(19) **Bundesrepublik Deutschland**
Deutsches Patent- und Markenamt

(10) **DE 600 30 896 T2** 2007.09.20

(12) # Übersetzung der europäischen Patentschrift

(97) **EP 1 475 643 B1**
(21) Deutsches Aktenzeichen: **600 30 896.0**
(96) Europäisches Aktenzeichen: **04 017 880.8**
(96) Europäischer Anmeldetag: **15.11.2000**
(97) Erstveröffentlichung durch das EPA: **10.11.2004**
(97) Veröffentlichungstag der Patenterteilung beim EPA: **20.09.2006**
(47) Veröffentlichungstag im Patentblatt: **20.09.2007**

(51) Int Cl.[8]: ***G01R 31/3185*** (2006.01)

(30) Unionspriorität:

167446 P	**23.11.1999**	**US**
619985	**20.07.2000**	**US**

(84) Benannte Vertragsstaaten:
AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LI, LU, MC, NL, PT, SE, TR

(73) Patentinhaber:
Mentor Graphics Corp., Wilsonville, Oreg., US

(74) Vertreter:
BOEHMERT & BOEHMERT, 28209 Bremen

(72) Erfinder:
Rajskli, Janusz, West Linn, OR 97068, US; Tyszer, Jerzy, 61-249 Poznan, PL; Kassab, Mark, Wilsonville, OR 97070, US; Mukherjee, Nilanjan, Wilsonville, OR 97070, US

(54) Bezeichnung: **Testmuster-Kompression für eine Testumgebung von integrierten Schaltungen**

Beschreibung

[0001] Diese Erfindung bezieht sich allgemein auf das Testen integrierter Schaltkreise und insbesondere auf die Erzeugung und auf das Geben von Testdaten in Form von Mustern oder Vektoren auf Abtastketten innerhalb eines Schaltkreises im Test.

Hintergrund der Erfindung

[0002] Während integrierte Schaltkreise mit immer höheren Niveaus der Schaltkreisdichte hergestellt werden, werden effiziente Testschemata wichtig, die eine sehr hohe Fehlerüberdeckung sicherstellen, während sie die Testkosten und den zusätzlichen Chip-Platzbedarf minimieren. Allerdings wird es mit herkömmlichen Testparadigmen schwieriger, eine hohe Fehlerüberdeckung mehrerer Typen von Fehlermodellen zu erzielen, während die Komplexität der Schaltkreise weiter zunimmt. Diese Schwierigkeit ergibt sich aus mehreren Gründen. Zunächst haben größere integrierte Schaltkreise ein sehr hohes und weiter steigendes Logik/Anschlußstift-Verhältnis, das einen Testdatenübertragungs-Engpaß bei den Chipanschlußstiften erzeugt. Zweitens erfordern größere Schaltkreise eine ungeheuer große Menge an Testdaten, die dann in einem externen Testgerät gespeichert werden müssen. Drittens erfordert das Geben der Testdaten auf einen großen Schaltkreis eine zunehmend lange Testanwendungszeit. Außerdem kann viertens das gegenwärtige externe Testgerät solche größeren Schaltkreise nicht bei ihrer Betriebsgeschwindigkeit testen.

[0003] Integrierte Schaltkreise werden gegenwärtig unter Verwendung einer Anzahl strukturierter Entwurf-für-Testfähigkeit-Techniken (DFT-Techniken) getestet. Diese Techniken beruhen auf dem allgemeinen Konzept, alle oder einige Zustandsvariablen (Speicherelemente wie Flipflops und Zwischenspeicher) direkt steuerbar und beobachtbar zu machen. Falls dies eingerichtet werden kann, kann ein Schaltkreis, soweit es das Testen von Kombinationsfehlern betrifft, als ein Kombinationsnetz behandelt werden. Die am häufigsten verwendete DFT-Methodik beruht auf Abtastketten. Wie im Patent der Vereinigten Staaten Nr. 4,503,537 gezeigt ist, nimmt sie an, daß während des Testens alle (oder nahezu alle) Speicherelemente zu einem oder zu mehreren Schieberegistern verbunden sind. Ein Schaltkreis, der für den Test entworfen worden ist, hat zwei Betriebsarten: eine Normalbetriebsart und eine Testbetriebsart oder Abtastbetriebsart. In der Normalbetriebsart führen die Speicherelemente ihre regulären Funktionen aus. In der Abtastbetriebsart werden die Speicherelemente zu Abtastzellen, die verbunden sind, um eine Anzahl von Schieberegistern zu bilden, die Abtastketten genannt werden. Diese Abtastketten werden verwendet, um eine Folge von Testmustern in den Schaltkreis zu schieben und um Schaltkreis- oder Testantworten auf die Testmuster herauszuschieben. Die Testantworten werden daraufhin mit fehlerfreien Antworten verglichen, um zu bestimmen, ob der Schaltkreis im Test (CUT) richtig arbeitet.

[0004] Die Abtastentwurfsmethodik hat wegen ihrer einfachen automatischen Testmustererzeugung (ATPG) und Silicium-Austestfähigkeiten umfassende Anwendung erlangt. ATPG-Software-Hilfsmittel sind heute so effizient, daß es möglich ist, Testfolgen (eine Sammlung von Testmustern) zu erzeugen, die eine fast vollständige Fehlerüberdeckung mehrerer Typen von Fehlermodellen einschließlich Hängenbleib-, Übergangs-, Wegverzögerungsfehlern und Überbrückungsfehlern sicherstellen. Wenn sich ein ATPG-Hilfsmittel auf einen besonderen potentiellen Fehler in einem Schaltkreis konzentriert, müssen üblicherweise nur eine kleine Anzahl von Abtastzellen (deterministisch spezifizierte Zellen), z. B. 2–5%, spezifiziert werden, um den besonderen Fehler zu erfassen. Die verbleibenden Abtastzellen in den Abtastketten (zufällig spezifizierte Zellen) werden mit zufälligen Bitwerten gefüllt. Auf diese Weise wird das Muster vollständig spezifiziert, wobei wahrscheinlicher einige zusätzliche Fehler erfaßt werden, und kann in einem Tester gespeichert werden.

[0005] Allerdings sind die Testmuster wegen der Anforderung des zufälligen Füllens stark überspezifiziert. Diese großen Testmuster erfordern umfangreichen Testerspeicher zum Speichern und eine beträchtliche Zeit, um sie von dem Tester auf einen Schaltkreis im Test zu geben. Fig. 1 ist ein Blockschaltplan eines herkömmlichen Systems **18** zum Testen digitaler Schaltkreise mit Abtastketten. In der Abtastbetriebsart gibt ein externes automatisches Testgerät (ATE) oder ein Tester **20** über Abtastketten **26** innerhalb des Schaltkreises eine Folge vollständig spezifizierter Testmuster **22** einzeln auf einen CUT **24**. Daraufhin wird der Schaltkreis unter Verwendung des Testmusters als Eingabe in der Normalbetriebsart betrieben, wobei die Testantwort auf die Testmuster in den Abtastketten gespeichert wird. Während der Schaltkreis wieder in der Abtastbetriebsart ist, wird die Antwort daraufhin zu dem Tester **20** geleitet, der die Antworten, ebenfalls einzeln, mit einer fehlerfreien Referenzantwort **28** vergleicht. Für große Schaltkreise wird dieser Zugang wegen großer Testfolgengrößen und langer Testanwendungszeiten undurchführbar. Es ist berichtet worden, daß die Menge der Testdaten in einem großen Entwurf ein Kilobit pro einzelnes logisches Gatter übersteigen kann. Die wesentliche Beschränkung dieses Zugangs ist, daß er zum Testen eines komplexen Schaltkreises einen teuren, speicheraufwändigen

Tester und eine lange Testzeit erfordert.

[0006] Wie im Patent der Vereinigten Staaten Nr. 4,503,537 gezeigt ist, können diese Zeit- und Speicherbeschränkungen in gewissem Umfang durch Annahme eines Systems des eingebauten Selbsttests (BIST-Systems) überwunden werden. Im BIST ist zusätzliche On-Chip-Schaltungsanordnung enthalten, um Testmuster zu erzeugen, Testantworten zu bewerten und den Test zu steuern. Im herkömmlichen Logik-BIST, wo als Testmuster Pseudozufallsmuster verwendet werden, können 95–96% Überdeckung von Hängenbleibfehlern erzielt werden, sofern Testpunkte zum Adressieren zufallsmusterresistenter Fehler genutzt werden. Durchschnittlich können je 1000 Gatter ein bis zwei Testpunkte erforderlich sein. Im BIST pflanzen sich alle Antworten zu beobachtbaren Ausgaben fort, wobei das Signaturregister bekannt sein muss. Unbekannte Werte verfälschen die Signatur und müssen daher durch zusätzliche Testlogik begrenzt werden. Obgleich Pseudozufallstestmuster einen erheblichen Prozentsatz von Hängenbleibfehlern zu überdecken scheinen, müssen diese Muster durch deterministische Muster ergänzt werden, die sich auf die verbleibenden, zufallsmusterresistenten Fehler konzentrieren. Sehr häufig übersteigt der zum Speichern der Zusatzmuster im BIST erforderliche Testerspeicher 50% des Speichers, der in dem oben beschriebenen deterministischen Zugang erforderlich ist. Eine weitere Beschränkung des BIST ist, daß durch Pseudozufallsmuster andere Typen von Fehlern wie etwa Übergangs- oder Wegverzögerungsfehler nicht effizient behandelt werden. Wegen der Komplexität der Schaltkreise und der im BIST inhärenten Beschränkungen ist es äußerst schwierig, wenn nicht unmöglich, eine Folge spezifizierter Testmuster bereitzustellen, die schwer zu testende Fehler vollständig überdecken.

[0007] Gewichtete Pseudozufallstests sind ein weiteres Verfahren, das verwendet wird, um das Problem der zufallsmusterresistenten Fehler zu behandeln. Im Prinzip erweitert dieser Zugang die Pseudozufallstestmuster-Generatoren, indem er die Wahrscheinlichkeiten der Eingabebits so verschiebt, daß die Tests, die für schwer zu testende Fehler notwendig sind, wahrscheinlicher auftreten. Im Allgemeinen kann ein Schaltkreis allerdings eine sehr große Anzahl von Folgen von Gewichtungen benötigt, wobei für jede Gewichtungsfolge eine Anzahl von Zufallsmustern angelegt werden muß. Obgleich die Menge der Testdaten im Vergleich zu vollständig spezifizierten deterministischen Testmustern üblicherweise verringert ist, steigt somit die resultierende Testanwendungszeit. Darüber hinaus lassen gewichtete Pseudozufallstests immer noch einen Anteil der Fehlerliste unüberdeckt. Einzelheiten von Testsystemen mit gewichteten Zufallsmustern und verwandte Verfahren sind in einer Anzahl von Literaturhinweisen einschließlich der Patente der Vereinigten Staaten Nr. 4,687,998, 4,801,870, 5,394,405, 5,414,716 und 5,612,963 zu finden. Gewichtete Zufallsmuster werden hauptsächlich als eine Lösung zum Komprimieren der Testdaten in dem Tester verwendet. Die Erzeugungs-Hardware scheint zu komplex zu sein, um sie auf dem Chip anzuordnen. Folglich werden die umfangreichen Testdaten außerhalb des Chips erzeugt und müssen durch verhältnismäßig langsame Testerkanäle zu dem Schaltkreis im Test geleitet werden. Die Testanwendungszeit kann effektiv viel länger als die von dem herkömmlichen deterministischen Zugang unter Verwendung von ATPG-Mustern verbrauchte sein.

[0008] Es sind mehrere Verfahren vorgeschlagen worden, um Testdaten zu komprimieren, bevor sie an den Schaltkreis im Test übertragen werden. Sie beruhen auf der Beobachtung, daß der Testkubus (d. h. die Anordnung von Testmusterbits, wie sie innerhalb der Abtastketten eines Schaltkreises im Test gespeichert werden) häufig eine große Anzahl unspezifizierter (unbedeutender) Stellen aufweisen. In B. Koenemann, "LFSR-Coded Test Patterns for Scan Designs", Proc. European Test Conference, S. 237–242 (1991), wurde erstmals ein als Reseeding von Schieberegistern mit linearer Rückkopplung (LFSRs) bekanntes Verfahren vorgeschlagen. Es wird ein n-Bit-LFSR mit einem festen Polynom betrachtet. Seine Ausgabesequenz ist dann durch den Startwert vollständig bestimmt. Somit liefert das rekursive Anwenden der Rückkopplungsgleichungen ein System linearer Gleichungen, das nur von den Startwertvariablen abhängt. Diese Gleichungen können mit den aufeinanderfolgenden Stellen der LFSR-Ausgabesequenz in Verbindung stehen. Folglich kann ein Startwert, der dem tatsächlichen Testmuster entspricht, durch Lösen des Systems linearer Gleichungen bestimmt werden, wobei jede Gleichung eine der spezifizierten Stellen in dem Testmuster repräsentiert. Das Laden des resultierenden Startwerts in das LFSR und sein nachfolgendes Takten erzeugen das gewünschte Testmuster. Allerdings ist ein Nachteil dieses Zugangs, daß der Startwert, der die Inhalte des Testkubus codiert, näherungsweise auf die Größe des LFSR beschränkt ist. Falls der Testkubus mehr spezifizierte Stellen als die Anzahl der Stufen im LFSR hat, kann der Testkubus nicht leicht mit einem Startwert codiert werden. Ein weiterer Nachteil dieses Zugangs ist die Zeit, die er erfordert. Ein Tester kann nicht das LFSR zusammen mit einem Startwert füllen, wenn der LFSR aus dem Startwert ein Testmuster erzeugt. Jeder dieser Schritte muß zu gegenseitig ausschließenden Zeiten erfolgen. Dies macht den Betrieb des Testers sehr ineffizient, d. h., wenn der Startwert seriell in das LFSR geladen wird, arbeiten die Abtastketten nicht; und wenn das Laden der Abtastketten stattfindet, kann der Tester keinen Startwert an das LFSR übertragen.

[0009] C. Fagot, O. Gascuel, P. Girard, C. Landrault, "On Calculating Efficient LFSR Seeds for Built-In Self

Test", European Test Workshop, Proceedings, S. 7–14 (Mai 1999) beschreibt ein weiteres Verfahren zum Berechnen von Startwerten für LFSRs, in dem ein simulationsbasiertes Verfahren verwendet wird, um effizient Startwerte zu berechnen, die zum Erzeugen von LFSR-Testsequenzen mit vorbestimmten Testlängen verwendet werden.

[0010] Ein weiteres Kompressionsverfahren, wie es in S. Hellebrand u. a., "Built-In Test For Circuits With Scan Based On Reseeding of Multiple Polynomial Linear Feedback Shift Registers", IEEE Trans. On Computers, Bd. C-44, S. 223–233 (1995), vorgeschlagen wurde, beruht auf dem Reseeding von Mehrfach-Polynom-LFSRs (MP-LFSRs). In diesem Verfahren wird eine verkettete Gruppe von Testkuben mit einer Anzahl von Bits codiert, die einen Startwert und einen Polynomidentifizierer spezifizieren. Der Inhalt des MP-LFSR wird für jede Testgruppe geladen und muß während der Dekomprimierung jedes Testkubus innerhalb der Gruppe beibehalten werden. Die Realisierung des Dekomprimierers umfaßt das Hinzufügen von zusätzlichen Speicherelementen, um das Überschreiben des Inhalts des MP-LFSR während der Dekomprimierung einer Gruppe von Testmustern zu vermeiden. Eine ähnliche Technik ist in S. Hellebrand u. a., "Pattern generation for a deterministic BIST scheme", Proc. ICCAD, S. 88–94 (1995), diskutiert worden. Durch Annahme des Konzepts von Startwerten variabler Länge, wie es im J. Rajski u. a., "Decompression of test data using variable-length seed LFSRs", Proc. VLSI Test Symposium, S. 426–433 (1995), und in J. Rajski u. a., "Test Data Decompression for Multiple Scan Designs with Boundary Scan", IEEE Trans. On Computers, Bd. C-47, S. 1188–1200 (1998) beschrieben ist, wurde das Reseeding von MP-LFSRs weiter verbessert. Diese Technik hat selbst für Testkuben mit einer stark veränderlichen Anzahl spezifizierter Stellen ein Potential für eine erhebliche Verbesserung der Testdaten-Codierungseffizienz. Dieselben Dokumente schlagen Dekomprimierungstechniken für Schaltkreise mit Mehrfach-Abtastketten und Mechanismen zum Laden von Startwerten in die Dekomprimiererstruktur durch den Boundary-Scan vor. Obgleich dieses Schema die Decodierungsfähigkeit stark verbessert, leidet es weiter an den zwei oben erwähnten Nachteilen: Startwertlängenbeschränkungen und wechselweise ausschließende Zeiten zum Laden des Startwerts und zum Erzeugen von Testmustern daraus.

[0011] Somit leiden die meisten Reseeding-Verfahren bis heute an den folgenden Beschränkungen. Zunächst ist die Codierungsfähigkeit des Reseeding durch die Länge des LFSR beschränkt. Im Allgemeinen ist es sehr schwierig, einen Testkubus zu codieren, der mehr spezifizierte Stellen als die Länge des LFSR besitzt. Zweitens erfolgen das Laden des Startwerts und die Testmustererzeugung daraus in zwei getrennten, nicht überlappenden Phasen. Dies führt zu einer schlechten Nutzung der Testerzeit.

[0012] Ein anderer Zugang zum Verringern der Testanwendungszeit und des Testdatenumfangs ist in I. Hamzaoglu u. a., "Reducing Test Application Time For Full Scan Embedded Cores", Proc. FTCS-29, S. 260–267 (1999), beschrieben. Dieses so genannte Parallel-Seriell-Vollabtastschema unterteilt die Abtastkette in mehrere Bereiche und schiebt über einen einzelnen Abtasteingang in jede Abtastkette dasselbe Testmuster ein. Selbstverständlich darf ein gegebenes Testmuster in einander entsprechenden Zellen in verschiedenen Ketten, die über denselben Eingang geladen werden, keine widersprüchlichen Werte enthalten. Obgleich teilweise spezifizierte Testkuben diese Operationen zulassen können, stützt sich die Leistungsfähigkeit dieses Schemas stark auf die Abtastkettenkonfiguration, d. h. auf die Anzahl der verwendeten Abtastketten und auf die Zuordnung der Speicherelemente zu den Abtastketten. In großen Schaltkreisen ist es unwahrscheinlich, daß eine solche Abbildung irgendeine gewünschte Form annimmt, so daß die Lösung nicht leicht skalierbar ist. Darüber hinaus muß ein Tester, der dieses Schema verwendet, Testmuster verschiedener Abtastkettenlängen behandeln können, ein Merkmal, das für viele Tester unüblich ist.

Zusammenfassung der Erfindung

[0013] Ein Verfahren gemäß der Erfindung wird verwendet, um ein komprimiertes Testmuster zu erzeugen, das auf Abtastketten in einem integrierten Schaltkreis im Test gegeben wird. Das komprimierte Testmuster wird in einem ATE gespeichert und auf Eingangskanäle in einen integrierten Schaltkreis gegeben, der getestet wird. Um die Übertragung von Daten (des komprimierten Musters) von dem ATE zu dem integrierten Schaltkreis zu synchronisieren, wird ein Taktsignal verwendet. Das komprimierte Muster wird in dem integrierten Schaltkreis dekomprimiert, um ein Testmuster zu erhalten, das an Abtastketten übergeben wird, um Fehler innerhalb des integrierten Schaltkreises zu testen. Die Abtastketten enthalten mehrere miteinander gekoppelte Abtastzellen (Speicherelemente), die das Testmuster speichern.

[0014] In einem Aspekt wird das komprimierte Muster unter Verwendung eines Testkubus erzeugt, wobei nur einem Teil der Abtastzellen vorbestimmte Werte zugeordnet werden. Die verbleibenden Abtastzellen in dem Testkubus können ohne Zuordnung bleiben und werden mit einem Pseudozufallsmuster gefüllt, das durch den Dekomprimierer während des Tests erzeugt wird. Somit erzeugt das ATPG-Hilfsmittel Testvektoren, ohne die

"unbedeutenden" Stellen mit Zufallsmustern zu füllen. Es werden symbolische Ausdrücke erzeugt, die mit den Abtastzellen in Verbindung stehen und die eine Funktion von außen angelegter Eingabevariablen sind. Durch Gleichsetzen der symbolischen Ausdrücke mit den den Abtastzellen zugeordneten Werten wird eine Folge von Gleichungen formuliert. Durch Lösen der Folge von Gleichungen unter Verwendung bekannter Techniken wird das komprimierte Muster erzeugt. Um die symbolischen Ausdrücke zu erzeugen, werden den Bits, die auf die Eingangskanäle gegeben werden, Eingabevariablen zugeordnet. Um die symbolischen Ausdrücke als eine lineare Kombination der Eingabevariablen zu erzeugen, wird eine symbolische Simulation des Dekomprimierers verwendet. Daraufhin werden die symbolischen Ausdrücke den Abtastzellen innerhalb der Abtastketten zugeordnet.

[0015] In einem weiteren Aspekt können nach dem Lösen der Gleichungen mit der vorhandenen Folge von Gleichungen inkrementell neue Gleichungen verkettet werden, um einen weiteren Fehler zu testen. Die resultierende neue Folge von Gleichungen kann gelöst werden, falls es eine Lösung gibt. Falls es keine Lösung gibt, werden die zuletzt verketteten Gleichungen gelöscht und wird ein weiterer Fehler ausgesucht. Der Prozeß des inkrementellen Verkettens von Gleichungen durch Auswählen neuer Fehler wird fortgesetzt, bis ein Grenzkriterium erreicht ist wie etwa ein vorbestimmter Grenzwert erfolgloser Versuche erreicht ist oder einer vorbestimmten Anzahl von Bits in dem Testkubus Werte zugeordnet worden sind.

[0016] In einer offenbarten Ausführungsform wird ein Verfahren offenbart, das ein komprimiertes Testmuster zum Testen eines integrierten Schaltkreises berechnet. In dieser Ausführungsform werden symbolische Ausdrücke erzeugt, die mit Abtastzellen innerhalb eines integrierten Schaltkreises in Verbindung stehen. Die symbolischen Ausdrücke sind eine Funktion der Eingabevariablen, die zusammen angelegt werden, wenn die Abtastzellen geladen werden. Es wird ein Testkubus erzeugt, der repräsentativ für ein in die Abtastzellen zu ladendes Testmuster ist. Der erzeugte Testkubus besitzt nur einen Teil der Abtastzellen, denen vorbestimmte Werte zugeordnet sind. Durch Gleichsetzen der zugeordneten Werte in den Abtastzellen mit den symbolischen Ausdrücken wird eine Folge von Gleichungen formuliert. Die Gleichungen werden (z. B. durch Ausführung eines Gauß-Jordan-Eliminierungsverfahrens) gelöst, um ein komprimiertes Testmuster zu erhalten. In einigen Realisierungen wird die Folge von Gleichungen inkrementell mit einer oder mit mehreren Gleichungen verkettet, nachdem die Gleichungen gelöst worden sind. Diese Realisierungen können ferner umfassen: (a) Versuchen, die verkettete Folge von Gleichungen zu lösen; (b) falls der Versuch, die Gleichungen zu lösen, fehlschlägt, Löschen der zuletzt verketteten Gleichungen und Verketten einer oder mehrerer anderer Gleichungen mit der Folge von Gleichungen; (c) falls der Versuch, die Gleichungen zu lösen, erfolgreich ist, inkrementelles Verketten zusätzlicher Gleichungen mit der Folge von Gleichungen; und (d) Wiederholen von (a)–(c), bis ein vorbestimmtes Grenzkriterium erreicht ist. In einigen Realisierungen enthält das Verfahren das Speichern des komprimierten Testmusters in einem ATE und das Übertragen des komprimierten Testmusters an einen integrierten Schaltkreis, der getestet wird, wobei der integrierte Schaltkreis einen Dekomprimierer enthält. In bestimmten Realisierungen umfaßt das Verfahren ferner: (a) Geben des komprimierten Testmusters auf den integrierten Schaltkreis; (b) Dekomprimieren des Testmusters; und (c) Laden der Abtastzellen innerhalb des integrierten Schaltkreises mit dem dekomprimierten Testmuster; (d) wobei die Akte von (a), (b) und (c) im Wesentlichen zusammen stattfinden. Die den Abtastzellen zugeordneten Werte können für irgendeine der offenbarten Realisierungen dieser Ausführungsformen so gewählt werden, daß potentielle Fehler in dem integrierten Schaltkreis getestet werden. Ferner kann der Akt des Erzeugens von symbolischen Ausdrücken enthalten: Zuordnen der Eingabevariablen zu Bits in Eingangskanälen in den integrierten Schaltkreis (wobei die Anzahl der Eingabevariablen größer als die Anzahl der Kanäle ist), Simulieren des Gebens der Eingabevariablen auf einen Dekomprimierer in dem integrierten Schaltkreis und Simulieren, daß der Dekomprimierer ununterbrochen zum Dekomprimieren der Eingabevariablen getaktet wird (wobei während eines oder mehrerer Taktzyklen eine oder mehrere zusätzliche Eingabevariable in den Dekomprimierer injiziert werden), und Erzeugen von Ausgabeausdrücken aus dem Dekomprimierer, die sich aus der Simulation ergeben (wobei die Ausgabeausdrücke jeweils auf den Eingabevariablen beruhen, die injiziert werden, während der Dekomprimierer ununterbrochen getaktet wird). In solchen Realisierungen kann das Verfahren ferner das Zuordnen jedes der Ausgabeausdrücke zu jeder Abtastzelle innerhalb einer Abtastkette in dem integrierten Schaltkreis enthalten. Der Akt des Erzeugens von symbolischen Ausdrücken kann außerdem das Verwenden einer Simulation oder einer mathematischen Darstellung eines Komprimierers enthalten. Der Akt des Erzeugens eines Testkubus kann das Zuordnen von Werten einer vorbestimmten logischen 1 oder einer vorbestimmten logischen 0 zu den Abtastzellen zum Testen eines Fehlers in dem integrierten Schaltkreis enthalten. Darüber hinaus kann der Akt des Formulierens der Folge von Gleichungen das In-Verbindung-Bringen jedes symbolischen Ausdrucks in einer eineindeutigen Beziehung mit einer Abtastzelle und für jede Abtastzelle mit einem vorbestimmten zugeordneten Wert das Gleichsetzen des mit dieser Abtastzelle in Verbindung stehenden symbolischen Ausdrucks mit dem vorbestimmten zugeordneten Wert enthalten.

[0017] In einer weiteren offenbarten Ausführungsform wird eine Vorrichtung offenbart, die ein komprimiertes Testmuster zum Testen eines integrierten Schaltkreises berechnet. In dieser Ausführungsform umfaßt die Vorrichtung: Mittel zum Erzeugen von symbolischen Ausdrücken, die mit Abtastzellen innerhalb des integrierten Schaltkreises in Verbindung stehen, wobei die symbolischen Ausdrücke mit dem Laden eines Dekomprimierers mit von außen angelegten Eingabevariablen zusammen, während der Dekomprimierer ununterbrochen dekomprimiert, in Verbindung stehen; Mittel zum Erzeugen eines Testkubus, wobei nur einem Teil der Abtastzellen vorbestimmte Werte zugeordnet werden; Mittel zum Formulieren einer Folge von Gleichungen durch Gleichsetzen der zugeordneten Werte in den Abtastzellen mit den symbolischen Ausdrücken; und Mittel zum Lösen der Gleichungen, um das komprimierte Testmuster zu erhalten.

[0018] In einer weiteren Ausführungsform wird ein Verfahren offenbart, das ein komprimiertes Testmuster zum Testen eines integrierten Schaltkreises berechnet. In dieser Ausführungsform werden unter Verwendung einer Simulation symbolische Ausdrücke erzeugt. Die symbolischen Ausdrücke stehen mit simulierten Abtastzellen eines integrierten Schaltkreises in Verbindung und sind eine Funktion simulierter Eingabevariablen, die zusammen auf einen simulierten Dekomprimierer gegeben werden, der mit den Abtastzellen gekoppelt ist, während der Dekomprimierer ununterbrochen dekomprimiert. Es wird ein Testkubus erzeugt, bei dem nur einem Teil der Abtastzellen vorbestimmte Werte zugeordnet sind. Durch Gleichsetzen der zugeordneten Werte in den Abtastzellen mit den symbolischen Ausdrücken wird eine Folge von Gleichungen formuliert. Um das komprimierte Testmuster zu erhalten, werden die Gleichungen gelöst. In bestimmten Realisierungen kann der Dekomprimierer die simulierten Abtastzellen zusammen laden, wenn die Eingabevariablen auf den Dekomprimierer gegeben werden.

[0019] Diese und weitere Aspekte und Merkmale der Erfindung werden im Folgenden anhand der beigefügten Zeichnung beschrieben.

Beschreibung von bevorzugten Ausführungsbeispielen

[0020] Die Erfindung wird im folgenden anhand von Ausführungsbeispielen unter Bezugnahme auf Figuren einer Zeichnung näher erläutert. Hierbei zeigen:

[0021] Fig. 1 ein Blockschaltplan eines herkömmlichen Systems zum Testen digitaler Schaltkreise mit Abtastketten;

[0022] Fig. 2 ein Blockschaltplan eines Testsystems gemäß der Erfindung zum Testen digitaler Schaltkreise unter Verwendung eines ATE;

[0023] Fig. 3 ist ein Blockschaltplan eines Dekomprimierers gemäß der Erfindung, der einen linearen endlichen Automaten (LFSM) und Phasenschieber enthält;

[0024] Fig. 4 zeigt ausführlicher eine erste Ausführungsform des Dekomprimierers aus Fig. 3, der mit Mehrfach-Abtastketten gekoppelt;

[0025] Fig. 5 ein Ablaufplan eines Verfahrens zum Komprimieren eines Testmusters;

[0026] Fig. 6 ein Ablaufplan eines Verfahrens zum Erzeugen von symbolischen Ausdrücken, die beim Komprimieren eines Testmusters verwendet werden;

[0027] Fig. 7 zeigt ein Beispiel von symbolischen Ausdrücken, die unter Verwendung des Verfahrens aus Fig. 6 erzeugt werden;

[0028] Fig. 8 zeigt eine weitere Ausführungsform der vorliegenden Erfindung mit Eingabevariablen, die über Eingangskanäle auf den integrierten Schaltkreis gegeben werden;

[0029] Fig. 9 zeigt eine weitere Ausführungsform eines Dekomprimierers;

[0030] Fig. 10 zeigt eine abermals weitere Ausführungsform des Dekomprimierers;

[0031] Fig. 11 ein Ablaufplan eines Verfahrens zum inkrementellen Verketten von symbolischen Ausdrücken für die Komprimierung.

[0032] Die Verfahren zum Kompaktieren von Testmustern, wie sie hier gezeigt und beschrieben sind, werden als Software realisiert, die in einem computerlesbaren Medium gespeichert ist und auf einem Universalcomputer ausgeführt wird. Zum Beispiel kann die Erfindung in Hilfsmitteln für den computergestützten Entwurf realisiert werden. Der Klarheit halber werden nur jene Aspekte der Software beschrieben, die für die Erfindung von Belang sind; Produkteinzelheiten, die im Gebiet gut bekannt sind, werden weggelassen. Aus demselben Grund wird die Computer-Hardware nicht ausführlicher beschrieben. Somit ist die Erfindung selbstverständlich nicht auf irgendeine spezifische Computersprache, auf irgendein spezifisches Computerprogramm oder auf irgendeinen spezifischen Computer beschränkt.

[0033] Fig. 2 ist ein Blockschaltplan eines Systems **30** gemäß der Erfindung zum Testen digitaler Schaltkreise mit Abtastketten. Das System enthält einen Tester **21** wie etwa ein externes automatisches Testgerät (ATE) und einen Schaltkreis **34**, der als alles oder als einen Teil davon einen Schaltkreis im Test (CUT) **24** enthält. Der Tester **21** stellt aus einer Ablage eine Folge komprimierter Testmuster **32** von Bits, ein Muster nach dem anderen, über Eingangskanäle **40** für den Schaltkreis **34** wie etwa einen IC bereit. Ein komprimiertes Muster, wie es beschrieben wird, enthält weitaus weniger Bits als ein herkömmliches unkomprimiertes Testmuster. Ein komprimiertes Muster braucht lediglich genug Informationen zu erhalten, um deterministisch spezifizierte Bits wiederherzustellen. Folglich hat ein komprimiertes Muster üblicherweise 2% bis 5% der Größe eines herkömmlichen Testmusters und erfordert für die Speicherung viel weniger Testspeicher als herkömmliche Muster. Außerdem erfordern komprimierte Testmuster viel weniger Zeit zum Übertragen von einem Tester zu einem CUT **24**.

[0034] Im Unterschied zu den oben beschriebenen früheren Reseeding-Techniken werden die komprimierten Testmuster **32** kontinuierlich ohne Unterbrechung von dem Tester **21** an die Abtastketten **26** innerhalb des CUT **24** geliefert. Während das komprimierte Testmuster durch den Tester **21** an die Eingangskanäle eines Dekomprimierers **36** innerhalb des Schaltkreises **34** geliefert wird, dekomprimiert der Dekomprimierer das komprimierte Muster zu einem dekomprimierten Muster von Bits. Daraufhin wird das dekomprimierte Testmuster auf die Abtastketten **26** gegeben. Dieses Geben wird üblicherweise ausgeführt, während das komprimierte Testmuster an den Schaltkreis **34** geliefert wird. Nachdem die Schaltkreislogik innerhalb des CUT **24** mit einem dekomprimierten Testmuster in den Abtastketten **26** getaktet worden ist, wird die Testantwort auf dieses Muster in den Abtastketten erfasst und für den Vergleich mit den darin gespeicherten komprimierten fehlerfreien Referenzantworten **41** an den Tester **21** übertragen.

[0035] In einer typischen Konfiguration hat der Dekomprimierer **36** einen Ausgang pro Abtastkette **26**, während es mehr Abtastketten als Eingangskanäle in den Dekomprimierer gibt. Allerdings sind andere Konfigurationen, in denen die Dekomprimiererausgänge kleiner oder gleich den Eingangskanälen sind, ebenfalls möglich. Der Dekomprimierer erzeugt in einer gegebenen Zeitdauer an seinen Ausgängen eine größere Anzahl dekomprimierter Bits als die Anzahl komprimierter Musterbits, die er während derselben Zeitdauer empfängt. Dies ist der Akt der Dekomprimierung, durch den der Dekomprimierer **36** eine größere Anzahl von Bits erzeugt, als an ihn in einer gegebenen Zeitdauer geliefert werden.

[0036] Um den Datenumfang der Testantwort und die Zeit zum Senden der Antwort an den Tester zu verringern, kann der Schaltkreis **34** Mittel zum Komprimieren der Testantwort, die von den Abtastketten **26** gelesen wird, enthalten. Eine Struktur zum Erzeugen einer solchen Komprimierung sind einer oder mehrere räumliche Kompaktierer **38**. Die durch die Kompaktierer **38** erzeugten komprimierten Testantworten werden daraufhin einzeln mit komprimierten Referenzantworten **40** verglichen. Falls eine Referenzantwort nicht an eine tatsächliche Antwort angepaßt ist, wird ein Fehler erfaßt.

[0037] Das Liefern eines komprimierten Testmusters an einen Schaltkreis, seine Dekomprimierung zu einem dekomprimierten Testmuster und das Geben des dekomprimierten Testmusters auf die Abtastketten werden synchron, kontinuierlich und im Wesentlichen zusammen ausgeführt. Allerdings kann die Rate, mit der jeder Schritt stattfindet, variieren. Auf Wunsch können alle Schritte synchron mit einer gleichen Taktrate ausgeführt werden. Andererseits können die Schritte mit verschiedenen Taktraten ausgeführt werden. Falls die Schritte mit der gleichen Taktrate ausgeführt werden oder falls die komprimieren Testmuster mit einer höheren Taktrate als der, mit der die dekomprimieren Testmuster auf die Abtastketten gegeben werden, geliefert und dekomprimiert werden, übersteigt die Anzahl der Ausgaben des Dekomprimierers **36** und der in Verbindung stehenden Abtastketten wie in Fig. 2 die Anzahl der Eingangskanäle des Dekomprimierers. In diesem ersten Fall wird die Dekomprimierung dadurch erzielt, daß mehr Dekomprimiererausgänge als Eingangskanäle bereitgestellt werden. Falls die komprimierten Testmuster mit einer niedrigeren Taktrate geliefert und mit einer höheren Taktrate dekomprimiert und auf die Abtastketten gegeben werden, kann die Anzahl der Ausgänge und der in Verbindung stehenden Abtastketten die gleiche wie, kleiner oder größer als die Anzahl der Eingangskanäle sein. In

diesem zweiten Fall wird die Dekomprimierung dadurch erreicht, daß die dekomprimierten Testmusterbits mit einer höheren Taktrate erzeugt werden als der, mit der die komprimierten Testmusterbits geliefert werden.

[0038] Fig. 3 ist ein Blockschaltplan eines Dekomprimierers gemäß der Erfindung. Der Dekomprimierer **36** umfaßt einen linearen endlichen Automaten (LFSM) **46**, der auf Wunsch über seine Abgriffe **48** mit einem Phasenschieber **50** gekoppelt ist. Der LFSM liefert höchst linear unabhängige Testmuster über den Phasenschieber an die Eingänge zahlreicher Abtastketten in dem CUT **24**. Der LFSM kann auf der Grundlage der kanonischen Formen von Schieberegistern mit linearer Rückkopplung, Zellularautomaten oder transformierten LFSRs, die durch Anwendung einer Anzahl von m-Sequenz-erhaltenden Transformationen erhalten werden können, aufgebaut sein. Die Ausgabe des LFSM wird auf den Phasenschieber gegeben, der sicherstellt, daß sich die dekomprimierten Muster, die innerhalb der Mehrfach-Abtastketten **26** zu irgendeiner gegebenen Zeit vorhanden sind, im Muster nicht überlappen (d. h. phasenverschoben sind).

[0039] Das hier beschriebenen Konzept der Dekomprimierung im kontinuierlichen Fluß stützt sich auf die oben erwähnte Tatsache, daß bei deterministischen Testmustern üblicherweise nur zwischen 2 bis 5% der Bits deterministisch spezifiziert sind, während die verbleibenden Bits während der Testmustererzeugung zufällig aufgefüllt werden. Als Testkuben werden Testmuster mit teilweise spezifizierten Bitstellen definiert, von denen ein Beispiel unten in Tabelle 2 erscheint. Diese teilweise spezifizierten Testkuben werden so komprimiert, daß der Testdatenumfang, der extern gespeichert werden muß, erheblich verringert ist. Je kleiner die Anzahl spezifizierter Bits in einem Testkubus ist, desto besser ist die Fähigkeit zum Codieren der Informationen zu einem komprimierten Muster. Die Fähigkeit zum Codieren von Testkuben zu einem komprimierten Muster wird genutzt, indem wenige den Schaltkreis im Test ansteuernde Dekomprimierer-Eingangskanäle vorhanden sind, die durch den Tester als virtuelle Abtastketten betrachtet werden. Allerdings sind die Speicherelemente des tatsächlichen CUT **24** zu einer großen Anzahl realer Abtastketten verbunden. Unter diesen Umständen kann den Schaltkreis sogar ein preiswerter Tester extern ansteuern, der wenige Abtastkanäle und einen ausreichend kleinen Speicher zum Speichern von Testdaten besitzt.

[0040] Fig. 4 zeigt den Dekomprimierer ausführlicher. Der LFSM ist in einem achtstufigen Typ-1-LFSR **52** implementiert, das ein primitives Polynom $h(x) = x^8 + x^4 + x^3 + x^2 + 1$ realisiert. Der Phasenschieber **50**, der in einer Anzahl von EXKLUSIV-ODER-Gattern implementiert ist, steuert acht Abtastketten **26** an, die jeweils acht Abtastzellen haben (wobei die veranschaulichten Abtastzellen Speicherelemente sind, die jeweils 1 Bit an Informationen speichern). Die Struktur des Phasenschiebers wird in der Weise gewählt, daß ein wechselweiser Abstand zwischen seinen Ausgangskanälen C0–C7 wenigstens acht Bit beträgt, wobei alle Ausgangskanäle durch 3-Eingangs-(Abgriff-)-EXKLUSIV-ODER-Funktionen mit den folgenden Formen angesteuert werden:

$C_0 = s_4 \oplus s_3 \oplus s_1$
$C_1 = s_7 \oplus s_6 \oplus s_5$
$C_2 = s_7 \oplus s_3 \oplus s_2$
$C_3 = s_6 \oplus s_1 \oplus s_0$
$C_4 = s_4 \oplus s_2 \oplus s_1$
$C_5 = s_5 \oplus s_2 \oplus s_0$
$C_6 = s_6 \oplus s_5 \oplus s_3$
$C_7 = s_7 \oplus s_2 \oplus s_0$

Tabelle 1

wobei C_i der i-te Ausgangskanal ist und s_k die k-te Stufe des LFSR angibt. Es wird angenommen, daß das LFSR jeden Taktzyklus über seine zwei Eingangskanäle **37a**, **37b** und Eingangsinjektoren **48a**, **48b** (EXKLUSIV-ODER-Gatter) zu der zweiten und zu der sechsten Stufe des Registers gespeist wird. Die am Kanal **37a** empfangenen Eingabevariablen "a" (komprimierte Testmusterbits) sind mit geradzahligen Indizes bezeichnet (a_0, a_2, a_4, ...) und die Variablen "a", die am Kanal **37b** empfangen werden, sind mit ungeradzahligen Indizes (a_1, a_3, a_5, ...) bezeichnet. Wie im Folgenden anhand von Fig. 5 weiter beschrieben wird, können bei Behandlung dieser externen Variablen als boolesche Variablen alle Abtastzellen konzeptionell mit symbolischen Ausdrücken gefüllt werden, die lineare Funktionen der Eingabevariablen sind, die durch den Tester **21** in das LFSR **52** injiziert werden. Ausgehend von dem Rückkopplungspolynom, dem Phasenschieber **50**, dem Ort der Injektoren **48a**, b sowie einer zusätzlichen Anfangsperiode von vier Taktzyklen, während der nur dem LFSR Testdaten zugeführt werden, kann der Inhalt jeder Abtastzelle innerhalb der Abtastketten **26** in Fig. 4 logisch bestimmt werden.

[0041] Fig. 5 zeigt einen Ablaufplan eines Verfahrens zum Komprimieren eines Testmusters. In einem ersten Prozeßblock **60** werden symbolische Ausdrücke erzeugt, die mit Abtastzellen in Verbindung stehen. Die erzeugten symbolischen Ausdrücke beruhen auf einer symbolischen Simulation des Dekomprimierers mit einem kontinuierlichen Strom externer Eingabevariablen, die an die Eingangskanäle in den Dekomprimierer geliefert werden. Zusammen, wenn die Eingabevariablen an den Dekomprimierer geliefert werden, ist der Dekompri-

mierer in seiner Normalbetriebsart, in der die Eingabevariablen dekomprimiert werden. Üblicherweise werden außerdem zusammen mit dem Dekomprimierer gekoppelte Abtastketten geladen, während die Eingabevariablen auf den Dekomprimierer gegeben werden. Fig. 6 liefert weitere Einzelheiten über eine Technik zum Erzeugen der symbolischen Ausdrücke und wird durch ein spezifisches Beispiel gefolgt.

[0042] Im Prozeßblock **62** (Fig. 5) wird ein Testkubus erzeugt. Der Testkubus ist ein deterministisches Muster von Bits, in dem jedes Bit einer Abtastzelle in den Abtastketten entspricht. Einigen der Abtastzellen sind Werte zugeordnet (z. B. eine logische 1 oder eine logische 0), während andere Abtastzellen "unbedeutend" sind. Diejenigen Abtastzellen, denen Werte zugeordnet sind, werden in der Komprimierungsanalyse verwendet und in dem komprimierten Testmuster dargestellt. Die verbleibenden Abtastzellen, die "unbedeutend" sind, brauchen in dem komprimierten Testmuster nicht dargestellt zu werden und werden mit Pseudozufallswerten gefüllt, die durch den Dekomprimierer erzeugt werden.

[0043] Im Prozeßblock **64** werden durch Gleichsetzen der im Prozeßblock **60** erzeugten symbolischen Ausdrücke mit den Abtastzellen im Prozeßblock **62** zugeordneten Werten mehrere Gleichungen formuliert. Üblicherweise werden bei der Formulierung der Gleichungen nur die Abtastzellen verwendet, denen Werte zugeordnet sind. Unter Verwendung einer verringerten Anzahl von Gleichungen kann die Komprimierung des Testkubus maximiert werden. Im Prozeßblock **66** werden die Gleichungen unter Verwendung bekannter Techniken wie etwa der Gauß-Jordan-Eliminierung gelöst.

[0044] Fig. 6 liefert weitere Einzelheiten über die Erzeugung der symbolischen Ausdrücke. Die Erzeugung der symbolischen Ausdrücke wird üblicherweise in einer simulierten Umgebung ausgeführt, obgleich mathematische Analysis ebenfalls verwendet werden kann. Im Prozeßblock **70** wird der Dekomprimierer zurückgesetzt. Zum Beispiel werden in dem Dekomprimierer aus Fig. 4 die Speicherelemente 0–7 in dem LFSR **52** auf ein logisches Tief zurückgesetzt. Im Prozeßblock **72** werden Eingabevariablen Bits in den Eingangskanälen zugeordnet. Zum Beispiel kann jedem Eingangskanal während eines oder mehrerer Taktzyklen eine andere Eingabevariable zugeordnet werden. Die Eingabevariablen werden auf den Dekomprimierer gegeben und durchlaufen in Übereinstimmung mit der Dekomprimiererlogik zyklisch die Speicherelemente innerhalb des Dekomprimierers. Die Ausgaben des Dekomprimierers als Funktion der Eingabevariablen werden unter Verwendung symbolischer Simulation bestimmt (Prozeßblock **74**). Insbesondere wird die symbolische Simulation des Dekomprimierers dadurch ausgeführt, daß der Dekomprimierer zusammen damit, daß die Eingabevariablen in den Dekomprimierer injiziert werden, in seiner Normalbetriebsart (d. h. dekomprimierend) betrieben wird. Außerdem gibt der Dekomprimierer zusammen mit der Injektion der Eingabevariablen, die dazu verwendet werden, Abtastzellen innerhalb der Abtastketten zu füllen, dekomprimierte Daten aus. Im Prozeßblock **76** wird jeder Ausgabeausdruck einer entsprechenden Abtastzelle innerhalb einer Abtastkette zugeordnet. Das heißt, die Ausdrücke und die Abtastzellen werden in Verbindung gebracht.

[0045] Somit werden für eine gegebene Struktur eines Dekomprimierers für jede Abtastzelle mehrere symbolische Ausdrücke als lineare Kombinationen der Eingabevariablen gebildet. In diesem Fall ist ein linearer Ausdruck durch die Folge beteiligter Variablen charakterisiert. Jeweils in einem oder in mehreren Taktzyklen des Dekomprimierers (je nach dem Taktungsschema) wird eine neue Folge von k Eingabevariablen, wobei k die Anzahl externer Eingänge ist, über die EXKLUSIV-ODER-Gatter in den LFSM geladen. Im Ergebnis kann jedes Speicherelement des LFSM mit linearen Kombinationen bereits injizierter Variablen in Verbindung stehen. Da der Dekomprimierer zu Beginn für jedes neue Testmuster zurückgesetzt wird, sind die entsprechenden Ausdrücke, die mit den Speicherelementen des LFSM in Verbindung stehen, leer. In jedem Zyklus wird auf der Grundlage der Ausdrücke anderer Speicherelemente und Eingangskanäle, die Quellen genannt werden, die das gegebene Speicherelement speisen, ein neuer mit einem gegebenen Speicherelement, das ein Zielelement eines LFSM genannt wird, in Verbindung stehender Ausdruck berechnet. Die besonderen Formen dieser Ausdrücke hängen von der Funktionalität, Struktur oder inneren Konnektivität des LFSM sowie von den Orten der Startwertvariablen-Injektionsstellen ab.

[0046] Das gleiche Prinzip der Erzeugung von symbolischen Ausdrücken betrifft die Ausgaben eines Phasenschiebers, der eine lineare Funktion der LFSM-Speicherelemente realisiert. Dadurch, daß in dem Galois-Feld Ausdrücke, die mit jenen Stufen des LFSM in Verbindung stehen, die über die EXKLUSIV-ODER-Funktion zum Ansteuern aufeinanderfolgender Ausgaben des Phasenschiebers verwendet werden, insgesamt modulo 2 miteinander addiert werden, werden für alle Ausgangskanäle des Phasenschiebers ähnliche symbolische Ausdrücke erzeugt. Die resultierenden Ausdrücke werden nachfolgend mit aufeinanderfolgenden Zellen von Abtastketten verknüpft, die durch diese Dekomprimiererausgaben angesteuert werden. Die Verknüpfung erfolgt zu Zeitpunkten, die genau denen des Schiebens der Abtastketten entsprechen. Falls der Dekomprimierer und die Abtastketten mit der gleichen Frequenz arbeiten, werden die Ausgabeausdrücke für jeden Zyklus des De-

komprimiererbetriebs berechnet. Falls die Abtastketten mit einer Frequenz arbeiten, die ein Teil derer des Dekomprimierers ist, werden die Dekomprimierergleichungen in jedem Taktzyklus berechnet, wobei aber die Abtastzellengleichungen nur für jene Zyklen berechnet werden, die einer Verschiebung entsprechen. Dieser Betrieb ermöglicht eine höhere Übertragungsrate zu dem Dekomprimierer, während der Leistungsverlust in dem Schaltkreis im Test verringert wird. Um eine unabhängige Verschiebungsoperation zu ermöglichen, können die Abtastketten weiter in mehrere Gruppen mit getrennten Schiebetakten unterteilt werden. In diesem Fall imitiert die Berechnung von symbolischen Ausdrücken genau den Schaltkreisbetrieb.

[0047] Fig. 7 gibt die Ausdrücke für die 64 Abtastzellen in Fig. 4, wobei die in Fig. 4 von 0 bis 7 numerierten Abtastketten den in Fig. 4 identifizierten Abtastketten C7, C1, C6 ... entsprechen. Die Ausdrücke für jede Abtastkette in Fig. 4 sind in der Reihenfolge aufgeführt, in der die Informationen in die Kette geschoben werden, d. h. der oberste Ausdruck repräsentiert die zuerst hereingeschobenen Daten. Die Eingabevariablen a_0, a_2, a_4, ... sind dem Eingangskanal **37a** zugeordnet, während a_1, a_3, a_5, ... **37b** zugeordnet sind. Ausgehend von der Dekomprimiererlogik werden die Ausgaben des Dekomprimierers wie in Fig. 7 gezeigt als eine Funktion der Eingabevariablen bestimmt. In dem veranschaulichten Beispiel wird in jedem Taktzyklus die Ausgabe des Dekomprimierers in die Abtastketten geladen. Folglich steht mit jeder Abtastzelle eine Gleichung in Verbindung.

[0048] Entweder vor oder nach der Erzeugung der symbolischen Ausdrücke wird ein Testkubus erzeugt. Der Testkubus ist ein deterministisches Muster, das auf dem getesteten Fehler beruht. Es wird angenommen, daß der Dekomprimierer **36** in Fig. 4 ein Testmuster erzeugen soll, das auf dem folgenden teilweise spezifizierten Testkubus in Tabelle 2 beruht (horizontal sind hier die Inhalte der acht Abtastketten gezeigt, wobei die linke äußere Spalte die Informationen repräsentiert, die zuerst in die Abtastketten geschoben werden):

x x x x x x x x Abtastkette 0
x x x x x x x x Abtastkette 1
x x x x 1 1 x x Abtastkette 2
x x 0 x x 1 x x Abtastkette 3
x x x x 0 x x 1 Abtastkette 4
x x 0 x 0 x x x Abtastkette 5
x x 1 x 1 x x x Abtastkette 6
x x x x x x x x Abtastkette 7

Tabelle 2

[0049] Die Variable x bezeichnet eine "Unbedeutend"-Bedingung. Das komprimierte Testmuster wird dadurch bestimmt, daß die Gleichungen aus Fig. 7 den zugeordneten Bits in dem Testkubus gleichgesetzt werden. Daraufhin kann durch Lösen des folgenden Systems von zehn Gleichungen aus Fig. 7 unter Verwendung irgendeiner Anzahl gut bekannter Techniken wie etwa Gauß-Jordan-Eliminierungstechniken ein entsprechendes komprimiertes Testmuster bestimmt werden. Die ausgewählten Gleichungen entsprechen den deterministisch spezifizierten Bits:

$$a_2 \oplus a_6 \oplus a_{11} = 1$$
$$a_0 \oplus a_1 \oplus a_4 \oplus a_8 \oplus a_{13} = 1$$
$$a_4 \oplus a_5 \oplus a_9 \oplus a_{11} = 0$$
$$a_0 \oplus a_2 \oplus a_5 \oplus a_{12} \oplus a_{13} \oplus a_{17} \oplus a_{19} = 1$$
$$a_1 \oplus a_2 \oplus a_4 \oplus a_5 \oplus a_6 \oplus a_8 \oplus a_{12} \oplus a_{15} = 0$$
$$a_0 \oplus a_1 \oplus a_3 \oplus a_5 \oplus a_7 \oplus a_8 \oplus a_{10} \oplus a_{11} \oplus a_{12} \oplus a_{14} \oplus a_{18} \oplus a_{21} = 1$$
$$a_2 \oplus a_3 \oplus a_4 \oplus a_9 \oplus a_{10} = 0$$
$$a_0 \oplus a_1 \oplus a_2 \oplus a_6 \oplus a_7 \oplus a_8 \oplus a_{13} \oplus a_{14} = 0$$
$$a_3 \oplus a_4 \oplus a_5 \oplus a_6 \oplus a_{10} = 1$$
$$a_0 \oplus a_1 \oplus a_3 \oplus a_7 \oplus a_8 \oplus a_9 \oplus a_{10} \oplus a_{14} = 1$$

Tabelle 3

[0050] Es kann überprüft werden, daß die resultierenden Startwertvariablen a_0, a_1, a_2, a_3 und a_{13} gleich dem Wert eins sind, während die verbleibenden Variablen den Wert null annehmen. Dieser Startwert erzeugt nachfolgend ein vollständig spezifiziertes Testmuster in der folgenden Form (die anfangs spezifizierten Stellen sind unterstrichen):

1 0 1 0 0 1 0 0
1 1 0 0 0 1 0 0
1 1 1 1 1 1 1 0
0 0 0 1 0 0 1 1
1 0 1 0 0 0 0 1
1 1 0 1 0 0 0 0
1 1 1 1 1 1 1 1
0 1 0 0 1 1 0 0

Tabelle 4

[0051] Wie beobachtet werden kann, ist das erzielte Komprimierungsverhältnis (das als die Anzahl der Abtastzellen, dividiert durch die Anzahl komprimierter Musterbits, definiert ist) $64/(2\,8 + 2{\cdot}4) \approx 2{,}66$. Die nicht spezifizierten Bits werden in Übereinstimmung mit der Logik des Dekomprimierers mit Pseudozufallswerten ausgefüllt.

[0052] Fig. 8 zeigt eine weitere Darstellung des Komprimierungsprozesses. Genauer enthält der Dekomprimierer einen LFSM **80** und einen in Verbindung stehenden linearen Phasenschieber **82**. Der LFSM wird durch eine Anzahl externer Eingangskanäle **84** gespeist, wobei der Phasenschieber ein EXKLUSIV-ODER-Netz umfaßt, das genutzt wird, um verschobene Versionen derselben Daten in ihren verschiedenen Ausgangskanälen zu vermeiden (Abtastketten **26**). Trotzdem der LFSM verhältnismäßig klein ist, kann er eine große Anzahl von Abtastketten ansteuern. Der Phasenschieber **82** liefert eine lineare Kombination der LFSM-Phasen-Ausgaben. Der LFSM erzeugt mittels Nutzung der "Verschiebe-und-Addiere"-Eigenschaft, gemäß der die Summe irgendeiner durch den LFSM erzeugten Sequenz und einer zyklischen Verschiebung von ihr selbst eine weitere zyklische Verschiebung dieser Sequenz ist, eine Sequenz mit dem gewünschten Abstand von anderen Sequenzen. Wie in Fig. 8 zu sehen ist, sind besonderen Abtastzellen vorbestimmte Werte zugeordnet, während andere Abtastzellen, wie durch ein "x" angegeben ist, "unbedeutend" sind. Nur Abtastzellen, denen vorbestimmte Werte zugeordnet sind, werden mit einem symbolischen Ausdruck gleichgesetzt.

[0053] Fig. 9 zeigt einen Dekomprimierer und Phasenschieber und wird zur Veranschaulichung der Ableitung von symbolischen Ausdrücken verwendet. Der Dekomprimierer enthält einen 8-Bit-LFSM und einen entsprechenden 4-Ausgangs-Phasenschieber. Die Architektur des LFSM ist durch Transformation eines 8-Bit-Schieberegisters mit linearer Rückkopplung, das das primitive Polynom $x^8 + x^6 + x^5 + x + 1$ realisiert, erhalten worden. Die Eingabevariablen a_1, a_2, ..., a_{14} werden in Paaren über zwei externe Eingänge geliefert, die mittels der EXKLUSIV-ODER-Gatter mit den Eingängen der Speicherelemente 1 und 5 verbunden sind. Der Phasenschieber besteht aus vier EXKLUSIV-ODER-Gattern, die mit den Ausgängen des LFSM verbunden sind. In der veranschaulichten Ausführungsform ist es erwünscht, daß die Ausgabeausdrücke erst erzeugt werden, wenn alle Speicherelemente des LFSM mit wenigstens einer Eingabevariablen gefüllt worden sind. Üblicherweise variiert die Anzahl der Zyklen, die zum Laden des LFSM erforderlich sind, je nach dem Entwurf. Zum Beispiel kann die Anzahl der Zyklen zum Laden des LFSM von dem LFSM und von der Anzahl der Eingangskanäle abhängen, wobei aber andere Kriterien genutzt werden können. Wie in Tabelle 5 veranschaulicht ist, sind in der veranschaulichten Ausführungsform nach vier aufeinanderfolgenden Taktzyklen alle Speicherelemente gefüllt. Der fortgesetzte Betrieb des Dekomprimierers liefert die im zweiten Teil von Tabelle 5 (beginnend beim 5-ten Zyklus) gesammelten Ausdrücke. Die seitdem an den Ausgängen des Phasenschiebers erzeugten Ausdrücke sind in Tabelle 6 dargestellt. Parallel zu dem oder zusammen mit dem Füllen der Abtastzellen werden neue Startwertvariablen in den LFSM injiziert.

Tabelle 5

7	6	5	4	3	2	1	0
0	0	a_1	0	0	0	a_2	0
0	0	a_3	a_1	0	0	a_4	a_2
a_2	0	a_5	$a_2 \oplus a_3$	a_1	0	a_6	a_4
a_4	a_2	a_7	$a_4 \oplus a_5$	$a_2 \oplus a_3$	a_1	a_8	a_6
a_6	$a_2 \oplus a_4$	$a_2 \oplus a_9$	$a_6 \oplus a_7$	$a_4 \oplus a_5$	$a_2 \oplus a_3$	$a_1 \oplus a_{10}$	a_8
a_8	$a_2 \oplus a_4 \oplus a_6$	$a_2 \oplus a_4 \oplus a_{11}$	$a_2 \oplus a_8 \oplus a_9$	$a_6 \oplus a_7$	$a_4 \oplus a_5$	$a_2 \oplus a_3 \oplus a_{12}$	$a_1 \oplus a_{10}$
$a_1 \oplus a_{10}$	$a_2 \oplus a_4 \oplus a_6 \oplus a_8$	$a_2 \oplus a_4 \oplus a_6 \oplus a_{13}$	$a_1 \oplus a_2 \oplus a_4 \oplus a_{11} \oplus a_{11}$	$a_2 \oplus a_8 \oplus a_9$	$a_6 \oplus a_7$	$a_4 \oplus a_5 \oplus a_{14}$	$a_2 \oplus a_3 \oplus a_{12}$

[0054] Wenn die symbolischen Ausdrücke bestimmt worden sind, wird für jeden Testkubus ein System von Gleichungen formuliert. Diese Gleichungen werden dadurch erhalten, daß die den spezifizierten Stellen des Testkubus entsprechenden symbolischen Ausdrücke ausgewählt werden (sie bilden die linken Seiten der Gleichungen) und die Werte dieser spezifizierten Stellen den jeweiligen Ausdrücken zugeordnet werden. Somit werden die rechten Seiten der Gleichungen durch die spezifizierten Stellen in den teilweise spezifizierten Testmustern definiert. Wie zu sehen ist, ist der Prozeß des Ermittelns einer geeigneten Codierung eines gegebenen Testkubus äquivalent dem Lösen eines Systems linearer Gleichungen in dem Galois-Feld modulo 2. Das Lösen der Gleichungen kann sehr effizient unter Verwendung einer Gauß-Jordan-Eliminierung ausgeführt werden, indem schnelle bitweise Operationen genutzt werden. Falls das System linearer Gleichungen eine Lösung besitzt, kann es als ein komprimiertes Testmuster behandelt werden, d. h., besitzt es die Eigenschaft, daß es dann, wenn es an den Dekomprimierer geliefert wird, das mit dem teilweise spezifizierten Anfangstestvektor konsistente dekomprimierte Testmuster liefert. Um die Wahrscheinlichkeit einer erfolgreichen Codierung zu erhöhen, ist es erwünscht, Testkuben mit der kleinsten erzielbaren Anzahl spezifizierter Stellen zu erzeugen. Folglich wird die Anzahl der Gleichungen verringert, während die gleiche Anzahl von Eingabevariablen aufrechterhalten wird, so daß die Wahrscheinlichkeit der Komprimierung erhöht wird.

Tabelle 6

3	2	1	0
$a_2 \oplus a_4 \oplus a_6$	$a_2 \oplus a_6 \oplus a_9$	$a_2 \oplus a_3 \oplus a_4 \oplus a_5 \oplus a_6 \oplus a_7$	$a_4 \oplus a_5 \oplus a_8$
$a_2 \oplus a_4 \oplus a_6 \oplus a_8$	$a_2 \oplus a_4 \oplus a_8 \oplus a_{11}$	$a_2 \oplus a_4 \oplus a_5 \oplus a_6 \oplus a_7 \oplus a_8 \oplus a_9$	$a_1 \oplus a_6 \oplus a_7 \oplus a_{10}$
$a_1 \oplus a_2 \oplus a_4 \oplus a_5 \oplus a_8 \oplus a_{10}$	$a_1 \oplus a_2 \oplus a_4 \oplus a_6 \oplus a_{10} \oplus a_{13}$	$a_1 \oplus a_4 \oplus a_6 \oplus a_7 \oplus a_8 \oplus a_9 \oplus a_{10} \oplus a_{11}$	$a_3 \oplus a_8 \oplus a_9 \oplus a_{12}$

[0055] Zur Veranschaulichung wird ein wie in Fig. 10 gezeigter beispielhafter 8-Bit-Dekomprimierer mit zwei externen Eingängen betrachtet, der vier 10-Bit-Abtastketten ansteuert. Falls dieser Schaltkreis ein Testmuster erzeugen soll, das auf dem folgenden in Tabelle 7 gezeigten teilweise spezifizierten Testkubus beruht (wobei hier die Inhalte der vier Abtastketten horizontal gezeigt sind):

x x x x x x x x x x
x x x 1 1 x 0 x x 1
x x 0 x x 1 x x 0 0
x 1 x 1 x x x x x x

Tabelle 7

wo x eine "Unbedeutend"-Bedingung bezeichnet, kann ein entsprechendes komprimiertes Muster durch Lösen des folgenden in Tabelle 8 gezeigten Systems von Gleichungen bestimmt werden (das sich aus der Struktur des Dekomprimierers aus Fig. 9 ergibt):

$$a_1 \oplus a_2 \oplus a_5 \oplus a_8 \oplus a_9 \oplus a_{11} = 0$$

$$a_5 \oplus a_6 \oplus a_7 \oplus a_9 \oplus a_{10} \oplus a_{11} \oplus a_{12} \oplus a_{13} \oplus a_{15} \oplus a_{18} \oplus a_{19} \oplus a_{20} \oplus a_{21} \oplus a_{23} \oplus a_{24} = 1$$

$$a_0 \oplus a_1 \oplus a_3 \oplus a_4 \oplus a_5 \oplus a_6 \oplus a_7 \oplus a_9 \oplus a_{12} \oplus a_{13} \oplus a_{14} \oplus a_{15} \oplus a_{17} \oplus a_{18} = 0$$

$$a_0 \oplus a_1 \oplus a_3 \oplus a_6 \oplus a_7 \oplus a_8 \oplus a_9 \oplus a_{11} \oplus a_{12} = 1$$

$$a_0 \oplus a_1 \oplus a_2 \oplus a_3 \oplus a_5 \oplus a_8 \oplus a_9 \oplus a_{10} \oplus a_{11} \oplus a_{13} \oplus a_{14} = 1$$

$$a_0 \oplus a_1 \oplus a_2 \oplus a_3 \oplus a_4 \oplus a_5 \oplus a_6 \oplus a_{11} \oplus a_{12} = 1$$

$$a_0 \oplus a_1 \oplus a_2 \oplus a_7 \oplus a_8 = 1$$

$$a_4 \oplus a_5 \oplus a_7 \oplus a_8 \oplus a_{11} \oplus a_{14} \oplus a_{15} \oplus a_{17} = 1$$

$$a_1 \oplus a_4 \oplus a_5 \oplus a_{12} \oplus a_{13} \oplus a_{15} \oplus a_{16} \oplus a_{19} \oplus a_{22} \oplus a_{23} \oplus a_{25} = 0$$

$$a_2 \oplus a_3 \oplus a_{10} \oplus a_{11} \oplus a_{13} \oplus a_{14} \oplus a_{17} \oplus a_{20} \oplus a_{21} \oplus a_{23} = 0$$

Tabelle 8

[0056] Es kann überprüft werden, daß die resultierenden Eingabevariablen a_2, a_3, a_5, a_6 und a_{12} gleich dem

Wert eins sind, während die verbleibenden Variablen den Wert null annehmen. Dieser Startwert erzeugt nachfolgend ein vollständig spezifiziertes Testmuster in der folgenden in Tabelle 9 gezeigten Form (wobei die anfangs spezifizierte Stelle jetzt unterstrichen ist):

0 1 1 1 0 1 1 0 0 0
0 1 0 1 1 0 0 0 0 1
1 0 0 1 1 1 1 1 0 0
0 1 0 1 0 1 1 1 1 0

Tabelle 9

[0057] Darüber hinaus wird die Komprimierung der Testmuster unter Verwendung der Informationen über den verfügbaren Codierungsraum unter Verwendung des ATPG-Algorithmus durchgeführt, der mehrere Fehler zum Ziel hat. In einer Form der vorliegenden Erfindung werden die Fehler in der Weise ausgewählt, daß es keine Überlappung in den zugeordneten Eingaben gibt. Falls für einen besonderen Fehler keine Zuordnungen ermittelt werden können, wird der Codierungsraum vollständig entlastet und auf einen weiteren Fehler angewendet. In der zweiten Form können sich Kegel der Erregung überlappen und Zuordnungen gemeinsam nutzen. Auf jeden Fall wird die Erzeugung eines Testkubus für einen gegebenen Fehler durch den Versuch befolgt, eine momentane Folge von Gleichungen wie oben gezeigt zu lösen.

[0058] Um die Effizienz der Komprimierung weiter zu verbessern, wird eine inkrementelle Betriebsart ermöglicht, in der mit dem bereits vorhandenen System von Gleichungen neue Gleichungen für einen nächsten zum Ziel gesetzten Fehler verkettet werden. Folglich werden die Schritte für vorhergehende Fehler, die durch den gleichen Testkubus überdeckt werden, nicht wiederholt. Tatsächlich wird mittels Nutzung von Informationen, die von den vorhergehenden Schritten verfügbar sind, weiter eine Gauß-Jordan-Eliminierung ausgeführt. Falls es immer noch eine Lösung gibt, wird die ATPG aufgerufen, die einen inkrementellen Testkubus für einen nachfolgenden Fehler erzeugen kann. Anderenfalls werden die zuletzt erzeugten Eingabezuordnungen aufgehoben und wird ein weiterer Fehler zum Ziel gesetzt, bis ein bestimmter Grenzwert erfolgloser Versuche erreicht worden ist. Im letzteren Fall wird mit der entsprechenden Folge linearer Gleichungen ein neuer Testkubus erzeugt.

[0059] Fig. 11 ist ein Ablaufplan eines Verfahrens, das die inkrementelle Betriebsart der Komprimierung veranschaulicht. Im Prozeßblock **90** wird ein Fehler ausgewählt und ein Testkubus zum Testen des ausgewählten Fehlers erzeugt. Wie zuvor beschrieben wurde, wird die Testkubuserzeugung unter Verwendung eines deterministischen Testmusters ausgeführt. Im Prozeßblock **92** wird der Testkubus verwendet, um, wie zuvor beschrieben wurde, eine Folge von Gleichungen zu erzeugen. Im Block **94** wird eine Bestimmung vorgenommen, ob die Gleichungen lösbar sind. Falls die Gleichungen lösbar sind, wird die Folge von Gleichungen im Prozeßblock **96** inkrementell mit einer oder mit mehreren neuen Gleichungen verkettet, die einen oder mehrere weitere Fehler testen. Wie durch den Pfeil **98** gezeigt ist, wird daraufhin eine Bestimmung vorgenommen, ob die Folge verketteter Gleichungen lösbar ist. Dieser Prozeß wird fortgesetzt, bis der Versuch zum Lösen der Folge von Gleichungen fehlschlägt. Bei einem fehlgeschlagenen Versuch zum Lösen der Gleichungen wird der Prozeßblock **94** negativ beantwortet, was veranlaßt, daß die Steuerung zum Prozeßblock **100** übergeht. Im Ergebnis wird/werden im Prozeßblock **100** die zuletzt verkettete Gleichung oder die zuletzt verketteten Gleichungen gelöscht. Im Prozeßblock **102** wird eine Bestimmung vorgenommen, ob ein Grenzkriterium erreicht worden ist wie etwa, daß die Anzahl der Fehler gleich einem vorbestimmten Schwellenwert ist oder ihn übersteigt oder eine Anzahl spezifizierter Bits in einem Testkubus erreicht worden ist. Zum Beispiel kann die inkrementelle Verkettung abgebrochen werden, falls die Anzahl der Bits in dem Testkubus, denen Werte zugeordnet sind, 90% oder 95% der Anzahl der Eingabevariablen erreicht. Es kann irgendein gewünschtes Grenzkriterium genutzt werden. Falls das Grenzkriterium erreicht ist, wird das komprimierte Endmuster erzeugt (Prozeßblock **104**). Falls das nicht der Fall ist, wird im Prozeßblock **96** erneut versucht, die Gleichungen inkrementell zu verketten, um noch einen weiteren Fehler zu testen.

[0060] Nachdem die Prinzipien der Erfindung in beispielhaften Ausführungen veranschaulicht und beschrieben worden sind, sollte für den Fachmann auf dem Gebiet klar sein, daß die veranschaulichten Ausführungsformen in Bezug auf Gestaltung und die Einzelheiten geändert werden können, ohne von diesen Prinzipien abzuweichen. Obgleich der Dekomprimierer als in den integrierter Schaltkreis integriert gezeigt ist, kann er z. B. ebenfalls in einem ATE sein. In diesem Fall werden die Eingabevariablen dem Dekomprimierer zusammen extern zugeführt, während der Dekomprimierer betrieben wird, um die Ausgabeausdrücke des Dekomprimierers zu erzeugen. Obgleich die Erzeugung von symbolischen Ausdrücken unter Verwendung symbolischer Simulation eines Dekomprimierers gezeigt ist, kann diese Erzeugung außerdem durch andere Techniken als Simulation wie etwa durch mathematische Darstellung des Dekomprimierers (z. B. ein primitives Polynom) und

Erzeugen der Ausdrücke unter Verwendung mathematischer Analysis ausgeführt werden. Angesichts der vielen möglichen Ausführungsformen, auf die die Prinzipien der Erfindung angewendet werden können, soll die veranschaulichende Ausführungsform selbstverständlich diese Prinzipien lehren und keine Beschränkung des Umfangs der wie durch die beigefügten Ansprüche definierten Erfindung sein.

Patentansprüche

1. Verfahren zum Berechnen eines komprimierten Testmusters zum Laden von Abtastzellen innerhalb eines integrierten Schaltkreises (**34**) zum Testen von Fehlern, wobei das Verfahren die folgenden Schritte umfaßt:
– Vorsehen eines Dekomprimierers (**36**) mit einem oder mehreren Eingangsinjektoren (**48a**, **48b**) und einem oder mehreren logischen Gattern;
– Zuordnen von Eingabevariablen zu Bits, die auf den einen oder die mehreren Eingangsinjektoren (**48a**, **48b**) seriell gegeben werden; und
– Erzeugen eines komprimierten Testmusters mittels Nutzung von symbolischen Ausdrücken, die mit den Abtastzellen in Verbindung stehen, wobei die symbolischen Ausdrücke lineare Kombinationen der Eingabevariablen umfassen, die zusammen auf den Dekomprimierer (**36**) gegeben werden, wenn der Dekomprimierer die Eingabevariablen dekomprimiert, und wobei die linearen Kombinationen zumindest teilweise von den logischen Gattern des Dekomprimierers (**36**) bestimmt werden.

2. Verfahren nach Anspruch 1, dadurch gekennzeichnet, daß der Dekomprimierer (**36**) einen linearen endlichen Automaten (**46**) umfaßt, der an einen Phasenschieber (**50**) gekoppelt ist.

3. Verfahren nach Anspruch 1 oder 2, dadurch gekennzeichnet, daß die Eingangsinjektoren (**48a**, **48b**) EXKLUSIV-ODER-Gatter umfassen.

4. Verfahren nach einem der Ansprüche 1 bis 3, dadurch gekennzeichnet, daß der Dekomprimierer (**36**) als Reaktion auf ein Taktsignal einen Zustand ändert und während eines oder mehrerer Taktzyklen eine neue Folge von Eingabevariablen an den Eingangsinjektoren (**48a**, **48b**) des Dekomprimierers empfangen wird.

5. Verfahren nach einem der Ansprüche 1 bis 4, dadurch gekennzeichnet, daß der Dekomprimierer Mehrfach-Speicherelemente umfaßt, die zusammen mit den logischen Gattern verbunden sind.

6. Verfahren nach einem der Ansprüche 1 bis 5, gekennzeichnet durch die folgenden Schritte:
Anordnen des Dekomprimierers (**36**) innerhalb eines integrierten Schaltkreises (**34**);
Vorsehen von Abtastketten (**26**) innerhalb des integrierten Schaltkreises (**34**), wobei jede Abtastkette in Serie gekoppelte Mehrfach-Abtastketten umfaßt und wobei die Abtastketten (**26**) an den Dekomprimierer (**36**) gekoppelt sind; und
Erzeugen eines Testkubus, welcher vorbestimmte Werte einigen der Abtastzellen zuordnet, um Fehler innerhalb des integrierten Schaltkreises zu testen.

7. Verfahren nach einem der Ansprüche 1 bis 5, dadurch gekennzeichnet, daß das Erzeugen eines komprimierten Testmusters die folgenden Schritte umfaßt:
Erzeugen eines Testkubus mit einem Anteil von Abtastzellen, die Werten (**62**) zugeordnet sind;
Formulieren einer Folge von Gleichungen mittels Gleichsetzen der zugeordneten Werte in den Abtastzellen und der symbolischen Ausdrücke (**64**); und
Lösen der Gleichungen, um das komprimierte Testmuster (**66**) zu erhalten.

8. Verfahren nach Anspruch 7, dadurch gekennzeichnet, daß die Folge von Gleichungen mit einer oder mehreren Gleichungen (**96**) inkrementell verkettet wird.

9. Verfahren nach Anspruch 8, dadurch gekennzeichnet, daß das inkrementelle Verketten abgebrochen wird, wenn die Gleichungen nicht gelöst werden können und ein Grenzkriterium erreicht oder überstiegen wird (**102**).

10. Verfahren nach Anspruch 8, gekennzeichnet durch die folgenden Schritte: Versuchen des Lösens der Gleichungen und Löschen der inkrementell verketteten Gleichungen (**100**) und inkrementelles Verketten einer oder mehrerer neuer Gleichungen (**96**), wenn die Gleichungen nicht gelöst werden können.

11. Verfahren nach einem der Ansprüche 1 bis 10, dadurch gekennzeichnet, daß nur Bits in einem ATE

(**21**) gespeichert werden, die notwendig sind, um die vorbestimmten Werte in den Abtastzellen zu erzeugen, wohingegen die verbleibenden Bits in den Abtastzellen mittels Mustern gefüllt werden, die in dem Dekomprimierer (**36**) erzeugt werden.

12. Verfahren nach einem der Ansprüche 1 bis 11, dadurch gekennzeichnet, daß die Abtastzellen zusammen geladen werden, wenn die Eingangsvariablen auf den Dekomprimierer (**36**) gegeben werden.

13. Verfahren nach einem der Ansprüche 1 bis 12, dadurch gekennzeichnet, daß der Dekomprimierer (**36**) auf einem ATE (**21**) angeordnet ist.

14. Computerprogramm mit Programmcodemitteln zum Ausführen aller Schritte eines der Ansprüche 1 bis 13, wenn das Programm auf einem Computer läuft.

15. Computerprogramm nach Anspruch 14, gekennzeichnet durch eine Implementierung auf einem computerlesbaren Medium.

Es folgen 9 Blatt Zeichnungen

Anhängende Zeichnungen

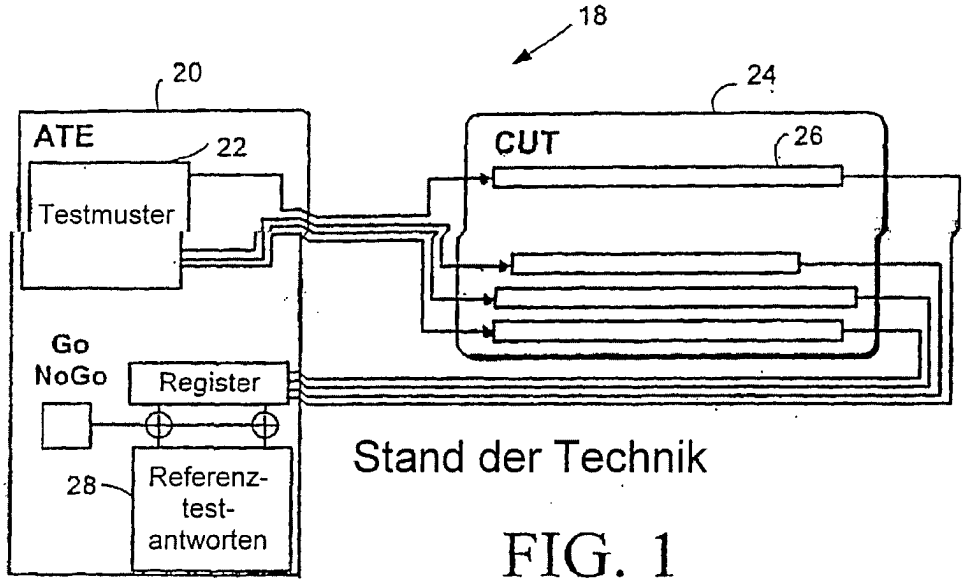


FIG. 1

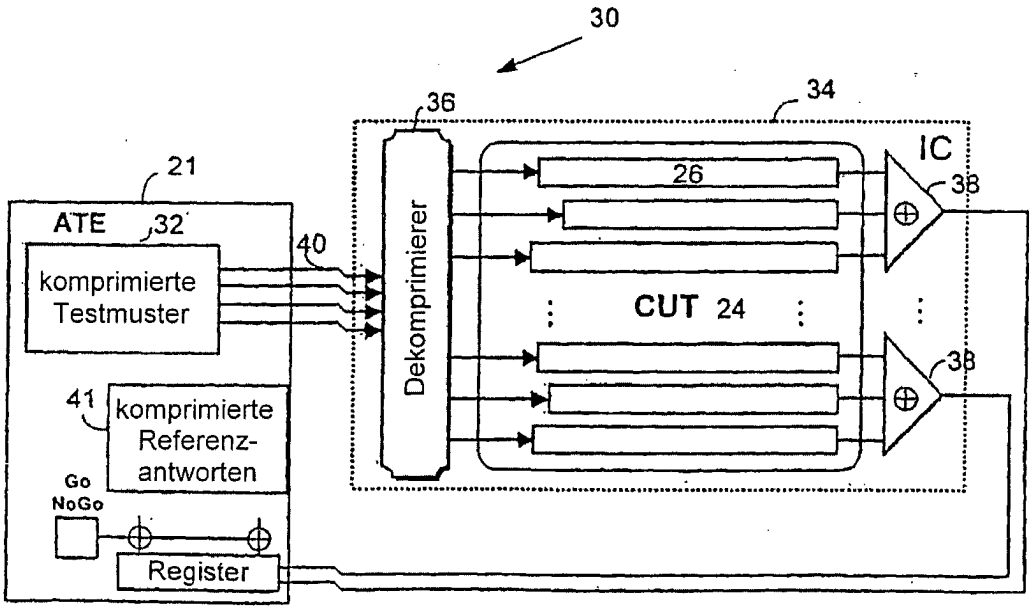


FIG. 2

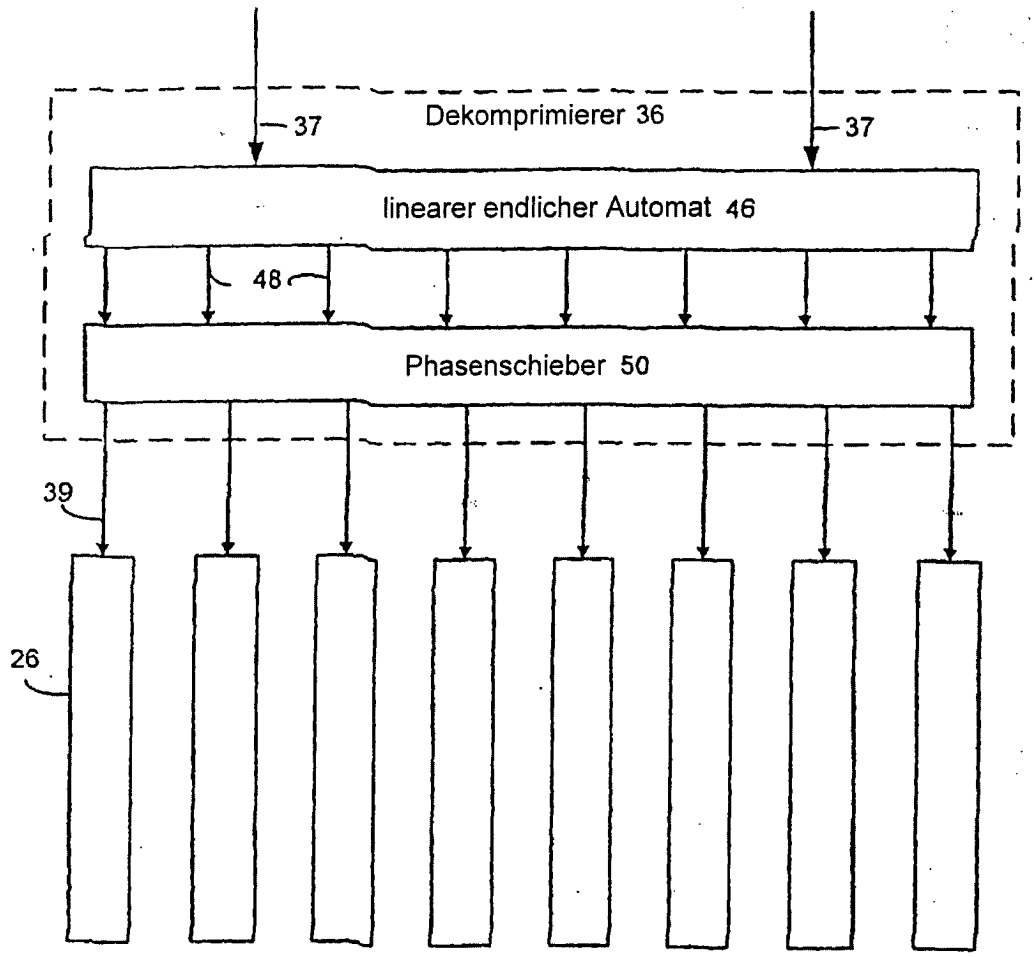

37
Dekomprimierer 36
37
linearer endlicher Automat 46
48
Phasenschieber 50
39
26

FIG. 3

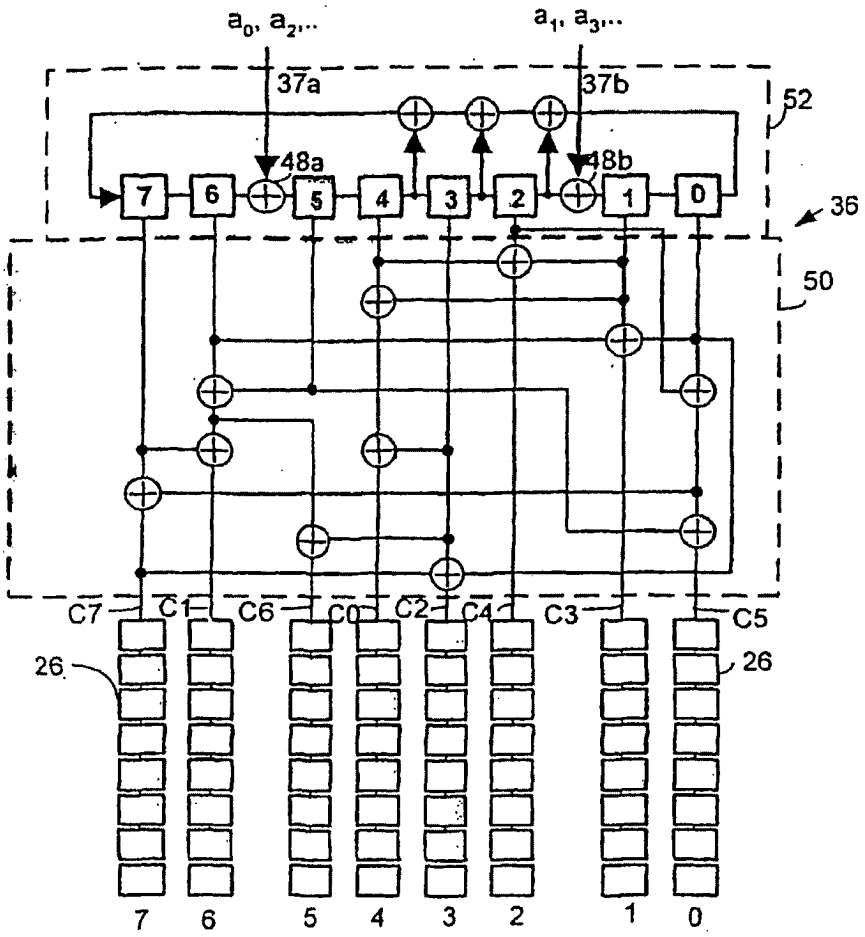

a0, a2,..
a1, a3,..
37a
37b
52
48a
48b
7
6
5
4
3
2
1
0
36
50
C7
C1
C6
C0
C2
C4
C3
C5
26
26
7
6
5
4
3
2
1
0

FIG. 4

FIG. 5

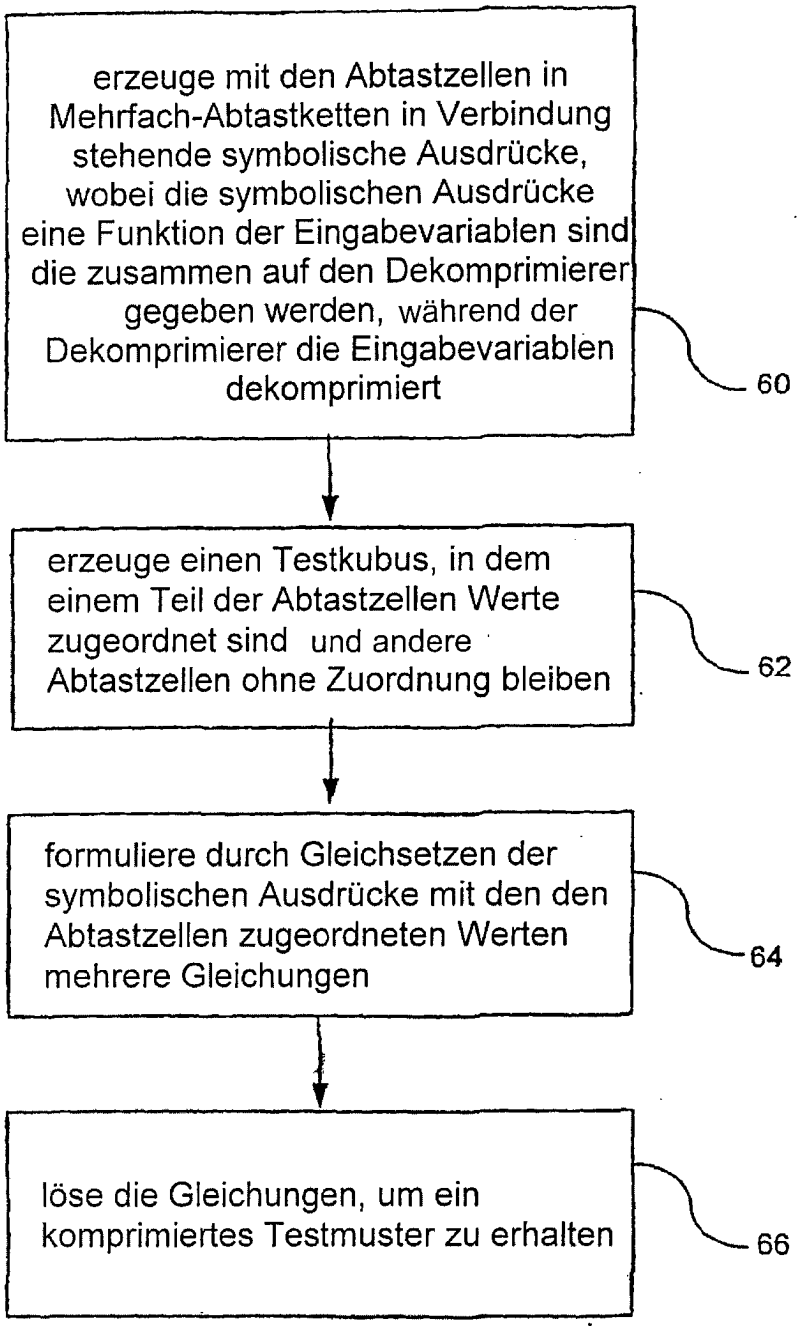
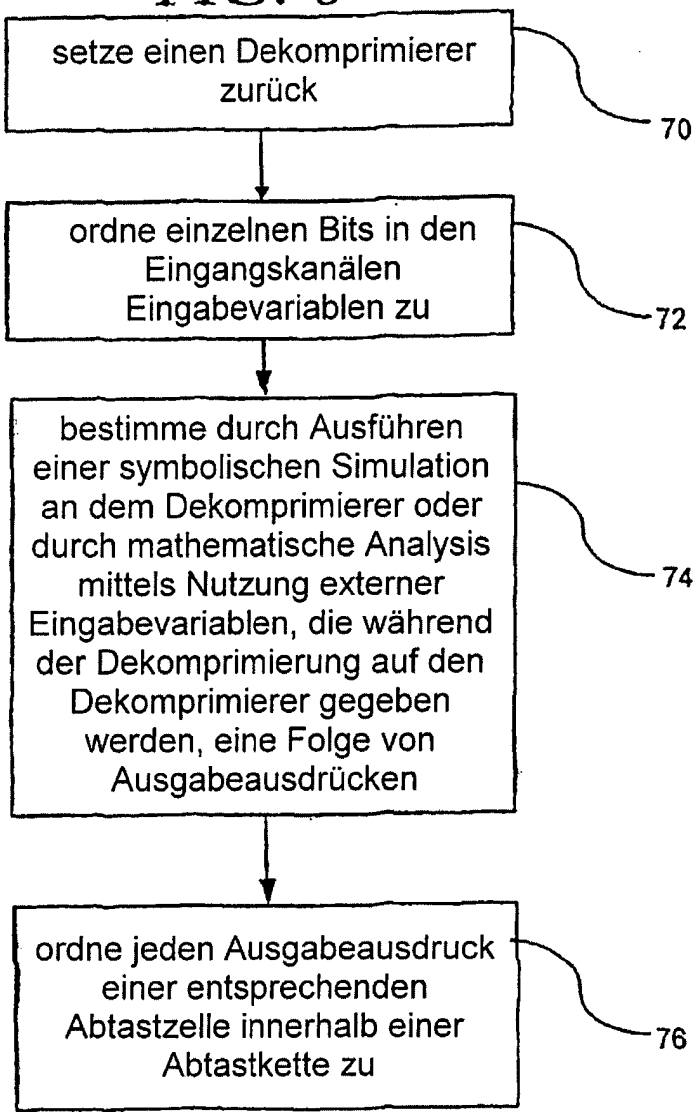

erzeuge mit den Abtastzellen in Mehrfach-Abtastketten in Verbindung stehende symbolische Ausdrücke, wobei die symbolischen Ausdrücke eine Funktion der Eingabevariablen sind die zusammen auf den Dekomprimierer gegeben werden, während der Dekomprimierer die Eingabevariablen dekomprimiert
60
erzeuge einen Testkubus, in dem einem Teil der Abtastzellen Werte zugeordnet sind und andere Abtastzellen ohne Zuordnung bleiben
62
formuliere durch Gleichsetzen der symbolischen Ausdrücke mit den den Abtastzellen zugeordneten Werten mehrere Gleichungen
64
löse die Gleichungen, um ein komprimiertes Testmuster zu erhalten
66

FIG. 6


setze einen Dekomprimierer zurück
70
ordne einzelnen Bits in den Eingangskanälen Eingabevariablen zu
72
bestimme durch Ausführen einer symbolischen Simulation an dem Dekomprimierer oder durch mathematische Analysis mittels Nutzung externer Eingabevariablen, die während der Dekomprimierung auf den Dekomprimierer gegeben werden, eine Folge von Ausgabeausdrücken
74
ordne jeden Ausgabeausdruck einer entsprechenden Abtastzelle innerhalb einer Abtastkette zu
76

Abtastkette 0

$a_2 \oplus a_4 \oplus a_5$

$a_4 \oplus a_6 \oplus a_7$

$a_1 \oplus a_6 \oplus a_8 \oplus a_9$

$a_1 \oplus a_3 \oplus a_8 \oplus a_{10} \oplus a_{11}$

$a_3 \oplus a_5 \oplus a_{10} \oplus a_{12} \oplus a_{13}$

$a_0 \oplus a_1 \oplus a_4 \oplus a_5 \oplus a_8 \oplus a_9 \oplus a_{11} \oplus a_{12} \oplus a_{13} \oplus a_{16}$

$a_0 \oplus a_2 \oplus a_7 \oplus a_9 \oplus a_{14} \oplus a_{16} \oplus a_{17}$

$a_0 \oplus a_2 \oplus a_4 \oplus a_9 \oplus a_{11} \oplus a_{16} \oplus a_{18} \oplus a_{19}$

Abtastkette 1

$a_1 \oplus a_2 \oplus a_3 \oplus a_6$

$a_0 \oplus a_1 \oplus a_3 \oplus a_4 \oplus a_5 \oplus a_8$

$a_2 \oplus a_3 \oplus a_5 \oplus a_6 \oplus a_7 \oplus a_{10}$

$a_0 \oplus a_1 \oplus a_4 \oplus a_5 \oplus a_7 \oplus a_8 \oplus a_9 \oplus a_{12}$

$a_2 \oplus a_3 \oplus a_6 \oplus a_7 \oplus a_9 \oplus a_{10} \oplus a_{11} \oplus a_{14}$

$a_0 \oplus a_5 \oplus a_7 \oplus a_{12} \oplus a_{14} \oplus a_{15}$

$a_0 \oplus a_2 \oplus a_3 \oplus a_6 \oplus a_7 \oplus a_{10} \oplus a_{11} \oplus a_{13} \oplus a_{14} \oplus a_{15} \oplus a_{18}$

$a_0 \oplus a_1 \oplus a_2 \oplus a_4 \oplus a_5 \oplus a_8 \oplus a_9 \oplus a_{13} \oplus a_{13} \oplus a_{15} \oplus a_{16} \oplus a_{17} \oplus a_{20}$

Abtastkette 2

a_3

$a_0 \oplus a_5$

$a_2 \oplus a_7$

$a_0 \oplus a_4 \oplus a_9$

$a_2 \oplus a_6 \oplus a_{11}$

$a_0 \oplus a_1 \oplus a_4 \oplus a_8 \oplus a_{13}$

$a_0 \oplus a_2 \oplus a_3 \oplus a_6 \oplus a_{10} \oplus a_{15}$

$a_1 \oplus a_2 \oplus a_4 \oplus a_5 \oplus a_8 \oplus a_{12} \oplus a_{17}$

Abtastkette 3

$a_0 \oplus a_1 \oplus a_5 \oplus a_7$

$a_2 \oplus a_3 \oplus a_7 \oplus a_9$

$a_4 \oplus a_5 \oplus a_9 \oplus a_{11}$

$a_6 \oplus a_7 \oplus a_{11} \oplus a_{13}$

$a_1 \oplus a_8 \oplus a_9 \oplus a_{13} \oplus a_{15}$

$a_0 \oplus a_3 \oplus a_{10} \oplus a_{11} \oplus a_{15} \oplus a_{17}$

$a_0 \oplus a_2 \oplus a_5 \oplus a_{12} \oplus a_{13} \oplus a_{17} \oplus a_{19}$

$a_2 \oplus a_4 \oplus a_7 \oplus a_{14} \oplus a_{15} \oplus a_{19} \oplus a_{21}$

Abtastkette 4

$a_0 \oplus a_4 \oplus a_7$

$a_0 \oplus a_2 \oplus a_6 \oplus a_9$

$a_0 \oplus a_1 \oplus a_7 \oplus a_4 \oplus a_8 \oplus a_{11}$

$a_0 \oplus a_2 \oplus a_3 \oplus a_4 \oplus a_6 \oplus a_{10} \oplus a_{13}$

$a_1 \oplus a_2 \oplus a_4 \oplus a_5 \oplus a_6 \oplus a_8 \oplus a_{12} \oplus a_{15}$

$a_1 \oplus a_3 \oplus a_4 \oplus a_6 \oplus a_7 \oplus a_8 \oplus a_{10} \oplus a_{14} \oplus a_{17}$

$a_1 \oplus a_3 \oplus a_5 \oplus a_6 \oplus a_8 \oplus a_9 \oplus a_{10} \oplus a_{12} \oplus a_{16} \oplus a_{19}$

$a_0 \oplus a_1 \oplus a_3 \oplus a_5 \oplus a_7 \oplus a_8 \oplus a_{10} \oplus a_{11} \oplus a_{12} \oplus a_{14} \oplus a_{18} \oplus a_{21}$

Abtastkette 5

$a_0 \oplus a_5 \oplus a_6$

$a_0 \oplus a_1 \oplus a_2 \oplus a_7 \oplus a_8$

$a_2 \oplus a_3 \oplus a_4 \oplus a_9 \oplus a_{10}$

$a_0 \oplus a_4 \oplus a_5 \oplus a_6 \oplus a_{11} \oplus a_{12}$

$a_0 \oplus a_1 \oplus a_2 \oplus a_6 \oplus a_7 \oplus a_8 \oplus a_{13} \oplus a_{16}$

$a_0 \oplus a_1 \oplus a_2 \oplus a_3 \oplus a_4 \oplus a_8 \oplus a_9 \oplus a_{10} \oplus a_{15} \oplus a_{16}$

$a_1 \oplus a_3 \oplus a_4 \oplus a_5 \oplus a_6 \oplus a_{10} \oplus a_{11} \oplus a_{12} \oplus a_{17} \oplus a_{18}$

$a_1 \oplus a_4 \oplus a_5 \oplus a_6 \oplus a_7 \oplus a_8 \oplus a_{12} \oplus a_{13} \oplus a_{14} \oplus a_{19} \oplus a_{20}$

Abtastkette 6

$a_0 \oplus a_1 \oplus a_2 \oplus a_6$

$a_1 \oplus a_2 \oplus a_3 \oplus a_4 \oplus a_8$

$a_3 \oplus a_4 \oplus a_5 \oplus a_6 \oplus a_{10}$

$a_1 \oplus a_5 \oplus a_6 \oplus a_7 \oplus a_8 \oplus a_{12}$

$a_0 \oplus a_1 \oplus a_3 \oplus a_7 \oplus a_8 \oplus a_9 \oplus a_{10} \oplus a_{14}$

$a_0 \oplus a_2 \oplus a_3 \oplus a_5 \oplus a_9 \oplus a_{10} \oplus a_{11} \oplus a_{12} \oplus a_{16}$

$a_0 \oplus a_2 \oplus a_4 \oplus a_5 \oplus a_7 \oplus a_{11} \oplus a_{12} \oplus a_{15} \oplus a_{14} \oplus a_{16}$

$a_0 \oplus a_2 \oplus a_4 \oplus a_6 \oplus a_7 \oplus a_9 \oplus a_{13} \oplus a_{14} \oplus a_{15} \oplus a_{16} \oplus a_{20} \oplus a_{19}$

Abtastkette 7

$a_2 \oplus a_3 \oplus a_5$

$a_0 \oplus a_4 \oplus a_5 \oplus a_7$

$a_0 \oplus a_1 \oplus a_6 \oplus a_7 \oplus a_9$

$a_1 \oplus a_2 \oplus a_4 \oplus a_8 \oplus a_9 \oplus a_{11}$

$a_0 \oplus a_3 \oplus a_4 \oplus a_6 \oplus a_{10} \oplus a_{11} \oplus a_{13}$

$a_1 \oplus a_2 \oplus a_5 \oplus a_6 \oplus a_8 \oplus a_{12} \oplus a_{13} \oplus a_{15}$

$a_1 \oplus a_3 \oplus a_4 \oplus a_7 \oplus a_8 \oplus a_{10} \oplus a_{14} \oplus a_{15} \oplus a_{17}$

$a_0 \oplus a_3 \oplus a_5 \oplus a_6 \oplus a_9 \oplus a_{10} \oplus a_{12} \oplus a_{16} \oplus a_{17}$

FIG. 7

FIG. 8

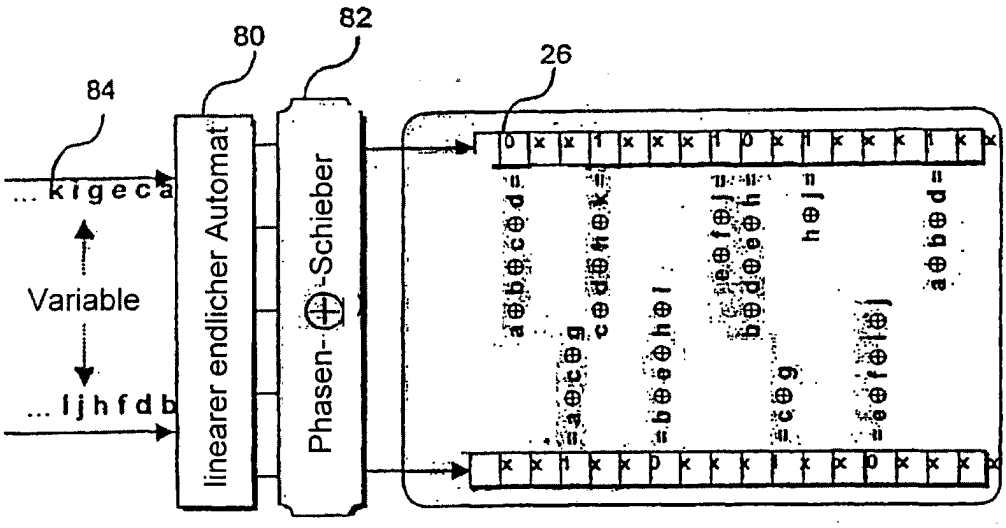

80
82
26
84
... k i g e c a
Variable
... l j h f d b
linearer endlicher Automat
Phasen-⊕-Schieber

FIG. 9

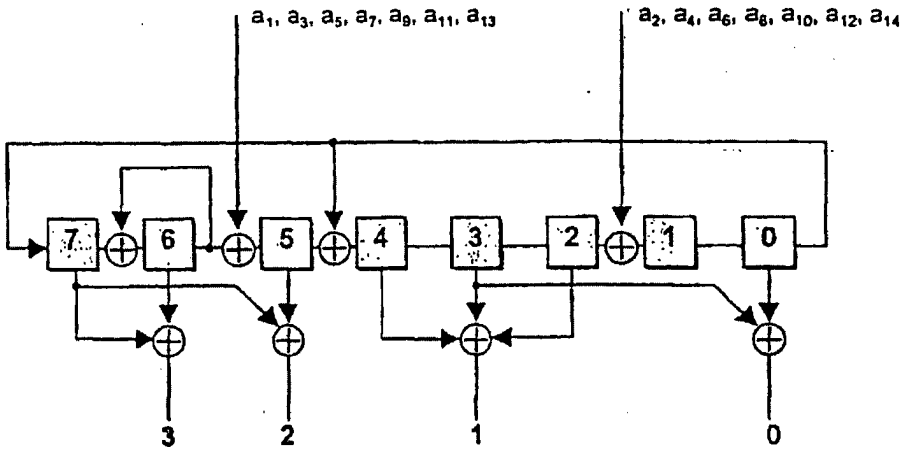

a1, a3, a5, a7, a9, a11, a13
a2, a4, a6, a8, a10, a12, a14
7
6
5
4
3
2
1
0
3
2
1
0

FIG. 10

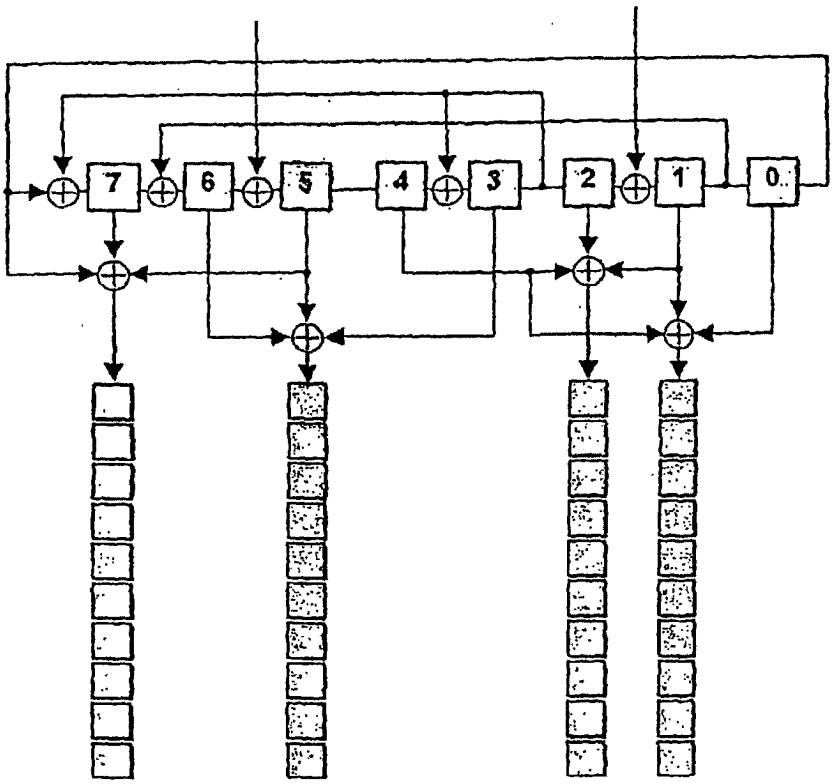
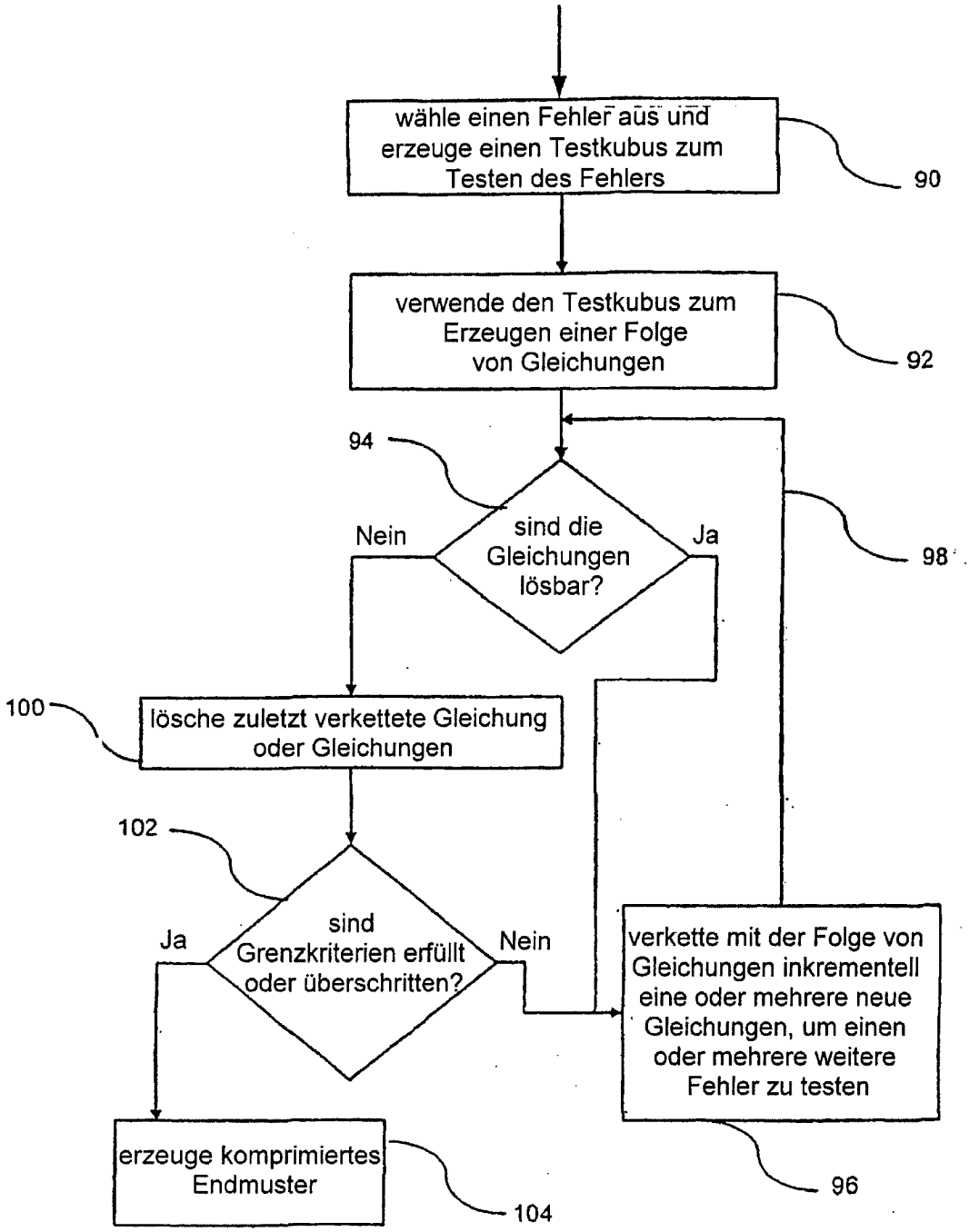


FIG. 11


wähle einen Fehler aus und erzeuge einen Testkubus zum Testen des Fehlers
90
verwende den Testkubus zum Erzeugen einer Folge von Gleichungen
92
94
sind die Gleichungen lösbar?
Nein
Ja
98
100
lösche zuletzt verkettete Gleichung oder Gleichungen
102
sind Grenzkriterien erfüllt oder überschritten?
Ja
Nein
verkette mit der Folge von Gleichungen inkrementell eine oder mehrere neue Gleichungen, um einen oder mehrere weitere Fehler zu testen
96
erzeuge komprimiertes Endmuster
104