z}$ und Übergangsmatrix P. Ist $(X_t)_{t\in N}$ irreduzibel und aperiodisch, so besitzt $(X_t)_{t\in N}$ eine eindeutig bestimmte ergodische Verteilung $q \in [0,1]^{|\mathfrak{Z}|}$ und diese ist stationär, d. h. es gilt $q = q \cdot P$.

Die Voraussetzungen des Satzes sind für den Simulated-Annealing Algorithmus z. B. erfüllt, wenn die Zustände gleichverteilt erzeugt werden, d.h. wenn

$$\forall\, i,j \in \mathfrak{Z} : \; g_{ij} = \frac{1}{|\mathfrak{Z}|}$$

gilt ([LaAa87]). Die ergodische Verteilung ist dann durch

$$\forall\, i \in \mathfrak{Z} : \; q_i := \frac{\exp\left(-\frac{k(i) - k_{opt}}{c}\right)}{\sum_{j\in\mathfrak{Z}} \exp\left(-\frac{k(j) - k_{opt}}{c}\right)}$$

gegeben, wobei k_{opt} den optimalen Wert der Kostenfunktion bezeichne. Wie man leicht nachrechnet, konvergiert diese Verteilung für $c \to 0$ gegen die Gleichverteilung auf der Menge der optimalen Zustände.

Vor der Implementierung dieses Algorithmusses müssen für ein beliebiges kombinatorisches Optimierungsproblem folgende problemspezifische Parameter festgelegt werden ("Cooling-Schedule"):

- der Anfangswert c_0 des Kontrollparameters,
- eine Regel zur Änderung des Kontrollparameters,
- die Länge der einzelnen Markoffketten,
- ein Stopkriterium.

Der Anfangswert des Kontrollparameters kann z. B. empirisch bestimmt werden. Eine einfache Regel zur Änderung des Kontrollparameters, die auch von [McSh87] verwendet wird, ist $c_k := \alpha \cdot c_{k-1}$, wobei $\alpha<1$ eine Konstante nahe bei 1 sei. Die Länge der Markoffketten kann beispielsweise von der Anzahl der akzeptierten Zustandsübergänge abhängig gemacht werden. Der Algorithmus kann abgebrochen werden, falls eine vorgegebene Anzahl von Schritten erreicht ist oder wenn die Endzustände der Markoffketten konstant bleiben. Weitere Möglichkeiten zum Aufbau eines "Cooling-Schedules" sind in [LaAa87] beschrieben.

Simulated-Annealing in der oben beschriebenen Form wurde von McCluskey und Shperling [McSh87] bereits auf das Problem OSS' angewandt, wobei bei der Bestimmung des Startwerts c_0 und bei der Formulierung des Stopkriteriums versucht wurde, problemspezifische Daten auszunutzen.

Es lassen sich jedoch bei einer noch stärkeren Einbeziehung der Problemstruktur günstigere Ergebnisse erzielen. Dazu wird im folgenden ein alternativer Erzeugungsmechanismus vorgeschlagen, der die Zustände nicht mehr gleichverteilt, sondern zielorientiert erzeugt. Zunächst wird ein geeigneter, möglichst kleiner Zustandsraum $\mathfrak{Z}$ definiert. Anders als beim Hill-Climbing Algorithmus werden als Zustände keine Teilmengen der Knotenmenge V von G = (V, E), sondern bestimmte Tupel von Knoten betrachtet, und zwar sei $Z \in \mathfrak{Z}$ äquivalent dazu, daß folgende vier Bedingungen erfüllt sind:

(i) $Z := (z_1,\ldots,z_n) \in V^n$ für ein n mit $1 \le n \le |V|$,

(ii) $Z(n) := \{z_1,\ldots,z_n\} \subset V$ ist zulässig im Sinne des Hill-Climbing Verfahrens,

(iii) für alle $i < n$ ist $Z(i) := \{z_1,\ldots,z_i\} \subset V$ nicht zulässig im Sinne des Hill-Climbing Verfahrens,

(iv) für alle $i = 1,\ldots,n-1$ gilt

$$z_{i+1} \in \mathcal{V}(v_i),$$

wobei v_i den Knoten mit der kleinsten Nummer in $G_{Z(i)}$ bezeichne, dessen Abhängigkeitswert das Limit ℓ überschreitet.

Das bedeutet, daß $\mathfrak{Z}$ genau die Tupel von Knoten enthält, die mit der Erzeugungsregel des Hill-Climbing Algorithmus erzeugt werden können, und damit auch alle optimalen Lösungen. Die Kosten k(Z) eines Zustands bestimmen sich wie beim Hill-Climbing durch die Anzahl der Knoten, die geschnitten werden müssen (d. h. $k(z_1,\ldots,z_n) := n$).

Für $Z := (z_1,\ldots,z_n) \in \mathfrak{Z}$ läuft der Erzeugungsmechanismus für den Simulated-Annealing Algorithmus folgendermaßen ab:

(i) Wähle ein $i \in \{0,\ldots,n-1\}$ mit Wahrscheinlichkeit 1/n und setze $Z' := (z_1,\ldots,z_i)$ für $i > 0$ und $Z' := \emptyset$ sonst.

(ii) Es sei v_i der Knoten mit der kleinsten Nummer in $G_{Z'(i)}$ mit $a(v_i) \ge \ell$. Wähle $z_{i+1} \in \mathcal{V}(v_i)$ mit Wahrscheinlichkeit $g(z_{i+1})/D_i$. Dabei sei

$$g(z) := \frac{1}{h(\{z_1,\ldots,z_i,z\})+1}$$

mit der heuristischen Funktion h des Hill-Climbing Algorithmus und

$$D_i := \sum_{z \in \mathcal{V}(v_i)} g(z)$$

gewählt.

(iii) Setze $Z' := (z_1,\ldots,z_{i+1})$ und $i := i+1$.

Schritt (ii) und (iii) werden solange wiederholt, bis $Z' \in \mathfrak{Z}$ erfüllt ist.

Es ist noch zu zeigen, daß mit diesen Erzeugungsregeln die Metropolisalgorithmen tatsächlich konvergieren. Dies folgt aber nach Satz 1, dessen Voraussetzungen weiterhin erfüllt bleiben, da mit diesem

Mechanismus alle möglichen Zustandsübergänge mit positiver Wahrscheinlichkeit erzeugt werden können, die erzeugten Markoffketten somit irreduzibel und mit den verwendeten a_{ij} damit auch aperiodisch ([LaAa87]) sind.

Das geschilderte problemspezifische Verfahren hat zwei wesentliche Vorteile:

1) Der Zustandsraum ist deutlich eingeschränkt. Er enthält jetzt nur noch zulässige Zustände, so daß unzulässige Zustände gar nicht in der Kostenfunktion berücksichtigt werden müssen.

2) Die Erzeugungsregel bewirkt, daß ausgehend von einem Zustand ein günstigerer neuer Zustand mit höherer Wahrscheinlichkeit als bei Gleichverteilung erreicht wird. Dadurch werden weniger oft Zustände verworfen und Rechenzeit wird eingespart.

6. Das Gesamtsystem

Obwohl das Simulated-Annealing Verfahren auf dem Hill-Climbing Algorithmus aufbaut, werden beide Verfahren zu Verfügung gestellt. Mit relativ geringem Rechenaufwand erzeugt der Hill-Climbing Algorithmus bereits zufriedenstellende Lösungen, während es mehrere CPU-Stunden kostet, mit dem Simulated-Annealing Verfahren die Gefahr von suboptimalen Lösungen zu reduzieren. Dieser hohe Rechenaufwand scheint dann gerechtfertigt, wenn bei einem konkreten Entwurf die Siliziumfläche aufgrund der Plazierungsverfahren tatsächlich sehr knapp bemessen ist oder wenn aufgrund einer hohen Auflage der Schaltung die Siliziumfläche zu minimieren ist. Ansonsten ist es günstiger, auf die schneller verfügbaren Ergebnisse des Hill-Climbing Verfahrens zurückzugreifen.

Das Gesamtsystem verlangt die Eingabe einer Schaltungsbeschreibung und entfernt daraus zunächst die Latches. Für das verbleibende Schaltnetz kann der Anwender eines der beiden Segmentierungsverfahren wählen. Im Rahmen einer Diplomarbeit [Korn87] wurde ein Algorithmus implementiert, der für das segmentierte Schaltnetz eine pseudo-erschöpfende, kompaktierte Testmenge erzeugt. Für die resultierende Schaltung werden anschließend die Systemlatches und die Segmentierungszellen entsprechend den LSSD-Regeln partitioniert und die Taktleitungen angegeben. Einen Überblick gibt Bild 13.

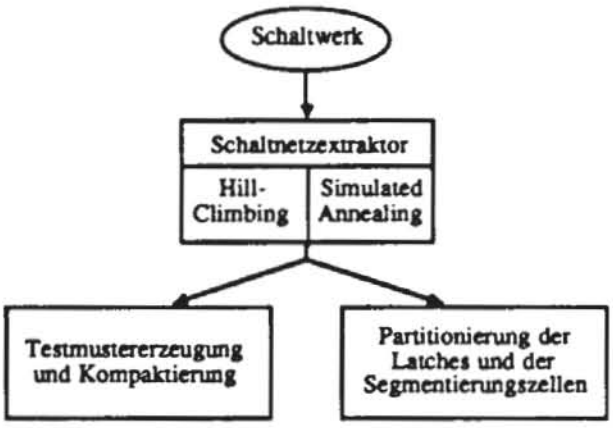


Bild 13: Überblick über das Gesamtsystem

Die Segmentierungszellen selbst wurden im Rahmen einer weiteren Diplomarbeit [Behr88] als CMOS-Komplexgatter entworfen. Sie stehen als selbstentworfene Grundzelle im Siliconcompiler GENESIL zur Verfügung, so daß damit automatisch vollständig testbare Schaltungen entworfen werden können.

7. Literatur

Aker85 Akers,S.B.: On the Use of Linear Sums in Exhaustive Testing; FTCS 85 (IEEE)

AnWi73 Williams, M.J.Y.; Angell, J.B.: Enhancing Testability of Large-Scale Integrated Circuits viaTest Points and Additional Logic; IEEE Trans. Comp., Vol. C-22, Nr. 1, 1973

Arch85 Archambeau, E.C: Network Segmentation for Pseudo-Exhaustive Testing; CRC Technical Report No. 85-10, Stanford, July 1985

Behr88 B. Behrens: Entwurf von Grundzellen für einen Silicon-Compiler zur Unterstützung des pseudo-erschöpfenden Tests, Diplomarbeit am Institut für Rechnerentwurf und Fehlertoleranz, Universität Karlsruhe, 1988

Brgl85 Brglez, F.; et al.: Accelerated ATPG and fault grading via testability analysis; Proceedings ISCAS 85, Kyoto 1985

Fell65 Feller, W.: An Introduction to Probability Theory and Its Applications; 2nd Edition, vol. 1&2, New York, London, Sydney, 1965

Fuji86 Fujiwara, H.: Logic Testing and Design for Testability; The MIT Press, 1985

Kom87 Kornas, F.: Untersuchungen und Implementierung von Algorithmen zur Unterstützung des pseudo erschöpfenden Selbsttests; Diplomarbeit am Institut für Rechnerentwurf und Fehlertoleranz, Universität Karlsruhe, 1987

Kirk83 Kirkpatrick, S., Gelatt, C. D., Jr.; Vecchi, M. P.: Optimization by Simulated Annealing; Science, Vol. 220, No. 4598, May 1983

KOEN79 Koenemann, B. et al.: Built-In Logik Block Observation Techniques; Proc. Test Conference, Cherry Hill 1979, New Jersey

LaAa87 Laarhoven, P. J. M.; Aarts, E. H. L.: Simulated Annealing: Theory and Applications; Dordrecht, Boston, Lancaster, Tokyo 1987

McBo81 McCluskey,E.J., Bozorgui-Nesbat,S.: Design for Autonomous Test; IEEE Trans. on Circuits and Systems, Vol. Cas-28, No. 11, November 1981

McCl84 McCluskey,E.J.: Verification Testing - A Pseudoexhaustive Test Technique; IEEE Trans. on Computers, Vol. c-33, No.6, June 1984

McCl84 McCluskey, E.J.: A Survey of Design for Testability Scan Techniques; in: VLSI Design, Dec. 1984

McCl86 McCluskey, E.J.: Logic Design Principles; Prentice-Hall, 1986

McSh87 McCluskey,E.J.; Shperling, I.: Circuit Segmentation for Pseudo-Exhaustive Testing via Simulated Annealing; CRC Technical Report No. 87-315, Stanford, February 1987

Metr53 Metropolis, N.; Rosenbluth, A.; Rosenbluth, M.; Teller, A.; Teller, E.: Equation of State Calculations by Fast Computing Machines, J. of Chem. Physics, No. 21, 1953, pp. 1087-1092

Pata83 Patashnik,O.: Circuit Segmentation for Pseudo-Exhaustive Testing; CRC Technical Report No. 83-14, Stanford, October 1983

RoLa84 Roberts, M.W., Lala,M.Sc.: An Algorithm for the Partitioning of logic circuits; IEE Proceedings, Vol. 131,Pt.E., No.4,July 1984

WaMc86 Wang,L.T., McCluskey,E.J.: Circuits for Pseudo-Exhaustive Test Pattern Generation; International Test Conference 1986, Paper 1.3.

Wu87a H.-J. Wunderlich: Probabilistische Verfahren für den Test hochintegrierter Schaltungen; Dissertation, Informatik-Fachberichte 140, Springer-Verlag 1987

Wu87b H.-J. Wunderlich: On Computing Optimized Input Probabilites for Random Tests, in: Proc. 24th Design Automation Conference, 1987, Miami Beach

Wu87c H.-J. Wunderlich: Self Test Using Unequiprobable Random Patterns
in: International Symposium on Fault-Tolerant Computing, FTCS-17, 1987, Pittsburgh

WuHe88 H.-J. Wunderlich; S. Hellebrand: Generating Pattern Sequences for the Pseudo-Exhaustive Test of MOS-Circuits;
in: International Symposium on Fault-Tolerant Computing, FTCS-18, 1988, Tokyo