Dr. rer. nat. *K.-A. Zech* und Dipl.-Ing. *W. Klarkowski*, Berlin[1])

Rechnerprogramme zur Unterstützung der Fehlerdiagnose digitaler Schaltungen

IA 1.1.1.2.; 4.0.1.2.

0. Einleitung

Die Kosten einer elektronischen Baueinheit werden in steigendem Maße neben dem Entwicklungsaufwand durch die Prüfung und Reparatur im Fertigungsprozeß bestimmt. Man versuchte daher schon früh, die Prüfung und Fehlersuche zu automatisieren oder wenigstens die Unterlagen hierzu rechnergestützt zu erarbeiten [1] und [2]. Auch in der DDR sind diesbezügliche Untersuchungen vorgenommen worden, deren Ergebnisse teilweise rechentechnisch implementiert wurden [3] [4] [32] und [33].

In dem folgenden Artikel sollen die Programme des Pakets SIMPER und TEGAL zur Simulation und Analyse sowie Testgenerierung für digitale Schaltungen vorgestellt werden, die im Institut für Nachrichtentechnik Berlin und in der Humboldt-Universität zu Berlin erarbeitet wurden [3] [5] [18] [34] und [35].

Diese Programme bewältigen folgende Aufgaben (Bild 1):

- Testsatzgenerierung für kombinatorische Schaltungen
- Testfolgengenerierung für synchron-getaktete sequentielle Schaltungen
- Testfolgenberechnung für asynchrone sequentielle Schaltungen, die durch ein quasi-synchrones Modell beschreibbar sind.
- Stochastische Testgenerierung für kombinatorische und sequentielle Schaltungen
- 2wertige Simulation mit typischen nichtlinearen Zeitparametern (präzise Zeitsimulation)
- 3wertige Simulation (Toleranzsimulation) als Form der Worst-case-Simulation mit extremen nichtlinearen Zeitparametern
- Prüfschrittabellensimulation
- Parallele Fehlersimulation.

Sie sind im wesentlichen in PL/1 formuliert und benötigen einen ESER-Rechner mit mindestens 256 K Bytes (Betriebssystem OS/ES-MFT).

In folgendem werden vorwiegend die Programme zur Unterstützung des Prüfprozesses in der Fertigung vorgestellt. Andere Programme des Pakets SIMPER werden der Vollständigkeit halber mit erwähnt[2]).

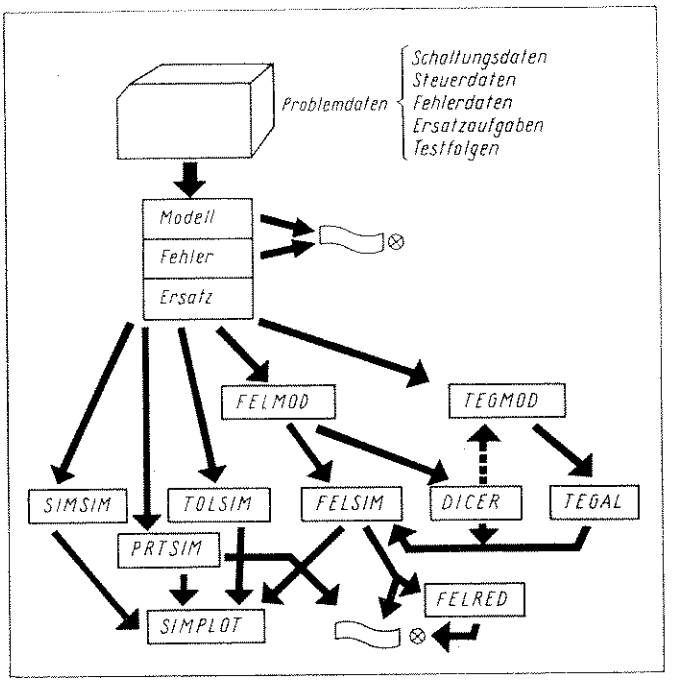


Bild 1. Programme des Programmpakets SIMPER sowie die zur Fehlersimulation (FELSIM) und Testgenerierung (TEGAL, DICER) gehörenden Programme

(FELRED kann zur Reduktion der Fehlerortungsinformation für den Prüfplatz eingesetzt werden — s. Abschn. 2; ⊗ für Prüfplatz)

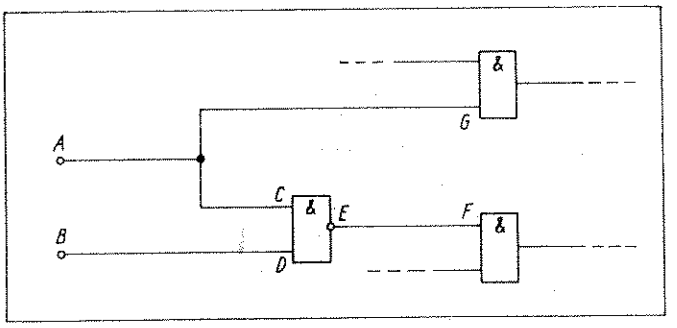


Bild 2. Beispiel für die Erzeugung der Fehlerdaten (Schaltungsausschnitt, vgl. Tafel)

1. Primärdaten

Vom Anwender der Programme sind folgende Daten vorzugeben bzw. bereitzustellen.

1.1. Schaltungsdaten

Diese bestehen aus einer Liste der verwendeten Bausteine und einer Leitungsliste, in der die untereinander verbundenen Bauelementestifte aufgeführt sind. Die Angaben werden auf Formblätter notiert, so daß der Erfassungs- und Korrekturaufwand relativ gering ist. Dargestellt werden die Schaltungsdaten gemäß dem entsprechenden Werkstandard des INT, der auch für Programme zur Unterstützung konstruktiver Schritte (Leiterplattenberechnung) gültig ist.

1.2. Bauelementekatalog

Die Parameter der Bauelemente (logische, zeitliche u. a. Größen) werden im Katalog der Bauelemente vereinbart. Der Katalo

[1]) Mitteilung aus dem ORZ des Instituts für Nachrichtentechnik Berlin, Abt. Technische Probleme, und dem ORZ der Humboldt-Universität zu Berlin, Abt. Mathematische Modellierung.

[2]) Eine ältere Version des Programmpakets SIMPER wurde in [5] vom Bearbeiterkollektiv ausführlicher vorgestellt.

kann auf einer Datei abgelegt und aktuell ergänzt werden, so daß der Nutzer in der Regel diese Daten nicht selbst erarbeiten muß.

1.3. Steuerdaten

Die Steuerdaten beschreiben die Zeitbedingungen des Simulationsexperiments, z. B. Länge einer Periode, Zeitquantgröße, Standardzeiten für Taktflanken u. a.

Darüber hinaus enthalten sie die Initial- und Eingangsbedingungen (Testfolge).

1.4. Fehlerdaten (Bild 2)

Vorgegebene Fehlerdaten beinhalten einzelne, vom Fehlersimulations- oder Testgenerierungsprogramm zu berücksichtigende Fehler(-klassen). Liegen solche nicht vor, werden alle möglichen logischen Einfachfehler (*s-a*-Fehler) angenommen und zu Klassen äquivalenter Fehler zusammengefaßt (vgl. Abschn. 4.1.).

1.5. Ersatzschaltungen und Ersetzungsaufgaben

Ersatzschaltungen, die für solche Funktionselemente eingesetzt werden können, die etwa für die Problemprogramme zu komplex sind, werden analog Abschn. 1.1. erfaßt. Die Ersetzungsaufgaben nennen für bestimmte Ersatzschaltungen diejenigen Funktionselemente, die durch diese ersetzt werden sollen.

2. Erarbeitung der rechnerinternen Modelle

Die primären Schaltungsdaten müssen auf Fehler überprüft und in eine den Problemprogrammen gemäße Form gebracht werden. Diese Aufgabe wird durch die Programme MODELL (Zwischenmodell sowie Strukturmodell für die reine Simulation), FELMOD (für die Fehlersimulation) und TEGMOD (für Testgenerierung) gelöst.

Tafel. Fehler zum Beispiel von Bild 2

Sämtliche Fehler		Fehler nach Klasseneinteilung	
		Klassennr.	Fehlernr.
1	A-0	1	1
2	A-1	2	2
3	B-0	3	3
4	B-1		7
5	C-0		5
6	C-1		10
7	D-0		12
8	D-1		4
9	E-0	4	8
10	E-1		6
11	F-0		9
12	F-1	5	11
13	G-0	6	...
14	G-1		13
.	.	7	...
.	.		14
.	.	8	.
		.	.
		.	.

Für TTL-Logik fallen die Fehler 5 und 13 weg (C-0 und G-0), da sie den Fehlern 1 (A-0) implizieren. Sie werden dann der Fehlerklasse 1 zugeordnet.

2.1. Programm MODELL

Durch MODELL werden die in der Regel auf Lochkarten vorliegenden Schaltungsdaten in ein internes Modell (Strukturmodell) überführt. Gleichzeitig werden sie auf Fehler überprüft und ggf. die Fehler ausgewiesen. Das Strukturmodell besteht aus Listen für Funktionselemente, Leitungen, Stifte (Pins), Typen, Schaltzeiten und verschiedenen Referenzen, die im Prüfprozeß benötigt werden.

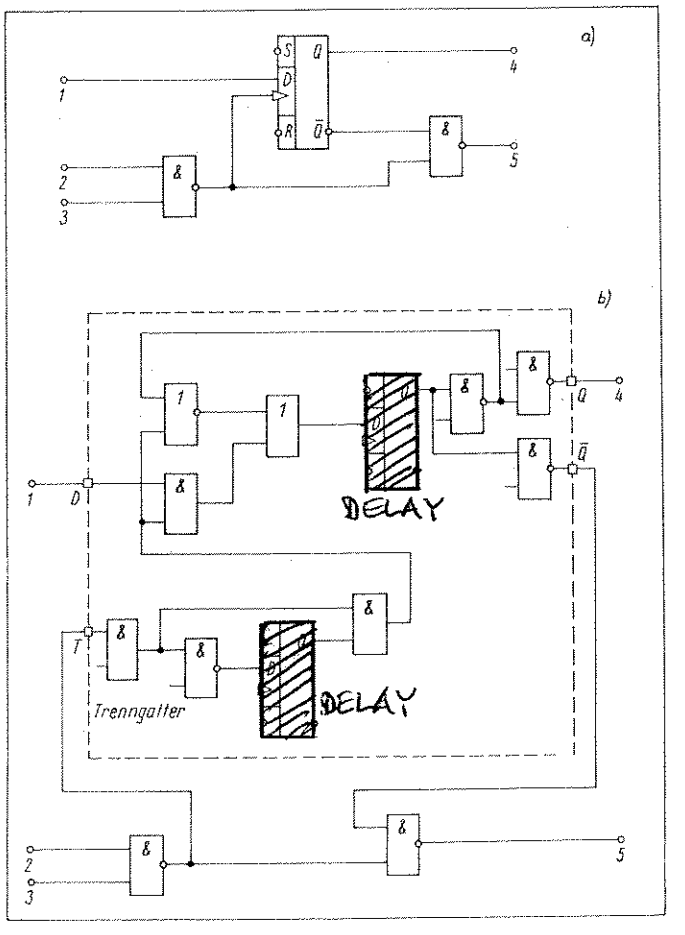
Für die rechnergestützte Schaltungsprüfung mit Hilfe des KRS 4201 kann ein Lochstreifen ausgegeben werden, der alle benötigten Angaben zur Schaltung enthält.

2.2. Programm FEHLER

Dieses Programm liest entweder – falls vorhanden – vom Anwender vorgegebene Fehlerklassen ein, oder es werden sämtliche in der Schaltung vorkommenden Stifte als mögliche Fehlerorte angesehen, auf denen sich sowohl *s-a-o-* als auch *s-a*-1-Fehler befinden können (stuck-type faults). Im letzten Fall werden dann die Fehler in Klassen von logisch nicht unterscheidbaren Fehlern unterteilt und Repräsentanten dieser Klassen bestimmt (Tafel). Nur diese werden von den Problemprogrammen berücksichtigt. Kurzschlüsse von Signalleitungen werden im Falle dominanter Werte auf Haftfehler zurückgeführt [10].

Bild 3. Beispiel für die Notwendigkeit, Ersatzschaltungen einzufügen

Bei dem D-Flip-Flop in 3 (a) ist der Takteingang logisch beschaltet. Um den Takt wie ein normales Signal zu behandeln, wird eine Ersatzschaltung für das D-Flip-Flop eingesetzt, die die schaltende Taktflanke erkennt. In der Ersatzschaltung steuern „ideale Schalter" den Taktablauf. In den Ein- bzw. Ausgängen werden bei Ausfächerungsleitungen Füll- (oder Trenn-)gatter eingefügt [18]. Bild 6 (b) zeigt das Ergebnis der Einsetzung.



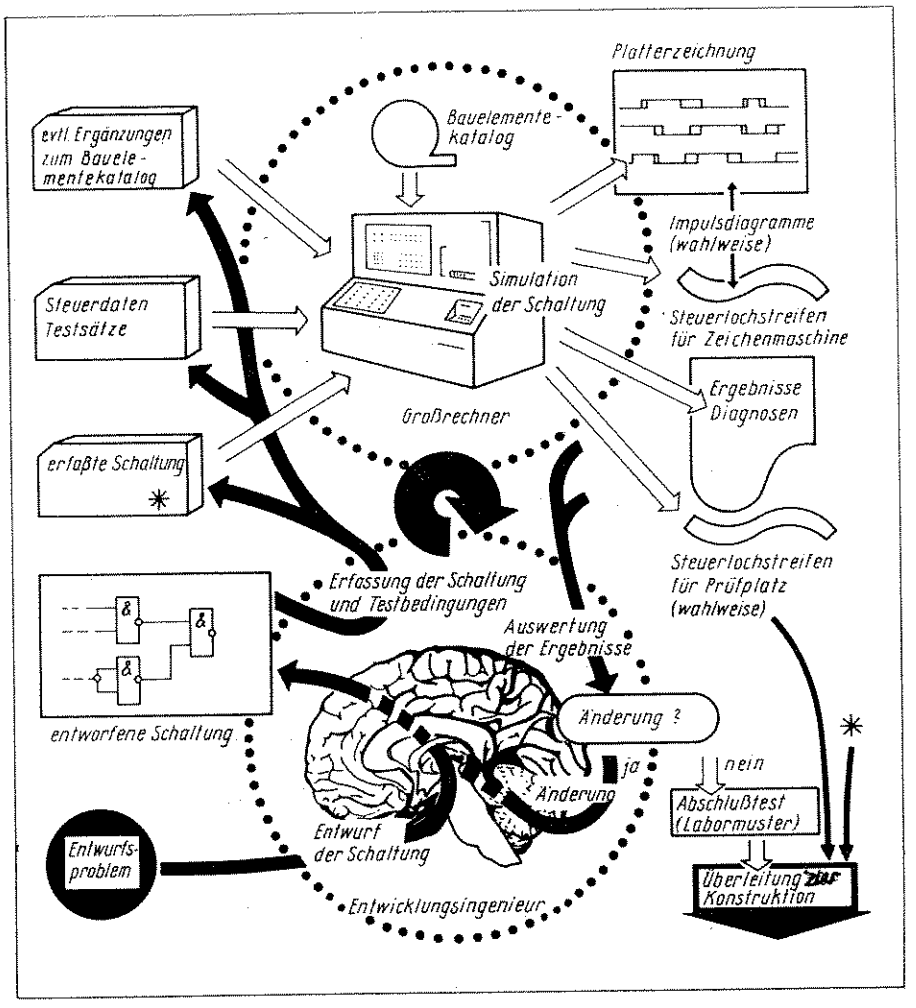


Bild 4
Rolle der Entwurfssimulation im Entwicklungsprozeß einer digitalen Schaltung

2.3. Programm ERSATZ

Bei Angabe von Ersetzungsaufgaben können für Funktionselemente, die in den Schaltungsdaten aufgeführt sind, Ersatzschaltungen eingesetzt werden. Diese ist i. allg. bei der Testsatzgenerierung durch TEGAL und der Fehlersimulation mit FELSIM notwendig, da z. Z. hier explizit nur die Grundgatter und getaktete Verzögerungsglieder (entsprechend D-Flip-Flops, bei denen neben dem Takt und dem D-Eingang nur der Q-Ausgang beschaltet sind) auftreten dürfen. Auch sind hier logisch umgeformte Taktsignale und die Verwendung anderer Steuersignale nicht erlaubt [18] (Bild 3).

ERSATZ läuft in der Regel nach dem Programm FEHLER, so daß Fehler innerhalb der Schaltkreismodule nicht berücksichtigt werden. Dies ist jedoch möglich bei umgekehrter Reihenfolge, obgleich häufig die verwendeten Ersatzschaltungen nicht mit den Innenschaltungen der modellierten Schaltkreise übereinstimmen. Das Ergebnis einer Testanalyse wird dadurch aber nicht „schwächer", die Anforderungen an die Testsatzgenerierung erhöhen sich. Die Anzahl der zu betrachtenden Fehler und damit der Rechneraufwand/Speicherbedarf können drastisch zunehmen.

2.4. Programme FELMOD und TEGMOD

Diese beiden Programme ermitteln aus den Ergebnissen von ERSATZ und MODELL sowie FEHLER die für die Fehlersimulation bzw. Testsatzgenerierung adäquaten Schaltungsmodelle.

3. Entwurfssimulation (Bild 4)

Unter Entwurfssimulation soll im Unterschied zur Fehlersimulation die Simulation einer Schaltung verstanden werden, die als korrekt produziert, nicht aber notwendig als korrekt entworfen gilt [6] und [7]. Die Simulation erfolgt ereignisorientiert mit selektiver Pfadverfolgung [8] und [9]. Das logische und zeitliche Verhalten der Funktionselemente wird modelliert durch sog. Verhaltensprozeduren, die jeweils für bestimmte Typengruppen vorliegen bzw. bei Bedarf neu erarbeitet werden müssen. Es können gegenwärtig SSI- und unterschiedliche MSI-Kreise der verschiedenen Reihen der TTL- und MOS-Logik u. a. bearbeitet werden.

Die einzelnen Verfahren werden im folgenden kurz erläutert (vgl. [5]).

3.1. Programm SIMSIM (Mittelwertsimulation)

Die Schaltung wird 2wertig simuliert unter Annahme der vom Bauelementehersteller angegebenen typischen Zeitparameter. Neben dem resultierenden Impulsablaufdiagramm werden Diagnosen ausgegeben, die „Warnungen" sind oder auf entdeckte Entwurfs- oder Dokumentationsfehler hinweisen. Ausgewiesen werden zu kurze oder gelöschte Impulse (Spikes), zu kurze Vorbereitungs- und Haltezeiten bei Flip-Flops und ähnliches. Hasards und Wettläufe werden nicht systematisch erfaßt, jedoch sind indirekte Aussagen darüber ableitbar.

Das Simulationsergebnis ist i. allg. zu optimistisch.

3.2. *Programm TOLSIM (Toleranzsimulation)*

Basierend auf den Werten 0, 1 und x (unbestimmt) wird eine Simulation mit Minimal- und Maximalzeiten der Funktionselemente als eine Form der Worst-case-Simulation durchgeführt. Der dritte Wert, x, beschreibt dabei ein unbestimmtes Signal, durch das ein Übergangsbereich modelliert wird oder das bei unbekannten Anfangsbelegungen von Speichern, bei Wettläufen usw. vorkommen kann. Wie bei der Mittelwertsimulation werden Diagnosen über mögliche Fehler ausgegeben. Das Ergebnis ist i. allg. zu pessimistisch.

3.3. *Programm PRTSIM (Prüfschrittabellensimulation)*

Diese Simulationsart ist eine Sonderform der Mittelwertsimulation, deren Aufgabe hauptsächlich darin besteht, die stationären Simulationsergebnisse für die Schaltungsprüfung (Steuerlochstreifen für KRS 4201 und Signallisten) auszugeben. Die Simulationsergebnisse für den Prüfplatz enthalten auch die (korrekten) Potentiale aller inneren Leitungen für eine Fehlerlokalisierung.

Literatur

[1] *Eldred, R. D.:* Test routines based on symbolic logic statements. Journal ACM **6** (1959) No. 1, pp. 33 – 36.

[2] *Roth, P. J.:* Diagnosis of automata failures: A calculus and a method. IBM Journal Res. & Dev. **10** (1966) pp. 278 – 291.

[3] *Bobey, K., Heinz, E.:* Diagnosetestberechnung für digitale Schaltungen. Nachrichtentechnik-Elektronik **25** (1975) H. 8, S. 287 – 290.

[4] *Dube, E.*, u. a.: Ein Algorithmus für den Prozeßrechner zur Fehlerortung von Steckeinheiten. Rechentechnik/Datenverarbeitung **9** (1972) 4. Beiheft, S. 31 – 35.

[5] *Gessner, E. E. E.*, u. a.: Rechnersimulation digitaler Schaltungen reduziert Labormessungen. Sozialistische Rationalisierung in der Elektronik/Elektrotechnik **8** (1976) S. 246 – 252.

[6] *Szygenda, S. A., Thompson, E. W.:* Modeling and digital Simulation for design verification and diagnosis. LEEE T-C-**25** (1976) No. 12, pp. 1242 – 1253.

[7] *Szygenda, S. A., Thompson, E. W.:* Digital systems simulation. Computer **8** (1975) No. 3, pp. 23 – 49.

[8] *Ulrich, E. G.:* Exclusive simulation of activity in digital networks. Com. ACM **12** (1969) No. 2, pp. 102 – 110.

[9] *Ulrich, E. G.:* Time-sequenced logical simulation based on circuit delay and selective tracing of active network paths. Proc. of the 20th. Nat. Conf. of ACM, 1965, pp. 437 – 488.

[10] *Kaposi, J. F.*, u. a.: Testing switching networks for shortcircuit faults. Electronic Letters **8** (1972) No. 24, pp. 586 – 587.

[11] *Armstrong, D. B.:* A deductive method for simulation faults in logic circuits. IEEE Trans. C-**21** (1972) No. 5, pp. 464 – 471.

[12] *Ermilov, V. A.:* Metod otbora Suščestvennych neispravnostej dlja diagnostiki cifrovych schem Avtomatika i Telemechanika (1971) H. 1, S. 159 – 167; H. 3, S. 107 – 113.

[13] *Chang, H. Y., Chappell, S. G.:* Deductive techniques for simulating logic circuits. Computer **8** (1975) No. 3, pp. 52 – 59.

[14] *Verma, J. P.;* u. a.: Automatic test-generation and test-verification of digital systems. Proc. 11th. design automation workshop, June 17 – 19, (1974) Denver, Colorado, pp. 149 – 158.

[15] *Chang, H. Y.;* u. a.: Comparison of parallel and deductive faults simulation methods. IEEE Trans. C-**23** (1974) No. 11, pp. 1132 bis 1138.

[16] *Jost, W.:* PROFET – Ein Programmsystem zur Simulation von Hardwarefehlern in Digitalrechnern. Informationen Fernsprech-Vermittlungstechnik **8** (1972) H. 3, S. 105 – 117.

[17] *Kubo, H.:* A procedure for generating test sequences to detect sequential circuit failures. NEC Res. Developments **12** (1968) No. 10, pp. 69 – 78.

[18] *Zech, K.-A.; Klarkowski, W.:* Synchron-getaktete Modelle für asynchrone Schaltungskonfigurationen. msr **20** (1977) H. 9, S. 504 bis 507.

[19] *Amar, A.;* u. a.: Diagnosis of large combinational networks. IEEE Trans. EC-**16** (1967) No. 10, pp. 675 – 680.

[20] *Chiang, A. C.;* u. a.: Path sensitization, partial Boolean difference and automated fault diagnosis. IEEE Trans. C-**21** (1972) No. 2, pp. 189 – 259.

[21] *Flomenhoft, M. J.*; u. a.: Algebraic techniques for finding tests for several fault types. Int. Symp. Fault-Toleraut Computing, June 1973, pp. 85 – 90.

[22] *Görke, W.:* Fehlerdiagnose digitaler Schaltungen. Stuttgart: BSB B. G. Teubner Verlagsgesellschaft 1973.

[23] *Plakk, M.:* Diagnose von Schaltnetzwerken, Diplomarbeit, TH Karl-Marx-Stadt, Sektion Informationstechnik, 1975.

[24] *Sellers, F. F.;* u. a.: Error-detecting logic. New York: Mc-Graw-Hill 1969.

[25] *Xuan Quynh, N.:* An algorithm for fault detection in logical combinational many-level fan-out networks. Elektronische Informationsverarbeitung und Kybernetik **12** (1976) H. 10, S. 483 – 496.

[26] *Muth, P.:* Erstellen von Fehlererkennungs-Experimenten für Schaltnetze und Schaltwerke unter Verwendung eines neunwertigen Schaltungsmodells. Nachr. techn. Fachber. **49** (1974) S. 175 – 183.

[27] *Arima, T.;* u. a.: A new heuristic test generation algorithm for sequential circuits. Proc. 11th design automation workshop, June 17 to 19, 1974, Denver, Colorado, pp. 169 – 176.

[28] *Agrawal, P.; Agrawal, V. D.:* On Monte Carlo Testing of Logic Tree Networks. IEEE TC-**25** (1976) No. 6, pp. 664 – 667.

[29] *Agrawal, P.; Agrawal, V. D.:* Probabilistic Analysis of Random Test Generation Method for Irredundant Combinational Logic Networks. IEEE TC-**24** (1975) No. 7, pp. 691 – 695.

[30] *Parker, K.:* Probabilistic Test Generation. Techn. Note. Nr. 18, Jan. 1973, Stanford University, Stanford, Cal., USA.

[31] *Fellberg, G.;* u. a.: Einige Bemerkungen zum Prüfen und Herstellen von Fehlererkennungshilfen bei digitalen Schaltungen. msr **18** (1975) H. 6, S. 206.

[32] *Habiger, E.:* Erfahrungen und Schlußfolgerungen aus erster Erprobung eines Programms zur rechnergestützten Generierung von Prüf- und Fehlerortungsunterlagen für digitale Steckeinheiten. Der VEM-Elektro-Anlagenbau **13** (1977) H. 2, S. 79 – 83.

[33] *Heine, M.:* Ein Verfahren zur Ermittlung von Testfolgen für sequentielle Schaltungen. Dissertation, TH Ilmenau 1976.

[34] *Heinz, E.; Bobey, K.:* Algorithmische Verfahren zur Testmengenberechnung für die Fehlererkennung und Fehlerlokalisierung in digitalen Schaltungen. Wiss. Z. d. HUB **27** (1978) H. 2.

[35] *Klarkowski, W.:* Die Programme TEGAL2 und TEGAL3 zur Testmengenberechnung für digitale Schaltungen. Wiss. Z. der HUB **27** (1978) H. 2.

[36] *Ulrich, E. G.; Baker, T.:* Concurrent simulation of nearly identical digital networks. Computer **7** (1974) No. 4, pp. 39 – 44.

[37] *Abramovici, M.; Breuer, M. A.; Kumas, K.:* Concurrent fault simulation and funktional level modeling. Proc. of 14th Design Automation Conference, New Orleans 1977, pp. 128 – 137.

msr 6976 (Beitrag wird im H. 4/1980 fortgesetzt)

Dr. rer. nat. *K.-A. Zech* und Dipl.-Ing. *W. Klarkowski*, Berlin

Rechnerprogramme zur Unterstützung der Fehlerdiagnose digitaler Schaltungen (Teil II)[1]

IA 1.1.1.2.; 4.0.1.2.

4. Fehlersimulation

4.1. *Fehlermodell*

Zugrundegelegt werden logische Einfachfehler (*s-a*-Fehler). Kurzschlüsse lassen sich nach [10] mit Hilfe einer Ersatzschaltung und *s-a*-Fehlern modellieren, so daß dieses Fehlermodell als ausreichend angesehen werden kann. Die Behandlung von mehrfachen *s-a*-Fehlern ist gegenwärtig nicht vorgesehen, prinzipiell aber möglich. Negations- und Pegelfehler bleiben unberücksichtigt, während intermittierende Fehler teilweise am Prüfplatz beachtet werden können.

4.2. *Prinzipien der Fehlersimulation [6] und [7]*

Die Fehlersimulation modelliert das Verhalten einer fehlerhaften Schaltung für vorgegebene Fehler und Eingangsbelegungen. Sie stellt fest, ob die Eingangsbelegungen (Tests) in der Lage sind, jeden der interessierenden Fehler an den Ausgangsklemmen der Schaltung sichtbar zu machen (Testanalyse).
Aus den Ergebnissen ist ableitbar, welche Fehler an einer fehlerhaften Reaktion der Schaltung beteiligt sein können, d. h., eine Fehlerinformation wird ermittelt. Erweist sich bei der Prüfung eine Schaltung als fehlerhaft, so kann mit Hilfe der Fehlerinformation der Fehler bis zu einem gewissen Grade lokalisiert werden.
Im Einsatz befinden sich gegenwärtig vorwiegend folgende vier Verfahren zur Testsatzanalyse:

— Einzelfehlersimulation (z. B. [31])
Nach Einbau eines Fehlers in das Schaltungsmodell wird die Schaltung mit typischen oder extremen Zeitparametern simuliert (Mittelwert- oder Toleranzsimulation)

Vorteil:
Das Zeitverhalten wird unter dem Fehlereinfluß relativ genau simuliert; fehlerbedingte dynamische Erscheinungen sind leicht zu analysieren.

Nachteil:
Da für jeden der in der Regel zahlreichen Fehlerrepräsentanten ein Simulationslauf erforderlich ist, ergeben sich erhebliche Rechenzeiten.

— Deduktive Fehlersimulation (Fehlerlistensimulation; [11] bis [14])
Nach jeder Eingangsänderung wird die neue Belegung aller Leitungen durch Simulation ermittelt. Gleichzeitig werden durch einfache algebraische Operationen für die einzelnen Anschlußstifte der Funktionselemente Fehlermengen berechnet.

Vorteil:
Nach [13] und [15] relativ geringe Rechenzeiten, insbesondere für große Schaltungen oder Schaltungen mit geringer Sequentialität.

Nachteil:
Relativ hoher Speicherplatzaufwand gegenüber der parallelen Fehlersimulation nach [15][3])

— Parallele Fehlersimulation [6] [7] [14] und [17]
Hier werden die korrekte Schaltung und eine Anzahl fehlerhafter Schaltungen gleichzeitig (parallel) 2- oder mehrwertig simuliert. Die Anzahl der günstig gleichzeitig simulierbaren „Exemplare" ist abhängig von der Datendarstellung im Wirtsrechner (Wortlänge). Ist n die Zahl der gleichzeitig betrachteten Fehler, so werden die Anschlußstifte in der Schaltung mit $n+1$-dimensionalen „Signalen" belegt, wobei eine festgelegte Komponente den Signalwert der korrekten Schaltung angibt. Fehler werden eingebaut (injiziert), indem der Fehlerwert des fehlerhaften Stiftes an entsprechender Stelle in das betreffende Signal eingetragen wird (s. Bild 5). Es eignen sich die präzise Mittelwert- und die statische Simulation sowie die Simulation mit Einheitsverzögerung. Mehrwertige Signalmodelle sind möglich.

Vorteil:
Durch gleichzeitige Behandlung mehrerer Fehler besteht ein geringer Rechenzeitaufwand. Nach [13] ist dies günstig bei kleineren Schaltungen oder bei hohen Sequentialitätsgrad.

Nachteil:
Nach [13] und [15] ist diese Fehlersimulation rechenzeitintensiver als die deduktive Simulation bei großen Schaltungen infolge mehrfacher Simulationsläufe durch die Zerlegung der Fehlermenge[3]).

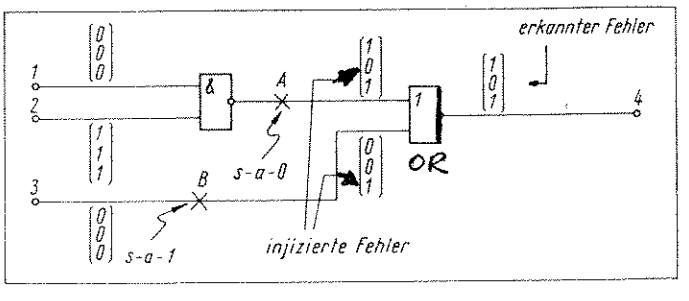
— Konkurrierende Simulation [36] und [37]
Die korrekte Schaltung wird vollständig simuliert. Nur fehlerhafte Schaltungsteile, deren Verhalten hinsichtlich der gerade simulierten Eingangsbelegung von denen der jeweils korrekten Schaltungsteile abweicht, werden „konkurrierend" mit diesen simuliert. Das Verfahren hat gegenüber den vorgenannten mehrere Vorteile: eine ereignisorientierte Simulation ist möglich, und das zeitliche Verhalten der Funktionselemente kann beliebig genau modelliert werden. Nachteilig ist der höhere Implementierungsaufwand für ein solches Verfahren.

4.3. *Programm FELSIM (Parallele Fehlersimulation)*

Untersuchungen haben ergeben, daß sich die Bitkettenstruktur der ESER-Rechner gut für eine Implementierung des Parallelsimulationskonzeptes eignet. Die Ausführungszeit für die stellenweise logische Verknüpfung von Bitketten etwa der Länge 2048

Bild 5. Beispiel bei dem zwei Fehler gleichzeitig (parallel) simuliert werden

Fehler 1: A fest auf 0; Fehler 2: B fest auf 1. Fehler 1 wird ausgewiesen, wenn an den Eingängen 010 anliegt, da sich die 2. Stelle von der 1. Stelle (korrekter Wert) unterscheidet. Fehler 2 wird nicht erkannt.



[1]) Teil I erschien im Heft 12/1979.
Die Literatur ist im Teil I enthalten.
[3]) Nach [6] sind die Aussagen allerdings abhängig von der Implementierung der Algorithmen.

unterscheidet sich nur unwesentlich von der bei Bitketten der Länge 32[4]).

Jeder Leitung (Signalquelle) sowie allen solchen „Signalen" wie Speicherinhalte, alte Taktbelegungen u. dgl. wird eine Bitkette zugeordnet. Die Länge dieser Bitkette ist nicht kleiner als $AR + 1$, wobei AR die Zahl der betrachteten Fehlerrepräsentanten ist. Der Wert $[s_0 s_1 s_2 \ldots s_{AR}]$ der Bitkette ist der entsprechende Signalwert, der folgendermaßen interpretiert wird: Für $i = 1, 2, \ldots, AR$ verfälscht der Fehler i das vorliegende Signal genau dann, wenn $s_i \neq s_0$. Dabei ist s_0 der Wert der korrekten Schaltung. Die s_i können die beiden Werte 0 und 1 annehmen (2-wertige Simulation).

Es ist möglich, drei Schaltungstypen mit FELSIM zu simulieren: kombinatorische, streng synchron-getaktete und asynchrone sequentielle Schaltungen. Bild 6 zeigt die diesen Typen zugeordneten Modelle.

Bei der Erarbeitung des Simulationsmodells (Programm FELMOD) werden die Gatter des kombinatorischen Teils so indiziert, daß ein Gatter mit dem Index i nur von Gattern angesteuert werden kann, deren Indizes kleiner als i sind. Ist ein solches Sortieren nicht vollständig möglich, so liegt eine asynchron-sequentielle Schaltung vor, und es werden solange Schleifen aufgetrennt, bis eine vollständige Sortierung möglich ist. Zu Beginn eines neuen Testschrittes wird zunächst entschieden, ob „getaktet" werden muß, was nur beim synchron-getakteten sequentiellen Fall möglich ist. Die „Taktung" wird so realisiert, daß der Speicherinhalt des durchzuschaltenden Flip-Flops an dessen Ausgang weitergegeben wird und die mit diesem verbundenen Elemente als „erregt" markiert werden. Danach werden die Eingangssignale, die sich im vorliegenden Testschritt änderten, an die entsprechenden Senken weitergegeben und letztere als erregt markiert. Anschließend werden, vom erregten Gatter mit dem niedrigsten Index beginnend, die Ausgangswerte aller erregten Elemente des kombinatorischen Teils ermittelt und die dazugehörigen Senken markiert. Ist der kombinatorische Teil vollständig durchlaufen, so wird im kombinatorischen und im synchron-sequentiellen Fall zum nächsten Testschritt übergegangen. Im asynchron-sequentiellen Fall hingegen wird überprüft, welche Rückführungen ihren Wert änderten. Die Wertänderungen werden realisiert, und der kombinatorische Teil wird erneut durchlaufen. Dies wird solange vollzogen, bis der Ruhezustand eingestellt ist oder eine Maximalzahl von Durchläufen überschritten wurde. Der letzte Fall wird als Oszillation gedeutet.

FELSIM realisiert also eine Null-Verzögerungssimulation. Sie ist relativ rechenzeitgünstig. Dynamische Effekte im Fehlerfall können kaum berücksichtigt werden. Algorithmen mit genauer Zeitablaufmodellierung sind sehr viel rechenzeitaufwendiger.

5. Testsatzgenerierung

5.1. *Fehlermodell*

Im allgemeinen findet das Modell des logischen Einfachfehlers (s-a-Fehler — vgl. Abschn. 4.1.) bei Testgenerierungsprogrammen Verwendung. Dies gilt auch für die hier beschriebenen Verfahren.

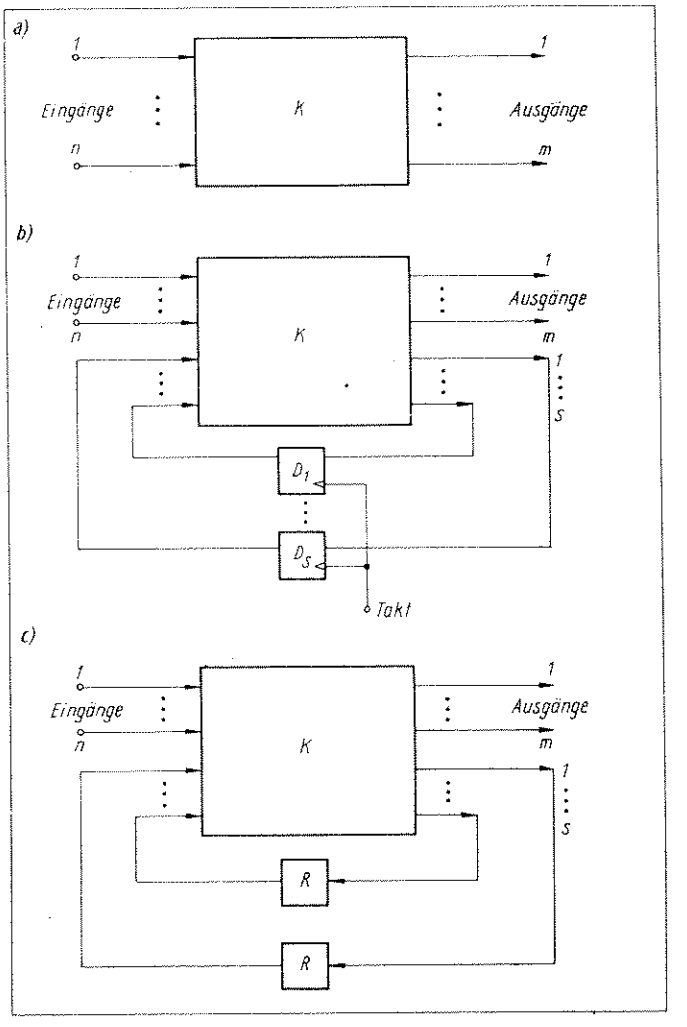


Bild 6

a) Kombinatorische Schaltung
Sie enthält nur die Gattertypen AND, NAND, OR, NOR, EXOR und den Negator.

b) synchron-getaktete Schaltung
Die „D"-Elemente sind getaktete Verzögerungen (ideale Schalter).

c) asynchrone Schaltung
Die „R"-Elemente halten den Signalwert an ihren Ausgang so lange, bis sich K tabilisiert hat. Nach ihrem Durchschalten wird K erneut durchlaufen.

5.2. *Prinzipien*

Bei der Testsatzgenerierung zur Fehlererkennung werden solche Eingangsbedingungen (Tests) ermittelt, die einen Fehler an einer Ausgangsklemme der Schaltung sichtbar machen. Diese Aufgabe ist gleichbedeutend damit, Eingangsbelegungen zu finden, bei denen ein Ausgang wesentlich vom Wert des Signals am Fehlerort abhängt. Bei sequentiellen Schaltungen besteht ein Test für einen Fehler aus einer in fester Reihenfolge angeordneten Menge von Tests, einem Eingabewort (Teiltestfolge).

Von den deterministischen Algorithmen zur Testsatzgenerierung haben sich in der Praxis die durchgesetzt, die Boolesche Differenzen [19] bis [25] und [33] oder den D-Algorithmus [2] [3] [17] [26] [27] [32] [34] und [35] verwenden. Darüber hinaus finden Verfahren der stochastischen Testgenerierung immer breitere Anwendung. Hierunter wird nicht die Prüfung einer Schaltung durch einen hardwaremäßigen realisierten und zufällig arbeitenden Impulsgenerator am Prüfplatz verstanden. Vielmehr werden durch einen Rechner pseudo-zufällige Testvorschläge erzeugt, die durch eine Testanalyse bewertet und von denen durch einen Auswahlalgorithmus verwendbare Tests ausgesondert werden [29] und [30].

[4]) Das Verhältnis der Ausführungszeiten eines logischen Verknüpfungsbefehls, z. B. des Speicher-Oder OC, ist zwar etwa 20:1; dieser Unterschied verschleift sich aber durch längenunabhängige Operationen, wie z. B. Adressierung und Unterprogrammaufruf.

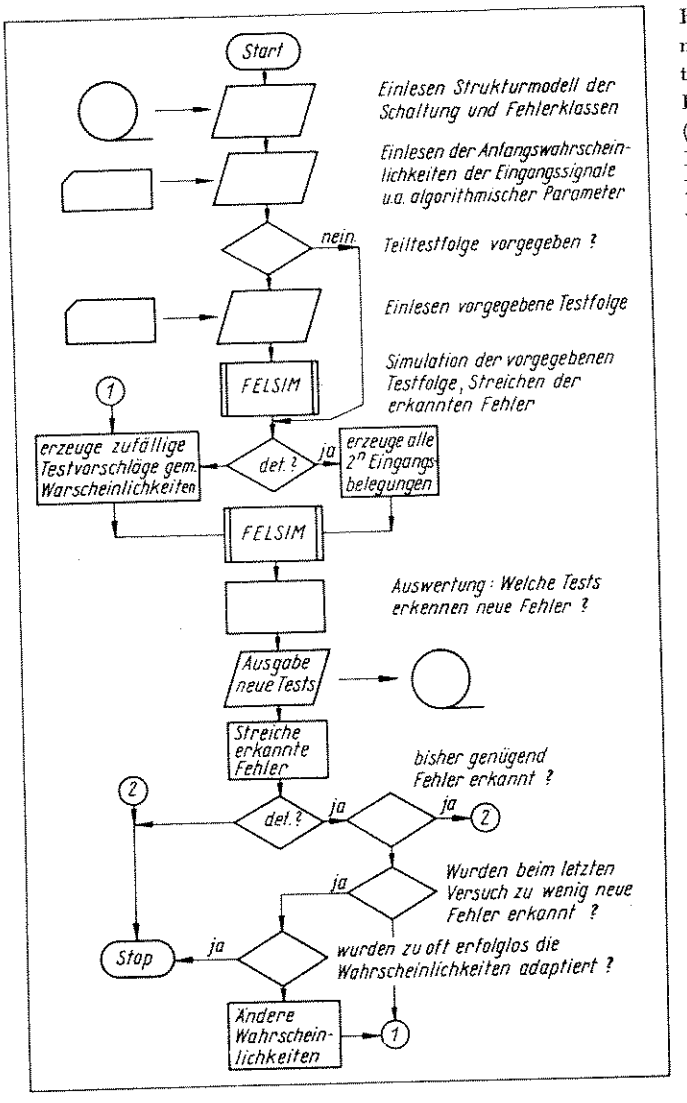


Bild 7. Prinzip des in DICER verwirklichten Algorithmus zur stochastischen Testgenerierung für kombinatorische Schaltungen

Den einzelnen Eingangssignalen werden Auftrittswahrscheinlichkeiten zugeordnet, nach denen die Testvorschläge berechnet werden. Diese Wahrscheinlichkeiten können durch Auswertung der Auswahlergebnisse modifiziert werden (Lernalgorithmus). Es gibt die Möglichkeit, die Wahrscheinlichkeiten der Eingangssignale optimal vorzugeben [28].

5.3. *Programm DICER (stochastische Testgenerierung)*

Da die Auswertung einer vorgegebenen Menge von Testvorschlägen durch deduktive oder parallele Simulation bei günstiger Implementierung klein gegenüber einer deterministisch-algorithmischen Erzeugung einer solchen Menge ist, lohnt es sich, eine gewisse Menge von Tests stochastisch zu ermitteln, bevor die deterministische Testgenerierung eingesetzt wird. Hierdurch werden für etwa 50—70% der Fehler relativ schnell Tests gefunden, so daß die rechenzeitaufwendige Testsatzgenerierung durch TEGAL (vgl. Abschn. 5.4.) nur noch Tests für den Rest der vorgegebenen Fehler zu ermitteln hat. Es erweist sich nicht als sinnvoll, einen nahezu vollständigen Testsatz ausschließlich durch stochastische Testgenerierung zu erzeugen.

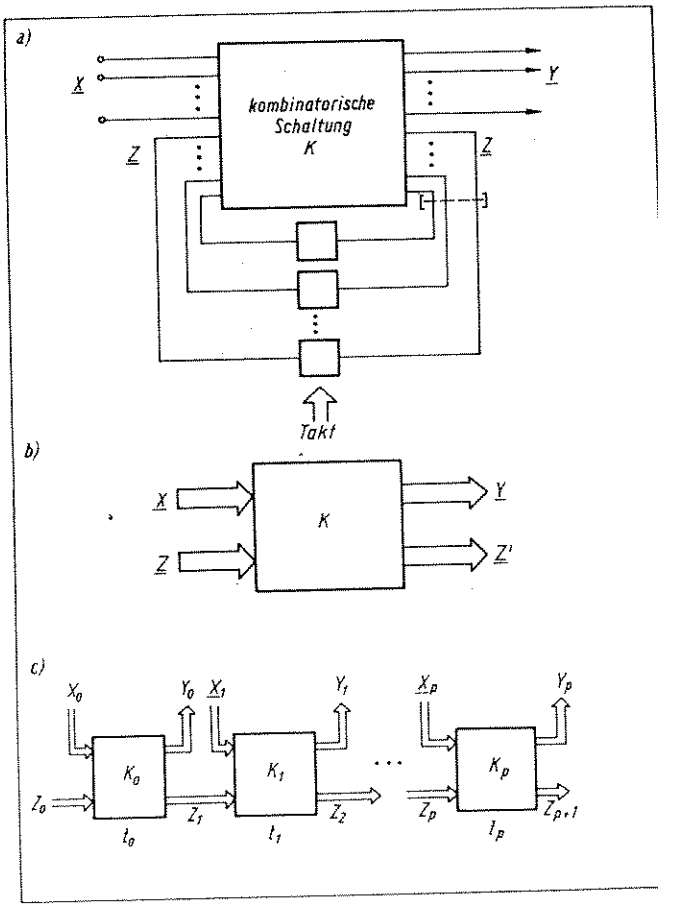
Bild 7 zeigt das Prinzip des in DICER verwirklichten Algorithmus zur stochastischen Testgenerierung für den Fall kombinatorischer Schaltungen.
Das Programm DICER läuft nach dem Programm FELMOD (Bild 1).
Eine ausführliche Beschreibung dieses Programmes und eine Beispielzusammenstellung befinden sich in Vorbereitung und werden veröffentlicht.

5.4. *Programm TEGAL (systematische Testfolgengenerierung)*

Der diesem Programm zugrunde liegende Algorithmus [3] ist eine Weiterentwicklung des D-Algorithmus [2] und [17]. Die durch das Programm FEHLER ermittelten Fehlerrepräsentanten werden durch TEGMOD dem Verfahrensprogramm übergeben. TEGMOD erzeugt außerdem das von TEGAL verwendete Strukturmodell der Schaltung (vgl. Bild 1). Dieses Strukturmodell enthält die Grundgatter AND, NAND, NOR und NOT, sowie den als besonderen Gattertyp modellierten Ausfächerungsknoten FAN-OUT. Speicherelemente werden durch eine kombinatorische Ersatzschaltung und ideale Speicher modelliert, so daß sich eine synchrone sequentielle Schaltung durch das im Bild 8a gezeigte Modell beschreiben läßt[5]). Einem Signaldurchlauf durch die sequentielle Schaltung über p Takte entspricht im Modell ein Durchlauf durch $p+1$ miteinander über die Speicher verkettete gleichartige „Kopien" des kombinatori-

Bild 8. Modellierung einer synchron-getakteten sequentiellen Schaltung für eine Testsatzgenerierung mit TEGAL

a) Ausgangsmodell
b) aus a) gewonnene „Kopie" (kombinatorischer Teil von a)
c) Verkettung von $p+1$ Kopien zur Generierung von Teiltestfolgen der maximalen Länge $p+1$



[5]) Die aus den Speichern in den kombinatorischen Teil führenden Eingänge heißen sekundäre Eingänge, die entsprechenden Ausgänge sekundäre Ausgänge.

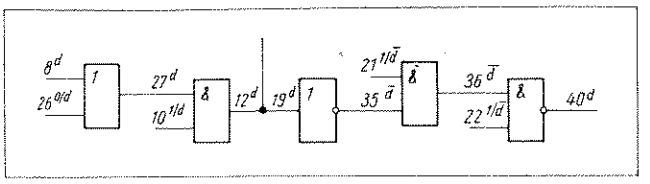


Bild 9. „Sensitivierung" des Signalweges 8-27-12-19-35-36-40 bei TEGAL

mus angewandt [3]. Ersterer belegt die freien Eingänge aller auf dem Signalweg liegenden Gatter so, daß sich das Signal über die Gatter fortpflanzen kann. Durch die Einführung von Bedingungspaaren [3] und [26] kann man eine wesentliche Abschwächung der Bedingungen erreichen, die an die freien Gattereingänge gestellt werden müssen (Sensitivitätsbedingungen). Beispielsweise wird ein Fehlersignal d an einem Eingang eines NAND-Gatters dann weitergeleitet, wenn die übrigen Ein-

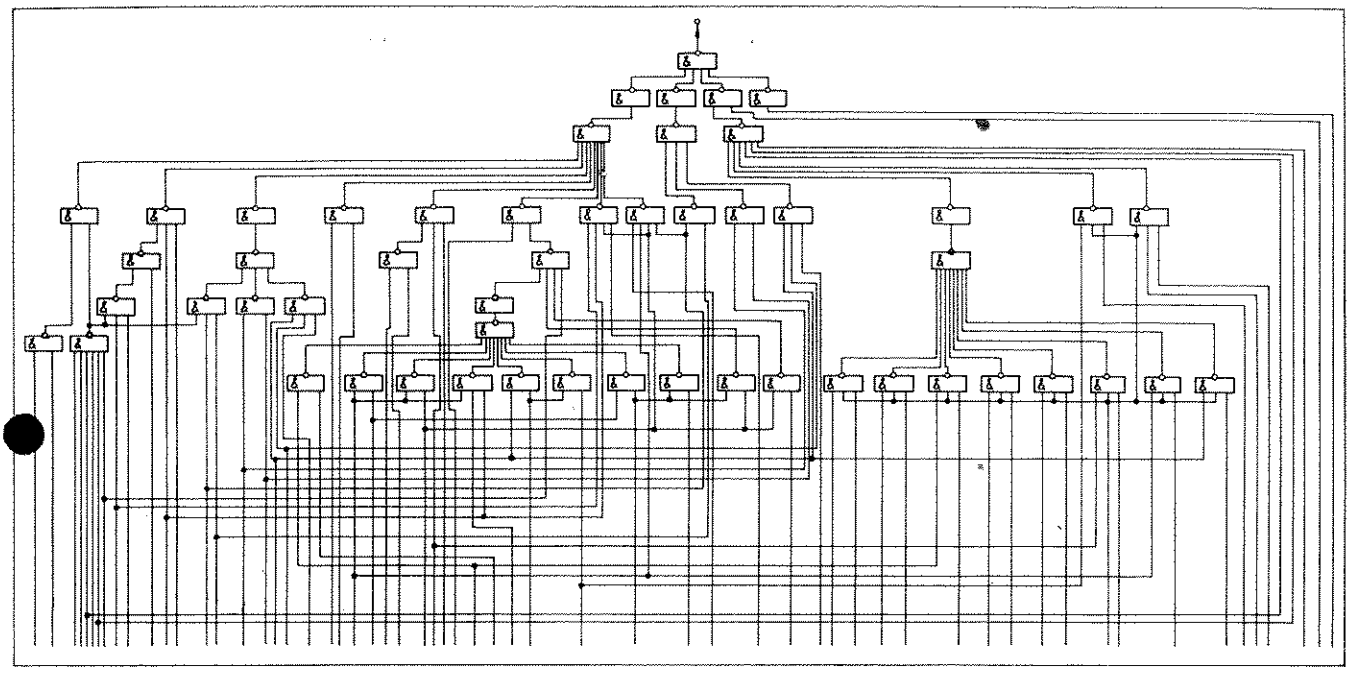


Bild 10. Kombinatorische Schaltung

schen Teils. p wird geeignet vorgegeben ($p \geqq 1$, für kombinatorische Schaltungen: $p = 0$) (Bild 8c).

Ausgehend von der Menge der Fehlerrepräsentanten werden Teiltestfolgen (TTF) erzeugt, die nicht länger als $p + 1$ sind, und mit der bis dahin erzeugten Testfolge verkettet. Alle mit einer TTF erkennbaren Fehler werden durch deduktive Fehlersimulation bestimmt (vgl. Abschn. 4.2.). Die Fehlermenge wird um die erkannten Fehler reduziert. Abgebrochen wird, wenn eine vorgegebene Mindestanzahl von Fehlern erkannt sind oder keine weiteren TTF, die nicht länger als $p + 1$ sind, gefunden werden können. Die nicht erkannten Fehler werden als solche ausgewiesen.

Vor der eigentlichen Testfolgenberechnung kann auf eine gegebene Folge die Fehlersimulation angewandt werden. Dadurch ist es möglich, eine Heimfolge zur Fehlererkennung zu nutzen und den im Fehlerstil durch sie eingestellten Zustand zu ermitteln. Weiterhin kann eine bereits vorliegende, empirisch generierte Testfolge des Entwicklers überprüft und ggf. ergänzt werden (vgl. auch Abschn. 4.3.).

Es schließt sich eine gezielte Berechnung von Teiltestfolgen an (Bild 9). Dafür ist die Kenntnis aller eindimensionalen (Verzweigungen also nicht beinhaltenden) Signalwege durch den kombinatorischen Teil K notwendig. Der Fehlermenge wird ein Fehler entnommen und der kürzestmögliche Signalweg vom Fehlerort zu einem primären Schaltungsausgang gesucht. Gibt es keinen, so wird ein Weg zu einem sekundären Ausgang gesucht und von dort ein Weg zu einem primären Ausgang der folgenden Kopie. Diese Wegverkettung wiederholt sich, bis sie erfolgreich ist oder bis die Anzahl der Teilwege größer als $p + 1$ wird. Auf die einzelnen Wege der Wegekombination werden nun der Sensitivierungs- und der Realisierungsalgorithgänge mit 1 oder ebenfalls mit d belegt werden. Auf diese Art können die durch eine Ausbreitung über den Speicherteil entsprechenden Pseudo-Mehrfachfehler in natürlicher Weise behandelt werden: Es ist z. B. möglich, daß an einem NAND-Eingang infolge des gerade betrachteten Fehlers eine 1 nicht realisiert werden kann, der Fehler sich dennoch über dieses Gatter ausbreitet. Beim klassischen D-Algorithmus [2] scheiterte in einem solchen Fall der Versuch, weitere waren notwendig. Insbesondere jedoch wirkt sich die genannte Abschwächung dadurch aus, daß nur eindimensionale Pfade betrachtet werden müssen, und führt sie gegenüber dem klassischen D-Algorithmus zu einer wesentlichen Reduktion der Verfahrensschritte.

Die so entstehenden Sensitivitätsbedingungen versucht der Realisierungsalgorithmus durch geeignete Belegungen der primären Eingänge zu erfüllen. Um auch die Bedingungen an den sekundären Eingängen zu erfüllen, wird bei der letzten Kopie begonnen; das jeweilige Eingangswort Z_i (vgl. Bild 8c) muß durch eine geeignete Realisierung der Vorkopie K_{i-1} erzeugt werden. Scheitert der Realisierungsversuch, dann wird das Verfahren mit einer weiteren Wegkombination wiederholt.

Eine realisierte TTF wird wieder mit deduktiver Fehlersimulation auf alle erkennbaren Fehler untersucht. Diese werden aus der Fehlermenge gelöscht und ein neuer Fehler zur Bearbeitung ausgewählt. Das Verfahren wird dadurch beschleunigt, daß Fehler, die sich in der vorher berechneten TTF bis in den Speicherteil fortgepflanzt haben, vorrangig behandelt werden. Können neue Fehler nicht mehr mit Teiltestfolgen der vorgegebenen Maximallänge gefunden werden, bricht das Verfahren ab.

Als Ergebnis der Testfolgenberechnung liegt eine Folge von Tests vor, denen als wesentlichen Information für jeden einzel-

nen Test die logische Belegung aller Fehlerorte im fehlerfreien Fall und die Mengen der an den primären Schaltungsausgängen erkennbaren Fehler zugeordnet sind. Daraus sind Angaben über die Fehlererkennung und -lokalisierung in beliebiger Form ableitbar (vgl. Abschnitt 4.3.). Die Form der Prüfunterlagen kann durch spezielle Ausgabeprogramme den jeweiligen Nutzerwünschen angepaßt werden. So können z. B. neben Listen auch Prüflochstreifen zur Steuerung eines Prüfautomaten erzeugt werden.

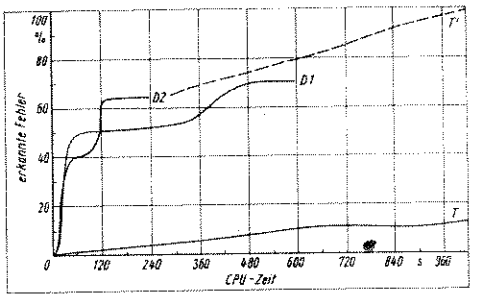


Bild 11. Effektivität der Testgenerierung für Beispiel von Bild 10

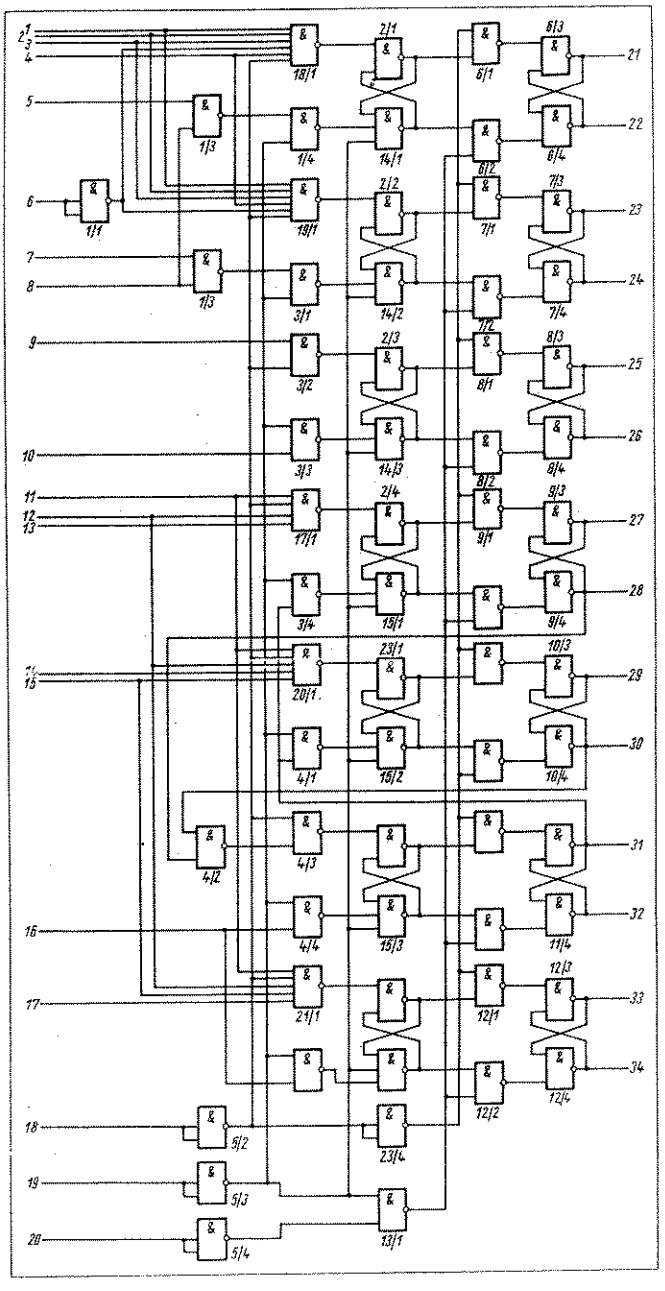
*D*1, *D*2 Stochastische Testgenerierung, *T* systematische Testgenerierung, *T'* Kombination von stochastischer und systematischer Generierung von Tests

6. Beispiel

Für die aus 53 NAND-Gattern bestehende kombinatorische Schaltung von Bild 10 seien Tests zu generieren. Die Schaltung zeichnet sich durch eine große Anzahl von Eingängen (60) aus. 192 Fehlerklassen wurden erzeugt.

Das Programm DICER wurde zweimal unter zufälligen Anfangsbedingungen des Zufallsgenerators auf diese Schaltung angewendet. Im ersten Fall wurden als Zeitschranke 10 min und als Anfangswahrscheinlichkeit für das Auftreten einer 1 im Test für alle Eingänge 0,65 vorgegeben. Nach etwa 8 min CPU-Zeit stochastischer Testgenerierung waren Tests für 71% aller Fehler gefunden. Im zweiten Fall ergab sich durch andere (zufällige) Anfangsbedingungen für den Zufallsgenerator eine andere Kurve (Bild 11). Nach etwa 2 min waren bereits 65% aller Fehler durch Tests abgedeckt. Anhand der resultierenden Effektivitätskurven (Bild 11) ist ersichtlich, daß der Effekt nach einer gewissen Zeit sehr zurückgeht, obwohl gelegentlich noch „Sprünge" auf Grund der Lernvorgänge auftreten.

Bild 12. Asynchron-getaktete Registerschaltung



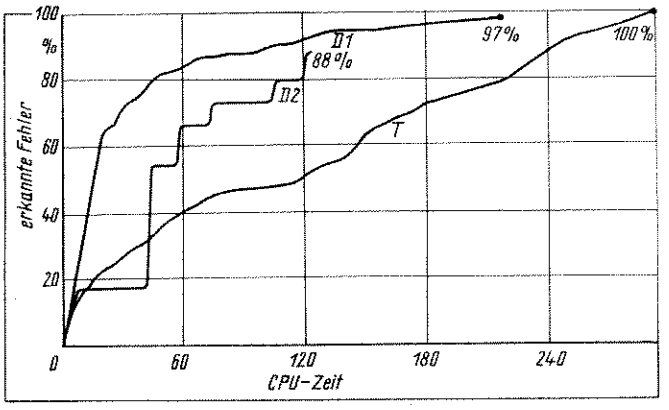


Bild 13. Effektivität der Testgenerierung für die Registerschaltung von Bild 12

*D*1 stochastische Testgenerierung für Originalschaltung, *D* 2 bzw. *T* stochastische bzw. systematische Testgenerierung für vereinfachte, synchron-getaktete Schaltung

Die systematische Testgenerierung durch TEGAL (Kurve T im Bild 11) ist für die Anfangsphase weniger effektiv, aber für den Bereich der Fehler oberhalb 70% durch die im Mittel mehr lineare Zuwachskurve günstiger als DICER. Es bietet sich also für praktische Zwecke eine Kombination der Verfahren (Kurve D2-T') an.

Bild 12 zeigt ein weiteres Beispiel. Es stellt eine kleine asynchrone Registerschaltung mit 20 Eingängen dar, bestehend aus 65 NAND-Gattern.

Für TEGAL wurde diese Schaltung in eine ähnliche, aber synchrongetaktete Schaltung überführt, indem für die nach dem Master-Slave-Prinzip arbeitenden Speicher synchron-getaktete Flip-Flops eingesetzt und für die Steuereingänge 18, 19 und 20 ein synchroner Takt eingeführt wurde. Auf diese Weise war sie dem TEGAL-Algorithmus besser angepaßt.

Die Effektivität der Programme DICER und TEGAL ist für diese Schaltung im Bild 13 dargestellt.

msr 6976