Dr. rer. nat. *K.-A. Zech*, Berlin

Anwendung endlicher Automaten bei der Entwurfsverifizierung

IA 1.7.1.

0. Einleitung

Zu den aufwendigsten Verifizierungsaufgaben beim Entwurf integrierter Systeme gehört das Testen der Layoutgeometrie auf Einhaltung der Entwurfsregeln (design rule checking — DRC). Diese sind eine Funktion der eingesetzten Technologie. Die in [2] eingeführten und auf der relativ skalierungsunabhängigen Längeneinheit λ basierenden Entwurfsregeln für einen einfachen nMOS-Siliziumgate-Depletionload-Prozeß (MC-Regeln) erleichtern sowohl den Entwurf als auch die nachfolgenden Verifizierungsschritte erheblich [9]. Grundsätzlich sind zwei Vorgehensweisen für die Darstellung eines Layouts zu unterscheiden [1] [3] und [4]. Bei der ersten wird ein Entwurf durch Mengen von Rechtecken bzw. Polygonzügen repräsentiert, die jeweils bestimmten Maskenebenen zugeordnet sind. Operationen wie Durchschnitts-, Vereinigungs- oder Differenzbildung sind hier relativ komplex. Die andere Darstellungsweise stützt sich auf ein λ-Raster, bei dem für jede Maskenebene jeder (diskretisierte) Rasterpunkt als zu einer Figur gehörig oder nicht gehörig markiert ist. Jedem Punkt wird also ein Boolescher Vektor zugeordnet, der alle wichtigen Informationen über diesen Punkt enthält: Ist Stelle i des Vektors Eins, so gehört der Bildpunkt in Ebene i zu einer Figur, sonst nicht. Es ist klar, daß die obengenannten mengentheoretischen Operationen bei dieser Darstellung wesentlich einfacher vollziehbar sind als bei der Polygondarstellung. Darüber hinaus legt die Lokalität der MC-Regeln die Verwendung der Rasterdarstellung nahe. Sie ist aber auch für eine Hardwareimplementierung der Entwurfsregelprüfung hervorragend geeignet. In [3] und [4] wurde eine Methode entwickelt, bei der es ausreicht, ein „Fenster" von 4λ mal 4λ streifenweise über den gerasterten Entwurf zu schieben und den Inhalt ($\geqq 16$ bit)-beispielsweise über Hardware — auszuwerten.

Verblüffend einfach ist der Ansatz nach [1]. In Abhängigkeit vom Bildpunktvektor (Pixel) und von Bewertungen je zweier Nachbarpunkte wird eine Bewertung des vorliegenden Pixels (bezüglich einer Entwurfsregel) vorgenommen. Sukzessive erhält so jeder Bildpunkt „seinen" Wert, der Auskunft darüber gibt, ob er oder seine Nachbarn die betrachtete Regel verletzen oder nicht. Die Autoren nennen ihren Ansatz automatentheoretisch, da die Punktbewertung durch einen über die Punktmenge wandernden Automaten vollzogen werden kann. In Abschn. 1. formuliert der Beitrag diesen Ansatz etwas deutlicher, so daß die Rolle des „Automaten" besser sichtbar wird. Diese Betrachtung führt dann im Abschn. 2. direkt zu einer Hardwarestruktur, bei der mehrere Automaten parallel (genauer: systolisch [2]) mehrere Rasterzeilen auf mehrere Entwurfsregeln hin überprüfen.

1. Eustace-Mukhopadhyay-Verfahren [1]

Die einfachsten der geometrischen Entwurfsregeln bestehen darin, daß Figuren gewisse Mindestbreiten haben müssen (Leiterbahnbreiten, Überlappungen, bei Komplementbildung entsprechend auch Abstände). Diese könnten dadurch verifiziert werden, daß jede Zeile (jede Spalte) des Entwurfs abgetastet wird und die Zugehörigkeit bzw. Nichtzugehörigkeit aufeinanderfolgender Punkte zu einer Figur einer Ebene erfaßt und abgezählt wird (Bild 1).

Problematisch ist hierbei die Notwendigkeit, die Daten vertikal nochmals ebenso zu bearbeiten, wobei die Speicheranordnung für diese Daten umzuorganisieren ist oder andere Maßnahmen getroffen werden müssen, um diese effektiv zu bewältigen. Schwieriger ist das Problem, die Diagonalen abzutesten (Bild 2).

Aus diesem Grund werden in jedem Rasterpunkt „Zählungen" nicht nur „von links nach rechts", sondern auch „von unten nach oben" durchgeführt, vorausgesetzt, links bzw. unten liegen die „Zählergebnisse" schon vor. Der gesamte Entwurf (bzw. der zu überprüfende Ausschnitt davon) erhält zu diesem Zweck an seinen Rändern je eine zusätzliche Zeile bzw. Spalte (im Bild 3 außerhalb der gestrichelten Linie). Die Bewertungen der Punkte der linken Spalte (Spalte 0) bzw. der untersten Zeile (Zeile 0) sind mit **0** initialisiert. Der gerasterte Entwurf wird aufgefaßt als Matrix $\boldsymbol{PM} = (P_{ij})$ mit $i = 0, \ldots, r+1$, $j = 0, \ldots, s+1$ und $P_{ij} \in \{0,1\}^m$, wenn r bzw. s die ursprüngliche Zeilen- bzw. Spaltenanzahl ist und m die Anzahl der Maskenebenen angibt. Jedem Pixel P_{ij} wird eine Bewertung z_{ij} zugeordnet, so daß sich die Bewertungsmatrix $\boldsymbol{ZM} = (z_{ij})$ ergibt. Die Berechnung dieser Bewertungsmatrix ist Aufgabe des DRC-Automaten A.

A ist als initialer Automat folgendermaßen definiert [5]:

$$A = [\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{Z}, f, g, z_0],$$

wobei $\boldsymbol{X}$ die Menge der Eingabesignale, $\boldsymbol{Y}$ die der Ausgabesignale, $\boldsymbol{Z}$ die Menge der inneren Zustände und $z_0 \in \boldsymbol{Z}$ der Anfangszustand von A ist sowie f und g die Zustandsüberführungs- bzw. Ausgabefunktion bezeichnen mit:

$$f: \boldsymbol{Z} \times \boldsymbol{X} \to \boldsymbol{Z},$$

$$g: \boldsymbol{Z} \times \boldsymbol{X} \to \boldsymbol{Y}.$$

Im einzelnen gilt folgendes:

$$\boldsymbol{X} = \boldsymbol{P} \times \boldsymbol{Z} \quad \text{mit} \quad \boldsymbol{P} \subseteq \{0,1\}^m,\ m \text{ Maskenzahl};$$

$$\boldsymbol{Z} = \{0, 1, 2, \ldots, n\},\ z_0 = 0;$$

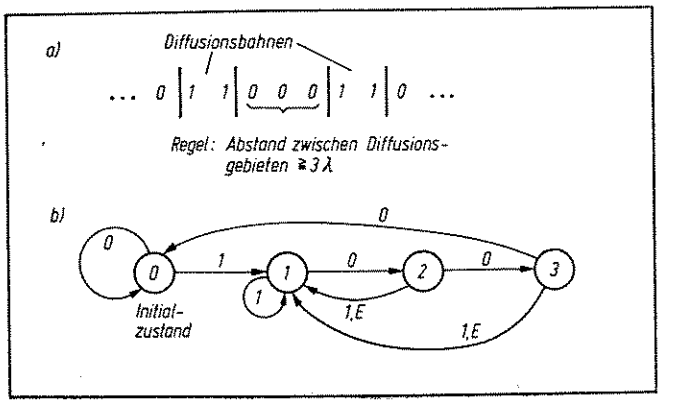
$$\boldsymbol{Y} = \{0, E\},\ E \text{ Fehlermeldung}.$$

Im Zusammenhang mit einem als Matrix $\boldsymbol{PM} = (P_{ij})$ gerasterten Bild, von dem Bild 3 beispielhaft eine Ebene zeigt [1], wird die Arbeitsweise von A wie folgt interpretiert:

A wandert zeilenweise von links nach rechts und von unten nach oben über $\boldsymbol{PM}$. A beginnt im Punkt [1, 0] und nimmt dabei seinen Initialzustand 0 an. Eingabesignal auf diesem Punkt ist Pixel P_{11} sowie der (initialisierte) Wert z_{01} der darunterliegenden Zeile von $\boldsymbol{ZM}$. A berechnet seinen Folgezustand $z' = f[0, [P_{11}, 0]]$ sowie als Ausgabe die Fehlermarkierung $y = g[0, [P_{11}, 0]]$ für den Punkt [1, 1], wandert im nächstem Takt eine Spalte weiter nach rechts (auf den Punkt [1, 1], hinterläßt auf dem alten Platz [1, 0] seinen alten Zustand $z_{10} = 0$ (d. h. trägt die Bewertung z_{10} in die Matrix $\boldsymbol{ZM}$ ein — genaugenommen

Bild 1. Automat zur Überprüfung der Abstände zwischen Diffusionsgebieten, die mindestens 3 λ betragen müssen (aus [1])

a) Zeilenausschnitt eines Rasters der Diffusionsebene; b) Automatengraph (die Pfeile sind Zustandsübergänge bei Eingabe des darangeschriebenen Wertes; die Ausgabe von „E" markiert eine Regelwidrigkeit)



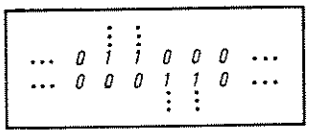


Bild 2. Beispiel dafür, daß zusätzlich zum horizontalen und vertikalen Zählen den Diagonalen besondere Aufmerksamkeit zu widmen ist (aus [1])

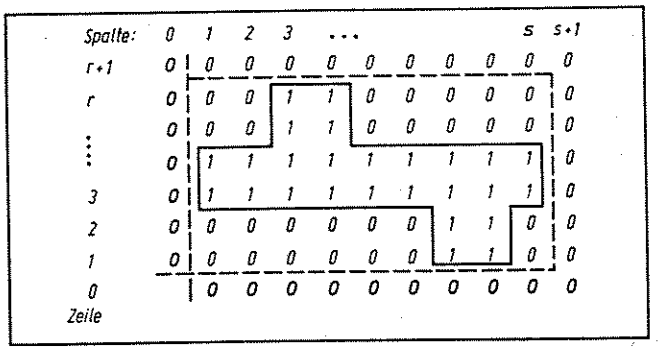


Bild 3. Beispiel aus [1]: Diffusionsebene

Fette Zahlen in Spalte 0 und Zeile 0: Anfangsbewertung auf zusätzlichen Randzeilen bzw. -spalten; sonst: 1 Diffusion, 0 keine Diffusion

eine Ausgabe) und nimmt z' als neuen Zustand an. Dieser Vorgang setzt sich in analoger Weise fort, indem A schrittweise nach rechts wandert. Am Ende der Zeile, nach Berechnung des Wertes z_{1s+1}, kehrt er zurück zur Spalte 0 und beginnt seine Wanderung in Zeile 2 erneut. Allgemein gilt für die Berechnung von $\boldsymbol{ZM}$:

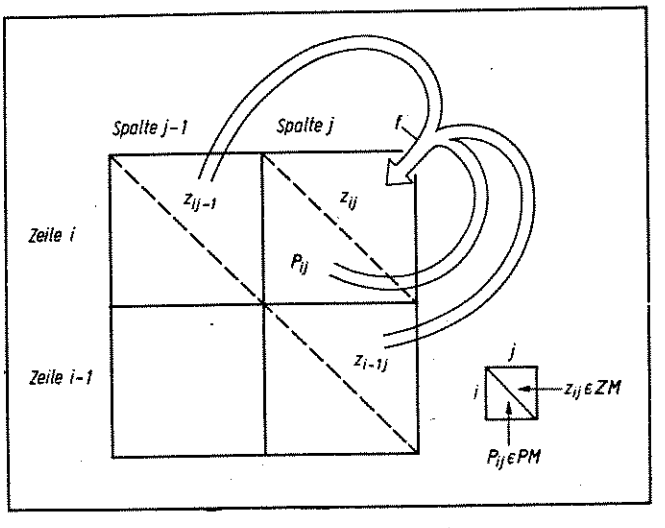
$$z_{ij} = f(z_{ij-1}, [P_{ij}, z_{i-1j}]) ,$$
$$y = g(z_{ij-1}, [P_{ij}, z_{i-1j}]) \qquad \text{mit } i, j > 0$$
$$\text{und} \quad z_{i0} = z_{0j} = 0$$

(Bild 4). z_{ij-1} heißt auch linker, z_{i-1j} auch unterer Wert.
Eine wesentliche Aufgabe besteht nun darin, f und g bezüglich der betrachteten Entwurfsregel geeignet festzulegen. In [1] sind die Tafeln für die Regeln „Diffusionsgebiet mindestens 2λ breit" sowie „Metallüberlappung bei Kontaktfenster mindestens 1λ" angegeben (Bild 5).
Zum Aufstellen von f und g werden im Abschn. 4. noch einige Hinweise gegeben.
Diese Methode ist nicht auf eine bestimmte Technologie beschränkt.

2. Systolische Hardwareimplementierung

Aus [1] geht hervor, daß das in Abschn. 1. beschriebene Konzept softwaremäßig realisiert wurde. Ferner wurde auf die wegen der

Bild 4. Illustration der Überführungsfunktion f von A, also der Ermittlung der Pixelbewertung für P_{ij}



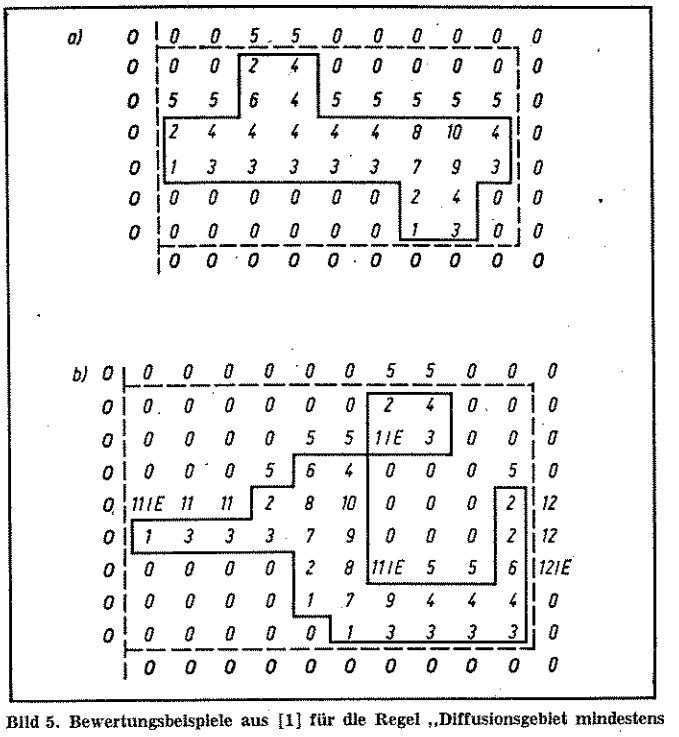


Bild 5. Bewertungsbeispiele aus [1] für die Regel „Diffusionsgebiet mindestens 2λ"

Jeder Wert Z_{ij} repräsentiert für die rechten und oberen Nachbarpunkte bedeutsame Informationen: 0 Punkt liegt außerhalb der Figur, sofern nicht 5, 11 oder 12; 1 linke untere Ecke einer Figur; 2 linker Figurenrand und hoch genug ($\geqq 2\lambda$); 3 unterer Figurenrand und breit genug ($\geqq 2\lambda$); 4 innerer Punkt einer Figur, hoch und breit genug; 5 über der Figur, darunter Figurenrandpunkt, der hoch genug ist; 6 erster Figurenpunkt einer nach links oben offenen Ecke, wobei die Figur darunter genügend hoch ist; 7 Eckpunkt einer nach links unten offenen Ecke. Von ihr aus nach rechts und oben ist die Diagonale zu testen: Werte 8, 9, 10; 11 Fehlerwert: Punkt oberhalb der Figur, die aber dort nicht hoch genug ist; 12 Fehlerwert, links davon Figurenrand, der dort nicht breit genug ist
a) einfache Figur ohne Fehler (vgl. Bild 3);
b) Figur mit einigen Fehlern (das Nicht-Fehlersignal ist weggelassen). Man beachte die sparsame Ausgabe der Fehlermeldung E trotz Fehlerzustandes: nur beim ersten Auftreten wird ein Fehler gemeldet.

Einfachheit dieses Prinzips günstige Möglichkeit hingewiesen, g und f in Hardware zu realisieren, z. B. durch eine PLA, um das Verfahren in der Ausführung zu beschleunigen.
In diesem Abschnitt wird eine Hardwarestruktur für diese Aufgabe vorgeschlagen.
Bild 6a zeigt einen Automaten, der gerade dabei ist, z_{ij} aus z_{ij-1} (linker Wert) und z_{i-1j} (unterer Wert) sowie dem Pixel P_{ij} zu ermitteln. Im vorigen Schritt hatte er z_{ij-1} berechnet (sofern $i > 1$ ist — sonst geht er vom Angfanswert $z_{i0} = 0$ aus). Das bedeutet aber, daß für einen weiteren Automaten A auf dem Punkt $[i+1, j-1]$ genügend Information vorliegt, um den Wert z_{i+1j-1} auf Zeile $i+1$ zu berechnen, vorausgesetzt, er begann im Punkt $[i+1, 0]$, als der erste Automat sich bereits im Punkt $[i, 1]$ befand. Bild 6b zeigt drei derartige identische Automaten, die — vergleichbar mit modernen Erntekolonnen — gleichzeitig drei Zeilen bearbeiten. Ist man in der Lage, die Fehlermeldungen $y \in \boldsymbol{Y}$, die von den Automaten während ihrer Wanderung über $\boldsymbol{PM}$ produziert werden, ohne Zwischenspeicherung auszuwerten, so ist es nicht mehr nötig, die gesamte Matrix $\boldsymbol{ZM}$ zu speichern. Befindet sich ein Automat A auf der Position $[i, j]$, so kann nach Eingabe in A der Wert z_{i-1j} vergessen werden; er wird nicht noch einmal benötigt. Nach Durchlauf einer Kette von k Automaten über k Zeilen muß lediglich eine Zeile aus $\boldsymbol{ZM}$, die „Zustandsspur" des in der obersten Zeile operierenden Automaten, zur Verfügung stehen, wenn k Zeilen höher ein erneuter Durchlauf erfolgen soll.
In der Realität werden sich natürlich die Bilddaten $\boldsymbol{PM}$ bewegen und wird die Automatenkette feststehen. Dabei ist die durch die räumliche Anordnung von Bild 6b beschriebene Relation in eine

zeitliche umzudenken. Bild 7 beschreibt diesen Vorgang. An dieser Stelle wird die Definition des Automaten etwas erweitert:

$$A' = [X', Y', Z', f', g', z_0']$$

mit

$$X' = X,\ Y' = Y \times Z,\ Z' = Z,\ f' = f$$

und

$$g'(z', x') = [g(z', x'), z'] .$$

Der Automat hißt also nicht nur ggf. eine Fehlerflagge (Ausgabe E), sondern gibt auch seinen gegenwärtigen Zustand — die Bewertung des aktuellen Pixels — an den nächsten Automaten weiter, der diesen Zustand als seinen unteren Wert in seine Eingabe einbezieht (Bild 8).

Da Eingabe-/Ausgabevorgänge, bezogen auf Verarbeitungsvorgänge, aufwendig sind, ist man bestrebt, je Eingabe-/Ausgabeoperation so viel Verarbeitung wie möglich zu realisieren. Parallelarbeit und insbesondere systolische[1]) Strukturen [2] aus identischen Zellen sind hervorragende Mittel, von einem relativ langsamen Speichermedium eingegebene Daten im Zusammenspiel mit höchstens lokalen und daher einfachen und schnellen Übertragungsprozessen maximal für die Verarbeitung auszunutzen. Im Fall des Automatenfeldes von Bild 7 werden die ermittelten Werte z_{ij} nicht an ein zentrales Medium zurückgegeben, sondern direkt an den Nachbarprozessor abgegeben, um anschließend „vergessen" zu werden. Dadurch werden $(k-1)(s+1)$ externe Speicherzugriffe je ks Bildpunkte eingespart, die bei der Hardwarerealisierung mit nur einem Automaten nötig gewesen wären. Durch einen speziell zugeschnittenen Speicher (Schieberegister, FIFO) kann die Aufgabe, jede k-te Wertezeile aufzubewahren und als unterste Wertezeile eines erneuten Durchlaufs wieder bereitzustellen, bewältigt werden. (Wird dieser Speicher physisch tatsächlich als Schiebespeicher realisiert, erfüllt er ebenfalls die genannten günstigen Bedingungen eines nur lokalen Datentransfers.) Es bleibt dann nur noch das Problem, die Pixel schnell, der Verarbeitungsgeschwindigkeit des Automatenfeldes gemäß, aus dem Bildspeicher zu lesen oder durch eine andere Einrichtung bereitzustellen.

Bisher wurde die Überprüfung auf nur je eine Entwurfsregel betrachtet. Für jede Entwurfsregel[2]) wäre demnach eine entsprechende Kette aufzubauen. Dies würde eine Vervielfachung der Bearbeitungsdauer durch erneute Eingabe der Bilddaten oder aber unregelmäßige und damit integrationsunfreundliche Verbindungsstrukturen bedeuten. Daher ist es sinnvoll, einmal gelesene Pixelwerte gleich für alle derart zu bearbeitenden Entwurfsregeln auszuwerten: Jeder DRC-Automat gibt dabei nicht nur seinen Zustand und eine Fehlerflagge aus, sondern auch den eingegebenen Pixelwert, um ihn für die Bearbeitung in einem Automaten, der auf andere Entwurfsregeln prüft, bereitzustellen. Eine Anordnung, die auf diese Weise auf mehrere (oder alle) Entwurfsregeln testet und die ebenfalls durch eine reguläre Struktur gekennzeichnet ist, zeigt Bild 9, Bild 10 den zugehörigen Prozessor A'':

$$A'' = [X'', Y'', Z'', f'', g'', z_0'']$$

mit

$$X'' = X', Y'' = Y' \times P, Z'' = Z \times P \quad \text{und}$$

$$f''([z_1, P_1], [P_2, z_2]) = [f(z_1, [P_2, z_2]), P_2] ,$$

$$g''([z_1, P_1], [P_2, z_2]) = [g(z_1, [P_2, z_2]), z_1, P_1]$$

bei $[z_1, P_1] \in Z''$ und $[P_2, z_2] \in X''$.

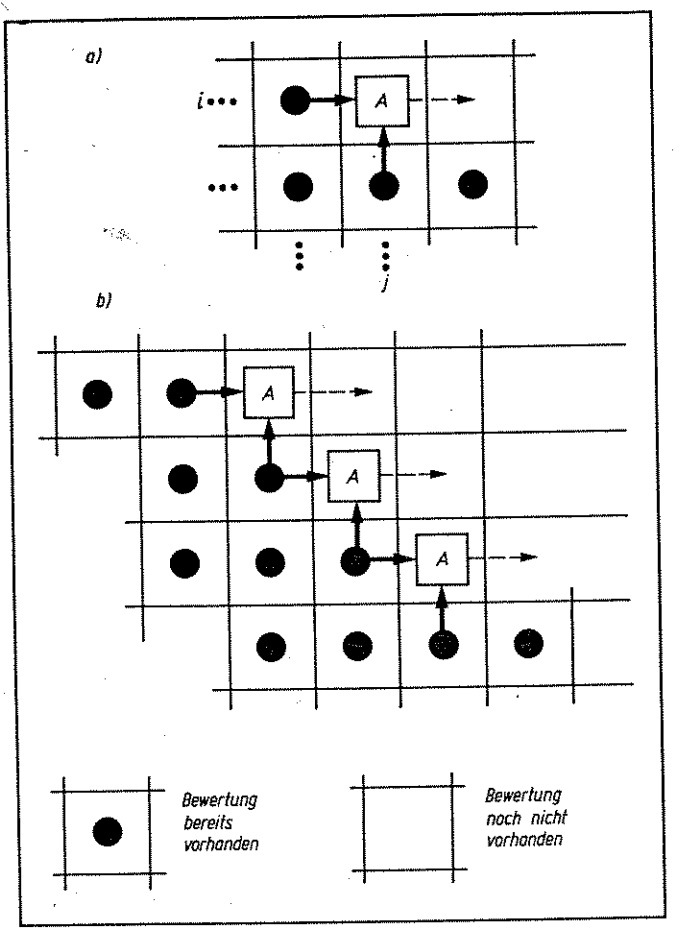
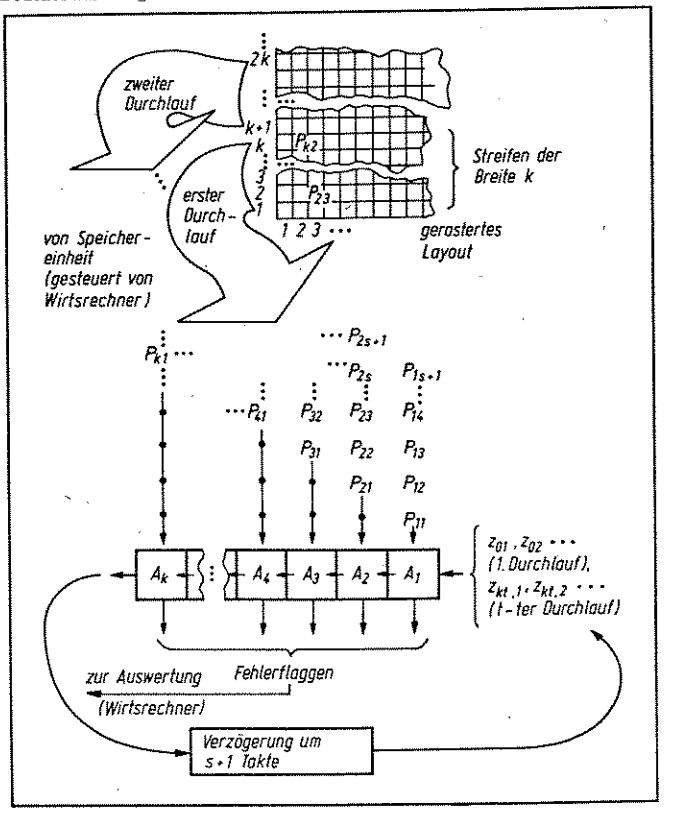


Bild 6. Automat

a) Einzelautomat nach [1]; b) Parallelarbeit mehrerer Exemplare des Automaten A

Bild 7. k identische Automaten $A_1 \ldots A_k$ bearbeiten als Automatenfeld parallel (genauer: systolisch) k Zeilen des Rasterbildes

Der Wirtsrechner kann im einfachsten Fall aus einer „intelligenten" Steuerung des Speichers, der das Rasterbild enthält, sowie aus einem komfortableren Drucker zur Ausgabe von Fehlermeldungen bestehen.



[1]) „Systolisch" ist abgeleitet vom rhythmisch-pulsierenden Vorgang des Blutkreislaufes: Verschiedene „Ströme" von Information pulsieren in verschiedenen Richtungen durch das Prozessorfeld und werden nur lokal verarbeitet.

[2]) [3] gibt etwa 15 unterschiedliche Regeln für einen einfachen nSGT-Prozeß an, die sich aus den MC-Regeln [2] ergeben. In [1] ist von 12 programmierten MOS-Regeln die Rede.

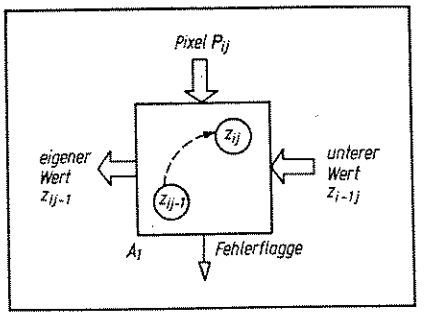
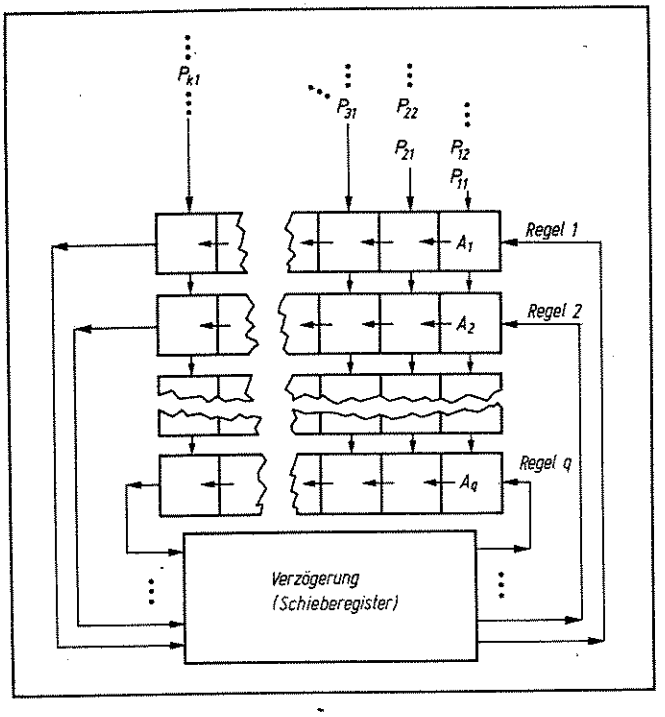


Bild 8. Automat aus dem Feld von Bild 7

Der vorherliegende Automat beliefert A_i mit dem unteren Wert z_{i-1j}. Er selbst ist noch im Zustand z_{ij-1}, der den linken Wert repräsentiert. Zusammen mit dem Pixelwert P_{ij} kann nun der Wert z_{ij} als neuer Zustand von A_i ermittelt werden. Während dieses Vorganges kann der linke Nachbar auf den alten Zustand z_{ij-1} von A_i zugreifen.



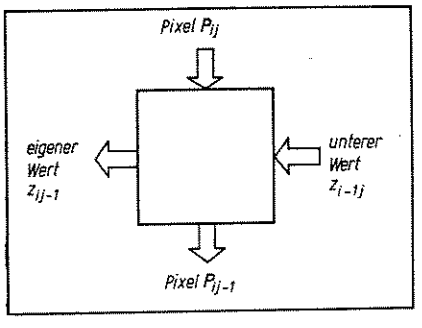


Bild 9. ▲ Zusätzliches Pipelining zur parallelen Bearbeitung von q Entwurfsregeln
(Fehlerflaggen nicht gezeigt)

Bild 10. E/A-Struktur eines Prozessors von Bild 9
Ein eingegebenes Pixel wird einen Taktschritt später weitergegeben

Bei Regeln, die nur „kleine" DRC-Automaten für ihre Überprüfung benötigen, ist natürlich auch die Zusammenfassung mehrerer Automaten zu einem denkbar.

Ein Problem ist es nun, wie und in welcher Form die Fehlermeldungen an die Peripherie dieses Automatenfeldes gelangen können, ohne irreguläre Signalleitungen verwenden zu müssen. Man könnte bei jeder Regel eine Fehlerleitung hinzuschalten und die Fehlerleitungen parallel zur Pixelweitergabe durch das Feld führen. Dann würde die Verbreiterung der Prozessoren in der Größenordnung $q \times$ Leitungsbreite liegen (einschließlich Abstand), wenn q die Regelzahl ist. Nimmt man einen leichten Informationsverlust beim Fehlernachweis in Kauf, kommt man mit weniger Fehlerleitungen aus. Es sei $q \leqq 2^d - 1$ die Regelzahl. Dann können auf $d \sim \mathrm{ld}\ q$ Leitungen die Meldungen 0 („kein Fehler"), 1 („Fehler bei Regel 1") bis q („Fehler bei Regel q") dargestellt werden. Es kann vorkommen, daß ein Pixel mehrere Regeln verletzt. Wird vereinbart, daß sich auf den Fehlerleitungen stets der später entdeckte Fehler durchsetzt, so kann aus der unteren Peripherie des zweidimensionalen Automatenfeldes abgelesen werden, ob sich am vorliegenden Pixel ein Entwurfsfehler bemerkbar machte, und, wenn das der Fall ist, welches die Regel mit der höchsten Nummer ist, die verletzt wurde. Eventuell vorhandene weitere Regelverstöße werden dann nicht gemeldet; man muß sich mit der Kenntnis des Fehlerortes und der Angabe **einer** der verletzten Regeln begnügen. Dies ist aber, da auf einen solchen Nachweis hin ohnehin das Layout manuell überprüft wird, nur eine geringe Einschränkung. Bild 11 zeigt einen Automaten, der dieses leistet.

Zur Auswertung der Fehlermeldungen stehen sodann am unteren Rande des Automatenfeldes $k \cdot \mathrm{ld}\ q$ binäre Signale bereit. Diese können auf $\mathrm{ld}\ k + \mathrm{ld}\ q$ Signale reduziert werden, wenn man eine zusätzliche Kette von Automaten gemäß Bild 12 einsetzt. In die Kette werden schrittweise parallel die Fehlersignale eingegeben. Liegt eine echte Fehlermeldung ($\neq 0$) vor, so setzt sie sich durch und wird mit der Nummer[3]) der Herkunftszeile nach links weitergegeben. Ganz links ist dann als Fehleraussage abzulesen, in mindestens welcher der Zeilen 1 ... k mindestens welche Regel verletzt ist, wobei ein äußerer Zählmechanismus die Spaltennummer des Fehlerortes bereitstellt. Läßt man überdies

[3]) Relative Zeilennummer, d. h. zwischen 1 und k liegend; die tatsächliche Zeilennummer wird aus dieser und der Durchlaufnummer t ermittelt.

Bild 11. ▶ Behandlung von Fehlermeldungen im Prozessor für die Regel p

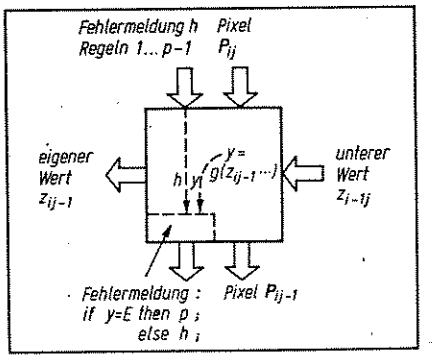
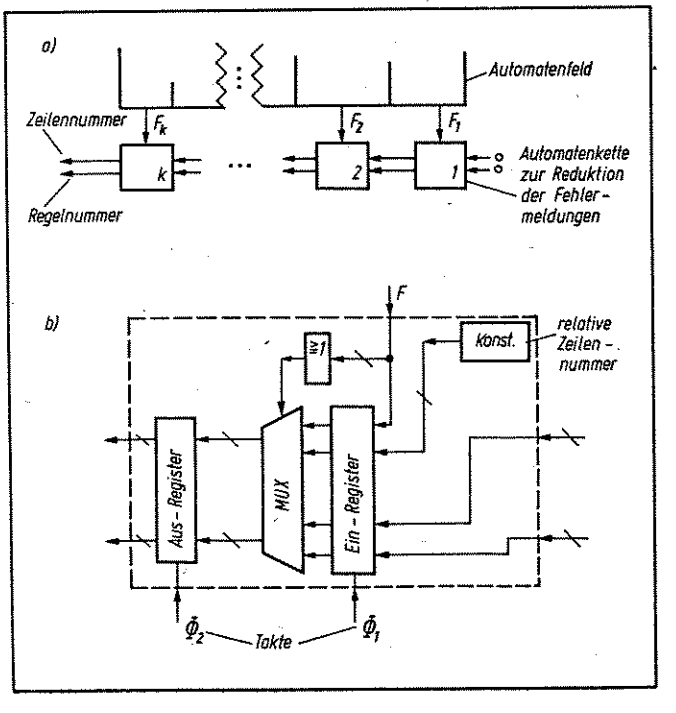


Bild 12. ▼ Reduktion der Fehlerinformation
a) Feld der Fehlerreduktoren als unterste Kette im Gesamtfeld;
b) Prinzipaufbau einer Reduktionszelle



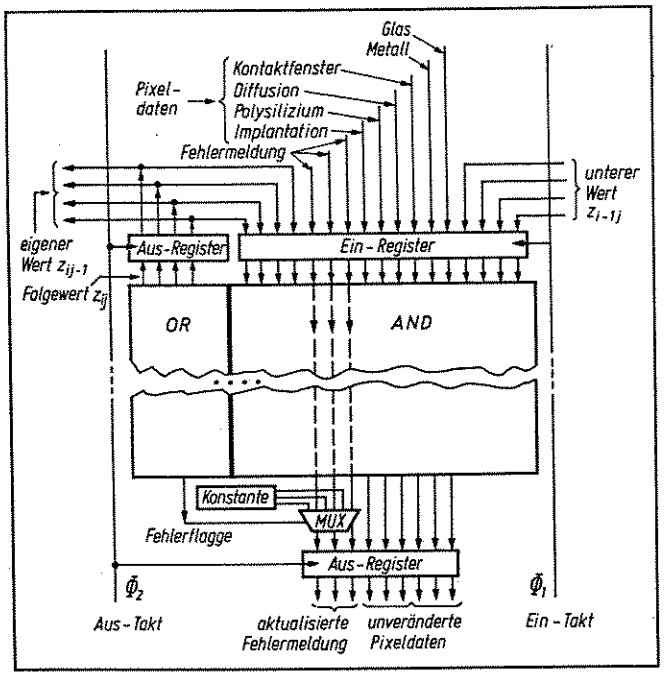


Bild 13. Prozessorrealisierung mit PLA, Registern und Multiplexer für den Fall von $\leqq 16$ Zuständen im Automaten zur Prüfung von gleichzeitig $\leqq 7$ Entwurfsregeln bei 6 Layoutebenen (vgl. [2])

Im Fehlerfall wird die übernommene Fehlernummer einfach über die Multiplexersteuerung durch die Konstante p überdeckt, wenn p die Nummer der Entwurfsregel ist, für die die PLA programmiert ist.
Anmerkungen: 1. Die Fehlermeldungen können auch links an der PLA vorbeigeführt werden, so daß die Prozessorbreite reduziert wird. Das notwendige Ein-Register wird dann vom Φ_1-Takt des linken Nachbarn gesteuert; 2. Die Fehlerflagge kann auch oben aus dem OR-Bereich herausgeführt und der Multiplexer an der Eingangsseite realisiert werden, oder die Flagge wird eingangsseitig ungetaktet dekodiert, durchläuft den AND-Bereich und steuert ausgangsseitig in einfacher Weise den Multiplexer.

die (relative) Zeilennummer nicht mitwandern, so ist nur die Aussage, ob im Durchlauf t in einer (bekannten) Spalte und innerhalb der Zeilen $t \cdot k + 1 \ldots (t+1) \cdot k$ mindestens die benannte Regel verletzt ist, wobei klar ist, daß dies keine Regel mit höherer Nummer betrifft.

3. Realisierungsmöglichkeit

Im allgemeinen sind die von den DRC-Automaten zu bearbeitenden Algorithmen so einfach, daß sie als PLA-Automat realisierbar sind [2] und [7]. Dies trifft mindestens für Figurenabmessungen bis drei oder vier λ zu, aber auch für andere Regeln, z. B. bei der Metallüberlappung beim einfachen Kontaktfenster [1].
Bild 13 zeigt eine Lösungsmöglichkeit, bei der eine PLA zum Einsatz kommt. Die gegenwärtig sich vollziehende Entwicklung der VLSI-Technologie [2] [6] und [9] wird es erlauben, mehrere oder viele derartige PLA-Strukturen auf einem einzigen Schaltkreis unterzubringen. Denbkar ist jedoch auch eine Realisierung durch Verknüpfung fertiger Schaltkreise, wie Register und FPLA.
In MOS-Technologie sind PLA, die Register und der Multiplexer sehr einfach zu realisieren [2] [6] und [8]. Für den im Bild 13 gezeigten Fall (6 Pixelbit, 4 Zustandsbit, 3 Bitleitungen für Fehlermeldungen) werden Ausmaße von maximal 200 λ (Breite) mal (170 λ + 16 λ mal Anzahl Konjunktionsterme) anzusetzen sein, was bei $\lambda = 2{,}5$ µm und 100 Termen eine Fläche von 500 µm × 4425 µm bedeutet [8].

4. Aufstellen der Automatenfunktionen

In [1] wird vermerkt, daß die Automatentabellen eigens zu diesem Zweck und unter Mühen aufgestellt wurden. So benötigten die Autoren für alle 12 nMOS-Entwurfsregeln mehrere Wochen. Es ist aber beispielsweise möglich, für vorgegebene Breite w die DRC-Automatentabellen zum Test auf die Einhaltung der Mindestbreite bzw. des Mindestabstandes $w\lambda$ algorithmisch zu generieren. Zu diesem Zweck wird für die Werte z zunächst von einer Vektordarstellung ausgegangen:

$$z = [H, B, E, C, OR, FR, FO] .$$

Dabei bedeuten die ersten zwei Angaben erreichte (gezählte) Strukturhöhe bzw. -breite, und die anderen fünf sind Flaggen, die letzten beiden Fehlerflaggen. Im einzelnen bedeuten:

H	Höhe, d. h. Anzahl aufeinanderfolgender nichtleerer Pixelwerte der vorliegenden Spalte, gezählt vom aktuellen Punkt an abwärts
B	Breite entsprechend linkswärts
E	markiert Bereich einer nach links unten offenen Ecke
C	markiert Bereich einer nach links oben offenen Ecke
OR	markiert einen leeren Punkt unmittelbar oberhalb eines nichtleeren Punktes (oberer Randpunkt)
ER	markiert „Rechtsfehler": Struktur nach rechts hin zu schmal
FO	markiert „Oben-Fehler": Struktur nach oben zu flach.

Mit einer so definierten Zustandsmenge kann auf einfache Weise ein partiell definierter Automat mit $Z \subseteqq \{1, \ldots, w\}^2 \times \{0, 1\}^5$ konstruiert werden, der das Verlangte leistet. Nach Minimierung der Zustandsmenge können eine Kodierung der Zustände, die gemäß der vorgesehenen Implementierung (PLA, ...) auch quasiminimal erfolgen kann, und eine anschließende Boolesche Minimierung die Programmiervorschrift für die Prozessorzelle liefern.

Zusammenfassung und Schlußbemerkung

Es wurde eine automatentheoretische Methode zur Entwurfsregelprüfung [1] vorgestellt und gezeigt, daß und wie diese sich hervorragend für eine systolische Hardwareimplementierung eignet, bei der durch paralleles Arbeiten gruppenweise identischer Prozessoren die bislang mit enormem Aufwand arbeitenden DRC-Softwaresysteme erheblich entlastet werden können. Großrechnereinsatz ist ggf. nur noch zum Füllen eines intelligenten Speichers mit den Rasterbilddaten des zu prüfenden Entwurfs und zum Übergeben von Referenzdaten zur nutzerfreundlichen Ausgabe der Fehlermeldungen durch ein intelligentes Ausgabegerät vonnöten.
Die stürmische Entwicklung der VLSI-Technologie [2] [6] und [9] läßt erwarten, daß diese sich bald selbst ihre spezifischen Verifizierungssysteme schafft bzw. sich verstärkt selbst in den Dienst nimmt.

Literatur

[1] *Eustace, R. A.; Mukhopadhyay, A.:* A Deterministic Finite Automation Approach to Design Rule Checking for VLSI. Proceedings 19th Design Automation Conference. June 14—16, 1982, Las Vegas, Nevada, USA, pp. 712—717.
[2] *Mead, C.; Conway, L.:* Introduction to VLSI Systems. Addison-Wesley. Reading. Mass., USA, 1980.
[3] *Baker, C. M.:* Artwork Analysis Tools for VLSI Circuits. MS Thesis, MIT Cambridge, Mass., USA, May 1980.
[4] *Baker, C. M.; Terman, C.:* Tools for Verifying Integrated Circuit Designs. Lambda, 4th Quarter 1980, pp. 22—30.
[5] *Starke, P. H.:* Abstrakte Automaten. Berlin: VEB Deutscher Verlag der Wissenschaften 1969.
[6] *Glasser, L. A.; Penfield Jr., P.:* An Interactive PLA Generator as an Archetype for a New VLSI Design Methodology. IEEE International Conference on Circuit and Computers. Oct. 1—3, 1980, Port Chester, New York, USA.
[7] *Eckardt, D.; Konrad, E.; Leupold, W.:* Entwurf komplexer digitaler Schaltungen. Band 175 der REIHE AUTOMATISIERUNGSTECHNIK. Berlin: VEB Verlag Technik 1976.
[8] *Hon, R. W.; Sequin, C. H.:* A Guide to LSI Implementation. 2nd Edition, XEROX PARC, Palo Alto, CA, USA, 1980.
[9] *Heinz, G.:* Grundzüge des höchstintegrierten Schaltkreisentwurfs (VLSI). INT-Mitteilungen, Ausgabe A, 2/82.

msr 7797